UNIVERZITA KOMENSKÉHO V BRATISLAVE
FAKULTA MATEMATIKY, FYZIKY A INFORMATIKY

KOLINEÁRNE ŠKÁLOVANIE V KVÁZINEWTONOVSKÝCH METÓDACH

BAKALÁRSKA PRÁCA

2013 Pavol Hronský

UNIVERZITA KOMENSKÉHO V BRATISLAVE
FAKULTA MATEMATIKY, FYZIKY A INFORMATIKY

KOLINEÁRNE ŠKÁLOVANIE V KVÁZINEWTONOVSKÝCH METÓDACH

BAKALÁRSKA PRÁCA

Študijný program:	Ekonomická a finančná matematika
Študijný odbor:	1114 Aplikovaná matematika
Školiace pracovisko:	Katedra aplikovanej matematiky a štatistiky
Vedúci práce:	doc. RNDr. Milan Hamala, CSc.

Bratislava 2013 **Pavol Hronský**

Univerzita Komenského v Bratislave
Fakulta matematiky, fyziky a informatiky

ZADANIE ZÁVEREČNEJ PRÁCE

Meno a priezvisko študenta: Pavol Hronský
Študijný program: ekonomická a finančná matematika (Jednoodborové štúdium, bakalársky I. st., denná forma)
Študijný odbor: 9.1.9. aplikovaná matematika
Typ záverečnej práce: bakalárska
Jazyk záverečnej práce: slovenský

Názov: Kolineárne škálovanie v kvázinewtonovských metódach

Cieľ: 1. Naštudovať najnovšie metódy kolineárneho škálovania a niektoré z nich naprogramovať.
2. Numerickým experimentom porovnať ich efektívnosť s BFGS metódou.

Vedúci: doc. RNDr. Milan Hamala, CSc.
Katedra: FMFI.KAMŠ - Katedra aplikovanej matematiky a štatistiky
Vedúci katedry: prof. RNDr. Daniel Ševčovič, CSc.

Dátum zadania: 11.10.2012

Dátum schválenia: 03.11.2012 doc. RNDr. Margaréta Halická, CSc.
garant študijného programu

.....
študent

.....
vedúci práce

Poďakovanie Na tomto mieste by som chcel poďakovať vedúcemu bakalárskej práce doc. RNDr. Milanovi Hamalovi, CSc. za venovaný čas, trpezlivosť, odbornú pomoc, cenné rady a pripomienky, ktoré významným spôsobom prispeli k vypracovaniu tejto bakalárskej práce.

Abstrakt

HRONSKÝ, Pavol: *Kolineárne Škálovanie v Kvázinewtonovských Metódach* [Bakalárska práca], Univerzita Komenského v Bratislave, Fakulta Matematiky, Fyziky a Informatiky, Katedra Aplikovanej Matematiky a Štatistiky, Školitel: doc. RNDr. Milan Hamala, CSc., Bratislava, 2013, 61 s.

Práca sa zaoberá metódami riešenia úloh na voľný extrém. Popisuje kvázinewtonovské metódy a detailnejšie rozoberá ich nadstavbu z oblasti kolineárneho škálovania. Kvázinewtonovské metódy sú zastúpené priamou BFGS metódou a metódy kolineárneho škálovania modifikovanou metódou navrhnutou Ariyawansom. Následne sa snaží porovnať efektívnosť vybraných metód na dopredu stanovených kritériach (čas, počet interácii a presnosť), za pomoci nami naprogramovaných algoritmov v programovacom jazyku MATLAB. Práca popisuje koncept numerických experimentov od potreby generovania rôznych typov úloh so známym optimálnym riešením, cez možnosti voľby metód na výpočet veľkosti optimálneho kroku, až po samotné experimentovanie na sériách náhodne vygenerovaných homogénnych, konvexných úlohách netriviálnych rozmerov a porovnávanie ich riešení.

Kľúčové slová: kvázinewtonovské metódy, BFGS metóda, metóda kolineárneho škálovania, kónický model, zlatý rez, kubická interpolácia, kónická interpolácia

Abstract

HRONSKÝ, Pavol: *Collinear Scaling in Quasinewton Methods* [Bachelor thesis], Comenius University in Bratislava, Faculty of mathematics, physics and informatics, Department of Applied Mathematics and Statistics, Supervisor: doc. RNDr. Milan Hamala, CSc., Bratislava, 2013, 61 p.

This thesis deals with methods solving problems of unconstrained optimization. It describes quasinewton methods and analyses their upgrade in the field of collinear scaling in detail. Quasinewton methods are represented by direct BFGS method and collinear scaling methods by modified method designed by Ariyawansa. Furthermore, it tries to compare the effectivness of chosen methods on in advance stated criteria (computing time, number of iterations and accuracy), with the help of our programmed algorithms in MATLAB programming language. Thesis describes the concept of the numerical experiments from need of generating several different types of problems with know optimum point, through options of choice of method solving the optimal length of the step, up to the numerical experiment itself applied on sets of randomly generated homogeneous, convex problems with nontrivial dimension and their solution comparison.

Key words: quasinewton methods, BFGS method, collinear scaling algorithm, conic model, golden ratio, cubic interpolation, conic interpolation

OBSAH

Obsah

ZOZNAM TABULIEK

Zoznam tabuliek

ÚVOD

Úvod

V dnešnej dobe sa čoraz častejšie stretávame s požiadavkou optimalizácie, či už v teórií, alebo v praxi. Riešenie optimalizačných úloh sa využíva v praxi vo viacerých odvetviach, napríklad pri modelovaní rôznych fyzikálnych javov, pri plánovaní nákladov, pri optimalizovaní portfólia a ďalších.

V optimalizačných úlohách sa spravidla stretávame s ohraničeniami v tvare rovníc alebo nerovníc. Tieto úlohy nazývame úlohami na viazaný extrém. Riešenie úloh na viazaný extrém je výpočtovo náročné, takéto úlohy sa však dajú upraviť na postupnosť úloh bez ohraničení, teda postupnosť úloh na voľný extrém. Na transformáciu sa používajú viaceré metódy, napríklad metódy sedlového bodu alebo metódy vnútorného bodu.

Za predpokladu, že vieme efektívne riešiť úlohy na voľný extrém nám takáto transformácia značne uľahčí cestu k požadovaným výsledkom. Preto je v tomto zmysle optimalizačná úloha bez ohraničení, resp. úloha na voľný extrém základným stavebným kameňom všetkých optimalizačných techník. V našej práci sa budeme zaoberať úlohou na voľný extrém, na ktorú vieme transformovať aj zložitejšie typy úloh. Najpoužívanejšími metódami na riešenie takých úloh sú kvázinewtonovské metódy. Vychádzajú z pôvodnej Newtonovej metódy, avšak ich algoritmus namiesto počítania Hessovej matice druhých derivácií využíva iba jej aproximáciu pomocou gradientov a tým značne znižuje výpočtovú náročnosť algoritmu. Kvázinewtonovské metódy využívajú kvadratický model, teda aproximujú okolie bodu, v ktorom sa nachádzame, kvadratickou funkciou.

V roku 1980 predstavil William C. Davidon vo svojom článku [4] metódy kolineárneho škálovania ako efektívnu nadstavbu kvázinewtonovských metód. Metódy kolineárneho škálovania využívajú kónické aproximácie. Výhodou kolineárnej transformácie oproti kvázinewtonovským metódam je schopnosť využitia informácie aj o funkčných hodnotách minimalizovanej funkcie, čo kvázinewtonovské metódy nedokážu. V 1990 publikoval Ariyawansa článok [2], v ktorom vylepšil Davidonov pôvodný algoritmus.

Cieľom predkladanej bakalárskej práce je experimentálne porovnať najnovší variant metódy kolineárneho škálovania Ariyawansu [2] s BFGS – kvázinewtonovskou metódou.

ÚVOD

V prvej kapitole sformulujeme úlohu na voľný extrém, ktorú budeme riešiť. Predstavíme základné metódy jej riešenia a rozoberieme podstatu kvázinewtonovských metód.

V druhej kapitole sa venujeme metódam kolineárneho škálovania. Uvádzame základné poznatky týchto metód a objasňujeme spojitosť s kvázinewtonovskými metódami. Taktiež navrhujeme algoritmus, použitý v neskorších častiach tejto práce.

Tretiu kapitolu venujeme numerickým experimentom. Uvádzame nami zvolený spôsob generovania náhodných kvadratických, bikvadratických a polynomiálnych funkcií n- premenných. Spomíname aj výpočet optimálneho kroku a samotné algoritmy, použité v numerických experimentoch. Vysvetľujeme priamo časti zdrojového kódu nášho porovnávacieho programu. Poslednú časť tejto kapitoly venujeme prezentácii výsledkov porovnávania.

1 Formulácia úlohy a metódy jej riešenia

1.1 Formulácia úlohy

V našej práci sa budeme zaoberať minimalizačnou úlohou na voľný extrém, ktorú symbolicky budeme zapisovať takto:

$$\min \{f(x)|x \in \mathbb{R}^n\}. \tag{1.1.1}$$

O funkcii $f(x) : \mathbb{R}^n \to \mathbb{R}$ budeme predpokladať, že má spojité druhé parciálne derivácie a je konvexná. Optimálne riešenie úlohy (1.1.1) budeme označovať symbolom $\hat{x} \in \mathbb{R}^n$.

Vo všeobecnosti vieme úlohy typu (1.1.1) riešiť iba iteračne, t.j. vychádzajúc z nejakého štartovacieho bodu $x_0 \in \mathbb{R}^n$ generujeme postupnosť bodov $\{x_k\}_1^\infty$, ktorá konverguje k optimálnemu riešeniu $\hat{x}$.

Obvykle sa postupnosť bodov $\{x_k\}$ generuje podľa nasledovnej schémy:

$$x_{k+1} = x_k + \lambda_k s_k,$$

kde bod $x_k \in \mathbb{R}^n$ je už známy bod, $0_n \neq s_k \in \mathbb{R}^n$ je tzv. spádový smer v ktorom funkcia $f(x)$ klesá a $\lambda_k > 0$ je dĺžka kroku v smere s_k. Presnejšie povedané, smer $s_k \in \mathbb{R}^n$ nazývame spádovým smerom funkcie $f(x)$ v bode $x_k \in \mathbb{R}^n$, ak existuje kladné číslo $\hat{\lambda}$ také, že pre všetky $0 < \lambda < \hat{\lambda}$ platí

$$f(x_k + \lambda s_k) < f(x_k).$$

Spádový smer sa spravidla generuje takto:

$$s_k = -H_k^\intercal \nabla f(x_k), \tag{1.1.2}$$

kde $H_k = H_k^\intercal$ je kladne definitná matica a $\nabla f(x_k)$ je gradient funkcie $f(x)$ v bode x_k. Špeciálne ak vo formuli (1.1.2) zoberieme $H_k = \mathbb{I}$, takýto smer sa nazýva Cauchyovský

a je to smer najväčšieho spádu. Pre newtonovský smer definujeme H_k v (1.1.2) ako

$$H_k = \left[\nabla^2 f(x_k)\right]^{-1},$$

kde $[\nabla^2 f(x_k)]$ je Hessova matica funkcie $f(x)$ v bode x_k. Newtonovský smer s_k sa získa riešením sústavy lineárnych rovníc

$$\nabla^2 f(x_k) s_k = -\nabla f(x_k).$$

Hessova matica je však náročná na výpočet, a preto sa v praxi preferujú tzv. kvázinewtonovské metódy, ktoré postupným generovaním matíc $\{B_j\}_1^n$ aproximujú Hessovu maticu v bode x_k, t.j. $B_k \approx [\nabla^2 f(x_k)]$. Postupnosť matíc $\{B_j\}_1^n$ sa generuje podľa schémy:

$$B_{j+1} = B_j + \Delta B, \tag{1.1.3}$$

pričom korekčná matica ΔB sa vytvára pre každú kvázinewtonovskú metódu iným spôsobom.

Pri voľbe kroku chceme, aby bolo nové riešenie x_{k+1} lepšie ako staré riešenie x_k v zmysle poklesu funkčnej hodnoty, t.j. $f(x_{k+1}) < f(x_k)$. Musíme teda zvoliť krok λ_k optimálne, aby sme na polpriamke $x_k + \lambda_k s_k$ minimalizovali funkciu $f(x)$. Nájsť optimálny krok teda znamená nájsť optimálne riešenie úlohy

$$\min \{\varphi(\lambda) \equiv f\left(x_k + \lambda s_k\right) | \lambda \in \mathbb{R}\}. \tag{1.1.4}$$

Takáto voľba kroku nám spravidla urýchľuje konvergenciu k optimálnemu riešeniu.

1.2 Kvázinewtonovské metódy

Vyššie spomenutá iteračná schéma sa využíva vo všetkých kvázinewtonovských metódach na optimalizáciu funkcie n premenných. Každá z týchto metód využíva pre aproximáciu účelovej funkcie v okolí bodu $x_k \in \mathbb{R}^n$ kvadratickú funckiu, od ktorej sa všetky vlastnosti metód odvádzajú. Využívaná kvadratická funkcia má tvar:

$$q(x_k + s) \approx f(x_k) + g_k^\intercal s + \frac{1}{2} s^\intercal G_k s. \tag{1.2.1}$$

Špecifikum kvázinewtonovských metód je fakt, že netrepotrebujeme poznať tvar Hessovej matice. Pomocou (1.1.3) sme schopní vytvoriť dostatočne dobrú aproximáciu Hessovej matice len za pomoci dvoch bodov x_{k+1} a x_k a ich príslušných gradientov g_{k+1} resp. g_k. Metóda generovania postupnosti matíc $\{B_k\}_1^\infty$ sa nazýva priamou, pretože aproximuje priamo Hessovu maticu.

Spádový smer budeme teda voliť ako riešenie systému lineárnych rovníc

$$B_k s_k = -g_k.$$

Matice B_k spĺňajú predpoklady kladnej definitnosti a symetrie popísané v [5]. Medzi najznámejšie kvázinewtonovské metódy patria DFP, BFGS a SR1.

V našej práci sa budeme zaoberať priamou BFGS metódou. Aby sme mohli uviesť vzťahy, ktoré platia pre túto metódu, musíme si najskôr zaviesť dve nové pomocné premenné:

$$p_k = x_{k+1} - x_k, \quad y_k = g_{k+1} - g_k.$$

Za pomoci týchto premenných môžeme definovať tvar korekčnej matice ΔB_k, ktorá sa dosadzuje do vzťahu (1.1.3) nasledovne:

$$\Delta B_k = \frac{y_k y_k^\intercal}{y_k^\intercal p_k} - \frac{B_k p_k p_k^\intercal B_k}{p_k^\intercal B_k p_k}. \tag{1.2.2}$$

Klasické metódy ako Cauchyho metóda, Newtonova metóda alebo aj kvázinewtonovské metódy nevyužívajú užitočnú informáciu o funkčných hodnotách $f(x)$ v bodoch

x_{k+1}, resp. x_k, ktoré častokrát potrebujeme poznať kvôli kontrole chodu toho daného algoritmu.

2 Metódy kolineárneho škálovania

2.1 Úvod do kolineárneho škálovania

Ako bolo spomenuté v úvode, Davidon publikoval v [4] návrh, akým vylepšiť spôsob výpočtu minima úlohy (1.1.1). Rozdiel spočíva v rôznej aproximácii účelovej funkcie. Zatiaľ čo newtonovské a kvázinewtonovské metódy sú založené na aproximácii účelovej funkcie funkciou (1.2.1), Davidon navrhol použiť v okolí bodu $x_k \in \mathbb{R}^n$ aproximáciu kónickou funkciou

$$F(s) \equiv f\left(x_k + s\right) \approx f_k + \frac{g_k^\intercal s}{1 + b^\intercal s} + \frac{\frac{1}{2} s^\intercal A_k s}{\left(1 + b^\intercal s\right)^2}. \tag{2.1.1}$$

Funkcia $F(s)$ spĺňa kľúčové interpolačné vlastnosti:

$$F(0_n) = f_k, \tag{2.1.2}$$

$$\nabla F(0_n) = g_k. \tag{2.1.3}$$

Metóda aproximovania kónickými funkciami sa inak nazýva aj metóda kolineárneho škálovania.

Definícia 2.1. Hladká funkcia sa nazýva kónickou, ak je podielom kvadratickej funckie a štvorca afinnej funkcie, t.j.:

$$c(x) = \frac{\frac{1}{2} x^\intercal A x + b^\intercal x + \gamma}{\left(1 + d^\intercal x\right)^2}, \quad d^\intercal x \neq -1. \tag{2.1.4}$$

V tomto bode však nie je zrejmé, že aj keď sa funkcie (2.1.1) a (2.1.4) podobajú, sú rovnakého typu. Ukážme si teda, že (2.1.1) je skutočne kónická funkcia.

Lema 1. *Funkcia tvaru*

$$h(x) = \gamma + \frac{g^\intercal x}{1 + d^\intercal x} + \frac{\frac{1}{2} x^\intercal A x}{\left(1 + d^\intercal x\right)^2}, \quad d^\intercal x \neq -1$$

je kónická.

Dôkaz.

$$h(x) = \frac{1}{(1+d^\intercal x)^2}\left[(1+d^\intercal x)^2\gamma + (1+d^\intercal x)(g^\intercal x) + \frac{1}{2}x^\intercal Ax\right],$$
$$(1+d^\intercal x)^2 h(x) = \frac{1}{2}x^\intercal Ax + g^\intercal x + (d^\intercal x)(g^\intercal x) + \gamma + 2\gamma(d^\intercal x) + \gamma(d^\intercal x)^2,$$
$$(1+d^\intercal x)^2 h(x) = \frac{1}{2}x^\intercal Ax + g^\intercal x + (x^\intercal d)(g^\intercal x) + \gamma + 2\gamma(d^\intercal x) + \gamma x^\intercal dd^\intercal x,$$
$$(1+d^\intercal x)^2 h(x) = \left[\frac{1}{2}x^\intercal Ax + \frac{1}{2}(x^\intercal d)(g^\intercal x) + \frac{1}{2}(x^\intercal g)(d^\intercal x) + \gamma x^\intercal dd^\intercal x\right] + \left[g^\intercal x + 2\gamma(d^\intercal x)\right] + \gamma,$$
$$(1+d^\intercal x)^2 h(x) = x^\intercal\left[\frac{1}{2}A + \frac{1}{2}(dg^\intercal + gd^\intercal) + \gamma dd^\intercal\right]x + \left[g + 2\gamma d\right]^\intercal x + \gamma.$$

Ak nahradíme

$$\tilde{A} = \left[\frac{1}{2}A + \frac{1}{2}(dg^\intercal + gd^\intercal) + \gamma dd^\intercal\right]$$

a

$$\tilde{g} = \left[g + 2\gamma d\right],$$

dostávame

$$(1+d^\intercal x)^2 h(x) = x^\intercal\tilde{A}x + \tilde{g}^\intercal x + \gamma,$$
$$h(x) = \frac{x^\intercal\tilde{A}x + \tilde{g}^\intercal x + \gamma}{(1+d^\intercal x)^2}. \tag{2.1.5}$$

□

Vidíme, že funkcia funkcia (2.1.1) je skutočne kónická.

2.2 Vlastnosti kónickej funkcie

Uvažujme kónickú funckiu (2.1.1). Chceme nájsť jej bod minima $\hat{s} \in \mathbb{R}^n$. Z matematickej analýzy vieme, že funkcia viacerých premenných nadobúda extrém práve vtedy, keď jej gradient $\nabla F(s) = 0_n$. Gradient kónickej funcie je:

$$\begin{aligned}
\nabla F(s) &= \frac{(1 + b^\intercal s)\, g_k - (g_k^\intercal s) b}{(1 + b^\intercal s)^2} + \frac{(1 + b^\intercal s)^2 A_k s - (1 + b^\intercal s)\,(s^\intercal A_k s) b}{(1 + b^\intercal s)^4} \qquad (2.2.1)\\
&= \frac{(1 + b^\intercal s)^2 g_k - (1 + b^\intercal s)\,(g_k^\intercal s) b + (1 + b^\intercal s) A_k s - (s^\intercal A_k s) b}{(1 + b^\intercal s)^3}\\
&= \frac{(1 + b^\intercal s)^2 g_k + (1 + b^\intercal s) A_k s - (1 + b^\intercal s)\,(b s^\top) g_k - (b s^\intercal) A_k s}{(1 + b^\intercal s)^3}\\
&= \frac{\left[(1 + b^\intercal s)\,\mathbb{I}\right]\left[(1 + b^\intercal s)\, g_k + A_k s\right] - \left[b s^\top\right]\left[\,(1 + b^\intercal s)\, g_k + A_k s\right]}{(1 + b^\intercal s)^3}\\
&= \frac{\left[(1 + b^\intercal s)\,\mathbb{I} - b s^\top\right]\left[\,(1 + b^\intercal s)\, g_k + A_k s\right]}{(1 + b^\intercal s)^3}\\
&= \frac{1}{1 + b^\intercal s}\left[\mathbb{I} - \frac{b s^\top}{1 + b^\intercal s}\right]\left[g_k + \frac{A_k s}{1 + b^\intercal s}\right]. \qquad (2.2.2)
\end{aligned}$$

Z definície vieme, že $\frac{1}{1+b^\intercal s}$ je kladná konštanta. Ukážeme, že matica $\left[\mathbb{I} - \frac{b s^\top}{1+b^\intercal s}\right]$ je regulárna, teda jej determinant je nenulový:

Lema 2. *Nech $a, b \in \mathbb{R}^n, a^\intercal b \neq 1$. Potom platí:*

$$\det\left[\mathbb{I} - b a^\intercal\right] = (1 - a^\intercal b).$$

Dôkaz. Z vektorov $a, b \in \mathbb{R}^n$ vytvoríme blokovú maticu

$$M = \begin{pmatrix} \mathbb{I} & b \\ a^\intercal & 1 \end{pmatrix}.$$

V matici budeme realizovať riadkové úpravy, ktoré nemenia jej determinant. Ako prvé začneme tým, že eliminujeme prvý prvok v druhom riadku. Dostávame hornú

trojuholníkovú maticu:

$$M_1 = \begin{pmatrix} \mathbb{I} & b \\ 0_n^\intercal & (1 - a^\intercal b) \end{pmatrix} => \det M_1 = (1 - a^\intercal b).$$

Teraz budeme postupovať opačne, aby sme dostali dolnú trojuholníkovú maticu:

$$M_2 = \begin{pmatrix} (\mathbb{I} - ba^\intercal) & 0 \\ a^\intercal & 1 \end{pmatrix} => \det M_2 = (\mathbb{I} - ba^\intercal).$$

Je zrejmé, že musí platiť $\det M = \det M_1 = \det M_2$, z čoho dostávame

$$\det (\mathbb{I} - ba^\intercal) = (1 - a^\intercal b).$$

□

V našom prípade položme

$$a = \frac{s}{1 + b^\intercal s},$$

potom

$$a^\intercal b = \frac{s^\intercal b}{1 + b^\intercal s}, \quad (1 - a^\intercal b) = \frac{1}{1 + b^\intercal s} > 0,$$

a teda dostávame

$$\det \left[\mathbb{I} - \frac{bs^\intercal}{1 + b^\intercal s} \right] = \frac{1}{1 + b^\intercal s} > 0.$$

Na základe uvedeného z (2.2.2) platí, že $\nabla F(s) = 0_n$ práve vtedy keď

$$\left[g_k + \frac{A_k s}{1 + b^\intercal s} \right] = 0_n. \tag{2.2.3}$$

Úpravami vzťahu (2.2.3) dostaneme bod, ktorý funkciu minimalizuje:

$$\begin{aligned}
g_k + \frac{A_k s}{1 + b^\intercal s} &= 0_n, \\
A_k s &= -(1 + b^\intercal s) g_k, \\
s &= -(1 + b^\intercal s) A_k^{-1} g_k, \qquad (2.2.4) \\
b^\intercal s &= -(1 + b^\intercal s) b^\intercal A_k^{-1} g_k, \\
b^\intercal s &= -b^\intercal A_k^{-1} - b^\intercal s b^\intercal A_k^{-1} g_k, \\
b^\intercal s + b^\intercal s b^\intercal A_k^{-1} g_k &= -b^\intercal A_k^{-1} g_k, \\
b^\intercal s (1 + b^\intercal A_k^{-1} g_k) &= -b^\intercal A^{-1} g_k, \\
b^\intercal s &= \frac{-b^\intercal A_k^{-1} g_k}{1 + b^\intercal A_k^{-1} g_k}, \\
1 + b^\intercal s &= 1 + \frac{-b^\intercal A_k^{-1} g_k}{1 + b^\intercal A_k^{-1} g_k}, \\
1 + b^\intercal s &= \frac{1}{1 + b^\intercal A_k^{-1} g_k}. \qquad (2.2.5)
\end{aligned}$$

Dosadením (2.2.5) do (2.2.4) dostávame finálnu podobu bodu minima kónickej funkcie

$$\hat{s} = \frac{-A^{-1} g}{1 + b^\intercal A^{-1} g}. \qquad (2.2.6)$$

V doterajšom texte tejto časti práce sme sa venovali kónickej funkcii, ktorou pri výpočte minima aproximujeme okolie bodu $x_k \in \mathbb{R}^n$. Avšak, ešte sme neukázali, že odvodený postup naozaj vedie k výpočtu minima funkcie, pretože nájdený extrém sme ešte nijak neklasifikovali. Aby sa skutočne jednalo o výpočet minima, je potreba aby $\nabla^2 F(\hat{s})$ bola kladne definitná.

Výpočet Hessovej matice funkcie $F(s)$ budeme odvádzať zo vzťahu (2.2.1):

$$\begin{aligned}\nabla^2 F(s) =& \frac{-g_k b^\intercal}{(1+b^\intercal s)^2} - b\left[\frac{g_k(1+b^\intercal s)^2 - 2(g_k^\intercal s)(1+b^\intercal s)b}{(1+b^\intercal s)^4}\right]^\intercal \\ &+ \frac{(1+b^\intercal s)^2 A_k - 2(1+b^\intercal s)A_k s b^\intercal}{(1+b^\intercal s)^4} \\ &- b\left[\frac{2A_k s(1+b^\intercal s)^3 - 3(1+b^\intercal s)^2 s^\intercal A_k s b}{(1+b^\intercal s)^6}\right]^\intercal \\ =& \frac{-b g_k^\intercal}{(1+b^\intercal s)^2} + \frac{-g_k b^\intercal}{(1+b^\intercal s)^2} + \frac{2(g_k^\intercal s)bb^\intercal}{(1+b^\intercal s)^3} + \frac{A}{(1+b^\intercal s)^2} \\ &- \frac{2A_k s b^\intercal}{(1+b^\intercal s)^3} - \frac{abs^\intercal A_k}{(1+b^\intercal s)^3} + \frac{3s^\intercal A_k s bb^\intercal}{(1+b^\intercal s)^4} \\ =& \frac{1}{(1+b^\intercal s)^2}[A - bg_k^\intercal - g_k b^\intercal] + \frac{2}{(1+b^\intercal s)^3}[g_k^\intercal s bb^\intercal - A_k s b^\intercal - bs^\intercal A_k] \\ &+ \frac{3}{(1+b^\intercal s)^4} s^\intercal A_k s bb^\intercal. \end{aligned} \tag{2.2.7}$$

Po dosadení extrémalneho bodu (2.2.6) do vzťahu (2.2.7) dostaneme Hessovu maticu v bode $\hat{s}$. Najprv si však spravíme niekoľko pomocných výpočtov:

$$b^\intercal \hat{s} = \frac{-b^\intercal A_k^{-1} g_k}{1 + b^\intercal A_k^{-1} g_k} \Rightarrow 1 + b^\intercal \hat{s} = \frac{1}{1 + b^\intercal A_k^{-1} g_k} \Rightarrow \frac{1}{1+b^\intercal \hat{s}} = 1 + b^\intercal A_k^{-1} g_k, \tag{2.2.8}$$

$$g_k^\intercal \hat{s} = \frac{-g_k^\intercal A_k^{-1} g_k}{1 + b^\intercal A_k^{-1} g_k}, \tag{2.2.9}$$

$$\hat{s}^\intercal A_k = \frac{-g^\intercal}{1 + b^\intercal A_k g_k}, \tag{2.2.10}$$

$$A_k \hat{s} = \frac{-g_k}{1 + b^\intercal A_k^{-1} g_k}, \tag{2.2.11}$$

$$\hat{s}^\intercal A_k \hat{s} = \frac{g_k^\intercal A_k^{-1} g_k}{(1 + b^\intercal A_k^{-1} g_k)^2}. \tag{2.2.12}$$

Za pomoci vzťahov (2.2.8)-(2.2.12) dosadených do (2.2.7) môžeme vypočítať Hessovu maticu v bode $\hat{s}$.

$$\begin{aligned}\nabla^2 F(\hat{s}) =&(1+b^\intercal A_k g_k)^2[A_k - g_k b^\intercal - b g_k^\intercal] + 3(1+b^\intercal A_k g_k)^4 \left[\frac{g_k A_k^{-1} g_k}{(1+b^\intercal A_k g_k)^2} b b^\intercal\right] \\ &+ 2(1+b^\intercal A_k g_k)^3 \left[\frac{-g_k^\intercal A_k g_k b b^\intercal}{1+b^\intercal A_k g_k} + \frac{b g_k^\intercal}{1+b^\intercal A_k g_k} + \frac{g_k b^\intercal}{1+b^\intercal A_k g_k}\right] \\ =&(1+b^\intercal A_k g_k)^2[A_k - b g_k^\intercal - g_k b^\intercal - 2 g_k^\intercal A_k^{-1} g_k b b^\intercal + 2 g_k b^\intercal + 2 b g_k^\intercal + 3 g_k^\intercal A^{-1} g_k b b^\intercal] \\ =&(1+b^\intercal A_k g_k)^2[A_k + b g_k^\intercal + g_k b^\intercal + g_k^\intercal A^{-1} g_k b b^\intercal]. \end{aligned} \tag{2.2.13}$$

Matica definovaná vzťahom (2.2.13) by mala byť kladne definitná, aby funkcia nadobúdala v bode $\hat{s}$ minimum.

Lema 3. *Matica (symetrická)*

$$M = A + g b^\intercal + b g^\intercal + (g^\intercal A^{-1} g) b b^\intercal$$

je kladne definitná.

Dôkaz.

$$\begin{aligned}\forall z \in \mathbb{R}^n : z^\intercal M z &= z^\intercal A z + z^\intercal g b^\intercal z + z^\intercal b g^\intercal z + (g^\intercal A^{-1} g) z^\intercal b b^\intercal z \\ &= z^\intercal A z + 2(z^\intercal g)(b^\intercal z) + (g^\intercal A^{-1} g)(b^\intercal z)^2 \\ &= \left[z^\intercal A z - \frac{(g^\intercal z)^2}{g^\intercal A^{-1} g}\right] + \left[\sqrt{g^\intercal A^{-1} g}(b^\intercal z) + \frac{g^\intercal z}{\sqrt{g^\intercal A^{-1} g}}\right]. \end{aligned} \tag{2.2.14}$$

Zrejme druhá zátvorka je nezáporná. Nezápornosť prvej zátvorky vyplýva z Cauchyho nerovnosti:

$$\forall x, y \in \mathbb{R}^n : (x^\intercal y)^2 \leq (x^\intercal x)(y^\intercal y), \tag{2.2.15}$$

v ktorej položíme

$$x = A^{\frac{1}{2}} z, \quad y = A^{-\frac{1}{2}} g. \tag{2.2.16}$$

Potom

$$(x^\intercal y) = g^\intercal z, \quad (x^\intercal x) = z^\intercal A z, \quad (y^\intercal y) = g^\intercal A^{-1} g,$$

a teda

$$(g^\intercal z)^2 \leq (z^\intercal Az)(g^\intercal A^{-1}g),$$

odkiaľ vyplýva nezápornosť prvej zložky v (2.2.14). Tým sme dokázali kladnú semidefinitnosť matice M.

Aby sme dokázali kladnú definitnosť matice M, musíme vylúčiť možnosť, že obidve zátvorky v (2.2.14) sú súčasne nulové. Druhá zátvorka bude nulová práve vtedy, keď $b^\intercal z = 0$ a $g^\intercal z = 0$. V Cauchyho nerovnosti (2.2.15) nastáva rovnosť práve vtedy, ak vektory $x, y \in \mathbb{R}^n$ sú kolineárne, t.j. $\exists \mu \neq 0$ také, že $x = \mu y$, čo podľa (2.2.16) znamená, že

$$z = \mu A^{-1}g. \tag{2.2.17}$$

Ak vzťah (2.2.17) dosadíme do (2.2.16) dostávame $(g^\intercal z) = \mu g^\intercal A^{-1}g = 0$, čo protirečí kladnej definitnosti matice A. □

2.3 Kvázinewtonovské metódy vyplývajúce z kolineárneho škálovania

Kónickú funkciu (2.1.1) môžeme preformulovať takto:

$$F(s) = f + g^\intercal \left(\frac{s}{1 + b^\intercal s}\right) + \frac{1}{2}\left(\frac{s}{1 + b^\intercal s}\right)^\intercal A \left(\frac{s}{1 + b^\intercal s}\right). \tag{2.3.1}$$

Z tvaru (2.3.1) priamo vidieť, že substitúcia

$$t = \frac{s}{1 + b^\intercal s} \quad \text{resp.} \quad s = \frac{t}{1 - b^\intercal t} \tag{2.3.2}$$

transformuje kónickú funkciu (2.1.1) na kvadratickú funkciu

$$F(t) = f + g^\intercal t + \frac{1}{2} t^\intercal A t. \tag{2.3.3}$$

Poznámka 1. (K odvodeniu inverznej transformácie (2.3.2)). Inverznú tranformáciu (2.3.2) vynásobíme zľava vektorom $b^\intercal$:

$$b^\intercal t = \frac{b^\intercal s}{1 + b^\intercal s},$$

odkiaľ úpravami dostaneme

$$b^\intercal t = 1 - \frac{1}{1 + b^\intercal s},$$

$$\frac{1}{1 + b^\intercal s} = (1 - b^\intercal t).$$

Potom

$$t = \frac{s}{1 + b^\intercal s},$$

$$t = (1 - b^\intercal t)s,$$

z čoho vyplýva

$$s = \frac{t}{1 - b^\intercal t}.$$

Substitúciu (2.3.2) nazývame kolineárnym škálovaním kvadratickej funkcie (2.3.3).

Uvedená transformácia (2.3.2) umožnuje zovšeobecniť kvázinewtonovské metódy z kvadratického modelu na kónický model. Idea spočíva v nasledovnej úvahe:

Kvázinewtonovskú podmienku odvodenú z kvadratického modelu (2.3.3) v priestore $t \in \mathbb{R}^n$ transforumujeme kolineárnycm škálovaním (2.3.2) do priestoru $s \in \mathbb{R}^n$ a získame analogickú kvázinewtonovskú podmienku pre kónický model (2.3.1).

$$\left\{\text{v priestore } t \in \mathbb{R}^n : B_{k+1}p_k = y_k\right\} => \left\{\text{v priestore } s \in \mathbb{R}^n : B_{k+1}p_k = \frac{1}{\gamma_k}r_k\right\}, \tag{2.3.4}$$

kde

$$p_k = x_{k+1} - x_k,$$

$$y_k = g_{k+1} - g_k,$$

$$r_k = g_{k+1} - \frac{1}{\gamma_k^2}g_k$$

a

$$\gamma_k = \frac{1}{f_k - f_{k+1} + \rho_k}p_k^\intercal g_k, \quad \rho_k = \sqrt{(f_k - f_{k+1})^2 - (p_k^\intercal g_k)(p_k^\intercal g_{k+1})}.$$

Technická realizácia je pomerne zložitá a detaily sú uvedené v monografii [8, str. 326-335].

Praktickým prínosom uvedenej transformácie (2.3.2) je dodatočný parameter $b \in \mathbb{R}^n$, ktorý môže „absorbovať“ ďalšie interpolačné podmienky, napr. rovnosť funkčných hodnôt minimalizovanej funkcie $f(x)$ a aproximujúcej funkcie $F(s)$.

Všeobecný tvar kolineárnej transformácie je

$$t = \frac{Js}{1 + b^\intercal s} \quad \text{resp.} \quad s = \frac{J^{-1}t}{1 - b^\intercal J^{-1}t}. \tag{2.3.5}$$

Základný algoritmus založený na kolineárnom škálovaní

Na základe vyššie odvodenej teórie a zovšeobecnenej transformácie kónickej funkcie na kvadratickú, bol odvodený v [8] jeden zo základných algoritmov kolineárneho škálovania, ktorého jadrom sa stala modifikácia BFGS kvázinewtonovskej metódy. Jeho schématický postup je nasledovný:

Vstup: štartovací bod $x_0 \in \mathbb{R}$, funkčný hodnota $f_0 = f(x_0)$, gradient $g_0 = \nabla f(x_0)$, symetrická, kladne definitná matica $C_0 = \mathbb{I}$.

Cyklus for $k = 0, 1, 2, \ldots$

1. IF $[g_k = 0]$ then STOP $[\hat{x} = x_k]$
2. Riešime sústavu lineárnych rovníc $B_k s_k = -g_k$. Výsledkom je spádový smer $s_k \in \mathbb{R}^n$.
3. V smere s_k vypočítame dĺžku optimálneho kroku $\lambda_k = \operatorname{argmin} \{\varphi(\lambda) \equiv f(x_k + \lambda s_k) \,|\lambda \in \mathbb{R}\}$.
4. $p_k = \lambda_k s_k$.
5. Vypočítame nový bod $x_{k+1} = x_k + p_k$.
6. Vypočítame funkčnú hodnotu $f_{k+1} = f(x_{k+1})$ a gradient g_{k+1}.
7. $\rho_k^2 = (f_k - f_{k+1})^2 - (g_{k+1}^\intercal s_k)(g_k^\intercal s_k)$.
8. $\gamma_k = \frac{-1}{f_k - f_{k+1} + \rho_k} g_k^\intercal s_k$.
9. $y_k = \gamma_k g_{k+1} - \frac{1}{\gamma_k} g_k$.
10. Vypočítame novú maticu

$$C_{k+1} = \gamma_k^2 \left[\left(\mathbb{I} - \frac{p_k y_k^\intercal}{p_k^\intercal y_k} \right) C_k \left(\mathbb{I} - \frac{p_k y_k^\intercal}{p_k^\intercal y_k} \right) + \frac{p_k p_k^\intercal}{p_k^\intercal y_k} \right].$$

Koniec cyklu.

2.4 Navrhovaný algoritmus kolineárneho škálovania

Ariyawansa vo svojom článku [2] navrhol algoritmus kolineárneho škálovania, ktorý odvodil na základe nasledujúcich predokladov:

A1) Nech množina $\mathbb{Z} \subseteq \mathbb{R}^n$ je otvorená a konvexná a nech funkcia $f(x) : \mathbb{Z} \to \mathbb{R}$ je silnokonvexná.

A2) Nech matica $\nabla^2 f(x)$ je na množine $\mathcal{L}(x_0) = \{x \in \mathbb{Z} | f(x) < f(x_0)\}$ Lipšicovská.

Oproti pôvodnému algoritmu sme v tejto práci spravili drobnú zmenu. Navhrnutý algoritmus je nasledovný:

Vstup: štartovací bod $x_0 \in \mathbb{Z}^n$, funkčný hodnota $f_0 = f(x_0)$, gradient $g_0 = \nabla f(x_0)$, symetrická, kladne definitná matica $B_0 = \mathbb{I}$.

Cyklus for $k = 0, 1, 2, \ldots$

1. IF $[g_k = 0]$ then STOP $[\hat{x} = x_k]$.
2. Riešime sústavu lineárnych rovníc $B_k s_k = -g_k$. Výsledkom je spádový smer $s_k \in \mathbb{R}^n$.
3. V smere s_k vypočítame dĺžku optimálneho kroku $\lambda_k = \operatorname{argmin} \{\varphi(\lambda) \equiv f(x_k + \lambda s_k) | \lambda \in \mathbb{R}\}$.
4. $p_k = \lambda_k s_k$.
5. Vypočítame nový bod $x_{k+1} = x_k + p_k$.
6. Vypočítame funkčnú hodnotu $f_{k+1} = f(x_{k+1})$ gradient g_{k+1}.
7. $\rho_k = \sqrt{(f_k - f_{k+1})^2 - (g_{k+1}^\intercal s_k)(g_k^\intercal s_k)}$.
8. $\gamma_k = \frac{-1}{f_k - f_{k+1} + \rho_k} g_k^\intercal s_k$.
9. $\tilde{v}_k = \gamma_k p_k$.
10. $r_k = g_{k+1} - \frac{1}{\gamma_k^2} g_k$.
11. $w_k = \frac{r_k}{r_k^\intercal \tilde{v}_k} - \frac{B_k \tilde{v}_{k\cdot}}{\tilde{v}_k^\intercal B_k \tilde{v}_k}$.

12. IF $w_k = 0$ THEN

$$\theta_k = 0.$$

ELSE

$$\theta_k > \frac{1}{1 - \frac{\left(r_k^\intercal B_k^{-1} r_k\right)\left(\tilde{v}_k^\intercal B_k \tilde{v}_k\right)}{\left(\tilde{v}_k^\intercal r_k\right)^2}}.$$

13. Vypočítame novú maticu

$$B_{k+1} = B_k - \frac{B_k \tilde{v}_k \tilde{v}_k^\intercal B_k}{\tilde{v}_k^\intercal B_k \tilde{v}_k} + \frac{r_k r_k^\top}{\tilde{v}_k^\intercal r_k} + \theta_k \left(\tilde{v}_k^\intercal B_k \tilde{v}_k\right) w_k w_k^\top.$$

Koniec cyklu.

Poznámka 2. Zmena sa udiala pri voľbe dĺžky optimálneho kroku. Oproti pôvodnému algoritmu sa v našej praci budeme snažiť držať pôvodnej filozofie ako pri kvázinewtonovských metódach, kde hľadáme optimálny krok za pomoci metódy zlatého rezu alebo jednorozmernej interpolácie polynómom na lúči v spádovom smere.

V našom algoritme však aproximujeme okolie bodu x_k kónickou funkciou, preto sa budeme snažiť nájsť optimálny krok za pomoci opäť buď metódou zlatého rezu, alebo jednorozmernej kónickej interpolácie, ktorej postup je odvodený v [3].

3 Numerický experiment

3.1 Koncept experimentu

Cieľom numerického experimentu bude praktické porovnanie efektívnosti metódy kolineárneho škálovania s kvázinewtonovskou metódou BFGS. Experiment budeme aplikovať na troch rôznych typoch konvexných funkcií:

I. kvadratická funkcia,

II. bikvadratická funkcia,

III. polynomiálna funkcia,

ktoré budeme náhodne generovať s dopredu známym optimálnym bodom, ktorý budeme označovať $\hat{x} \in \mathbb{R}^n$. Generované funkcie budú netriviálnych rozmerov hodnôt $100 \leq n \leq 1000$. Pri niektorých typoch funkcií, napr. kvadratickej by bolo možné testovať aj väčšie rozmery, avšak najmä polynomiálna funkcia by pri vyšších rozmeroch zabrala príliš veľa času, čo by celý testovací proces brzdilo.

Pri samotnom porovnávaní budeme pozorovať niekoľko veličín. Najdôležitejšou porovnávanou veličinou bude čas t, za ktorý dokáže algoritmus úlohu vyriešiť. Čas výpočtu priamo súvisí s počtom iterácií *„p.it."* potrebných na dosiahnutie určitej presnosti ϵ. Nakoniec, zaujímavým porovnávacím kritériom je dosiahnutá presnosť, v závislosti od počtu iterácii $\delta = ||x_{\max} - \hat{x}|\,|$.

Uvedené charakteristiky budeme sledovať ako priemer veličín z homogénnych sérií náhodne vygenerovaných úloh.

V texte sa vyskytnú výňatky z kódu programovacieho jazyka MATLAB, ktoré priblížia čitateľovi samotnú realizáciu aktuálneho problému.

3.2 Generovanie funkcií so známym minimom

V nasledujúcej časti si spomenieme tri typy funkcií, na ktorých budeme vybrané algoritmy testovať. Pri každom type si ukážeme, že funkcia je konvexná. Ako sme spomínali, poznanie optimálneho riešenia je výhodou, lebo nám umožňuje vyhodnotiť kľučové porovnávacie kritérium, ktorým je výsledná presnosť riešenia. Preto sa pri generovaní najkôr sústredíme na vytvorenie optimálneho riešenia $\hat{x} \in \mathbb{R}^n$, od ktorého sa odvinie finálny tvar funkcie.

V našom programe je funkcia pre generovanie úloh všeobecná, t.j. na základe vstupného parametru bude generovať tú danú úlohu. Jednotlivé generátory popíšeme v príslušných častiach práce. Tu len nasleduje stručný náhľad na koncept MATLABovskej funkcie *Generate.m*, ktorá úlohy vytvára.

```
function [D,G,h,C,alpha,gamma,xHat,x0] = Generate(n,rho,func)
    %%% VSTUP: %%%
    % n \\ rozmer funkcie
    % rho \\ vzdialenost pociatocneho bodu od bodu optima
    % func = 'Quadratic', 'Biquadratic', 'Polynomial' \\ typ funkcie
    %%% VYSTUP %%%
    % D,G,h,C,alpha,gamma \\ matice a vektory, ktore definuju funkcie
    % xHat \\ optimalny bod           x0 \\startovaci bod
    switch func
        case 'Biquadratic'
            % ...
        case 'Quadratic'
            % ...
        case 'Polynomial'
            % ...
        otherwise
            disp('Zly vstupny parameter func!');
    end
end
```

3.2.1 Kvadratická funkcia n- premenných

Ako prvú uvádzame kvadratickú funkciu n- premenných, ktorej zápis je nasledovný:

$$f(x) = \frac{1}{2}x^T Gx + h^T x, \tag{3.2.1}$$

kde matica G je symetrická a kladne definitná.

Dvojnásobným derivovaním vzťahu (3.2.1) dostávame

$$\nabla^2 f(x) = G.$$

Vzhľadom na fakt, že matica G je definovaná ako symetrická a kladne definitná, nie je pochýb o tom, že funkcia (3.2.1) je konvexná.

Derivovaním vzťahu (3.2.1) dostávame:

$$\nabla f(x) = Gx + h. \tag{3.2.2}$$

Z prvej nutnej podmienky vieme, že vzťah (3.2.2) musí spĺňať

$$\nabla f(\hat{x}) = 0_n, \tag{3.2.3}$$

z čoho automaticky vyplýva podmienka pre $h \in \mathbb{R}^n$

$$h = -G\hat{x}. \tag{3.2.4}$$

Preto pri vytváraní funkcie typu (3.2.1) potrebujeme vytvoriť len optimálny bod a kladne definitnú maticu. Z poznatkov z lineárnej algebry vieme, že pre ľubovoľnú maticu $A \in \mathbb{R}^{n\times n}$ platí, že $A^T A = B$ je kladne semidefinitná matica. Pripočítaním identity $\mathbb{I}_n$ dosávame kladne definitnú maticu $M = B + \mathbb{I}_n$. Takto vzniknutá matica však môže byť *zle* podmienená. Podmienenosť matice udáva jej podmienené číslo, ktoré indikuje ako veľmi môže aj malá nepresnosť ovplyvniť celý výpočet. Z pohľadu numerických výpočtov bude pre nás dobré, aby bola vzniknutá matica *dobre* podmienená.

Podľa [7] sa pre kladne definitnú maticu A pomer

$$c(A) = \frac{\lambda_{\max}}{\lambda_{\min}}$$

nazýva podmieneným číslom matice, kde λ_i sú vlastné čísla matice A.

Aby sme čo najviac zamedzili chybám, ktoré budú vznikať pri numerických výpočtoch, budeme prihliadať aj na faktor podmieneného čísla, a preto všetky matice, ktoré budeme používať na generovanie funkcií, budú spĺňať $c(A) \leq 10^3$.

Poďme sa pozrieť, ako bude vyzerať samotné generovanie v programovacom jazyku MATLAB:

```
% Generovanie kvadratickej funkcie v tvare: f(x) = 1/2*x'*G*x +h'*x
% Optimalny bod
xHat = randi([-5,5],n,1)./10;
% Vytvaranie dobre podmienenej symetrickej, kladne definitnej matice
A = randi([-10,10],n,n);
[U,] = qr(A);
E = diag(randi([1,1000],n,1))./100;
G = U*E*U';
% Optimalizacia vektora h tak, aby xHat bol bod optima
h = - G*xHat;
% Startovaci bod
x0 = randi([-5,5],n,1);
x0 = xHat+rho*(x0)/norm(x0);
% Nulovanie parametrov, ktore v kvadratickej funkcii nevystupuju
D=zeros(n,n); C=zeros(n,1); alpha=0; gamma=0;
```

3.2.2 Bikvadratická funkcia n- premenných

Ako ďalšiu predstavujeme bikvadratickú funkciu n- premenných, ktorej zápis je nasledovný:

$$f(x) = \frac{1}{4}(x^T Dx)^2 + \frac{1}{2}x^T Gx + h^T x, \tag{3.2.5}$$

kde matica G je symetrická, kladne definitná a matica D je diagonálna s kladnými vlastnými číslami.

Dvojnásobným derivovaním vzťahu (3.2.5) dostávame

$$\nabla^2 f(x) = 2(Dx)(Dx)^T + (x^T Dx)D + G,$$

kde prvý sčítanec je určite aspoň kladne semidefinitná matica, druhý sčítanec tak isto a nakoniec je pripočítaná kladne definitná matica, čo dáva dokopy kladne definitnú maticu, z čoho vyplýva, že takto vygenerovaná funkcia $f(x)$ je konvexná.

Derivovaním vzťahu (3.2.5) dostávame:

$$\nabla f(x) = (x^T Dx)Dx + Gx + h. \tag{3.2.6}$$

Z prvej nutnej podmienky vieme, že vzťah (3.2.6) musí spĺňať

$$\nabla f(\hat{x}) = 0_n, \tag{3.2.7}$$

z čoho automaticky vyplýva podmienka pre $h \in \mathbb{R}^n$

$$h = -\left[(\hat{x}^T D\hat{x})D\hat{x} + G\hat{x}\right]. \tag{3.2.8}$$

Prepísaním do jazyka MATLAB dostávame:

```
% Generovanie bikvadratickej funkcie v tvare: f(x) = ...
   1/4*(x'*D*x)^2+1/2*x'*G*x +h'*x
% Optimalny bod
xHat = randi([-5,5],n,1)./10;
```

```
% Dobre podmienena diagonalna matica
D = diag(randi([1,1000],n,1))./100;
% Vytvaranie dobre podmienenej symetrickej, kladne definitnej matice
A = randi([-10,10],n,n);
[U,] = qr(A);
E = diag(randi([1,1000],n,1))./100;
G = U*E*U';
% Optimalizacia vektora h tak, aby xHat bol bod optima
h = -( (xHat'*D*xHat)*D*xHat + G*xHat);
% Startovaci bod
x0 = randi([-5,5],n,1);
x0 = xHat+rho*(x0)/norm(x0);
% Nulovanie parametrov, ktore v kvadratickej funkcii nevystupuju
C=zeros(n,1); alpha=0; gamma=0;
```

3.2.3 Polynomiálna funkcia n- premenných

Ako poslednú funkciu, na ktorej budeme vybrané metódy testovať uvádzame polynomiálnu funkciu v tvare:

$$f(x) = \left[\sum_{i=1}^{m} \alpha_i(\gamma_i + C_i^T x)^{2k_i}\right] + \frac{1}{2}x^T G x + h^T x, \tag{3.2.9}$$

kde G je symetrická, kladne definitná, $C \in \mathbb{R}^{m\times n}$, $\alpha > 0_m$, $\gamma \in \mathbb{R}^m$ a pre k_i platí, že $1 < l \leq m$ a $k_1 = 1, k_2 = 2, \ldots, k_l = l, k_{l+1} = 1, k_{l+2} = 2, \ldots$.

Dvojnásobným derivovaním vzťahu (3.2.9) dostávame

$$\nabla^2 f(x) = \left[\sum_{i=1}^{m} 2\alpha_i k_i(2k_i - 1)(\gamma_i + C_i^T x)^{2k_i-2} C_i C_i^T\right] + G,$$

čo sa dá v maticovom tvare prepísať ako

$$\nabla^2 f(x) = C^T D_2 C + G,$$

kde $D_2 = diag\left(2\alpha_i k_i(2k_i - 1)(\gamma_i + C_i^T x)^{2k_i-2}\right)$ je diagonálna matica s nezápornými členmi, z čoho vyplýva, že funkcia je konvexná.

Derivovaním vzťahu (3.2.5) dostávame:

$$\nabla f(x) = \left[\sum_{i=1}^{m} 2\alpha_i k_i(\gamma_i + C_i^T x)^{2k_i-1} C_i\right] + Gx + h = C^T D_1 e + Gx + h, \tag{3.2.10}$$

kde G je symetrická, kladne definitná matica, $D_1 = diag\left(2\alpha_i k_i(\gamma_i + C_i^T x)^{2k_i-1}\right)$ a $e^T = [1, 1, 1, \ldots, 1]$. Z prvej nutnej podmienky vieme, že vzťah (3.2.10) musí spĺňať

$$\nabla f(\hat{x}) = 0_n, \tag{3.2.11}$$

z čoho automaticky vyplýva podmienka pre $h \in \mathbb{R}^n$

$$h = -\left[C^T D_1 e + G\hat{x}\right]. \tag{3.2.12}$$

Prepísaním do jazyka MATLAB dostávame:

```
% Generovanie polynomialnej funkcie v tvare: f(x) = ...
    sum[alpha*(gamma+C'*x).^(2*k)]+1/2*x'*G*x +h'*x
% Optimalny bod
xHat = randi([-5,5],n,1);
% Vytvaranie dobre podmienenej symetrickej, kladne definitnej matice
A = randi([-10,10],n,n);
[U,] = qr(A);
E = diag(randi([1,1000],n,1))./100;
G = U*E*U';
% Druhy rozmer matice C
m = randi([1,n],1,1);
% Stanovuje maximalny exponent v zada ni na 10
l = 5;
% Vektor parnych exponentov
kk=0:m-1;
kk=2.*(mod(kk,l)+1);

C = randi([-5,5],m,n)./10;
% vektory alpha a gamma
alpha = randi([1,10],m,1);
gamma = randi([-5,5],m,1);
% Preddefinovanie vektora d1, pre rychlejsie pristupovanie k jeho ...
    zlozkam.
d1 = zeros(m,1);
% Vypocet diagonalnych prvkov matice D1
for i=1:m
    d1(i)=alpha(i).*kk(i).*(gamma(i)+C(i,:)*xHat).^(kk(i)-1);
end
% Optimalizacia vektora h tak, aby xHat bol bod optima
h = -G*xHat - C'*diag(d1)*ones(m,1);
% Startovaci bod
x0 = randi([-5,5],n,1);
x0 = xHat+rho*(x0)/norm(x0);
```

```
% Nulovanie parametrov, ktore v kvadratickej funkcii nevystupuju
D=zeros(n,n);
```

3.3 Výpočet optimálneho kroku

V skúmaných algoritmoch (BFGS a kolineárne škálovanie) používame iteračnú schému

$$x_{k+1} = x_k + \lambda_k s_k,$$

kde $s_k \in \mathbb{R}^n$ je spádový smer vytvorený príslušnou metódou a $\lambda_k > 0$ aproximuje optimálnu dĺžku kroku podľa:

$$\hat{\lambda} = \operatorname{argmin} \{\varphi(\lambda) \equiv f(x_k + \lambda s_k) \,|\lambda \in \mathbb{R}\}.$$

V samotných algoritmoch bude vystupovať funkcia *Lambda.m*, ktorá bude združovať všetky 3 typy voľby optimálneho kroku a na základe vstupného parametru sa zvolí jedna z možností. Výstavba funkcie *Lambda.m* je nasledovná:

```
function lambda = Lambda(D,G,h,C,alpha,gamma,x,s,method,LambdaEps)
    %%% VSTUP: %%%
    % D,G,h,C,alpha,gamma \\ matice a vektory definujuce funkciu
    % x \\ bod xk v ktorom sa v iteracnej scheme momentalne nachadzame
    % method = 'goldenRatio', 'conicInter', 'cubicInter' \\ typ ...
       metody na vypocet dlzky optimalneho kroku lambda
    % s \\ spadovy smer
    % Eps \\ zvolena presnost vysledku
    %%% VYSTUP: %%%
    % lambda \\ velkost optimalneho kroku
    switch method
        case 'goldenRatio'
            % ...
        case 'CubicInter'
            % rekurzivnu funkciu cubicInter si blizsie popiseme neskor
            lambda = cubicInter(D,G,h,C,alpha,gamma,x,0,1.5,s);
        case 'ConicInter'
            % rekurzivnu funkciu conicInter si blizsie popiseme neskor
            lambda = conicInter(D,G,h,C,alpha,gamma,x,0,1.5,s);
```

```
        otherwise
            disp('Wrong input in parameter "method"!');
    end
end
```

Zmena sa týkala voľby dĺžky optimálneho kroku. Ako základ sme si pre tieto experimenty pripravili klasickú metódu metódu zlatého rezu. Táto metóda je bližšie popísana v [5], a preto si len uvedieme samotné prevedenie v programovacom jazyku MATLAB:

```
%% metoda Zlateho Rezu
% zaciatocny interval <a,b>
a = 0;
b = 1.5;
% nastavenie konstant
t = double(sqrt(5)-1)/2;
c1 = a +(1-t)*(b-a);
c2 = a +t*(b-a);
% spustenie samotneho cyklu so startovacimi funkcnymi hodnotami
f1 = F(D,G,h,C,alpha,gamma,(x+c1*s));
f2 = F(D,G,h,C,alpha,gamma,(x+c2*s));
while ((b-a) > LambdaEps)
    if (f1 ≥ f2)
        a = c1;
        c1 = c2;
        c2 = a + t*(b-a);
        f1 = f2;
        f2 = F(D,G,h,C,alpha,gamma,(x+c2*s));
        if((b-a) < LambdaEps)
            lambda = c2;
            return;
        end
    elseif (f1 < f2)
        b = c2;
        c2 = c1;
        c1 = a+(1-t)*(b-a);
```

```
        f2 = f1;
        f1 = F(D,G,h,C,alpha,gamma,(x+c1*s));
        if((b-a) < LambdaEps)
            lambda = c1;
            return;
        end
    end
end
```

V metóde je kľúčové, akú jemnosť algoritmu povolíme, t.j. s akou presnosťou nájdeme veľkosť optimálneho kroku λ_k. To nám udáva premenná *LambdaEps*. Nie je však dobré nastavovať presnosť na ultra nízke hodnoty, napr. 10^{-6} a menej. Takáto presnosť neprinesie do výsledku skoro nič nové, naopak výrazne spomalí výpočtový proces. Preto v našej práci budeme ako konvenciu používať $LambdaEps = 10^{-3}$.

Budeme však pracovať aj so zložitejšími metódami, ktoré oproti metóde zlatého rezu skracujú čas potrebný na získanie optimálnej dĺžky kroku. Pre oba vybrané algoritmy použijeme rekurzívne interpolačné metódy im „šité“ na mieru. Obe metódy budú vychádzať z rovnakých základov. Na lúči v spádovom smere budeme na začiatok poznať body λ_0 a λ_1, funkčné hodnoty v daných bodoch f_0 a f_1 a derivácie v daných bodoch f'_0 a f'_1. Zároveň budeme predpokladať, že $0 \leq \lambda_0 < \lambda_1$ a $f'_0 < 0 < f'_1$. Pre kvázinewtonovskú metódu použijeme rekurzívnu metódu aproximácie kubickým polynómom odkonzultovanú v [6].

```
function [lambda] = cubicInter(D,G,h,C,alpha,gamma,x,l1,l2,s)
    %%% VSTUP: %%%
    % D,G,h,C,alpha,gamma \\ matice a vektory definujuce funkciu
    % x \\ bod xk v ktorom sa v iteracnej scheme momentalne nachadzame
    % s \\ spadovy smer
    % l1, l2 \\ pociatocne 2 body na luci v spadovom smere (0,1.5)
    %%% VYSTUP: %%%
    % lambda \\ velkost optimalneho kroku
    Eps = 0.001;
    % Vypocet funkcnych hodnot a derivacii v bodoch l1 a l2
```

```
    f1 = F(D,G,h,C,alpha,gamma,x+l1*s);
    f2 = F(D,G,h,C,alpha,gamma,x+l2*s);
    df1 = dF(D,G,h,C,alpha,gamma,x+l1*s)'*s;
    df2 = dF(D,G,h,C,alpha,gamma,x+l2*s)'*s;
    % Dielcie vypocty potrebne v interpolacnej formule
    f12 = (f2-f1)/(l2-l1);
    Z = (df1+df2-3*f12);
    W = sqrt(Z^2-df1*df2);
    % Interpolacna formula
    l = l1+(l2-l1)*(W+Z-df1)/(2*W+df2-df1);
    % Vypocet derivacie v novovzniknutom bode
    dfk = dF(D,G,h,C,alpha,gamma,x+l*s)'*s;
    % Test blizkosti bodov
    if(norm(l1-l2)<1e-10)
        lambda = l;
        return
    end
    % Rekurzia
    if(dfk < -Eps)
        lambda = cubicInter(D,G,h,C,alpha,gamma,x,l,l2,s);
    elseif(dfk > Eps)
        lambda = cubicInter(D,G,h,C,alpha,gamma,x,l1,l,s);
    else
        lambda = l;
    end
end
```

Na druhej strane, pre metódu kolineárneho škálovania budeme používať rekurzívnu metódu interpolácie kónickou funkciou, ktorej odvodenie je detailnejšie popísané v [3].

```
function [lambda] = conicInter(D,G,h,C,alpha,gamma,x,l1,l2,s)
    %%% VSTUP: %%%
    % D,G,h,C,alpha,gamma \\ matice a vektory definujuce funkciu
    % x \\ bod xk v ktorom sa v iteracnej scheme momentalne nachadzame
    % s \\ spadovy smer
```

```
    % l1, l2 \\ pociatocne 2 body na luci v spadovom smere (0,1.5)
    %%% VYSTUP: %%%
    % lambda \\ velkost optimalneho kroku
    Eps = 0.001;
    % Vypocet funkcnych hodnot a derivacii v bodoch l1 a l2
    f1 = F(D,G,h,C,alpha,gamma,x+l1*s);
    f2 = F(D,G,h,C,alpha,gamma,x+l2*s);
    df1 = dF(D,G,h,C,alpha,gamma,x+l1*s)'*s;
    df2 = dF(D,G,h,C,alpha,gamma,x+l2*s)'*s;
    % Dielcie vypocty potrebne v interpolacnej formule
    Δ = l2-l1;
    alfa = (f1-f2)*Δ;
    beta = df1*df2;
    R = ((alfa + (Δ/abs(Δ))*sqrt(alfa^2-beta))^3)/(beta*df1);
    Δk = -Δ/(1+R);
    % Interpolacna formula
    l = Δk + l2;
    % Vypocet derivacie v novovzniknutom bode
    dfk = dF(D,G,h,C,alpha,gamma,x+l*s)'*s;
    % Test blizkosti bodov
    if(norm(l1-l2)<1e-15)
        lambda = l;
        return
    end
    % Rekurzia
    if(dfk < -Eps)
        lambda = conicInter(D,G,h,C,alpha,gamma,x,l,l2,s);
    elseif(dfk > Eps)
        lambda = conicInter(D,G,h,C,alpha,gamma,x,l1,l,s);
    else
        lambda = l;
    end
end
```

3.4 Algoritmus I.: Kvázinewtonovská BFGS- priama

Prejdime k samotnému algoritmu priamej BFGS kvázinewtonovskej metódy. Stručne popíšeme jej schématický postup:

Vstup: štartovací bod $x_0 \in \mathbb{R}^n$, gradient $g_0 = \nabla f(x_0)$, symetrická, kladne definitná matica $B_0 = \mathbb{I}$.

Cyklus for $k = 0, 1, 2, \ldots$

1. If $g_k = 0$ then STOP $[\hat{x} = x_k]$.
2. Riešime sústavu lineárnych rovníc $B_k s_k = -g_k$. Výsledkom je spádový smer $s_k \in \mathbb{R}^n$.
3. V smere s_k vypočítame dĺžku optimálneho kroku $\lambda_k = \operatorname{argmin} \{\varphi(\lambda) \equiv f(x_k + \lambda s_k) \,|\lambda \in \mathbb{R}\}$.
4. $p_k = \lambda_k s_k$.
5. Vypočítame nový bod $x_{k+1} = x_k + p_k$.
6. Vypočítame gradient $g_{k+1} = \nabla f(x_{k+1})$ a vektor $y_k = g_{k+1} - g_k$.
7. Za pomoci vektorov p_k a y_k a matice B_k zostrojíme korekčnú maticu ΔB_k podľa (1.2.2).
8. Vypočítame novú maticu $B_{k+1} = B_k + \Delta B_k$.

Koniec cyklu.

Uvedený schématický postup prevedieme do jazyku MATLAB a popíšeme.

```
% Poznamka: hodnoty s koncovkou "s" su tie, ktore pocitame v kroku k. ...
   Hodnoty s koncovkou "k" su tie, ktore vypocitame ako k+1.
function [x,kolko] = BFGS(D,G,h,C,alpha,gamma,x0,xHat,Lmax,Eps)
    %%% VSTUP: %%%
    % D,G,h,C,alpha,gamma \\ matice a vektory definujuce funkciu
    % x0 \\ startovaci bod
    % xHat \\ optimalny bod
    % Lmax \\ maximalny pocet iteracii
```

```
% Eps \\ presnost
%%% VYSTUP: %%%
% x \\ vysledny bod
% kolko \\ vysledny pocet iteracii

% Pocitadlo iteracii
kolko = 0;
% Rozmer funkcie
n = length(xo);
% Nastavenie startovacich hodnot
B0 = eye(n);
Bs = B0;
xs = x0;
gs = dF(D,G,h,C,alpha,gamma,xs);
% Zaciatok cyklu
while(kolko < Lmax)
    % Riesime sustavu aby sme ziskali spadovy smer ss
    ss = -linsolve(Bs,gs);
    % Hladame dlzku optimalneho kroku metodou kubickej interpolacie
    lam = Lambda(D,G,h,C,alpha,gamma,xs,ss,'CubicInter',1e-3);
    % Vypocet noveho body, funkcnej hodnoty a gradientu v danom bode
    ps = lam*ss;
    xk = xs + ps;
    gk = dF(D,G,h,C,alpha,gamma,xk);
    Fk = Fvypis(D,G,h,C,alpha,gamma,xk);
    % Test konvergencie
    if(norm(gk) < Eps)
        x = xk;
        return
    end

    ys = gk-gs;

    % OPTIMALIZACIA vypoctovej narocnosti konstrukcie novej ...
       matice B_{k+1}
    % Povodny tvar:
```

```
        % Bk = Bs+(ys*ys')/(ps'*ys)-(Bs*ps*ps'*Bs)/(ps'*Bs*ps);
        % Pomocne optimalizacne vypocty
        a = Bs*ps;
        ∆ = ps'*a;
        Delta = 1/∆;
        beta = ps'*ys;
        Beta = 1/beta;
        % Novy vypocet B_{k+1} za pomoci novych premennych
        Bk = Bs+(Beta*ys)*ys'-(Delta*a)*a';
        % Posunutie cyklu o 1 dopredu
        gs = gk;
        Bs = Bk;
        xs = xk;
        kolko = kolko + 1;
        % Restart
        if(rem(kolko,n)==0)
            Bs=eye(n);
        end
    end
end
```

Špeciálne by sme chceli upriamiť pozornosť na časť o optimalizácii výpočtovej náročnosti. V dnešnej dobe aj bežné počítače majú dostatočnú kapacitu na spracovanie primeraného množstva dát a príkazov. V našej práci sa však zaoberáme optimalizačnými úlohami netriviálnych rozmerov, čo už nemožno považovať za bežné. Preto sa naskytá nutnosť optimalizovať niektoré výpočty. V našom prípade takáto optimalizácia ušetrila na kvadratickej úlohe pre $n = 1000$ až 60% času, keď sme výpočet korekčnej matice ΔB_k prepísali tak, aby sa výpočty prevádzali v rádoch nanajvýš $O(n^2)$.

3.5 Algoritmus II.: Upravené kolineárne škálovanie navrhnuté Ariyawansom

Schématický popis navrhovaného algoritmu má tvar:

Vstup: štartovací bod $x_0 \in \mathbb{Z}^n$, funkčný hodnota $f_0 = f(x_0)$, gradient $g_0 = \nabla f(x_0)$, symetrická, kladne definitná matica $B_0 = \mathbb{I}$.

Cyklus for $k = 0, 1, 2, \ldots$

1. IF $[g_k = 0]$ then STOP $[\hat{x} = x_k]$.
2. Riešime sústavu lineárnych rovníc $B_k s_k = -g_k$. Výsledkom je spádový smer $s_k \in \mathbb{R}^n$.
3. V smere s_k vypočítame dĺžku optimálneho kroku $\lambda_k = \operatorname{argmin} \{\varphi(\lambda) \equiv f(x_k + \lambda s_k) \,|\lambda \in \mathbb{R}\}$.
4. $p_k = \lambda_k s_k$.
5. Vypočítame nový bod $x_{k+1} = x_k + p_k$.
6. Vypočítame funkčnú hodnotu $f_{k+1} = f(x_{k+1})$ a gradient g_{k+1}.
7. $\rho_k = \sqrt{(f_k - f_{k+1})^2 - (g_{k+1}^T s_k)(g_k^T s_k)}$.
8. $\gamma_k = \frac{-1}{f_k - f_{k+1} + \rho_k} g_k^T s_k$.
9. $\tilde{v}_k = \gamma_k p_k$.
10. $r_k = g_{k+1} - \frac{1}{\gamma_k^2} g_k$.
11. $w_k = \frac{r_k}{r_k^\top \tilde{v}_k} - \frac{B_k \tilde{v}_k}{\tilde{v}_k^\top B_k \tilde{v}_k}$.
12. IF $w_k = 0$ THEN
$$\theta_k = 0.$$
ELSE
$$\theta_k > \frac{1}{1 - \frac{\left(r_k^\top B_k^{-1} r_k\right)\left(\tilde{v}_k^\top B_k \tilde{v}_k\right)}{\left(\tilde{v}_k^\top r_k\right)^2}}.$$

13. Vypočítame novú maticu

$$B_{k+1} = B_k - \frac{B_k \tilde{v}_k \tilde{v}_k^\top B_k}{\tilde{v}_k^\top B_k \tilde{v}_k} + \frac{r_k r_k^\top}{\tilde{v}_k^\top r_k} + \theta_k \left(\tilde{v}_k^\top B_k \tilde{v}_k\right) w_k w_k^\top .$$

Koniec cyklu.

V tejto časti si ukážeme implementáciu do jazyka MATLAB s optimalizáciou výpočtového času.

```
% Poznamka: hodnoty s koncovkou "s" su tie, ktore pocitame v kroku k. ...
   Hodnoty s koncovkou "k" su tie, ktore vypocitame ako k+1.
function [x,kolko] = Ariyawansa(D,G,h,C,alpha,gamma,xo,xopt,Lmax,Eps)
    %%% VSTUP: %%%
    % D,G,h,C,alpha,gamma \\ matice a vektory definujuce funkciu
    % x0 \\ startovaci bod
    % xHat \\ optimalny bod
    % Lmax \\ maximalny pocet iteracii
    % Eps \\ presnost
    %%% VYSTUP: %%%
    % x \\ vysledny bod
    % kolko \\ vysledny pocet iteracii

    % Pocitadlo iteracii
    kolko = 0;
    % Rozmer funkcie
    n = length(xo);
    % Nastavenie startovacich hodnot
    Bo = eye(n);
    Bs = Bo;
    xs = xo;
    gs = dF(D,G,h,C,alpha,gamma,xs);
    Fs = F(D,G,h,C,alpha,gamma,xs);
    % Zaciatok cyklu
    while(kolko < Lmax)
        % Riesime sustavu aby sme ziskali spadovy smer ss
```

```
ss = -linsolve(Bs,gs);
% Hladame dlzku optimalneho kroku metodou konickej interpolacie
lam = Lambda(D,G,h,C,alpha,gamma,xs,ss,'ConicInter',1e-3);
% Vypocet noveho body, funkcnej hodnoty a gradientu v danom bode
ps = lam*ss;
xk = xs + ps;
gk = dF(D,G,h,C,alpha,gamma,xk);
Fk = F(D,G,h,C,alpha,gamma,xk);
% Test konvergencie
if(norm(gk) < Eps)
    x = xk;
    return
end
% Dielcie vypocty v metodach kolinearneho skalovania
rho = sqrt((Fs-Fk)^2-(ps'*gs)*(ps'*gk));
gam = (-ps'*gs)/(Fs-Fk+rho);
v = gam*ps;
ys = gam*gk-(1/gam)*gs;
r = ys/gam;
w = r/(r'*v)-(Bs*v)/(v'*Bs*v);
% OPTIMALIZACIA vypoctovej narocnosti konstrukcie novej ...
   matice B_{k+1}
% Povodny tvar:
% Bk = Bs - ((Bs'*v)*(v'*Bs))/ ...
    ((v'*Bs)*v)+(r*r')/(v'*r)+psi*(((v'*Bs)*v)*w)*w';
% Pomocne vypocty
a = v'*r;
aa = 1/a;
b = Bs*v;
bb = v'*b;
if(w == 0)
    psi = 0;
else
    psi = 1/(1-((r'*(Bs\r))*(bb))*(aa)^2)+Eps;
end
bbb = 1/bb;
```

```
        % Novy vypocet B_{k+1} za pomoci novych premennych
        Bk = Bs +(aa*r)*r'-(bbb*b)*b'+psi*(bb*w)*w';
        % Posunutie cyklu o 1 dopredu
        gs = gk;
        Bs = Bk;
        xs = xk;
        Fs = Fk;
        kolko = kolko + 1;
        % Restart
        if(rem(kolko,n)==0)
            Bs=eye(n);
        end
    end
end
```

3.6 Tabuľky výsledkov riešení úloh

3.6.1 Ilustratívne príklady

Ako prvé si uvedieme niekoľko ilustratívnych príkladov z priebehu našeho testovania. Prezentujeme po dvoch tabuľkách na každý typ zvolených funkcií. Jedna tabuľka sa bude týkať priamej kvázinewtonovskej metódy BFGS a druhá metódy kolineárneho škálovania. Tabuľky pre úsporu miesta nezobrazujú každý jeden krok interačného procesu, ale len kroky, v ktorých sa vzdialenosť od bodu optima $\hat{x} \in \mathbb{R}^n$ skrátila 10-násobne. Pre ilustráciu budeme voliť veľké rozmery úloh.

Ako prvú uvádzame kvadratickú funkciu. Rozmer úlohy je $n = 2000$ a štartovací bod je vzdialený od optimálneho riešenia $\rho = 2000$. Pri testovaní sme ako podmienku konvergencie použili normu gradientu bodu $x_k \in \mathbb{R}^n$ s presnosťou $\epsilon = 10^{-3}$. Výpočet dĺžky optimálneho kroku sme robili rekurzívnou interpolačnou metódou, pri ktorej bola opäť použitá podmienka na hodnotu derivácie v bode $\lambda_k \in \mathbb{R}$ s presnosťou na $\epsilon = 10^{-3}$. Výsledné tabuľky majú tvar:

Kvázninewtonovská metóda - BFGS				
k	$\|\|x_k - \hat{x}\|\|$	$\|\|\nabla f(x_k)\|\|$	$f(\hat{x}) - f(x_k)$	λ_k
0	2.000e+03	1.989e+08	-7.783e+09	
280	1.998e+02	2.539e+02	-1.420e+03	3.020e-04
1998	1.426e+01	8.685e-02	-1.731e-01	5.043e-02
1999	1.270e-03	1.011e-05	-2.154e-09	4.590e-01
čas výpočtu: 652.7335s				

Tabuľka 1: Ilustračný príklad 1: Kvadratická funkcia - BFGS.

a

Metóda kolineárneho škálovania				
k	$\|\|x_k - \hat{x}\|\|$	$\|\|\nabla f(x_k)\|\|$	$f(\hat{x}) - f(x_k)$	λ_k
0	2.000e+03	1.989e+08	-7.783e+09	
303	1.994e+02	9.330e+04	-5.643e+05	3.269e-04
1161	1.991e+01	3.435e+01	-6.359e+00	8.869e-04
1997	1.841e+00	3.531e-01	-5.196e-03	1.327e-01
2003	5.634e-03	2.281e-02	-1.134e-05	1.037e-01
2004	1.743e-04	2.302e-04	-1.641e-08	1.283e-01
čas výpočtu: 1577.7479s				

Tabuľka 2: Ilustračný príklad 1: Kvadratická funkcia - kolineárne škálovanie.

3.6 Tabuľky výsledkov riešení úloh

MATLAB kód, ktorý vygeneruje kvadratickú úlohu daných rozmerov a následne na ňu aplikuje metódy BFGS a kolineárneho škálovania je:

```
% generovanie kvadratickej ulohy prislusnych rozmerov:
%       n = 2000
%     rho = 2000
[D,G,h,C,alpha,gamma,xHat,x0] = Generate(2000,2000,'Quadratic');
% spustenie casovaca pre BFGS
tic;
% aplikovanie metody na vygenerovanu ulohu, vystup:
%       xBFGS = riesenie ulohy, t.j. vypocitany optimalny bod
%   kolkoBFGS = pocet iteracii potrebnych na vypocitanie bodu optima
[xBFGS,kolkoBFGS] = ...
   BFGS(D,G,h,C,alpha,gamma,x0,xHat,5e3,1e-3,'CubicInter');
% zaznamenanie dlzky casu vypoctu
casBFGS = toc;

% spustenie casovaca pre kolinearne skalovanie
tic;
% aplikovanie metody na vygenerovanu ulohu, vystup:
%         xARI = riesenie ulohy, t.j. vypocitany optimalny bod
%     kolkoARI = pocet iteracii potrebnych na vypocitanie bodu optima
[xARI,kolkoARI] = ...
   Ariyawansa(D,G,h,C,alpha,gamma,x0,xHat,5e3,1e-3,'CubicInter');
% zaznamenanie dlzky casu vypoctu
casARI = toc;
```

Ako druhú uvádzame bikvadratickú funkciu. Rozmer úlohy je $n = 1000$ a štartovací bod je vzdialený od optimálneho riešenia $\rho = 1000$. Pri testovaní sme ako podmienku konvergencie použili normu gradientu bodu $x_k \in \mathbb{R}^n$ s presnosťou 10^{-3}. Výpočet dĺžky optimálneho kroku sme robili rekurzívnou interpolačnou metódou, pri ktorej bola opäť použitá podmienka na deriváciu v bode $\lambda_k \in \mathbb{R}$ s presnosťou na 10^{-3}. Výsledné tabuľky majú tvar:

Kvázninewtonovská metóda - BFGS				
k	$\|\|x_k - \hat{x}\|\|$	$\|\|\nabla f(x_k)\|\|$	$f(\hat{x}) - f(x_k)$	λ_k
0	1.000e+03	1.267e+08	-3.465e+10	
109	9.593e+01	2.598e+05	-4.283e+06	4.564e-05
611	9.983e+00	2.694e+03	-1.983e+03	1.557e-04
1015	9.953e-01	1.500e+02	-8.874e+00	2.030e-01
1026	6.897e-02	1.141e+01	-3.609e-02	3.758e-01
1033	8.370e-03	7.510e-01	-5.757e-04	2.529e-01
1040	6.810e-04	1.465e-01	-4.747e-06	4.180e-01
1046	5.282e-05	2.608e-02	-9.453e-08	5.844e-01
1053	1.466e-05	7.194e-04	-1.397e-09	1.500e-01
čas výpočtu: 93.3817s				

Tabuľka 3: Ilustračný príklad 2: Bikvadratická funkcia - BFGS.

a

Metóda kolineárneho škálovania				
k	$\|\|x_k - \hat{x}\|\|$	$\|\|\nabla f(x_k)\|\|$	$f(\hat{x}) - f(x_k)$	λ_k
0	1.000e+03	1.267e+08	-3.465e+10	
79	9.896e+01	2.109e+05	-3.465e+06	1.458e-05
650	9.959e+00	1.433e+03	-1.361e+03	5.423e-05
986	9.485e-01	7.355e+01	-7.190e+00	2.065e-03
1016	8.846e-02	8.336e+00	-7.436e-02	2.000e-01
1024	6.097e-03	7.127e-01	-2.417e-04	1.793e-01
1028	8.903e-04	7.527e-02	-6.039e-06	1.804e-01
1033	8.306e-05	4.431e-03	-6.100e-08	1.991e-01
čas výpočtu: 176.4848s				

Tabuľka 4: Ilustračný príklad 2: Bikvadratická funkcia - kolineárne škálovanie.

MATLAB kód, ktorý vygeneruje bikvadratickú úlohu daných rozmerov a následne na ňu aplikuje metódy BFGS a kolineárneho škálovania je:

```
% generovanie kvadratickej ulohy prislusnych rozmerov:
%       n = 1000
%     rho = 1000
[D,G,h,C,alpha,gamma,xHat,x0] = Generate(2000,2000,'Biquadratic');
% spustenie casovaca pre BFGS
tic;
% aplikovanie metody na vygenerovanu ulohu, vystup:
%       xBFGS = riesenie ulohy, t.j. vypocitany optimalny bod
%   kolkoBFGS = pocet iteracii potrebnych na vypocitanie bodu optima
[xBFGS,kolkoBFGS] = ...
   BFGS(D,G,h,C,alpha,gamma,x0,xHat,5e3,1e-3,'CubicInter');
% zaznamenanie dlzky casu vypoctu
casBFGS = toc;

% spustenie casovaca pre kolinearne skalovanie
tic;
% aplikovanie metody na vygenerovanu ulohu, vystup:
%         xARI = riesenie ulohy, t.j. vypocitany optimalny bod
%     kolkoARI = pocet iteracii potrebnych na vypocitanie bodu optima
[xARI,kolkoARI] = ...
   Ariyawansa(D,G,h,C,alpha,gamma,x0,xHat,5e3,1e-3,'ConicInter');
% zaznamenanie dlzky casu vypoctu
casARI = toc;
```

Ako poslednú uvádzame polynomiálnu funkciu. Rozmer úlohy je $n = 200$ a štartovací bod je vzdialený od optimálneho riešenia $\rho = 100$. Pri testovaní sme ako podmienku konvergencie použili normu gradientu bodu $x_k \in \mathbb{R}^n$ s presnosťou 10^{-3}. Výpočet dĺžky optimálneho kroku sme robili rekurzívnou interpolačnou metódou, pri ktorej bola opäť použitá podmienka na deriváciu v bode $\lambda_k \in \mathbb{R}$ s presnosťou na 10^{-3}. Výsledné tabuľky majú tvar:

Kvázninewtonovská metóda - BFGS				
k	$\|\|x_k - \hat{x}\|\|$	$\|\|\nabla f(x_k)\|\|$	$f(\hat{x}) - f(x_k)$	λ_k
0	1.000e+02	1.067e+06	-3.888e+07	
129	9.560e+00	7.342e+02	-4.104e+02	2.156e-04
200	9.207e-01	4.778e-01	-4.856e-02	1.723e-02
202	8.337e-05	5.841e-04	-9.313e-10	2.440e-03
čas výpočtu: 0.4053s				

Tabuľka 5: Ilustračný príklad 3: Polynomiálna funkcia - BFGS.

a

Metóda kolineárneho škálovania				
k	$\|\|x_k - \hat{x}\|\|$	$\|\|\nabla f(x_k)\|\|$	$f(\hat{x}) - f(x_k)$	λ_k
0	1.000e+02	1.067e+06	-3.888e+07	
129	9.942e+00	4.385e+02	-1.230e+01	2.254e-04
164	8.243e-01	3.759e-01	-4.856e-02	3.738e-05
182	2.844e-05	5.985e-04	0.000e+00	1.321e-2
čas výpočtu: 1.2143s				

Tabuľka 6: Ilustračný príklad 3: Polynomiálna funkcia - kolineárne škálovanie.

MATLAB kód, ktorý vygeneruje polynomiálnu úlohu daných rozmerov a následne na ňu aplikuje metódy BFGS a kolineárneho škálovania je:

```
% generovanie kvadratickej ulohy prislusnych rozmerov:
%       n = 200
%     rho = 100
[D,G,h,C,alpha,gamma,xHat,x0] = Generate(2000,2000,'Polynomial');
% spustenie casovaca pre BFGS
tic;
% aplikovanie metody na vygenerovanu ulohu, vystup:
%       xBFGS = riesenie ulohy, t.j. vypocitany optimalny bod
%   kolkoBFGS = pocet iteracii potrebnych na vypocitanie bodu optima
[xBFGS,kolkoBFGS] = ...
   BFGS(D,G,h,C,alpha,gamma,x0,xHat,5e3,1e-3,'CubicInter');
% zaznamenanie dlzky casu vypoctu
casBFGS = toc;

% spustenie casovaca pre kolinearne skalovanie
tic;
% aplikovanie metody na vygenerovanu ulohu, vystup:
%         xARI = riesenie ulohy, t.j. vypocitany optimalny bod
%     kolkoARI = pocet iteracii potrebnych na vypocitanie bodu optima
[xARI,kolkoARI] = ...
   Ariyawansa(D,G,h,C,alpha,gamma,x0,xHat,5e3,1e-3,'ConicInter');
% zaznamenanie dlzky casu vypoctu
casARI = toc;
```

3.6.2 Porovnanie dvoch metód výpočtu optimálneho kroku

Poznámka 3. V tejto časti všetky testy prebiehali rovnakým spôsobom. Testovanie prebiehalo vždy na sérii 20 homogénnych úloh. Konvergentné kritérium bolo nastavené na normu gradientu v bode $x_k \in \mathbb{R}^n$ s presnosťou $\epsilon = 10^{-3}$.

Pri výpočte veľkosti optimálneho kroku $\lambda_k \in \mathbb{R}$ sme použili dve metódy. Metódu zlatého rezu, ktorého presnosť bola určená šírkou intervalu o veľkosti $\epsilon = 10^{-3}$. Rovnako tak pri rekurzívnych interpolačných metódach, kubickej a kónickej, sme ako konvergentné kritérium brali hodnotu derivácie v bode $\lambda_k \in \mathbb{R}$ s presnosťou $\epsilon = 10^{-3}$.

Použité symboly v tabuľkách sú nasledovné:

- n = počet premenných,
- ρ = vzdialenosť počiatočného bodu,
- t = čas v sekundách,
- p.it. = počet iterácií.

Ako prvé uvedieme porovnanie metód na výpočet optimálneho kroku, t.j. metódy zlatého rezu verzus kubická, resp. kónická interpolácia.

Kvadratická funkcia	BFGS		KŠ	
$n = 100$, $\rho = 1000$	t	$p.it.$	t	$p.it.$
Zlatý rez	0.1924	106	0.3126	103
Interpolácia	0.1036	99	0.2276	99
$n = 250$, $\rho = 1000$	t	$p.it.$	t	$p.it.$
Zlatý rez	0.8782	262	1.8464	255
Interpolácia	0.6738	250	1.6059	248
$n = 500$, $\rho = 1000$	t	$p.it.$	t	$p.it.$
Zlatý rez	7.0110	519	15.7076	501
Interpolácia	5.9270	500	14.5720	498
$n = 1000$, $\rho = 1000$	t	$p.it.$	t	$p.it.$
Zlatý rez	73.2714	1006	209.7459	1006
Interpolácia	62.4417	999	213.3414	1000

Tabuľka 7: Porovnanie metód výpočtu optimálneho kroku pri kvadratickej funkcii.

Môžeme vypozorovať, že pre kvadratickú funkciu interpolačné metódy až na jeden prípad predbehli metódu zlatého rezu, dokonca v niektorých prípadoch s citeľným

predstihom. Priblížme si ako to vyzeralo pri bikvadratickej funkcii.

Bikvadratická funkcia	BFGS		KŠ	
$n = 100$, $\rho = 1000$	t	$p.it.$	t	$p.it.$
Zlatý rez	0.4713	168	0.4209	131
Interpolácia	0.4917	184	0.5265	145
$n = 250$, $\rho = 1000$	t	$p.it.$	t	$p.it.$
Zlatý rez	1.4267	283	1.9761	277
Interpolácia	1.3821	308	2.1901	298
$n = 500$, $\rho = 1000$	t	$p.it.$	t	$p.it.$
Zlatý rez	8.2780	585	17.9460	523
Interpolácia	9.1056	556	18.2099	601
$n = 1000$, $\rho = 1000$	t	$p.it.$	t	$p.it.$
Zlatý rez	93.9470	1082	154.4279	934
Interpolácia	102.4579	1050	183.4978	1007

Tabuľka 8: Porovnanie metód výpočtu optimálneho kroku pri bikvadratickej funkcii.

Pri bikvadratickej funkcii už interpolačné metódy mierne zaostávali, hlavne čo sa týka kónickej interpolácie. Polynomiálnu funkciu sme skúmali pri stupni polynómu 6. Výsledky dopadli nasledovne:

Polynomiálna funkcia	BFGS		KŠ	
$n = 100$, $\rho = 100$	t	$p.it.$	t	$p.it.$
Zlatý rez	0.3324	111	0.8930	113
Interpolácia	0.1157	101	0.7953	100
$n = 150$, $\rho = 100$	t	$p.it.$	t	$p.it.$
Zlatý rez	0.5200	150	1.2761	155
Interpolácia	0.1822	149	1.1214	150
$n = 200$, $\rho = 100$	t	$p.it.$	t	$p.it.$
Zlatý rez	0.5956	225	1.4087	227
Interpolácia	0.3976	199	1.1949	201

Tabuľka 9: Porovnanie metódu výpočtu optimálneho kroku pri polynomiálnej funkcii.

Správanie sa interpolačných metód pri polynomiálnej funkcii stupňa 6 zdanlivo napodobňuje správanie sa pri kvadratickej funkcii. Opäť raz interpolačné metódy predčili metódu zlatého rezu ako v počte iterácií potrebných na výpočet optimálneho riešenia $\hat{x}$, tak aj v čase potrebnom k dosiahnutiu tohoto cieľu.

Odpozorovali sme, že interpolačné metódy skutočne priniesli to, čo sme od nich čakali, a to zefektívnenie výpočtu veľkosti optimálneho kroku oproti klasickej metóde

zlatého rezu. Preto sa v ďalšej časti sústredíme iba na výsledky, ktoré sme nadobudli pri aplikovaní interpolačných metód na výpočet dĺžky optimálneho kroku.

3.6.3 Porovnanie BFGS metódy a metódy kolineárneho škálovania

V poslednej časti našej práce sa budeme venovať porovnaniu efektívnosti zvolených algoritmov. Testovanie bude vždy prebiehať na sérii 20 homogénnych úloh s netriviálnymi rozmermy $100 \leq n \leq 1000$. Pozornosť zameriame na nasledujúce porovnávané veličiny:

- Čas, za ktorý algoritmus vygenerovanú úlohu vyriešil.
- Počet iterácii, ktoré potreboval na vyriešenie problému.
- Dosiahnutá presnosť pri pevne zvolenom počte iterácií, ktorý bude pre kvadratickú funkciu n, pre bikvadratickú $2n$ a pre polynomiálnu $3n$.

Prvé dve porovnávacie kritéria sme vlastne už realizovali v časti 3.6.2 (dokonca pri dvoch rôznych met'dach výpočtu optimálneho kroku). Vidíme, že metóda kolineárneho škálovania bola o niečo lepšia ako kvázinewtonovská BFGS metóda v počte iterácii. Avšak je skoro dvakrát pomalšia.

Prejdime k porovnávaniu presnosti pri pevne zvolenom počte iterácií "max". Formálne dosiahnutú presnosť zapíšeme

$$\delta = ||x_{max} - \hat{x}|\,|.$$

Pri testovaní kvadratickej a bikvadratickej funkcie budeme testovať série 20 úloh rozmerov $n = 100, 500, 1000$. Pre polynomiálnu použijeme rozmery úloh $n = 100, 150, 200$. Ako prvú uvedieme kvadratickú funkciu:

metóda	$n = 100$ max=100	$n = 500$ max=500	$n = 1000$ max=1000
BFGS	1.008^{-9}	2.139^{-8}	3.885^{-7}
KŠ	5.384^{-6}	2.650^{-4}	1.807^{-4}

Tabuľka 10: Porovnanie výslednej presnosti pri kvadratickej funkcii.

Z výsledkov vyplýva, že pri kvadratickej funkcii je BFGS metóda lepšia ako metóda kolineárneho škálovania, čo je v súlade s teóriu, keďže BFGS metóda priamo vychádza z kvadratického modelu.

Nasledujúca tabuľka sa týka testovania na bikvadratickej funkcii:

metóda	$n = 100$ max=200	$n = 500$ max=1000	$n = 1000$ max=2000
BFGS	9.667^{-10}	7.259^{-6}	8.323^{-7}
KŠ	8.639^{-8}	8.710^{-6}	6.946^{-5}

Tabuľka 11: Porovnanie výslednej presnosti pri bikvadratickej funkcii.

Z výsledkov opäť vyplýva, že BFGS metóda je lepšia ako metóda kolineárneho škálovania. Avšak rozdiel už nie je až taký markantný.

Pre polynomiálnu funkciu dopadli testy nasledovne:

metóda	$n = 100$ max=300	$n = 150$ max=450	$n = 200$ max=600
BFGS	1.008^{-7}	2.139^{-6}	3.885^{-7}
KŠ	5.384^{-7}	2.650^{-6}	1.807^{-4}

Tabuľka 12: Porovnanie výslednej presnosti pri polynomiálnej funkcii.

Z pozorovania vyplýva, že pri polynomiálnej funkcii sa úlohy nižších rozmerov riešia porovnateľne dobre pri oboch metódach, avšak pre väčšie rozmery sa rozdiel opäť prehlbuje v neprospech metódy kolineárneho škálovania.

ZÁVER

Záver

Cieľom bakalárskej práce bolo naštudovať si najnovšie metódy kolineárneho škálovania, ktoré pôsobia ako efektívna nadstavba pre kvázinewtonovské metódy na poli nelineárneho programovania. Následne niektorú z nich naprogramovať v programovacom jazyku MATLAB a porovnať jej efektívnosť s kvázinewtonovskou BFGS metódou.

Teoretické zázemie pre štúdium kolineárneho škálovania nám poskytla literatúra od Davidona [4] a Sun, Yuan-a [8], ktorú sme pre porozumenie širších súvislostí rozšírili o vlastné dôkazy a odvodenia niektorých vzťahov.

Pre programátorskú časť boli veľmi obohacujúce články od Ariyawansu [1, 2], Bjorstad, Nocedala [3] a kniha od Hamalu [5], z ktorých sme čerpali väčšinu myšlienok použitých v práci. Už len samotné programovanie metód bolo pre autora veľmi podnetné a v určitom zmysle ho posunulo v zmýšľaní ďalej. Bolo nutné spraviť veľké množstvo experimentov na rôznych typoch konvexných úloh netriviálnych rozmerov ($100 \leq n \leq 1000$), ktoré bežali na počítači neprestajne niekoľko hodín až dní, pričom programovacieho kódu je len cca 320 riadkov. Tu sa ukázala sila „efektívneho" programovania, kedy sme boli schopní optimalizáciou programového kódu docieliť veľké skrátenie výpočtového času.

Práca by mala byť prínosom pre každého, koho daná problematika zaujíma. Nachádza sa v nej teoretické prepojenie klasických kvázinewtonovských metód s metódami kolineárneho škálovania. Čitateľ aj bez základných vedomostí z danej oblasti nelineárneho programovania by nemal mať problém problematike porozumieť a pochopiť základné princípy riešenia úloh za pomoci iteračných procesov. V texte sa vyskytujú časti programu, ktoré ako názorná ukážka dopomáhajú porozumieť praktickej aplikácii naštudovanej teórie.

Na záver uvádzame najskôr porovnanie dvoch metód, pomocou ktorých počítame dĺžku optimálneho kroku a následne porovnanie efektívnosti jednotlivých metód, ktoré riešia úlohu na voľný extrém podľa troch parametrov (výpočtový čas, počet iterácii pre dosiahnutie určitej presnoti a celková presnosť pri dopredu stanovenom maximálnom počte iterácií), pri riešení sérii homogénnych náhodne, generovaných konvexných úloh.

ZÁVER

Z výsledkov vyplynulo, že z porovnania metód na výpočet dĺžky optimálneho kroku vyšli ako lepšie interpolačné metódy pred metódou zlatého rezu. Pri porovnaní optimalizačných metód obstála lepšie kvázinewtonovská BFGS.

LITERATÚRA

Literatúra

[1] Ariyawansa, K. A.: *Deriving Collinear Scaling Algorithms as Extension of Quasi-Newton Methods and the Local Convergence of DFP and BFGS- related collinear Scaling Algorithms*, Math. Prog. 49 (1990), 23-48

[2] Ariyawansa, K. A.: *Global Convergence of a Class of Collinear Scaling Algorithms with Inexact Line Searches on Convex Functions*, Computing 63 (1999), 145-169

[3] Bjorstad, P., Nocedal, J.: *Analysis of a New Algorithm for One-Dimensional Minimization*, Computing 22 (1979), 93-100

[4] Davidon, W. C.: *Conic Approximation and Collinear Scalings for Optimizers*, SIAM Journal on Numerical Analysis 17 (1980), 268-281

[5] Hamala, M., Trnovská, M.: *Nelineárne programovanie, teória a algoritmy*, EPOS, Bratislava, 2013

[6] Hamala, M.: *osobná komunikácia*, FMFI UK, Bratislava, 2013

[7] Strang, G.: *Linerar algebra and its applications*, Thompson Learning, 1988

[8] Sun, W., Yuan, Y. X.: *Optimization Theory and Methods - Nonlinear Programming*, Springer, 2006