UNIVERZITA KOMENSKÉHO V BRATISLAVE
FAKULTA MATEMATIKY, FYZIKY A INFORMATIKY

NOVÁ METÓDA KONJUGOVANÝCH GRADIENTOV

BAKALÁRSKA PRÁCA

2012 Marián PITONIAK

UNIVERZITA KOMENSKÉHO V BRATISLAVE
FAKULTA MATEMATIKY, FYZIKY A INFORMATIKY

NOVÁ METÓDA KONJUGOVANÝCH GRADIENTOV

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 2012 **Marián PITONIAK**

Univerzita Komenského v Bratislave
Fakulta matematiky, fyziky a informatiky

ZADANIE ZÁVEREČNEJ PRÁCE

Meno a priezvisko študenta: Marián Pitoniak
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: Nová metóda konjugovaných gradientov

Cieľ: Experimentálne porovnanie efektívnosti rôznych formúl MKG.

Vedúci: doc. RNDr. Milan Hamala, CSc.

Dátum zadania: 16.10.2011

Dátum schválenia: 27.10.2011

doc. RNDr. Margaréta Halická, CSc.
garant študijného programu

.....
študent

.....
vedúci

Poďakovanie

Chcem sa poďakovať svojmu vedúcemu bakalárskej práce doc. RNDr. Milanovi Hamalovi, CSc. za cenné rady a kritické pripomienky, ktoré mi pomohli pri vypracovaní tejto bakalárskej práce.

Abstrakt v štátnom jazyku

PITONIAK, Marián: Nová metóda konjugovaných gradientov [Bakalárska práca], Univerzita Komenského v Bratislave, Fakulta matematiky, fyziky a informatiky, Katedra aplikovanej matematiky a štatistiky; vedúci: doc. RNDr. Milan Hamala, CSc., Bratislava, 2011, 94 strán

V práci sa venujeme iteračnej metóde nazvanej metóda konjugovaných gradientov. Používa sa napríklad pri hľadaní voľného minima konvexnej funkcie. V roku 2005 autori W. Hager a H. Zhang prišli s novou modifikáciou metódy konjugovaných gradientov. Cieľom tejto práce je experimentálne porovnanie efektívnosti ich návrhu s modifikáciami navrhovanými v predchádzajúcich desaťročiach. Veľká časť práce je venovaná programovej realizácii metódy konjugovaných gradientov a metód s ňou súvisiacich. V práci sú ďalej popísané riadené experimenty, v ktorých používame rôzne predpisy konvexných účelových funkcií. Cieľom experimentov je overiť výsledky autorov W. Hagera a H. Zhanga týkajúce sa zistenej vyššej efektívnosti ich modifikácie metódy konjugovaných gradientov. Naše výsledky v závere práce potvrdzujú tieto zistenia. V prípadoch nejednoznačného výsledku problém hlbšie analyzujeme a snažíme sa príčinu nejednoznačnosti objaniť.

Kľúčové slová: iteračná metóda, nová metóda konjugovaných gradientov, minimalizácia konvexnej funkcie, programová realizácia MKG, experimentálne porovnanie efektívnosti formúl parametra MKG

Abstract

PITONIAK, Marián: A new conjugate gradient method [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, 2011, 94 pages

This work is focused on an iterative method called a conjugate gradient method. This method is used for example for finding an unconstrained minimum of convex function. In 2005 authors W. Hager and H. Zhang proposed a new modification of the conjugate gradient method. The aim of this work consists in an experimental comparison of their proposed modification to the modifications created in previous decades. A huge part of this work is dedicated to program implementation of the conjugate gradient method and related methods. Later in the work controlled experiments are introduced, in which we use different convex objective functions. The aim of the experiments is to verify results given by the authors W. Hager and H. Zhang, in which their modification of the conjugate gradient method seems to reach higher efficiency. Our results at the end of this work confirm their finding. In cases of uncertain results we proceed in further analysis of the problem and try to explain the reason of the uncertainty.

Keywords: an iterative method, a new conjugate gradient method, minimization of a convex function, program implementation of the CGM, an experimental comparison of various formulas of a CGM parameter

Obsah

Zoznam obrázkov

Zoznam tabuliek

Zoznam použitých symbolov

$\|.\|$	Euklidovská norma vektora
g_k	vektor gradientu funkcie f v bode x_k (platí $g_k = \nabla f(x_k)$)
MKG	skratka pre metódu konjugovaných gradientov
β	parameter MKG
β^{HS}	parameter MKG autorov Hestenesa-Stiefela
β^{FR}	parameter MKG autorov Fletchera-Reevesa
β^{PRP}	parameter MKG autorov Polaka-Riebièreho-Polyaka
β^{DY}	parameter MKG autorov Daia-Yuana
β^{HZ}	parameter MKG autorov Hagera-Zhanga
λ_{min}	najmenšia vlastná hodnota matice
λ_{max}	najväčšia vlastná hodnota matice
$\kappa(M)$	číslo podmienenosti matice M
ρ	vzdialenosť dvoch bodov x, y ($\|x - y\| = \rho$)
$\|I\|$	dĺžka intervalu I
p_k	vektor rozdielu bodov x_{k+1} a x_k

Úvod

Metóda konjugovaných gradientov je iteračná metóda používaná v optimalizačných úlohách týkajúcich sa hľadania voľného extrému účelovej funkcie. Metóda sa riadi tzv. „bod-smer-krok"pravidlom. Toto pravidlo znamená, že štartujúc zo zadaného bodu, metóda najprv nájde smer optimalizácie a následne určí, o aký krok sa v danom smere posunie. Pri tvorbe nového smeru optimalizácie metóda konjugovaných gradientov využíva smer najstrmšieho spádu a predchádzajúci použitý smer optimalizácie.

Metóda konjugovaných gradientov sa používa najmä pri riešení úloh na hľadanie voľného extrému, ktoré dosahujú robustné rozmery. Dôvodom je jej relatívne nízka pamäťová náročnosť, ale aj nízka výpočtová náročnosť jej jednotlivých krokov. Uvedené výhody vyplývajú z faktu, že pri výpočtoch používa metóda konjugovaných gradientov gradientnú informáciu. Na rozdiel od newtonovej alebo kvázinewtonovej metódy tak nepotrebuje poznať Hessovu maticu druhých parciálnych derivácií účelovej funkcie. Práve pre tieto dôvody je táto metóda aj naďalej skúmaná a zdokonaľovaná. Dôkazom sú pomerne nedávno vydané články, ale aj knihy venované tejto metóde. Z knižných titulov by sme radi vyzdvihli publikácie [15] a [10], v ktorých je metóda konjugovaných gradientov podrobne a zrozumiteľne vysvetlená.

Počiatky metódy konjugovaných gradientov siahajú do päťdesiatych a šesťdesiatych rokov minulého storočia. Pôvodnou motiváciou bolo nájsť čo najlepší algoritmus na riešenie systému lineárnych rovníc.

V roku 1952 autori M.R. Hestenes a E. Stiefel publikovali článok [9], v ktorom predstavili novú iteračnú metódu riešenia systému n lineárnych rovníc so symetrickou kladne definitnou maticou. Nazvali ju metóda konjugovaných gradientov.

Ďalším pomyselným míľnikom bol rok 1964, kedy autori R. Fletcher a C. Reeves predstavili v článku [5] upravený algoritmus metódy konjugovaných gradientov, ktorý bol zameraný na minimalizáciu funkcie n premenných. Práve táto metóda výrazne ovplyvnila oblasť nelineárneho programovania.

Ďalších štyridsať rokov výskumu metódy konjugovaných gradientov prinieslo nové úpravy od mnohých známych autorov, ktorým sa v našej práci taktiež venujeme.

V roku 2005 autori W.W. Hager a H. Zhang publikovali článok [6], v ktorom prezentovali výsledky svojho najnovšieho výskumu. V článku navrhli a analyzovali novú

metódu konjugovaných gradientov, ktorú nazvali CG metóda. Ich výskum a snaha overiť ich výsledky sa stali podnetom pre vznik tejto bakalárskej práce.

Hlavným cieľom tejto práce bolo porovnať efektivitu CG metódy s metódami navrhovanými v predchádzajúcich desaťročiach. Popri naplnení tohto cieľa sme sa snažili čitateľovi čo najlepšie priblížiť pozadie fungovania všetkých častí, z ktorých sa metóda konjugovaných gradientov skladá.

Našu prácu, pozostávajúcu z teoretickej a praktickej časti, sme rozdelili do piatich kapitol. Potrebná teória pre vysvetlenie problematiky je obsiahnutá v prvých dvoch kapitolách.

Úvodná kapitola obsahuje niekoľko základných pojmov z nelineárneho programovania, ktoré sú potrebné pre uvedenie metódy konjugovaných gradientov. Následne uvedieme definíciu tejto metódy a v zbytku kapitoly sa zameriame na jej konvergenčné vlastnosti v prípade konvexných účelových funkcií.

Druhá kapitola popisuje historický vývoj metódy konjugovaných gradientov od jej vzniku po nedávnu minulosť. Tento vývoj zachytávame prostredníctvom vývoja algebraického tvaru parametra MKG od viacerých známych autorov, ktorí tak prispeli k rozšíreniu poznatkov o metóde konjugovaných gradientov. V závere kapitoly uvádzame novú metódu konjugovaných gradientov autorov W. Hagera a H. Zhanga, prezentovanú v článku [6], ktorá je zároveň nosnou témou tejto práce.

Tretiu kapitolu už zaraďujeme do praktickej časti našej práce. Vysvetľujeme v nej princípy metód, ktoré priamo súvisia s problematikou metódy konjugovaných gradientov alebo experimentami plánovanými pre ďalšie kapitoly. Následne popisujeme naše programové realizácie týchto metód, pričom sa venujeme aj riešeniu problémov vznikajúcich pri ich aplikácii v praxi.

Štvrtá kapitola obsahuje niekoľko počiatočných experimentov. Ich cieľom bolo optimálne doľadenie programových realizácií popísaných v predchádzajúcej kapitole.

Piata kapitola je záverečnou kapitolou našej práce. Jej obsahom sú experimenty zamerané na porovnanie efektívnosti rôznych formúl metódy konjugovaných gradientov. Výsledky týchto experimentov sú priebežne analyzované a ukončené našimi závermy. Všeobecné zhrnutie dosiahnutých výsledkov uvádzame v závere našej práce.

1 Úvod do metódy konjugovaných gradientov

Majme úlohu na voľný extrém

$$\min\{f(x) : x \in \mathbb{R}^n\}, \tag{1}$$

kde $f : \mathbb{R}^n \longmapsto \mathbb{R}$ je spojite diferencovateľná konvexná funkcia. Túto úlohu budeme riešiť metódou konjugovaných gradientov (ďalej MKG), ktorá je iteračnou metódou.

1.1 Všeobecná iteračná metóda

Iteračná metóda (metóda postupných aproximácií) je proces, ktorý zo zadaného štartovacieho bodu $x_0 \in \mathbb{R}^n$ generuje postupnosť bodov $x_1, x_2, \ldots$ podľa schémy

$$x_{k+1} = x_k + \alpha_k d_k, \tag{2}$$

kde $\alpha_k > 0$ je kladná dĺžka kroku a $d_k \in \mathbb{R}^n$ smer. Cieľom metódy je postupné približovania sa k optimálnemu riešeniu riešenej úlohy.

Krok α_k môžeme voliť konštantný, optimálny alebo ako kombináciu týchto dvoch možností. V prípade optimálneho kroku ide o riešenie úlohy

$$\alpha_k = \arg\min_{\alpha>0} f(x_k + \alpha d_k) \tag{3}$$

Podľa toho, aký typ smerového vektora d_k je v algoritme použitý, môžeme rozdeliť algoritmy iteračných metód na:

- stochastické - použitý je náhodný smerový vektor d_k
- cyklické - v algoritmoch je vopred zvolená n-tica vektorov tvoriaca bázu v $\mathbb{R}^n$
- gradientné - ako vektor d_k je použitý záporný gradient funkcie $f(x)$ v bode x_k alebo nejaká jeho transformácia

Zatiaľ iba uvedieme, že MKG zaraďujeme práve do skupiny gradientných metód.

Definícia 1.1. *Smer* $\mathbf{0} \neq d_k \in \mathbb{R}^n$ *nazývame spádový smer, ak platí:*

$$\exists \hat{\lambda} > 0, \text{ že } \forall \lambda \in (0, \hat{\lambda}): f(x_k + \lambda d_k) < f(x_k)$$

Lema 1.2. *Ak vektor* d_k *spĺňa* $g_k^T d_k < 0$*, potom vektor* d_k *predstavuje spádový smer.*

Dôkaz:
Označme $\varphi(\lambda) = f(x_k + \lambda d_k)$. Potom platí

$$\varphi'(0) = g_k^T d_k < 0 \tag{4}$$

To znamená, že existuje také $\hat{\lambda} > 0$, pre ktoré potom $f(x_k + \hat{\lambda} d_k) < f(x_k)$. Na základe tohto poznatku a Definície 1.1 je vektor d_k spádový smer. ■

Spádové smery d_k môžeme generovať pomocou $d_k = -H_k g_k$, kde H_k je kladne definitná symetrická matica, pretože podľa Lemy 1.2 je takto vytvorený smer spádový. Cauchyho spádový smer $d_k = -g_k$ je špeciálnym prípadom takto voleného smeru d_k ($H_k = I$). MKG, ktorú definujeme v nasledujúcej podkapitole, generuje spádové smery.

1.2 Metóda konjugovaných gradientov

Metóda konjugovaných gradientov je iteračná metóda, ktorá na základe štartovacieho bodu $x_0 \in \mathbb{R}^n$ generuje postupnosť bodov $x_1, x_2, \ldots$ podľa schémy (2).

Výraz α_k na pravej strane (2) predstavuje kladnú dĺžku kroku iterácie v smere spádového vektora d_k. Je riešením úlohy popísanej v (3).

Vektor d_k je vytváraný nasledovne:

$$d_{k+1} = -g_{k+1} + \beta_k d_k, \qquad k \geq 0, \qquad d_0 = -g_0 \tag{5}$$

kde g_{k+1} je gradient funkcie $f(x)$ z úlohy (1) v bode x_{k+1}.

Skalár β_k (nazývaný parameter MKG) je konštruovaný tak, aby MKG použitá na optimalizáciu kvadratickej funkcie skonvergovala do optimálneho riešenia za nanajvýš n krokov. Spolu s d_k výraz $\beta_k d_k$ predstavuje aditívnu korekciu Cauchyovského smeru $-g_{k+1}$. Parameter MKG β_k má rôzne algebraické tvary, avšak v prípade, že je funkcia $f(x)$ kvadratická a v algoritme je použitý optimálny krok α_k, teória vraví, že sú tieto

tvary ekvivalentné. Medzi tie známejšie patrí tvar Fletchera-Reevesa, Hestenesa-Stiefela alebo Polaka-Riebièreho-Polyaka (parametru β_k je venovaná kapitola 2).

Lema 1.3. *Ak* $\alpha_k = \arg\min_{\alpha>0} f(x_k + \alpha d_k)$, *potom* $g_{k+1}^T d_k = 0$.

Dôkaz:
Z predpokladu vyplýva, že ak označíme $\varphi(\lambda) = f(x_k + \lambda d_k)$, potom z nutnej podmienky pre nájdenie minima dostaneme

$$0 = \varphi'(\alpha_k) = g_{k+1}^T d_k. \quad \blacksquare$$

Tvrdenie 1.4. *Ak je krok* α_k *zvolený optimálne, potom je vektor* d_{k+1} *definovaný v (5) spádový.*

Dôkaz:
Vynásobme rovnosť (5) vektorom g_{k+1} zľava. Dostaneme

$$g_{k+1}^T d_{k+1} = -g_{k+1}^T g_{k+1} + \beta_k g_{k+1}^T d_k \tag{6}$$

Pretože je splnený predpoklad Lemy 1.3, platí $g_{k+1}^T d_k = 0$, takže môžeme (6) zapísať ako

$$g_{k+1}^T d_{k+1} = -\|g_{k+1}\|^2 < 0 \tag{7}$$

Z Lemy 1.2 vyplýva, že vektor d_{k+1} je naozaj spádový. $\blacksquare$

1.3 Konvergencia metódy konjugovaných gradientov

1.3.1 Konvergencia pre rýdzokonvexnú kvadratickú funkciu

Uvažujme úlohu (1), pričom za funkciu $f(x)$ vezmeme rýdzokonvexnú kvadratickú funkciu. Nech teda $f : \mathbb{R}^n \longmapsto \mathbb{R}$ má nasledovný tvar:

$$f(x) = \frac{1}{2} x^T Q x + b^T x \tag{8}$$

,kde Q je kladnedefinitná symetrická matica a $b \in \mathbb{R}^n$.

Definícia 1.5. *Nech Q je symetrická matica. Vektory d_i a d_j nazývame Q-konjugované (Q-ortogonálne), ak $d_i^T Q d_j = 0$ $(i \neq j)$.*

Nasledujúca veta hovorí o užitočnosti použitia Q-konjugovaných vektorov v algoritme (2) na minimalizáciu kvadratickej funkcie:

Veta 1.6. *(Veta o konjugovaných smeroch) Majme úlohu min $f(x) = \frac{1}{2}x^T Q x + b^T x$, postupnosť* n *nenulových Q-ortogonálnych vektorov $\{d_i\}_{i=0}^{n-1}$ a ľubovoľný bod $x_0 \in \mathbb{R}^n$. Potom postupnosť bodov $\{x_k\}$ generovaná podľa*

$$x_{k+1} = x_k + \alpha_k d_k, \qquad k \geq 0 \tag{9}$$

kde $\alpha_k = \arg\min f(x_k + \alpha d_k)$ má tvar

$$\alpha_k = -\frac{d_k^T g_k}{d_k^T Q d_k} \tag{10}$$

a

$$g_k = Q x_k + b \tag{11}$$

konverguje k optimálnemu riešeniu $\hat{x}$ úlohy min $f(x) = \frac{1}{2}x^T Q x + b^T x$ nanajvýš za n *krokov. To znamená $\hat{x} = x_n$.*

Dôkaz: (uvedený dôkaz je použitý z [8])

$$x_n = x_0 + \sum_{i=0}^{n-1} \alpha_i d_i = x_{k+1} + \sum_{i=k+1}^{n-1} \alpha_i d_i \tag{12}$$

$$g(x_n) = g\left(x_{k+1} + \sum_{i=k+1}^{n-1} \alpha_i d_i\right) = Q x_{k+1} + \left(\sum_{i=k+1}^{n-1} \alpha_i Q d_i\right) + b = g(x_{k+1}) + \sum_{i=k+1}^{n-1} \alpha_i Q d_i \tag{13}$$

Skalárnym súčinom (13) a vektora d_k zľava pre $k = 0, \ldots, n-2$ dostaneme

$$d_k^T g(x_n) = d_k^T g(x_{k+1}) + \sum_{i=k+1}^{n-1} \alpha_i d_k^T Q d_i, \qquad k = 0, \ldots, n-2 \tag{14}$$

Vďaka *Q-ortogonalite* vektorov d_k je celá suma v rovnosti (14) rovná 0. Výraz $d_k^T g(x_{k+1})$ je podľa Lemy 1.3 rovnako nulový, pretože krok α_i bol hľadaný optimálne.

Nech $k = n - 1$

$$d_k^T g(x_n) = d_{n-1}^T g(x_n) = 0$$

z rovnakého dôvodu ako v predchádzajúcom prípade (opätovné použitie lemy 1.3).

Vidíme teda, že vektor $g(x_n)$ je ortogonálny na n lineárne nezávislých vektorov d_k. Jediný vektor spĺňajúci toto kritérium je nulový vektor. Z $g(x_n) = \mathbf{0}$ vyplýva, že bod x_n je optimálnym riešením úlohy $\min f(x) = \frac{1}{2}x^T Q x + b^T x$ (lokálne minimum rýdzokonvexnej funkcie je zároveň aj jej globálnym minimom). ■

Ak metóda konjugovaných gradientov spĺňa predpoklady Vety 1.6, jej n-kroková konvergencia je dokázaná. Je preto potrebné ešte ukázať, že smerové vektory d_k generované predpisom (5) spĺňajú podmienku Q-ortogonality. Nasledujúcu vetu uvádzame bez dôkazu (čitateľ si ho môže naštudovať napríklad z [10, str.276]).

Veta 1.7. *(Veta o konjugovaných smeroch) Pre metódu konjugovaných gradientov, definovanú vzťahmi*

$$\begin{aligned} x_{k+1} &= x_k + \alpha_k d_k \\ d_{k+1} &= -g_{k+1} + \beta_k d_k, \qquad d_0 = -g_0 \\ \alpha_k &= -\frac{g_k^T d_k}{d_k^T Q d_k} \\ \beta_k &= \frac{g_{k+1}^T g_{k+1}}{g_k^T g_k} \end{aligned} \tag{15}$$

platia nasledujúce vlastnosti:

1. $d_k^T Q d_i = 0$ *pre* $0 \leq i < k$
2. $[g_0, g_1, \ldots, g_k] = [g_0, Qg_0, \ldots, Q^k g_0]$
3. $[d_0, d_1, \ldots, d_k] = [d_0, Qd_0, \ldots, Q^k d_0]$

Metóda konjugovaných gradientov, aplikovaná na rýdzokonvexnú kvadratickú funkciu, založená na vzťahoch (15), vytvára podľa vlastnosti 1 Vety 1.7 postupnosť Q-konjugovaných smerov. Vezmúc ľubovoľný bod x_0 z $\mathbb{R}^n$, uvedená metóda dosiahne podľa Vety 1.6 optimálne riešenie úlohy za nanajvýš n krokov.

1.3.2 Konvergencia pre všeobecnú konvexnú funkciu

V tejto časti textu sa budeme zaoberať konvergenciou metódy konjugovaných gradientov aplikovanej na všeobecnú konvexnú C^1 diferencovateľnú funkciu.

Pod pojmom konvergencie metódy chápeme dosiahnutie stavu, keď

$$g_k = \mathbf{0} \text{ pre nejaké } k \geq 0 \quad \text{alebo } \liminf_{k\to\infty} \|g_k\| = 0 \tag{16}$$

Podmienka $g_k = \mathbf{0}$, čo je ekvivalentné s $\|g_k\| = 0$, je nutnou a zároveň postačujúcou podmienkou nájdenia minima konvexnej funkcie, keďže v prípade takej funkcie je jej lokálne minimum zároveň aj jej globálnym minimom.

Začneme uvedením niekoľkých pojmov a definícií, s ktorými budeme ďalej pracovať. Podmienka efektívneho spádu je v algoritmoch často vyžadovaná.

Definícia 1.8. *Ak existuje taká konštanta $c > 0$, že*

$$g_k^T d_k < -c\|g_k\|^2 \tag{17}$$

pre všetky $k \geq 0$, potom hovoríme, že smer d_k spĺňa podmienku efektívneho spádu.

V každej iterácii MKG minimalizujeme funkciu $f(x)$ v smere vektora d_k. Riešime teda úlohu (3), ku ktorej môžeme pristúpiť dvoma spôsobmi.

Prvým je nájdenie riešenia α_k, ktoré je presným minimom funkcie jednej premennej $\varphi(\alpha) = f(x_k + \alpha d_k)$.

Druhým, častejšie používaným postupom, je nájdenie približného riešenia úlohy (3). Existuje niekoľko kritérií, ktoré by malo takéto riešenie spĺňať, za všetky spomenieme štandardné, silné a približné Wolfeho kritérium.

Štandardné Wolfeho kritérium pozostáva z nasledujúcich dvoch podmienok:

$$f(x_k + \alpha_k d_k) - f(x_k) \leq \delta \alpha_k g_k^T d_k \tag{18}$$

$$\sigma g_k^T d_k \leq g_{k+1}^T d_k \tag{19}$$

kde d_k je spádový smer a vzťah δ a σ je určený nerovnosťami $0 < \delta \leq \sigma < 1$.

Silné Wolfeho kritérium pozostáva z podmienky (18) a sprísnenia podmienky (19) na

$$-\sigma g_k^T d_k \geq |g_{k+1}^T d_k|, \tag{20}$$

pričom opäť $0 < \delta \leq \sigma < 1$.

V článku [6, str.181] autori W. Hager a H. Zhang predstavili tzv. *približné Wolfeho kritérium* v tvare

$$\sigma g_k^T d_k \leq g_{k+1}^T d_k \leq (2\delta - 1) g_k^T d_k \tag{21}$$

kde pre δ a σ platia vzťahy $0 < \delta < \frac{1}{2}$ a $\delta < \sigma < 1$.
Vidíme, že prvá nerovnosť v (21) je totožná so vzťahom (19). Autori sa zamerali na prvú podmienku štandardného Wolfeho kritéria. Ak označíme

$$\phi(\alpha) = f(x_k + \alpha d_k) \tag{22}$$

potom vzťah (18) nadobúda tvar

$$\frac{\phi(\alpha_k) - \phi(0)}{\alpha_k} \leq \delta \phi'(0)$$

W. Hager a H. Zhang v [6, str.180] upozorňujú, že táto podmienka je v blízkom okolí lokálneho minima funkcie $\phi(\alpha)$ neraz ťažko splniteľná vzhľadom na veľké nepresnosti vo výsledku rozdielu $\phi(\alpha_k) - \phi(0)$. Dôvodom je skutočnosť, že v spomínanom prípade $\phi(\alpha_k) \approx \phi(0)$. Pri operácii odčitovania dvoch približne rovnakých čísel dochádza v počítačovej aritmetike ku pomerne veľkej relatívnej chybe výpočtu.

Autori nahradili funkciu $\phi(\alpha)$ kvadratickou interpolačnou funkciou $q(\alpha)$, spĺňajúcou interpolačné podmienky $\phi(\alpha) = q(\alpha)$ v bode $\alpha = 0$ a $\phi'(\alpha) = q'(\alpha)$ v bodoch $\alpha = 0$ a $\alpha = \alpha_k$. Z týchto podmienok potom vyjadrili jednotlivé koeficienty interpolačnej funkcie $q(\alpha)$:

$$\phi(\alpha) = q(\alpha) = a\alpha^2 + b\alpha + c \qquad \text{v bode } \alpha = 0, \text{ čo vedie k}$$

$$\phi(0) = c \tag{23}$$

$$\phi'(\alpha) = q'(\alpha) = 2a\alpha + b \qquad \text{v bode } \alpha = 0, \text{ čo vedie k}$$

$$\phi'(0) = b \tag{24}$$

$$\phi'(\alpha) = q'(\alpha) = 2a\alpha + b \qquad \text{v bode } \alpha = \alpha_k, \text{ čo vedie k}$$

$$\frac{\phi(\alpha_k) - \phi'(0)}{2\alpha} = a \tag{25}$$

Použitím týchto vypočítaných výrazov získali interpolačnú funckia $q(\alpha)$ v tvare

$$q(\alpha) = \frac{\phi(\alpha_k) - \phi'(0)}{2\alpha}\alpha^2 + \phi'(0)\alpha + \phi(0) \tag{26}$$

Dosadením tejto funkcie do (18) dostaneme po jednoduchých úpravách druhú nerovnosť v (21). Výhodou tejto podmienky je jej vyššia presnosť výpočtov[1] v okolí lokálneho minima funkcie $\phi(\alpha)$ vďaka vyjadreniu prostredníctvom derivácií (podrobnejšie o tejto problematike v [6, str.180-181]).

Viaceré dôkazy konvergencie MKG vyžadujú dodatočné predpoklady o funkcii $f(x)$ ako napríklad:

Definícia 1.9. *(Lipšicovskosť gradientu) Hovoríme, že gradient $\nabla f(x)$ funkcie $f(x)$ je Lipšicovský v okolí $\mathbb{N}$, ak existuje konštanta $0 < L < \infty$ taká, že*

$$\|\nabla f(x) - \nabla f(y)\| \leq L\|x - y\| \qquad \textit{pre všetky } x, y \in \mathbb{N}, \tag{27}$$

kde $\mathbb{N}$ je nejaké okolie množiny

$$\mathbb{L} = \{x \in \mathbb{R}^n | f(x) \leq f(x_0)\}$$

Definícia 1.10. *(Predpoklad ohraničenosti) Množina $\mathbb{L}$ je ohraničená, ak existuje konštanta $K < \infty$ taká, že*

$$\|x\| \leq K \qquad \textit{pre všetky } x \in \mathcal{L} \tag{28}$$

[1] rádu strojového epsilon v porovnaní s $\sqrt{\text{stroj.eps.}}$ v prípade použitia prvej podmienky štandardného Wolfeho kritéria

Mnoho dôkazov konvergencie metódy konjugovaných gradientov pracuje s nasledujúcou vetou, známou aj ako Zoutendijkova podmienka [7, str.4].

Veta 1.11. *Uvažujme ľubovoľnú iteračnú metódu formy (2), kde d_k spĺňa podmienku spádovosti (4) a α_k spĺňa štandardné Wolfeho kritérium (18) a (19). Ak je splnená podmienka Lipšicovskosti gradientu (27), potom platí*

$$\sum_{k=0}^{\infty} \frac{(g_k^T d_k)^2}{||d_k||^2} < +\infty. \tag{29}$$

Jedným zo spôsobov, ako dokázať konvergenciu metódy konjugovaných gradientov, je popri Zoutendijkovej podmienke ukázať, že smer d_k spĺňa podmienku efektívneho spádu (17) a že existuje konštanta $m > 0$ taká, že platí:$\|d_k\| \leq m\ k$.
Spojenie týchto tvrdení umožňuje dokázať [7, str.5], že platí

$$\liminf_{k\to\infty} \|g_k\| = 0 \tag{30}$$

2 Algebraické tvary parametra β v MKG

V tejto kapitole sa budeme venovať historickému vývoju formy parametra β metódy konjugovaných gradientov. Podkapitoly 2.1 až 2.5 približujú tzv. štandardné voľby parametra, ktorých tvar sa v priebehu realizácie algoritmu nemení. Záverečná podkapitola prináša trochu odlišný pohľad na túto problematiku (dynamická voľba tvaru parametra β podľa situácie). Pri štandardných voľbách tvaru parametra β sa pokúsime priblížiť aj historické pozadie jeho vývoja, ale najmä jeho klady a zápory v súvislosti s problematikou konvergencie MKG.

2.1 Hestenes-Stiefel, 1952

V roku 1952 autori M.R. Hestenes a E. Stiefel uverejnili článok [9], v ktorom predstavili iteračnú metódu, nazvanú metóda konjugovaných gradientov. Riešili ňou systém n lineárnych rovníc $Ax = b$ o n neznámich, kde A je symetrická kladne definitná matica. Na problematiku riešenia tohto systému narazíme napríklad pri minimalizácii kvadratickej funkcie v tvare $f(x) = \frac{1}{2}x^T Ax - b^T x + c$. Riešenie systému $Ax = b$ metódou konjugovaných gradientov teda zároveň dáva optimálne riešenie minimalizačnej úlohy spomínanej konvexnej funkcie (pretože matica A je kladne definitná). MKG autorov M. Hestenesa a E. Stiefela dosahuje riešenie úlohy za nanajvýš n krokoch. Ich článok je zameraný na porovnanie uvedenej metódy s vtedy používanou Gaussovou eliminačnou metódou. Autori sa snažili ukázať, že, z hľadiska použitia vtedajšej výpočtovej techniky na ich realizáciu, MKG prevyšovala Gaussovu eliminačnú metódu.

Ich iteračná schéma sa riadi vzťahmi (2) a (5), pričom parameter β má tvar

$$\beta_k^{HS} = \frac{g_{k+1}^T y_k}{d_k^T y_k}, \qquad \text{kde} \quad y_k = g_{k+1} - g_k, \quad d_k^T y_k \neq 0, \quad k \geq 0 \tag{31}$$

Dosadením (31) do (5) dostaneme

$$d_{k+1} = -g_{k+1} + \frac{g_{k+1}^T y_k}{d_k^T y_k} d_k, \qquad k \geq 0. \tag{32}$$

Vynásobením výrazu (32) vektorom y_k sprava dostaneme

$$d_{k+1}^T y_k = -g_{k+1}^T y_k + \beta_k^{HS} d_k^T y_k = g_{k+1}^T(-y_k + y_k) = 0. \tag{33}$$

Všimnime si, že tento výsledok nezávisí od použitého spôsobu nájdenia optimálneho kroku α_k v MKG.

Uvažujme opäť kvadratickú funkciu $f(x) = \frac{1}{2}x^T Ax - b^T x + c$, kde A je symetrická kladne definitná matica. Pre g_k platí $g_k = Ax_k - b$. Preto

$$y_k = g_{k+1} - g_k = Ax_{k+1} - b - Ax_k + b = A(x_{k+1} - x_k) = Ap_k \tag{34}$$

Využime tieto poznatky pre vzťah (33).

$$0 = d_{k+1}^T y_k = d_{k+1}^T Ap_k = \alpha_k d_{k+1}^T Ad_k, \qquad \text{kde} \quad \alpha_k > 0, \tag{35}$$

preto $d_{k+1}^T Ad_k = 0$. Znamená to, že v prípade konvexnej kvadratickej funkcie spĺňa vektor d_{k+1} z (32) Definíciu 1.5 bez ohľadu na použitý spôsob nájdenia optimálneho kroku.

Skalár β_k^{HS} môže nadobúdať aj záporné hodnoty. Pre zabezpečenie konvergencie pre prípad všeobecnej konvexnej funkcie musí byť splnených niekoľko predpokladov (čerpáme z [7, str.10]). Smery d_k musia spĺňať (17) a v algoritme treba na hľadanie optimálneho kroku použiť kritérium (18) a (19). Nakoniec sa ešte parameter β^{HS} ohraničí na nezáporné hodnoty

$$\beta_k^{HS+} = \max\{\beta_k^{HS}, 0\}, \quad k \geq 0 \tag{36}$$

2.2 Fletcher-Reeves, 1964

V roku 1964 v článku [5] autori R. Fletcher a C. Reeves predstavili gradientnú optimalizačnú metódu na hľadanie voľného minima funkcie n premenných. Za významné vlastnosti ich metódy považovali jej jednoduchosť a primeranú pamäťovú náročnosť. Autori zvolili parameter β MKG v tvare

$$\beta_k^{FR} = \frac{\|g_{k+1}\|^2}{\|g_k\|^2}, \qquad k \geq 0 \tag{37}$$

Všimnime si, že β_k^{FR} nadobúda iba kladné hodnoty (až na prípad $||g_{k+1}|| = 0$, čo ale znamená koniec riešenia úlohy).

V prípade minimalizácie konvexnej kvadratickej funkcie a použitia presného optimálneho kroku α_k je dokázaná n-kroková konvergencia tejto metódy (Veta 1.6 a Veta 1.7).

Praktickou nevýhodou tejto metódy je náchylnosť na tzv. *zasekávanie* (preklad anglického výrazu jamming), po prvýkrát odhalenou v roku 1977 M.J.D. Powellom v článku [13]. Práve táto nevýhoda metódy FR ju často diskvalifikuje z použitia v praxi.

Poznámka: *Zasekávanie* je stav, v ktorom metóda robí veľké množstvo iterácií (používa veľmi malý krok) bez významného priblíženia sa k optimálnemu riešeniu $\hat{x}$.

Parameter β_k^{HS} (prezentovaný v predchádzajúcej podkapitole) je napríklad voči tejto situácii imúnny. V prípade nastatia *zasekávania* je vektor $x_{k+1} - x_k$ približne nulový. To znamená, že aj vektor $y_k = g_{k+1} - g_k$ je veľmi blízky nulovému, teda čitateľ výrazu (31) je blízky 0. Podľa (5) sa vektor d_{k+1} stáva vektorom $-g_{k+1}$, čo je smer najstrmšieho spádu. Parameter β_k^{HS} tak obsahuje pre túto situáciu akýsi vbudovaný reštart, keďže samotná metóda MKG začína použitím záporného gradientného smeru.

Konvergencia metódy FR v prípade nekvadratickej účelovej funkcie a použitia približného optimálneho kroku bola dokázaná v roku 1985 autorom M. Al-Baalim v [1]. Použitím silných Wolfeho podmienok (18) a (20) s hodnotou konštanty $\sigma < \frac{1}{2}$ metóda generuje smery s vlastnosťou (17) a na základe Vety 1.11 Al-Baali preukázal jej konvergenciu.

2.3 Polak-Riebi$\grave{e}$re-Polyak, 1969

PRP metóda[2] autorov E. Polaka, G. Riebièreho a B.T. Polyaka prezentovaná v textoch [11] a [12] používa parameter MKG v tvare

$$\beta_k^{PRP} = \frac{g_{k+1}^T y_k}{\|g_k\|^2}, \qquad \text{kde} \quad y_k = g_{k+1} - g_k, \quad k \geq 0 \tag{38}$$

Rovnako ako v prípade (31), aj táto metóda je vďaka špecifickému tvaru parametra β_k^{PRP} odolná voči tzv. *zasekávaniu* (z ang. výrazu jamming), ktorého podstatu sme vysvetlili v Poznámke na strane 25.
Podobne ako v prípade (31), aj parameter β_k^{PRP} môže nadobúdať záporné hodnoty.

V prípade silnokonvexnej účelovej funkcie a použitia presného optimálneho kroku metóda PRP konverguje (podrobne preskúmané v [11]).

Ak ale účelová funkcia nie je silnokonvexná alebo je v prípade silnokonvexnej funkcie použitý iba približný optimálny krok, konvergencia PRP metódy nie je zabezpečená. Na zabezpečenie konvergencie je podľa [14] potrebné upraviť tvar β^{PRP} (38) na

$$\beta_k^{PRP+} = \max\{0, \frac{g_{k+1}^T y_k}{||g_k||^2}\}. \tag{39}$$

Potom za predpokladu, že smer d_k spĺňa podmienku (17) a v algoritme je použité štandardné Wolfeho kritérium (18) a (19) na nájdenie optimálneho kroku, metóda PRP+ bude konvergovať (popísané v [7, str. 9]).

[2]slovným spojením PRP metóda označujeme MKG s tvarom parametra β autorov Polaka, Riebièreho, Polyaka

2.4 Dai-Yuan, 1999

V roku 1999 autori Y.H. Dai a Y. Yuan v článku [3] predstavili novú verziu MKG - DY metódu[3]. Najvýznamnejšou prednosťou tejto metódy je jej konvergencia za podmienky aplikácie štandardného Wolfeho kritéria (18) a (19) pre nájdenie optimálneho kroku a splnenia predpokladov Vety 1.11 (myšlienka dôkazu popísaná v článku [7]).

Autori použili tvar parametra MKG

$$\beta_k^{DY} = \frac{||g_{k+1}||^2}{d_k^T y_k}, \qquad \text{kde} \quad y_k = g_{k+1} - g_k, \quad d_k^T y_k \neq 0, \quad k \geq 0 \tag{40}$$

Smery d_k vytvárané v DY metóde sú za predpokladu, že krok α_k spĺňa podmienku (18) a (19) štandardného Wolfeho kritéria, vždy spádové. Pre smer d_k platí

$$g_k^T d_k = g_k^T(-g_k + \beta_{k-1}^{DY} d_{k-1}) = -\|g_k\|^2 + g_k^T d_{k-1} \frac{\|g_k\|^2}{d_{k-1}^T y_{k-1}} = \|g_k\|^2 \left(\frac{d_{k-1}^T g_k - d_{k-1}^T y_{k-1}}{d_{k-1}^T y_{k-1}} \right)$$

$$= \|g_k\|^2 \left(\frac{d_{k-1}^T g_{k-1}}{d_{k-1}^T g_k - d_{k-1}^T g_{k-1}} \right) \leq \|g_k\|^2 \left(\frac{d_{k-1}^T g_{k-1}}{(\sigma - 1) d_{k-1}^T g_{k-1}} \right) = \|g_k\|^2 (\frac{1}{\sigma - 1}) < 0$$

Podľa Lemy 1.2 je preto smer d_k spádový.

2.5 Hager-Zhang, 2005

MKG autorov W.W. Hagera a H. Zhanga patrí do jednoparametrickej triedy MKG. To znamená, že do formuly parametra β MKG zakomponovali nový parameter (ozn. θ). Ako sami píšu v článku [7, str.13], cieľom ich práce bolo vyvinúť verziu MKG, v ktorej by bola zabezpečená efektívna spádovosť (17) bez ohľadu na presnosť použitú pri hľadaní optimálneho kroku. Autori modifikovali parameter (31) nasledujúcim spôsobom:

$$\beta_k^{HZ} = \beta_k^{HS} - \theta_k (\frac{\|y_k\|^2 g_{k+1}^T d_k}{(d_k^T y_k)^2}), \qquad \text{kde } \theta_k \geq 0, \quad k \geq 0, \tag{41}$$

pričom predpokladajú $d_k^T y_k \neq 0$, aby bol parameter β_k^{HS} riadne definovaný.

[3] DY metóda označuje MKG s tvarom parametra β autorov Daia a Yuana

Tvrdenie 2.1. *V prípade použitia štandardného Wolfeho kritéria (18) a (19) platí* $d_k^T y_k > 0$

Dôkaz: Využijeme druhú podmienku štandardného Wolfeho kritéria (podmienka (19) na strane 19).

$$g_{k+1}^T d_k \geq \sigma g_k^T d_k, \qquad \text{pričom} \qquad 0 < \sigma < 1. \tag{42}$$

Od oboch strán nerovnosti (42) odčítame výraz $g_k^T d_k$

$$(g_{k+1} - g_k)^T d_k \geq (\sigma - 1) g_k^T d_k. \tag{43}$$

Vieme, že $(\sigma - 1) < 0$ a vektory d_k sú vytvárané ako spádové (Lema 1.2). Potom platí $y_k^T d_k \geq 0$ ■ Výraz $\frac{\|y_k\|^2 g_{k+1}^T d_k}{(d_k^T y_k)^2}$ v (41) má tri veľmi významné vlastnosti:

1. Je imúnny voči škálovaniu účelovej funkcie (β_k^{HZ} sa nemení pri prenásobení účelovej funkcie kladným skalárom).

2. Je odolný voči tzv. *zasekávaniu* (z ang. jamming), ktorého podstatu sme vysvetlili v Poznámke na strane 25.

Nech je použité približné Wolfeho kritérium (21). Potom podľa [7, str.13] platí

$$\frac{|g_{k+1}^T d_k|}{d_k^T y_k} \leq \frac{max\{\sigma, (1-2\delta)\}|g_k^T d_k|}{(1-\sigma)|g_k^T d_k|} = \frac{max\{\sigma, (1-2\delta)\}}{(1-\sigma)} \tag{44}$$

a ak smer d_k spĺňa podmienku (17), tak

$$\frac{1}{d_k^T y_k} \leq \frac{1}{(\sigma - 1)|g_k^T d_k|} = \frac{1}{(1-\sigma)(-g_k^T d_k)} \leq \frac{1}{(1-\sigma)c\|g_k\|^2}. \tag{45}$$

Spojením (44) a (45) a prenásobením $\|y_k\|^2$ dostávame

$$\frac{\|y_k\|^2 |g_{k+1}^T d_k|}{(d_k^T y_k)^2} \leq \left(\frac{max\{\sigma, (1-2\delta)\}}{c(1-\sigma)^2}\right)\left(\frac{\|y_k\|}{\|g_k\|}\right)^2. \tag{46}$$

Z (46) vyplýva, že ak nastáva situácia tzv. *zaseknutia* (pozri Poznámku na strane 25), celý výraz sa stáva zanedbateľným ($\|y_k\| \approx 0$ a $\|g_k\| > 0$). Parameter (41) sa zjednodušuje na parameter (31), ktorý je voči *zaseknutiu* odolný .

3. Vylepšuje efektívnu spádovosť (vzťah (17)) vytváraných smerov d_k.

Podľa [7, str.14] platí

$$g_{k+1}^T d_{k+1} \leq -\left(1 - \frac{1}{4\theta_k}\right) \|g_{k+1}\|^2. \tag{47}$$

To znamená, že pre $\theta_k = \tilde{\theta} > \frac{1}{4}$ smer d_k spĺňa podmienku (17) s konštantou $c = (1 - (4\tilde{\theta})^{-1})$.

Parameter θ_k verzie Hagera a Zhanga je relatívnou váhou medzi dôrazom na efektívny spád (vzťah 17) a podmienku konjugovanosti MKG (35). Pre $\theta_k \to 0$ sa stáva parameter (41) parametrom (31), o ktorom sme už v podkapitole 2.1 ukázali, že spĺňa (35). Naopak pre $\theta_k \to \infty$ sa konštanta $c = (1 - (4\theta_k)^{-1})$ blíži k 1, čo podporuje efektívnu spádovosť vektora d_k. Autori na základe svojich experimentálnych výsledkov zvolili v článku [6] hodnotu $\tilde{\theta} = 2$.

V článku [6] autori dokázali konvergenciu MKG s parametrom β^{HZ} pre prípad silnokonvexnej účelovej funkcie. V dôkaze predpokladali, že

1. $f(x)$ je silnokonvexná funkcia na množine $\mathbb{L} = \{x \in \mathbb{R}^n : f(x) \leq f(x_0)\}$.

2. $f(x)$ má Lipšicovský gradient (Definícia 1.9).

3. Pri hľadaní optimálneho kroku α_k je použité Wolfeho kritérium (18) a (19).

Pre funkcie, ktoré nie sú silnokonvexné, konvergencia MKG s parametrom β^{HZ} zabezpečená nie je. Autori navrhli upraviť parameter β^{HZ} v tvare (41) na

$$\beta_k^{HZ+} = \max\{\beta_k^{HZ}, \eta_k\}, \qquad \text{kde} \quad \eta_k = \frac{-1}{\|d_k\| \min\{\eta, \|g_k\|\}}, \tag{48}$$

kde $\eta > 0$. Podrobný dôkaz konvergencie je uvedený v [6].

2.6 Hybridné algoritmy

Na základe predchádzajúcich podkapitol možno spomínané parametre metódy konjugovaných gradientov rozdeliť nasledovne

- parametre β^{FR} a β^{DY} síce za určitých predpokladov teoreticky konvergujú, no v praxi častokrát podliehaju tzv. *zasekávaniu*, čo ich jednoznačne vylučuje z použitia v praktických algoritmoch
- parametre β^{HS} a β^{PRP} sú naopak voči tzv. *zasekávaniu* imúnne, no ich konvergencia nie je zabezpečená

Pre tieto dôvody je prirodzená snaha o skombinovanie týchto parametrov tak, aby výsledok odstránil už uvedené negatívne aspekty správania sa týchto parametrov. Takéto kombinácie parametrov dávajú priestor novej triede algoritmov MKG, nazývanej aj *hybridné algoritmy*.

Ich algoritmus výberu parametra β_k je založený na výbere jedného z parametrov na základe situácie charakterizovanej určitou nerovnicou. Ako príklad takej metódy uvádzame verziu D. Touati-Ahmeda a C. Storeyho uvedenú v [7, str.10], ktorá kombinuje parametre Fletchera-Reevesa a Polaka-Ribiereho-Polyaka podmienkou

$$\beta_k = \begin{cases} \beta_k^{PRP} & \text{ak } 0 \leq \beta_k^{PRP} \leq \beta_k^{FR} \\ \beta_k^{FR} & \text{inak} \end{cases} \tag{49}$$

Cieľom takejto úpravy je zamedziť tzv. *zasekávaniu* metódy Fletchera-Reevesa. V takej situácii $0 \approx \beta_k^{PRP} \leq \beta_k^{FR} \approx 1$, čo je prvá vetva výrazu (49) a teda relatívna váha pri tvorbe d_k sa posunie ku Cauchyovskému spádovému smeru $-g_k$, ktorý je najlepším riešením vzniknutého problému. Ďalšie verzie hybridných algoritmov od iných autorov nájde čitateľ napríklad v [7, str.10].

3 Algoritmy

3.1 Generátory úloh

3.1.1 Generátor kvadratických úloh

Pre potreby experimentov sme vytvorili generátor kvadratických úloh

$$\min \ \{f(x) = \frac{1}{2}x^T Gx + h^T x | x \in \mathbb{R}^n\} \tag{50}$$

s vopred stanoveným optimálnym riešením $\hat{x} \in \mathbb{R}^n$, kde G je kladne definitná symetrická nxn matica a $x, h \in \mathbb{R}^n$.

Vstupy programu:

- n - rozmer požadovanej úlohy (dĺžka vektora $\hat{x}$)
- $\kappa(G)$ - číslo podmienenosti matice G
- ρ - vzdialenosť štartovacieho bodu x_0 od optimálneho riešenia $\hat{x}$ v euklidovskej norme

Výstupy programu:

- kladne definitná symetrická matica G
- optimálne riešenie $\hat{x}$ úlohy (50)
- štartovací bod x_0 a vektor h

Postup:

1. Generovanie optimálneho riešenia $\hat{x}$ úlohy (50).
2. Generovanie kladne definitnej symetrickej matice G s požadovaným číslom podmienenosti $\kappa(G)$. Pri konštruovaní matice G použitá náhodná dolná trojuholníková matica L, náhodná regulárna matica A, náhodná ortogonálna matica Q a špeciálna diagonálna matica D. Schéma konštrukcie

$$LL^T = A \to QR(A) = Q \to QDQ^T = G, \tag{51}$$

kde QR(A) označuje štandardnú funkciu programu Matlab realizujúcu QR rozklad regulárnej matice A.

3. Generovanie vektora $h \in \mathbb{R}^n$ tak, aby bol vektor $\hat{x}$ optimálnym riešením vygenerovanej úlohy (50), teda $h = -G\hat{x}$.

4. Generovanie štartovacieho bodu $x_0 \in \mathbb{R}^n$.

Generátor (zdrojový kód sa nachádza v prílohe A.1, str. 74) sa skladá zo štyroch blokov.

Blok 1 vytvára optimálne riešenie $\hat{x}$ kvadratickej úlohy (50) ako celočíselný n-zložkový náhodný vektor z rovnomerného rozdelenia na intervale $\langle -\omega, \omega \rangle$, kde ω predstavuje kladnú konštantu. V našom programe ω=10. Veľkosť ω netreba preháňať, aby funkčná hodnota v optime nebola zbytočne veľká.

Blok 2 generuje symetrickú kladne definitnú maticu G. Ako je načrtnuté v (51), tento úsek generátora najprv vytvára náhodnú regulárnu maticu $A = LL^T$, kde L je dolná trojuholníková matica s kladnou hlavnou diagonálou. Z QR rozkladu matice A využíva iba ortogonálnu maticu Q rozmerov nxn. Na vytvorenie matice G podľa (51) potrebuje diagonálnu maticu D s kladnou diagonálou a zadaným číslom podmienenosti $\kappa(G)$. Číslom podmienenosti kladne definitnej diagonálnej matice je podiel jej najväčšej a najmenšej vlastnej hodnoty. Keďže v prípade diagonálnej matice sú prvky jej diagonály zároveň jej vlastnými hodnotami, platí

$$\kappa(D) = \frac{\lambda_{max}}{\lambda_{min}} = \frac{d_{max}}{d_{min}}, \tag{52}$$

kde d_{max} označuje najväčší a d_{min} naopak najmenší prvok diagonály D. Stačí preto vytvoriť kladnú diagonálnu maticu s prvkami z intervalu $\langle d_{min}, d_{max} \rangle$. Rozloženie vlastných hodnôt na intervale $\langle \lambda_{min}, \lambda_{max} \rangle$ bolo predmetom experimentu v podkapitole 4.1. Blok 2 končí vytvorením matice $G = QDQ^T$.

Blok 3 generuje vektor $h \in \mathbb{R}^n$. Keďže je bod $\hat{x}$ optimálnym riešením generovanej kvadratickej úlohy (50), musí spĺňať nutnú podmienku optima

$$\left.\frac{df(x)}{dx}\right|_{x=\hat{x}} = \left.\frac{d(\frac{1}{2}x^TGx + h^Tx)}{dx}\right|_{x=\hat{x}} = G\hat{x} + h = 0 \tag{53}$$

Odtiaľ pre vektor h musí platiť

$$h = -G\hat{x} \tag{54}$$

Blok 4 generuje štartovací bod x_0 tak, aby spĺňal podmienku $\|x_0 - \hat{x}\| = \rho$. Bod x_0 je konštruovaný ako priesečník sféry s polomerom ρ so stredom v bode $\hat{x}$ a náhodnej polpriamky s počiatočným bodom $\hat{x}$.

3.1.2 Generátor bikvadratických úloh

Pre potreby experimentov generujeme bikvadratické úlohy

$$\min \{f(x) = \frac{1}{4}(x^T Dx)^2 + \frac{1}{2}x^T Gx + h^T x | x \in \mathbb{R}^n\} \tag{55}$$

s vopred stanoveným optimálnym riešením $\hat{x} \in \mathbb{R}^n$, kde D je diagonálna kladne definitná matica, G je symetrická kladne definitná matica a $x, h \in \mathbb{R}^n$.
Vstupy programu:

- n - rozmer požadovanej úlohy (rozmer riešenia $\hat{x}$)
- $\kappa()$ - číslo podmienenosti matíc G a D
- ρ - vzdialenosť štartovacieho bodu x_0 od optimálneho riešenia $\hat{x}$ v euklidovskej norme

Výstupy programu:

- kladne definitná symetrická matica G
- kladne definitná diagonálna matica D
- optimálne riešenie $\hat{x}$ úlohy (55)
- štartovací bod x_0 a vektor h

Generátor (zdrojový kód sa nachádza v prílohe A.2) pracuje na rovnakých princípoch ako generátor kvadratických úloh z prechádzajúcej podkapitoly 3.1.1.

Tvorba optimálneho riešenia bikvadratickej funkcie zodpovedá bloku 1 generátora kvadratických úloh podkapitoly 3.1.1.

Tvorba štartovacieho bodu s požadovanou vzdialenosťou od optimálneho riešenia $\hat{x}$ zodpovedá bloku 4 generátora kvadratických úloh podkapitoly 3.1.1.

Symetrickú kladnedefinitnú maticu G a diagonálnu kladne definitnú maticu D získame postupom totožným bloku 2 generátora kvadratických úloh podkapitoly 3.1.1.

Blok 3 generátora bikvadratických úloh vytvára vektor h. Keďže optimálne riešenie $\hat{x}$ úlohy (55) musí spĺňať

$$\left.\frac{df(x)}{dx}\right|_{x=\hat{x}} = \left.\frac{d(\frac{1}{4}(x^T Dx)^2 + \frac{1}{2}x^T Gx + h^T x)}{dx}\right|_{x=\hat{x}} = 0, \tag{56}$$

teda

$$(\hat{x}^T D\hat{x})D\hat{x} + G\hat{x} + h = 0, \tag{57}$$

vektor h je jednoznačne určený ako

$$h = -(\hat{x}^T D\hat{x})D\hat{x} - G\hat{x} \tag{58}$$

3.2 Programy určujúce optimálny krok α_k

V každej iterácii MKG riešime úlohu (3), ktorá zodpovedá hľadaniu minima konvexnej funkcie jednej premennej. Účelom programov popisovaných v tejto podkapitole je nájsť riešenie takéhoto problému.

3.2.1 Program aplikujúci metódu zlatého rezu

Úlohou programu je nájsť minimum konvexnej funkcie jednej premennej $\varphi(\alpha) = f(x + \alpha d)$. Naša matlabovská realizácia nazvaná *Zlatyrez* je uvedená v prílohe A.3 na strane 77.

Vstupy programu:

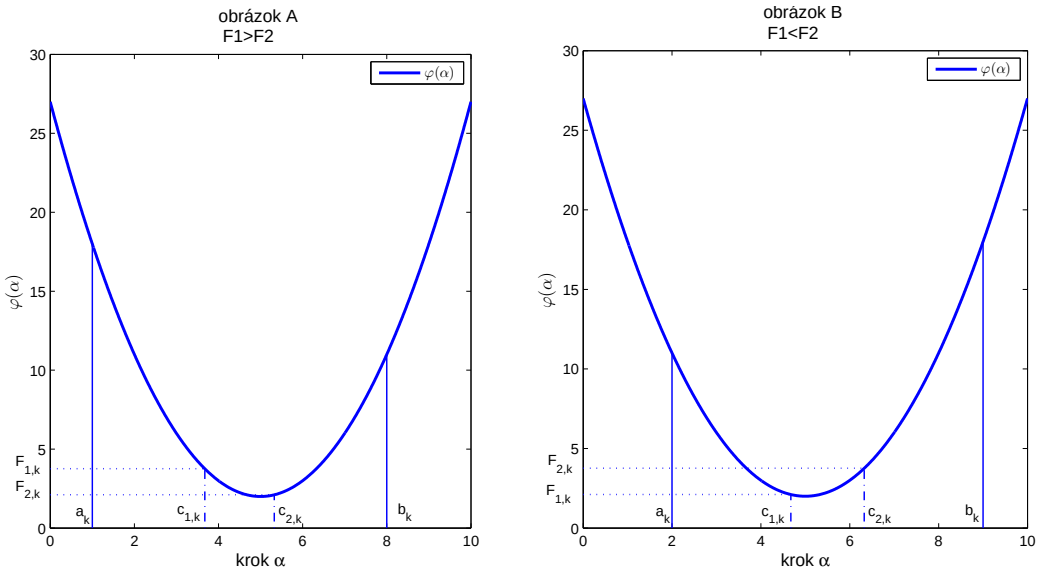
- x_k - bod, v ktorom sa aktuálne nachádza MKG
- d_k - vektor udávajúci smer minimalizácie funkcie $f(x)$
- w - fixný počet iterácií, ktoré má program vykonať

Výstupy programu:

- α - aproximácia optimálneho kroku α_k

Algoritmus

Uvažujme situáciu zobrazenú na ľavej časti Obr. 1. Máme informáciu o funkčných hodnotách $\varphi(a_k)$ a $\varphi(b_k)$. Na začiatku programu $k = 0$, kde k reprezentuje k-te upresnenie pôvodného intervalu $< a_0, b_0 >$. Algoritmus programu postupuje v troch krokoch.



Obr. 1: K-ta iterácia metódy zlatého rezu pre dva rôzne prípady.

Krok 1 Algoritmus určí na intervale $I_k =< a_k, b_k >, \quad k \geq 0$ dva nové body $c_{1,k}$ a $c_{2,k}$ podľa predpisu

$$c_{i,k} = a_k + z_i(b_k - a_k), \quad i = 1, 2 \tag{59}$$

kde $z_2 = \frac{\sqrt{5}-1}{2}$ a $z_1 = 1 - z_2$. Pre tieto špeciálne zvolené body platí

$$c_{1,k} + c_{2,k} = a_k + b_k \quad \text{a} \quad \frac{c_{1,k}}{c_{2,k}} = \frac{c_{2,k}}{b_k}. \tag{60}$$

Krok 1 algoritmu končí zaznamenaním funkčných hodnôt funkcie $\varphi(\alpha)$ v bodoch $c_{1,k}$ a $c_{2,k}$ (označíme $F_{1,k}$ a $F_{2,k}$).

Krok 2 Na základe funkčných hodnôt $F_{1,k}$ a $F_{2,k}$ algoritmus rozhodne o spresnení intervalu I_k na nový interval $I_{k+1} =< a_{k+1}, b_{k+1} >$, pričom $I_{k+1} \subset I_k$, $|I_{k+1}| =$

$z_2|I_k|$. Pri určovaní intervalu I_{k+1} sa algoritmus riadi nasledujúcim kritériom:

- Ak $F_{1,k} > F_{2,k}$, potom sa minimum nachádza medzi bodmi $c_{1,k}$ a b_k. Preto

$$a_{k+1} = c_{1,k}, \quad b_{k+1} = b_k, \quad I_{k+1} =< a_{k+1}, b_{k+1} > .$$

  Navyše, bod $c_{2,k}$ intervalu I_k je totožný bodu $c_{1,k+1}$ intervalu I_{k+1}. Situácia je vyobrazená na ľavej časti Obr. 1 (obrázok A).

- Ak $F_{1,k} < F_{2,k}$, potom sa minimum nachádza medzi bodmi a_k a $c_{2,k}$. Preto

$$a_{k+1} = a_k, \quad b_{k+1} = c_{2,k}, \quad I_{k+1} =< a_{k+1}, b_{k+1} > .$$

  Navyše, bod $c_{1,k}$ intervalu I_k je totožný bodu $c_{2,k+1}$ intervalu I_{k+1}. Situácia je vyobrazená na pravej časti Obr. 1 (obrázok B).

Kým hodnota k nedosiahne hodnotu vstupného parametra w, algoritmus pokračuje krokom 3.

Krok 3 Algoritmus by mal opäť vykonať krok 1. Situácia sa však pozmenila, nakoľko existuje informácia o funkčnej hodnote až v troch rôznych bodoch (a_{k+1}, b_{k+1}, $c_{i,k+1}$ $i = 1 \quad \vee \quad i = 2$). Preto algoritmus vytvorí a vypočíta funkčnú hodnotu iba v jednom ďalšom bode (ak už existuje $c_{1,k+1}$, potom vytvorí $c_{2,k+1}$ a naopak). Následne pokračuje krokom 2.

Naša matlabovská realizácia *Zlatyrez* sa v princípe nelíši od hore popísaného algoritmu. Súčasťou programu *Zlatyrez* je pomocná funkcia, ktorú sme nazvali *Prvá fáza*. Krok 1 algoritmu zlatého rezu totiž potrebuje vstupný interval I_0, ktorý ale na začiatku nepoznáme.

Cieľom funkcie *Prvá fáza* je nájsť interval I, ktorý obsahuje bod $\hat{\alpha} = \arg\min \varphi(\alpha)$. Pri tomto hľadaní používa funkčné hodnoty funkcie $\varphi(\alpha)$. Vypočíta funkčnú hodnotu v bodoch α_k a α_{k+1} (na začiatku k=0) podľa

$$\alpha_{k+1} = \alpha_k + k(\text{konšt}), \quad k \geq 1, \quad \alpha_{-1} = 0, \quad \alpha_0 = 0, \quad \alpha_1 = 1, \tag{61}$$

kde konšt predstavuje fixnú konštantu (v našom programe konšt=$\frac{1+\sqrt{5}}{2}$). Ak

$$\varphi(\alpha_k) \leq \varphi(\alpha_{k+1}), \quad k \geq 0 \tag{62}$$

potom $I =< \alpha_{k-1}, \alpha_{k+1} >$. Inak funkcia zvýši k o 1 a podmienku (62) overí pre patričnú dvojicu bodov.

Bolo tiež potrebné ošetriť porovnávanie funkčných hodnôt $F_{1,k}$ a $F_{2,k}$. Ak by boli hodnoty príliž blízko seba, numerické chyby by mohli spôsobiť nesprávny výsledok rozhodovacieho kritéria. Preto ešte pred samotným porovnávaním hodnôt overujeme, či relatívna chyba rozdielu $F_{2,k} - F_{1,k}$ nie je za hranicou počítačovej presnosti. V prípade pozitívnej odpovede program ihneď zastavíme, nakoľko rozhodovacie kritérium by už nebolo dôveryhodné. Za optimálny krok potom používame bod b_k posledného získaného intervalu I_k.

3.2.2 Program aplikujúci algoritmus W. Hagera a H. Zhanga

Program LSHZ v prílohe A.4 na strane 79 je upravenou realizáciou návrhu prezentovaného v článku [6, str. 184] autorov W. Hagera a H. Zhanga. Ide o metódu nájdenia približného optimálneho kroku α_k v k-tej iterácii MKG. Cieľom autorov W. Hagera a H. Zhanga bolo vytvoriť algoritmus s vyššou presnosťou a efektívnosťou v porovnaní s konkurenčnými algoritmami.
Vstupy programu:

- x_k - bod, v ktorom sa aktuálne nachádza MKG
- d_k - vektor udávajúci smer minimalizácie
- Maxit - maximálny počet iterácií, ktorý umožníme programu vykonať

Výstupy programu:

- α - aproximácia optimálneho kroku α_k

V prípade programu *Zlatyrez* sme hľadanie minima konvexnej funkcie jednej premennej $\varphi(\alpha) = f(x + \alpha d)$ realizovali prostredníctvom porovnávania funkčných hodnôt v dvoch špeciálne zvolených bodoch intervalu I_k. Autori W. Hager a H. Zhang sa

zamerali na hľadanie nulovej deriváciu funkcie $\varphi(\alpha)$. Motiváciou bola lepšia presnosť výpočtu derivácie v porovnaní s výpočtom funkčnej hodnoty v počítačovej aritmetike.

Princíp metódy spočíva v nájdení dvoch bodov a_0 a b_0 takých, aby spĺňali

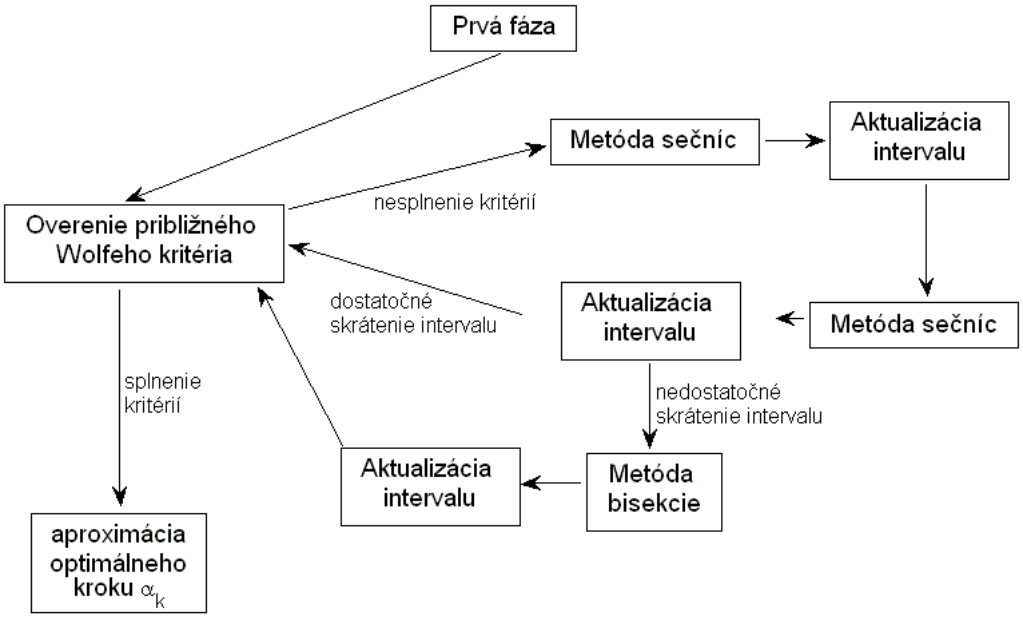
$$\varphi'(a_0) < 0, \quad \varphi(a_0) \leq \varphi(0), \quad \varphi'(b_0) \geq 0. \tag{63}$$

Tieto body budú tvoriť interval I_0, ktorý určite obsahuje hľadaný optimálny krok α_k, pre ktorý platí $\varphi'(\alpha_k) = 0$. Ak ani jeden z krajných bodov tohto intervalu nebude považovaný za dostatočnú aproximáciu optimálneho kroku (presnejšie vysvetlené nižšie), autori W. Hager a H. Zhang navrhli algoritmus spresnenia intervalu I_j na $I_{j+1}, \quad j \geq 0$, kde j reprezentuje číslo vnútornej iterácie programu LSHZ, pričom

$$I_{j+1} \subset I_j \quad \text{a zároveň} \quad |I_j| \to 0 \text{ pre } j \to \infty. \tag{64}$$

Preto pre dostatočne veľké j algoritmus nájde vyhovujúcu aproximáciu optimálneho kroku α_k. V nasledujúcej časti popíšeme schému algoritmu autorov W. Hagera a H. Zhanga, jeho úpravy a programovú realizáciu, ktorú sme urobili.

Program sa riadi nasledujúcou schémou



Obr. 2: Schéma programu LSHZ.

Prvá fáza je časť programu LSHZ, ktorej úlohou je podobne, ako v prípade pomocnej funkcie *Prvá fáza* programu *Zlatyrez*, nájsť počiatočný interval $I_0 =< a_0, b_0 >$, ktorý obsahuje optimálny krok α_k. Body a_0, b_0 musia spĺňať (63). Algoritmus je založený na zisťovaní znamienka derivácie funkcie $\varphi(\alpha)$. Koniec nastáva vtedy, ak je objavený bod b_0, pre ktorý platí $\varphi'(b_0) \geq 0$. Výstupom je interval I_0.

Overovaním približných Wolfeho podmienok je kontrolovaná dostatočnosť aproximácie optimálneho kroku. Ide o podmienky (21), ktorých odvodenie sme priblížili v podkapitole 1.3 na strane 20. Podmienky sa overujú pre krajné body intervalu I_j. V prípade ich splnenia získavame aproximáciu optimálneho kroku α_k a program LSHZ končí. V opačnom prípade program pokračuje ďalej.

Na získanie presnejšej aproximácie optimálneho kroku α_k autori W. Hager a H. Zhang navrhli použiť lineárnu interpoláciu v podobe *metódy sečníc*. Výstupom je bod, ktorý sme označili c.

Po získaní bodu c program LSHZ rozhoduje o jeho polohe vzhľadom na optimálny krok α_k. To sa deje v pomocnom programe *Aktualizácia intervalu*.

- Ak sa bod c nachádza mimo aktuálneho intervalu I_j, potom neposkytuje žiadnu dodatočnú informáciu o polohe optimálneho kroku.

- Ak $c \in I_j$ a platí $\varphi'(c) > 0$, potom sa optimálny krok nachádza naľavo od bodu c. To umožňuje upresniť interval I_j na $I_j =< a_j, c >$.

- Ak $c \in I_j$ a platí $\varphi'(c) < 0$, potom sa optimálny krok nachádza napravo od bodu c. To umožňuje upresniť interval I_j na $I_j =< c, b_j >$.

Autori W. Hager a H. Zhang zároveň navrhli v j-tej iterácii zopakovať metódu sečníc a aktualizáciu intervalu I_j ešte raz. Zdôvodnenie uvádzajú v [6, str. 183]. Musíme tiež upozorniť, že algoritmus pomocného programu *Aktualizácia intervalu* je v ich podaní omnoho komplikovanejší (podrobne v [6, str. 182]). V našej práci ale pracujeme iba s konvexnými funkciami, preto sme mohli ich algoritmus zjednodušiť.

Po upresnení intervalu I_j program LSHZ overuje, či prišlo k dostatočnému skráteniu jeho dĺžky. Podmienkou je $|I_j| <$ konšt $|I_{j-1}|$, kde veľkosť konšt určuje riešiteľ (v našej realizácii používame konšt= 0,66).

- Ak sa interval I_j skrátil dostatočne, program pokračuje overovaním približných Wolfeho podmienok.
- Ak $|I_j| >$ konšt $|I_{j-1}|$, autori W. Hager a H. Zhang navrhli získať bod c *metódou bisekcie* a interval I_j aktualizovať. Následne program LSHZ pokračuje overovaním približných Wolfeho podmienok.

Zmyslom tejto časti programu LSHZ je zabezpečiť splnenie (64).

Overovaním približných Wolfeho podmienok (21) sa končí vnútorná iterácia j programu LSHZ a začína sa $j+1$ iterácia. Počas nášho experimentovania sme sa nestretli so situáciou, v ktorej program LSHZ urobil viac ako 2 vnútorné iterácie. Z programátorského hľadiska je ale potrebné zakomponovať určitú bezpečnostnú poistku v zmysle horného ohraničenia počtu vnútorných iterácií. Preto jedným zo vstupov programu je aj konštanta MAXit.

Program LSHZ s podrobným komentárom všetkých obsiahnutých funkcií sa nachádza v prílohe A.4 na strane 79.

3.3 Programová realizácia MKG

Programové realizácie MKG sa nachádzajú v prílohách A.5 a A.6. Ide o realizácie metódy konjugovaných gradientov s parametrami MKG v tvaroch β_k^{FR} a β_k^{HZ+}.
Vstupy programu:

- x_0 - štartovací bod úlohy
- *macro* - maximálny počet makroiterácií programu
- *Fval* - predpis účelovej funkcie
- *Fder* - predpis derivácie účelovej funkcie

Výstupy programu:

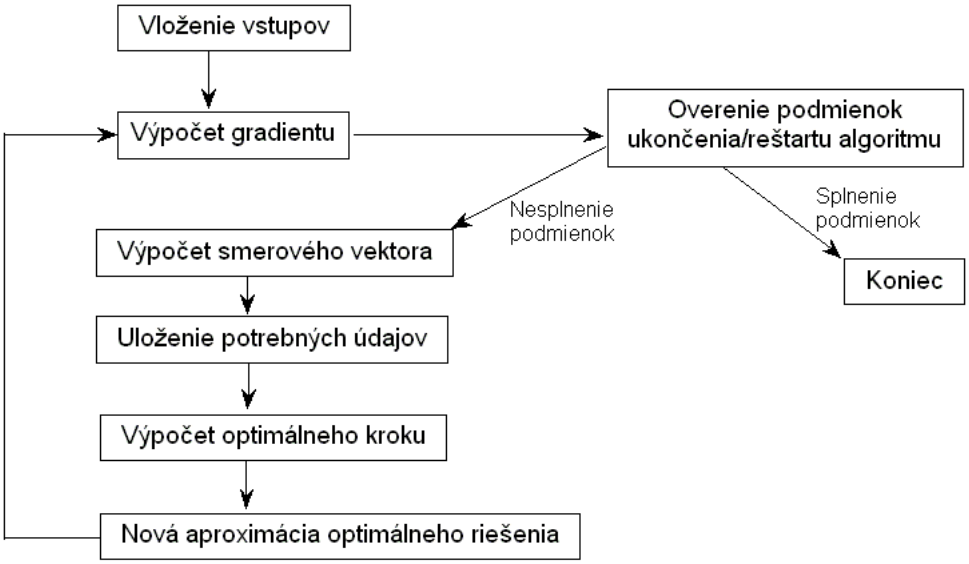
- x_1- optimálne riešenie úlohy

Popis programu:

V texte budeme používať pojmy mikro- a makroiterácia. Za jednu mikroiteráciu považujeme postupnosť príkazov od výpočtu smeru d_k po výpočet smeru d_{k+1}.

Program môže urobiť maximálne $n-1$ takýchto mikroiterácií, kde n označuje rozmer úlohy. Dôvodom je, že po skončení $n-1$ miktoiterácie už máme n lineárne nezávislých vektorov d_k, ktoré tvoria celú bázu priestoru. Preto algoritmus reštartujeme a poslednú získanú aproximáciu optimálneho riešenia použijeme ako nový štarovací bod. Makroiterácie budú počet týchto reštartov zaznamenávať.

Všetky realizácie je možné popísať spoločnou schémou



Obr. 3: Schéma programu MKG.

Blok Vloženie vstupov záhrňa okrem uloženia vstupných parametrov programu aj deklaráciu všetkých pomocných premenných a konštánt, ktoré bude program potrebovať. Ide aj o konštanty, ktoré používajú prípadné pomocné programy (ako napríklad programy *Zlatyrez* alebo LSHZ).

Výpočet gradientu- gradient je jedným zo stavebných kameňov MKG.

Nasleduje Overenie podmienok ukončenia/reštartu algoritmu. V závislosti od nastavenia týchto podmienok môžu nastať tieto situácie

- Reštart algoritmu nastane vtedy, ak počet mikroiterácií dosiahne hodnotu $n-1$, kde n označuje rozmer úlohy. V takom prípade už totiž program vytvoril maximálny počet lineárne nezávislých vektorov d_k. Ďalším dôvodom je naakumulovanie výpočtových chýb, preto je reštart algoritmu nielen nutný, ale aj výhodný.

- V prípade, ak nezadáme žiadne špeciálne podmienky ukončenia programu, program sa ukončí, akonáhle počet makroiterácií dosiahne hodnotu vstupného parametra *makro*.

- V prípade špecifikovania podmienok ukončenia algoritmu sú tieto podmienky overované spolu s predchádzajúcou podmienkou. Môže ísť o podmienky typu

  1. $\|g_k\| < \varepsilon$,
  2. $\|x_k - \hat{x}\| < \varepsilon$,
  3. $\|f(x_k) - f(\hat{x})\| < \varepsilon$,

  kde ε je nami stanovená konštanta.
  Podmienku 2 a 3 je samozrejme možné použiť iba v prípade, že optimálne riešenie $\hat{x}$ poznáme. To je napríklad situácia riadeného experimentu.

V prípade nesplnenie podmienok ukončenia programu nasleduje výpočet nového smeru d_k podľa (5). V závislosti od názvu realizácie MKG je použitý príslušný parameter β_k MKG.

Časť Uloženie potrebných údajov je pasáž, v ktorej program ukladá tie informácie, ktoré sa použijú vo výpočtoch v nasledujúcej iterácii. Ide v prvom rade o bod x_k a gradient g_k. Ostatné premenné sa v priebehu ďalších výpočtov prepíšu.

Nasleduje výpočet optimálneho kroku α_k. Spôsob výpočtu závisí od účelovej funkcie. V prípade konvexnej kvadratickej funkcie je možné použiť vzorec z Vety 1.6. Alternatívami sú metódy popísane v predchádzajúcich podkapitolách 3.2.1 a 3.2.2.

Podľa (2) program aproximuje optimálne riešenie $\hat{x}$ novým bodom x_{k+1}. Následne sa program vracia do časti Výpočet gradientu a opakuje celý algoritmus.

Realizácie v prílohách A.5 a A.6 sú v tzv. základom tvare. To znamená, že nie sú špecifikované žiadne špeciálne podmienky ukončenia programu. Program skončí po dosiahnutí požadovaného počtu maktoiterácií. V niektorých experimentoch však budeme špeciálne podmienky ukončenia používať. V takom prípade na to vopred upozorníme.

4 Pomocné experimenty

4.1 Číslo podmienenosti matice a rozloženie jej vlastných čísel

Cieľ experimentu

Cieľom tohto experimentu je preskúmať vzťah medzi rozložením vlastných čísel matice G z kvadratickej funkcie $f(x) = \frac{1}{2}x^T Gx + h^T x$, ktorú vytvárame v podkapitole 3.1.1 a počtom iterácií MKG potrebných na dosiahnutie určitej presnosti v jej funkčnej hodnote pre rôzne rozmery úloh.

Pozadie experimentu

Budeme rozlišovať 6 typov rozloženia vlastných čísel:

1. Rovnomerná ,ostromonotónna postupnosť vlastných čísel.
2. Prvú polovicu vlastných čísel bude tvoriť λ_{min}, zvyšok bude rovnomerná ,ostromonotónna postupnosť.
3. Tento typ je veľmi podobný predchádzajúcemu, ale s tým rozdielom, že prvá polovica bude rovnomerná ,ostromonotónna postupnosť vlastných čísel a zvyšok bude tvoriť λ_{max}.
4. Tretinu tvorí λ_{min}, druhú tretinu rovnomerná ostromonotónna postupnosť vlastných čísel (od λ_{min} po λ_{max}) a zvyšok λ_{max}.
5. Vlastné hodnoty budú z rovnomerného rozdelenia,ale neusporiadame ich do monotónnej postupnosti.
6. Tento typ bude pozostávať z dvoch skupín vlastných hodnôt. Vrámci skupiny sa hodnoty budú od seba iba nepatrne líšiť o malé konštantné ϵ. Pre lepšiu predstaviteľnosť môžeme vlastné hodnoty tohto typu rozloženia napísať ako

$$\lambda_{min}, \lambda_{min} + \epsilon, \lambda_{min} + 2\epsilon, \ldots, \lambda_{min} + \frac{n}{2}\epsilon, \lambda_{max} - \frac{n}{2}\epsilon, \lambda_{max} - \left(\frac{n}{2} - 1\right)\epsilon, \ldots, \lambda_{max}$$

kde n je rozmer úlohy a pre ϵ platí $\epsilon \ll \lambda_{min}$

Číslo podmienenosti matice G je rovnaké ako číslo podmienenosti kladne definitnej diagonálnej matice D (vyplýva to zo spôsobu konštrukcie matice G popísanej v podkapitole 3.1.1). Navyše, podľa (52) je toto číslo určené ako podiel najväčšej a najmenšej zložky diagonály matice D.

Predpokladajme, že chceme vytvoriť maticu G s číslom podmienenosti $\kappa(G) = 1000$. Takúto maticu môžeme podľa (52) vytvoriť prostredníctvom viacerých diagonálnych matíc D, ktoré sa značne odlišujú vo veľkosti svojich zložiek. Ak diagonálnu maticu D s najväčšou vlastnou hodnotou λ_{max} a najmenšou vlastnou hodnotou λ_{min} označíme $D(\lambda_{min}, \lambda_{max})$, môžeme použiť napríklad matice

$$D(0.001, 1) \qquad D(1, 1000) \qquad D(10^3, 10^6)$$

V našom algoritme sme používali $D(10^{-3}, \kappa(G)/10^3)$ z toho dôvodu, aby sme zbytočne nevytvárali veľmi veľké funkčné hodnoty. Máme teda určené λ_{min},λ_{max} a rozloženie intervalu $\langle \lambda_{min}, \lambda_{max} \rangle$.

Postup experimentu

Kvadratickú úlohu budeme riešiť programom *Metóda Fletchera-Reevesa* (v prílohe A.5) s použitím formulky (15) na výpočet optimálnej dĺžky kroku. Najprv ale určíme potrebné vstupné parametre:

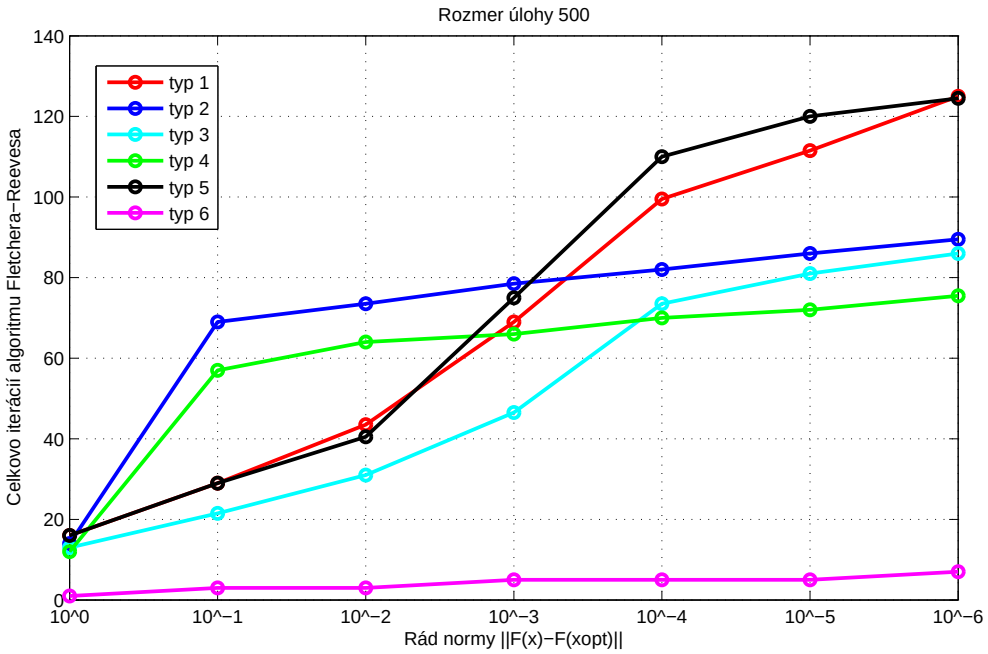
- rozmer úlohy: $n = \{50, 100, 200, 300, 500\}$
- číslo podmienenosti matice G: $\kappa(G) = 100000$
- vzdialenosť štartovacieho bodu x_0 od optimálneho riešenia $\hat{x}$: ρ=100
- najmenšie a najväčšie vlastné číslo matice D: $\lambda_{min} = 0.001$, $\lambda_{max} = 100$
- konštanta $\epsilon = 10^{-6}$
- rozsah rádov presnosti $\|f(x_k) - f(\hat{x})\|$, ktoré budeme skúmať: $\langle -6, 0 \rangle$

Na základe týchto údajov vytvoríme súbor 100 úloh s rovnakým typom rozloženia vlastných čísel a rovnakým rozmerom. Tento súbor budeme riešiť algoritmom Fletchera-Reevesa, pričom budeme zaznamenávať údaj o dosiahnutí rádu $\|f(x_k) - f(\hat{x})\|$ a čísle

zodpovedajúcej iterácie. Na základe zadaných vstupov získame 6 matíc, pretože skúmame 6 typov rozložení vlastných čísel matice G. Budú rozmeru 100x7 (100 úloh a 7 kontrolovaných rádov presnosti). Z týchto matíc vypočítame medián počtu iterácií algoritmu Fletchera-Reevesa potrebných na dosiahnutie určitého rádu v $\|f(x_k) - f(\hat{x})\|$ pre daný typ rozloženia vlastných čísel. Výstupom experimentu pre určitý rozmer bude teda 6 vektorov dĺžky 7.

Analýza výstupov

Zo šiestich vytvorených súborov úloh (takúto šesticu sme vytvorili pre všetky skúmané rozmery) sme pre každý súbor (ten obsahuje úlohy s určitým typom rozloženia vlastných čísel) dostali šesť vektorov s údajmi o mediáne počtu potrebných iterácií algoritmu Fletchera-Reevesa na dosiahnutie požadovanej presnosti $f(x_k)$. Pre lepšiu prezentáciu výsledkov sme urobili grafický výstup (strana 85). Na horizontálnej osi sa nachádza rád dosiahnutej presnosti v funkčnej hodnote $f(x_k)$. Na vertikálnej osi je celkový počet iterácií algoritmu FR potrebný na dosiahnutie prislúchajúcej presnosti. Ako je vidieť na Obr. 4, výsledky sú naozaj pozoruhodné.



Obr. 4: Vzťah rozloženia vlastných čísel a náročnosti na výpočet takej úlohy

Napríklad v prípade rozloženia vlastných čísel podľa typu 6 dokáže algoritmus naj-

lepšiu požadovanú presnosť (rádu 10^{-6}) dosiahnuť za približne 10 iterácií, čo je v porovnaní s rozmerom úlohy $n = 500$ excelentné. Dôvodom je podstata tvorby rozloženia typu 6. V tomto prípade má matica v podstate iba 2 vlastné hodnoty, keďže $\epsilon \ll \lambda_{min}$. Potom najväčšie vlastné číslo v skupine s λ_{min} má hodnotu $\lambda_{249} = \lambda_{min} + 249\epsilon = 0.001249$. Teda z globálneho hľadiska sa vlastné čísla v tejto skupine (podobne je tomu v druhej skupine) medzi sebou takmer nelíšia a tento fakt sa významne prejavil aj na náročnosti výpočtu daného typu úlohy.

Rozloženie podľa typu 1 a 5 zodpovedá tomu, čo sme očakávali. Teda ak sú vlastné čísla rovnomerne rozložené na intervale $\langle \lambda_{min}, \lambda_{max} \rangle$, takýto typ úloh je veľmi zložité riešiť, pretože celý priestor je rôzne ponaťahovaný. Z nameraných hodnôt by sa dala urobiť lineárna regresia, ktorá by mohla pomerne dobre popisovala vzťah medzi dosiahnutou prenosťou a potrebným počtom iterácií. Taktiež možno pozorovať, že neexistuje významný rozdiel v tom, či pri generovaní vlastné hodnoty usporiadame do monotónnej postupnosti alebo nie.

Typ 3 má graf veľmi podobný predchádzajúcim dvom typom. Vysvetľujeme si to tým, že tento typ má v podstate rovnakú stavbu ako predošlé dva typy, len s tým rozdielom, že typ 3 má až polovicu vlastných čísel rovnú λ_{max}, a preto k riešeniu konverguje rýchlejšie (v zmysle menšieho počtu potrebných iterácií).

Grafy pre typy 2 a 4 sú taktiež pozoruhodné. Hoci pri týchto typoch je algoritmus zo začiatku veľmi pomalý, po prekonaní rádu -1 dokonverguje do najvyššej požadovanej presnosti do 20 iterácií. Ide zhruba o tretinu tých, ktoré potreboval na dosiahnutie presnosti rádu -1. Menšia náročnosť výpočtu úlohy s rozložením vlastných čísel typu 4 oproti typu 2 je spôsobená tým, že typ 4 má až dve oblasti s rovnakým vlastným číslom.

Záver

Vykonaný experiment priniesol niekoľko dôležitých pozorovaní:

1. Experiment naznačuje, že spôsob rozloženia vlastných čísel medzi najmenšou vlastnou hodnotou λ_{min} a najväčšou vlastnou hodnotou λ_{max} má nezanedbateľný vplyv na náročnosť výpočtu generovanej úlohy. Preto pri generovaní úloh nestačí brať ohľad iba na vstupné číslo podmienenosti. Ak by sme tak urobili,

mohla by ľahko nastať situácia, kedy by jedna úloha vyžadovala dvoj/trojnásobok počtu iterácií druhej úlohy, pričom by obe boli generované s rovnakým číslom podmienenosti. To by značne znehodnocovalo štatistický výskum.

2. V ďalších experimentoch budeme pracovať s rozložením vlastných čísel ako v type 1. Rozhodli sme sa tak preto, že ide o najnáročnejší typ (čo sa týka dosiahnutia rádu určitej presnosti vo funkčnej hodnote), pričom počet potrebných iterácií rastie stabilne

3. Bez ohľadu na typ rozloženia vlastných čísel je zrejmé, že na dosiahnutie vysokej presnosti je v prípade riešenia kvadratickej úlohy potrebných nanajvýš n iterácií, kde n označuje rozmer úlohy. Pozorovanie je v súlade s prezentovanou teóriou (Veta 1.6).

4.2 Dynamizácia zlatého rezu

Cieľ experimentu

Cieľom tohto experimentu je preskúmať možnosť vylepšenia programu *Zlatyrez* (popis na strane 34) dynamickou úpravou jeho predpísaného počtu iterácií (ďalej iba dynamizácia). Jedným z jeho vstupných parametrov je konštanta w, ktorá udáva počet iterácií, ktoré má program realizovať. Zaujíma nás, či možno hovoriť o maximálnej hodnote konštanty w, ktorá je ešte efektívna (pojem efektívnosti vysvetlený v časti Postup experimentu). Ďalej sa núka možnosť preskúmania dynamickej zmeny hodnoty w za účelom dosiahnutia rovnakých výsledkov ako v prípade konštantnej hodnoty w, ale s menšou výpočtovou náročnosťou programu *Zlatyrez*.

Postup experimentu

Experiment vykonáme na úlohe s kvadratickou účelovou funkciou v tvare $f(x) = \frac{1}{2}x^T Gx + h^T x$. Hodnoty vstupných parametrov:

- rozmer úlohy: $n = 100$
- číslo podmienenosti matice G: $\kappa(G) = 10^3$

- typ rozloženia vlastných čísel v matici G: prvý typ (zadefinované v úvode podkapitoly 4.1)
- vzdialenosť štartovacieho bodu od optima: $\rho = 100$
- vstupný parameter w programu *Zlatyrez*:{9, 14, 19, 23, 28}

Používame malé číslo podmienenosti, pretože sa zameriavame na otestovanie možnej úpravy programu. V prípade preukázania jej opodstatnenosti môžeme zakomponovať aj väčšiu mieru výpočtových chýb v zmysle väčšieho čísla podmienenosti. w iterácií metódy zlatého rezu skráti interval dĺžky 1 na dĺžku $\left(\frac{\sqrt{5}-1}{2}\right)^w$, čo zodpovedá skráteniu na 10^{-r} dĺžky pôvodného intervalu, kde r označuje rád skrátenia. Hodnoty vstupného parametra w sme určili z vzťahu

$$\left(\frac{\sqrt{5}-1}{2}\right)^w = 10^{-r}, \qquad r = 2, \ldots, 6 \tag{65}$$

tak, aby sme pôvodný interval skrátili rádom $\{2, \ldots, 6\}$.

Experimentálnu úlohu budeme riešiť metódou Fletchera-Reevesa (program na strane 83) a program ukončíme po $n - 1$ mikroiteráciach. Výstupom programu bude textový súbor zachytávajúci priebeh riešenia úlohy. Tento súbor bude obsahovať číslo mikroiterácie, v ktorej aproximácia x_k dosiahla nový rád presnosti a dosiaľ vykonaný počet výpočtov (ozn. náročnosť) funkčnej hodnoty $\varphi(\alpha) = f(x + \alpha d)$ v programe *Zlatyrez*. Na záver budú uvedené rovnaké informácie pre poslednú mikroiteráciu. Úlohu vyriešime pre všetky vstupné hodnoty konštanty w. Bude nás zaujímať hodnota $\|x_{99} - \hat{x}\|$, príslušná hodnota náročnosti a aj celkový vývoj v náročnosti. Analýzou výstupných údajov sa pokúsime zodpovedať otázku existencie maximálnej efektívnej hodnoty w.

Poznámka: Za efektívnu hodnotu konštanty w budeme považovať hodnotu, ktorá dosahuje rovnaké alebo lepšie výsledky v $\|x_{99} - \hat{x}\|$ v porovnaní s ostatnými hodnotami, pričom bude potrebovať najmenej výpočtov funkčných hodnôt v programe *Zlatyrez*.

Druhou časťou experimentu bude pokus o dynamizáciu programu *Zlatyrez* s cieľom zníženia jeho výpočtovej náročnosti. Zaznamenáme si údaj o $\|x_{99} - \hat{x}\|$ pre $w = \omega_1$. Našu dynamizáciu navrhneme nasledovne

1. Začneme s $w = \omega_2, \quad \omega_2 < \omega_1$.

2. Po každých $\frac{n}{\omega_2 - \omega_1 + 1}$ pridáme jednu iteráciu programu *Zlatyrez* ($w = w + 1$). Na začiatku teda $w = \omega_2$ a na konci $w = \omega_1$.

Podobný postup zopakujeme pre začiatočnú hodnotu $w = \omega_3, \quad \omega_2 < \omega_3 < \omega_1$. Porovnaním získaných hodnôt $\|x_{99} - \hat{x}\|$ (aj náročnosti) pre hodnotu ω_1, dynamizáciu so začiatočnou hodnotou ω_2 a dynamizáciu so začiatočnou hodnotou ω_3 rozhodneme, či má náš návrh dynamizácie opodstatnenie. Cieľom dynamizácií s ω_2 a ω_3 je dosiahnuť rovnakú presnosť v $\|x_{99} - \hat{x}\|$, ako v prípade hodnoty ω_1, za približne rovnaký počet miktoiterácií metódy Fletchera-Reevesa, ale s menším celkovým počtom potrebných výpočtov funkčných hodnôt v programe *Zlatyrez*.

Analýza výstupov

V Tab. 8 prílohy C na strane 87 sa nachádzajú výstupy z experimentu pre všetky skúmané hodnoty konštanty w. Najlepší výsledok v $\|x_{99} - \hat{x}\|$ zaznamenala hodnota $w = 28$ s výsledkom rádu -5. Program *Zlatyrez* však vykonal až 1862 výpočtov funkčnej hodnoty.

Po prezretí ostatných výsledkov sme zistili, že hoci hodnoty $w = 14$ a $w = 19$ dosiahli iba rád rovný -4, sú veľmi blízko výsledku hodnoty $w = 28$ a potrebovali citeľne menej výpočtov funkčnej hodnoty. Zamerali sme sa preto na hodnoty $w = 14$ a $w = 19$.

Všimli sme si, že rády $\|x_k - \hat{x}\|$ boli dosahované, až na záver tabuľky, v rovnakých mikroiteráciách programu *F*letcher-Reeves. Hodnota $w = 14$ však potrebovala menej výpočtov funkčnej hodnoty (stĺpec náročnosť). Preto sa hodnota $w = 14$ javila ako efektívna v zmysle našej poznámky na strane 48. Jedinou pochybnosťou bola lepšia presnosť výsledku v prípade $w = 28$.

Uvažovali sme, či by sa mohol výsledok pre hodnotu $w = 14$ pri väčšom počte povolených mikroiterácií zlepšiť a odstrániť tak naše pochybnosti. Úplný výstup riešenej úlohy na stranách 88 a 89 poskytol dôležité informácie. Zamerali sme sa na údaje z posledných iterácií pre hodnoty $w = 14$ a $w = 28$. Posledné stĺpce vo výstupoch udávajú, ako sa vypočítaná hodnota účelovej funkcie líši od optimálnej hodnoty. Dosahované presnosti sú rádu -11, resp. -12. Pri takýchto rádoch začína programová realizácia metódy zlatého rezu zlyhávať pri omnoho skoršej iterácii v dôsledku nerozlíšiteľnosti

funkčných hodnôt dvoch susedných bodov (spomenuté v 3.2.1). To znamená, že hoci sme nastavili hodnoty $w = 14$ alebo $w = 28$, v skutočnosti program *Zlatyrez* v závere vykonal iba približne štyri iterácie. Preto nemožno očakávať už žiadne podstatné zlepšenie výsledku polohy pripustením vyššieho počtu mikroiterácií Fletchera-Reevesa.

Ako vidieť z Tab. 9 na strane 90, naše očakávania sa naplnili. Ide o výstup riešenia tej istej experimentálnej úlohy, pričom sme umožnili urobiť o jednu makroiteráciu navyše. Pre obe hodnoty w sa už ale výsledná hodnota $\|x_k - \hat{x}\|$ nijako významne nezlepšila. Podľa našej definície efektívnosti nemožno prehlásiť hodnotu $w = 14$ za maximálnu efektívnu hodnotu.

Napriek výsledkom prvej časti sme sa pokúsili o v úvode popisovanú dynamizáciu, pričom sme pracovali s hodnotou $\omega_1 = 28$. Otestovali sme dva postupy, pri ktorých začiatočná hodnota konštanty w bola $\omega_2 = 14$ a $\omega_3 = 19$. Výsledky sa nachádzajú v Tab. 10 na strane 91. Uvedené postupy nedosahujú takú presnosť ako v prípade voľby hodnoty $w = 28$. Z hľadiska náročnosti na výpočet funkčných hodnôt sa nachádzajú medzi voľbami hodnôt $w = 14$ a $w = 28$. Podarilo sa teda dosiahnuť nižšiu výpočtovú náročnosť, no keďže presnosť výsledku sa nezlepšila, stačilo by za daných okolností zvoliť hodnotu $w = 14$.

Záver

Cieľom prvej časti experimentu bolo preskúmať možnú existenciu maximálnej efektívnej hodnoty konštanty w. Hoci hodnota $w = 14$ nespĺňa naše kritériá efektívnosti, na základe analýzy podrobného výstupu riešenej úlohy má k ich splneniu z testovaných hodnôt najbližšie. Dôvodom je dosiahnutá vysoká presnosť v funkčnej hodnote a druhá najlepšia presnosť v polohe. Na jednej strane teda nemôžeme potvrdiť dosiahnutie cieľa prvej časti experimentu, na druhej strane ale takisto nemôžeme zamietnuť možnosť existencie efektívnej hodnoty.

V druhej časti experimentu sme sa snažili o dynamizáciu programu *Zlatyrez*. Na základe našich výsledkov môžeme zamietnuť nami skúmaný postup dynamizácie. Hoci dochádza k úspore počtu vykonaných výpočtov funkčných hodnôt, nedosahuje sa cieľová presnosť výsledku, a preto zvolený postup dynamizácie stráca zmysel. Naše výsledky sme skúmali iba na kvadratickej funkcii, no podobné výsledky očakávame aj

u iných typov funkcií. V ďalších experimentoch budeme v prípade použitia programu *Zlatyrez* pracovať s čo najmenšou hodnotou parametra w.

4.3 Experimentovanie s programom LSHZ

Cieľ experimentu

Cieľom tohto experimentu je preskúmať vplyv hodnôt konštánt δ a σ na výkon programu LSHZ (popísaný v 3.2.2). Tieto konštanty sú používané v približnom Wolfeho kritériu (21), ktoré je v programe LSHZ overované. Autori W.Hager a H. Zhang algoritmu LSHZ v článku [6] navrhli používať hodnoty $\delta = 0.1$ a $\sigma = 0.9$, s ktorými mala ich realizácia algoritmu najlepšie výsledky. My ich odporúčanie otestujeme na úlohách s bikvadratickou účelovou funkciou.

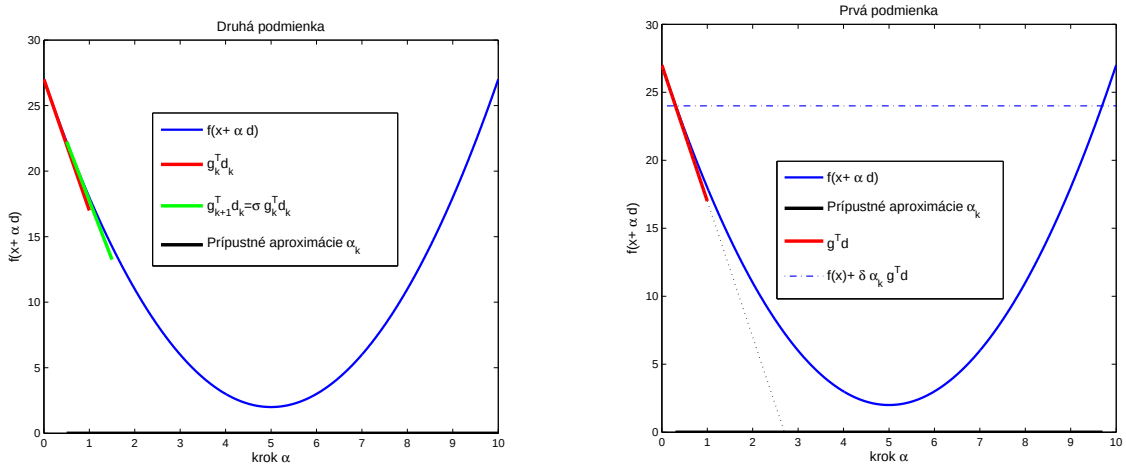
Pozadie experimentu

Približné Wolfeho kritérium bolo odvodené zo štandardného Wolfeho kritéria pozostávajúceho z podmienok (18) a (19) (odvodenie na strane 20). Tie majú tento tvar

$$f(x_k + \alpha_k d_k) - f(x_k) \leq \delta \alpha_k g_k^T d_k \qquad (18)$$

$$\sigma g_k^T d_k \leq g_{k+1}^T d_k \qquad (19)$$

kde d_k je spádový smer a vzťah δ a σ je určený nerovnicami $0 < \delta \leq \sigma < 1$.
Hodnota konštanty σ podmienky (19) má vplyv na vzdialenosť prípustnej aproximácie optimálneho kroku α_k od kroku $\alpha = 0$. Na Obr. 5 - *Druhá podmienka* môžeme vidieť, že zvolená hodnota konštanty σ v tomto prípade umožňuje za aproximáciu α_k vziať až hodnoty väčšie ako $\frac{1}{2}$. Malé hodnoty tejto konštanty zabraňujú akceptácii príliž krátkych aproximácií optimálneho kroku α_k. Na druhej strane, podmienka (19) sama o sebe umožňuje pridlhé aproximácie α_k. Tomuto bráni podmienka (18), ktorej úlohou je zabezpečiť primeraný pokles v funkčnej hodnote účelovej funkcie (Obr. 5 - *Prvá podmienka*). Voľba $\delta \to 0$ a $\sigma \to 1$ znamená, že vyžadujeme nižšiu presnosť aproximácie α_k. Z tohto pohľadu sa zdajú byť hodnoty $\delta = 0.1$ a $\sigma = 0.9$ rozumne zvolené. Zároveň môžeme z Obr. 5 vidieť, že na výslednú aproximáciu optimálneho kroku vplýva hodnota



Obr. 5: Grafické znázornenie podmienok štandardného Wolfeho kritéria.

konštanty δ viac než hodnota konštanty σ, pretože je reštriktívnejšia. Preto budeme v experimente skúmať iba vplyv konštanty δ. Malé hodnoty δ umožňujú realizovať iterácie MKG s relatívne malým poklesom v funkčnej hodnote. Naopak hodnoty blížiace sa $\delta = 0.5$ kladú dôraz na výrazný pokles v funkčnej hodnote. Z tohto dôvodu budeme v experimente sledovať veličinu $\|f(x_k) - f(\hat{x})\|$. Vplyv hodnoty δ by sa mal prejaviť na počte mikroiterácií MKG potrebných na dosiahnutie určitého rádu $\|f(x_k) - f(\hat{x})\|$ ako aj na výpočtovej náročnosti programu LSHZ (sledovanej cez počet výpočtov derivácií v tomto programe). Očakávame takýto scenár:

1. Ak zvolíme δ blízko 0 (ozn. δ_1), potom by program mohol spočiatku efektívnejšie vylepšovať presnosť $f(x_k)$ v porovnaní s voľbou δ blízko 0.5 (ozn. δ_2). Z hľadiska počtu potrebných mikroiterácií MKG môže byť situácia vyrovnaná, no voľba δ_1 by mala potrebovať menší počet výpočtov derivácií v programe LSHZ (oproti δ_2). Preto hovoríme o vyššej efektivite.

2. S rastúcou presnosťou $f(x_k)$ by sa prílišná benevolentnosť v aproximácii optimálneho kroku α_k v prípade δ_1 mala prejaviť v náraste počtu potrebných mikroiterácií MKG oproti δ_2. To by mohlo viesť k vyrovnaniu počtu potrebných výpočtov derivácií pre obe voľby δ alebo dokonca k lepším výsledkom δ_2.

Postup experimentu

Účinnosť rôznych hodnôt konštanty δ budeme porovnávať na základe výsledkov riešenia súboru 20 úloh s bikvadratickou účelovou funkciou. Úlohu budeme riešiť programom *Metóda Hagera-Zhanga* (v prílohe A.6).

Zvolíme tieto vstupné parametre

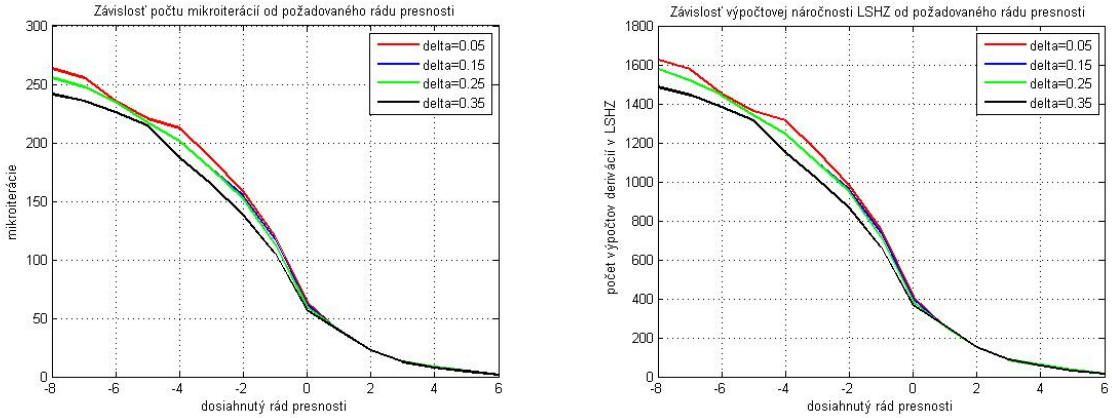
- rozmer úloh: $n = 100$
- číslo podmienenosti matíc G a D v bikvadratickej úlohe: $\kappa(G) = \kappa(D) = 10^5$
- vzdialenosť štartovacieho bodu od optima: $\|x_0 - \hat{x}\| = 100$
- skúmané hodnoty konštanty δ v LSHZ: $\{0.05, 0.15, 0.25, 0.35\}$
- hodnota konštanty σ v LSHZ: 0.9
- maximálny povolený počet makroiterácií: macro= 4

Počas riešenia úlohy budeme priebežne zaznamenávať dosiahnutý rád $\|f(x_k) - f(\hat{x})\|$ a k nemu príslušné číslo mikroiterácie a doposiaľ vykonaný počet výpočtov derivácií v programe LSHZ. Na základe vyriešenia skušobnej úlohy sme určili, že údaje budeme zapisovať pri dosiahnutí rádov presnosti $6, 5, \ldots, -7, -8$. Výsledky štatisticky spracujeme a vyhodnotíme.

Analýza výstupu

Výstup experimentu sa nachádza v prílohe C na strane 92. Obr. C.1 prezentuje výsledky pre hodnoty konštanty δ=$\{0.05, 0.15, 0.25, 0.35\}$.

Údaje v tabuľke sú mediánom počtu mikroiterácií a mediánom počtu vykonaných výpočtov derivácií v LSHZ z vyriešených 20 úloh pre každý z dosiahnutých rádov $\|f(x_k) - f(\hat{x})\|$. Dôvodom tohto štatistického spracovania je možná variabilita v čísle podmienenosti vygenerovaných 20 úloh. Určením čísla podmienenosti matíc G a D sme nezabezpečili rovnaké číslo podmienenosti v bode optima $\hat{x}$ pre všetky úlohy. Mimo tohto bodu navyše mení táto veličina svoju hodnotu. Graficky uvádzame výsledok experimentu na Obr. 6. Naše očakávania sa takmer naplnili. Z výstupu aj grafov vyplýva, že so zvyšujúcou sa presnosťou hodnoty účelovej funkcie, väčšia hodnota konštanty



Obr. 6: Grafická reprezentácia výstupu z experimentu z podkapitoly 4.3.

δ zlepšuje výkon programu LSHZ. Myslíme tým potrebu nižšieho počtu potrebných mikroiterácií ako aj menšiu výpočtovú náročnosť programu LSHZ.

To vidíme napríklad na vzájomnej polohe červenej a čiernej krivky. Vysvetľujeme si to tým, že menšie hodnoty δ umožňujú používať „horšie" aproximácie optimálneho kroku. Na jednej strane to znamená menej potrebných výpočtov derivácie v LSHZ, no na strane druhej to má za následok slabší pokles v funkčnej hodnote účelovej funkcie. Tento fakt sa prejaví na vyššom počte potrebných miktoiterácií. Táto situácia sa iba prehlbuje, čo vyúsťuje aj do vyššieho počtu výpočtov derivácií v LSHZ pre menšie hodnoty konštanty δ. Totožnosť výsledkov pre hodnoty $\delta = 0,15$ a $\delta = 0,25$ je dôsledkom použitia mediánových informácií.

Záver výstupu

Tento experiment sme realizovali na úlohách s bikvadratickou účelovou funkciou. Použili sme číslo podmienenosti 10^5. Naše záverečné zhrnutie sa vzťahuje na takýto typ úloh. Z rovnakých experimentov pre nižšie čísla podmienenosti sme zistili, že pre malé čísla podmienenosti (pod 1000) nie sú markantné rozdiely vo voľbe hodnoty konštanty δ vo vzťahu ku rádu $\|f(x_k) - f(\hat{x})\|$.

Na základe popísaného experimentu sme prišli k záveru, že v prípade bikvadratickej účelovej funkcie nie je hodnota $\delta = 0.1$ najoptimálnejšia. Väčšie hodnoty tejto konštanty sú schopné dosahovať rovnaké výsledky pri menšej výpočtovej náročnosti. Pre veľmi zle podmienené úlohy môže byť rozdiel vo výsledkoch menej výrazný ako je tomu v našom experimente.

5 Experimentálne porovnanie efektívnosti formúl MKG

V tejto kapitole budeme porovnávať efektívnosť metódy konjugovaných gradientov pri riešení minimalizačnej úlohy rôznych typov účelových funkcií, keď použijeme parameter β v tvare β^{HZ+}, β^{HS+}, β^{FR} a β^{DY}. Zavedieme nasledujúce zjednodušujúce označenie

- *metóda FR* - metóda konjugovaných gradientov s parametrom β v tvare β^{FR}
- *metóda DY* - metóda konjugovaných gradientov s parametrom β v tvare β^{DY}
- *metóda HS* - metóda konjugovaných gradientov s parametrom β v tvare β^{HS+}
- *metóda HZ* - metóda konjugovaných gradientov s parametrom β v tvare β^{HZ+}

Na začiatku experimentovania sme sa museli rozhodnúť, ktorý z programov *Zlatyrez* alebo *LSHZ* použiť na nájdenie optimálneho kroku α_k. Na základe niekoľkých pokusov sme sa rozhodli pre program *LSHZ*. Naše rozhodnutie zdôvodňujeme nasledovne: určitá presnosť v funkčnej hodnote optimálneho riešenia úlohy sa dosahuje skôr ako tá istá presnosť vyžadovaná od aproximácie optima $\hat{x}$. Podobne je tomu v prípade gradientnej informácie ($\|g_k\|$). V našich experimentoch plánujeme efektívnosť metód porovnávať práve na základe údajov z $\|x_k - \hat{x}\|$ a $\|g_k\|$. Metóda zlatého rezu je postavená na porovnávaní funkčných hodnôt v dvoch susedných bodoch. Program *Zlatyrez* nie je zo svojej podstaty schopný dosahovať vysoké presnosti v spomínaných ukazovateľoch, keďže sa funkčné hodnoty stávajú nerozlíšiteľné. Na druhej strane, program *LSHZ*, ktorý používa derivácie, dokáže naše požiadavky splniť.

V programových realizáciách uvedených metód teda použijeme na nájdenie optimálneho kroku α_k program *LSHZ*, ktorý využíva približné Wolfeho kritérium (21). Pre *metódu FR* platí, že v prípade použitia iba približného optimálneho kroku je potrebné namiesto štandardného alebo približného Wolfeho kritéria použiť silné Wolfeho kritériu (20) s hodnotou konštanty $\sigma < \frac{1}{2}$ (podľa [7]). Našim cieľom je preskúmať efektívnosť použitých tvarov parametra β, preto budeme od programu *LSHZ* požadovať rovnakú presnosť aproximácie optimálneho kroku pre všetky programové realizácie. Z už uvedených dôvodov budeme v programe *LSHZ* používať hodnotu $\sigma = 0.4$. Hodnotu konštanty δ , ktorú program *LSHZ* taktiež používa, sme určili na 0,3. Vychádzali sme

z výsledkov experimentu v podkapitole 4.3 v prípade bikvadratickej účelovej funkcie. Pre iné typy účelových funkcií nebudeme túto hodnotu meniť, keďže z hľadiska porovnania efektívnosti metód nemá na výsledky vplyv.

V prípade programovej realizácie *metódy HZ* sme sa museli rozhodnúť, akú hodnotu parametra θ použiť v predpise (41) paramatra β^{HZ+}. Autori W.Hager a H.Zhang používali vo svojich experimentoch hodnotu $\theta = 2$. Pri našom experimentovaní sme zistili, že hodnota $\theta = 0,5$ dosahovala lepšie výsledky, preto sme sa rozhodli pre túto hodnotu.

Efektívnosť skúmaných metód budeme porovnávať na základe počtu mikroiterácii MKG potrebných na splnenie cieľových podmienok. Vo všeobecnosti budeme robiť tieto experimenty:

Prvý typ Zameriame sa na skúmanie výsledkov v $\|x_k - \hat{x}\|$. Zadáme rozmer úlohy a počas jej riešenia budeme zaznamenávať číslo mikroiterácie, v ktorej prišlo k vylepšeniu rádu $\|x_k - \hat{x}\|$. Riešenie úlohy zastavíme, ak nastanú situácie

1. $\|x_k - \hat{x}\| \leq 10^{-8}$ alebo
2. $\|g_k\| \leq 10^{-7}$.

Následne preskúmame získané údaje o počtoch mikroiterácií.

Druhý typ Zameriame sa na informácie z $\|g_k\|$, ktoré sú pri riešení reálnych úloh rozhodujúce. Pri vylepšení rádu hodnoty tohto ukazovateľa si zaznamenáme zodpovedajúce číslo mikroiterácie a aj čas[4], ktorý ubehol od začiatku riešenia úlohy. Získané časové údaje použijeme na zostrojenie grafu Dolanovej-Morého benchmarku. Údaje o počte mikroiterácií použijeme ako doplňujúce informácie tohto grafu.

Benchmark autorov E. Dolanovej a J. Morého je popísaný v článku [4]. Proces jeho tvorby vysvetlíme na jednoduchom príklade. Uvažujme súbor n úloh a p programov, ktoré budú tento súbor riešiť. Pre každý program zaznamenáme čas, ktorý potreboval na vyriešenie každej zo 100 úloh. Získame tak $n.p$ údajov. Vrámci každej úlohy zistíme najlepší čas, ktorý dosiahol niektorý z p programov. Týmto časom predelíme

[4] Využijeme štandardné funkcie programu Matlab- tic a toc. Tieto funkcie merajú celkový uplynutý čas medzi ich volaniami

všetky časy pre danú úlohu. Získame tak hodnoty od 1 vyššie, pričom hodnotu 1 bude mať najrýchlejší program, vyššie hodnoty budú mať ostatné programy. Hodnoty interpretujeme ako násobky najlepšieho času. Následne pristúpime k tvorbe samotného grafu. Vertikálna os bude predstavovať percentuálnu časť zo súboru generovaných úloh. Horizontálna os bude predstavovať násobky najlepšieho času (časovú jednotku označíme písmenom τ). Pre každý z p programov získame graf, ktorého body reprezentujú percento zo súboru úloh, ktoré daný program vyriešil za násobok najlepšieho času. Napríklad body pre $\tau = 1$ hovoria, v koľkých percentách úloh boli dané programy najrýchlejšie. Graf, ktorý sa bude náchádzať najvyššie, reprezentuje najefektívnejší program, pretože tento program dokázal spomedzi všetkých programov za určitý násobok najlepšieho času vyriešiť najväčšie percento generovaných úloh.

Pre experimenty sme si zvolili tri rôzne konvexné účelové funkcie. Jednou z nich je bikvadratická funkcia, ktorej sa budeme venovať v prvej podkapitole. Zvyšné dve funkcie sme vybrali z článku [2]. Článok obsahuje predpisy štandardných testovacích funkcií na hľadanie voľného extrému.

5.1 Bikvadratická účelová funkcia

Tento experiment vykonáme na minimalizačných úlohách s bikvadratickou účelovou funkciou v tvare

$$f(x) = \frac{1}{4}(x^T D x)^2 + \frac{1}{2}x^T G x + h^T x \tag{66}$$

kde D je kladne definitná diagonálna matica, G je kladne definitná symetrická matica a h je vektor z $\mathbb{R}^n$. Úlohy vygenerujeme programom *Generátor bikvadratických úloh* uvedeným na strane 76. Štartovací bod x_0 vždy vygenerujeme tak, aby spĺňal $\|x_0 - \hat{x}\| = 100$.

Experiment 1a

Ide o experiment prvého typu.

- rozmer úlohy: $n = 100$
- číslo podmienenosti matíc D a G: $\kappa(D) = \kappa(G) = 1,5\ 10^6$

- počet vygenerovaných úloh: 101
- maximálny povolený počet makroiterácií: $macro = 14$
- sledované rády v norme $\|x_k - \hat{x}\|$: {0, -2, -4}

V prípade úloh s bikvadratickou účelovou funkciou sme neboli schopní zabezpečiť rovnaké číslo podmienenosti pre všetky úlohy. Preto sme pristúpili ku riešeniu súboru úloh, pričom sme zaznamenali údaje o každej úlohe a výsledky nakoniec štatisticky spracovali. Výstup z tohto experimentu sa nachádza v Tab. 11 na strane 93. Pre pohodlie čitateľa v texte úvadzame niektoré jeho časti. Prvé, čo si môžeme všimnúť, je

rozmer úloh=100	Číslo podmienenosti v optime úlohy min 2.56e+006	priemer 2.91e+006	max 3.15e+006	
Dosiahnutie rádu 0 v $\|x_k - \hat{x}\|$	priemer	medián	min	max
HS	71	31	7	683
FR	93	43	7	440
HZ	70	27	6	490
DY	82	40	7	388

Tabuľka 1: Prvá časť Tab. 11.

spomínaná variabilita v čísle podmienenosti v optime generovaných úloh. Vyplýva to z konštrukcie nášho generátora bikvadratických úloh, kde zadávame čísla podmienenosti matíc D a G a nie číslo podmienenosti v optime úlohy.

V tabuľke sa nachádzajú získané údaje o počte potrebných mikroiterácií MKG na dosiahnutie rádu 0 v $\|x_k - \hat{x}\|$. Údaje o priemernom počte potrebných mikroiterácií sú porovnateľné, najlepšie výsledky dosahujú *metódy HS* a *HZ* s počtom mikroiterácií 71, resp. 70. Najhorší výsledok dosahuje *metóda FR* s priemerom 93 mikroiterácií. Stĺpec medián poskytuje veľmi dôležité údaje. Najlepší výsledok naďalej dosahuje *metóda HZ* s 27 mikroiteráciami, najhorší stále *metóda FR* s 43 mikroiteráciami. Hodnoty v stĺpci medián sú však približne polovičné oproti hodnotám v stĺpci priemer. Na jednej strane teda napríklad v prípade *metódy HZ* stačilo v polovici úloh najviac 27 iterácií, no v priemere to bolo až 70. To naznačuje veľký rozptyl v nameranom počte potrebných mikroiterácií v jednotlivých úlohách. Tento postreh potvrdzujú hodnoty v stĺpcoch min a max, ktoré obsahujú najmenší a najväčší počet mikroiterácií potrebných na dosiahnutie požadovaného rádu. V ukazovateľovi max je najlepšia *metóda DY* s počtom

mikroiterácií 388. *Metóda HZ* potrebovala v niektorej úlohe až 490 mikroiterácií. Išlo však o ojedinelý prípad, keďže metóda dosiahla najlepšie výsledky v ukazovateľoch priemer a medián.

V prostrednej časti Tab. 11 na strane 93 sa nachádzajú získané údaje o počte potrebných mikroiterácií MKG na dosiahnutie rádu -2 v $\|x_k - \hat{x}\|$. V ukazovateľovi priemer je najlepšia *metóda HZ* s priemerným počtom mikroiterácií 195 (nasleduje *metóda DY* s výsledkom horším iba o 2 mikroiterácie). V ukazovateľovi medián sa poradie týchto dvoch metód nemení. Pozorujeme teda zmenu v efektivite meranej počtom potrebných mikroiterácií, kedy naďalej vedie *metóda HZ*, no druhou najlepšou sa javí *metóda DY*. Vidíme tiež, že sa v súbore vyskytla úloha, s ktorou mala *metóda HZ* problémy, pretože potrebovala až 691 mikroiterácií. Na druhej strane ale *metóda HZ* vyriešila jednu z úloh iba za 55 mikroiterácií, čo je najlepší výsledok zo sledovaných metód.

V poslednej časti Tab. 11 sa nachádzajú získané údaje o počte potrebných mikroiterácií MKG na dosiahnutie rádu -4 v $\|x_k - \hat{x}\|$.

Dosiahnutie rádu -4 v $\|x_k - \hat{x}\|$	priemer	medián	min	max
HS	275	246	85	983
FR	288	253	124	880
HZ	264	248	84	797
DY	262	245	121	698

Tabuľka 2: Posledná časť Tab. 11.

V ukazovateľovi priemer sú stále najlepšie *metódy HZ* a *DY*. V ukazovateľovi medián sa k nim pridáva *metóda HS*. V ukazovateľoch min a max je porovnateľná *metóda HS* s *HZ* a *metóda FR* s *DY*. Z týchto dvojíc sú najlepšie *metódy HZ* a *DY*, pretože majú najnižšie hodnoty maximálneho počtu potrebných mikroiterácií.

V tomto experimente vynikala *metóda HZ* s najnižšími hodnotami v ukazovateľoch priemer a medián a s uspokojivými hodnotami v ukazovateľoch min a max. So zvyšujúcou sa požadovanou konečnou presnosťou optimálneho riešenia sa porovnateľnou metódou stala *metóda DY*.

Experiment 1b

Ide o experiment prvého typu. Zaujímalo nás, či sa závery experimentu 1a zmenia, ak zvýšime rozmer úlohy. Rozmer sme preto zvýšili na trojnásobok. Taktiež sme povolili nižší maximálny počet makroiterácií z výpočtových dôvodov. Inak sa vstupné údaje do experimentu nezmenili.

- rozmer úlohy: $n = 300$
- číslo podmienenosti matíc D a G: $\kappa(D) = \kappa(G) = 1,5\ 10^6$
- počet vygenerovaných úloh: 101
- maximálny povolený počet makroiterácií: $macro = 6$
- sledované rády v norme $\|x_k - \hat{x}\|$: {0,-2,-4}

Výstup z tohto experimentu sa nachádza v Tab. 12 na strane 94.

Z výsledkov pre dosiahnutie rádu 0 v $\|x_k - \hat{x}\|$ vidíme, že sa situácia oproti rozmeru 100 takmer nezmenila. V ukazovateľoch priemer a medián sú *metódy HS* a *HZ* identické. Ukazovateľ max naznačuje, že sa v súbore nachádzala úloha, s ktorou mala *metóda HZ* problémy, pretože potrebovala až 431 mikroiterácií (ostatné metódy majú nižšie hodnoty v stĺpci max). *Metóda FR* je opäť najhoršia.

V prostrednej časti Tab. 12 si všimnime najmä stĺpec max. *Metódy HS* a *HZ* tam nadobúdajú hodnoty nad 1600. Vrámci 6 povolených makroiterácií sa už v týchto úlohách výrazne priblížili ku svojej hranici. *Metóda HZ* dosiahla najlepšie výsledky v ukazovateľoch medián a priemer. Je to pozoruhodný výsledok napriek spomínaným okolnostiam.

Výsledky pre rád -4 ukazujú zaujímavý poznatok. *Metóda DY* si poradila s každou z úloh zo súboru za maximálne 1476 mikroiterácií. Ak sa však pozrieme na *metódu HZ*, tá dosiahla najlepší výsledok v ukazovateli medián. Taktiež dosiahla skoro najlepší výsledok v ukazovateli min. Oproti *metóde DY* ide o rozdiel až 33 mikroiterácií. Priemerný výsledok má oproti tejto metóde o 2 mikroiterácie lepší. V kontexte s faktom, že *metóda HZ* potrebovala v niektorej úlohe až 2039 mikroiterácií, to naznačuje, že v riešení ostatných úloh musela byť *metóda HZ* efektívnejšia v porovnaní s *metódou DY*.

Pri číslach podmienenosti v optime riešenej úlohy približne 10^6 sa zdá byť *metóda HZ* najefektívnejšia. Pre nižšie požadované rády hodnôt $\|x_k - \hat{x}\|$ dosahuje podobné výsledky *metóda HS*. S rástúcou presnosťou začína *metóda HS* zaostávať a je nahradená *metódou DY*.

Experiment 1c

Ide o experiment prvého typu. Doterajšie experimenty sme urobili s hodnotou čísla podmienenosti matíc D a G 1,5 10^6. Experiment zopakujeme pre väčšie číslo podmienenosti.

- rozmer úlohy: $n = 100$
- číslo podmienenosti matíc D a G: $\kappa(D) = \kappa(G)$ =1,5 10^9
- počet vygenerovaných úloh: 101
- maximálny povolený počet makroiterácií: $macro = 14$
- sledované rády v norme $\|x_k - \hat{x}\|$: {0,-2,-4}

Výstup sa nachádza v Tab. 13 na strane 94.

V prvých dvoch častiach Tab. 13 (rád 0 a rád -2) dosahuje najlepšie výsledky *metóda HS*. *Metóda HZ* dosahuje rovnaké alebo porovnateľné výsledky. Chceme tiež opätovne upozorniť na hodnoty v stĺpcoch priemer a medián, ktoré sa výrazne odlišujú. Dôvodom je variabilita v čísle podmienenosti generovaných úloh.

Z výsledkov dosiahnutia najvyššej skúmanej presnosti (rád -4 v $\|x_k - \hat{x}\|$) usudzujeme, že si *metóda HZ* počínala rovnako dobre ako *metóda HS*. Svedčia o tom podobné výsledky vo všetkých ukazovateľoch.

Záver experimentu 1

Z výsledkov dosiaľ vykonaných experimentov môžeme povedať, že z hľadiska počtu mikroiterácií potrebných na dosiahnutie cieľovej presnosti je *metóda HZ* zo skúmaných metód najefektívnejšia. Hoci sa vyskytovali úlohy, v ktorých si táto metóda počínala citeľne horšie oproti konkurencii, z hľadiska celého súboru úloh dokázala väčšinu úloh

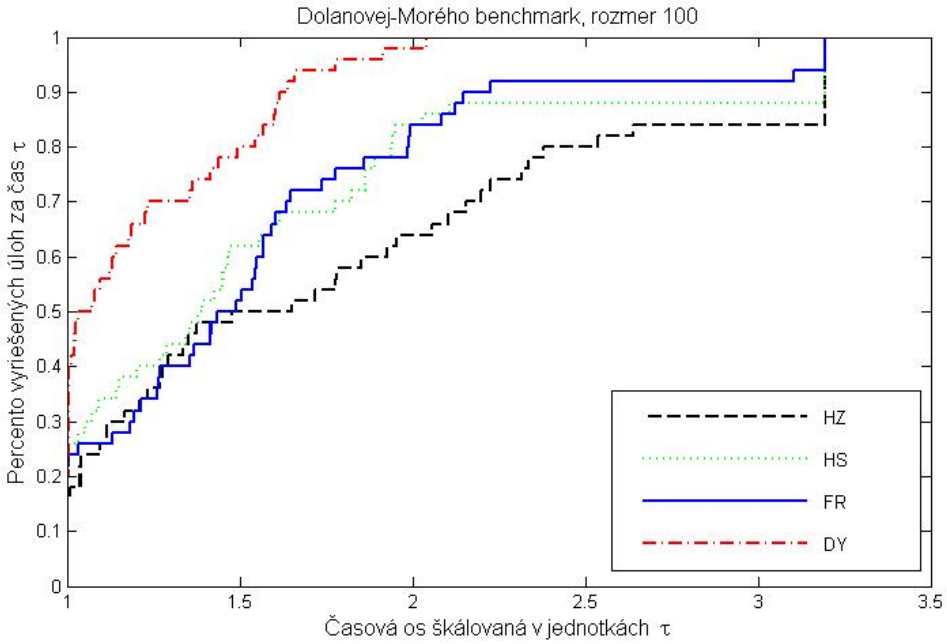
vyriešiť najrýchlejšie. Za naopak najmenej efektívnu metódu považujeme *metódu FR*, ktorá dosahovala vo väčšine ukazovateľov až na pár výnimiek najhoršie výsledky.

Experiment 2

Ide o experiment druhého typu. V experimente sme sa zamerali na gradientné stopovacie kritérium. Riešenie úlohy sme zastavili, ak $\|g_k\| \leq 10^{-2}$. Zaznamenávali sme ubehnutý čas a výsledky spracovali do Dolanovej-Morého benchmarku.

- rozmer úlohy: $n = 100$
- číslo podmienenosti matíc D a G: $\kappa(D) = \kappa(G) = 1,5\ 10^6$
- počet vygenerovaných úloh: 50
- maximálny povolený počet makroiterácií: $macro = 14$
- stopovacie kritérium: $\|g_k\| \leq 10^{-2}$

Výsledkom tohto experimentu je graf na Obr. 7. Graf naznačuje, že *metóda DY* bola



Obr. 7: Porovnanie efektívnosti skúmaných metód na úlohách s bikvadratickou účelovou funkciou (rozmer 100)

zo skúmaných metód najefektívnejšia. Najlepší výsledok dosiahla v približne 45 percentách úloh, pričom úlohy v generovanom súbore dokázala vyriešiť za maximálne

dvojnásobok najlepšieho času. *Metóda HS* dosiahla najlepší výsledok v 16 percentách úloh. Z tvaru grafu tejto metódy je zrejmé, že sa v súbere nachádzali úlohy, ktoré metóda za povolený počet makroiterácií nedokázala vyriešiť. Z toho dôvodu dosahuje 100 percentnú úspešnosť až pri hodnote $\tau = 3,25$. *Metódy FR* a *HS* sú výsledkovo porovnateľné. V niektorých hodnotách τ je lepšia *metóda HS*, v iných zasa *metóda FR*. Všimnime si tiež, že do $\tau = 2,25$ majú krivky prudko rastúci tvar. Znamená to, že výsledky metód boli veľmi podobné. V generovanom súbore sa ale nachádzalo 10 percent úloh, s ktorými mali *metódy HS* a *FR* problémy, čo sa prejavilo na tvare ich kriviek. Keď sme preskúmali údaje o počte vykonaných mikroiterácií, situácia sa vyjasnila. V súbore boli úlohy, na ktoré potrebovali *metódy HS*, *HZ* alebo *FR* mimoriadne veľa mikroiterácií, čo sa následne prejavilo na celkom čase riešenia úlohy. Interpretácia Obr. 7 teda nespočíva úplne vo výpočtovej náročnosti jednotlivých algebraických formúl parametra β. Výsledky možno zhrnúť tak, že *metóda DY* vyriešila vygenerovaný súbor úloh najkonzistentnejšie. To znamená, že počty potrebných mikroiterácií MKG boli v jednotlivých úlohách približne rovnaké. *Metódy HZ*, *HS* a *FR* boli v riešení úloh tiež úspešné, no ich výsledky boli menej vyrovnané. Inak povedané, niektoré úlohy vyriešili bleskovo, v iných potrebovali neprimerane viac mikroiterácií, ako ich konkurencia.

Podobné výsledky sme dostali z rovnakého experimentu, v ktorom sme ale zvýšili rozmer úloh na 300. Graf sa nachádza na strane 86 (Obr. B.2). Najlepšou sa ukázala *metóda DY*. Posunula sa ale hodnota τ, pri ktorej *metóda DY* vyriešila aj poslednú úlohu.

Záver experimentu 2

Výsledky experimentu 2 na prvý pohľad nekorešpondujú s výsledkami experimentu 1. Z posledných komentovaných výsledkov totiž vyplýva, že najefektívnejšou metódou je *metóda DY*. Táto metóda je v prípade bikvadratickej účelovej funkcie naozaj najefektívnejšia, ak požadujeme nielen vysokú presnosť funkčnej hodnoty, ale aj vysokú presnosť bodu x_k, ktoré toto riešenie vytvára. V experimente 1 sme sa zameriavali na ukazovateľ $\|x_k - \hat{x}\|$, pričom posledným zaznamenávaným rádom bola hodnota -4. V experimente 2 sme použili omnoho silnejšie ukončovacie kritérium, keď sme sledovali ukazovateľ $\|g_k\|$. Po preskúmaní výstupov sme zistili nasledovné:

- na začiatku riešenia úlohy: $\|x_0 - \hat{x}\| = 100$ a $\|g_0\| = 10^{11}$
- v experimente 1 sme riešenie zastavili, ak: $\|x_k - \hat{x}\| \leq 10^{-3}$, čo zodpovedalo $\|g_k\| = 10^4$
- v experimente 2 sme riešenie zastavili, ak: $\|g_k\| \leq 10^{-2}$, čo zodpovedalo $\|x_k - \hat{x}\| \leq 10^{-6}$

V experimente 2 sme teda požadovali okrem presného riešenia aj veľmi presnú aproximáciu bodu, ktoré toto riešenie vytvára. Už v experimente 1 bolo možné si všimnúť, ako sa *metóda DY* postupne zlepšuje a vyrovnáva sa *metóde HZ*.

Chceme ale ešte raz zdôrazniť, že sme v experimentoch skúmali iba efektívnosť rôznych formúl parametra β MKG. Z toho dôvodu sme kládli rovnaké podmienky na program *LSHZ* v prípade každej zo skúmaných metód. Podmienky sme stanovili veľmi prísne, aby bol program *LSHZ* použiteľný aj v prípade *metódy FR*. Ak by sme skúmali efektívnosť programovej realizácie metód ako celku, bolo by možné pre každú z metód upraviť hodnoty konštánt vstupujúcich do programu *LSHZ*. Získané výsledky by sa mohli úplne líšiť od tých, ktoré tu prezentujeme.

5.2 Diagonálna funkcia 2

Táto funkcia má predpis

$$f(x) = \sum_{i=1}^{n} \left(\exp(x_i) - \frac{x_i}{i} \right) \tag{67}$$

kde $x \in \mathbb{R}^n$ a štandardným štartovacím bodom je $x_0 = (1, \frac{1}{2}, \ldots, \frac{1}{n})^T$. Optimálnym bodom $\hat{x}$ je bod $(0, \ldots, \ln\left(\frac{1}{i}\right), \ldots, \ln\left(\frac{1}{n}\right))^T$. Skúmané metódy budeme testovať na dvoch typoch experimentov popísaných na začiatku kapitoly 5.

Ak nespomenieme inak, skúmaným metódam neumožníme žiadnu makroiteráciu. To znamená, že sa riešenie úlohy ukončí, ak program dosiahne $n - 1$ mikroiteráciu (za predpokladu, že ešte skôr nenastanú ukončovacie podmienky uvedené v popise experimentov).

Experiment 3

Ide o experiment prvého typu. Zvolili sme rozmer úlohy 10 000. Vzdialenosť štartovacieho bodu x_0 od bodu $\hat{x}$ je $\rho = 827$. Výstupné údaje sa nachádzajú v Tab. 3.

Rozmer úlohy = 10 000

rád hodnoty $\|x_k - \hat{x}\|$ ($10^{\text{rád}}$)	HZ	HS	FR	DY
0	187	155	1296	1054
-1	371	389	2386	1932
-2	461	499	3503	2823
-3	584	700	4628	3716
-4	805	934	5753	4598
-5			6879	5488
podmienka $\|g_k\| \leq 10^{-7}$ splnená v	917	1024	7542	5889

Tabuľka 3: Diagonálna 2 funkcia- počet mikroiterácií MKG vo vzťahu ku cieľovému rádu $\|x_k - \hat{x}\|$.

Výsledky tohto experimentu sú prekvapujúce. Vidíme, že *metódy HZ* a *HS* dosahujú podmienku $\|g_k\| \leq 10^{-7}$ zhruba okolo tisícej mikroiterácie, čo je iba jedna desatina z celkového rozmeru úlohy. Priebeh riešenia majú veľmi podobný, v závere je *metóda HZ* rýchlejšia (v zmysle menšieho počtu potrebných mikroiterácií). Úplným protikladom sú výsledky *metód DY* a *FR*. Ich počty mikroiterácií sú v tisícoch. Ukončovaciu podmienku dosahujú v 5889 (*metóda DY*) a 7542 mikroiterácii (*metóda FR*). Ide teda o výrazne odlišné výsledky riešenia tej istej úlohy pri rovnakej prípustnej presnosti aproximácie optimálneho kroku α_k. Test sme preto zopakovali s tým, že sme si nechali vypisovať informácie z každej iterácie, najmä hodnoty $\|x_k - \hat{x}\|$, $\|g_k\|$ a $\|f(x_k) - f(\hat{x})\|$. Výsledky uvedené v Tab. 3 sme potvrdili. Ukázalo sa, že *metódy DY* a *FR* robia veľmi malý pokrok v $\|x_k - \hat{x}\|$, čo sa následne prejavuje na $\|g_k\|$ a $\|f(x_k) - f(\hat{x})\|$. Takúto situáciu sme popísali v kapitole 2 v Poznámke na strane 25. Ide o tzv. „zasekávanie", voči ktorému sú *metódy HZ* a *HS* imúnne, čo sa prejavilo aj na ich odlišných výsledkoch.

Experiment 4

Ide o experiment druhého typu, kde sme zvolili rovnaký rozmer úlohy 10 000 ako v prípade experimentu 3. Vzhľadom na výsledky tohto experimentu už vieme, že údaje, ktoré získame, budú ovplyvnené tzv. „zasekávaním" *metód DY* a *FR*. Preto pôjde iba o overenie výsledkov a nie o porovnanie výpočtovej náročnosti, ako sme pôvodne plá-

novali. Z tohto dôvodu uverejníme iba údaje o počte mikroiterácií (skúmanie časových údajov nemá v tomto prípade zmysel). Výstupné údaje sa nachádzajú v Tab. 4.

Rozmer úlohy = 10000

rád hodnoty $\|g_k\|$ ($10^{\text{rád}}$)	HZ	HS	FR	DY
0	1	1	1	1
-1	11	9	7	6
-2	48	43	798	612
-3	128	135	1915	1439
-4	311	239	3043	2331
-5	407	445	4149	3203
-6	545	616	5293	4089
-7	692	807	6399	4989

Tabuľka 4: Diagonálna 2 funkcia- počet mikroiterácií MKG vo vzťahu ku cieľovému rádu $\|g_k\|$.

Výsledky sú presne také, aké sme očakávali. Úplne korešpondujú s výsledkami experimentu 3. Najlepšie sú *metódy HZ* a *HS*, no pre dosiahnutie finálnej presnosti je najlepšia *metóda HZ*. Taktiež si môžeme všimnúť, že dosiahnutá presnosť v $\|g_k\|$ korešponduje s rádom $\|x_k - \hat{x}\|$ medzi -3 a -4 v prípade *metód HZ* a *HS*. Z tohto dôvodu Tab. 3 neobsahuje údaje o ráde -5 v $\|x_k - \hat{x}\|$ pre tieto metódy (metódy skôr dosiahnu gradientné ukončovacie kritérium).

5.3 Hagerova funkcia

Táto funkcia má predpis

$$f(x) = \sum_{i=1}^{n} \left(\exp(x_i) - \sqrt{i} x_i \right) \tag{68}$$

kde $x \in \mathbb{R}^n$ a štandardným štartovacím bodom je jednotkový vektor $x_0 = (1, \ldots, 1)^T$. Funkcia nadobúda svoje minimum v bode $\hat{x} = \frac{1}{2}(0, \ldots, \ln(i), \ldots, \ln(n))^T$.

Experiment 5

Ide o experiment prvého typu, v ktorom sme zvolili rozmer úlohy 10 000. Počiatočná vzdialenosť ρ bodov x_s a $\hat{x}$ má hodnotu 315. Výstupné údaje sa nachádzajú v Tab. 5.

Rozmer úlohy = 10000

rád hodnoty $\|x_k - \hat{x}\|$ ($10^{\text{rád}}$)	HZ	HS	FR	DY
0	5	5	7	7
-1	12	12	18	17
-2	23	24	42	37
-3	35	35	63	55
-4	46	46	83	72
-5	58	58	105	90
-6	69	69	124	107
-7	79	79	145	124

Tabuľka 5: Hagerova funkcia- počet mikroiterácií MKG vo vzťahu ku cieľovému rádu $\|x_k - \hat{x}\|$.

Prvé, čo si môžeme vo výstupe všimnúť, je malý počet mikroiterácií MKG potrebných na dosiahnutie požadovanej presnosti v $\|x_k - \hat{x}\|$. Vzhľadom na rozmer úlohy ide o zarážajúce hodnoty. Uvažovali sme teda, čo mohlo spôsobiť takýto rýchly výpočet. Prišli sme k týmto záverom

1. Štartovací bod x_0 sa nachádza pomerne blízko bodu $\hat{x}$: $\|x_0 - \hat{x}\| = 3,14 \; . \; 10^2$ a $\|x_0 - \hat{x}\|_\infty = 3,6$.

2. Hessova matica funkcie (68) má tvar diagonálnej matice s diagonálou $(\exp(x_1), \ldots, \exp(x_n))^T$. Číslo podmienenosti tejto matice má v bode $\hat{x}$ hodnotu 100, v bode x_0 1. Ide teda o mimoriadne dobre podmienenú úlohu.

Tieto dôvody vysvetľujú rýchlosť vyriešenia úlohy.

Ďalšou zaujímavosťou sú jednotlivé výsledky metód. *Metódy HZ* a *HS* sú najrýchlejšie s počtom mikroiterácií 79 pre dosiahnutie poslednej sledovanej presnosti. *Metódy FR* a *DY* potrebovali nezanedbateľne viac mikroiterácií. Z týchto dvoch metód je *metóda FR* najhoršia, pretože potrebovala až 145 mikroiterácií, čo je takmer dvojnásobok potrebných mikroiterácií *metódy HZ* alebo *HS*.

Nakoniec si všimnime, že *metódy HZ* a *HS* dosahujú takmer totožné výsledky. Znamená to, že rozšírenie parametra β^{HZ} o výraz $\frac{||y_k||^2 g_{k+1}^T d_k}{(d_k^T y_k)^2}$ nemalo takmer žiadny prínos.

Preto v skutočnosti $\beta^{HZ} = \beta^{HS}$. Už v kapitole 2 sme uviedli, že táto situácia nástava napríklad v prípade „zasekávania"(Poznámka na strane 25). Zároveň to vysvetľuje výrazne horšie výsledky *metódy FR* a *DY*.

Experiment 6

Výsledky experimentu 5 nás inšpirovali k overeniu výpočtovej náročnosti skúmaných metód. Budeme však testovať iba *metódy HZ* a *HS*, keďže zvyšné dve metódy podliehali „zasekávaniu", a preto by ich výsledky nemali žiadnu relevantnú informačnú hodnotu. Naopak *metódy HZ* a *HS* dosahovali rovnaké výsledky, preto by časové údaje o náročnosti výpočtov mohli priniesť zaujímavé informácie. Zameriame sa na gradientné ukončovacie kritérium, keďže podľa neho sa pri riešení reálnej úlohy riadime.

Očakávame lepšie výsledky *metódy HS*, keďže *metóda HZ* má zložitejší tvar parametra β, ktorý sa ale v konečnom dôsledku bude rovnať parametru β^{HS}. Myslíme si preto, že by sa zbytočné aritmetické operácie pri výpočte β^{HZ} mohli prejaviť na horších výpočtových časoch. Keďže pôjde zrejme iba o nepatrné zhoršenie, zvýšili sme rozmer testovacej úlohy, aby sme prípadné rozdiely zvýraznili. Zároveň tú istú úlohu vyriešime viackrát a výsledné časy spriemerujeme, aby sme aspoň čiastočne eliminovali apriórne chyby v meraniach týchto časov.

Vstupné údaje experimentu

- rozmer úlohy: $n = 100000$
- počet opakovaní riešenia úlohy: 20
- skúmané metódy: *metóda HZ* a *metóda HS*

Z experimentu sme získali tieto priemerné časové údaje (uvádzané v sekundách)

rád hodnoty $\|g_k\|$	0	-1	-2	-3	-4	-5	-6	-7
HZ	4.51	7.41	10.24	12.93	15.61	19.02	21.83	25.15
HS	4.63	7.33	10.03	13.02	15.67	18.87	21.72	24.74

Tabuľka 6: Hagerova funkcia- výpočtové časy (v sek.) vo vzťahu ku cieľovému rádu $\|g_k\|$.

Získané údaje naše očakávania nepotvrdili. Aj keď má *metóda HZ* prevažne horšie zaznamenané časy, ide o zanedbateľné rozdiely. Navyše, pri kontrole počtu potrebných mikroiterácií sa ukázalo, že *metóda HS* bola predsalen rýchlejšia.

rád hodnoty $\|g_k\|$	0	-1	-2	-3	-4	-5	-6	-7
HZ	23	42	62	81	100	123	141	163
HS	24	42	61	80	99	121	140	159

Tabuľka 7: Hagerova funkcia- počet mikroiterácií vo vzťahu ku cieľovému rádu $\|g_k\|$.

Dôvodom malých odchýlok v časoch je preto skôr niekoľko mikroiterácií vykonaných naviac a tiež apriórna miera nepresnosti v meraní časov. Zbytočné výpočty v β^{HZ} teda neboli natoľko významné, aby sa badateľne prejavili v horších výsledkoch.

Záver

Hlavným cieľom tejto bakalárskej práce bolo porovnať efektivitu MKG autorov W. Hagera a H. Zhanga s metódami iných autorov navrhnutými v priebehu desaťročí vývoja MKG. Pre splnenie tohto cieľa sme začali našu prácu teoretickými kapitolami, ktorých úlohou bolo dostatočne priblížiť skúmanú problematiku.

Základnú teóriu metódy konjugovaných gradientov sme uviedli v kapitole 1. Tá zahŕňa definíciu MKG v podkapitole 1.2, konvergenčné vety tejto metódy v prípade rýdzokonvexnej kvadratickej a v prípade všeobecnej konvexnej funkcie v podkapitole 1.3. Prezentované vety, tvrdenia a lemy sme sa snažili počas dôkazov pre lepšie porozumenie čitateľa dôsledne komentovať a jednotlivé kroky vysvetľovať. V úvodnej kapitole sme sa tiež venovali problematike nájdenia optimálneho kroku α_k (ide o minimalizačnú úlohu (3), ktorú je potrebné riešiť v každej iterácii MKG). Uviedli sme v praxi používané kritériá pre nájdenie približného optimálneho kroku, ale aj relatívne nový algoritmus, ktorý prezentovali v článku [6] autori W. Hager a H. Zhang.

Kapitolu 2 sme venovali historickému vývoju algebraického tvaru parametra α_k MKG. Zamerali sme sa na výhody a nevýhody jednotlivých tvarov v kontexte konvergencie MKG pri konvexnej účelovej funkcii.

V ďalšej kapitole sme sa zamerali na programovací aspekt nášho cieľa bakalárskej práce. Za najdôležitejšie časti tejto kapitoly považujeme podkapitoly 3.2 a 3.3. V nich sme teoretické poznatky prezentované v kapitolách 1 a 2 previedli do programových realizácií metódy hľadania optimálneho kroku α_k a metódy konjugovaných gradientov. Všetky zdrojové kódy použité v tejto práci sú súčasťou Prílohy A.

Štvrtá kapitola obsahuje prvé experimenty, ktoré dopomohli k optimálnemu nastaveniu programových realizácií popisovaných v kapitole 3. Experiment v podkapitole 4.1 preukázal nezanedbateľnosť vplyvu rozloženia vlastných čísel na výpočtovú náročnosť generovanej úlohy, hoci tá mala zafixovanú hodnotu čísla podmienenosti (Obr. 4 situáciu jasne vysvetľuje). V podkapitole 4.2 sme sa snažili o vlastné vylepšenie metódy zlatého rezu popísanej v podkapitole 3.2.1. Na základe našich výstupov z vykonaných experimentov (nachádzajú sa na strane 87) sme nami navrhovanú úpravu programovej realizácie zamietli. Náš neúspech však neznamená, že myšlienka vylepšenia tejto metódy nie je uskutočniteľná.

V našej poslednej kapitole sme sa venovali hlavnému cieľu tejto bakalárskej práce- experimentálnemu porovnaniu efektívnosti rôznych formúl MKG. Porovnávali sme tieto tvary parametra β MKG: β^{HS+} s tvarom (36), β^{HZ+} s tvarom (48), β^{FR} s tvarom (37) a β^{DY} s tvarom (40). Experimenty sme robili na troch účelových funkciách, konkrétne na bikvadratickej s predpisom (66), diagonálnej 2 s predpisom (67) a Hagerovej s predpisom (68).
V prípade bikvadratickej účelovej funkcie sme zistili, že ak nevyžadujeme príliš vysokú presnosť v $\|x_k - \hat{x}\|$ alebo v $\|g_k\|$, potom je metóda HZ spomedzi skúmaných metód naozaj najefektívnejšia. Dokazujú to výsledky experimentov uvedené v Tab. 11, Tab. 12 a Tab. 13 na strane 93. Pri vysokej požadovanej presnosti metóda HZ svoju efektivitu stráca. Najefektívnejšou sa stáva metóda DY ako uvádzame na strane 62.
V ostatných dvoch prípadoch skúmaných účelových funkcií naše experimenty na stranách 64 až 66 preukázali vyššiu efektívnosť metódy HZ oproti konkurenčným testovaným metódam ako to bolo deklarované v závere článku [6] autorov tejto metódy.

Prínos teoretickej časti našej bakalárskej práce vidíme v zhrnutí najdôležitejších aspektov MKG. Prvé dve kapitoly našej práce môžu poslúžiť na doplnenie poznatkov o MKG v prípade hlbšieho záujmu študentov o túto metódu.

Praktická časť našej práce bola zameraná na programové realizácie metód popisovaných v tejto práci, na riešenie problémov spojených s prechodom od teórie k praktickému použitiu a nakoniec na porovnanie efektívnosti popisovaných metód. V priebehu práce sme upozorňovali na problémy, s ktorými sa môže čitateľ stretnúť pri vytváraní programových realizácií teoretických metód. Toto považujeme za ďalší dôležitý prísnos našej práce.

Tvorba tejto bakalárskej práce priniesla autorom mnoho nových poznatkov z numerickej matematiky či algebry. Taktiež musíme spomenúť programátorskú časť, ktorá tvorí významnú zložku tejto práce. Popri rozšírení si poznatkov o matematickom programe Matlab tým myslíme vytváranie programových realizácií teoretických metód a riešenie problémov vznikajúcich obmedzenými možnosťami výpočtovej techniky. V neposlednom rade autori získali cenné skúsenosti z oblasti experimentálneho porovnania matematických metód a formálneho spracovania získaných výsledkov.

Zoznam použitej literatúry

[1] Al-Baali M.: *Descent property and global convergence of the Fletcher-Reeves method with inexact line search*, IMA J. Numer. Anal., 5, 1985, str. 121-124

[2] Andrei N.: *An unconstrained optimization test function collection*, Advanced Modeling and Optimization, Volume 10, 2008, str 4-5

[3] Dai Y.H., Yuan Y.: *Convergence properties of the Fletcher-Reeves method*, IMA J. Nonlinear conjugate gradient methods 19,Numer. Anal., 16, 1996, str 155-164

[4] Dolan E., Moré J.: *Benchmarking optimization software with performance profiles*,Math. Program.,Ser. A 91, 2002, str 201-213, dostupné na internete (22.5.2012):http://www.tops-scidac.org/papers/DM_benchmark.pdf

[5] Fletcher R. and Reeves C.: *Function minimization by conjugate gradients*, Comput. J., 7, 1964, str 149-154

[6] Hager W.W., Zhang H.: *A new conjugate gradient method with guaranteed descent and an efficient line search*, SIAM J. OPTIM., Vol 16, No. 1, str 170-192, dostupné na internete (1.12.2011): http://www.math.ufl.edu/~hager/papers/CG/cg_descent.pdf

[7] Hager W.W., Zhang H.: *A survey of nonlinear conjugate gradient methods*, Pacific J. Optim. 2,2006, str. 35–58

[8] Hamala M.: *Prednášky z nelineárneho programovania*, FMFI UK, Bratislava, 2011

[9] Hestenes M.R. and Stiefel E.: *Methods of conjugate gradients for solving linear systems*, J. Research Nat. Bur. Standards, 49, 1952, str 409-436, dostupné na internete (1.12.2011): http://www.stanford.edu/class/cme324/classics/hestenes-stiefel.pdf

[10] Luenberger D.G.,Ye Y.: *Linear and nonlinear programming*, Third edition, Springer, New York, 2008

[11] Polak E.,Ribiere G.: *Note sur la convergence de directions conjugees*, Rev. Francaise Informat Recherche Opertionelle, 3e Annee 16, 1969, str. 35-43

[12] Polyak B.T.: *The conjugate gradient method in extreme problems*, USSR Comp. Math.and Math. Phys., 9, 1969, str 94-112

[13] Powell M. J. D. :*Restart procedures of the conjugate gradient method*, Math. Prog., 2 , 1977,str. 241-254

[14] Powell M.J.D.: *Nonconvex minimization calculations and the conjugate gradient method*, Numerical Analysis (Dundee, 1983), Lecture Notes in Mathematics, Vol. 1066, Springer-Verlag, Berlin, 1984, str. 122-141

[15] Pytlak R.: *Conjugate gradient algorithms in nonconvex optimization*, Nonconvex Optimization and Its Applications, Vol. 89, Springer-Verlag, Berlin, 2009

Príloha A

A.1 Generátor kvadratických úloh

```
function [G,h,xopt,xs]=Generkva(condnum,n,ro,typ)
%   generator kvadratickej ulohy s kladne definitnou maticou G
%   ucelova funkcia f(x)=1/2 x'Gx +h'x
%=====================================================================
%   VSTUPY:
%       condnum -  cislo podmienenosti matic
%       n       -  rozmer ulohy
%       ro      -  vzdialenost startovacieho x od optimalneho v euk. norme
%       typ     -  parameter rozvrhnutia vlastnych cisel v matici D
%=====================================================================
%   VYSTUPY:
%       A      -   kladne definitna matica
%       h      -   vektor
%       xopt   -   optimalne riesenie vygenerovanej kvadratickej ulohy
%       xs     -   startovaci bod, ktory od xopt vzdialeny v norme o ro
%=====================================================================

%   pouzivane konstanty
xoptmin=10;                     %pre generovanie optimalneho x

%=====================================================================
%   BLOK 1: celociselne optimalne riesenie kvadratickej ulohy
xopt=randi([-xoptmin,xoptmin],[n,1]);   %[rozsah zloziek],[rozmer vektora]

%=====================================================================
%   BLOK 2: vytvorenie symetrickej kladnedefinitnej matice G

L=tril(randi([-5,5],[n,n]),-1);     %dolnatroj.matica bez hlavnej diagonali
v=randi([1,5],[n,1]);               %kladna hlavna diagonala
L=diag(v)+L;                        %dolnatrojuhol. s hlavnou diagonalou
G=L*L';                             %regularna matica
[L,~]=qr(G);                        %L je ortogonalna matica

%   vytvorenie diagonalnej matice so zadanym cislom podmienenosti
%   najprv diagonalu s urcenym rozlozenim vlastnych cisel
switch typ
%           najvacsie cislo v(1), najmensie v(end)
%           podla parametra typ urobi particne rozlozenie vlastnych cisel
    case {1}
%           rovnomerna ,ostromonotonna postupnost vlastnych cisel
    vrch=condnum*0.001;
    v=0.001+(vrch-0.001)*rand(n-2,1);
    v=round(v*1000)/1000;
    v=-sort(-[vrch;v;0.001]);
    case {2}
%           polovica najmensie vlastne cislo, druha polovica
%           rovnomerna ,ostromonotonna postupnost
    v=ones(n,1);
    vrch=condnum*0.001;
```

```
    v(1:round(n/2))=0.001;
    v(round(n/2)+1:n-1)=0.002+(0.99*vrch-0.002)...
    *rand(length(v(round(n/2)+1:n-1)),1);
    v(n)=vrch;
    v=-sort(-v);
    case {3}
%           polovica najvacsie vla. cislo, druha polovica
%           rovnomerna ,ostromonotonna postupnost
    v=ones(n,1);
    vrch=condnum*0.001;
    v(1:round(n/2))=vrch;
    v(round(n/2)+1:n-1)=0.002+(0.99*vrch-0.002)...
    *rand(length(v(round(n/2)+1:n-1)),1);
    v(n)=0.001;
    v=-sort(-v);
    case {4}
%           rovnake cislo - rovnomerna ostromonotonna pos. - rovnake cislo
    v=ones(n,1);
    vrch=condnum*0.001;
    v(1:round(n/3))=vrch;
    v(round(n/3)+1:round(2*n/3))=0.002+(0.99*vrch-0.002)...
    *rand(length(v(round(n/3)+1:round(2*n/3))),1);
    v(round(2*n/3)+1:n)=0.001;
    v=-sort(-v);
    case{5}
%           vla. cisla z rovnomerneho roz., neusporiadane do monotonnej pos.
    v=ones(n,1);
    vrch=condnum*0.001;
    v(2:n-1)=0.002+(0.99*vrch-0.002)*rand(length(v(2:n-1)),1);
    v(1)=vrch; v(n)=0.001;
    case{6}
%           2 skupiny skoro rovnakych cisel- podobne cisla
%           v skupine, ale zmenene o male epsilon
    eps=10^(-6);
    v=ones(n,1);
    vrch=condnum*0.001;
    for i=1:1:floor(n/2)
        v(i)=vrch+eps*(1-i);
    end
    for i=n:-1:floor((n/2)+1)
        v(i)=0.001+eps*(n-i);
    end
end
%   vytvorenie kladne definitnej matice G s urcenym cislom podmienenosti
G=L*diag(v)*L';

%=====================================================================
%   BLOK3 : dopocitanie spravneho h, aby bolo xopt optimalnym riesenim
h=-G*xopt;

%=====================================================================
%   BLOK4: vytvorenie startovacieho bodu vzd. od optima v norme o ro
v=randi([-xoptmin,xoptmin],[n,1]);   %[rozsah zloziek],[rozmer vektora]
xs=xopt+(ro/(norm(v-xopt)))*(v-xopt);
end
```

A.2 Generátor bikvadratických úloh

```
function [D,G,h,xopt,xs]=Generbikva(condnum,n,ro)
%   generator bikvadratickej ulohy s diag. D a sym.G – obe kladnedefinitne
%   ucelova funkcia f(x)=1/4 (x'Dx)^2 + 1/2 x'Gx +h'x
%=====================================================================
%   VSTUPY:
%       n       – rozmer ulohy
%       condnum – cislo podmienenosti matic
%       ro      – vzdialenost startovacieho x0 od optimalneho v euk. norme
%=====================================================================
%   VYSTUPY:
%       B       – kladnedefinitna diagonalna matica D
%       A       – kladne definitna symetricka matica G
%       h       – vektor
%       xopt    – optimalne riesenie vygenerovanej kvadratickej ulohy
%       xs      – startovaci bod, ktory od xopt vzdialeny v norme o ro
%=====================================================================

%   pouzivane konstanty
xoptmin=20;                     %pre generovanie optimalneho x

%=====================================================================
%   BLOK 1: optimalne riesenie ulohy,celociselne
xopt=randi([-xoptmin,xoptmin],[n,1]);

%=====================================================================
%   BLOK 2: vytvorenie symetrickej kladnedefinitnej matice
L=tril(randi([-3,3],[n,n]),-1);         %dolnatrojuhol. bez diagonali
v=randi([1,5],[n,1]);                   %vytvaram + diagonalu
L=diag(v)+L;                            %dolantrojuhol.
G=L*L';                                 %regularna matica
[L,~]=qr(G);                            %L je ortogonalna matica
%   vytvorenie diagonalnej matice s cislom podmienenosti stanovenym
vrch=condnum*0.001;
v=0.001+(vrch-0.001)*rand(n-2,1);
v=round(v*1000)/1000;           %vo v diagonala matice s urcenym
v=-sort(-[vrch;v;0.001]);       %cislom podmienenosti
G=L*diag(v)*L';                 %predstavuje G
%   vytvorenie diagonalnej kladne definitnej matice D
v=0.001+(vrch-0.001)*rand(n-2,1);
v=round(v*1000)/1000;
v=-sort(-[vrch;v;0.001]);
D=diag(v);

%=====================================================================
%   BLOK 3: dopocitanie vektora h
h=-G*xopt - (xopt'*D*xopt)*(D*xopt);

%=====================================================================
%   BLOK 4: vytvorenie startovacieho bodu v norme od optima o ro
v=randi([-xoptmin,xoptmin],[n,1]);   %[rozsah zloziek],[rozmer vektora]
xs=xopt+(ro/(norm(v-xopt)))*(v-xopt);
end
```

A.3 Realizácia zlatého rezu

```
function [b]=Zlatyrez(x,d,w)

%   program zlateho rezu na urcenie priblizneho optimalneho kroku
%=========================================================================
%   VSTUPY:
%       x- bod, v ktorom sa nachdzam
%       d- smer, v ktorom z bodu x idem minimalizovat
%       w- konstanta- fixny pocet iteracii ZR, ktore maju prebehnut
%   VYSTUPY:
%       c1- priblizny optimalny krok
%=========================================================================

global Fval;            %zapis funkcie pre vypocet hodnoty ucelovej funkcie
z2=0.5*(sqrt(5)-1);
z1=1-z2;

%   prva faza-optimalne zuzit interval, aby "rychlejsi a presnejsi ZR"
[a,b]=PrvafazaZR(x,d,1+z2);

c1=a+z1*(b-a);              %vypocet laveho vnutorneho bodu (a,b)
c2=a+z2*(b-a);              %vypocet praveho vnutorneho bodu (a,b)
F1=Fval(x+c1*d);            %vypocet funkcnej hodnoty v c1
F2=Fval(x+c2*d);            %vypocet funkcnej hodnoty v c2
k=1;

while(k<w)  %algoritmus opakujem az po stanoveny pocet iteracii w
            %sledujem,ci plati F1 priblizne= F2,
            %co by viedlo k neistym vysledkom ZR
   if((abs(F1-F2)/(abs(F1)+1))<10^-15)
       return;
   end
   if((F2/(sign(F1)*F1))>sign(F1))       %ak v c2 vacsia hodnota ako v c1
       b=c2;                             %optimum vlavo na intervale (a,c2)
       c2=c1;
       F2=F1;
       c1=a+z1*(b-a);
       F1=Fval(x+c1*d);
   else
       a=c1;                             %optimu vpravo na intervale (c1,b)
       c1=c2;
       F1=F2;
       c2=a+z2*(b-a);
       F2=Fval(x+c2*d);
   end
   k=k+1;
end
end

%=========================================================================
%=========================================================================
```

```
function [a,b]=PrvafazaZR(x0,d,posun)

%   prva faza ZR, ktora urci pociatocny interval, ktory sa bude skracovat
%=========================================================================
%   VSTUPY:
%       x0–aktualny bod, v ktorom sa algoritmus MKG nachadza
%       d – smer minimalizacie
%       posun– konstanta, o ktoru sa bude pravy krajny bod posuvat
%=========================================================================
%   VYSTUPY:
%       a–lavy krajny bod pociatocneho intervalu pre ZR
%       b–pravy krajny bod pociatocneho intervalu pre ZR
%=========================================================================
%POSTUP:
%   zacnem v bode x0+parameter*d
%   vycislim funkcnu hodnotu, porovnam s predchadzajucou
%   ak nastane situacia F(bod1)>F(bod2)<F(bod3)
%       potom vystupom budu krajne body intervalu (bod1,bod3)
%=========================================================================

global Fval;
F0=Fval(x0);
a=0;                              %zacinam v bodoch x0 a x0+d
c=0;
b=1;
x=x0+b*d;
F1=Fval(x);
it=1;

while((F1/(sign(F0)*F0))<sign(F0))
%   viem,ze Fval(a)<Fval(c)<Fval(b), takze optimum vpravo od a
    a=c;        %posuvam lavy bod, aby som vyuzil ziskanu informaciu
    c=b;
    b=b+it*posun;
    it=it+1;
%   posuvam sa o hodnotu k*(1+z2) -> optimalny posun, ak b=1 nestaci
    F0=F1;
    x=x0+b*d;
    F1=Fval(x);
end
end
```

A.4 Line search autorov W. Hagera a H. Zhanga

```
function [alfa]=LSHZ(x,d,Maxit)

%   line search prezentovany v clanku Hagera Zhanga
%=====================================================================
%   VSTUPY:
%       x     – aktualny bod, v ktorom sa MKG nachadza
%       d     – smer optimalizacie
%       Maxit – maximalny pocet cyklov v algoritme (poistka pre zacyklenie)
%=====================================================================
%   VYSTUPY:
%       alfa  – priblizny optimalny krok
%=====================================================================

global gam;

%   ziskanie pociatocneho intervalu na zuzenie
[a,b,Dl,Dp]=Prvafaza(x,d); %vid popis v Prvafaza()
%   Dl–> derivacia v bode x+a*d            Dp–> derivacia v bode x+b*d

P=0;
%=====================================================================

%   kontrola splnenia priblizneho Wolfeho kriteria
[hodnota,D0,~]=Wolfeapr(x,a,d,0,Dl);  %ak hodnota=1,Wolfe splneny
%   D0–> derivacia v bode x –> Fder(x+0*d)'*d
if(hodnota==1)                    alfa=a;return;  end
if(Wolfeapr(x,b,d,D0,Dp)==1)      alfa=b;return;  end
clear hodnota;
%=====================================================================

while(P<=Maxit)              %algoritmus spresnovania pociatocneho intervalu
%   secant2 metoda autorov Hagera,Zhanga, vid popis secant2()
    [A,B,Dl,Dp]=secant2(a,b,x,d,Dl,Dp);
    %poznam derivacie v bodoch x+a*d,x+b*d
    %zaznamenam derivacie x+A*d,x+B*d do premennych Dl,Dp

    if((B–A)> gam*(b–a))      %ak nedoslo k dostatocnemu skrateniu,bisekcia
%       metoda bisekcie
        c=(A+B)/2;
       [A,B,Dl,Dp]=updateInt(A,B,c,x,d,Dl,Dp);
    end

%   kontrola splnenia priblizneho Wolfeho kriteria
    if(Wolfeapr(x,A,d,D0,Dl)==1)     alfa=A;return;  end
    if(Wolfeapr(x,B,d,D0,Dp)==1)     alfa=B;return;  end
    P=P+1;
%   aktualizacia intervalu, ktory algoritmus spresnuje
    a=A;
    b=B;
end
end
%=====================================================================
%=====================================================================
```

```
function [an,bn,DerL,DerP]=secant2(a,b,x,d,DerL,DerP)

%   UCEL:   spresnenie intervalu, ktory obsahuje optimalny krok
%           skumam funkciu 1 premennej: f(x+alfa*d) s premennou alfa
%           derivacia v tvare f'(x+alfa*d)*d
%=========================================================================
%   VSTUPY:
%       a   - lavy krajny bod skumaneho intervalu
%       b   - pravy krajny bod skumaneho intervalu
%       x   - aktualny bod MKG              d- smer optimalizacie
%       DerL- derivacia v bode x+a*d    DerP-derivacia v bode x+b*d
%   VYSTUPY:
%       an  - aktualizovany lavy krajny bod skumaneho intervalu
%       bn  - aktualizovany pravy krajny bod skumaneho intervalu
%       DerL- derivacia v x+an*d        DerP-derivacia v x+bn*d
%=========================================================================

[c]=secant(a,b,x,d,DerL,DerP);      %vid secant()
%   update intervalu, pozri komentar vo funkcii updateINT()
[av,bv,DerLn,DerPn]=updateInt(a,b,c,x,d,DerL,DerP);
if(c==av)
    [an,bn,DerL,DerP]=updateInt(av,bv,secant(a,av,x,d,DerL,DerLn),x,d,DerLn,DerPn);
    %vychadzam z toho,ze c=av teda Fc=DerLn pre potreby secantu
elseif(c==bv)
    [an,bn,DerL,DerP]=updateInt(av,bv,secant(bv,b,x,d,DerPn,DerP),x,d,DerLn,DerPn);
else
    an=av;bn=bv;
%   ak nastala tato moznost, potom v update intervale nastala situacia
%   ze c bolo mimo a,b a teda derivacie krajnych bodov ostavaju
%   DerL,DerP
end
end
%=========================================================================
%=========================================================================

function [an,bn,poa,pob]=updateInt(a,b,c,x,d,poa,pob)
%   aktualizacia intervalu
%   UCEL: spresnenie intervalu, ktory obsahuje optimalny krok
%=========================================================================
%   VSTUPY:
%       a- lavy krajny bod skumaneho intervalu
%       b- pravy krajny bod skumaneho intervalu
%       c- bod z intervalu <a,b> -> novy krajny bod spresneneho intervalu
%       x- aktualny bod MKG             d- smer optimalizacie
%       poa- derivacia v bode x+a*d     pob-derivacia v bode x+b*d
%   VYSTUPY:
%       an- aktualizovany lavy krajny bod skumaneho intervalu
%       bn- aktualizovany pravy krajny bod skumaneho intervalu
%       poa- derivacia v x+an*d         pob-derivacia v x+bn*d
%=========================================================================

global Fder;
if(c<a || c>b)   %ak c mimo <a,b>, interval nemam ako spresnit
    an=a;bn=b;return;
else
```

```
    Fc=Fder(x+c*d)'*d;
    if(Fc > 0)  %ak derivacia +, optimalny krok je niekde nalavo od c
        an=a;bn=c;pob=Fc;return
    else        %optimalny krok je niekde napravo od c
        an=c;bn=b;poa=Fc;return
    end
end
end
%=====================================================================
%=====================================================================

function [c]=secant(a,b,x,d,poa,pob)
%   metoda secnic
%   UCEL: urcenie presnejsieho bodu, ze f'(x+bod*d)*d=0
%=====================================================================
%   VSTUPY:
%       a- lavy krajny bod skumaneho intervalu
%       b- pravy krajny bod skumaneho intervalu
%       x- aktualny bod MKG            d- smer optimalizacie
%       poa- derivacia v bode x+a*d    pob-derivacia v bode x+b*d
%   VYSTUPY:
%       c- bod vysledkom metody secantov
%=====================================================================

global Fder;
%   ak nemam vstupnu informaciu, musim si ju vypocitat
if(pob==0)  pob=Fder(x+b*d)'*d;   end
if(poa==0)  poa=Fder(x+a*d)'*d;   end

%   c= ((a * pob)-(b * poa))/(pob - poa)  teoreticky predpis
c=a+(b-a)*(-poa/(pob-poa));             %vypoctovo stabilnejsia formula

if(isnan(c))
   fprintf('chyba v secant algoritme \n');
   pause;
   error('v secante c=Nan');
end
end
%=====================================================================
%=====================================================================

function [h,pom0,pomA]=Wolfeapr(x,alfa,d,pom0,pomA)
%   priblizne Wolfeho kriterium
%   UCEL: overenie splnenia oboch podmienok priblizneho Wolfeho kriteria
%         pre bod x+alfa*d
%=====================================================================
%   VSTUPY:
%       x- aktualny bod MKG             d- smer optimalizacie
%       alfa- aproximacia optimalneho kroku
%       pom0- derivacia v bode x        pomA-derivacia v bode x+alfa*d
%   VYSTUPY:
%       h- hodnota 1, ak su splnene podmienky priblizneho Wolfeho kriteria
%       pom0- derivacia v bode x        pomA-derivacia v bode x+alfa*d
%=====================================================================

global Fder;
```

```
global delta;
global sig;
h=0;

%   ak nemam potrebne vstupne udaje(derivacie), tak ich vypocitam
if(pomA==0)      pomA=Fder(x+alfa*d)'*d;      end
if(pom0==0)      pom0=Fder(x)'*d;             end

if(pom0>=0)
%   situacia moze nastat, ak smer d nespadovy alebo numericke chyby
    eh=errordlg('Koniec algoritmu! Derivácia v alfa=0 nezáporná!');
    uiwait(eh);
    fclose('all');
    error('program ukonceny');
end

%   prva adruha podmienka priblizneho Wolfeho kriteria
%   vztah delta a sigma    delta < min(0.5,sigma)
if((2*delta*pom0>=pom0+pomA)&&(pomA>=sig*pom0))
    h=1;
end
end
%=========================================================================
%=========================================================================

function [a,b,Fa,Fb]=Prvafaza(x0,d)
%   prva faza
%   UCEL: ziskanie pociatocneho intervalu, ktory obsahuje optimalny krok
%=========================================================================
%   VSTUPY:
%       x0- aktualny bod           d-smer optimalizacie
%   VYSTUPY:
%   a- lavy krajny bod vystupneho intervalu    b-pravy krajny bod -||-
%   Fa-derivacia v bode a                      Fb-derivacia v bode b
%=========================================================================

global Fder;
a=0;
Fa=0;
b=1;
Fb=Fder(x0+b*d)'*d;
it=1;

while(Fb<0)  %hladam bod, ktory ma + derivaciu-> protilahla strana udolia
    a=b;
    Fa=Fb;
    b=b+2*it;
    it=it+1;
    Fb=Fder(x0+b*d);
end
end
```

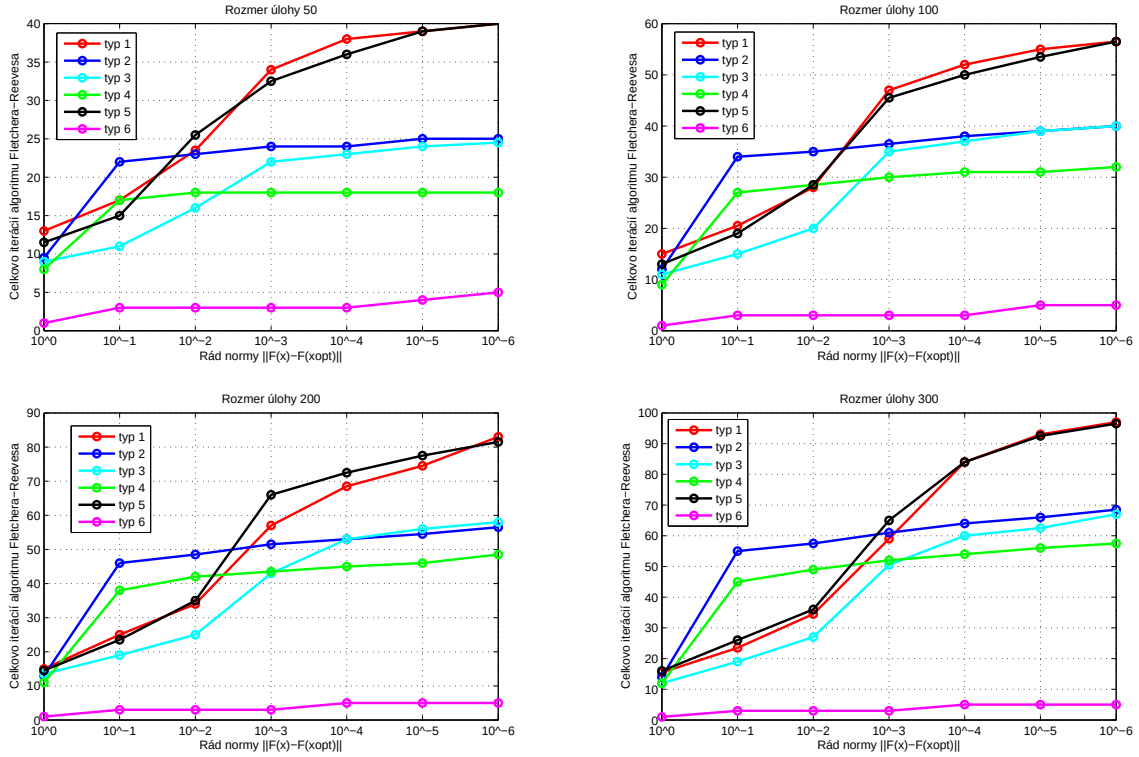
A.5 Metóda Fletchera-Reevesa

```
function [x1]=FletcherReeves(xs,macro,Fval,Fder)

%   realizacia MKG s parametrom MKG autorov Fletchera,Reevesa
%   na najdenie opt. kroku pouzita metoda zlateho rezu
%=====================================================================
%   VSTUPY:
%       xs– startovaci bod
%       macro– maximalny pocet makroiteracii programu
%       Fval– function handle na vypocet fun. hodnoty ucelovej funkcie
%       Fder– function handle na vypocet derivacie ucelovej funkcie
%=====================================================================
%   VYSTUPY:
%       x1– optimalne riesenie
%=====================================================================
%   deklarovanie potrebnych premennych
j=0;                                    %riadenie postu makroiteracii
n=length(xs);                           %zistenie rozmeru ulohy
%=====================================================================
%=====================================================================

x0=xs;
g0=Fder(x0);                            %vypocet gradientu v pociatocnom bode
%   urcenie poctu nutnych fixnych iteracii zlateho rezu
[ZRit]=Zlatyrez1(x0,–g0,norm(g0,2),4);
%=====================================================================

while(j<=macro)                          %strazenie poctu makroiteracii
    d=–g0;                              %prvy smer podla definicie –grad.

%   metoda najdenia opt. kroku–> ak kvadraticka ucelova funkcia, pouzitelna
%   formula pre opt. krok, inak metoda zlateho rezu
%alfa=–(d'*g0)/(d'*G*d);
    alfa=Zlatyrez(x0,d,ZRit);
    x1=x0+alfa*d;
    g1=Fder(x1);
    k=1;                    %strazenie poctu mikroiteracii–iba tolko linearne
    while(k<n)              %nezavislych smerov, aky je rozmer riesenej ulohy
%       konstrukcia smeru podla formuly (5) v kapitole 1
        d=–g1 + ((norm(g1,2)/norm(g0,2))^2)*d;
        g0=g1;
        x0=x1;
        alfa=Zlatyrez(x0,d,ZRit);
%    alfa=–(d'*g0)/(d'*G*d);
        x1=x0 + alfa*d;
        g1=Fder(x1);
        k=k+1;
    end         %restart po doiahnuti max. poctu lin. nezavislych smerov
x0=x1;
g0=g1;
j=j+1;
end
end
```

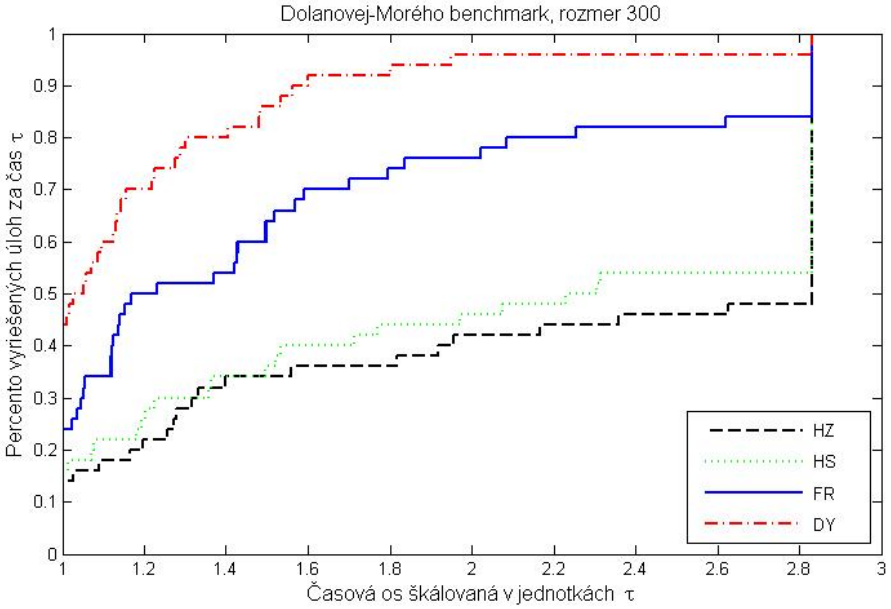
A.6 Metóda Hagera-Zhanga

```
function [x1]=HagerZhang(xs,macro,Fval,Fder)

%   realizacia MKG s parametrom MKG autorov Hagera,Zhanga
%   na najdenie opt. kroku pouzita metoda Hagera,Zhanga
%=========================================================================
%   VSTUPY:
%       xs– startovaci bod
%       macro– maximalny pocet makroiteracii programu
%       Fval– function handle na vypocet fun. hodnoty ucelovej funkcie
%       Fder– function handle na vypocet derivacie ucelovej funkcie
%=========================================================================
%   VYSTUPY:
%       x1– optimalne riesenie
%=========================================================================

%   pouzite konstanty a pomocne premenne
    j=0;                        %strazi pocet makroiteracii
    n=length(xs);               %rozmer riesenej ulohy
    const1=0.01;                %vo vypocte parametra MKG Hagera a Zhanga
    global delta;global sig;global gam;
    delta=0.1;                  %konstanta do priblizneho Wolfeho kriteria
    sig=0.9;                    %konstanta do priblizneho Wolfeho kriteria
    gam=0.66;   %konstanta do programu LSHZ–> kontrola skratenia intervalu
%=========================================================================

x0=xs;
g0=Fder(x0);
[ZRit]=Zlatyrez1(x0,–g0,norm(g0,2),4);

while(j<=macro)
    d=–g0;
    alfa=LSHZ(x0,d,ZRit);                           %optimalny krok cez LSHZ
    x1=x0+alfa*d;
    g1=Fder(x1);
    k=1;
    while(k<n)
        y=g1–g0;
        Beta=(g1'*y – 2*((norm(y,2)^2)/(d'*y))*g1'*d)/(d'*y);
%   novy smer podla formuly (5) v kapitole 1
        d=–g1 + max(Beta,–1/(norm(d,2) * min(norm(g0,2),const1)))*d;
        g0=g1;
        x0=x1;
        alfa=LSHZ(x0,d,ZRit);                       %optimalny krok cez LSHZ
        x1=x0 + alfa*d;
        g1=Fder(x1);
        k=k+1;
    end
x0=x1;
g0=g1;
j=j+1;
end
end
```

Príloha B



Obr. B.1: Podkapitola 4.1: Vzťah rozloženia vlastných čísel a náročnosti na výpočet takej úlohy



Obr. B.2: Podkapitola 5.1, experiment 2: Porovnanie efektívnosti skúmaných metód na úlohách s bikvadratickou účelovou funkciou

Príloha C

Hodnota koštanty ω= 9		
iterácia	$\|x_k - \hat{x}\|$	náročnosť
1	5.54e+001	13
17	9.83e+000	238
51	9.59e-001	725
60	9.71e-002	856
66	8.75e-003	942
99	9.49e-004	1292

(a)

Hodnota koštanty ω= 14		
iterácia	$\|x_k - \hat{x}\|$	náročnosť
1	5.52e+001	18
17	9.84e+000	323
50	9.86e-001	961
60	8.39e-002	1151
64	5.15e-003	1226
68	7.95e-004	1291
99	1.94e-004	1508

(b)

Hodnota koštanty ω= 19		
iterácia	$\|x_k - \hat{x}\|$	náročnosť
1	5.52e+001	23
17	9.84e+000	408
50	9.75e-001	1183
60	7.96e-002	1400
64	4.36e-003	1480
67	6.99e-004	1529
99	1.29e-004	1736

(c)

Hodnota koštanty ω= 23		
iterácia	$\|x_k - \hat{x}\|$	náročnosť
1	5.52e+001	27
17	9.84e+000	474
50	9.68e-001	1288
59	9.68e-002	1490
63	8.20e-003	1567
67	9.94e-004	1634
99	5.20e-004	1841

(d)

Hodnota koštanty ω= 28		
iterácia	$\|x_k - \hat{x}\|$	náročnosť
1	5.52e+001	32
17	9.84e+000	522
50	9.68e-001	1333
59	9.68e-002	1529
63	9.05e-003	1610
66	8.63e-004	1662
72	9.16e-005	1728
99	6.80e-005	1862

(e)

Tabuľka 8: Výstup prvej časti experimentu z podkapitoly 4.2

Úplný výstup prvej časti experimentu z podkapitoly 4.2[5]

Fixný počet iterácií programu ZlatyRez: 14

it	||(x_k)-x^||	#ZRit	||grad(x_k)||	f(x_k)-f(x^)
1	5.5e+001	18	1.4e+001	+2.8e+002
2	4.0e+001	37	6.3e+000	+8.3e+001
3	2.9e+001	56	3.6e+000	+2.9e+001
4	2.4e+001	75	2.2e+000	+1.4e+001
5	2.0e+001	94	1.7e+000	+6.8e+000
6	1.7e+001	113	9.8e-001	+2.4e+000
7	1.6e+001	132	5.6e-001	+1.4e+000
8	1.6e+001	151	4.1e-001	+8.7e-001
9	1.5e+001	170	2.5e-001	+6.6e-001
10	1.5e+001	189	2.1e-001	+5.4e-001
11	1.4e+001	208	2.0e-001	+4.6e-001
12	1.3e+001	227	1.7e-001	+3.9e-001
13	1.3e+001	246	1.7e-001	+3.3e-001
14	1.2e+001	266	1.9e-001	+2.6e-001
15	1.1e+001	285	1.9e-001	+1.9e-001
16	1.0e+001	304	1.2e-001	+1.4e-001
17	9.8e+000	323	8.0e-002	+1.2e-001
18	9.6e+000	342	7.7e-002	+1.1e-001
19	9.3e+000	361	6.6e-002	+9.6e-002
20	9.1e+000	380	6.3e-002	+8.8e-002
21	8.7e+000	400	6.7e-002	+8.0e-002
22	8.3e+000	420	6.9e-002	+6.9e-002
23	7.8e+000	439	6.0e-002	+6.0e-002
24	7.5e+000	458	4.6e-002	+5.5e-002
25	7.3e+000	477	3.3e-002	+5.1e-002
26	7.1e+000	496	3.8e-002	+4.9e-002
27	6.8e+000	516	4.8e-002	+4.5e-002
28	6.2e+000	535	4.3e-002	+4.1e-002
29	5.6e+000	555	5.8e-002	+3.6e-002
30	4.0e+000	575	7.5e-002	+2.3e-002
31	3.0e+000	594	4.6e-002	+1.4e-002
32	2.7e+000	613	3.8e-002	+1.2e-002
33	2.4e+000	632	3.5e-002	+8.7e-003
34	2.2e+000	651	3.4e-002	+6.5e-003
35	2.1e+000	670	2.6e-002	+4.5e-003
36	2.0e+000	689	1.8e-002	+3.4e-003
37	1.9e+000	708	1.6e-002	+2.9e-003
38	1.9e+000	727	1.1e-002	+2.4e-003
39	1.9e+000	746	8.6e-003	+2.2e-003
40	1.8e+000	766	9.3e-003	+2.0e-003
41	1.8e+000	786	9.7e-003	+1.8e-003
42	1.8e+000	805	5.8e-003	+1.7e-003
43	1.7e+000	824	4.3e-003	+1.6e-003
44	1.7e+000	844	4.9e-003	+1.6e-003
45	1.7e+000	863	5.0e-003	+1.5e-003
46	1.6e+000	883	6.2e-003	+1.5e-003
47	1.6e+000	902	6.9e-003	+1.4e-003
48	1.4e+000	922	1.1e-002	+1.2e-003
49	1.1e+000	942	1.1e-002	+9.2e-004
50	9.9e-001	961	5.8e-003	+7.8e-004
51	9.4e-001	980	4.5e-003	+7.4e-004
52	8.9e-001	1000	5.8e-003	+6.8e-004
53	7.4e-001	1020	8.7e-003	+5.6e-004
54	5.5e-001	1039	8.5e-003	+4.1e-004
55	4.1e-001	1058	7.4e-003	+3.0e-004
56	2.1e-001	1078	7.7e-003	+1.4e-004
57	1.3e-001	1097	3.5e-003	+7.2e-005
58	1.2e-001	1116	2.2e-003	+5.8e-005
59	1.0e-001	1135	2.1e-003	+4.9e-005
60	8.4e-002	1151	2.5e-003	+3.8e-005
61	5.5e-002	1170	3.0e-003	+2.4e-005
62	2.4e-002	1189	1.9e-003	+9.2e-006
63	1.3e-002	1208	1.3e-003	+4.1e-006
64	5.2e-003	1226	9.1e-004	+1.3e-006
65	2.2e-003	1244	4.0e-004	+2.9e-007
66	1.4e-003	1261	2.1e-004	+8.9e-008
67	1.0e-003	1277	1.1e-004	+3.6e-008
68	8.0e-004	1291	9.7e-005	+1.6e-008
69	6.6e-004	1305	4.6e-005	+5.1e-009
70	6.2e-004	1317	2.7e-005	+2.4e-009
71	5.9e-004	1326	1.8e-005	+1.2e-009
72	5.7e-004	1336	9.3e-006	+5.8e-010
73	5.6e-004	1345	5.8e-006	+4.7e-010
74	5.5e-004	1354	5.8e-006	+3.8e-010
75	5.5e-004	1362	2.4e-006	+3.2e-010
76	5.5e-004	1367	1.8e-006	+3.2e-010
77	5.4e-004	1372	1.6e-006	+3.1e-010
78	5.4e-004	1380	2.0e-006	+3.0e-010
79	5.3e-004	1385	2.0e-006	+3.0e-010
80	5.2e-004	1392	2.5e-006	+2.8e-010
81	4.9e-004	1399	3.5e-006	+2.7e-010
82	4.7e-004	1406	3.6e-006	+2.5e-010
83	4.5e-004	1411	3.1e-006	+2.4e-010
84	4.3e-004	1416	2.8e-006	+2.3e-010
85	4.1e-004	1423	2.9e-006	+2.2e-010
86	3.9e-004	1430	3.1e-006	+2.0e-010
87	3.7e-004	1436	3.2e-006	+1.9e-010
88	3.5e-004	1442	3.0e-006	+1.7e-010
89	3.3e-004	1448	2.8e-006	+1.6e-010
90	3.1e-004	1454	2.7e-006	+1.4e-010
91	2.9e-004	1460	2.5e-006	+1.3e-010
92	2.8e-004	1466	2.3e-006	+1.1e-010
93	2.6e-004	1472	2.1e-006	+1.0e-010
94	2.5e-004	1478	2.0e-006	+9.3e-011
95	2.2e-004	1485	2.4e-006	+7.9e-011
96	2.1e-004	1490	2.4e-006	+7.2e-011
97	2.1e-004	1494	2.4e-006	+7.2e-011
98	2.1e-004	1498	2.4e-006	+7.2e-011
99	1.9e-004	1504	1.9e-006	+6.3e-011

[5] it- mikroiterácia programu, ||x_k - x^||označuje normu rozdielu aproximácie x_k a optima $\hat{x}$, ZRit označuje počet výpočtov funkčných hodnôt v programe *ZlatyRez*, ||g_k||označuje gradient $\|g_k\|$ a ||f(x_k)-f(x^)||označuje rozdiel funkčných hodnôt $f(x_k)$ a $f(\hat{x})$

Úplný výstup prvej časti experimentu z podkapitoly 4.2[6]

Fixný počet iterácií programu ZlatyRez: 28

it	||(x_k)-x^||	#ZRit	||grad(x_k)||	f(x_k)-f(x^)
1	5.5e+001	32	1.5e+001	+2.8e+002
2	4.0e+001	65	6.3e+000	+8.3e+001
3	2.9e+001	98	3.6e+000	+2.9e+001
4	2.4e+001	131	2.2e+000	+1.4e+001
5	2.0e+001	164	1.7e+000	+6.8e+000
6	1.7e+001	197	9.8e-001	+2.4e+000
7	1.6e+001	230	5.7e-001	+1.4e+000
8	1.6e+001	258	4.1e-001	+8.7e-001
9	1.5e+001	290	2.5e-001	+6.6e-001
10	1.5e+001	320	2.1e-001	+5.4e-001
11	1.4e+001	350	2.0e-001	+4.6e-001
12	1.3e+001	379	1.7e-001	+3.9e-001
13	1.3e+001	408	1.7e-001	+3.3e-001
14	1.2e+001	438	2.0e-001	+2.6e-001
15	1.1e+001	467	1.9e-001	+1.9e-001
16	1.0e+001	495	1.2e-001	+1.4e-001
17	9.8e+000	522	8.0e-002	+1.2e-001
18	9.6e+000	550	7.7e-002	+1.1e-001
19	9.3e+000	576	6.6e-002	+9.6e-002
20	9.1e+000	603	6.3e-002	+8.8e-002
21	8.7e+000	627	6.7e-002	+8.0e-002
22	8.3e+000	656	6.9e-002	+6.9e-002
23	7.8e+000	682	6.0e-002	+6.0e-002
24	7.5e+000	710	4.6e-002	+5.5e-002
25	7.3e+000	736	3.3e-002	+5.1e-002
26	7.1e+000	760	3.8e-002	+4.9e-002
27	6.8e+000	786	4.8e-002	+4.5e-002
28	6.2e+000	810	4.3e-002	+4.1e-002
29	5.6e+000	837	5.8e-002	+3.6e-002
30	4.0e+000	864	7.5e-002	+2.3e-002
31	3.0e+000	890	4.6e-002	+1.4e-002
32	2.7e+000	916	3.8e-002	+1.2e-002
33	2.4e+000	942	3.5e-002	+8.7e-003
34	2.2e+000	965	3.4e-002	+6.5e-003
35	2.1e+000	989	2.6e-002	+4.5e-003
36	2.0e+000	1013	1.8e-002	+3.4e-003
37	1.9e+000	1038	1.6e-002	+2.9e-003
38	1.9e+000	1062	1.1e-002	+2.4e-003
39	1.9e+000	1085	8.6e-003	+2.2e-003
40	1.8e+000	1109	9.3e-003	+2.0e-003
41	1.8e+000	1134	9.7e-003	+1.8e-003
42	1.8e+000	1157	5.8e-003	+1.7e-003
43	1.7e+000	1179	4.2e-003	+1.6e-003
44	1.7e+000	1201	4.7e-003	+1.6e-003
45	1.7e+000	1223	5.1e-003	+1.5e-003
46	1.6e+000	1244	6.2e-003	+1.5e-003
47	1.5e+000	1267	7.6e-003	+1.4e-003
48	1.3e+000	1290	1.2e-002	+1.1e-003
49	1.0e+000	1311	8.6e-003	+8.4e-004
50	9.7e-001	1333	4.6e-003	+7.6e-004
51	9.2e-001	1350	4.8e-003	+7.2e-004
52	8.3e-001	1373	7.6e-003	+6.4e-004
53	6.6e-001	1397	9.2e-003	+5.0e-004
54	4.8e-001	1420	7.2e-003	+3.6e-004
55	3.2e-001	1443	8.6e-003	+2.3e-004
56	1.6e-001	1466	5.5e-003	+9.4e-005
57	1.2e-001	1488	2.8e-003	+6.5e-005
58	1.1e-001	1509	2.1e-003	+5.4e-005
59	9.7e-002	1529	2.2e-003	+4.5e-005
60	7.4e-002	1551	2.7e-003	+3.3e-005
61	3.4e-002	1572	2.4e-003	+1.4e-005
62	1.7e-002	1592	1.5e-003	+6.1e-006
63	9.0e-003	1610	1.2e-003	+2.8e-006
64	3.3e-003	1628	7.0e-004	+7.1e-007
65	1.5e-003	1645	3.1e-004	+1.7e-007
66	8.6e-004	1662	1.5e-004	+4.6e-008
67	5.7e-004	1677	8.8e-005	+1.9e-008
68	2.6e-004	1689	5.4e-005	+4.5e-009
69	1.7e-004	1701	2.7e-005	+1.4e-009
70	1.3e-004	1709	1.3e-005	+5.6e-010
71	1.0e-004	1719	1.1e-005	+2.0e-010
72	9.2e-005	1728	5.4e-006	+5.8e-011
73	8.9e-005	1735	2.3e-006	+3.0e-011
74	8.9e-005	1742	1.5e-006	+2.6e-011
75	8.8e-005	1748	1.3e-006	+2.3e-011
76	8.7e-005	1753	1.1e-006	+2.1e-011
77	8.5e-005	1758	9.3e-007	+1.8e-011
78	8.4e-005	1763	7.9e-007	+1.7e-011
79	8.3e-005	1768	7.2e-007	+1.5e-011
80	8.2e-005	1773	6.7e-007	+1.4e-011
81	8.0e-005	1778	6.4e-007	+1.3e-011
82	7.9e-005	1783	5.9e-007	+1.2e-011
83	7.7e-005	1788	5.5e-007	+1.1e-011
84	7.6e-005	1793	5.5e-007	+1.0e-011
85	7.1e-005	1800	7.8e-007	+8.3e-012
86	6.8e-005	1806	8.2e-007	+7.0e-012
87	6.8e-005	1810	8.2e-007	+7.0e-012
88	6.8e-005	1814	8.2e-007	+7.0e-012
89	6.8e-005	1818	8.2e-007	+7.0e-012
90	6.8e-005	1823	8.2e-007	+7.0e-012
91	6.8e-005	1828	8.2e-007	+7.0e-012
92	6.8e-005	1831	8.2e-007	+7.0e-012
93	6.8e-005	1834	8.2e-007	+7.0e-012
94	6.8e-005	1838	8.2e-007	+7.0e-012
95	6.8e-005	1842	8.2e-007	+7.0e-012
96	6.8e-005	1846	8.2e-007	+7.0e-012
97	6.8e-005	1850	8.2e-007	+7.0e-012
98	6.8e-005	1854	8.2e-007	+7.0e-012
99	6.8e-005	1858	8.2e-007	+7.0e-012

[6] it- mikroiterácia programu, ||x_k - x^||označuje normu rozdielu aproximácie x_k a optima $\hat{x}$, ZRit označuje počet výpočtov funkčných hodnôt v programe *ZlatyRez*, ||g_k||označuje gradient $\|g_k\|$ a ||f(x_k)-f(x^)||označuje rozdiel funkčných hodnôt $f(x_k)$ a $f(\hat{x})$

Hodnota konštanty ω= 14		
iterácia	$\|x_k - \hat{x}\|$	náročnosť
1	5.52e+001	18
17	9.84e+000	323
50	9.86e-001	961
60	8.39e-002	1151
64	5.15e-003	1226
68	7.95e-004	1291
199	1.75e-004	2242

(a)

Hodnota konštanty ω= 28		
iterácia	$\|x_k - \hat{x}\|$	náročnosť
1	5.52e+001	32
17	9.84e+000	522
50	9.68e-001	1333
59	9.68e-002	1529
63	9.05e-003	1610
66	8.63e-004	1662
72	9.16e-005	1728
199	6.59e-005	2340

(b)

Tabuľka 9: Prvá časť experimentu z podkapitoly 4.2-pridanie makroiterácie

Hodnota konštanty ω= 14		
iterácia	$\|x_k - \hat{x}\|$	náročnosť
1	5.52e+001	18
17	9.84e+000	323
50	9.86e-001	961
60	8.39e-002	1151
64	5.15e-003	1226
68	7.95e-004	1291
99	1.94e-004	1508

(a)

Hodnota konštanty ω= 28		
iterácia	$\|x_k - \hat{x}\|$	náročnosť
1	5.52e+001	32
17	9.84e+000	522
50	9.68e-001	1333
59	9.68e-002	1529
63	9.05e-003	1610
66	8.63e-004	1662
72	9.16e-005	1728
99	6.80e-005	1862

(b)

Hodnota konštanty ω= od 14 po 28		
iterácia	$\|x_k - \hat{x}\|$	náročnosť
1	5.52e+001	18
17	9.83e+000	339
50	9.76e-001	1093
60	8.00e-002	1308
64	4.64e-003	1386
68	8.65e-004	1447
99	2.17e-004	1676

(c)

Hodnota konštanty ω= od 19 po 28		
iterácia	$\|x_k - \hat{x}\|$	náročnosť
1	5.52e+001	23
17	9.84e+000	415
50	9.74e-001	1220
60	7.90e-002	1438
64	3.71e-003	1515
67	6.54e-004	1562
99	1.08e-004	1758

(d)

Tabuľka 10: Výstup druhej časti experimentu z podkapitoly 4.2

hodnota delta=0,05			hodnota delta=0,15		
dosiahnutý rád presnosti v účelovej funkcii	**číslo miktoiterácie**	**počet výpočtov derivácií v LSHZ**	**dosiahnutý rád presnosti v účelovej funkcii**	**číslo miktoiterácie**	**počet výpočtov derivácií v LSHZ**
6	2	16	6	2	16
5	5	36	5	5	36
4	9	62	4	9	62
3	13	88	3	13	88
2	23	153	2	23	153
1	40	260	1	40	260
0	64	413	0	62	400
-1	121	760	-1	118	743
-2	159	986	-2	155	963
-3	187	1158	-3	178	1102
-4	213	1316	-4	202	1247
-5	221	1363	-5	218	1343
-6	236	1455	-6	235	1445
-7	256	1581	-7	248	1522
-8	264	1629	-8	256	1578

hodnota delta=0,25			hodnota delta=0,35		
dosiahnutý rád presnosti v účelovej funkcii	**číslo miktoiterácie**	**počet výpočtov derivácií v LSHZ**	**dosiahnutý rád presnosti v účelovej funkcii**	**číslo miktoiterácie**	**počet výpočtov derivácií v LSHZ**
6	2	16	6	2	16
5	5	36	5	5	34
4	9	62	4	8	58
3	13	88	3	13	90
2	23	154	2	23	153
1	40	260	1	40	263
0	60	387	0	57	368
-1	114	720	-1	106	667
-2	153	952	-2	139	869
-3	178	1102	-3	165	1019
-4	202	1247	-4	188	1154
-5	218	1343	-5	215	1316
-6	235	1445	-6	226	1385
-7	248	1522	-7	236	1446
-8	256	1578	-8	242	1490

Obr. C.1: Výstup experimentu z podkapitoly 4.3

rozmer úloh= 100	Číslo podmienenosti v optime úlohy min 2.56e+006	 priemer 2.91e+006	 max 3.15e+006	
Dosiahnutie rádu 0 v $\|x_k - \hat{x}\|$	priemer	medián	min	max
HS	71	31	7	683
FR	93	43	7	440
HZ	70	27	6	490
DY	82	40	7	388
Dosiahnutie rádu -2 v $\|x_k - \hat{x}\|$	priemer	medián	min	max
HS	215	177	65	890
FR	219	174	68	701
HZ	195	161	55	691
DY	197	162	70	597
Dosiahnutie rádu -4 v $\|x_k - \hat{x}\|$	priemer	medián	min	max
HS	275	246	85	983
FR	288	253	124	880
HZ	264	248	84	797
DY	262	245	121	698

Tabuľka 11: Podkapitola 5.1, experiment 1a - počet mikroiterácií MKG vo vzťahu ku cieľovému rádu $\|x_k - \hat{x}\|$.

rozmer úloh= 300	Číslo podmienenosti v optime úlohy min 2.77e+006	 priemer 3.08e+006	 max 3.27e+006	
Dosiahnutie rádu 0 v $\|x_k - \hat{x}\|$	priemer	medián	min	max
HS	29	16	9	370
FR	41	24	9	396
HZ	29	16	9	431
DY	38	21	9	395
Dosiahnutie rádu -2 v $\|x_k - \hat{x}\|$	priemer	medián	min	max
HS	405	378	50	1680
FR	390	383	72	1450
HZ	348	287	56	1748
DY	357	389	66	1400
Dosiahnutie rádu -4 v $\|x_k - \hat{x}\|$	priemer	medián	min	max
HS	540	516	95	1930
FR	573	550	162	1650
HZ	498	399	108	2039
DY	500	484	141	1476

Tabuľka 12: Podkapitola 5.1, experiment 1b - počet mikroiterácií MKG vo vzťahu ku cieľovému rádu $\|x_k - \hat{x}\|$.

rozmer úloh= 100	Číslo podmienenosti v optime úlohy min 2.56e+009	 priemer 2.91e+009	 max 3.15e+009	
Dosiahnutie rádu 0 v $\|x_k - \hat{x}\|$	priemer	medián	min	max
HS	85	28	6	491
FR	101	45	8	462
HZ	81	30	7	362
DY	92	38	8	467
Dosiahnutie rádu -2 v $\|x_k - \hat{x}\|$	priemer	medián	min	max
HS	208	168	82	768
FR	242	245	119	680
HZ	209	180	83	477
DY	228	198	90	570
Dosiahnutie rádu -4 v $\|x_k - \hat{x}\|$	priemer	medián	min	max
HS	257	250	87	782
FR	295	267	136	1476
HZ	254	253	88	776
DY	272	256	142	683

Tabuľka 13: Podkapitola 5.1, experiment 1c - počet mikroiterácií MKG vo vzťahu ku cieľovému rádu $\|x_k - \hat{x}\|$.