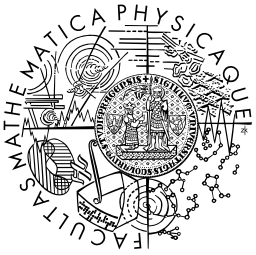


MATEMATICKO-FYZIKÁLNÍ
FAKULTA
Univerzita Karlova

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 2023

Prohlašuji, že jsem tuto bakalářskou práci vypracoval(a) samostatně a výhradně s použitím citovaných pramenů, literatury a dalších odborných zdrojů. Tato práce nebyla využita k získání jiného nebo stejného titulu.

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 algoritmů, konkrétně LSQR a CGLS, a poté jejich porovnání v oblasti teorie a oblasti praktického použití a výpočtů. Nejprve je důležité položit základy pro tyto algoritmy pomocí sdružených gradientů a Lanczosovy tridiagonalizace. Oba algoritmy jsou teoreticky ekvivalentní, ale v praxi je potřeba mezi nimi rozlišit, který je vhodnější pro daný výpočet.

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 algorithms, namely LSQR and CGLS, and then their comparison in the field of theory and the field of practical application and computation. First, it is important to lay the foundations for these algorithms by using conjugate gradients and Lanczos tridiagonalisation. Both algorithms are theoretically equivalent, but in practice it is necessary to distinguish between them which is more appropriate for a given calculation.

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

Obsah

Úvod

CGLS a LSQR jsou matematicky ekvivalentní algoritmy pro řešení systému normálních rovnic. Na CGLS lze nahlížet jako na vhodně upravenou verzi metody sdružených gradientů aplikovanou na systém normálních rovnic. Algoritmus LSQR je založen na Golub-Kahanově bidiagonalizaci a na následné konstrukci aproximace řešení pomocí řešení problému nejmenších čtverců s bidiagonální maticí.

Cílem práce je shrnout základní poznatky o těchto algoritmech, vyjasnit detaily ohledně ekvivalence obou algoritmů (vztahy mezi vektory) a následně provést srovnání jejich numerického chování. Numerické experimenty se zaměří například na porovnání ztráty ortogonality, zpoždění konvergence, či hladiny limitní přesnosti.

Na úvod si přiblížíme samotný problém nejmenších čtverců. Ne vždy totiž musí nutně existovat řešení soustavy rovnic $Ax = b$. I přesto má cenu se snažit hledat vektor x splňující

$$Ax \approx b,$$

přičemž je nutné vždy upřesnit, v jakém slova smyslu má být Ax blízko b.

Pro motivaci definice problému nejmenších čtverců uvažujme, že chybami je zatížen pouze vektor pravé strany b. Naším cílem tedy bude nalézt co nejmenší vektor změny f pravé strany b tak, aby x bylo řešením soustavy

$$Ax = b + f.$$

Budeme se držet definice problému nejmenší čtverců definované v [7]: Nechť $A \in \mathbb{C}^{n\times m}, b \in \mathbb{C}^n$. Pak problém nejmenších čtverců budeme nazývat úlohu určení $x \in \mathbb{C}^m$ tak, aby platilo

$$\min_{x,f} \|f\| \text{ za podmínky} Ax = b + f.$$

Existuje několik způsobů řešení problému nejmenších čtverců dle [7]. Mezi známé metody patří QR rozklad, singulární rozklad, ale i různé typy iteračních metod. Jediným a prakticky použitelným řešením pro rozsáhlé úlohy jsou iterační metody, zpravidla založené na Krylovových podprostorech rostoucí dimenze. Mezi iterační metody patří například LSQR a CGLS probírané v této práci.

Značení

Matice značíme A,B,..., vektory značíme v,w,... a skaláry jako $\alpha, \beta, \ldots$. Pro vektor v bude $\|v\|$ označovat Euklidovskou normu $\|v\| = (v^T v)^{1/2}$. Pro matici A bude $\|A\|$ označovat spektrální normu a $\|A\|_F$ označovat Frobeniovu normu, $\|A\|_F = (\sum a_{ij}^2)^{1/2}$. Budeme používat aritmetiku s konečnou přesností (FPA).

1. LSQR

Jelikož matice A bude často velká a řídká, nebo bude vyjádřitelná jako násobek matic či případně bude mít speciální strukturu, tak je A používána jen pro spočtení násobků vektoru a matice v podobě Av a $A^T v$.

Metoda LSQR je založena na Golub-Kahanově bidiagonalizaci [3].Tato metoda generuje posloupnost aproximací $\{x_k\}$ a to tak, že reziduální norma $\|r_k\|$ je minimalizována přes příslušný Krylovův prostor, kde $r_k = b - Ax_k$.

1.1 Motivace pomocí Lanczosova algoritmu

Nechť $B \in \mathbb{R}^{n\times n}$ je symetrická pozitivně definitní matice a buď b startovací vektor. Naším úkolem bude vygenerovat ortogonální bázi $\mathcal{K}_k(B, b)$. Pak vzniklá matice T_k je projekcí B do této báze.

Algoritmus 1 Lanczosova tridiagonalizace [4]

1: **input** B, b, $v_0 := 0$
2: $\beta_1 v_1 = b$,
3: **for** $i = 1,2,\ldots$ **do**
4: $\quad w_i = Bv_i - \beta_i v_{i-1}$
5: $\quad \alpha_i = v_i^T w_i$
6: $\quad \beta_{i+1} v_{i+1} = w_i - \alpha_i v_i$
7: **end for**

Kde každé $\beta_i > 0$ tak, aby $\|v_i\| = 1$, přičemž se algoritmus zastaví, pokud nastane $\beta_i = 0$ pro nějaké i. Dle [7] je T_k symetrická pozitivně definitní. Situace po k krocích by se dala popsat takto:

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

kde

$$T_k := \begin{pmatrix} \alpha_1 & \beta_2 & & \\ \beta_2 & \alpha_2 & \ddots & \\ & \ddots & \ddots & \beta_k \\ & & \beta_k & \alpha_k \end{pmatrix}$$

a $V_k := [v_1, v_2, \ldots, v_k]$. Navíc platí $V_k^T V_k = I$. V přesné aritmetice by algoritmus skončil s $\beta_i = 0$ pro nějaké i. V konečné aritmetice se ale toto většinou nestane a proto je potřeba nějakého jiného zastavovacího kritéria, více o zastavovacích kritériích je zde v [5], v této práci se jimi ale nebudeme zabývat.

Nyní předpokládejme, že chceme vyřešit systém rovnic $Bx = c$, kde B je symetrická pozitivně definitní. Následující odvození bylo převzato z [5] a upraveno pro potřeby práce. Přenásobením (1.1) vektorem y_k splňující $x_k = V_k y_k$, a jehož poslední prvek je η_k, dostaneme $BV_k y_k = V_k T_k y_k + \beta_{k+1} v_{k+1} \eta_k$. Jelikož platí $V_k(\beta_1 e_1) = b$ z definice (konkrétně druhý řádek algoritmu), tak pokud y_k a x_k jsou definovány pomocí rovnic

$$\begin{aligned} T_k y_k &= \beta_1 e_1, \\ x_k &= V_k y_k, \end{aligned} \tag{1.2}$$

budeme mít $Bx_k = b + \eta_k \beta_{k+1} v_{k+1}$ s dobrou přesností. Tedy x_k je možno volit jako přesné řešení perturbované soustavy rovnic a bude řešit původní rovnici, pokud je $\eta_k \beta_{k+1}$dostatečně malé. Argumenty zmíněné výše nejsou úplné, ale dodávají dobrou motivaci pro definování posloupnosti vektorů $\{x_k\}$ dle (1.2).

1.2 Bidiagonalizace

Na konci minulé sekce jsme se dostali k motivaci pro definování posloupností vektorů x_k. Chceme řešit soustavu $Bx = c$ pro $B = A^T A, c = A^T = b$. Jedna z možností dle [3] je použití Krylovových prostorů, tedy aby vektor x_k patřil do příslušného Krylovova prostoru $\mathcal{K}_k(A^T A, A^T r_0)$. Lanczosova tridiagonalizace provádí ortogonální transformaci reálné symetrické matice do symetrické tridiagonální podoby. Golub-Kahanova bidiagonalizace provádí ortogonální redukci nesymetrické obdélníkové matice do horní či dolní bidiagonální podoby. Předchozí použití Lanczosova algoritmu nám dá podobu bidiagonalizačního procesu díky Golubovi a Kahanovi. Budeme jej nazývat *Bidiag1*. Vztah mezi tridiagonalizací a bidiagonalizací je k nalezení v [5].

Jako vstupní data uvažujeme matici $A \in \mathbb{R}^{n \times m}$ s plnou sloupcovou hodností a vektor pravých stran $b \in \mathbb{R}^n$.

Algoritmus 2 *Bidiag*1 [3]

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

Normalizační koeficienty $\alpha_i > 0$ a $\beta_i > 0$ jsou dle [3] vybrány tak, že $\|u_i\| = \|v_i\| = 1$. Algoritmus v přesné aritmetice skončí pro $\alpha_i = 0$ nebo $\beta_i = 0$ pro nějaké i.

Když zadefinujeme 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 $\mathcal{K}_k(A^T A, A^T r_0)$, navíc sloupečky matice U_k tvoří ortonormální bázi Krylovova prostoru $\mathcal{K}_k(AA^T, Ar_0)$ jak je popsáno v [1].

Když z koeficientů α_i a β_i sestavíme matici

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

tak můžeme napsat rekurenční vztahy, jako ty uvedené v [5]:

$$U_{k+1}(\beta_1 e_1) = b, \tag{1.3}$$

$$AV_k = U_{k+1}B_k, \tag{1.4}$$

$$A^T U_{k+1} = V_k B_k^T + \alpha_{k+1} v_{k+1} e_{k+1}^T. \tag{1.5}$$

Pokud zleva přenásobíme vztah (1.4) a následně za vzniklé $A^T U_{k+1}$ bychom dosadili ze vztahu (1.5), dostali bychom vztah (1.1) pro $B = A^T A$ a $T_k = B_k^T B_k$.

Pokud bychom použili přesnou aritmetiku, dostali bychom $U_{k+1}^T U_{k+1} = I$ a $V_k^T V_k = I$, jak je popsáno v [5].

1.3 Algoritmus LSQR

Výpočty v bidiagonalizačním algoritmu 2 lze použít ke spočtení problému nejmenších čtverců, tedy $\min\|b - Ax\|$. Všude uvažujeme počáteční volbu $x_0 = 0$. Následující odvození od (1.6) až po (1.9) je převzato z [5]. Označme

$$\begin{aligned} x_k &= V_k y_k, & (1.6)\\ r_k &= b - Ax_k, & (1.7)\\ t_{k+1} &= \beta_1 e_1 - B_k y_k, & (1.8) \end{aligned}$$

pro nějaký vektor y_k. Následně pomocí

$$\begin{aligned} r_k &\overset{(1.7)}{=} b - Ax_k \\ &\overset{(1.3)}{=} U_{k+1}(\beta_1 e_1) - Ax_k \\ &\overset{(1.6)}{=} U_{k+1}(\beta_1 e_1) - AV_k y_k \\ &\overset{(1.4)}{=} U_{k+1}(\beta_1 e_1) - U_{k+1}B_k y_k \\ &= U_{k+1}(\beta_1 e_1 - B_k y_k) \\ &\overset{(1.8)}{=} U_{k+1} t_{k+1}. \end{aligned}$$

lze odvodit vztah

$$r_k = U_{k+1} t_{k+1}. \tag{1.9}$$

Jelikož chceme $\|r_k\|$ minimální a jelikož je navíc U_{k+1} teoreticky ortonormální, ihned připadá v úvahu zvolit y_k tak, aby minimalizovalo $\|t_{k+1}\|$. Tím pádem se dostáváme k problému nejmenších čtverců

$$\min \|\beta_1 e_1 - B_k y_k\|, \tag{1.10}$$

který lze řešit například pomocí QR rozkladu, což nám dává základ pro LSQR algoritmus. Konkrétně bude QR rozklad dle [1] spočten pomocí Givensových rotací, v algoritmu 3 se jedna o řádky 11 až 18. V každém kroku algoritmu se spočtou koeficienty c,s,α,β, aby $-s\alpha + c\beta = 0$, kde $(\alpha^2 + \beta^2)^{1/2}$ je délka vektoru (α, β).

Pokud bychom nebrali v potaz zaokrouhlovací chyby, tak by ve vztahu (1.1) platilo $V_k^T V_k = I$ a algoritmus by se ukončil s $\beta_{k+1} = 0$ nebo $\alpha_{k+1} = 0$ pro nějaké k, jak je uvedeno v [5]. Nicméně vztah (1.1) platí až na úroveň strojové přesnosti.

Jelikož algoritmus LSQR je postaven na základě bidiagonalizačního algoritmu 2, tak i zde dle [6] tvoří sloupečky V_k ortonormální bázi Krylovova prostoru $\mathcal{K}_k(A^T A, A^T r_0)$.

Algoritmus 3 LSQR [5]

input A, b
$\beta_1 u_1 = b$
$\alpha_1 v_1 = A^T u_1$
$w_1 = v_1$
$x_0 = 0$
$\bar{\phi}_1 = \beta_1$, $\bar{\rho}_1 = \alpha_1$
for $k = 1,2,\ldots$ **do**
    $\beta_{k+1} u_{k+1} = A v_k - \alpha_k u_k$
    $\alpha_{k+1} v_{k+1} = A^T u_{k+1} - \beta_{k+1} v_k$

    $\rho_k = (\bar{\rho}_k^2 + \beta_{k+1}^2)^{1/2}$
    $c_k = \bar{\rho}_k / \rho_k$
    $s_k = \beta_{k+1} / \rho_k$

    $\theta_{k+1} = s_k \alpha_{k+1}$
    $\bar{\rho}_{k+1} = -c_k \alpha_{k+1}$
    $\phi_k = c_k \bar{\phi}_k$
    $\bar{\phi}_{k+1} = s_k \bar{\phi}_k$

    $x_k = x_{k-1} + (\phi_k / \rho_k) w_k$
    $w_{k+1} = v_{k+1} - (\theta_{k+1} / \rho_k) w_k$
end for

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

$$\begin{aligned} w_i^T A^T A w_j = 0, \quad i \neq j, \\ v_{k+1} \perp \mathcal{K}_k(A^T A,\, A^T r_0). \end{aligned} \tag{1.11}$$

2. CGLS

2.1 Motivace pomocí metody sdružených gradientů

Nyní diskutujme algoritmus CG dle [7] a to včetně algoritmu 4. Nejprve uvažujme zadanou soustavu rovnic $Bx = c$. Budeme předpokládat matici $B \in \mathbb{R}^{n\times n}$ symetrickou pozitivně definitní a vektor pravých stran $c \in \mathbb{R}^n$.

Uvažujme aproximaci řešení x_k a zadefinujeme si pomocný funkcionál

$$F(x_k) = \frac{1}{2}\|x - x_k\|_B^2 - \frac{1}{2}\|x\|_B^2, \tag{2.1}$$

který ukazuje, že minimalizace hodnoty F v daném podprostoru $\mathbb{R}^n$ je matematicky ekvivalentní minimalizaci energetické normy $\|x-z\|_B$. Nechť x_0 je počáteční aproximace a nechť je posloupnost x_k konstruována pomocí předpisu

$$x_k = x_{k-1} + \gamma_{k-1}p_{k-1}, \quad k = 1, 2, \ldots, \tag{2.2}$$

kde p_k představuje směrový vektor v kroku k. Novou aproximaci řešení určíme pomocí nalezení minima $\|x-x_{k-1}-\gamma p_{k-1}\|_B$. Jednoduchými úpravami lze odvodit

$$\|x - x_k\|_B^2 = \|x - x_{k-1}\|_B^2 - 2\gamma_{k-1}p_{k-1}^T r_{k-1} + \gamma_{k-1}^2 p_{k-1}^T B p_{k-1} \tag{2.3}$$

a tedy minima v daném směru je dosaženo pro

$$\gamma_{k-1} = \frac{p_{k-1}^T r_{k-1}}{p_{k-1}^T B r_{k-1}}. \tag{2.4}$$

Nezávisle na volbě skaláru δ_k minimalizace podél přímky $x_{k-1} + \gamma p_{k-1}$ (viz (2.4)) implikuje

$$p_k^T r_k = r_k^T r_k + \delta_k p_{k-1}^T r_k = r_k^T r_k = \|r_k\|^2. \tag{2.5}$$

Směrové vektory p_k volíme jako $p_k = r_k + \delta_k p_{k-1}$, aby byly B-ortogonální. Důsledkem se může iterační proces hledání aproximace zastavit v jednom ze dvou případů: buď $p_k = 0$ a nebo $\gamma_k = 0$. V prvním případě platí

$$0 = p_k^T r_k = \|r_k\|^2,$$

tedy $r_k = 0$ a $Bx_k = c$. V druhém případě je opět $p_k^T r_k = 0$ se stejným výsledkem. Nyní už je možné sestavit algoritmus CG pro $Bx = c$:

Následně si uveďme pár vlastností dokázaných v [7] týkajících se vektorů z algoritmu CG:

$$\begin{aligned} p_{k-1}^T r_k &= 0, \\ p_{k-1}^T B p_k &= 0, \\ r_i^T r_j &= 0, \quad i \neq j, \\ p_i^T B p_j &= 0, \quad i \neq j, \\ r_k &\perp \mathcal{K}_k(B, r_0). \end{aligned} \tag{2.6}$$

Algoritmus 4 CG [7]

input B, c, x_0
$z_0 = c - Bx_0$
$p_0 = z_0$
for $k = 1,2,\ldots$ **do**
  $\gamma_{k-1} = \frac{z_{k-1}^T z_{k-1}}{p_{k-1}^T B p_{k-1}}$
  $x_k = x_{k-1} + \gamma_{k-1} p_{k-1}$
  $z_k = z_{k-1} - \gamma_{k-1} B p_{k-1}$
  $\delta_k = \frac{z_k^T z_k}{z_{k-1}^T z_{k-1}}$
  $p_k = z_k + \delta_k p_{k-1}$
end for

Navíc je také možno odvodit tento důležitý vztah pro k-tý Krylovovský prostor a k-té lineární obaly

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

Algoritmus 4 se dá modifikovat tak, že použijeme $B = A^T A$ a $c = A^T b$, vše ostatníme ponecháme beze změny, čímž vznikne následující algoritmus:

Algoritmus 5 modifikace CG

input $A^T A$, $A^T b$, x_0
$z_0 = A^T b - A^T A x_0$
$p_0 = z_0$
for $k = 1,2,\ldots$ **do**
  $\gamma_{k-1} = \frac{z_{k-1}^T z_{k-1}}{p_{k-1}^T A^T A p_{k-1}}$
  $x_k = x_{k-1} + \gamma_{k-1} p_{k-1}$
  $z_k = z_{k-1} - \gamma_{k-1} A^T A p_{k-1}$
  $\delta_k = \frac{z_k^T z_k}{z_{k-1}^T z_{k-1}}$
  $p_k = z_k + \delta_k p_{k-1}$
end for

2.2 Algoritmus CGLS

Nyní už stačí algoritmus 5 trochu upravit do podoby uvedené v [1], čímž vznikne algoritmus známý jako CGLS. Je dobré si povšimnout, že platí

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

Pro CGLS se dají dle 2.6 obdobným způsobem odvodit následující vztahy, které budou využity později:

$$\begin{aligned} p_i^T A^T A p_j = 0, \quad i \neq j, \\ A^T r_k \perp \mathcal{K}_k(A^T A,\, A^T r_0). \end{aligned} \tag{2.7}$$

Algoritmus 6 CGLS [1]

input A, b, x_0
$r_0 = b - Ax_0$
$z_0 = p_0 = A^T r_0$
for $k = 0,1,\ldots$ **do**
    $t_k = Ap_k$
    $\gamma_k = \|z_k\|^2 / \|t_k\|^2$
    $x_{k+1} = x_k + \gamma_k p_k$
    $r_{k+1} = r_k - \gamma_k t_k$
    $z_{k+1} = A^T r_{k+1}$
    $\delta_{k+1} = \|z_{k+1}\|^2 / \|z_k\|^2$
    $p_{k+1} = z_{k+1} + \delta_{k+1} p_k$
end for

3. Porovnání CGLS a LSQR

Věta 1. *Nechť $A \in \mathbb{R}^{n\times m}$je matice s plnou sloupcovou hodností, $b \in \mathbb{R}^n$ je vektor a uvažujme algoritmy 3 a 6. Pak je vektor v_k nenulový násobek z_k a vektor w_{k+1} je nenulový násobek p_k.*

Důkaz. Dle [5] jsou algoritmy matematicky ekvivalentní, tedy i aproximace x_k musí být stejné. Tedy v obou algoritmech se musí x_k rovnat. Toto lze rozepsat jako

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

a z toho plyne lineární závislost p_k a w_{k+1}.

Jelikož v_k a z_k patří do stejného k-tého Krylovova prostoru a navíc platí, že $v_k \perp \mathcal{K}_k(A^TA,\, A^Tr_0)$ a $z_k \perp \mathcal{K}_k(A^TA,\, A^Tr_0)$, tak máme jednoznačně určenou lineární závislost vektorů v_k a z_k.

□

4. Numerické experimenty

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

Vlivem konečné aritmetiky dochází k tomu, že chyba aproximace měřena například normou rezidua začne stagnovat na určité úrovni, té říkáme *hladina limitní přesnosti*. Aproximace x_k se pak již dále vlivem počítačové aritmetiky neblíží k přesnému řešení, [7].

Matice illc1033 a well1033 jsou speciálně vybrané matice ze stránky [2] tak, aby byl ukázán rozdíl mezi algoritmy LSQR a CGLS. Na stránce [2] je k nalezení databáze matic, včetně vlastností illc1033 a well1033.

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 obě matice ortonormální. Tedy stačí spočítat, jak blízko k ortogonalitě jsou, neboli spočíst $\|I - V_k^T V_k\|_F$ a $\|I - Z_k^T Z_k\|_F$.

V prvním experimentu uvažujeme Strakošovu matici dle [7]. Strakošova matice s parametry je diagonální matice a má v intervalu $[\lambda_1, \lambda_n]$ vlastní čísla

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

V tomto textu máme matici Strakoš24, což je Strakošova matice s parametry $\rho = 0.1, n = 24, \lambda_1 = 1, \lambda_n = 24$.

Na obrázku 4.1 je možné vidět, že oba algortimy vcelku rychle dokonvergovaly k hladině limitní přesnosti, i když u šesté iterace se zdá být LSQR o trošku lepší.

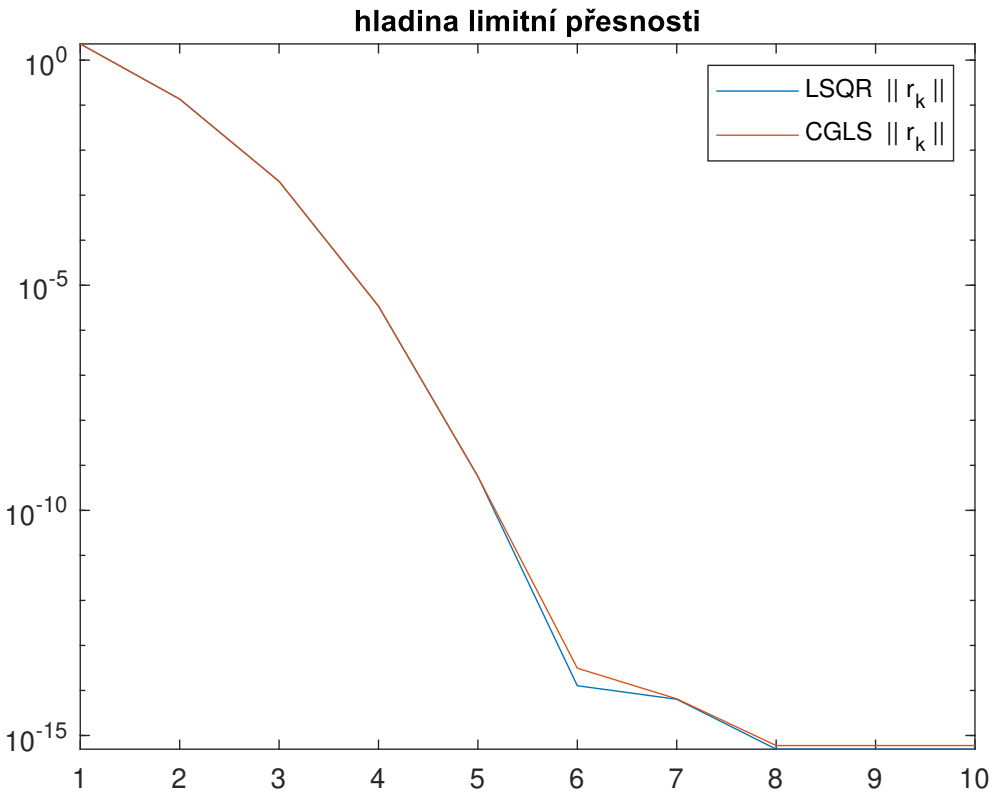
Na obrázku 4.2 je vidět, že ačkoliv oba algortimy mají skoro stejnou křivku, tak LSQR má o něco menší ztrátu ortogonality.

Na obrázku 4.3 je vidět převaha LSQR nad CGLS, protože LSQR konverguje k hladině limitní přesnosti rychleji, než CGLS.

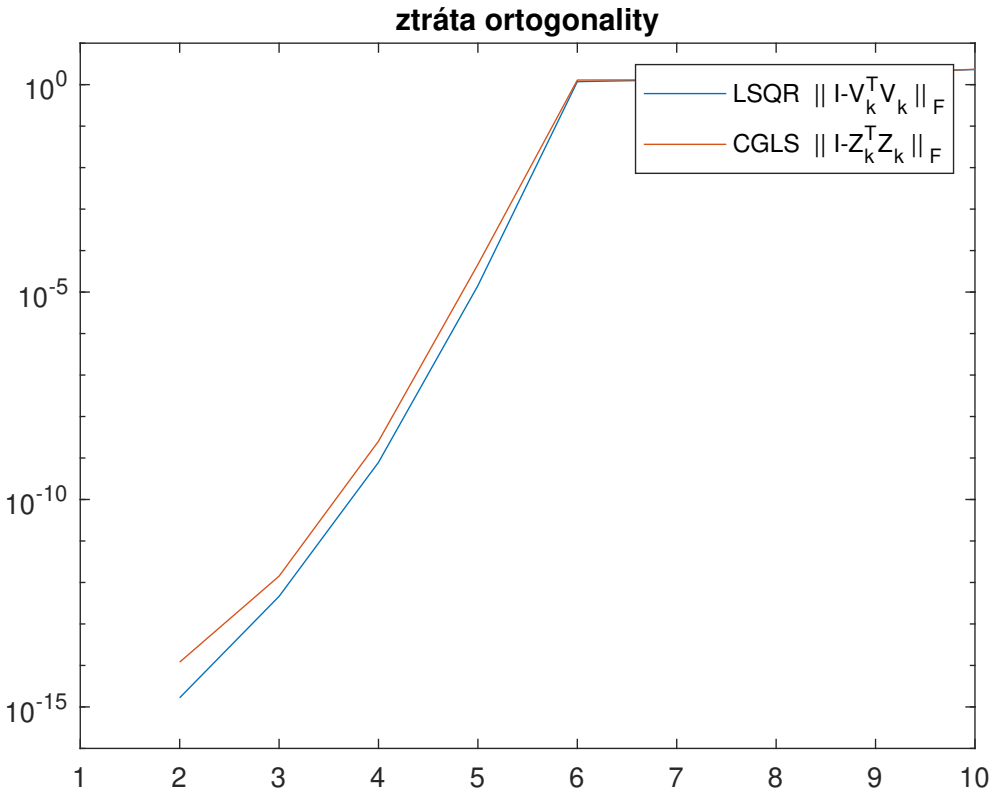
Na obrázku 4.4 je vidět, že ztráta ortogonality je pro oba algoritmy v podstatě totožná, tudíž v tomto ohledu jsou oba algoritmy stejně dobré.

Na obrázku 4.5 je vidět, že CGLS dosáhne hladiny limitní přesnosti dříve, než LSQR, tedy v tomto případě je CGLS lepší než LSQR.

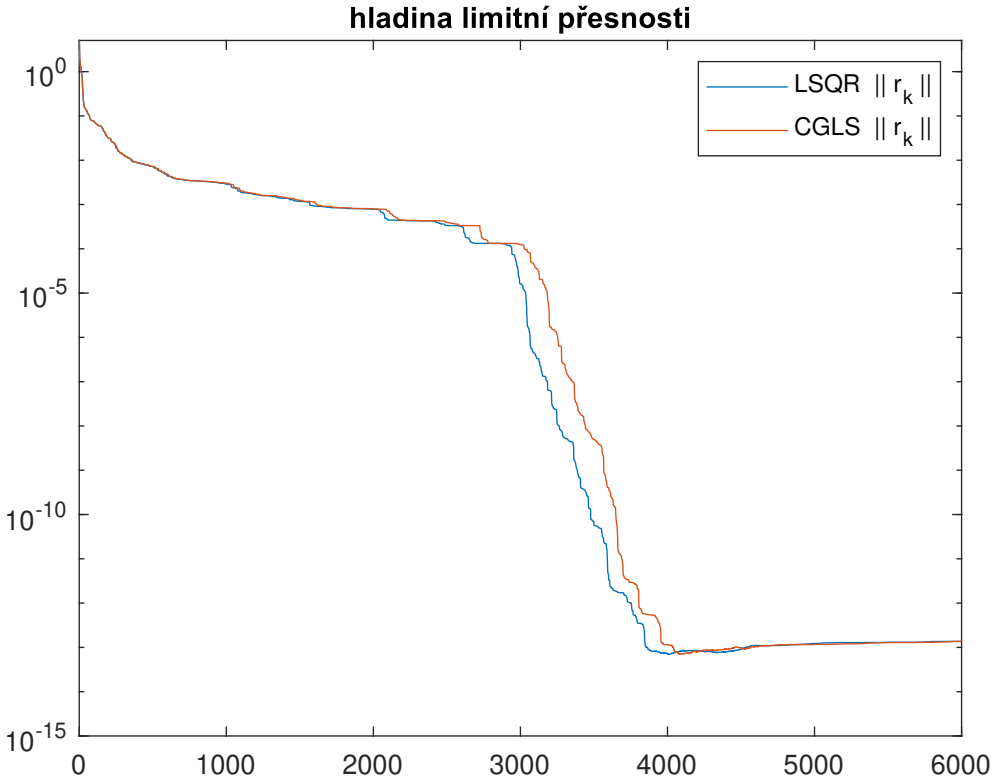
Na obrázku 4.6 je vidět, že ztráta ortogonality je pro oba algoritmy v podstatě totožná, tudíž v tomto ohledu jsou oba algoritmy stejně dobré.



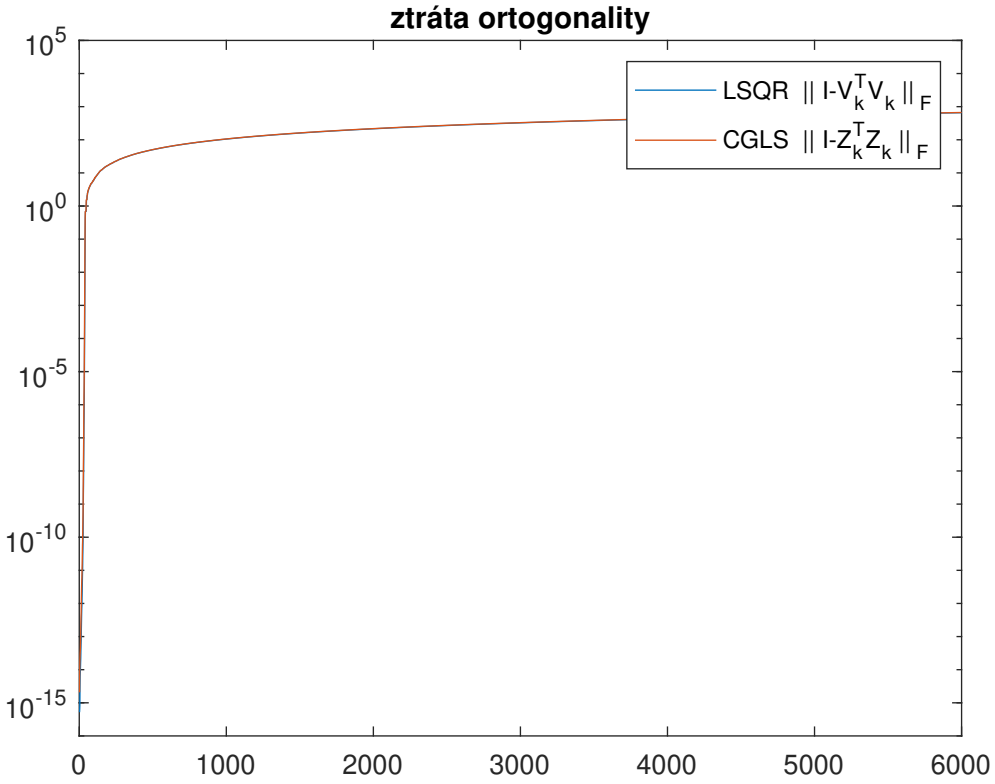
Obrázek 4.1: Matice Strakoš24, normy reziduí při počítání algoritmy CGLS a LSQR.



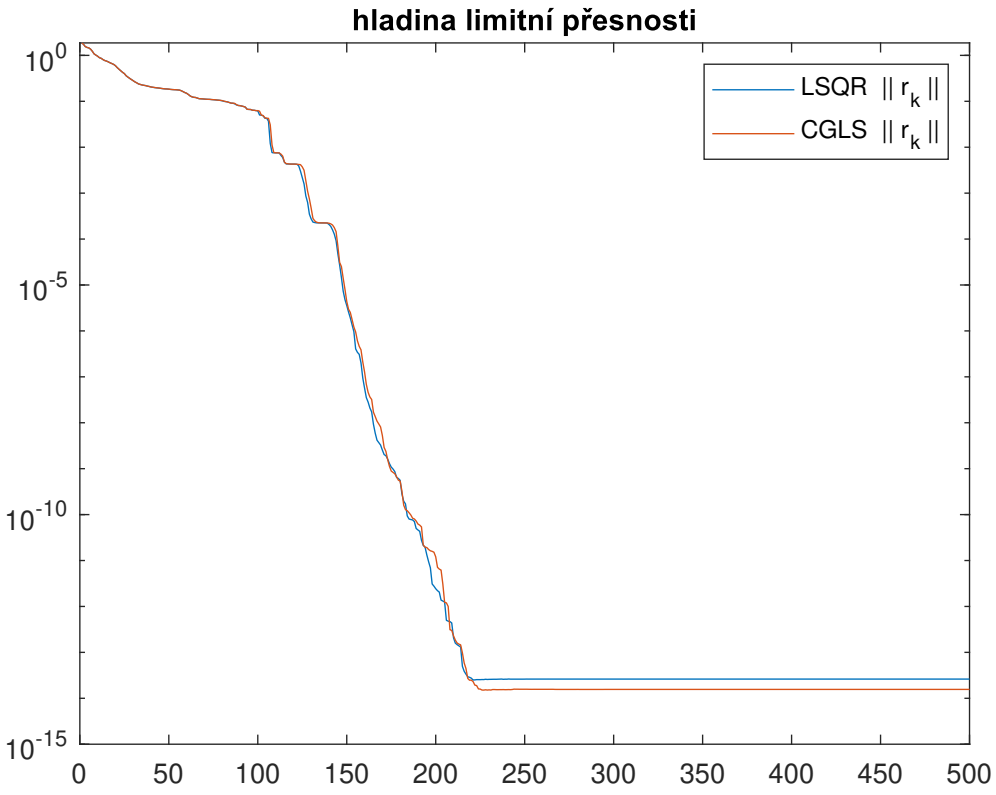
Obrázek 4.2: Matice Strakoš24, ztráta ortogonality při počítání algoritmy CGLS a LSQR.



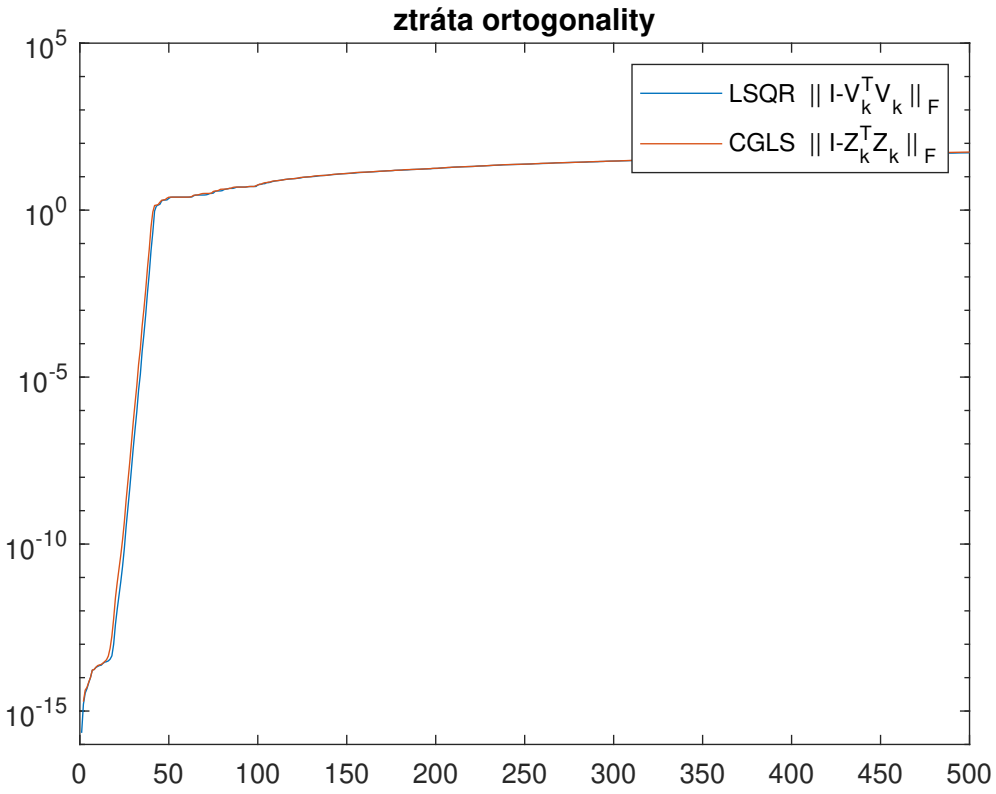
Obrázek 4.3: Matice illc1033, normy reziduí při počítání algoritmy CGLS a LSQR.



Obrázek 4.4: Matice illc1033, ztráta ortogonality při počítání algoritmy CGLS a LSQR.



Obrázek 4.5: Matice well1033, normy reziduí při počítání algoritmy CGLS a LSQR.



Obrázek 4.6: Matice well1033, ztráta ortogonality při počítání algoritmy CGLS a LSQR.

Závěr

V prvních kapitole jsme si ukázali odvození algoritmu LSQR pomocí Lanczosovy tridiagonalizace a Golub-Kahanovy bidiagonalizace. Následně jsme si v druhé kapitole ukázali odvození algoritmu CGLS pomocí algoritmu CG a pár menších úprav. Byly také vytyčeny nejdůležitější vlastnosti vektorů z algoritmů LSQR a CGLS, především vztahy ke Krylovovským prostorům, aby bylo možné je mezi sebou porovnat.

Ve třetí kapitole jsme se podívali na porovnání algoritmů v přesné aritmetice. Byly porovnány pouze vektory, koeficienty by bylo také dobré porovnat pro bližší zjištění vztahů mezi vektory. Bylo by zajímavé najít ony vztahy mezi koeficienty, zda by nemohly pomoci k novým objevům v oblasti řešení problému nejmenších čtverců, to se ale už do práce z časových důvodů nevešlo.

Ve čtvrté kapitole jsme se podívali na chování algoritmů při praktických výpočtech. Na uvedených příkladech bylo demonstrováno, že nelze z příkladů jednoznačně určit, že by jeden algoritmus byl lepší než druhý. Zodpovědět, ve kterých oblastech je výhodnější upřednostnit jeden algoritmus před druhým, je sice nad rámec práce, ale rozhodně stojí tuto debata zmínit.

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.

[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] 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.

[5] 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.

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

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