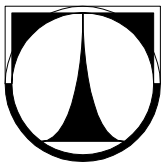
TECHNICKÁ UNIVERZITA V LIBERCI
Fakulta mechatroniky, informatiky a mezioborových studií



BAKALÁŘSKÁ PRÁCE

Liberec 2009 Jaroslav Ševic

TECHNICKÁ UNIVERZITA V LIBERCI
Fakulta mechatroniky, informatiky a mezioborových studií

Studijní program: B2612 – Elektrotechnika a informatika
Studijní obor: Elektronické informační a řídící systémy

Golub-Kahanova bidiagonalizace, její implementace a numerické experimenty

Golub-Kahan bidiagonalization, its implementation, and numerical experiments

Bakalářská práce

Autor: **Jaroslav Ševic**
Vedoucí práce: Ing. Martin Plešinger, Ph.D.
Konzultant: Ing. Pavel Jiránek, Ph.D.

V Liberci 27. května 2009

[zadání]

Prohlášení

Byl jsem seznámen s tím, že na mou bakalářskou práci se plně vztahuje zákon č. 121/2000 o právu autorském, zejména § 60 (školní dílo).

Beru na vědomí, že TUL má právo na uzavření licenční smlouvy o užití mé bakalářské práce a prohlašuji, že **s o u h l a s í m** s případným užitím mé bakalářské práce (prodej, zapůjčení apod.).

Jsem si vědom toho, že užít své bakalářské práce či poskytnout licenci k jejímu využití mohu jen se souhlasem TUL, která má právo ode mne požadovat přiměřený příspěvek na úhradu nákladů, vynaložených univerzitou na vytvoření díla (až do jejich skutečné výše).

Bakalářskou práci jsem vypracoval samostatně s použitím uvedené literatury a na základě konzultací s vedoucím bakalářské práce a konzultantem.

Datum: 29. května 2009

Podpis:

Poděkování

Chtěl bych poděkovat vedoucímu bakalářské práce Ing. Martinu Plešingerovi, Ph.D. za obětavou spolupráci i za čas, který mi při konzultacích věnoval a za podporu při řešení problémů.

Anotace

Tato bakalářská práce se zabývá implementací algoritmů pro transformaci matice na bidiagonální tvar. Tuto transformaci lze provést několika způsoby, například užitím Householderových transformací nebo užitím Golub-Kahanova iteračního bidiagonalizačního algoritmu. Protože se při těchto výpočtech nevyhneme použití aritmetiky s konečnou přesností, je důležitým aspektem přesnost spočtených matic. Jedním z faktorů, který můžeme sledovat je ortogonalita spočtených vektorů, které produkuje Golub-Kahanova bidiagonalizace. Je známým faktem, že pro zachování ortogonality je zapotřebí provádět reortogonalizaci. Úplná reortogonalizace je však velmi náročná na strojový čas počítače, proto je nutné při praktických výpočtech provádět pouze reortogonalizaci částečnou. V této práci se stručně zabýváme různými strategiemi neúplné reortogonalizace a porovnáváme získané výsledky jak z hlediska jejich přesnosti, tak s ohledem na náročnost výpočtu.

Klíčová slova

Golub-Kahanova bidiagonalizace, ortogonální transformace, Gram-Schmidtův proces, reortogonalizace.

OBSAH

Úvod

Využívání výpočetní techniky při řešení reálných úloh vede přirozeně k formulacím těchto úloh jazykem matematiky a zejména jazykem lineární algebry. Úlohy formulované jako problémy lineární algebry jsou jedny z mála, které dokážeme alespoň v principu řešit pomocí výpočetní techniky (přesněji řečeno některé z nich). Zřejmý fakt, že prostředky počítačů jsou a vždy budou omezené (omezená velikost paměti, konečná nenulová doba provádění elementárních aritmetických a logických operací), klade jisté nároky na algoritmy, které k řešení takových úloh použijeme. Omezenost paměťových prostředků počítače je příčinou toho, že musíme počítat jen s omezenou (konečnou) přesností (např. iracionální čísla nelze zobrazit v počítači vůbec), čímž se dopouštíme chyb zaokrouhlováním. Důležitým aspektem používaných algoritmů je jejich stabilita, v jistém smyslu odolnost vůči šíření zaokrouhlovacích chyb. Analýza šíření zaokrouhlovacích chyb patří k jedněm z předních témat studia numerické lineární algebry. Na druhou stranu konečná nenulová doba provádění elementárních operací nás vede k otázkám týkajícím se rychlosti provádění algoritmů. Ta je přímo závislá zejména na množství aritmetických operací (množství logických operací bývá zpravidla zanedbatelné, zejména pro velmi rozsáhlé úlohy) a samozřejmě na uspořádání dat v paměti počítače a práci algoritmu s pamětí. *Stabilita* algoritmu a *výpočetní cena* jsou požadavky, které často jdou proti sobě. Poznamenejme, že zde i v celém následujícím textu používáme pro náročnost výpočtu termín výpočetní cena, nikoliv termín komplexita (složitost) jak je zvykem například u třídících nebo vyhledávacích algoritmů. Termín komplexita necháváme rezervovaný pravě pro význam kombinatorické složitosti. U algoritmů numerické lineární algebry, zejména u algoritmů iteračních, kde si v podstatě vybíráme (na základě různých kritérií) kdy výpočet zastavíme, není vhodný.

Jednou z úloh lineární algebry je tak zvaný (úplný) problém nejmenších čtverců. Jedná se o lineární aproximační problém (v podstatě soustavu n lineárních rovností pro m neznámých, která nemá řešení v klasickém slova smyslu)

$$Ax \approx b, \qquad A \in R^{nxm}, \qquad x \in R^m, \qquad b \in R^n.$$

V případě úlohy nejmenších čtverců řešení aproximačního problému hledáme tak, že minimálním způsobem poopravíme pravou stranu b takovým způsobem, aby opravený systém měl řešení v klasickém slova smyslu. Je-li toto řešení nejednoznačné vybíráme nejmenší z nich (minimálním, resp. nejmenším zde máme na mysli vektor s nejmenší standardní Eukleidovskou normou). V případě úlohy úplných nejmenších čtverců se snažíme minimálním způsobem poopravit pravou stranu b i matici A tak, aby opravený systém

měl řešení v klasickém slova smyslu (minimální opravou matice je myšlen nejmenší součet kvadrátů jednotlivých prvků opravy, tzv. Frobeniova norma matice opravy). S řešitelností úplného problému nejmenších čtverců je to podstatně složitější. Obě úlohy, jak obyčejný tak úplný problém nejmenších čtverců, jsou ortogonálně invariantní, jinými slovy nezávislé na bázích zvolených v prostorech R^n a R^m.

Nedávno bylo ukázáno, že zcela obecné ortogonálně invariantní lineární aproximační problémy tvaru $Ax \approx b$ lze redukovat na tzv. *core problém* [12, 14]. Ten je ve většině případů snáze řešitelný než problém původní, přičemž řešení původní úlohy a core problému jsou (až na ortogonální transormaci) identická. Přechod ke core problému tak může usnadnit výpočet řešení původního aproximačního problému, navíc teorie core problému přináší významný a nový vhled do teorie lineárních aproximačních problémů jako takových. Přechod ke core problému je realizován transformací rozšířené matice soustavy

$[b|A]$

na horní bidiagonální tvar. Tato transformace lze v podstatě provést dvěma způsoby, jednak přímou bidiagonalizací rozšířené matice (převodem matice $[b|A]$ na horní bidiagonální tvar) pomocí ortogonálních transformací, nebo pomocí iteračního algoritmu, tzv. Golub-Kahanovy bidiagonalizace [6] matice A (tentokrát se jedná o transformaci na dolní bidiagonální tvar) nastartované startovacím vektorem $s = b/\|b\|$, tedy normalizovanou pravou stranou.

V této práci se zabýváme pouze implementací vybraných algoritmů realizujících bidiagonalizaci v prostředí MATLABu a studiem některých numerických aspektů těchto algoritmů (ztrátou ortogonality při výpočtu ortogonálních transformací, přesností spočtené bidiagonální matice, okrajově též dobou výpočtu). Text je organizován následujícím způsobem. V první kapitole stručně popisujeme základní ortogonální transformace (Givensovy rotace a Householderovy reflexe), jejich konstrukci, použití při výpočtu a vzájemný vztah obou transformací. V druhé kapitole se věnujeme výkladu tak zvaného QR rozkladu matice, který je vhodným nástrojem dobře využitelným při popisu některých partií bidiagonalizačních algoritmů (reortogonalizace). Ve třetí kapitole se věnujeme popisu jednotlivých algoritmů realizujících bidiagonalizaci a ve čtvrté kapitole pak ilustrujeme numerické vlastnosti a chování vybraných algoritmů. V páté kapitole jen velmi stručně popisujeme software, který byl pro účely testování numerických vlastností vyvinut. V závěru shrneme výsledky jichž bylo v práci dosaženo.

1. Ortogonální transformace

Ortogonální transformace jsou transformace realizované ortogonálními maticemi (v komplexním případě se jedná o unitární matice). Ortogonální a unitární matice jsou normální matice (tj. $AA^T = A^T A$), jejichž všechna vlastní čísla (tj. čísla $\lambda \in C$, pro která existuje nenulový vektor $x \in C^n$ tak, že $Ax = x\lambda$) leží na jednotkové kružnici v komplexní rovině. Matice $Q \in C^{nxn}$ je unitární, jestliže

$$Q\overline{Q}^T = I_n = \overline{Q}^T Q,$$

kde I_n je jednotková matice dimenze *n*, Q^T značí matici transponovanou (transponováním rozumíme výměnu řádků za sloupce) k matici Q, $\overline{Q}$ značí matici komplexně sdruženou k matic Q. Matice Q je ortogonální, jestliže je unitární a reálná, $Q \in R^{nxn}$, tj.

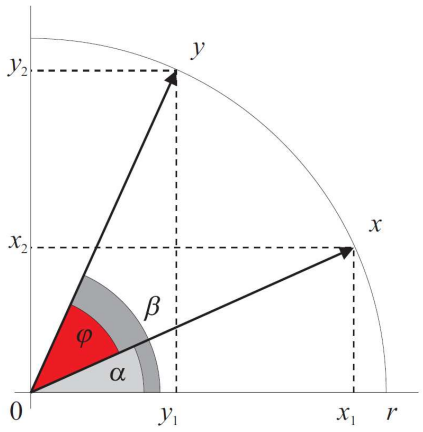
$$QQ^T = I_n = Q^T Q.$$

V následujícím textu se seznámíme s dvěma základními typy ortogonálních transformací v R^n

- rotací o úhel φ v rovině určené dvěma vektory (Givensovou rotací),
- zrcadlením podle nadroviny dané normálovým vektorem (Householderovou reflexí).

1.1 Givensova rotace

Začneme popisem elementární rotace v rovině, v R^2, pro kterou platí $y = G(\varphi)x$. Matice $G(\varphi)$ realizuje pootočení libovolného vektoru x o úhel φ proti směru hodinových ručiček. Tvar matice $G(\varphi)$ lze odvodit například pomocí obrázku 1.1. Vektor x chceme pootočit o úhel φ tak, abychom dostali vektor y. Zřejmě musí být splněna podmínka rovnosti $\|x\|_2 = \|y\|_2 = r$.



Obr. 1.1: Elementární rotace v rovině, vektor x pootočením o úhel φ transformujeme na vektor y.

Uvažujme pro jednoduchost $r = 1$ a vyjádřeme oba vektory pomocí jejich souřadnic

$$x = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} \cos(\alpha) \\ \sin(\alpha) \end{bmatrix}, \qquad y = \begin{bmatrix} y_1 \\ y_2 \end{bmatrix} = \begin{bmatrix} \cos(\beta) \\ \sin(\beta) \end{bmatrix}.$$

Užitím elementárních goniometrických vztahů a rovnosti $\beta = \varphi + \alpha$ dostaneme

$$y = \begin{bmatrix} \cos(\varphi + \alpha) \\ \sin(\varphi + \alpha) \end{bmatrix} = \begin{bmatrix} \cos(\varphi)\cos(\alpha) - \sin(\varphi)\sin(\alpha) \\ \sin(\varphi)\cos(\alpha) + \cos(\varphi)\sin(\alpha) \end{bmatrix} =$$

$$= \begin{bmatrix} \cos(\varphi) & -\sin(\varphi) \\ \sin(\varphi) & \cos(\varphi) \end{bmatrix} \begin{bmatrix} \cos(\alpha) \\ \sin(\alpha) \end{bmatrix} \equiv G(\varphi)x.$$

Pro jednoduchost zavedeme označení $c \equiv \cos(\varphi)$, $s \equiv \sin(\varphi)$, matice pak bude mít tvar

$$G(\varphi) = \begin{bmatrix} c & -s \\ s & c \end{bmatrix},$$

nazveme ji *elementární Givensovou rotací*.

Nulování složky vektoru v pomocí Givensovy rotace, složené Givensovy rotace

Matici $G(\varphi)$ lze použít k nulování složek vektoru. Máme nenulový vektor $x \in R^2$, $x = [x_1 \quad x_2]^T$, který chceme zobrazit na vektor y,

$$y = \begin{bmatrix} \|x\|_2 \\ 0 \end{bmatrix} = \begin{bmatrix} c & -s \\ s & c \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix}.$$

Chceme tedy vektor x otočit o úhel $\varphi = -\alpha$ (viz obrázek 1.1), prvky Givensovy rotace $G(\varphi)$ jsou pak dány vztahy

$$c = \cos(\varphi) = \cos(\alpha) = \frac{x_1}{\|x\|_2}, \qquad s = \sin(\varphi) = -\sin(\alpha) = \frac{x_2}{\|x\|_2}.$$

Nyní máme obecný n-prvkový vektor $x \in R^n$, budeme hledat takové Givensovy rotace, které postupně vynulují co možná největší počet prvků vektoru. Vektor y budeme konstruovat postupně, podle schématu

$$x = \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix} = \begin{bmatrix} * \\ * \\ * \\ * \end{bmatrix} \rightarrow \begin{bmatrix} * \\ * \\ \bullet \\ \bullet \end{bmatrix} \rightarrow \begin{bmatrix} * \\ \bullet \\ \bullet \\ 0 \end{bmatrix} \rightarrow \begin{bmatrix} \bullet \\ \bullet \\ 0 \\ 0 \end{bmatrix} \rightarrow \begin{bmatrix} * \\ 0 \\ 0 \\ 0 \end{bmatrix} = \begin{bmatrix} \|x\|_2 \\ 0 \\ \vdots \\ 0 \end{bmatrix} = y,$$

kde symbol $*$ značí obecně nenulový prvek, dvojice symbolů $\bullet$ značí prvky, které jsou

aktuálně modifikovány elementární rotací. Maticově zapsáno

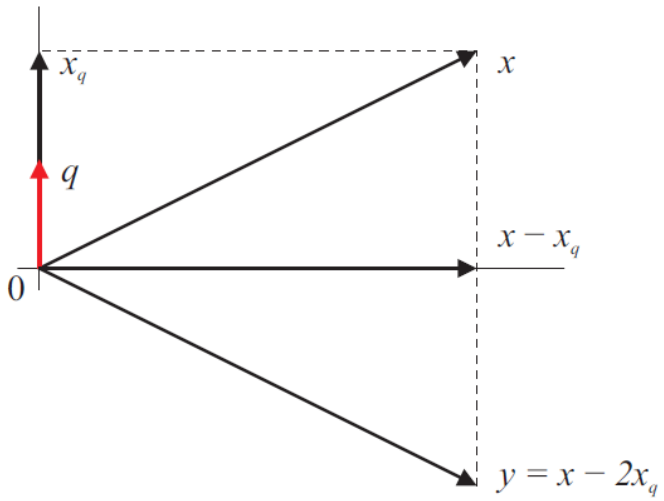
$$y = G_{12}G_{23}\ldots G_{n-1,n}x \equiv \Gamma x, \qquad \text{kde} \qquad G_{i,i+1} = \begin{bmatrix} I_{i-1} & 0 & 0 \\ 0 & G(\varphi) & 0 \\ 0 & 0 & I_{n-i-1} \end{bmatrix}$$

a matici $\Gamma = G_{12}G_{23}\ldots G_{n-1,n}$ budeme nazývat složená Givensova rotace. Matice Γ je vyjádřena jako součin elementárních rotací a je třeba dát pozor na pořadí jednotlivých činitelů. Prvky vektoru nulujeme postupně, ne nutně v naznačeném pořadí. Vždy vybereme rovinu, ve které vektor pootočíme tak, aby se vynulovala jedna ze dvou jeho souřadnic v této rovině.

1.2 Householderova reflexe

Druhou ortogonální transformací je Householderova reflexe (zrcadlení, odraz). Odvození tzv. Householderovy matice provedeme v obecném n-rozměrném případě. V R^n je dána $(n-1)$-rozměrná nadrovina $\mathcal{H}$, kterou můžeme jednoznačně popsat jejím normálovým vektorem q, předpokládejme $\|q\|_2 = 1$. Nadrovina $\mathcal{H}$ je nadrovinou zrcadlení. Libovolný vektor $x \in R^n$ můžeme rozložit na složku ležící ve směru normálového vektoru $x_q = (qq^T)x$ a na složku ortogonální na q, tedy na složku ležící v dané nadrovině, viz obrázek 1.2. Vektor y, zrcadlový obraz vektoru x podle nadroviny $\mathcal{H}$, získáme jednoduše. Složku x_q zaměníme za $(-x_q)$ a složka ležící v nadrovině $\mathcal{H}$ se nezmění, tedy

$$y = (x - x_q) - x_q = x - 2x_q = (I_n - 2qq^T)x \equiv H(q)x.$$



Obr. 1.2: Householderova reflexe. Matici reflexe získáme jako rozdíl jednotkové matice a dvojnásobku projektoru qq^T do směru normály nadroviny zrcadlení, tedy $H(q) = \left(I_n - 2qq^T\right)$.

Matici $H(q) = \left(I_n - 2qq^T\right)$ nazýváme *elementární Householderovou reflexí*. Realizuje zrcadlení podle nadroviny dané normálovým vektorem q.

Matice $H(q)$ je symetrická, ortogonální, je tedy sama sobě inverzí. Nezáleží na orientaci vektoru q, neboť pro $\tilde{q} = (-q)$, platí $\tilde{q}\tilde{q}^T = qq^T$, tedy

$$H(q) = H^T(q) = H^{-1}(q) = H(-q).$$

Často potřebujeme zkonstruovat Householderovu matici na základě znalosti vektoru x (vzoru) a vektoru y (obrazu). Jsou-li dány vektory $x, y \in R^n$ tak, že $x \neq y$ a $\|x\|_2 = \|y\|_2$, pak vektor

$$q = \frac{x - y}{\|x - y\|_2}$$

je zřejmě normálovým vektorem nadroviny zrcadlení, která zrcadlí x na y.

Složené Householderovy reflexe

Nyní se ještě seznámíme se speciální složenou reflexí. Nechť $Q = \{q_1,..., q_m\} \subset R^n$ je ortonormální soubor vektorů tj. vektory q_i jsou normalizované $\|q_i\|_2 = 1$ a vzájemně ortogonální, $q_i^T q_j = \delta_{ij}$, $\delta_{ij} = 0$ pro $i \neq j$, $\delta_{ij} = 1$ pro $i = j$. Tyto vektory v R^n jednoznačně určují $(n - m)$-rozměrnou nadrovinu (ortogonální doplněk prostoru, který je množinou vektorů Q generován), jíž jsou normálové vektory. Nechť matice $H(q_i)$, $i = 1 \ldots m$ jsou elementární reflexe. Z ortogonality vektorů q_i, q_j pak plyne

$$H(q_i)H(q_j) = \left(I_n - 2q_iq_i^T\right)\left(I_n - 2q_jq_j^T\right) = I_n - 2q_iq_i^T - 2q_jq_j^T = H(q_j)H(q_i)$$

komutativita elementárních reflexí. Tedy složená reflexe vzniklá pronásobením všech matic $H(q_i)$, je nezávislá na pořadí násobení. Výslednou matici můžeme vyjádřit ve tvaru

$$\prod_{i=1}^{m} H(q_i) = \prod_{i=1}^{m} \left(I_n - 2q_iq_i^T\right) = I_n - 2\sum_{i=1}^{m} q_iq_i^T = I_n - 2QQ^T,$$

kde $Q = [q_1, \ldots, q_m] \in R^{nxm}$. Householderovy reflexe ve vzájemně ortogonálních směrech jsou tedy komutativní.

Nulování složek vektoru v R^n pomocí Householderových reflexí

Householderovu reflexi můžeme využít k nulování prvků vektoru. Pro libovolnou reflexi H platí, že $\|x\|_2 = \|Hx\|_2 = \|y\|_2$. Mějme vektor $x \in R^n$, budeme hledat takovou reflexi, aby platilo

$$x = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix} \rightarrow \begin{bmatrix} \|x\|_2 \\ 0 \\ \vdots \\ 0 \end{bmatrix} = Hx = y,$$

přičemž předpokládáme $x \neq y$ (v opačném případě je $H = I_n$). Dosadíme-li $y = \|x\|_2 e_1$ do vztahu

$$q = \frac{x - y}{\|x - y\|_2} \qquad \text{získáme} \qquad q = \frac{x - \|x\|_2 e_1}{\left\|\left(x - \|x\|_2 e_1\right)\right\|_2},$$

pro $H(q) = \left(I_n - 2qq^T\right)$ pak platí $y = H(q)x$.

1.3 Vztah mezi Givensovými rotacemi a Householderovými reflexemi

Každou elementární i složenou Givensovu rotaci lze složit z Householderových reflexí. Mějme dánu Givensovu rotaci $G = \begin{bmatrix} c & -s \\ s & c \end{bmatrix}$ a definujme dva normalizované vektory

$$q_1 = \frac{\sqrt{2}}{2} \begin{bmatrix} \sqrt{1-c} \\ \sqrt{1+c} \end{bmatrix}, \qquad q_2 = \begin{bmatrix} 0 \\ 1 \end{bmatrix},$$

zřejmě

$$\|q_1\|_2 = \frac{1}{2}(1 - c + 1 + c) = 1 = \|q_2\|_2.$$

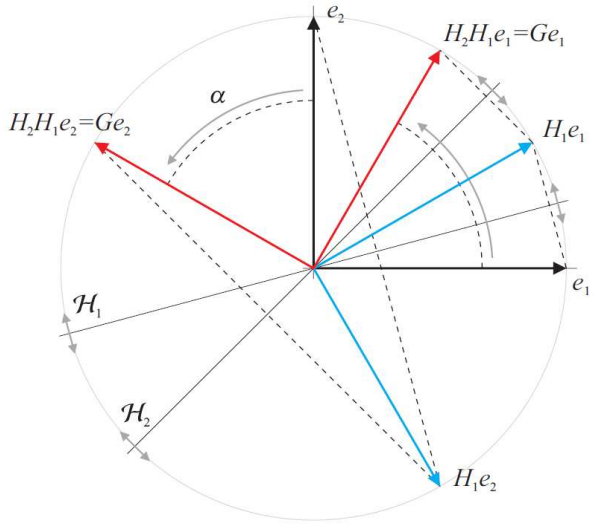
Zkonstruujeme jím odpovídající Householderovy matice

$$H_1 = I_n - 2q_1q_1^T = \begin{bmatrix} c & -s \\ -s & -c \end{bmatrix}, \qquad H_2 = \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix},$$

kde $s \equiv \sqrt{1 - c^2}$. Pak

$$H_2H_1 = \begin{bmatrix} c & -s \\ s & c \end{bmatrix} = G.$$

Libovolnou elementární rotaci G lze ze dvou reflexí složit nekonečně mnoha způsoby. Ke každému libovolně zvolenému normalizovanému vektoru q_1 existuje takový normalizovaný vektor q_2, že pro $H_1 = I_n - 2q_1q_1^T$, $H_2 = I_n - 2q_2q_2^T$ platí $H_2H_1 = G$. Na obrázku 1.3 vidíme další variantu konstrukce Givensovy rotace G užitím dvou nadrovin zrcadlení.



Obr. 1.3: Způsob, jak zkonstruovat rotaci $G(\alpha)$, $\alpha = \pi/3$, pomocí dvou reflexí H_1, H_2. $\mathcal{H}_1$, $\mathcal{H}_2$ jsou nadroviny zrcadlení, v tomto případě přímky.

V předcházejícím textu a na obrázku 1.3 jsme ukázali, jak lze elementární (a tedy i složenou) rotaci vyjádřit pomocí reflexí. Obecně však nelze (složenou) reflexi vyjádřit pomocí rotací. Naznačíme jednoduchý důkaz sporem.

Předpokládejme, že elementární reflexi H lze vyjádřit jako součin dvou elementárních rotací $H = G_1G_2$. Snadno ověříme, že pro (elementární) rotaci G platí $\det(G) = 1$. Pro elementární reflexi H naopak platí $\det(H) = -1$. Užitím vztahu pro determinant součinu matic $W = UV$

$$\det(W) = \det(U)\det(V),$$

dojdeme ke sporu, čímž je důkaz v podstatě dokončen.

Elementární reflexi nelze složit z rotací. Pomocí rotací zřejmě nelze vyjádřit ani žádná taková reflexe, která je složena z lichého počtu elementárních reflexí. Pro více informací o ortogonálních transformacích viz např. učební texty [8, 9, 2].

2. QR rozklad

Dvojici matic Q a R nazveme QR rozkladem (faktorizací) obecné obdélníkové matice $A \in R^{nxm}$ hodnosti $r = \text{rank}(A) \leq \min\{m, n\}$ pokud platí, že

$$A = QR,$$

a $Q \in R^{nxn}$ je ortogonální matice (tj. $Q^T Q = QQ^T = I_n$) a $R \in R^{nxm}$ je matice s nulovými poddiagonálními prvky (horní trojúhelníková matice) stejných rozměrů jako matice A. Často se též můžeme setkat s ekonomickou formou QR rozkladu

$$A = Q_r R_{rm},$$

kde $Q_r \in R^{nxr}$ je matice obsahující prvních r ortonormálních sloupců matice Q (tj. $Q_r^T Q_r = I_r$) a $R_{rm} \in R^{rxm}$ je matice s nulovými poddiagonálními prvky (horní trojúhelníková matice) obsahující prvních r řádků matice *R*. QR rozklad lze spočítat užitím Givensových rotací a Householderových reflexí.

2.1 QR rozklad užitím Givensových rotací

Označme a_j j-tý sloupec matice $A = [a_1, \ldots, a_n]$. Maticí Γ_1 označíme složenou Givensovou rotaci, která realizuje transformaci

$$a_1 = \begin{bmatrix} a_{11} \\ a_{21} \\ \vdots \\ a_{n1} \end{bmatrix} \rightarrow \begin{bmatrix} \|a_1\|_2 \\ 0 \\ \vdots \\ 0 \end{bmatrix} = \Gamma_1 a_1 \equiv r_1.$$

Budeme-li tuto složenou rotaci aplikovat na matici *A* dostaneme

$$A = \begin{bmatrix} \bullet & * & * & * \\ \bullet & * & * & * \\ \bullet & * & * & * \\ \bullet & * & * & * \end{bmatrix} \rightarrow \begin{bmatrix} * & * & * & * \\ 0 & * & * & * \\ 0 & * & * & * \\ 0 & * & * & * \end{bmatrix} = \Gamma_1 A \equiv A^{(1)},$$

kde symbol * značí obecně nenulový prvek, symboly $\bullet$ vyznačují prvky, pomocí kterých konstruujeme složenou rotaci.

Nyní budeme transformovat (nulovat) druhý sloupec matice $A^{(1)} = [r_1, a_2^{(1)}, ..., a_n^{(1)}]$ označený $a_2^{(1)}$. Přirozeně chceme zachovat již vytvořené nulové prvky ve sloupci r_1. Na druhý sloupec tedy aplikujeme jen $n-2$ elementárních rotací (nulujeme pouze

poddiagonální prvky)

$$a_2^{(1)} = \begin{bmatrix} \frac{a_{12}^{(1)}}{a_{22}^{(1)}} \\ a_{32}^{(1)} \\ \vdots \\ a_{n2}^{(1)} \end{bmatrix} \rightarrow \begin{bmatrix} \frac{a_{12}^{(1)}}{\left\| a_{2:n,2}^{(1)} \right\|_2} \\ 0 \\ \vdots \\ 0 \end{bmatrix} = \Gamma_2 a_2^{(1)} \equiv r_2 .$$

Aplikací složené rotace Γ_2 na matici $A^{(1)} = \Gamma_1 A$ dostaneme

$$A^{(1)} = \begin{bmatrix} * & * & * & * \\ 0 & \bullet & * & * \\ 0 & \bullet & * & * \\ 0 & \bullet & * & * \end{bmatrix} \rightarrow \begin{bmatrix} * & * & * & * \\ 0 & * & * & * \\ 0 & 0 & * & * \\ 0 & 0 & * & * \end{bmatrix} = \Gamma_2 \Gamma_1 A \equiv A^{(2)}.$$

Analogicky pokračujeme dále. Obecně v *k*-tém kroku, *k* = 1,...,*n* konstruujeme složenou rotaci tak, aby nulovala $n - k$ poddiagonálních prvků. Jde o složenou rotaci, která pracuje s vektorem délky *n* – *k* + 1. Matice označená Γ_k je blokově diagonální s obecně dvěma bloky. První blok je jednotková matice řádu $k - 1$, druhý blok je složená rotace dimenze *n* – *k* + 1. Pro *k* = 1 matice Γ_1 jednotkový blok neobsahuje (jednotkový blok má dimenzi 0), pro *k* = *n* je $\Gamma_n = I_n$. Poslední transformační maticí, kterou má smysl uvažovat, je tedy Γ_{n-1},

$$A^{(n-2)} = \begin{bmatrix} * & * & * & * \\ 0 & * & * & * \\ 0 & 0 & \bullet & * \\ 0 & 0 & \bullet & * \end{bmatrix} \rightarrow \begin{bmatrix} * & * & * & * \\ 0 & * & * & * \\ 0 & 0 & * & * \\ 0 & 0 & 0 & * \end{bmatrix} = \Gamma_{n-1} \ldots \Gamma_1 A \equiv A^{(n-1)} \equiv R.$$

Všechny matice Γ_i jsou ortogonální, jejich součin je tedy také ortogonální maticí, označme

$Q \equiv \Gamma_1^T \Gamma_2^T \ldots \Gamma_{n-1}^T$, pak zřejmě platí $A = QR$.

Matice *Q* je z konstrukce vždy čtvercová dimenze *n* a ortogonální. Matice $R = [r_1, \ldots, r_n]$ je horní trojúhelníková a z konstrukce je vždy stejné dimenze jako matice *A* (připomeňme, že matici *A* jsme uvažovali obecně obdélníkovou, $A \in R^{nxm}$).

2.2 QR rozklad užitím Householderových reflexí

Postup s užitím Householderových reflexí je konstrukčně stejný, pouze místo matic Γ_i používáme matice H_i aplikované, stejně jako v předchozím případě, na vektory zmenšující se

délky, tedy schématicky

$$\begin{bmatrix} \bullet & * & * & * \\ \bullet & * & * & * \\ \bullet & * & * & * \\ \bullet & * & * & * \end{bmatrix} \to \begin{bmatrix} * & * & * & * \\ 0 & \bullet & * & * \\ 0 & \bullet & * & * \\ 0 & \bullet & * & * \end{bmatrix} \to \begin{bmatrix} * & * & * & * \\ 0 & * & * & * \\ 0 & 0 & \bullet & * \\ 0 & 0 & \bullet & * \end{bmatrix} \to \begin{bmatrix} * & * & * & * \\ 0 & * & * & * \\ 0 & 0 & * & * \\ 0 & 0 & 0 & * \end{bmatrix},$$

kde jsou postupně matice $A \to H_1 A \to (H_2 H_1)A \to \ldots \to (H_{n-1} \ldots H_2 H_1)A \equiv R$. Označme

$$Q \equiv H_1 H_2 \ldots H_{n-1}, \qquad \text{pak zřejmě platí} \qquad A = QR.$$

(Připomeňme, že matice elementárních Householderových reflexí jsou symetrické.)

2.3 Vzájemná zaměnitelnost rotací a reflexí

Složenou nebo elementární reflexi (resp. rotaci) W můžeme, za předpokladu, že to lze, viz kapitola 1.3, vyjádřit pomocí rotací (resp. reflexí), spočteme-li QR rozklad $W = QR$ této matice právě užitím rotací (resp. reflexí). Zvolíme-li vhodný postup vyjde matice $R = \text{diag}(1, \ldots, 1, \pm 1)$. V případě, že nastane $R = I_n$, platí $W = Q$ a příslušnou transformační matici jsme vyjádřili pomocí transformací druhého typu, v opačném případě W takto vyjádřit nelze.

2.4 QR rozklad užitím Gram-Schmidtova ortogonalizačního procesu

Nechť je dán libovolný soubor *lineárně nezávislých* vektorů $\{a_1, \ldots, a_m\} \subset R^n$ reprezentovaný maticí $A \equiv [a_1, \ldots, a_m] \in R^{nxm}$. Budeme se snažit nalézt ortogonální bázi prostoru $R(A) = \text{span}(a_1, \ldots, a_m)$. Postup, který použijeme se nazývá Gram-Schmidtův ortogonalizační proces. Algoritmus lze snadno odvodit, vyjdeme-li z QR rozkladu matice $A = QR$, na který se budeme dívat po sloupcích. Zřejmě pro první sloupec platí

$$a_1 = q_1 r_{11},$$

pro k-tý sloupec obecně máme

$$a_k = \sum_{i=1}^{k-1} q_i r_{ik} + q_k r_{kk},$$

kde r_{ik} značí i-tý prvek k-tého sloupce matice R. Vezmeme tedy první vektor a_1, podělíme ho jeho normou a získáme tak první vektor ortogonální báze, označíme

$$r_{11} \equiv \|a_1\|_2, \qquad q_1 \equiv \frac{a_1}{r_{11}}.$$

Druhý vektor a_2 ortogonalizujeme proti q_1 (odečteme od a_2 jeho ortogonální projekci do směru q_1) a zbytek normalizujeme, tedy

$$r_{12} \equiv q_1^T a_2,$$
$$z \equiv a_2 - q_1 r_{12} = a_2 - \left(q_1 q_1^T\right) a_2 = \left(I_n - q_1 q_1^T\right) a_2,$$
$$r_{22} \equiv \|z\|_2,$$
$$q_2 \equiv \frac{z}{r_{22}}.$$

Třetí vektor ortogonalizujeme proti vektorům q_1, q_2, zbytek normalizujeme, čímž dostaneme vektor q_3. Postup se opakuje, dokud nevyčerpáme celou množinu vektorů. V *k*-tém kroku pracujeme s vektorem a_k, pomocný vektor vyjádříme

$$z \equiv \left(I_n - Q_{k-1} Q_{k-1}^T\right) a_k = a_k - Q_{k-1} Q_{k-1}^T a_k = a_k - \sum_{i=1}^{k-1} q_i q_i^T a_k,$$

kde $Q_j \equiv [q_1, \ldots, q_j]$. Vzhledem k tomu, že soubor vektorů je lineárně nezávislý, bude vektor z vždy nenulový. Označíme-li

$$R_{mm} \equiv \begin{bmatrix} r_{11} & r_{12} & \cdots & r_{1m} \\ 0 & r_{22} & & r_{2m} \\ \vdots & \ddots & \ddots & \vdots \\ 0 & \cdots & 0 & r_{mm} \end{bmatrix} \in R^{mxm},$$

pak platí

$$A = Q_m R_{mm}.$$

Dále zřejmě platí

$$\operatorname{span}(a_1, \ldots, a_m) = R(A) = R(Q_m) = \operatorname{span}(q_1, \ldots, q_m),$$

z konstrukce vektorů q_i navíc vyplývá

$$\begin{aligned} \operatorname{span}(a_1) &= R(Q_1) = \operatorname{span}(q_1), \\ \operatorname{span}(a_1, a_2) &= R(Q_2) = \operatorname{span}(q_1, q_2), \\ &\vdots \\ \operatorname{span}(a_1, \ldots, a_k) &= R(Q_k) = \operatorname{span}(q_1, \ldots, q_k), \qquad \forall k = 1, \ldots, m. \end{aligned}$$

Pokud je soubor vektorů $\{a_1, \ldots, a_m\}$ *lineárně závislý,* pak existuje takové uspořádání vektorů, že pro nějaké r, $1 \le r < m$ platí $\forall k = r$, $a_i \in \operatorname{span}(a_1, \ldots, a_r)$. Číslo *r* je hodnost

matice $A = [a_1, \ldots, a_m]$ (v případě lineárně nezávislého souboru vektorů je $r = m$), rozklad matice má tvar

$$A = [q_1, \ldots, q_r] \begin{bmatrix} r_{11} & r_{12} & \cdots & r_{1r} & \cdots & r_{1m} \\ 0 & r_{22} & & r_{2r} & \cdots & r_{2m} \\ \vdots & \ddots & \ddots & \vdots & & \vdots \\ 0 & \cdots & 0 & r_{rr} & \cdots & r_{rm} \end{bmatrix} \equiv Q_r R_{rm}.$$

Výpočet pomocného vektoru – paralelní a sekvenční algoritmus

Vztah pro výpočet pomocného vektoru v *k*-tém kroku

$$z \equiv \left(I_n - Q_{k-1}Q_{k-1}^T\right)a_k = a_k - Q_{k-1}Q_{k-1}^T a_k = a_k - \sum_{i=1}^{k-1} q_i q_i^T a_k,$$

lze (s využitím ortogonality vektorů q_i) přepsat dvěma matematicky ekvivalentními způsoby. První „paralelní“ zápis lze algoritmicky nejlépe vyjádřit vztahem

$$z = a_k - (a_k, q_1)q_1 - (a_k, q_2)q_2 - (a_k, q_3)q_3 \ldots - (a_k, q_{k-1})q_{k-1},$$

přepsáno matematicky

$$z = \left(I_n - q_1q_1^T - q_2q_2^T - q_3q_3^T \ldots - q_{k-1}q_{k-1}^T\right)a_k.$$

Druhý „sekvenční“ zápis vyjádříme algoritmicky vztahy

$$z_1 = a_k - (a_k, q_1)q_1,$$

$$z_2 = z_1 - (z_1, q_2)q_2,$$

$$z_3 = z_2 - (z_2, q_3)q_3,$$

$$\vdots$$

$$z = z_{k-2} - (z_{k-2}, q_{k-1})q_{k-1},$$

přepsáno matematicky

$$z = \left(I_n - q_{k-1}q_{k-1}^T\right)\ldots\left(I_n - q_3q_3^T\right)\left(I_n - q_2q_2^T\right)\left(I_n - q_1q_1^T\right)a_k.$$

Tyto dva zápisy jsou sice matematicky ekvivalentní, ale povedou na algoritmy s různými numerickými vlastnostmi.

Algoritmické zápisy Gram-Schmidtova procesu

V dalším textu vyložíme dvě základní varianty Gram-Schmidtova procesu, klasický a modifikovaný algoritmus. Klasický algoritmus se od modifikovaného liší pouze rozdělením vnitřního cyklu, v němž dochází k ortogonalizaci vektoru a_k proti již spočtené ortonormální

bázi $q_1, \ldots, q_{k-1}$. Zde využijeme právě paralelního resp. sekvenčního zápisu výpočtu pomocného vektoru.

Matematicky jsou oba algoritmy ekvivalentní, zásadně se však liší v konečné aritmetice a paralelizovatelnosti. Pro zápis algoritmu budeme uvažovat soubor $\{a_1, \ldots, a_m\} \subset R^n$ lineárně nezávislých vektorů, platí tedy $m \leq n$.

2.4.1 Klasický Gram-Schmidtův algoritmus

Klasický Gram-Schmidtův algoritmus (CGS) vychází z paralelních vztahů. V *i*-tém kroku algoritmu spočítáme všechny skalární součiny $r_{ik} = (a_k, q_i), \quad i = 1, \ldots, k-1,$ pak provádíme update pomocného vektoru *z*. Výpočet $k - 1$ skalárních součinů můžeme provádět současně (paralelně). CGS algoritmus je tak velmi dobře paralelizovatelný. Nevýhodou tohoto algoritmu je, že chyby k nimž dochází vlivem zaokrouhlování v konečné aritmetice, se mohou při výpočtu pomocného vektoru *z* kumulovat a mohou způsobit významnou ztrátu ortogonality mezi sloupci matice Q_m.

CGS – klasický Gram-Schmidtův algoritmus

```
00    r_11 = ||a_1||_2
01    q_1 = a_1 / r_11
02    for k = 2 : m do
03        z = a_k
04        for i = 1 : k − 1 do
05            r_ik = q_i^T z
06        end
07        for i = 1 : k − 1 do
08            z = z − q_i r_ik
09        end
10        r_kk = ||z||_2
11        q_k = z / r_kk
12    end
```

2.4.2 Modifikovaný Gram-Schmidtův algoritmus

Modifikovaný Gram-Schmidtův algoritmus (MGS) vychází ze sekvenčních vztahů. Od CGS se liší pouze spojením vnitřních cyklů (tj. vymazání řádků 06 a 07 z předchozího

algoritmu). Výpočet každého skalárního součinu r_{ik}, $1 < i < k$ následuje až po výpočtu updatu $z = z - q_{i-1} r_{i-1,k}$, nelze je tedy provádět paralelně. MGS je příkladem algoritmu, který lze paralelizovat jen na nejnižší úrovni, tj. na úrovni vektorových operací. Ztráta paralelizmu je vyvážena tím, že chyby vznikající při updatech vektoru z jsou částečně eliminovány. Při ortogonalizaci proti vektoru q_i dojde k částečnému opravení chyb ve směru q_i, které ve vektoru z vznikly při předchozích operacích. Ztráta ortogonality mezi sloupci matice Q_m je u modifikovaného algoritmu výrazně nižší než u algoritmu klasického.

MGS – modifikovaný Gram-Schmidtův algoritmus

```
00    r_11 = ||a_1||_2
01    q_1 = a_1 / r_11
02    for k = 2 : m do
03        z = a_k
04        for i = 1 : k − 1 do
05            r_ik = q_i^T z
06            z = z − q_i r_ik
07        end
08        r_kk = ||z||_2
09        q_k = z / r_kk
10    end
```

2.4.3 Iterační zpřesnění (reortogonalizace)

Výhodou klasického Gram-Schmidtova algoritmu (CGS) je paralelnost výpočtu, ale výsledky jsou méně přesné. Naopak modifikovaný Gram-Schmidtův algoritmus (MGS) má přesnější výsledky, ale používá sekvenční výpočet. Pokud provedeme CGS algoritmus, resp. celý jeho vnější cyklus (s drobnými úpravami), dvakrát (tzv. iterační zpřesnění), získáme výsledky s přesností srovnatelnou a často dokonce lepší než u MGS algoritmu. Navíc algoritmus zůstane snadno paralelizovatelný.

3. Bidiagonalizace

Jedná se o transformaci matice na bidiagonální tvar. Matici nazveme horní bidiagonální tehdy, má-li nenulovou pouze hlavní diagonálu a první naddiagonálu a všechny ostatní prvky jsou nulové, tedy

$$L = \begin{bmatrix} \beta_1 & \alpha_1 & & & \\ & \beta_2 & \alpha_2 & & \\ & & \beta_3 & \alpha_3 & \\ & & & \ddots & \ddots \end{bmatrix}.$$

V následujícím textu se seznámíme s metodami pro získání matic v bidiagonálním tvaru

- výpočtem pomocí Householderových reflexí případně Givensových rotací,
- přímým výpočtem jednotlivých prvků bidiagonální matice (Golub-Kahanův algoritmus [6]).

První postup je přesnější, nepatrně náročnější a tudíž je vhodný pro malé obecně husté úlohy. Druhý postup je méně přesný, rychlejší a výpočetně méně náročný, tudíž je vhodný i pro velké řídké problémy.

3.1 Bidiagonalizace pomocí Householderových matic

Bidiagonalizace pomocí Householderových matic je první z metod pro výpočet bidiagonální matice, kterou zmíníme. Je-li naším cílem vysoká přesnost spočtených matic, je nejčastěji používanou metodou. Je též poměrně jednoduchá na implementaci. Takto získané výsledky jsou velmi přesné. Metoda spočívá v transformaci dané matice A na bidiagonální matici L za pomoci dvou složených Householderových reflexí tak, že platí

$$L = H_L^T A H_P,$$

kde H_L je levá (složená) Householderova transformace nulující prvky ve sloupcích a H_P je pravá (složená) Householderova transformace nulující prvky v řádcích. Tyto matice jsou sestaveny jako součin elementárních Householderových transformací

$$H_L = H_1 H_3 H_5 \ldots,$$

$$H_P = H_2 H_4 H_6 \ldots.$$

kde jednotlivé matice H_i mají tvar $H_i = I - 2qq^T$ (podrobně popsáno v kapitole 1.2).

Výslednou bidiagonální matici vytváříme podle schématu:

$$H_1A = \begin{bmatrix} * & * & * & * & * \\ \bullet & * & * & * & * \\ \bullet & * & * & * & * \\ \bullet & * & * & * & * \\ \bullet & * & * & * & * \end{bmatrix}, \qquad H_1AH_2 = \begin{bmatrix} * & * & \bullet & \bullet & \bullet \\ & * & * & * & * \\ & * & * & * & * \\ & * & * & * & * \\ & * & * & * & * \end{bmatrix},$$

$$H_3H_1AH_2 = \begin{bmatrix} * & * & & & \\ & * & * & * & * \\ & \bullet & * & * & * \\ & \bullet & * & * & * \\ & \bullet & * & * & * \end{bmatrix}, \qquad H_3H_1AH_2H_4 = \begin{bmatrix} * & * & & & \\ & * & * & \bullet & \bullet \\ & & * & * & * \\ & & * & * & * \\ & & * & * & * \end{bmatrix},$$

$$H_5H_3H_1AH_2H_4 = \begin{bmatrix} * & * & & & \\ & * & * & & \\ & & * & * & * \\ & & \bullet & * & * \\ & & \bullet & * & * \end{bmatrix}, \qquad H_5H_3H_1AH_2H_4H_6 = \begin{bmatrix} * & * & & & \\ & * & * & & \\ & & * & * & \bullet \\ & & & * & * \\ & & & * & * \end{bmatrix},$$

$$H_7H_5H_3H_1AH_2H_4H_6 = \begin{bmatrix} * & * & & & \\ & * & * & & \\ & & * & * & \\ & & & * & * \\ & & & \bullet & * \end{bmatrix}, \qquad L = \begin{bmatrix} * & * & & & \\ & * & * & & \\ & & * & * & \\ & & & * & * \\ & & & & * \end{bmatrix},$$

kde * symbolizuje obecně nenulový prvek matice, $\bullet$ symbolizuje prvek vynulovaný v daném kroku výpočtu.

Analogicky lze matici *A* převést na dolní bidiagonální tvar (tj. na matici mající nenulovou pouze hlavní diagonálu a první poddiagonálu).

3.1.1 Zjednodušení výpočtu užitím vnitřního QR rozkladu

Je-li zadaná matice A^{nxm} výrazně obdélníková tak, že $n >> m$ můžeme před vlastní bidiagonalizací provést nejprve QR rozklad, kterým matici *A* rozložíme na součin matic $A = Q_m R_{rm}$ (princip výpočtu je popsán v kapitole 2). Matice Q_m má stejnou velikost jako původní matice *A*, matice R_{rm} je horní trojúhelníková. Následně bidiagonalizujeme matici R_{rm}. Analogicky, je-li $n << m$, pak můžeme provést před vlastní bidiagonalizací QR rozklad matice $A^T = Q_n R_{rn}$ (tzv. LQ rozklad matice *A*) a následně bidiagonalizovat matici R_{rn}^T. Na závěr je nutno matice H_L a H_P, které nám při výpočtu vyjdou, vynásobit maticí Q_m,

případně Q_n tak, abychom získali rozklad původní matice A. Tedy v prvním případě dostaneme

$$A = Q_m R_{rm},$$

$$H_L^T R_{rm} H_P = L,$$

$$R_{rm} = H_L L H_P^T,$$

$$A = (Q_m H_L) L H_P^T.$$

V druhém případě analogicky dostaneme

$$A = R_{rn}^T Q_n^T,$$

$$A = H_L L H_P^T Q_n^T = H_L L (Q_n H_P)^T.$$

Vhodným užitím QR, resp. LQ rozkladu se při výpočtu bidiagonalizace sníží výpočetní nároky a zkrátí čas potřebný k výpočtu, neboť pracujeme s menšími maticemi.

3.2 Bidiagonalizace pomocí Givensových matic

Další metoda bidiagonalizace je využití Givensových matic (jejich konstrukce je podrobně popsána v kapitole 1.1). Podstata výpočtu bidiagonální matice je stejná jako u Householderových matic, liší se pouze v ortogonální transformaci použité pro nulování vektoru (sloupce či řádku bidiagonalizované matice). Výpočet lze i zde zjednodušit užitím vnitřního QR, resp. LQ rozkladu. Metodu nebudeme tudíž dále podrobněji popisovat.

3.3 Golub-Kahanova bidiagonalizace

Tato metoda je založena na Golub-Kahanově bidiagonalizačním algoritmu (také nazývána Lanczosova bidiagonalizace nebo Lanczos-Golub-Kahanova bidiagonalizace), viz [6]. Tento přístup je rozdílný od metody využívající Householderových či Givensových matic, ale při jejím popisu z těchto metod vyjdeme (podobně jako jsme při popisu Gram-Schmidtova algoritmu vyšli z existence QR rozkladu spočteného např. užitím Householderových matic). Nechť $A = ULV^T$, kde $U \equiv H_L$, $V \equiv H_P$ a kde L je tentokrát *dolní* bidiagonální matice. Prvky bidiagonální matice jsou zřejmě nezáporné. Porovnáním sloupců v rovnicích $AV = UL$ a $A^T U = VL^T$ získáme algoritmus. Ten je obecně závislý na startovacím vektoru $s \in R^n$. Výstup matematicky ekvivalentní s Householderovou metodou získáme pro $s = [1,0,\ldots,0]^T$.

Golub-Kahanův algoritmus

00 $\beta_1 = \|s\|_2$

01 $u_1 = s / \beta_1$

02 $w_L = A^T u_1$

03 $\alpha_1 = \|w_L\|_2$

04 $v_1 = w_L / \alpha_1$

05 $w_R = Av_1 - \alpha_1 u_1$

06 $\beta_2 = \|w_R\|_2$

07 $u_2 = w_R / \beta_2$

08 **for** $j = 2,3,4,\ldots$

09 $w_L = A^T u_j - v_{j-1}\beta_j$

10 $\alpha_j = \|w_L\|_2$

11 $v_j = w_L / \alpha_j$

12 $w_R = Av_j - \alpha_j u_j$

13 $\beta_{j+1} = \|w_R\|_2$

14 $u_{j+1} = w_R / \beta_{j+1}$

15 **end**

Algoritmus výpočtu končí, je-li $\alpha_j = 0$ a $\beta_j = 0$. Uvažujeme, že normalizační koeficienty $\alpha_1,\ldots,\alpha_k$ a $\beta_1,\ldots,\beta_{k+1}$ jsou nenulové (kladné), pak pro libovolný startovací vektor *s* lze dokázat, že $\{u_j\}_{j=1}^{k+1} \subset R^n$, $\{v_j\}_{j=1}^{k} \subset R^m$ jsou dvě množiny vzájemně ortogonálních vektorů, viz [6]. Označme

$$U_j = [u_1,\ldots,u_j] \in R^{nxj}, \qquad V_j = [v_1,\ldots,v_j] \in R^{mxj},$$

pak platí

$$U_k^T A V_k = \begin{bmatrix} \alpha_1 & & & \\ \beta_2 & \alpha_2 & \ddots & \\ & \ddots & \ddots & \\ & & \beta_k & \alpha_k \end{bmatrix} \equiv L_k, \qquad U_{k+1}^T A V_k = \begin{bmatrix} L_k \\ \hline e_k^T \beta_{k+1} \end{bmatrix} \equiv L_{k+}.$$

Obecně tento algoritmus nedává úplný rozklad matice *A*, ale generuje pouze restrikci matice *A* na jistých Krylovových podprostorech rostoucí dimenze. Golub-Kahanův algoritmus je tedy vhodný, když chceme spočítat jen částečnou bidiagonalizaci matice *A*. Opět

i zde lze výpočet zjednodušit tím, že data předpřipravíme užitím vnitřního QR, resp. LQ rozkladu.

3.3.1 Golub-Kahanův algoritmus bez reortogonalizace

Výhodou prosté implementace tohoto algoritmu (bez reortogonalizace) je rychlost a nízké paměťové nároky. Když počítáme malou část bidiagonální matice, pracujeme pouze s několika sloupci matice *U* a *V*. Pokud je našim cílem spočítat pouze bidiagonální matici a nikoliv matice *U* a *V*, stačí si pamatovat pouze dva předcházející vektory.

Nevýhodou této prosté implementace je úplná ztráta ortogonality sloupců matic *U* a *V*. Normalizační koeficienty α_j a β_j jsou počítány nepřesně a tudíž se tato implementace může zdát prakticky nepoužitelná. Pro zachování ortogonality a přesnosti celého výpočtu je tedy nutné provádět v algoritmu reortogonalizaci produkovaných vektorů. V následujícím textu se budeme zabývat několika implementacemi Golub-Kahanova algoritmu, ve kterých se zaměříme zejména na různé strategie reortogonalizace produkovaných vektorů.

Zde je ovšem na místě zmínit, že pro opravdu velké problémy (reálně v současnosti počítané úlohy dosahující dimenze až $m, n \sim 10^9$) není jiná možnost než reortogonalizaci neprovádět. Tehdy je prostá implementace jedinou možnou volbou. V kapitole 4.5 se pokusíme na numerických experimentech ospravedlnit její použití.

3.3.2 Golub-Kahanův algoritmus s reortogonalizací

Lepších výsledků Golub-Kahanovy bidiagonalizace, měřeno kvalitou ortogonality spočtených vektorů, dosáhneme, když současně během výpočtu reortogonalizujeme pomocné vektory w_L a w_R proti předchozím vektorům $v_{j-1}, v_{j-2}, \ldots$, resp. $u_j, u_{j-1}, \ldots$ (sloupcům matic *V*, resp. *U*). Na reortogonalizaci lze v podstatě nahlížet jako na aplikaci klasického (CGS, kapitola 2.3.1) nebo modifikovaného (MGS, kapitola 2.3.2) Gram-Schmidtova otrogonalizačního procesu uvnitř Golub-Kahanova algoritmu.

Výhodou může být zlepšení ortogonality vektorů a stability výpočtu, v případě CGS nebo MGS s iteračním zpřesněním dokonce srovnatelnou s Householderovou metodou. Výpočetní čas se reortogonalizací zvýší, ale stále je srovnatelný s metodou využívající ortogonální transformace. Naopak paměťové nároky mohou být významně nižší než u Householderovy metody (zvlášť při výpočtu pouze částečné bidiagonalizace). Nevýhodou oproti prosté implementaci Golub-Kahanova algoritmu je, že výpočetní cena jedné iterace obecně roste s počtem iterací.

3.3.3 Ortogonalizační strategie

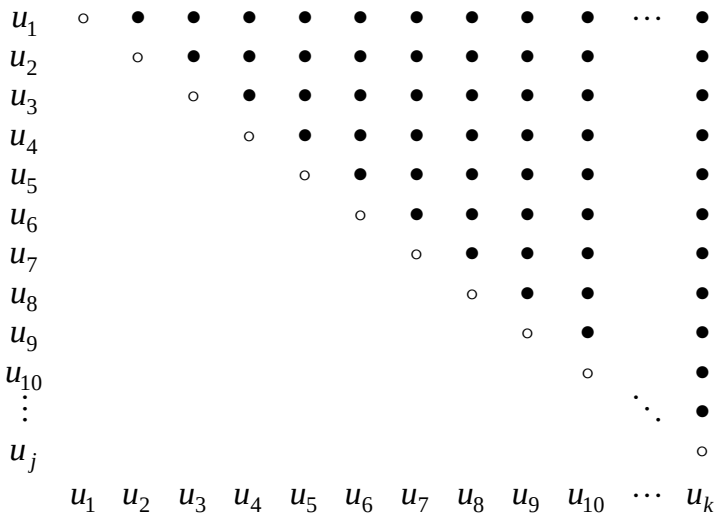
Ortogonalizace jak jsme již zmínili, je v podstatě prováděna Gram-Schmidtovým procesem, tedy jde v principu o výpočet QR rozkladů matic $[u_1, \ldots, u_k]$ a $[v_1, \ldots, v_k]$, které jsou ovšem Golub-Kahanovým algoritmem produkovány postupně po sloupcích. Golub-Kahanova bidiagonalizace a reortogonalizující QR rozklad tak běží simultánně. Různými ortogonalizačními strategiemi, které dále popíšeme, budeme určovat, které vektory u_j, resp. v_j při ortogonalizaci použijeme (z pohledu QR rozkladu tím vlastně určujeme tvar zaplnění horní trojúhelníkové matice *R*), přičemž naší snahou bude dosáhnout co největší přesnosti a zároveň co nejnižší ceny výpočtu (tedy co nejmenšího zaplnění matice *R*). Ve většině případů můžeme reortogonalizaci provádět jak pomocí klasického (CGS) tak modifikovaného (MGS) Gram-Schmidtova procesu. V dalším popisu se zaměříme pouze na vektory u_j.

3.3.3.1 Plná reortogonalizace

Plnou reortogonalizací rozumíme to, když provádíme reortogonalizaci vektoru u_k vždy proti všem předchozím vektorům $u_{k-1}, u_{k-2}, \ldots, u_1$. Použijeme ji tehdy, potřebujeme-li zajistit maximální možnou ortogonalitu vektorů. Skutečně stabilní implementace dosáhneme pouze, když vektory w_L (a w_R) reortogonalizujeme proti všem předešlým vektorům dvakrát (CGS, MGS s iteračním zpřesněním, viz kapitola 2.4), [4, 5]. Jednonásobná reortogonalizace není dostatečná, více než dvojnásobná reortogonalizace je zbytečná. Když provedeme dvakrát plnou reortogonalizaci dosáhneme výsledky srovnatelné s Householderovou metodou.

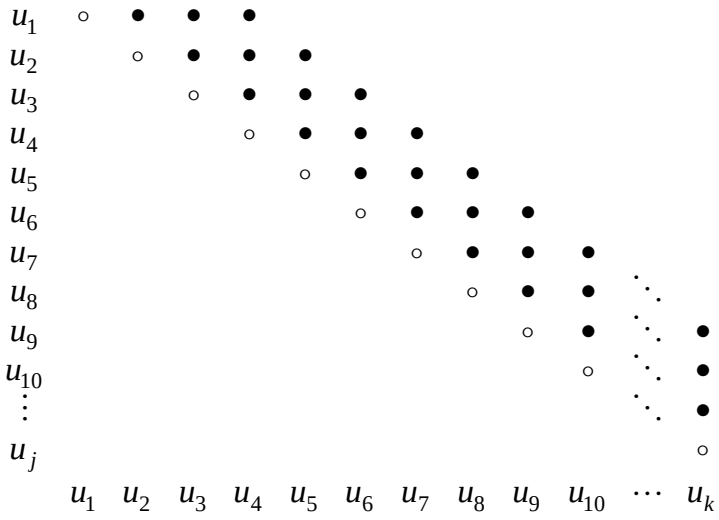
Pokud ovšem bidiagonalizujeme velmi velkou matici, plnou reortogonalizaci nelze použít, jelikož je velmi náročná na paměť a na čas.

Následující schéma ilustruje, které vektory používáme při reortogonalizaci vektorů u_j (tj. ukazuje tvar zaplnění matice *R*):



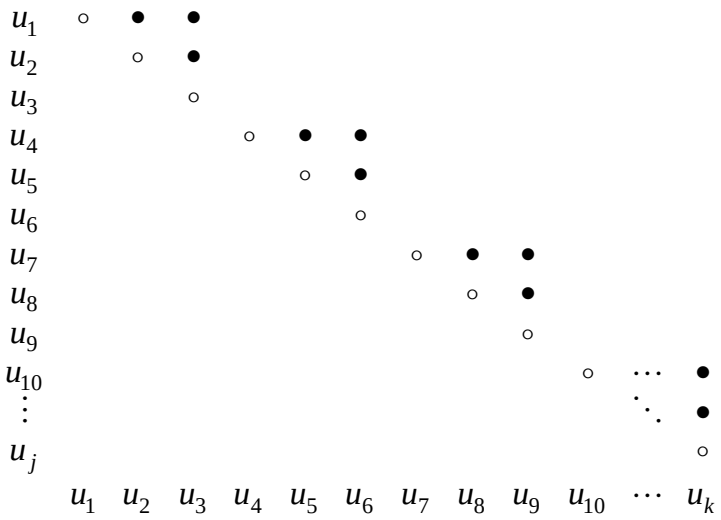
3.3.3.2 Pásová reortogonalizace

Jelikož plná ortogonalizace je velmi náročná, používáme i jiné ortogonalizační strategie. Jednou z nich je pásová reortogonalizace. Ortogonalizujeme vždy proti *l* předchozím vektorům. Výhodou je, že tím dosáhneme konstantní výpočetní ceny iterace. Hodnotu *l* můžeme zvolit tak, abychom maximálně využili paměťových prostředků a výkonových možností počítače. Vzhledem k tomu, že ortogonalizace není úplná, spočtené výsledky mohou obsahovat chyby. Nevýhodou je, že chyby se pravděpodobně mohou výpočtem šířit nežádoucím způsobem, viz následující kapitola 3.3.3.3. Následující schéma ilustruje tvar zaplnění matice *R* při pásové reortogonalizaci pro $l = 3$:



3.3.3.3 Restartovaná reortogonalizace

Počet vektorů, proti kterým ortogonalizujeme, se postupně v každém kroku zvyšuje tak jako u úplné reortogonalizace. Po *l* krocích ale dojde k restartu celého procesu a ortogonalizaci začínáme provádět znovu od začátku postupně proti 0, 1, 2, ..., $l - 1$ *předchozím* vektorům. Restartem se může zamezit šíření některých chyb algoritmem (viz předchozí kapitola 3.3.3.2), usuzujeme tak na základě faktu, že ortogonalizace stejného typu (restartovaná) se používá také např. v některých implementacích metody GMRES, narozdíl od ortogonalizace pásové, která se v metodě GMRES využívá jen zřídka, [3]. Při použití této strategie reortogonalizace sice nedosáhneme konstantní ceny iterace, ale jsme schopni maximální cenu iterace dobře omezit. Následující schéma ilustruje tvar zaplnění matice *R* při restartované reortogonalizaci pro *l* = 3:

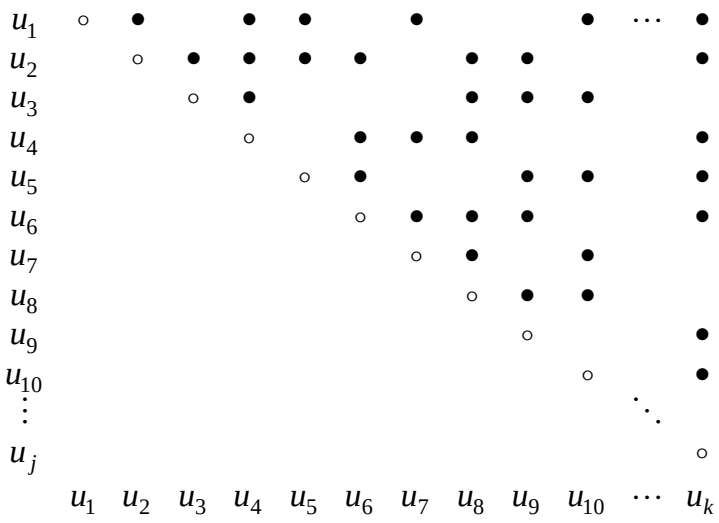


(Poznamenejme, že *l* zde značí počet kroků do restartu, nikoliv maximální počet vektorů, proti kterým reortogonalizujeme, ten je zde roven $l - 1$; srovnej s kapitolou 3.3.3.2.)

3.3.3.4 Parciální reortogonalizace

Při parciální reortogonalizaci, viz [15], provádíme reortogonalizaci vektoru u_k proti vektoru u_j jen tehdy, platí-li pro jejich skalární součin $|(u_j, u_k)| > \varepsilon$, $j = 1,..., k - 1$. Tedy je-li absolutní hodnota jejich skalárního součinu větší než předem dané kladné ε. Výhodou metody je, že můžeme ušetřit práci tím, že vektory s malými skalární součiny (tedy vektory již ortogonální) z výpočtu vyloučíme. Problémem ovšem je jak volit ε, může se stát, že všechny skalární součiny budou větší než námi zvolené ε a dostaneme v podstatě plnou ortogonalizaci (viz numerické experimenty v kapitole 4.2.4). Jinými slovy, při této strategii nejsme schopni omezit maximální cenu iterace. Následující schéma ilustruje možnou

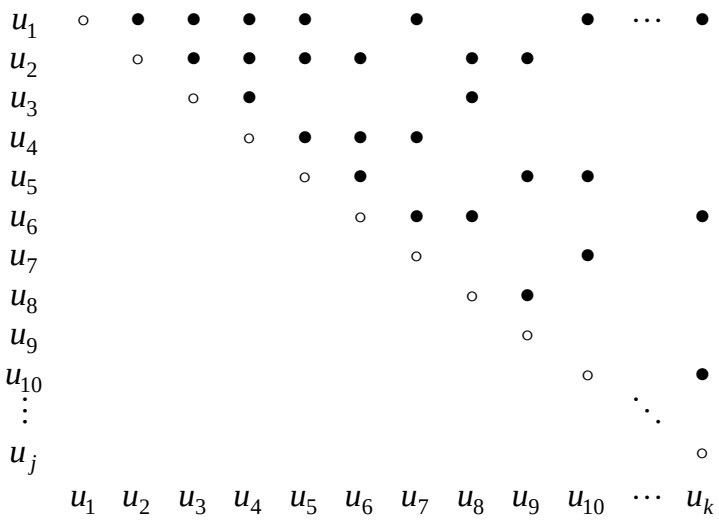
strukturu matice R (struktura může být zcela nahodilá):



3.3.3.5 Výběrová reortogonalizace

Motivováni snahou omezit cenu iterace můžeme reortogonalizaci provádět pouze proti l v absolutní hodnotě největším skalárním součinům $\left|\left(u_j, u_k\right)\right|$. Tento postup pracovně nazýváme „výběrovou reortogonalizací“. Tato strategie je reálně použitelná pouze pro ortogonalizaci pomocí klasického Gram-Schmidtova procesu, jelikož musíme nejprve znát všechny skalární součiny a pak z nich vybrat l největších. Narozdíl od všech předchozích, doposud diskutovaných strategií, které lze zřejmě realizovat pomocí CGS i MGS, včetně iteračního zpřesnění. Nevýhodou, i když ne velkou, oproti předchozí strategii je nutnost provedení výběru skalárních součinů, které použijeme. Velkou výhodou oproti předchozí strategii je snížení ceny iterace (zde nesmíme zapomenout na fakt, že v k-té iteraci musíme stejně jako u předchozí strategie vždy spočítat $k - 1$ skalárních součinů). Následující schéma ilustruje možnou strukturu matice R (struktura může být nahodilá, ale pro vektory u_j, $j > 3$,

musí být v každém sloupci právě tři symboly $\bullet$):

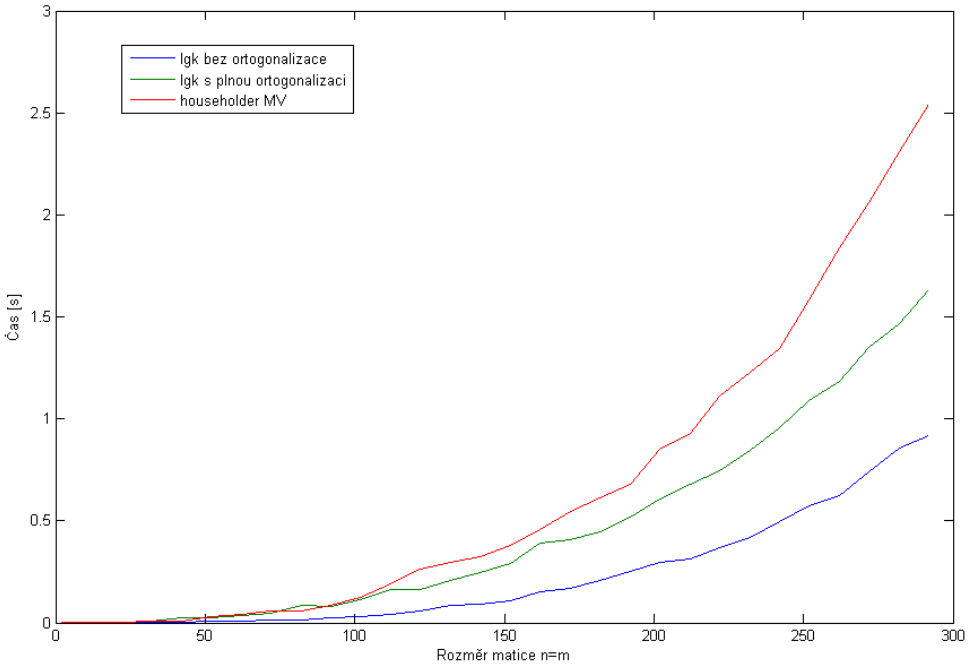


Poznamenejme ještě, že existují i další, složitější reortogonalizační strategie, které zde ovšem nebudeme zmiňovat, např. selektivní reortogonalizace viz [13]. Numerická stabilita bidiagonalizace je též studována a nové metody jejího výpočtu navrhovány např. v pracích [1, 16]. Za zmínku stojí i nedávné výsledky prezentované C. C. Paigem na 17. Householderově sympoziu v Zeuthen v roce 2008, [11]. Zde je diskutována stabilita bidiagonalizace, ve které reortogonalizujeme pouze jednu sadu vektorů, např. $\{u_j\}_{j=1}^{k}$, zatím co druhou sadu vektorů $\{v_j\}_{j=1}^{k}$ necháme nereortogonalizovanou.

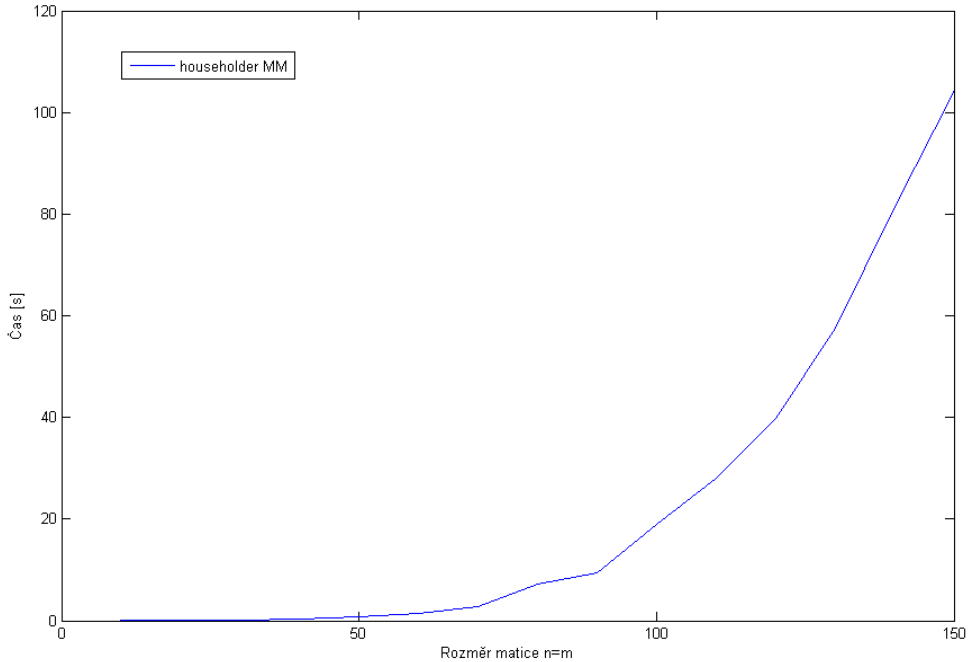
4. Numerické experimenty

Všechny následující testy jsou prováděny na matici SHAW, viz [7], není-li uvedeno jinak.

4.1 Porovnání výpočetních časů



Obr. 4.1: Výpočetní časy pro jednotlivé metody pro různé dimenze úlohy.



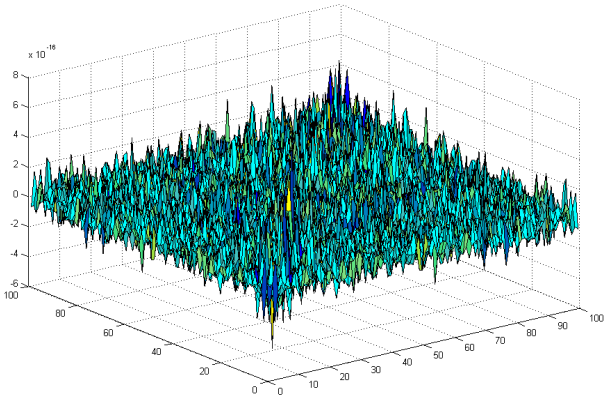
Obr. 4.2: Výpočetní časy pro Householderovu metodu MM (matrix-matrix product) pro různé dimenze úlohy.

Na grafu na obrázku 4.1 vidíme porovnání výpočetního času bidiagonalizace Householderovou metodou používající pouze součiny matic s vektory (matrix-vector product) a Golub-Kahanovy bidiagonalizace bez a s dvojnásobnou plnou reortogonalizací. Z grafu je patrné, že nejrychlejší je výpočet Golub-Kahanovým algoritmem bez reortogonalizace. Abychom ale dosáhli dobrých výsledků, musíme provést dvojnásobnou plnou reortogonalizaci, výpočet pak trvá zhruba dvojnásobnou dobu (poznamenejme, že na rychlost výpočtu Golub-Kahanova algoritmu nemá vliv to, zda použijeme CGS nebo MGS reortogonalizaci, počet aritmetických operací je identický). I tak je ale tento výpočet stále rychlejší než výpočet Householderovou metodou a navíc dává srovnatelné výsledky. Na obrázku 4.2 je doba výpočtu bidiagonalizace Householderovou metodou implementovanou tak, že konstruujeme přímo Householderovy matice a provádíme součin matic (matrix-matrix product). Výpočet bidiagonálních matic tímto postupem je velmi zdlouhavý, už pro matici o velikosti 150 trvá více než 100 s, výpočet tímto způsobem je reálně nepoužitelný.

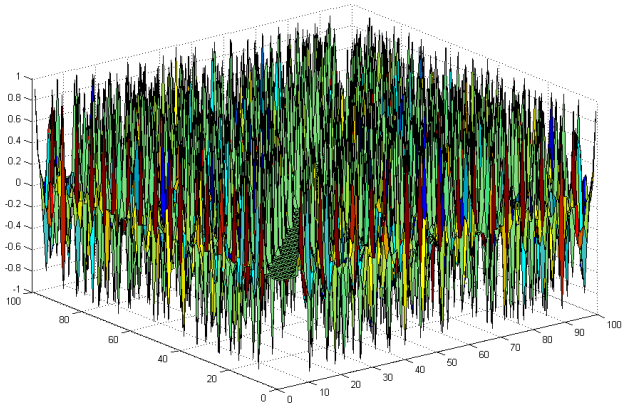
4.2 Test ortogonality matice *U*

Všechny následující testy jsou provedeny matici SHAW dimenze 100. Provádíme kontrolu ortogonality vektorů $u_1, \ldots, u_j$, sloupců matice *U*. Sloupce matice *U* jsou ortogonální, platí-li že součin $U^T U$ je roven jednotkové matici *I*. Následující grafy ilustrují „povrch“ matice $U^T U - I$, která by měla být teoreticky nulová. (Poznamenejme, že pro vektory v_j můžeme pozorovat podobné výsledky.)

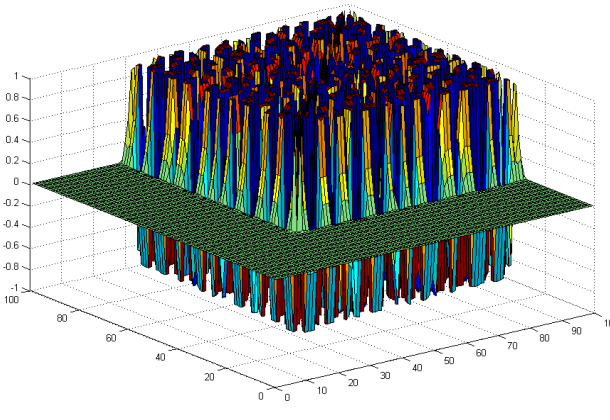
4.2.1 Porovnání základních metod



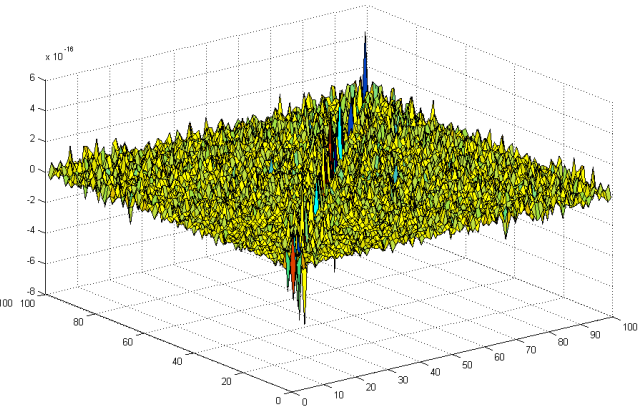
Obr. 4.3: Householderova bidiagonalizace.



Obr. 4.4: Golub-Kahanova bidiagonalizace bez reortogonalizace.



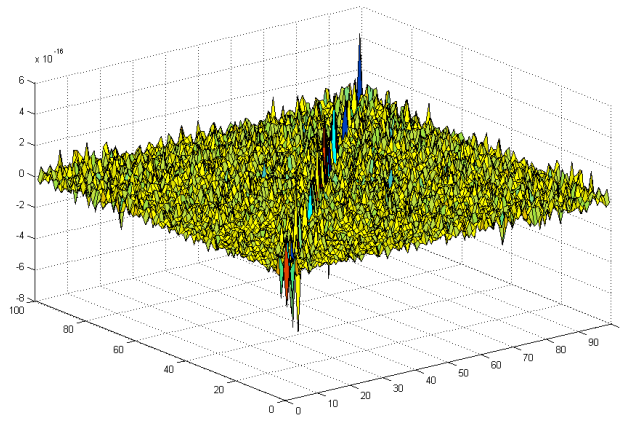
Obr. 4.5: Golub-Kahanova bidiagonalizace s jednonásobnou plnou CGS reortogonalizací.



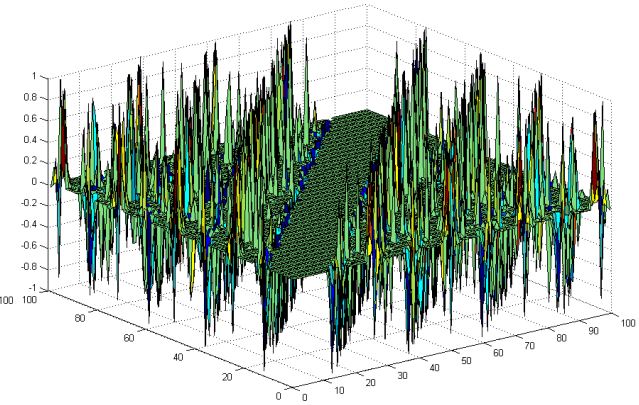
Obr. 4.6: Golub-Kahanova bidiagonalizace s dvojnásobnou plnou CGS reortogonalizací.

Na grafu na obrázku 4.3 je výsledek po Householderově bidiagonalizaci. Vidíme, že hodnoty se pohybují v řádu 10^{-16}, tedy na úrovni strojové přesnosti, což je výborný výsledek. Na dalších obrázcích jsou výsledky po Golub-Kahanově bidiagonalizaci. Na obrázku 4.4 je výpočet bez reortogonalizace, vidíme pouze několik ortogonálních vektorů kolem diagonály, jinak dochází k úplné ztrátě ortogonality. Provedeme-li plnou jednonásobnou reortogonalizaci (obrázek 4.5), je prvních zhruba 30 vektorů dokonale ortogonálních a poté opět dojde ke ztrátě ortogonality. Při provedení dvojnásobné plné reortogonalizace vidíme výsledek srovnatelný s Householderovým výpočtem, hodnoty se rovněž pohybují v řádu 10^{-16}.

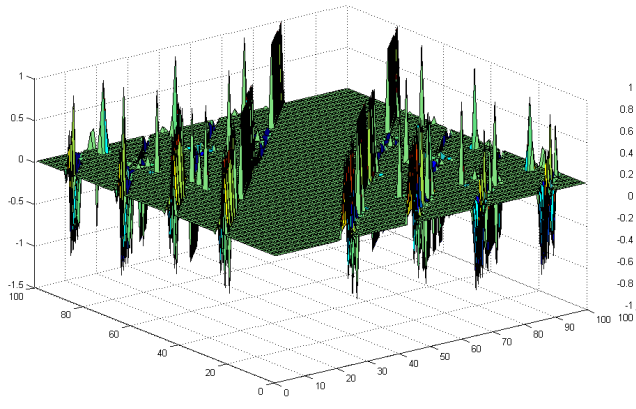
4.2.2 Porovnání různých pásových reortogonalizací



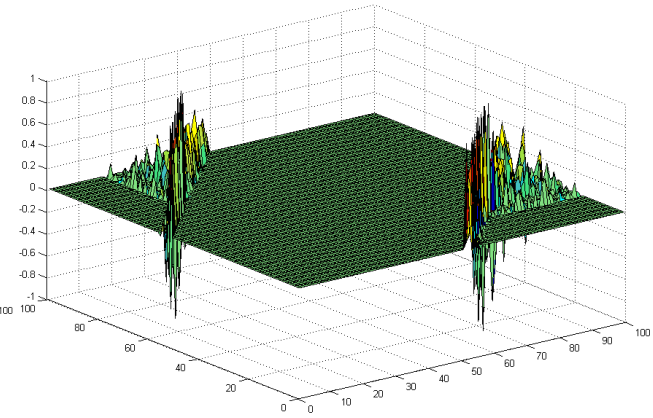
Obr. 4.7: Golub-Kahanova bidiagonalizace s dvojnásobnou plnou CGS reortogonalizací.



Obr. 4.8: Golub-Kahanova bidiagonalizace s pásovou CGS reortogonalizací proti10 vektorům.



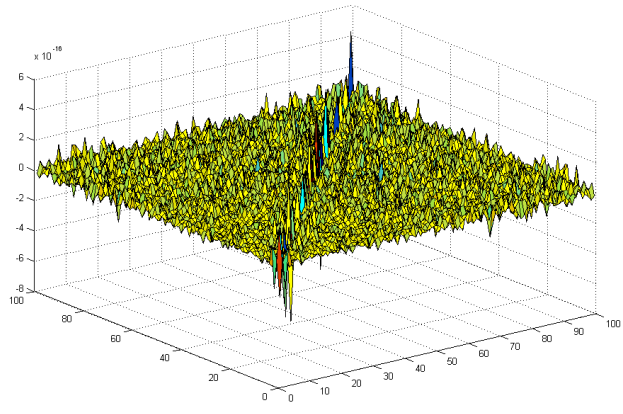
Obr. 4.9: Golub-Kahanova bidiagonalizace s pásovou CGS reortogonalizací proti 20 vektorům.



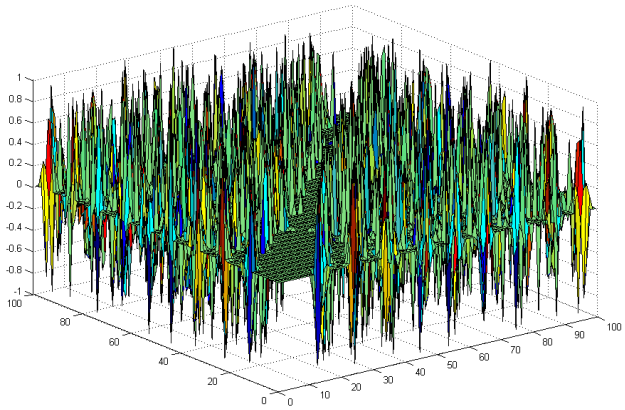
Obr. 4.10: Golub-Kahanova bidiagonalizace s pásovou CGS reortogonalizací proti 50 vektorům.

Grafy na obrázcích 4.7–4.10 znázorňují ztrátu ortogonality při výpočtu Golub-Kahanovy bidiagonalizace s pásovou reortogonalizací v porovnání s plnou dvojnásobnou reortogonalizací. Při reortogonalizaci proti l vektorům, je matice U do vzdálenosti l vektorů kolem diagonály dokonale ortogonální, protože dosud v podstatě reortogonalizujeme proti všem předchozím vektorům. Poté se ortogonalita ztratí, během dalších l kroků se opět trochu zlepší, u větších l dosáhne opět poměrně dobré ortogonality.

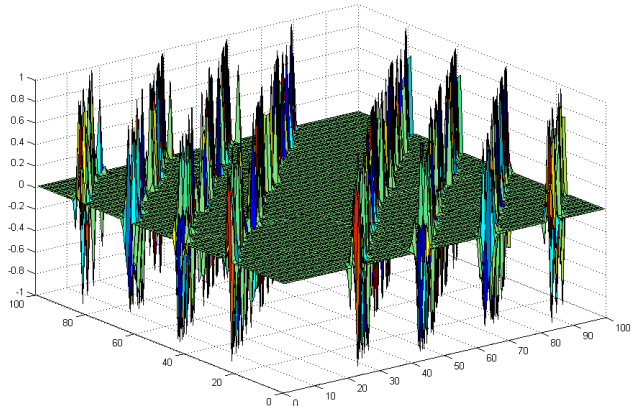
4.2.3 Porovnání různých restartovaných reortogonalizací



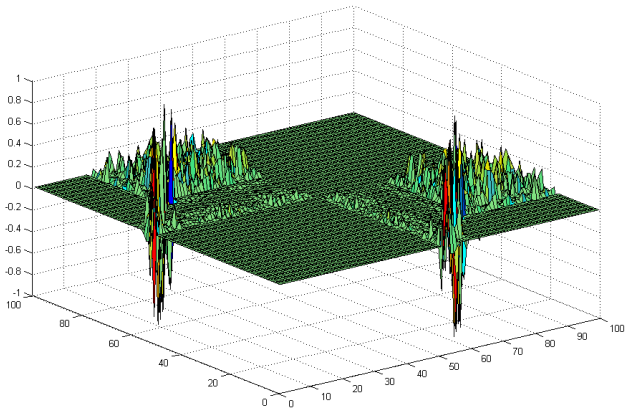
Obr. 4.11: Golub-Kahanova bidiagonalizace s dvojnásobnou plnou CGS reortogonalizací.



Obr. 4.12: Golub-Kahanova bidiagonalizace s restartovanou CGS reortogonalizací, restart po 10 krocích.



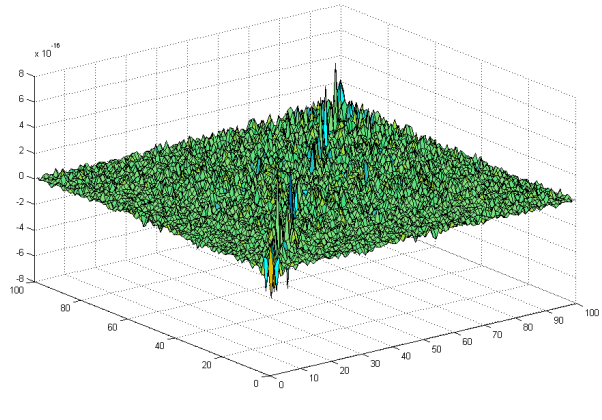
Obr. 4.13: Golub-Kahanova bidiagonalizace s restartovanou CGS reortogonalizací, restart po 20 krocích.



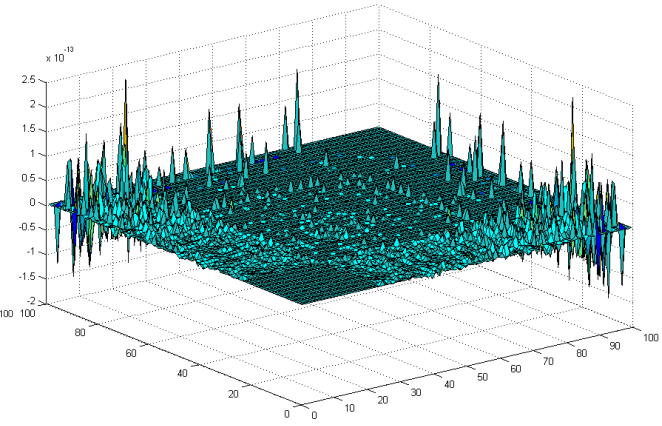
Obr. 4.14: Golub-Kahanova bidiagonalizace s restartovanou CGS reortogonalizací, restart po 50 krocích.

Grafy na obrázcích 4.11–4.14 porovnávají ztrátu ortogonality po výpočtu Golub-Kahanovy bidiagonalizace s restartovanou reortogonalizací v porovnání s plnou dvojnásobnou reortogonalizací. Grafy jsou podobné jako u pásové reortogonalizace s tím rozdílem, že kolem diagonály není pás ortogonálních vektorů, ale vznikají tam čtverce (nejlépe to je vidět na obrázku 4.14), což je způsobeno restartováním reortogonalizace. Po l krocích dojde k restartu reortogonalizace, tím dojde ke ztrátě ortogonality vektorů, která se opět postupně zlepšuje až do dalšího restartu.

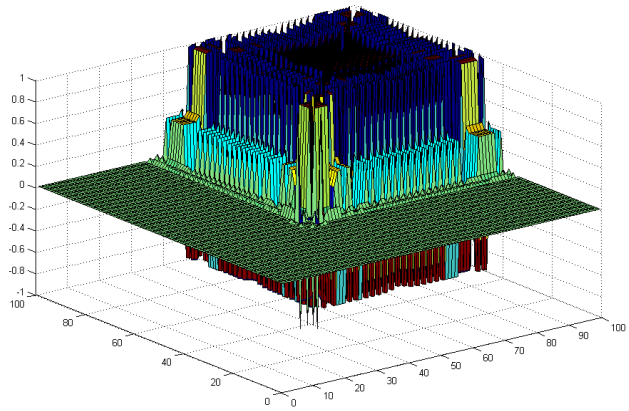
4.2.4 Porovnání různých parciálních reortogonalizací



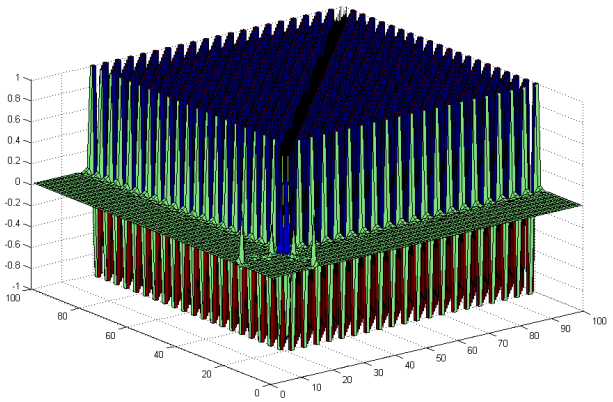
Obr. 4.15: Golub-Kahanova bidiagonalizace s parciální CGS reortogonalizací, $\varepsilon = 10^{-40}$.



Obr. 4.16: Golub-Kahanova bidiagonalizace s parciální CGS reortogonalizací, $\varepsilon = 10^{-30}$.



Obr. 4.17: Golub-Kahanova bidiagonalizace s parciální CGS reortogonalizací, $\varepsilon = 10^{-20}$.



Obr. 4.18: Golub-Kahanova bidiagonalizace s parciální CGS reortogonalizací, $\varepsilon = 10^{-10}$.

Grafy na obrázcích 4.15–4.18 porovnávají parciální reortogonalizace pro různé ε. Pro ε menší než 10^{-40} jde již prakticky o plnou reortogonalizaci, hodnoty se pohybují v řádu 10^{-16}, tedy na úrovni strojové přesnosti. Pro ε v řádech okolo 10^{-30} je ortogonalita vektorů ještě velmi dobrá, hodnoty jsou v řádu 10^{-13}. Pro ε větší než 10^{-20} je několik prvních vektorů ortogonálních, dále se již ortogonalita úplně ztrácí. Pro srovnání kvality ortogonalizace v závislosti na velikosti ε, viz tabulka 4.1, ve které je zaznamenán počet vektorů použitých k reortogonalizaci pro různá ε.

ε	jednonásobná reortogonalizace		dvojnásobná reortogonalizace	
	počet vektorů, proti kterým ortogonalizujeme	$\left\|\left(U^T U - I\right)\right\|_2$	počet vektorů, proti kterým ortogonalizujeme	$\left\|\left(U^T U - I\right)\right\|_2$
10^{-40}	4950	61,0486	9882	$9{,}1681*10^{-16}$
10^{-35}	4950	61,0486	9878	$8{,}3485*10^{-16}$
10^{-30}	4950	61,0486	8117	$4{,}2153*10^{-13}$
10^{-25}	4950	61,0486	8004	$1{,}7523*10^{-6}$
10^{-20}	4949	59,4565	8656	53,5468
10^{-15}	2419	46,5596	7410	63,1746
10^{-14}	3087	42,6313	6024	44,2842
10^{-13}	3333	61,4119	6538	59,2544
10^{-12}	3045	26,8552	401	38,3179
10^{-11}	756	73,4045	7040	33,9670
10^{-10}	1394	84,9974	6380	80,9978
10^{-9}	1867	75,6449	3190	78,6932
10^{-8}	1224	86,0100	5795	69,8308
10^{-7}	1469	84,7420	6992	54,6580
10^{-6}	1888	88,7654	3386	87,1373
10^{-5}	2812	71,3611	7327	65,0647
10^{-4}	2717	42,1967	7226	62,0071
10^{-3}	896	34,8693	1456	18,2769
10^{-2}	1447	30,8741	4762	13,9418
10^{-1}	1299	23,7160	1399	11,7313
10^{-0}	108	18,9250	108	18,9250

Tab. 4.1: Počet (levých) vektorů z Golub-Kahanovy bidiagonalizace, proti kterým ortogonalizujeme a ztráta ortogonality, oboje při použití parciální CGS reortogonalizace pro různé hodnoty ε.

Z tabulky 4.1 opět vidíme, že pro velmi malé $\varepsilon \sim 10^{-40}$ provádíme v podstatě úplnou reortogonalizaci (tj. 4950, resp. 9900 ortogonalizací při jednonásobné, resp. dvojnásobné reortogonalizaci).

Zajímavý jev pozorovaný v prezentovaných datech je, že *závislost počtu ortogonalizací na ε není monotónní*, jak bychom mohli očekávat. Může se tedy stát, že při zvýšení prahu (a tedy snížení citlivosti) se počet ortogonalizací (a tedy výpočetní cena) zvýší, viz např. $\varepsilon = 10^{-6}$ a 10^{-5}. Na tento jev lze nahlížet tak, že reortogonalizací proti „nevhodnému“ vektoru si můžeme „přidělat práci“ v dalších výpočtech. Otázka, které vektory jsou tyto „nevhodné“ a zda jsou opravdu nevhodné nebo zda je právě proti takovým vektorům nutné ortogonalizovat by mohla být předmětem dalšího studia. Vidíme, že ztráta ortogonality

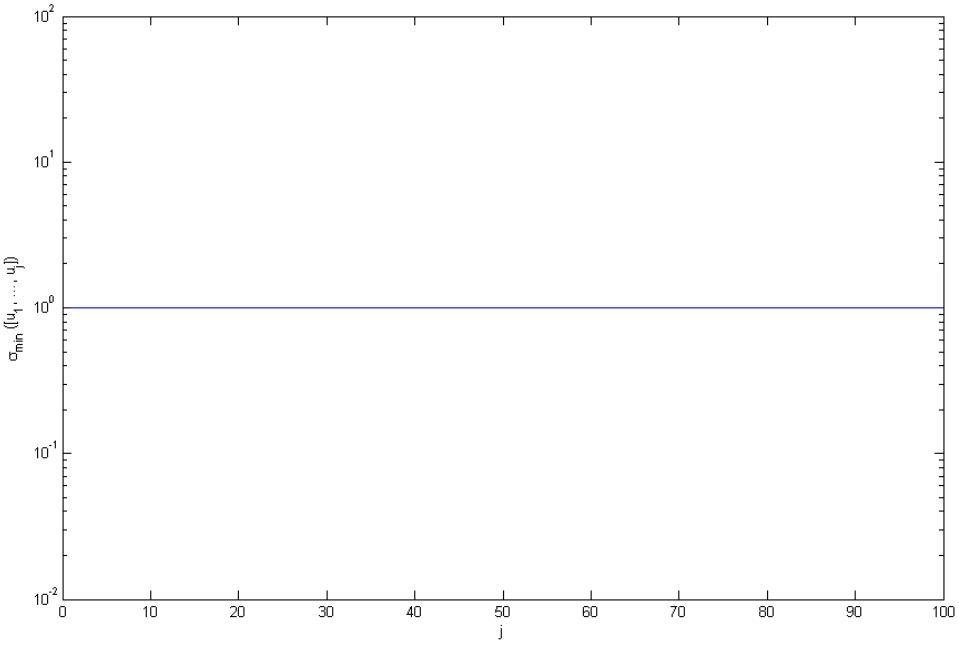
měřená normou $\left(U^T U - I\right)$ je prakticky vždy, s výjimkou velmi nízkých prahů při dvojité reortogonalizaci, úplná. Podobné výsledky můžeme sledovat i pro vektory v_j.

4.3 Test ortogonality matice *U* užitím singulárního rozkladu

Test ortogonality můžeme provádět i výpočtem nejmenších singulárních čísel matice $[u_1, \ldots, u_j]$. K tomuto účelu použijeme algoritmus

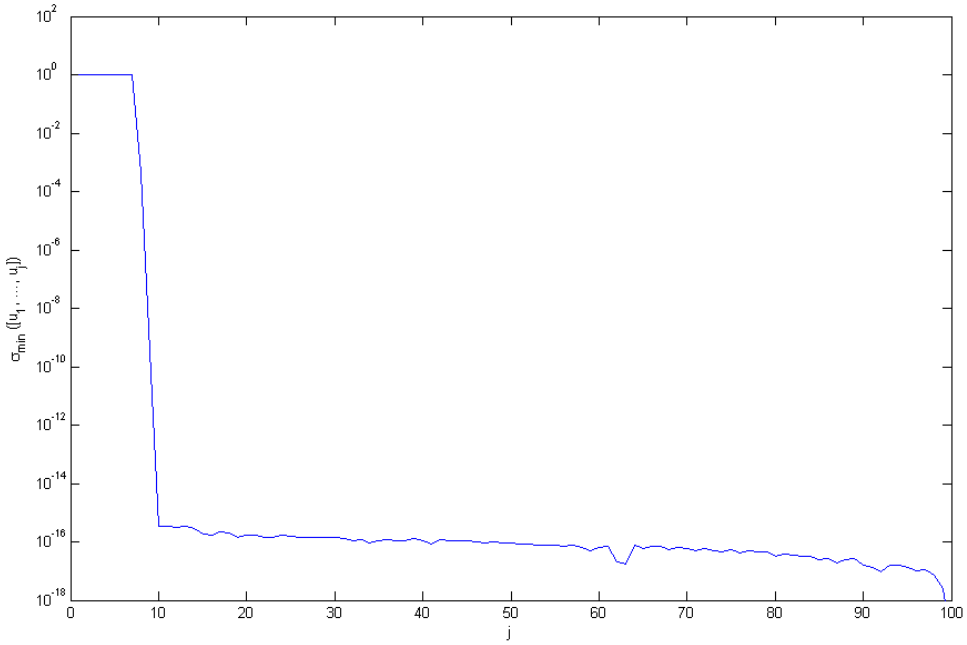
```
00  ...
01  for  j = 1 : k
02       data(j) = min(svd(U(:,1:j)))
03  end;
04  semilogy(data)
05  ...
```

Příkazem svd(*A*) MATLAB vypočítá všechna singulární čísla matice *A*, z nich pak vybereme nejmenší. Nejmenším singulárním číslem můžeme měřit ztrátu ortogonality v matici $[u_1, \ldots, u_j]$. Správně by pro matici s ortogonálními sloupci měla být všechna singulární čísla rovna jedné.



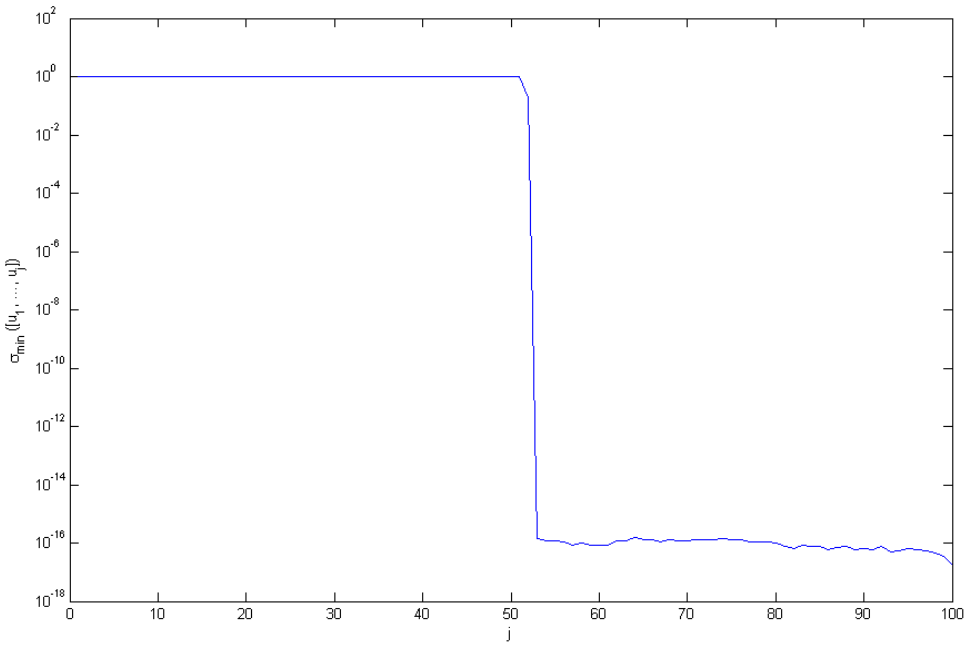
Obr. 4.19: Householderova bidiagonalizace, Golub-Kahanova bidiagonalizace s dvojnásobnou plnou reortogonalizací (CGS i MGS).

Dokonalá ortogonalita, singulární čísla jsou vždy rovna jedné. Takovýto výsledek dává výpočet Householderovou bidiagonalizací a Golub-Kahanovou bidiagonalizací s dvojnásobnou plnou reortogonalizací (CGS i MGS).



Obr. 4.20: Golub-Kahanova bidiagonalizace bez reortogonalizace.

Golub-Kahanova bidiagonalizace bez reortogonalizace. Z grafu vidíme, že ortogonalita drží přibližně pro prvních 10 prvků, poté se úplně ztratí, kdy hodnoty singulárních čísel klesnou až k řádu 10^{-16}, tedy na úroveň relativní strojové přesnosti.



Obr. 4.21: Golub-Kahanova bidiagonalizace s pásovou reortogonalizací proti 50 vektorům nebo restartovanou reortogonalizací s restartem po 50 krocích.

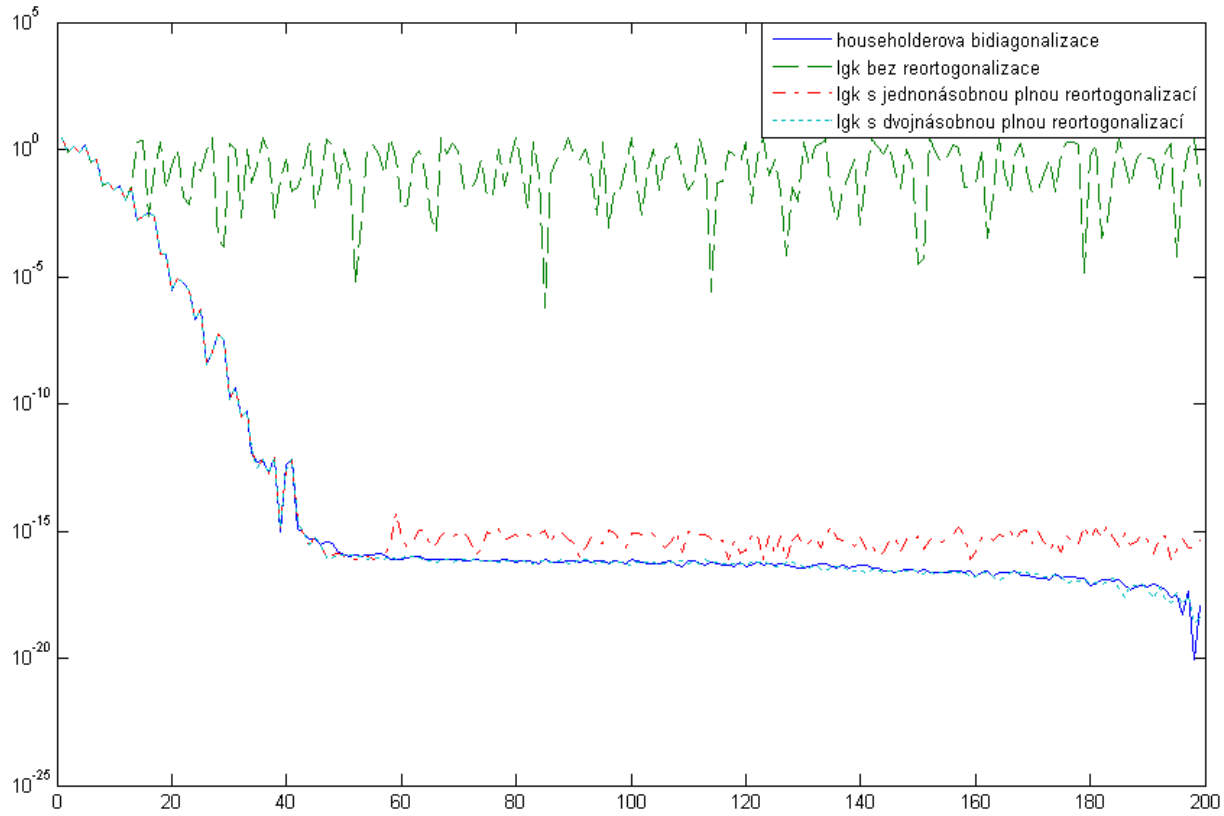
Průběh pásové reortogonalizace proti předchozím 50 vektorům. Totožný průběh má i restartovaná reortogonalizace s restartem po 50 krocích. V obou případech se ortogonalita ztratí po prvních 50 krocích, poté se vektory stanou numericky lineárně závislé.

4.4 Srovnání normalizačních koeficientů – nenulových prvků bidiagonální matice

V následujících experimentech používáme matici SHAW dimenze 100. Pomocí MATLABovských příkazů diag(L) a diag(L, -1) získáme z bidiagonální matice L, obě diagonály. Z nich pak vytvoříme následujícím způsobem jeden vektor

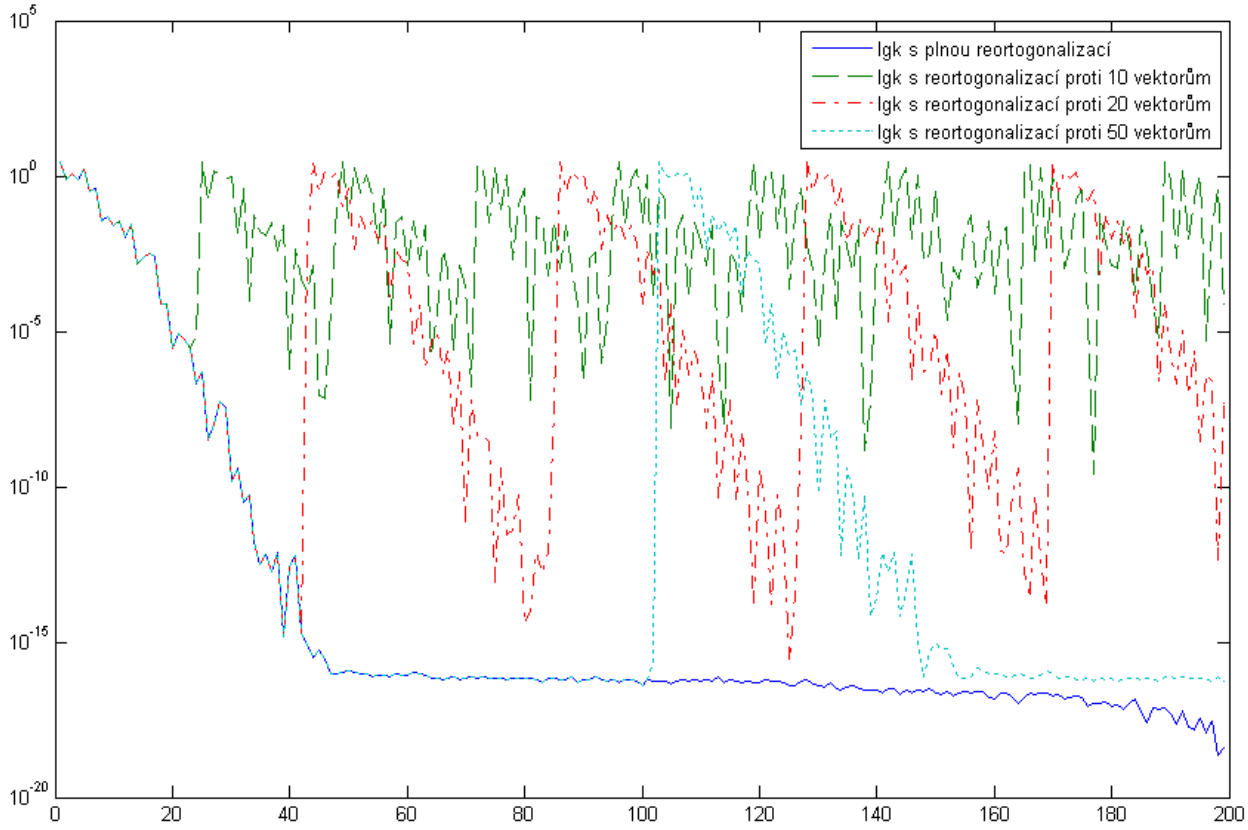
$$[\beta_1, \alpha_1, \beta_2, \alpha_2, \ldots, \beta_{100}, \alpha_{100}].$$

Takto to uděláme pro různé způsoby výpočtů bidiagonalizace. Získáme vektory diagonál a ty pak vyneseme do grafů a porovnáme diagonální koeficienty.



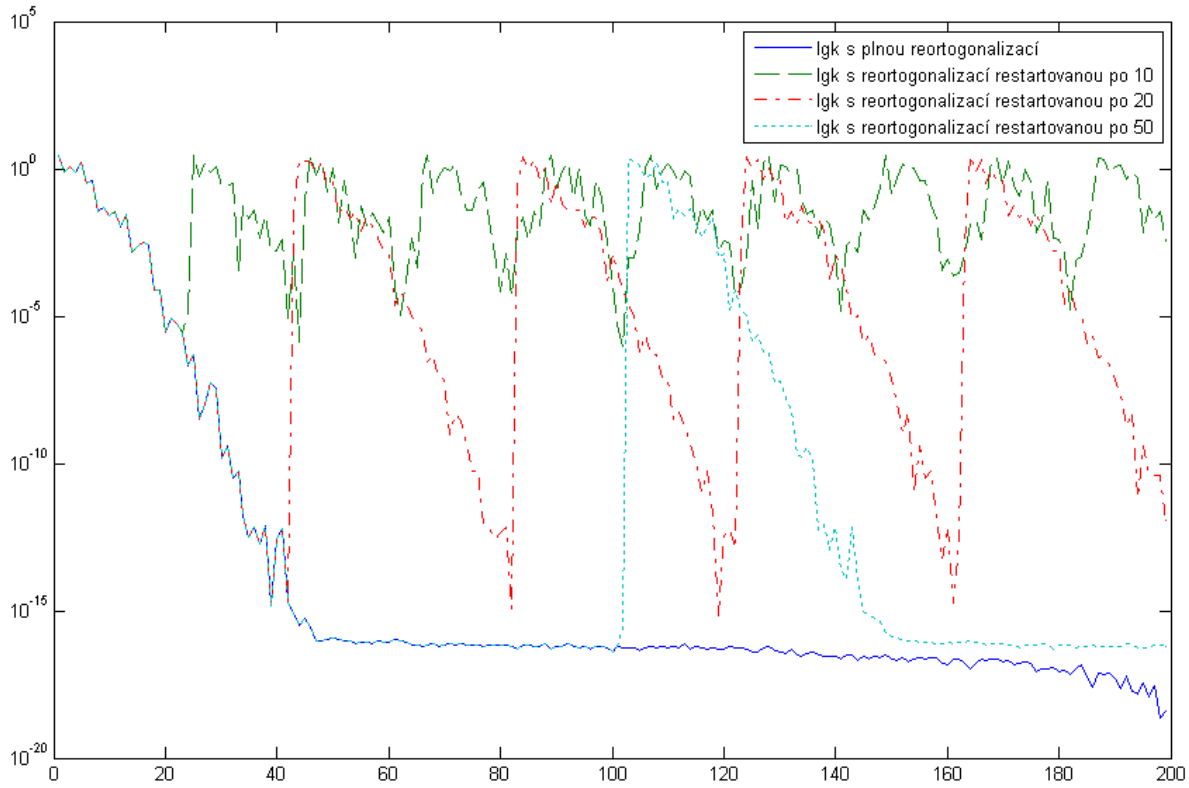
Obr. 4.22: Průběhy diagonálních koeficientů bidiagonálních matic získaných různými metodami.

Z grafu vidíme, že diagonální koeficienty získané Householderovou bidiagonalizací a Golub-Kahanovou bidiagonalizací s dvojnásobnou plnou CGS reortogonalizací jsou prakticky totožné. Výsledky získané Golub-Kahanovou bidiagonalizací s jednonásobnou plnou CGS reortogonalizací se od „přesných“ (rozuměj nejpřesnější jaké dokážeme dosáhnout v dané aritmetice) hodnot příliš neliší. Naopak výsledky získané Golub-Kahanovou bidiagonalizací bez reortogonalizace jsou velmi špatné.



Obr. 4.23: Průběhy diagonálních koeficientů bidiagonálních matic získaných Golub-Kahanovou bidiagonalizací s pásovou reortogonalizací.

Zde vidíme srovnání Golub-Kahanovy bidiagonalizace s plnou reortogonalizací a pásovou reortogonalizací proti l předchozím vektorům. Vždy do l kroků je průběh prvků po pásové reortogonalizaci totožný s plnou reortogonalizací. Po každých l krocích se hodnota prvků diagonály výrazně zhorší a poté se pozvolna začne přibližovat „přesným“ hodnotám. Z grafu je vidět, že nejhoršího výsledku dosahuje ortogonalizace proti 10 vektorům, která se v žádné fázi dobrým výsledkům nepřiblíží.



Obr. 4.24: Průběhy diagonálních koeficientů bidiagonálních matic získaných Golub-Kahanovou bidiagonalizací s restartovanou reortogonalizací.

Zde jsou průběhy diagonálních prvků Golub-Kahanovy bidiagonalizace s restartovanou ortogonalizací rovněž ve srovnání s plnou dvojnásobnou reortogonalizací. Graf je velice podobný průběhům pásové reortogonalizace, vždy při restartu dojde k zhoršení výsledků, které se pak pozvolna blíží ideálu. Opět platí, že restart po 10 krocích je nedostačující, protože se výsledky vůbec nepřiblíží „přesným“ hodnotám, výsledky jsou poměrně špatné.

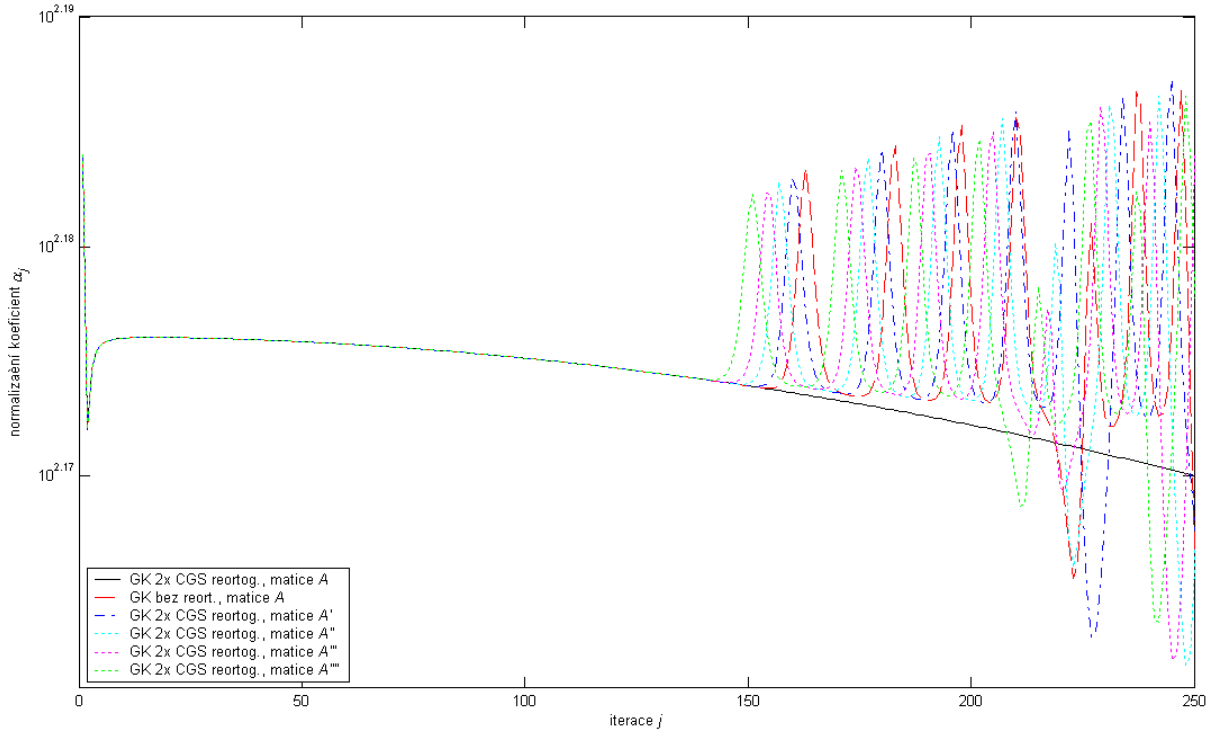
4.5 Praktická použitelnost prostého Golub-Kahanova algoritmu

V předchozích kapitolách jsme ukázali, že prostá Golub-Kahanova bidiagonalizace (tedy bez jakékoliv reortogonalizace) dává velmi špatné výsledy. Dochází k úplné ztrátě ortogonality mezi spočtenými vektory a prvky bidiagonální matice prakticky nemají ani jednu platnou cifru. Víme však, že pro rozsáhlé úlohy si musíme vystačit pouze s algoritmem bez reortogonalizace, ta by byla pro opravdu velké matice neúnosně drahá a prakticky neproveditelná. Na následujícím příkladu se pokusíme naznačit, jakým způsobem se můžeme na prostou Golub-Kahanovu bidiagonalizaci dívat, jak nahlížet na výsledky, které získáme její aplikací.

V tomto příkladu budeme uvažovat diagonální matici s diagonálními prvky ekvidistantně rozloženými v intervalu [100,200] s krokem 0.2

A = diag(100.0,100.2,100.4,...,199.8,200.0) $\in$ $R^{501x501}$

a normalizovaný startovací vektor $s = 501^{-1/2}[1,...,1]^T$. Na následujících obrázcích 4.25–4.26 jsou grafy vykreslující koeficienty α_j, respektive β_{j+1} pro j = 1,..., 250 spočtené Golub-Kahanovou bidiagonalizaci s dvojnásobnou úplnou CGS reortogonalizací (plná černá čára) a bez reortogonalizace (červená čárkovaná čára).



Obr. 4.25: Koeficienty α_j pro j = 1,..., 250.

Dále jsou v grafech na obrázcích 4.25–4.26 vykresleny normalizační koeficienty α_j, β_j pro modifikované úlohy s maticemi

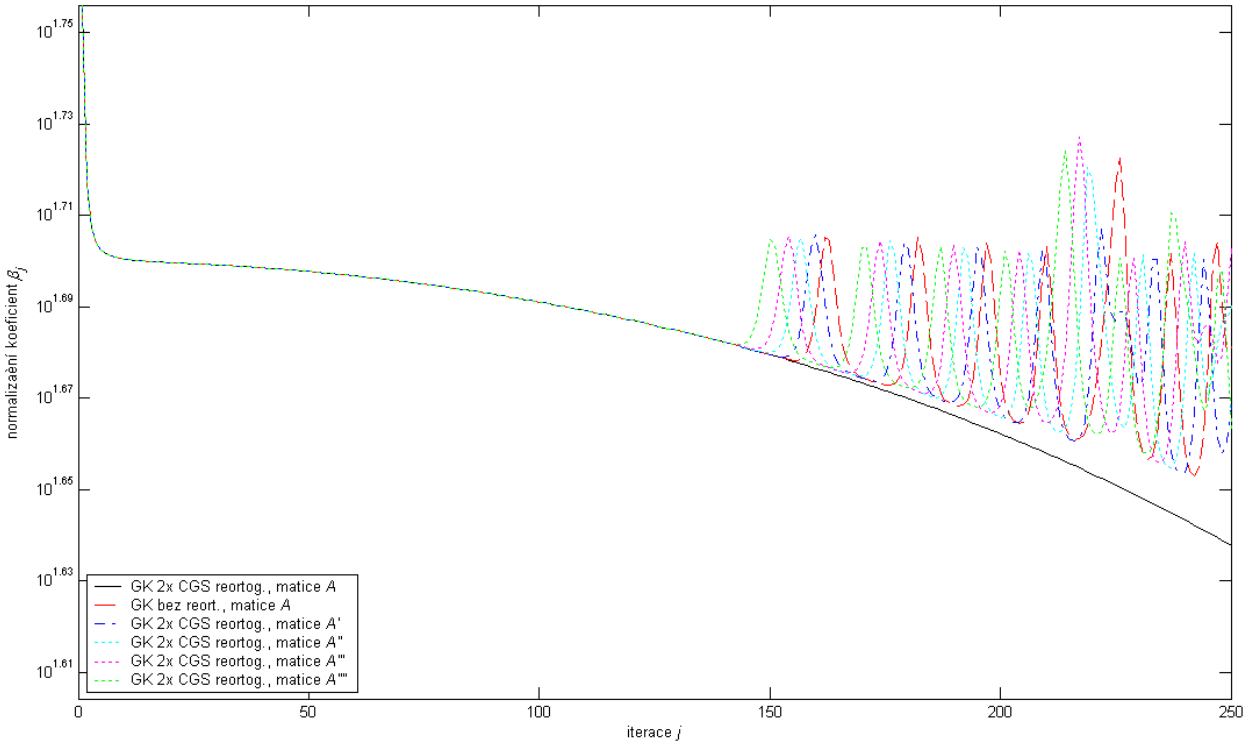
$A' = \text{diag}(\ldots, a_{ii}(1 - \varepsilon_M / 2), a_{ii}(1 + \varepsilon_M / 2), \ldots)$,

$A'' = \text{diag}(\ldots, a_{ii}(1 - \varepsilon_M), a_{ii}(1 + \varepsilon_M), \ldots)$,

$A''' = \text{diag}(\ldots, a_{ii}(1 - 4\varepsilon_M), a_{ii}(1 + 4\varepsilon_M), \ldots)$,

$A'''' = \text{diag}(\ldots, a_{ii}(1 - 16\varepsilon_M), a_{ii}(1 + 16\varepsilon_M), \ldots)$,

dvojnásobné dimenze (tj. 1002) oproti původní matici A a normalizovaným startovacím vektorem $s = 1002^{-1/2}[1,...,1]^T$; zde a_{ii} značí i-tý diagonální prvek původní matice A a ε_M tak zvanou *relativní strojovou přesnost* (*machine precision*).



Obr. 4.26: Koeficienty β_j pro j = 1,..., 250.

Relativní strojová přesnost je nejmenší kladné číslo takové, že jeho součet s jedničkou je v aritmetice s plovoucí řádovou čárkou (aritmetice s konečnou přesností) větší než jedna. Toto číslo je důležitou charakteristikou každé aritmetiky s plovoucí řádovou čárkou, pro standardně používaný formát double (použitý i v našich výpočtech) je $\varepsilon_M = 2^{-52} \approx 2{,}2204 * 10^{-16}$. Polovina strojové přesnosti $\delta \equiv \varepsilon_M / 2$ se obvykle nazývá *zaokrouhlovací jednotka*.

Vidíme tedy, že matice A', A'', A''', A'''' jsme obdrželi tak, že jsme zdvojili diagonální prvky a_{ii} původní matice A, přičemž jsme vždy jednu kopii zmenšili o hodnotu $a_{ii} = \eta / 2$ a druhou kopii o stejnou hodnotu zvětšili, kopie se tudíž navzájem liší o hodnotu $a_{ii} = \eta$, kde postupně $\eta = \varepsilon_M, 2\varepsilon_M, 8\varepsilon_M$ a $32\varepsilon_M$. Tyto modifikované úlohy řešíme Golub-Kahanovou bidiagonalizaci s dvojnásobnou úplnou CGS reortogonalizací viz grafy na obrázcích 4.25–4.26 (vykresleno postupně čerchovanou tmavomodrou čarou, tečkovanou světlemodrou, tečkovanou fialovou a tečkovanou zelenou čarou).

Vidíme, že prostá Golub-Kahanova bidiagonalizace bez jakékoliv reortogonalizace vykazuje v tomto příkladu velmi podobné chování jako „přesná“ (rozuměj nejpřesnější jaké dokážeme dosáhnout v dané aritmetice) bidiagonalizace spočtená s dvojnásobnou úplnou

reortogonalizací ovšem aplikovaná na modifikovaný, přibližně zdvojený (ve výše specifikovaném smyslu) problém. Viz také např. [10].

Přesné výsledky získané z modifikovaných problémů tak můžeme interpretovat jako nepřesné výsledky získané prostou Golub-Kahanovou bidiagonalizací aplikovanou na problém původní ovšem v aritmetice s plovoucí řádovou čárkou mající zaokrouhlovací jednotku rovnou postupně jednotlivým η.

Naopak, nepřesné výsledky získané prostou bidiagonalizací původního problému můžeme interpretovat jako přesné výsledky aplikované ovšem na modifikovaný problém dvojnásobné dimenze takový, že každý diagonální prvek a_{ii} je zdvojen, přičemž jedna jeho kopie je o hodnotu $a_{ii}\delta / 2$ zmenšena a druhá zvětšena. Interval mezi oběma kopiemi je tedy úměrný velikosti daného diagonálního prvku a zaokrouhlovací jednotce použité aritmetiky.

Připomeňme, že zde jsme pro jednoduchost a přehlednost uvažovali pouze diagonální problém. Pro problémy s maticí v obecném tvaru je situace přirozeně složitější. Stejně tak jsme zde vůbec nediskutovali jak vypadají vektory, které algoritmus produkuje. Sledovali jsme pouze normalizační koeficienty. I přes to je tento výsledek nesmírně důležitý neboť může ospravedlnit použití prosté (nereortogonalizované) Golub-Kahanovy bidiagonalizace.

5. Návod k užívání programů

5.1 Householderova bidiagonalizace

Hlavní program Householderovy bidiagonalizace spustíme se zadanými parametry pro výpočet.

Spuštění:

[U, L, V] = hh_bidiag(A, s, k, p, ‘typ’, qrf, vp),

význam výstupních parametrů:

L – bidiagonální matice,

U, V – pomocné matice,

význam vstupních parametrů:

A – zadaná matice, kterou chceme zbidiagonalizovat,

s – startovací vektor výpočtů,

k – počet iterací,

p – vypočítá navíc jestě jeden půlkrok bidiagonalizace (true/false),

‘typ’ – způsob výpočtu:

- ‘MM’ – tvorba celých Householderových matic (MM od matrix-matrix product),
- ‘MV’ – využívá zjednodušený algoritmus (nepočítá s celými Householderovými maticemi, MV od matrix-vector product),

qrf – použití QR rozkladu – výběr mezi LQ nebo QR:

- 0 – rozklad se neprovede,
- 1 – QR rozklad,
- 2 – LQ rozklad,
- (–1) – program vybere,

vp – zadání počtu platných cifer pro výpočet ve vysoké přesnosti (ten je realizovaný aritmetikou vpa – *various precision arithmetic*, která je standardní součástí MATLABu).

Seznam programů, které se spouštějí hlavním programem nebo je lze spustit i samostatně s příslušnými parametry:

[U, L, V] = hh_bidiag_MM(A, s, k, p, qrf),

[U, L, V] = hh_bidiag_MV(A, s, k, p, qrf).

Pomocné programy samostatně nespustitelné:

aux_qr.m (realizuje QR případně LQ rozklad matice *A*),

aux_hholder.m (produkuje Householderovy matice, resp. vektory, pomocí kterých je konstruuje).

5.2 Golub-Kahanova bidiagonalizace

Spustí se hlavní program Golub-Kahanova bidiagonalizace s reortogonalizací a zadají se parametry pro výpočet.

Spuštění:

[U, L, V] = lgk_bidiag(A, s, k, p, ‘typ’, ‘druh’, [param]),

význam výstupních parametrů:

L – bidiagonální matice,

U, V – matice pomocných vektorů,

význam vstupních parametrů:

A – zadaná matice, kterou chceme zbidiagonalizovat,

s – startovací vektor výpočtů,

k – počet iterací,

p – vypočítá navíc jestě jeden půlkrok bidiagonalizace (true/false),

‘typ’– typ reortogonalizace - klasický/modifikovaný Gram-Schmidt (‘CGS’/‘MGS’),

‘druh’ – strategie reortogonalizace – pásová, restartovaná, parciální (‘band’/‘rest’/‘parc’),

[param] – vektor parametrů, se liší pro různé strategie:

- Strategie ‘band’ – [reort, refin, vp]:

  reort – proti kolika vektorům reortogonalizujeme:

  - 0 – reortogonalizace se neprovede,
  - (–1) – proti všem,
  - *l* – proti *l* předchozím vektorům,

  refin – vícenásobná reortogonalizace:

  - 0 – neprovede se,
  - *t* – provede se *t* krát,

vp – zadání počtu platných cifer pro výpočet ve vysoké přesnosti (ten je realizovaný aritmetikou vpa – *various precision arithmetic*, která je standardní součástí MATLABu).

- Strategie ‘rest’ – [rest, refin, vp]:
  rest – po kolika krocích se ortogonalizace restartuje,
  refin – vícenásobná reortogonalizace:
  - 0 – neprovede se,
  - *t* – provede se *t* krát,

  vp – zadání počtu platných cifer pro výpočet ve vysoké přesnosti.

- Strategie ‘parc’ – [epsilon, refin, vp]:
  epsilon – kladné číslo, se kterým se porovnává skalární součin a rozhoduje se, zda se daný vektor do ortogonalizace použije nebo ne,
  refin – vícenásobná reortogonalizace :
  - 0 – neprovede se,
  - *t* – provede se *t* krát,

  vp – zadání počtu platných cifer pro výpočet ve vysoké přesnosti.

Seznam programů, které se spouštějí hlavním programem nebo je lze spustit i samostatně s příslušnými parametry:

[U, L, V] = lgk_bidiag_CGS(A, s, k, p, reort, refin),
[U, L, V] = lgk_bidiag_MGS(A, s, k, p, reort, refin),
[U, L, V] = lgk_bidiag_CGS_restart_ort(A, s, k, p, rest, refin),
[U, L, V] = lgk_bidiag_MGS_restart_ort(A, s, k, p, rest, refin),
[U, L, V] = lgk_bidiag_CGS_parc_ort(A, s, k, p, epsilon, refin),
[U, L, V] = lgk_bidiag_MGS_parc_ort(A, s, k, p, epsilon, refin),

první dvě metody realizují pásovou reortogonalizaci, u ostatních je metoda zřejmá z názvu. (Další ortogonalizační strategie dosud nejsou implementovány.)

Závěr

Na sérii testů a vhodně vybrané matici se podařilo ověřit některé známé vlastnosti algoritmů pro transformaci matice na horní respektive dolní bidiagonální tvar. Podařilo se ukázat, že pro výpočet dobrých ortogonálních bází je nezbytné použít buď bidiagonalizaci postavenou na Householderových maticích, nebo Golub-Kahanův iterační algoritmus ovšem s dvojitou úplnou reortogonalizací; výsledky jsou vizuálně shodné (nebo velmi podobné) pro reortogonalizaci realizovanou pomocí klasického (CGS) i modifikovaného (MGS) Gram-Schmidtova procesu. Použijeme-li neúplnou reortogonalizaci (pásovou, restartovanou, nebo parciální s velkou prahovou hodnotou) a nebo použijeme-li úplnou reortogonalizaci ale pouze jednonásobnou, může být ztráta ortogonality mezi spočtenými vektory velmi významná. Poznamenejme ještě, že ne vždy je ztráta ortogonality tak markantní jako na prezentovaných obrázcích, zde jsme velmi špatných výsledků algoritmů dosáhli tím, že jsme pracovali s velmi obtížnou maticí.

Stejně tak jako ztrátu ortogonality se podařilo ověřit, že při nedostatečné reortogonalizaci se spočtené normalizační koeficienty (prvky bidiagonální matice) velmi liší od hodnot spočtených přesnější metodou, dokonce nemusí mít ani jednu platnou cifru. Přesto se však u reálných, velmi rozsáhlých úloh musíme spokojit právě jen s prostou Golub-Kahanovou bidiagonalizací bez jakékoliv následné reortogonalizace spočtených vektorů. Navzdory špatnému chování takového algoritmu (úplná ztráta ortogonality vektorů, úplná ztráta všech platných cifer normalizačních koeficientů, tj. prvků bidiagonální matice) lze ukázat, že se výpočet realizovaný tímto algoritmem chová v jistém smyslu rozumně (dochází k jistému zpožďování ve výpočtu; např. některé vektory se začnou po čase ve výpočtu objevovat ve více kopiích, a pod.). Toto chování částečně ilustruje obrázek 4.25. Analýza tohoto chování však nebyla předmětem této práce.

Podařilo se implementovat základní algoritmy pro výpočet bidiagonalizace a některé pokročilejší reortogonalizační techniky v prostředí MATLABu. Předpokládáme, že softwarový balík, který byl v rámci této práce vytvořen bude ještě rozšířen o některé další bidiagonalizační techniky a další metody (strategie) reortogonalizace použitelné v Golub-Kahanově algoritmu. Celý balík, rozšířený o metody realizující jednotlivé testy (ztráta ortogonality, atd.), bude volně dostupný na internetu a bude postupně rozšiřován o další metody, které mají vztah k problematice řešení lineárních aproximačních problémů (klasického či úplného problému nejmenších čtverců) pomocí teorie core problému, což bylo

úvodní motivací pro vznik této práce. Tento software bude sloužit jako laboratoř pro studium chování transformací vyjevujících výše zmíněný core problém.

Literatura

[1] J. L. BARLOW, N. BOSNER, Z. DRMAČ: *A new stable bidiagonal reduction algorithm*, Linear Algebra Appl., 397 (2005), pp. 35–84.

[2] J. BAŠTINEC, M. NOVÁK: *Moderní numerické metody*, učební texty VUT Brno, 2009. (http://www.umat.feec.vutbr.cz/~novakm/MMNM2009.pdf)

[3] J. DONGARRA, I. DUFF, D. SORENSEN, H. VAN DER VORST: *Solving linear systems on vector and shared memory computers*, Philadelphia, PA, SIAM Publications, 1991.

[4] L. GIRAULD, J. LANGOU, M. ROZLOŽNÍK: *On the loss of orthogonality in the Gram-Schmidt orthogonalization process*, Computers & Mathematics with Applications, 50 (2005), pp. 1069–1075.

[5] L. GIRAULD, J. LANGOU, M. ROZLOŽNÍK, J. VAN DEN ESHOF: *Rounding error analysis of the classical Gram-Schmidt orthogonalization process*, Numerische Mathematik, 101 (2005), pp. 87–100.

[6] G. H. GOLUB, W. KAHAN: *Calculating the singular values and pseudo-inverse of a matrix*, SIAM Journal on numerical analysis, ser. B, 2 (1965), pp. 205–224.

[7] P. C. HANSEN: *Regularization Tools Version 4.1* (*for Matlab Version 7.3*). (http://www2.imm.dtu.dk/~pch/Regutools/index.html)

[8] I. HNĚTYNKOVÁ, M. PLEŠINGER, Z. STRAKOŠ, P. TICHÝ: *Numerické metody algebry*, manuskript učebního textu.

[9] M. KUBÍČEK, M. DUBCOVÁ, D. JANOVSKÁ: *Numerické metody a algoritmy*, učební text VŠCHT Praha, 2005. (http://vydavatelstvi.vscht.cz/knihy/uid_isbn-80-7080-558-7/pages-img/168.html)

[10] D. P. O'LEARY, Z. STAKOŠ, P. TICHÝ: *On senzitivity of Gauss-Christoffel quadrature*, Numerische Mathematik, 107 (2007), pp. 147–174.

[11] C. C. PAIGE: *Orthonormal completion of an array of unit length vectors*, Book of Abs., 17th Householder Symposium, Zeuthen, Germany (2008), pp. 131–132.

[12] C. C. PAIGE, Z. STRAKOŠ: *Core problem in linear algebraic systems*, SIAM Journal on Matrix Analysis and Applications, 27 (2006), pp. 861–875.

[13] B. N. PARLETT, D. S. SCOTT: *The Lanczos algorithm with selective ortogonalization*, American Mathematical Society, vol. 33, no. 145, (1979), pp. 217–238.

[14] M. PLEŠINGER: *The total least squares problem and reduction of data in* $AX \approx B$, Ph.D. Thesis, TU Liberec, 2008.

[15] H. D. SIMON: *The Lanczos algoritm with partial reorthogonalization*, AMS Mathematics of computations, vol. 42, no. 165 (1984), pp. 115–141.

[16] H. D. SIMON, H. ZHA: *Low-rank matrix approximation using the Lanczos bidiagonalization process with applications*, SIAM Publications, Journal on Scientific Computing, 21 (2000), pp. 2257–2274.