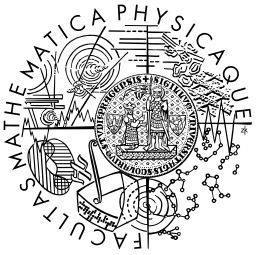


MATEMATICKO-FYZIKÁLNÍ
FAKULTA
Univerzita Karlova

BAKALÁŘSKÁ PRÁCE

Petr Mrňák

Numerické srovnání algoritmů CGLS a LSQR

Katedra numerické matematiky

Vedoucí bakalářské práce: doc. RNDr. Petr Tichý, Ph.D.

Studijní program: Obecná matematika

Studijní obor: MOMP

Praha 2024

Prohlašuji, že jsem tuto bakalářskou (resp. diplomovou, rigorózní, dizertační) práci vypracoval(a) samostatně a výhradně s použitím citovaných pramenů, literatury a dalších odborných zdrojů. Beru na vědomí, že se na moji práci vztahují práva a povinnosti vyplývající ze zákona č. 121/2000 Sb., autorského zákona v platném znění, zejména skutečnost, že Univerzita Karlova má právo na uzavření licenční smlouvy o užití této práce jako školního díla podle § 60 odst. 1 autorského zákona.

V ..... dne ....

Podpis autora

Rád bych poděkoval doc. RNDr. Petrovi Tichému, Ph.D. za jeho ochotu při konzultacích a odbornou pomoc. Dále bych rád poděkoval rodině a přátelům za to, že mě udržovali v dobré náladě během náročného studia.

Název práce: Numerické srovnání algoritmů CGLS a LSQR

Autor: Petr Mrňák

Katedra: Katedra numerické matematiky

Vedoucí bakalářské práce: doc. RNDr. Petr Tichý, Ph.D., Katedra numerické matematiky

Abstrakt: Tato bakalářská práce se zabývá představením dvou matematicky ekvivalentních algoritmů CGLS a LSQR, na které lze nahlížet jako na verze metody sdružených gradientů aplikované na systém normálních rovnic. Tato práce se věnuje jejich porovnání jak z teoretického hlediska (ukázání vztahů mezi vektory a koeficienty), tak i z praktického hlediska (chování obou algoritmů při výpočtech v aritmetice s konečnou přesností).

Klíčová slova: systém normálních rovnic, CGLS, LSQR, numerické chování

Title: Numerical comparison of the CGLS and LSQR algorithms

Author: Petr Mrňák

Department: Department of Numerical Mathematics

Supervisor: doc. RNDr. Petr Tichý, Ph.D., Department of Numerical Mathematics

Abstract: This bachelor thesis deals with the introduction of two mathematically equivalent algorithms, CGLS and LSQR, which can be viewed as versions of the method of conjugate gradients applied to a system of normal equations. This thesis is devoted to their comparison both from a theoretical point of view (showing the relations between vectors and coefficients) and from a practical point of view (the behaviour of both algorithms in finite precision arithmetic).

Keywords: system of normal equations, CGLS, LSQR, numerical behaviour

Obsah

Úvod

Uvažujme matici $A \in \mathbb{R}^{m\times n}$ a vektor $b \in \mathbb{R}^m$. V této práci se zaměříme se na typ úloh $Ax \approx b$, kterým se někdy říká lineární aproximační problémy, přičemž řešení $Ax = b$ nemusí vůbec existovat. Pro jednoduchost budeme v práci používat předpoklad $\text{rank}(A) = n$, tj. budeme uvažovat matice s plnou sloupcovou hodností. Dále je nutné upřesnit, v jakém smyslu má být Ax blízké vektoru b.

Přirozeným způsobem, jak měřit blízkost Ax k vektoru b, je euklidovská norma residua. Aproximaci x se poté snažíme určit tak, aby byla norma residua minimální

$$\min_y \|b - Ay\|. \tag{1}$$

Výše popsaný problém budeme nazývat *problémem nejmenších čtverců*. Lze ukázat, že řešení tohoto problému vždy existuje a pokud má A plnou sloupcovou hodnost, pak je řešení jednoznačné. Ekvivalentně můžeme na formulaci lineárního aproximačního problému nahlížet tak, že chybami je zatížen pouze vektor pravé strany b. Naším cílem je tudíž nalézt co nejmenší změnu f pravé strany b takovou, že x je řešením modifikované soustavy

$$Ax = b + f.$$

Budeme se držet definice problému nejmenší čtverců definované v [8]: Nechť $A \in \mathbb{R}^{m\times n}, b \in \mathbb{R}^n$. Pak problém nejmenších čtverců budeme nazývat úlohu určení $x \in \mathbb{R}^m$ tak, aby platilo

$$\min_x \|f\| \quad \text{za podmínky} \quad Ax = b + f.$$

Existuje několik způsobů, jak řešit problém nejmenších čtverců, viz [8]. Jedním z nich je využití faktu, že euklidovská norma je unitárně invariantní, tedy je možné problém nejmenších čtverců transformovat pomocí unitárních matic a k řešení využít QR rozklad nebo singulární rozklad matice A. Jiné způsoby řešení nabízejí formulace problému nejmenších čtverců v podobě, ve které x je řešením soustavy normálních rovnic nebo rozšířené soustavy rovnic se sedlobodovou maticí.

V této práci se zaměříme na řešení problému nejmenších čtverců iteračními metodami. Budeme uvažovat dva matematicky ekvivalentní algoritmy LSQR a CGLS, na které lze nahlížet jako na verze metody sdružených gradientů aplikované na systém normálních rovnic. Jelikož jsou oba algoritmy odvozené odlišným způsobem, zajímá nás detailně jejich teoretický vztah a také odlišnosti v numerickém chování. Pro praxi je potom klíčové zodpovědět otázku, zda je jeden z nich vhodnější pro praktické výpočty než druhý.

Značení

Čísla, množiny a vektorové prostory

$\mathbb{R}$	množina reálných čísel
$\{a,b,c,\dots\}$	množina čísel s prvky $a,b,c,\dots$
$\mathcal{V}$	lineární vektorový prostor
0	nulový vektor $\mathcal{V}$

Vektory

x	sloupcový vektor
$x \equiv [\xi_1,\dots,\xi_n]^T$	zápis vektoru x po prvcích
x^T	transponovaný vektor k vektoru x
e_i	i-tý sloupec jednotkové matice
$\text{span}\{q_1,\dots,q_m\}$	prostor generovaný vektory $q_1,\dots,q_m$

Matice

A	matice
$A \equiv [a_{i,j}]_{i,j=1}^{n,m}$	prvky matice $A \in \mathbb{C}^{n\times m}$
a_j	j-tý sloupec matice A
$A = [a_1,\dots,a_m]$	zápis matice A po sloupcích
A^T	transponovaná matice k matici A
A^{-1}	inverzní matice k regulární matici A
$\text{rank}(A)$	hodnost matice A
$I,\ I_n$	jednotková matice, jednotková matice dimenze n
$\text{diag}(d_1,\dots,d_n)$	čtvercová diagonální matice s prvky $d_1,\dots,d_n$ na diagonále
$\|x\|$	euklidovská norma vektoru x
$\|A\|_2, \|A\|$	spektrální norma matice A
$\|A\|_F$	Frobeniova norma matice A
ε	strojová přesnost počítače
$\mathcal{K}_k(A,b)$	k-tý Krylovův podprostor generovaný maticí A a vektorem b

1. Metody pro symetrické matice

Metody řešení soustavy normálních rovnic, které jsou naším cílem, jsou inspirovány metodami pro problémy se symetrickou (pozitivně definitní) maticí, které popisujeme níže. Konkrétně se jedná o Lanczosův algoritmus pro počítání ortogonální báze vhodného Krylovova podprostoru a o metodu sdružených gradientů pro řešení systému $Ax = b$ se symetrickou pozitivně definitní maticí. Známý vztah mezi koeficienty a vektory těchto dvou algoritmů nám poté umožní odvození vztahu mezi koeficienty a vektory algoritmů pro řešení systému normálních rovnic CGLS a LSQR.

1.1 Metoda sdružených gradientů

Uvažujme soustavu rovnic $Bx = c$, kde $B \in \mathbb{R}^{n\times n}$ je pozitivně definitní matice a $c \in \mathbb{R}^n$ vektor pravé strany. Nechť x_* je přesné řešení této soustavy rovnic.

Metoda sdružených gradientů (CG) je odvozena dle [4] jako metoda pro minimalizaci kvadratického funkcionálu

$$F(x) = \frac{1}{2}x^T Bx - x^T c = \frac{1}{2}\|x - x_*\|_B^2 - \frac{1}{2}\|x_*\|_B^2,$$

neboť platí $\nabla F = Bx - c$, tedy hledání minima je ekvivalentní problému hledání řešení dané soustavy rovnic.

Strategie minimalizace je založena na následujících rekurencích, pomocí kterých aktualizujeme aproximace x_k a směrové vektory, viz p_k [8]:

$$\begin{aligned} x_k &= x_{k-1} + \gamma_{k-1}p_{k-1}, \\ r_k &= r_{k-1} - \gamma_{k-1}Bp_{k-1}, \\ p_k &= r_k + \delta_k p_{k-1}. \end{aligned}$$

Zde je γ_{k-1} voleno tak, aby

$$\gamma_{k-1} = \underset{\gamma}{\operatorname{argmin}}\|x_{k-1} + \gamma p_{k-1}\|_B$$

a δ_k tak, aby vektor p_k byl B-ortogonální k p_{k-1}. Po algebraických úpravách lze vyjádřit γ_{k-1} a δ_k ve tvaru:

$$\gamma_{k-1} = \frac{p_{k-1}^T r_{k-1}}{p_{k-1}^T Br_{k-1}} = \frac{r_{k-1}^T r_{k-1}}{p_{k-1}^T Bp_{k-1}},$$

$$\delta_k = -\frac{p_{k-1}^T Br_k}{p_{k-1}^T Bp_{k-1}} = \frac{r_k^T r_k}{r_{k-1}^T r_{k-1}}.$$

Lze dokázat, že touto volbou docílíme nejen lokální ortogonality

$$r_k \perp p_{k-1}, \quad p_k \perp_B p_{k-1},$$

ale i globální ortogonality

$$r_i^T r_j = 0, \quad p_i^T Bp_j = 0 \quad i,j = 0,\dots k, \quad i \neq j.$$

Důsledkem je, že sdružené gradienty naleznou nejlepší aproximaci splňující

$$x_k = \underset{x \in \mathcal{S}_k}{\operatorname{argmin}} \|x_* - x\|_B,$$

kde $\mathcal{S}_k = x_0 + \operatorname{span}\{p_0, \ldots, p_{k-1}\}$.

Nastartujeme-li algoritmus s volbou $p_0 = r_0$, pak lze ukázat, že platí tento důležitý vztah pro k-tý Krylovovský prostor a k-té lineární obaly

$$\operatorname{span}\{r_0, \ldots, r_{k-1}\} = \mathcal{K}_k(B, r_0) = \operatorname{span}\{p_0, \ldots, p_{k-1}\}.$$

Nyní už je možné sestavit algoritmus CG pro $Bx = c$.

Algoritmus 1 CG [8]

input B, c, x_0
$r_0 = c - Bx_0$
$p_0 = r_0$
for $k = 1,2,\ldots$ **do**
  $\gamma_{k-1} = \frac{r_{k-1}^T r_{k-1}}{p_{k-1}^T B p_{k-1}}$
  $x_k = x_{k-1} + \gamma_{k-1} p_{k-1}$
  $r_k = r_{k-1} - \gamma_{k-1} B p_{k-1}$
  $\delta_k = \frac{r_k^T r_k}{r_{k-1}^T r_{k-1}}$
  $p_k = r_k + \delta_k p_{k-1}$
end for

1.2 Lanczosův algoritmus

Nechť $B \in \mathbb{R}^{n \times n}$ je symetrická matice, $c \in \mathbb{R}^n$ je vektor pravé strany. Nechť $k < d$, kde d je maximální dimenze, které může Krylovův podprostor $\mathcal{K}_k(B, c)$ dosáhnout (stupeň c vzhledem k B). Naším úkolem bude vygenerovat ortogonální bázi $\mathcal{K}_k(B, c)$, což lze provést následujícím algoritmem:

Algoritmus 2 Lanczosova tridiagonalizace [5]

input B, c, $v_0 = 0$
$\tilde{\beta}_1 v_1 = b$,
for $i = 1,2,\ldots,k$ **do**
  $w = Bv_i - \tilde{\beta}_i v_{i-1}$
  $\tilde{\alpha}_i = v_i^T w$
  $\tilde{\beta}_{i+1} v_{i+1} = w - \tilde{\alpha}_i v_i$
end for

V algoritmu je použit zápis $\tilde{\beta}_i v_i = w$, což znamená, že vektor v_i a koeficient $\tilde{\beta}_i$ spočteme následujícím způsobem: spočteme normalizační koeficient $\tilde{\beta}_i = \|w\|$ a poté dopočteme vektor v_i jako $v_i = \frac{w}{\tilde{\beta}_i}$.

Vektory $v_1, \ldots, v_{k+1}$ po k iteracích splňují následující maticový vztah:

$$BV_k = V_k T_k + \tilde{\beta}_{k+1} v_{k+1} e_k^T, \tag{1.1}$$

kde

$$T_k = \begin{pmatrix} \tilde{\alpha}_1 & \tilde{\beta}_2 & & \\ \tilde{\beta}_2 & \tilde{\alpha}_2 & \ddots & \\ & \ddots & \ddots & \tilde{\beta}_k \\ & & \tilde{\beta}_k & \tilde{\alpha}_k \end{pmatrix} \tag{1.2}$$

se říká Jacobiho matice a $V_k = [v_1, v_2, \ldots, v_k]$. Navíc platí $V_k^T V_k = I$. Rozepíšeme-li (1.1) po sloupcích, dostaneme pro $j = 1, \ldots, k$ rekurenci

$$\tilde{\beta}_{j+1} v_{j+1} = A v_j - \tilde{\alpha}_j v_j - \tilde{\beta}_j v_{j-1}, \tag{1.3}$$

viz [8]. Pokud je matice B navíc pozitivně definitní, tak je i vygenerovaná matice T_k pozitivně definitní. Argumentovat lze tím, že vlastní čísla matice T_k vždy leží v otevřeném intervalu určeném nejmenším a největším vlastním číslem matice B. Je-li matice B pozitivně definitní, má kladná vlastní čísla a tudíž je i matice T_k pozitivně definitní.

1.3 Vztah mezi CG a Lanczosovým algoritmem

Budeme uvažovat symetrickou pozitivně definitní matici B, pravou stranu c a počáteční aproximaci x_0. Uvažujeme-li počáteční vektor $r_0 = c - Bx_0$, potom Lanczosův algoritmus aplikovaný na matici B a vektor r_0 generuje ortonormální bázi $\{v_1, \ldots, v_k\}$ Krylovova podprostoru $\mathcal{K}_k(B, r_0)$. Aproximaci x_k metody sdružených gradientů, která leží ve varietě $x_0 + \mathcal{K}_k(B, r_0)$, lze vyjádřit ve tvaru

$$x_k = x_0 + V_k y_k, \tag{1.4}$$

kde $V_k = [v_1, \ldots, v_k]$ a y_k je (zatím neznámý) souřadnicový vektor. Odpovídající residuum

$$r_k = c - Bx_k = r_0 - BV_k y_k, \tag{1.5}$$

je ortogonální k prostoru $\mathcal{K}_k(B, r_0)$, tedy i ke všem bázovým vektorům $v_1, \ldots, v_k$, tj platí

$$0 = V_k^* r_k = \|r_0\| e_1 - V_k^* B V_k y_k = \|r_0\| e_1 - T_k y_k, \tag{1.6}$$

kde T_k je Jacobiho matice z Lanczosova algoritmu.

Z výše popsaného lze nahlédnout, že CG aproximaci x_k je možné počítat pomocí Lanczosova algoritmu. Tuto aproximaci x_k vyjádříme ve tvaru

$$x_k = x_0 + V_k y_k, \quad \text{kde} \quad \|r_0\| e_1 = T_k y_k.$$

V [8] jsou odvozeny podrobnější vztahy mezi koeficienty: Pokud obě metody aplikujeme na systém $Bx = c$ s počáteční aproximací x_0 a se stejným počátečním vektorem $r_0 = v_1$, tak jsou generovány nenulové ortogonální bázové vektory w_j Krylovových podprostorů takové, že platí

$$w_j \in \mathcal{K}_k(B, r_0) \quad a \quad w_{j+1} \perp \mathcal{K}_k(B, r_0), \quad j = 0, \ldots, k-1.$$

V Lanczosově algoritmu se jedná o ortonormální Lanczosovy vektory, tedy $w_j = v_{j+1}$, naopak v CG jsou těmito bázovými vektory residua, tedy $w_j = r_j$. Ortogonální bázové vektory w_j jsou těmito podmínkami (1.3) určeny jednoznačně

(až na násobek nenulovým skalárem). To tedy implikuje, že normalizovaná residua r_j jsou až na znaménko rovna Lanczosovým vektorům v_{j+1}. Pomocí srovnání znaménka koeficientu u nejvyšší mocniny B získáme vztah

$$v_{j+1} = (-1)^j \frac{r_j}{\|r_j\|}, \quad j = 0, \ldots, k.$$

Nyní se podíváme na odvození přesných vztahů mezi koeficienty $\tilde{\alpha}_j, \tilde{\beta}_j$ vyskytujími se v Lanczosově algoritmu a γ_j, δ_j vyskytujícími se v CG metodě. Residua r_k a směrové vektory p_k jsou počítány pomocí dvou dvoučlenných rekurencí,

$$r_j = r_{j-1} - \gamma_{j-1} A p_{j-1}, \quad p_j = r_j + \delta_j p_{j-1}.$$

Algebraickými úpravami lze tyto dva vztahy přepsat pomocí jedné tříčlenné rekurence, tedy platí

$$\begin{aligned} r_k &= r_{j-1} - \gamma_{j-1} A p_{j-1} \\ &= r_{j-1} - \gamma_{j-1} A (r_{j-1} + \delta_{j-1} p_{j-2}) \\ &= r_{j-1} - \gamma_{j-1} A r_{j-1} - \gamma_{j-1}\delta_{j-1} \frac{r_{j-2} - r_{j-1}}{\gamma_{j-2}} \\ &= -\gamma_{j-1} A r_{j-1} + \left(1 + \frac{\gamma_{j-1}\delta_{j-1}}{\gamma_{j-2}}\right) r_{j-1} - \frac{\gamma_{j-1}\delta_{j-1}}{\gamma_{j-2}} r_{j-2}. \end{aligned}$$

Přenásobením celého vztahu faktorem $(-1)^j \frac{1}{\|r_{j-1}\|\gamma_{j-1}}$ a drobnou úpravou získáme vztah

$$\begin{aligned} \frac{\|r_j\|}{\gamma_{j-1}\|r_{j-1}\|} \left[(-1)^j \frac{r_j}{\|r_j\|}\right] =& B \left[(-1)^{j-1} \frac{r_{j-1}}{\|r_{j-1}\|}\right] - \left(\frac{1}{\gamma_{j-1}} + \frac{\delta_{j-1}}{\gamma_{j-2}}\right) \left[(-1)^{j-1} \frac{r_{j-1}}{\|r_{j-1}\|}\right] \\ &- \frac{\delta_{j-1}\|r_{j-2}\|}{\gamma_{j-2}\|r_{j-1}\|} \left[(-1)^{j-2} \frac{r_{j-2}}{\|r_{j-2}\|}\right]. \end{aligned}$$

Konečně, využijeme toho, že

$$\sqrt{\delta_j} = \frac{\|r_j\|}{\|r_{j-1}\|} \quad \text{a} \quad v_{j+1} = (-1)^j \frac{r_j}{\|r_j\|},$$

a dostáváme, že

$$\frac{\sqrt{\delta_j}}{\gamma_{j-1}} = A v_j - \left(\frac{1}{\gamma_{j-1}} + \frac{\delta_{j-1}}{\gamma_{j-2}}\right) v_j - \frac{\sqrt{\delta_{j-1}}}{\gamma_{j-2}} v_{j-1}. \tag{1.7}$$

Porovnáním (1.7) s tříčlennou rekurencí (1.3) pro Lanczosovy vektory dostaneme pro $j \geq 1$

$$\tilde{\beta}_{j+1} = \frac{\delta_j}{\gamma_{j-1}}, \quad \tilde{\alpha}_j = \frac{1}{\gamma_{j-1}} + \frac{\delta_{j-1}}{\gamma_{j-2}}, \tag{1.8}$$

přičemž položíme $\delta_0 = 0, \gamma_{-1} = 1$. Vztahy (1.8) lze rovněž zapsat v maticovém tvaru. Položíme-li

$$L_k = \begin{pmatrix} \frac{1}{\sqrt{\gamma_0}} & & & \\ \sqrt{\frac{\delta 1}{\gamma_0}} & \ddots & & \\ & \ddots & \ddots & \\ & & \sqrt{\frac{\delta_{k-1}}{\gamma_{k-2}}} & \frac{1}{\sqrt{\gamma_{k-1}}} \end{pmatrix}$$

platí dle (1.2)

$$T_k = L_k L_k^T.$$

Metoda sdružených gradientů tak implicitně počítá Choleského rozklad matice T_k.

2. LSQR a CGLS

Nyní uvažujme systém normálních rovnic s $B = A^T A$. Z předchozí kapitoly víme, že lze použít metodu sdružených gradientů a Lanczosův algoritmus. Tyto algoritmy aplikované na systém normálních rovnic je však vhodné algebraicky modifikovat kvůli zlepšení jejich numerického chování. Níže uvádíme dva nejpoužívanější algoritmy, CGLS a LSQR. Oba tyto algoritmy jsou matematicky ekvivalentní s metodou sdružených gradientů aplikovanou na systém normálních rovnic.

2.1 Algoritmus CGLS

Algoritmus 1 lze aplikovat na $B = A^T A$ a $c = A^T b$, vše ostatní ponecháme beze změny, čímž vznikne následující algoritmus:

Algoritmus 3 CG pro systém normálních rovnic

input A, b, x_0
$z_0 = A^T b - A^T A x_0$
$p_0 = z_0$
for $k = 0,1,\ldots$ **do**
  $t_k = A p_k$
  $\gamma_k = \|z_k\|^2 / \|t_k\|^2$
  $x_{k+1} = x_k + \gamma_k p_k$
  $z_{k+1} = z_k - \gamma_k A^T t_k$
  $\delta_{k+1} = \|z_{k+1}\|^2 / \|z_k\|^2$
  $p_{k+1} = z_{k+1} + \delta_{k+1} p_k$
end for

Nyní už stačí Algoritmus 3 trochu upravit do podoby uvedené v [1], čímž vznikne Algoritmus 4 známý jako CGLS. Je dobré si povšimnout, že platí

$$z_k = c - B x_k = A^T b - A^T A x_k = A^T (b - A x_k) = A^T r_k.$$

Algoritmus 4 CGLS [1]

input A, b, x_0
$r_0 = b - A x_0$
$z_0 = p_0 = A^T r_0$
for $k = 0,1,\ldots$ **do**
  $t_k = A p_k$
  $\gamma_k = \|z_k\|^2 / \|t_k\|^2$
  $x_{k+1} = x_k + \gamma_k p_k$
  $r_{k+1} = r_k - \gamma_k t_k$
  $z_{k+1} = A^T r_{k+1}$
  $\delta_{k+1} = \|z_{k+1}\|^2 / \|z_k\|^2$
  $p_{k+1} = z_{k+1} + \delta_{k+1} p_k$
end for

Z vlastností metody sdružených gradientů [8] plynou pro oba výše zmíněné algoritmy následující vztahy, které budou využity později:

$$\begin{aligned} p_i^T A^T A p_j &= 0, \quad i \neq j, \\ A^T r_k &\perp \mathcal{K}_k(A^T A,\, A^T r_0), \\ x_k &\in x_0 + \mathcal{K}_k(A^T A,\, A^T r_0). \end{aligned} \tag{2.1}$$

Z koeficientů γ_i, δ_i z Algoritmu 4 lze vytvořit matici

$$L_k = \begin{pmatrix} \frac{1}{\sqrt{\gamma_0}} & & & \\ \sqrt{\frac{\delta 1}{\gamma_0}} & \ddots & & \\ & \ddots & \ddots & \\ & & \sqrt{\frac{\delta_{k-1}}{\gamma_{k-2}}} & \frac{1}{\sqrt{\gamma_{k-1}}} \end{pmatrix}$$

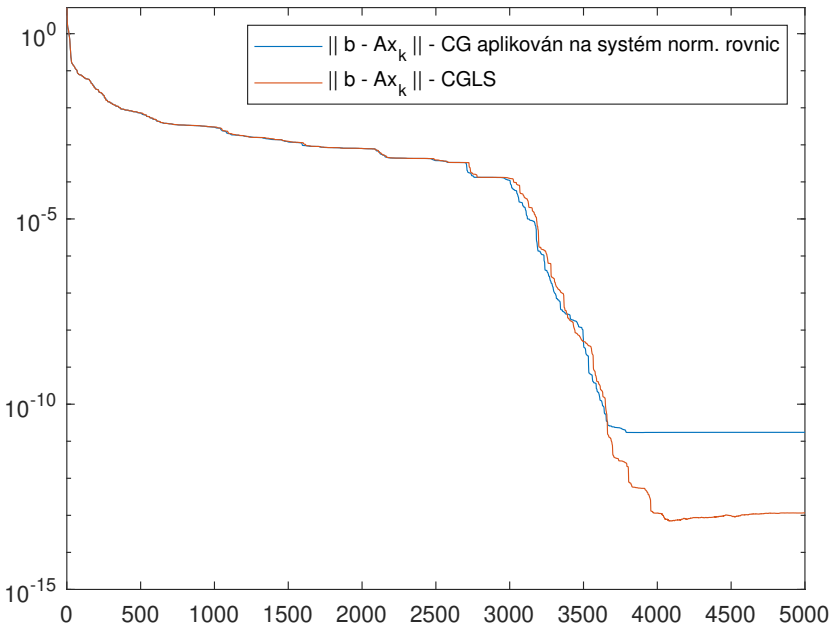
stejně jako v sekci 1.3 o srovnání algoritmů. Aplikujeme-li nyní Lanczosův Algoritmus 2 na matici $A^T A$ a startovací vektor $A^T r_0$, získáme tridiagonální matici T_k. Z výsledků v odstavci 1.3 poté plyne, že opět

$$T_k = L_k L_k^T,$$

tj. CGLS počítá implicitně Choleského rozklad matice T_k z Lanczosova algoritmu aplikovaného na $A^T A$ a $A^T r_0$.

2.2 Srovnání CG pro normální rovnice a CGLS

V minulé sekci bylo nastíněno, že algoritmus 3 lze drobnou úpravou převést na Algoritmus 4. Tato úprava je zásadní, neboť díky ní získá Algoritmus 4 lepší numerické vlastnosti. Rozdíl si ukážeme na jednom konkrétním příkladu a to matici illc1033 ze Sparse matrix collection [2]. Tato matice má rozměry 1033×320 a její číslo podmíněnosti je $1{,}88 * 10^4$.



Na obrázku jsou znázorněny dvě křivky. Červená křivka znázorňuje normu skutečného residua $b - Ax$, kde x_k je počítáno Algoritmem 4. Modrá křivka znázorňuje normu skutečného residua s x_k počítaným Algoritmem 3. Jelikož jsou oba algoritmy matematicky ekvivalentní, obě křivky by měly být teoreticky totožné. Prvních zhruba 2500 iterací si obě křivky těsně odpovídají, což naznačuje, že algoritmy dávají v aritmetice s konečnou přesností podobné výsledky. Po ustálení normy skutečného residua na hladině limitní přesnosti je vidět, že lepším algoritmem je Algoritmus 4 (CGLS) oproti Algoritmu 3 (CG pro normální rovnice). Podrobnější numerickou studii různých algoritmů pro řešení systému normálních rovnic lze nalézt v [1]. Dále již tedy nebude Algoritmus 3 brán v potaz kvůli nevhodným numerickým vlastnostem.

2.3 Golub-Kahanova bidiagonalizace

V sekci 1.2 jsme se seznámili s Lanczosovým Algoritmem 2 schopným vygenerovat ortogonální bázi Krylovova prostoru $\mathcal{K}_k(A^TA, A^Tb)$.

Nyní bychom mohli aplikovat Lanczosův Algoritmus 2 na systém normálních rovnic $A^TAx = A^Tb$, kde $A \in \mathbb{R}^{m\times n}$ je matice s plnou sloupcovou hodností a $b \in \mathbb{R}^m$ je vektor pravé strany a počítat ortogonální bázi Krylovova prostoru $\mathcal{K}_k(A^TA, A^Tb)$. Existuje však numericky vhodnější způsob, jak generovat bázi tohoto Krylovova prostoru, a tím je algoritmus Golub-Kahanovy bidiagonalizace (dále jen zkráceně bidiagonalizace). Zatímco Lanczosův Algoritmus 2 konstruoval jednu posloupnost vektorů, Golub-Kahanova bidiagonalizace (Algoritmus 5) konstruuje simultánně dvě posloupnosti vektorů. Vztah mezi tridiagonalizací a bidiagonalizací je k nalezení v [6].

Algoritmus 5 *Bidiag* [3]

1: **input** A, b
2: $\beta_1 u_1 = b$
3: $\alpha_1 v_1 = A^T u_1$
4: **for** $i = 1,2,\ldots$ **do**
5: $\quad \beta_{i+1} u_{i+1} = Av_i - \alpha_i u_i$
6: $\quad \alpha_{i+1} v_{i+1} = A^T u_{i+1} - \beta_{i+1} v_i$
7: **end for**

Normalizační koeficienty $\alpha_i > 0$ a $\beta_i > 0$ jsou vybrány tak, že $\|u_i\| = \|v_i\| = 1$. Označíme-li matice $U_k = [u_1, u_2, \ldots, u_k], V_k = [v_1, v_2, \ldots, v_k]$, tak sloupečky matice V_k tvoří ortonormální bázi Krylovova prostoru $\mathcal{K}_k(A^TA, A^Tb)$, navíc sloupečky matice U_k tvoří ortonormální bázi Krylovova prostoru $\mathcal{K}_k(AA^T, Ab)$ jak je popsáno v [1]. Platí tedy

$$U_k^T U_k = I_k = V_k^T V_k.$$

Sestavíme-li z koeficientů α_i a β_i matici

$$B_k = \begin{pmatrix} \alpha_1 & & & \\ \beta_2 & \alpha_2 & & \\ & \beta_3 & \ddots & \\ & & \ddots & \alpha_k \\ & & & \beta_{k+1} \end{pmatrix},$$

tak vektory počítané Algoritmem 5 splňují vztahy

$$\begin{aligned} U_{k+1}(\beta_1 e_1) &= b, && (2.2)\\ AV_k &= U_{k+1}B_k, && (2.3)\\ A^T U_{k+1} &= V_k B_k^T + \alpha_{k+1} v_{k+1} e_{k+1}^T. && (2.4) \end{aligned}$$

Přenásobíme-li zleva vztah (2.3) maticí A^T a dosadíme ze vztahu (2.4), dostaneme

$$\begin{aligned} A^T A V_k &= A^T U_{k+1} B_k \\ &= \left(V_k B_k^T + \alpha_{k+1} v_{k+1} e_{k+1}^T\right) B_k \\ &= V_k B_k^T B_k + \alpha_{k+1}\beta_{k+1} v_{k+1} e_{k+1}^T, \end{aligned}$$

což odpovídá vztahu (1.1) pro $B = A^T A$ a pro

$$T_k = B_k^T B_k. \tag{2.5}$$

Jinak řečeno, na vektory V_k konstruované pomocí Algoritmu bidiagonalizace 5 lze nahlížet jako na Lanczosovy vektory, kde Lanczosův Algoritmus 2 je aplikovaný na matici $A^T A$ a vektor $A^T b$. Současně, na tridiagonální matici $B_k^T B_k$ lze nahlížet jako na matici Lanczosových koeficientů T_k.

2.4 Algoritmus LSQR

Bázové vektory generované v bidiagonalizačním algoritmu 5 lze použít ke spočtení aproximace řešení problému nejmenších čtverců, viz (1). Všude uvažujeme počáteční volbu $x_0 = 0$. Následující odvození od (2.6) až po (2.8) je převzato z [6]. Aproximaci x_k hledáme v Krylovově prostoru $\mathcal{K}_k(A^T A, A^T b)$ a Algoritmus bidiagonalizace 5 nám dá ortogonální bázi. Pak lze x_k hledat ve tvaru

$$\begin{aligned} x_k &= V_k y_k, && (2.6)\\ r_k &= b - A x_k, && (2.7) \end{aligned}$$

pro nějaký vektor y_k. Následně pomocí

$$\begin{aligned} r_k &\overset{(2.7)}{=} b - A x_k \\ &\overset{(2.2)}{=} U_{k+1}(\beta_1 e_1) - A x_k \\ &\overset{(2.6)}{=} U_{k+1}(\beta_1 e_1) - A V_k y_k \\ &\overset{(2.3)}{=} U_{k+1}(\beta_1 e_1) - U_{k+1} B_k y_k \\ &\overset{(2.3)}{=} U_{k+1}(\beta_1 e_1 - B_k y_k), \end{aligned}$$

lze odvodit vztah

$$r_k = U_{k+1}(\beta_1 e_1 - B_k y_k). \tag{2.8}$$

Z požadavku minimální normy residua $\|r_k\|$ plyne, že y_k musí minimalizovat normu $\|\beta_1 e_1 - B_k y_k\|$. Tím pádem se dostáváme k problému nejmenších čtverců

$$\min \|\beta_1 e_1 - B_k y_k\|, \tag{2.9}$$

který lze řešit například pomocí QR rozkladu matice B_k. Tento rozklad označíme jako $B_k = Q_k R_k$ a jelikož je $B_k \in \mathbb{R}^{(k+1)\times k}$ dolní bidiagonální, je vzniklá matice $R_k \in \mathbb{R}^{k\times k}$ horní bidiagonální. To lze nahlédnout například tak, že si představíme vytváření QR rozkladu pomocí Givensových rotací nulujících poddiagonální prvky. Matici R_k budeme uvažovat ve tvaru

$$R_k = \begin{pmatrix} \rho_1 & \theta_2 & & & \\ & \rho_2 & \theta_3 & & \\ & & \ddots & \ddots & \\ & & & \ddots & \theta_k \\ & & & & \rho_k \end{pmatrix}.$$

Toto nám dává základ pro LSQR, viz Algoritmus 6 , kde bude QR rozklad dle [1] spočten pomocí Givensových rotací, v Algoritmu 6 se jedna o řádky 11 až 18.

Algoritmus 6 LSQR [6]

1: **input** A, b
2: $\beta_1 u_1 = b$
3: $\alpha_1 v_1 = A^T u_1$
4: $w_1 = v_1$
5: $x_0 = 0$
6: $\bar{\phi}_1 = \beta_1$, $\bar{\rho}_1 = \alpha_1$
7: **for** $k = 1,2,\ldots$ **do**
8: $\quad \beta_{k+1} u_{k+1} = A v_k - \alpha_k u_k$
9: $\quad \alpha_{k+1} v_{k+1} = A^T u_{k+1} - \beta_{k+1} v_k$
10:
11: $\quad \rho_k = (\bar{\rho}_k^2 + \beta_{k+1}^2)^{1/2}$
12: $\quad c_k = \bar{\rho}_k / \rho_k$
13: $\quad s_k = \beta_{k+1} / \rho_k$
14:
15: $\quad \theta_{k+1} = s_k \alpha_{k+1}$
16: $\quad \bar{\rho}_{k+1} = -c_k \alpha_{k+1}$
17: $\quad \phi_k = c_k \bar{\phi}_k$
18: $\quad \bar{\phi}_{k+1} = s_k \bar{\phi}_k$
19:
20: $\quad x_k = x_{k-1} + (\phi_k / \rho_k) w_k$
21: $\quad w_{k+1} = v_{k+1} - (\theta_{k+1} / \rho_k) w_k$
22: **end for**

Jelikož algoritmus LSQR zahrnuje bidiagonalizační algoritmus 5, tak i zde dle [7] tvoří sloupečky V_k ortonormální bázi Krylovova prostoru $\mathcal{K}_k(A^T A, A^T r_0)$.

Kromě bázových vektorů u_k a v_k generuje LSQR Algoritmus 6 ještě vektory w_k, které hrají roli směrových vektorů při výpočtu aproximací x_k. Pro LSQR

algoritmus lze pomocí [1] odvodit následující vlastnosti pro vektory z algoritmu, které použijeme později:

$$\begin{aligned} w_i^T A^T A w_j &= 0, \quad i \neq j, \\ v_{k+1} &\perp \mathcal{K}_k(A^T A,\, A^T r_0), \\ x_k &\in x_0 + \mathcal{K}_k(A^T A,\, A^T r_0). \end{aligned} \tag{2.10}$$

3. Porovnání CGLS a LSQR

Hlavní myšlenkou srovnávání algoritmů CGLS (algoritmus 4) a LSQR (algoritmus 6) je využití jejich blízkého vztahu k metodě sdružených gradientů a k Lanczosovu Algoritmu 2, u kterých jsou vztahy mezi koeficienty a vektory známy. Věnujme se nejprve vysvětlení matematické ekvivalence obou algoritmů. Aproximace x_k generované algoritmem CGLS jsou jednoznačně určeny podmínkami

$$A^T r_k \perp \mathcal{K}_k(A^T A,\, A^T r_0),$$

viz (2.1). V následujícím lemmatu ukážeme, že stejnými podmínkami jsou určeny i aproximace konstruované algoritmem LSQR.

Lemma 1. *Pro vektory generované algoritmem LSQR (Algoritmus 6) platí*

$$A^T(b - Ax_k) = \alpha_{k+1} v_{k+1} e_{k+1}^T (\beta_1 e_1 - B_k y_k).$$

Důkaz. Z faktu, že y_k je řešením (2.9), platí $B_k^T B_k y_k = B_k^T \beta_1 e_1$. Jelikož platí $x_k = V_k y_k$ dle (2.6) a $AV_k = U_{k+1} B_k$ dle (2.3), dostaneme

$$\begin{aligned} A^T(b - Ax_k) &= A^T U_{k+1}(\beta_1 e_1 - B_k y_k) \\ &= (V_k B_k^T + \alpha_{k+1} v_{k+1} e_{k+1}^T)(\beta_1 e_1 - B_k y_k) \\ &= V_k(B_k^T \beta_1 e_1 - B_k^T B_k y_k) - \alpha_{k+1} v_{k+1} e_{k+1}^T B_k y_k \\ &= -\alpha_{k+1} v_{k+1} \left[e_{k+1}^T B_k y_k \right], \end{aligned}$$

kde jsme použili, že $e_{k+1}^T e_1 = 0$.

□

Předchozí lemma nám říká, že vektor $A^T(b - Ax_k)$ je nenulovým násobkem vektoru v_{k+1} Podmínku ortogonality v_{k+1} vůči prostoru $\mathcal{K}_k(A^T A, A^T r_0)$ lze tedy vnímat jako ortogonalitu vektoru $A^T(b - Ax_k)$ k tomu saméme Krylovovu prostoru. Jinak řečeno, aproximace x_k počítané LSQR splňují stejné určující podmínky jako aproximace počítané metodou CGLS a oba algoritmy jsou tudíž matematicky ekvivalentní. Následující věta ukazuje, jaké vektory si v obou algoritmech vzájemně odpovídají.

Věta 2. *Nechť $A \in \mathbb{R}^{m \times n}$je matice s plnou sloupcovou hodností, $b \in \mathbb{R}^m$ je vektor a uvažujme algoritmy LSQR (Algoritmus 6) a CGLS (Algoritmus 4). Pak je vektor v_{k+1} nenulový násobek z_k a vektor w_{k+1} je nenulový násobek p_k.*

Důkaz. Jak jsme odůvodnili výše, oba algoritmy jsou matematicky ekvivalentní a konstruují stejné aproximace řešení x_k. Platí tedy

$$\gamma_k p_k = x_{k+1} - x_k = \frac{\phi_{k+1}}{\rho_{k+1}} w_{k+1},$$

a z toho plyne lineární závislost p_k a w_{k+1}. To, že je v_{k+1} nenulovým násobkem z_k plyne bezprostředně z předchozího lemmatu.

□

Před uvedením následující věty je dobré si vyjádřit matici T_k jiným způsobem. Ze vztahu mezi tridiagonalizací (Algoritmus 2) a bidiagonalizací (Algoritmus 5) nám plyne, že tridiagonální matice z Lanczosova Algoritmu 2 lze získat pomocí $T_k = B_k^T B_k$, viz (2.5). S využitím QR rozkladu matice B_k dostaneme

$$\begin{aligned} T_k &= B_k^T B_k \\ &= (Q_k R_k)^T Q_k R_k \\ &= R_k^T Q_k^T Q_k R_k \\ &= R_k^T R_k. \end{aligned} \tag{3.1}$$

LSQR algoritmus tedy implicitně počítá Choleského rozklad matice T_k. Tento fakt nám pomůže určit vztah mezi koeficienty algoritmů LSQR a CGLS.

Věta 3. *Nechť $A \in \mathbb{R}^{m\times n}$je matice s plnou sloupcovou hodností, $b \in \mathbb{R}^m$ je vektor a uvažujme algoritmy LSQR (Algoritmus 6) a CGLS (Algoritmus 4). Pak platí*

$$\gamma_{i-1} = \frac{1}{\rho_i^2} \quad a \quad \delta_i = \frac{\theta_{i+1}^2}{\rho_i^2} \qquad pro \qquad i = 1, \ldots, k-1.$$

Důkaz. Ze vztahu (3.1) víme, že CGLS algoritmus 4 počítá implicitně Choleského rozklad T_k ve tvaru $T_k = L_k L_k^T$, kde matice L_k obsahuje koeficienty z algoritmu CGLS. Z diskuse před větou víme, že LSQR algoritmus 6 počítá implicitně Choleského rozklad T_k ve tvaru $T_k = R_k^T R_k$, kde matice R_k obsahuje koeficienty z algoritmu LSQR. Z jednoznačnosti Choleského rozkladu plyne $L_k = R_k^T$, neboli platí maticová rovnost

$$\begin{pmatrix} \frac{1}{\sqrt{\gamma_0}} & & & \\ \sqrt{\frac{\delta 1}{\gamma_0}} & \ddots & & \\ & \ddots & \ddots & \\ & & \sqrt{\frac{\delta_{k-1}}{\gamma_{k-2}}} & \frac{1}{\sqrt{\gamma_{k-1}}} \end{pmatrix} = \begin{pmatrix} \rho_1 & & & \\ \theta_2 & \rho_2 & & \\ & \theta_3 & \ddots & \\ & & \ddots & \ddots \\ & & & \theta_k & \rho_k \end{pmatrix}.$$

Porovnáním prvků matic dostaneme vztahy mezi koeficienty

$$\gamma_{i-1} = \frac{1}{\rho_i^2} \quad \text{a} \quad \delta_i = \frac{\theta_{i+1}^2}{\rho_i^2} \qquad \text{pro} \qquad i = 1, \ldots, k-1.$$

□

Za zmínku ještě stojí fakt, že CGLS během jedné iterace updatuje 3 vektory x_k, p_k, z_k, kdežto LSQR updatuje 4 vektory x_k, w_k, v_k, u_k. Vektory x_k, p_k a z_k mají svůj protějšek v algoritmu LSQR, viz Věta 2. Čtvrtý vektor z LSQR, u_k, lze vnímat jako pomocný vektor, který je zakomponován do výpočtu vektoru v_k v algoritmu LSQR.

4. Numerické experimenty

Prvně si přiblížíme, co a jak budeme měřit a zkoumat.

Vlivem počítačové aritmetiky dochází k tomu, že chyba aproximace měřena například normou residua začne stagnovat na určité úrovni, té říkáme *hladina limitní přesnosti*. Aproximace x_k se pak již dále vlivem počítačové aritmetiky neblíží k přesnému řešení, [8].

Druhý jev, který můžeme pozorovat, je zpoždění konvergence, to je způsobeno ztrátou ortogonality. Ztrátu ortogonality měříme tak, že z vektorů, u nichž měříme ztrátu ortogonality, vytvoříme sloupečky matic

$$V_k = [v_1, \ldots, v_k] \text{ (pro LSQR)},$$

$$Z_k = \left[\frac{z_0}{\|z_0\|}, \ldots, \frac{z_{k-1}}{\|z_{k-1}\|}\right] \text{ (pro CGLS)}.$$

V ideálním případě budou mít obě matice ortonormální sloupce, tedy $V_k^T V_k = I = Z_k^T Z_k$. Míru ortogonality tedy můžeme měřit například pomocí $\|I - V_k^T V_k\|_F$ a $\|I - Z_k^T Z_k\|_F$.

V prvním experimentu uvažujeme Strakošovu matici dle [8], abychom demonstrovali fungování algoritmů při ztrátě ortogonality. Strakošova matice s parametry je diagonální matice a má v intervalu $[\lambda_1, \lambda_n]$ vlastní čísla

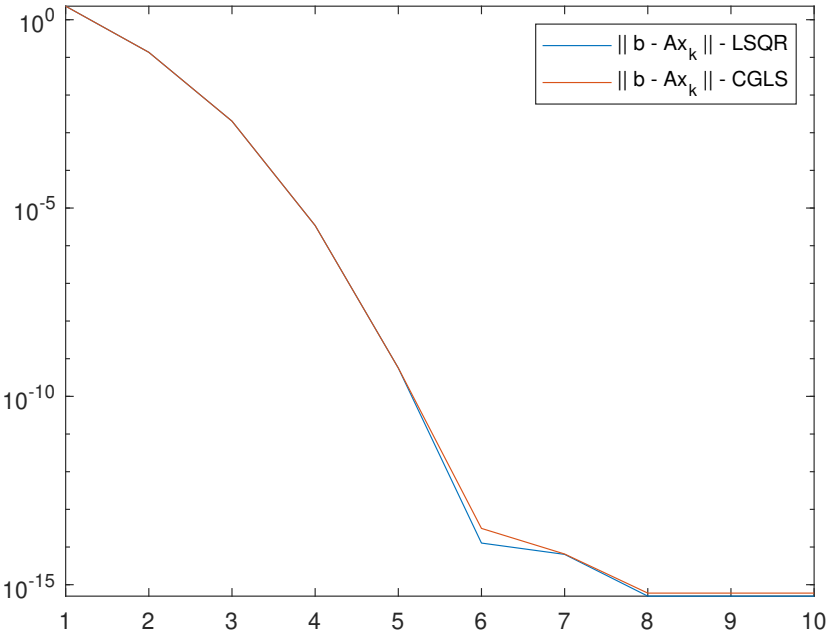
$$\lambda_i = \lambda_1 + \left(\frac{i-1}{n-1}\right)(\lambda_n - \lambda_1)\rho^{n-i}, \quad i = 1, \ldots, n,$$

kde $\rho \in (0,1]$ je parametr ovlivňující rozložení vlastních čísel. V této práci máme matici Strakoš24, což je Strakošova matice s parametry $\rho = 0.1, n = 24, \lambda_1 = 1, \lambda_n = 24$.

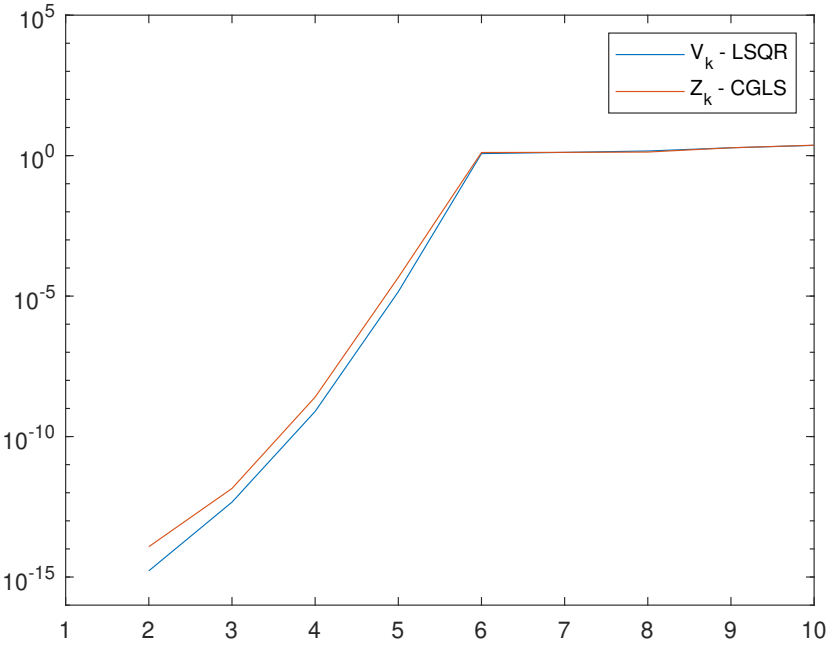
Na obrázku 4.1 jsou znázorněny dvě křivky. Červená křivka znázorňuje normu skutečného residua $b - Ax$, kde x_k je počítáno Algoritmem 4. Modrá křivka znázorňuje normu skutečného residua s x_k počítaným Algoritmem 6. Jelikož jsou oba algoritmy matematicky ekvivalentní, obě křivky by měly být teoreticky totožné. Prvních zhruba 5 iterací si obě křivky těsně odpovídají, což naznačuje, že algoritmy dávají v aritmetice s konečnou přesností podobné výsledky. Po ustálení normy skutečného residua na hladině limitní přesnosti je vidět, že křivky obou algoritmů jsou nerozeznatelné a tedy je těžké usoudit, který z algoritmů dosáhl lepší hladiny limitní přesnosti.

Na obrázku 4.2 jsou znázorněny dvě křivky. Červená křivka znázorňuje ztrátu ortogonality pro Algoritmus 4. Modrá křivka znázorňuje ztrátu ortogonality pro Algoritmus 6. Prvních zhruba 5 iterací je vidět, že menší ztrátu ortogonality má Algoritmus 6 (LSQR) oproti Algoritmu 4 (CGLS). Během dalších iterací si obě křivky těsně odpovídají, což naznačuje, že algoritmy dávají v aritmetice s konečnou přesností podobné výsledky.

V druhém experimentu se podíváme blíže jen na skutečné normy residuí metod CGLS a LSQR aplikovaných na problémy nejmenších čtverců s maticemi illc1033 a well1033. Matice illc1033 a well1033 jsme vybrali ze Suite sparse matrix collection [2] tak, aby byl ukázán rozdíl v numerickém chování algoritmu

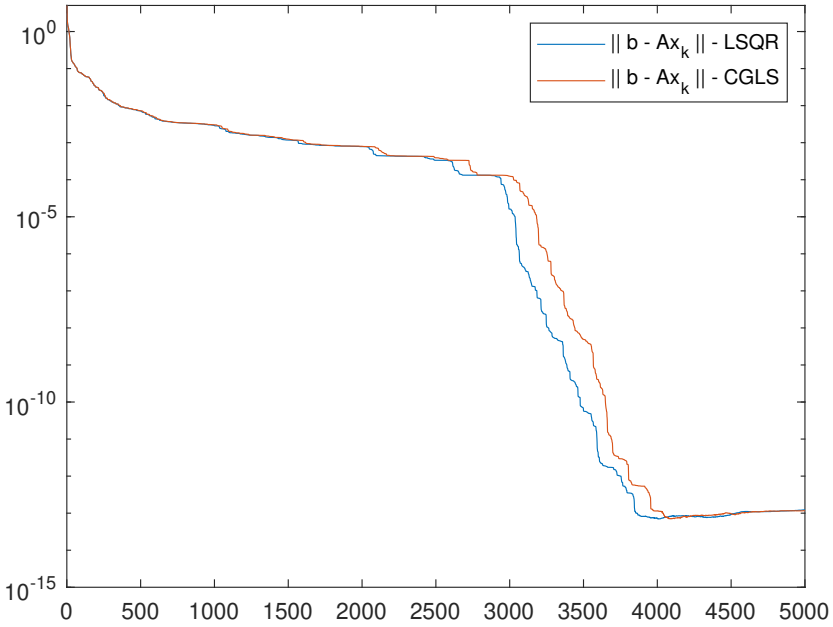


Obrázek 4.1: Skutečná norma residua pro algoritmy LSQR a CGLS aplikované na matici Strakoš24



Obrázek 4.2: Ztráta ortogonality pro algoritmy LSQR a CGLS aplikované na matici Strakoš24

LSQR a CGLS. Obě matice mají rozměry 1033×320 a obě mají hodnost 320, tedy plnou sloupcovou hodnost. Matice illc1033 má normu zhruba 2,1444 a číslo podmíněnosti zhruba $1{,}8888 * 10^4$. Matice well1033 má normu zhruba 1,8065 a číslo podmíněnosti zhruba $1{,}6613 * 10^2$. Na stránce [2] je k nalezení databáze matic, včetně podrobnějších vlastností matic illc1033 a well1033.



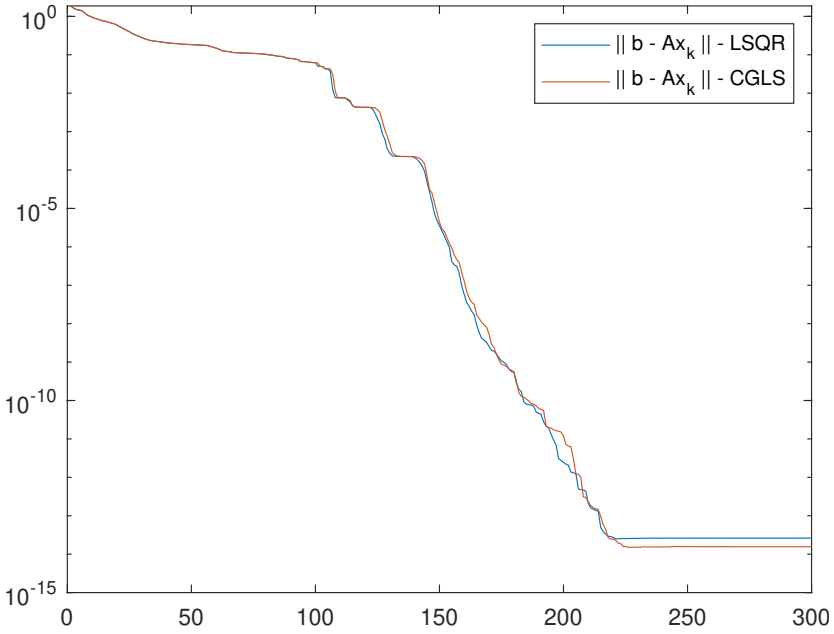
Obrázek 4.3: Skutečná norma residua pro algoritmy LSQR a CGLS aplikované na matici illc1033

Na obrázku 4.3 pozorujeme, že prvních zhruba 2000 iterací si obě křivky těsně odpovídají, což naznačuje, že algoritmy dávají v aritmetice s konečnou přesností podobné výsledky. Po ustálení normy skutečného residua na hladině limitní přesnosti je vidět, že lepší hladiny limitní přesnosti dosáhl Algoritmus 4 (CGLS) oproti Algoritmu 6 (LSQR).

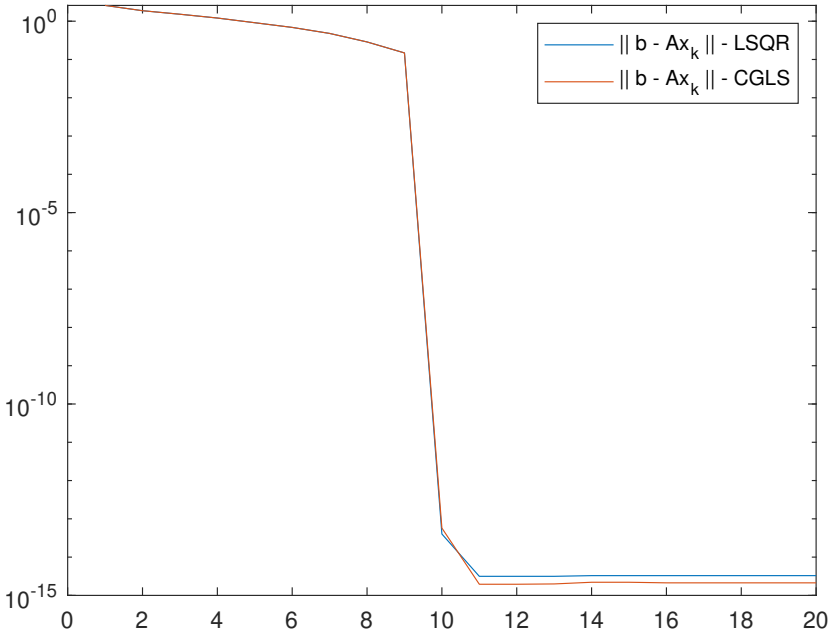
Na obrázku 4.4 pozorujeme, že prvních zhruba 100 iterací si obě křivky těsně odpovídají, což naznačuje, že algoritmy dávají v aritmetice s konečnou přesností podobné výsledky. Po ustálení normy skutečného residua na hladině limitní přesnosti je vidět, že lepší hladiny limitní přesnosti dosáhl Algoritmus 4 (CGLS) oproti Algoritmu 6 (LSQR).

Ve třetím experimentu se podíváme blíže jen na výpočty skutečné normy residua matic P(10,10,1,0.8) a P(10,10,1,11). Matice P(m,n,d,p) jsou speciálně vybrané matice z [1] tak, aby byl ukázán rozdíl mezi algoritmy LSQR a CGLS. Obě matice mají rozměry 10×10 a obě mají hodnost 10, tedy plnou sloupcovou hodnost. Matice P(10,10,1,0.8) má normu zhruba 1 a číslo podmíněnosti zhruba 6,3096. Matice P(10,10,1,11) má normu zhruba 1 a číslo podmíněnosti zhruba $1 * 10^{11}$. V [1] je k nalezení podrobnější popis testovacích matic typu P(m,n,d,p), včetně jejich podrobnějších vlastností.

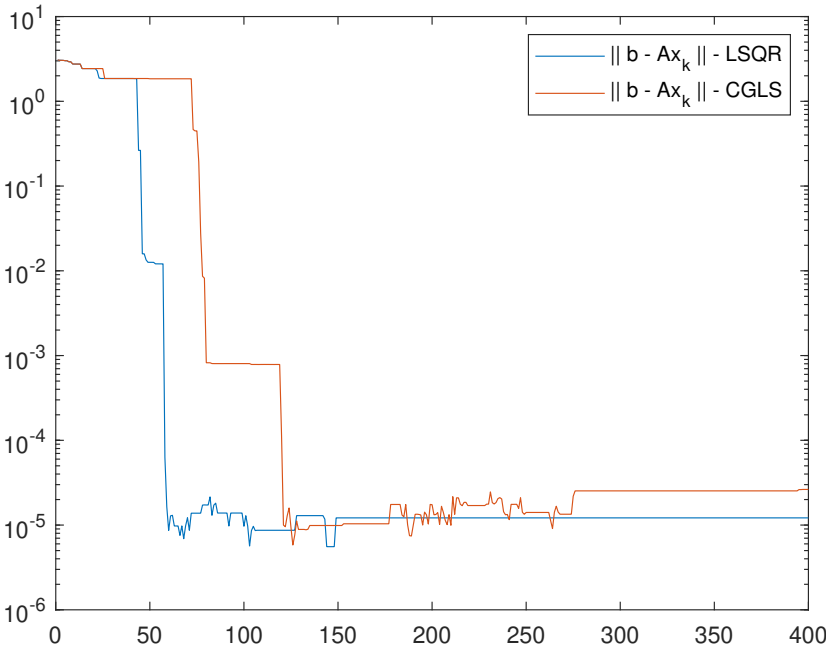
Na obrázku 4.5 pozorujeme, že prvních zhruba 15 iterací si obě křivky těsně odpovídají, což naznačuje, že algoritmy dávají v aritmetice s konečnou přesností podobné výsledky. Po ustálení normy skutečného residua na hladině limitní přesnosti je vidět, že lepší hladiny limitní přesnosti dosáhl Algoritmus 6 (LSQR) oproti Algoritmu 4 (CGLS).



Obrázek 4.4: Skutečná norma residua pro algoritmy LSQR a CGLS aplikované na matici well1033



Obrázek 4.5: Skutečná norma residua pro algoritmy LSQR a CGLS aplikované na matici P(10,10,1,0.8)



Obrázek 4.6: Skutečná norma residua pro algoritmy LSQR a CGLS aplikované na matici P(10,10,1,11)

Na obrázku 4.6 pozorujeme, že prvních zhruba 50 iterací si obě křivky těsně odpovídají, což naznačuje, že algoritmy dávají v aritmetice s konečnou přesností podobné výsledky. Po ustálení normy skutečného residua na hladině limitní přesnosti je vidět, že lepší hladiny limitní přesnosti dosáhl Algoritmus 6 (LSQR) oproti Algoritmu 4 (CGLS).

Na závěr experimentů je nutné podotknout, že v prodělaných experimentech se ani jeden z algoritmů neukázal být ve všech ohledech lepší než druhý. Na provedených experimentech je vidět, že CGLS má obvykle více zpožděnou konvergenci oproti LSQR. Na druhou stranu, CGLS obvykle dosahuje nepatrně lepší hladiny limitní přesnosti než LSQR.

Závěr

V prvních kapitole jsme uvažovali metodu sdružených gradientů a Lanczosův algoritmus, které jsme aplikovali na problémy se symetrickou (pozitivně definitní) matici. Ukázali jsme detailně vztahy mezi vektory a koeficienty těchto dvou algoritmů.

V druhé kapitole jsme si ukázali odvození algoritmů CGLS a LSQR pomocí předchozích algoritmů z první kapitoly. Algoritmus CGLS vznikl upravením metody sdružených gradientů pro systém normálních rovnic a LSQR vznikl pomocí Golub-Kahanovy bidiagonalizace. Ukázali jsme si nejdůležitější vlastnosti vektorů z algoritmů LSQR a CGLS, především vztahy ke Krylovovým prostorům, aby bylo možné je mezi sebou porovnat.

Ve třetí kapitole jsme se zabývali teoretickým porovnáním algoritmů LSQR a CGLS. Bylo ukázáno, že pro aproximace řešení konstruované algoritmy CGLS a LSQR platí stejné určující podmínky, tím pádem jsou oba algoritmy matematicky ekvivalentní. Byly odvozeny vztahy mezi vektory i koeficienty obou algoritmů včetně pomocného lemmatu a pozorování.

Ve čtvrté kapitole jsme studovali chování algoritmů při praktických výpočtech. Dívali jsme se na normy skutečných residuí, na hladinu limitní přesnosti a ztrátu ortogonality. Na uvedených příkladech je vidět, že CGLS má obvykle více zpožděnou konvergenci oproti LSQR. Na druhou stranu, CGLS obvykle dosahuje nepatrně lepší hladiny limitní přesnosti než LSQR. Zodpovědět, ve kterých případech je výhodnější upřednostnit jeden algoritmus před druhým, je již nad rámec této práce.

Seznam použité literatury

[1] Å. Björck, T. Elfving, and Z. Strakoš. Stability of conjugate gradient and Lanczos methods for linear least squares problems. *SIAM J. Matrix Anal. Appl.*, 19(3):720–736 (electronic), 1998.

[2] T. A. Davis and Y. Hu. The university of Florida sparse matrix collection. *ACM Trans. Math. Softw.*, 38(1), dec 2011, http://sparse.tamu.edu/.

[3] G. Golub and W. Kahan. Calculating the singular values and pseudo-inverse of a matrix. *J. Soc. Indust. Appl. Math. Ser. B Numer. Anal.*, 2:205–224, 1965.

[4] M. R. Hestenes and E. Stiefel. Methods of conjugate gradients for solving linear systems. *J. Research Nat. Bur. Standards*, 49:409–436 (1953), 1952.

[5] C. Lanczos. An iteration method for the solution of the eigenvalue problem of linear differential and integral operators. *J. Research Nat. Bur. Standards*, 45:255–282, 1950.

[6] C. C. Paige and M. A. Saunders. LSQR: an algorithm for sparse linear equations and sparse least squares. *ACM Trans. Math. Software*, 8(1):43–71, 1982.

[7] Y. Saad. *Iterative methods for sparse linear systems*. Society for Industrial and Applied Mathematics, Philadelphia, PA, second edition, 2003.

[8] J. D. Tebbens, I. Hnětynková, M. Plešinger, Z. Strakoš, and P. Tichý. *Analýza metod pro maticové výpočty : základní metody*. Matfyzpress, Praha, 2012.