FAKULTA MATEMATIKY, FYZIKY A INFORMATIKY
UNIVERZITY KOMENSKÉHO
v Bratislave

EKONOMICKÁ A FINANČNÁ MATEMATIKA

DIPLOMOVÁ PRÁCA

2003 Jana Richmanová

FAKULTA MATEMATIKY, FYZIKY A INFORMATIKY
UNIVERZITY KOMENSKÉHO
v Bratislave

EKONOMICKÁ A FINANČNÁ MATEMATIKA

DIPLOMOVÁ PRÁCA

Tenzorové metódy riešenia úlohy na voľný extrém

Autor: Jana Richmanová
Vedúci diplomovej práce: Doc. RNDr. Milan Hamala, CSc.

Bratislava 2003

Čestne vyhlasujem, že som túto diplomovú prácu vypracovala samostatne, len s použitím uvedenej literatúry.

.....

V Bratislave 4.apríla 2003 Jana Richmanová

Poďakovanie

Ďakujem aj touto cestou Doc. RNDr. Milanovi Hamalovi, CSc. za odborné vedenie, cenné rady a pripomienky pri písaní diplomovej práce.

OBSAH

Úvod 1

1. Formulácia úlohy 2

2. Štandardné metódy riešenia úlohy (1.1.) (druhého rádu) 2

   2.1. Kvadratický model 2

   2.2. Newtonova a modifikovaná Newtonova metóda 3

   2.3. Kvázinewtonovské metódy 4

3. Tenzorová metóda riešenia úlohy (1.1.) 5

   3.1. Základné definície a operácie s tenzormi 5

   3.2. Tenzorový model (štvrtého rádu) 7

   3.3. Bouarichova metóda 8

   3.4. Výpočet tenzorového smeru v prípade regulárnej Hessovej matice 12

   3.5. Výpočet tenzorového smeru v prípade singulárnej Hessovej matice 16

   3.6. Optimálna dĺžka kroku 19

   3.7. Metóda zlatého rezu 20

4. Numerické experimenty 22

   4.1. Voľba testovacích funkcií 22

   4.2. Metodika experimentov 24

   4.3. Výsledky experimentov 25

      4.3.1. Ilustratívne príklady 25

      4.3.2. Testovanie na sériách po 100 úloh 29

      4.3.3. Vyhodnotenie na opakovaných sériách úloh 41

Záver 44

Literatúra 46

Príloha 47

ÚVOD

Tenzorovú metódu ako metódu na riešenie úlohy na voľný extrém prvý krát predstavili Schnabel a Chow v roku 1991 [2] ako novú metódu pre malé úlohy do 50 premenných. Vo svojej práci sa odvolávajú na článok Schnabela a Franka z roku 1984 [3], v ktorom autori predstavili tenzorovú metódu ako metódu na riešenie sústav nelineárnych rovníc. V roku 1997 bol v rovnakom časopise publikovaný článok Aliho Bouarichu [1], v ktorom autor opisuje tenzorovú metódu pre rozsiahle riedke úlohy ($n \leq 10^4$). Všetky tieto práce obsahujú aj bohaté numerické experimenty, na základe ktorých sa tenzorová metóda javí ako efektívna, dokonca efektívnejšia ako štandardné metódy, za ktoré považujeme Newtonovu a kvázinewtonovské metódy. Základný rozdiel medzi tenzorovou metódou a štandardnými metódami spočíva v použitom modeli, ktorým sa lokálne aproximuje optimalizovaná funkcia. Pri Newtonovej metóde je to kvadratický model, ktorý pozostáva z Taylorovho rozvoja do druhého rádu s presnými hodnotami gradientu a Hessovej matice. Pri kvázinewtonovských metódach je použitá namiesto presnej Hessovej matice len jej aproximácia. Tenzorové metódy používajú na lokálnu aproximáciu minimalizovanej funkcie model štvrtého rádu, ktorý pozostáva z Taylorovho rozvoja do druhého rádu plus členy tretieho a štvrtého rádu, ktoré aproximujú tretiu a štvrtú deriváciu. Tieto aproximujúce členy majú špeciálny tvar (tenzory hodnosti dva a jedna), čím umožňujú jednoduchú interpoláciu funkčných hodnôt a gradientu v dvoch rôznych bodoch. Cieľom predkladanej práce je porovnať tenzorovú metódu s modifikovanou Newtonovou metódou pri riešení malých úloh. Teoretický základ čerpáme z už spomínaného článku Aliho Bouarichu [1]. Pri štúdiu článku sme narazili na niekoľko nejasností a nepresností a to v kapitole venovanej riešeniu tenzorového modelu v prípade singulárnej Hessovej matice, ktoré sa nám podarilo odstrániť.

KAPITOLA 1

FORMULÁCIA ÚLOHY

Nech $f \in C^2$ je konvexná. Budeme riešiť úlohu na nájdenie (globálneho) minima funkcie f, čo symbolicky zapisujeme takto :

$$Min\Big\{ f(x) | x \in \Re^n \Big\} \tag{1.1}$$

Na riešenie úlohy (1.1) existujú rôzne metódy, z ktorých najznámejšie stručne popíšeme v nasledujúcej kapitole.

KAPITOLA 2

ŠTANDARDNÉ METÓDY RIEŠENIA ÚLOHY (1.1) (druhého rádu)

2.1. Kvadratický model.

Ako sme už povedali, štandardné metódy sú založené na kvadratickom modeli, ktorý pozostáva z Taylorovho rozvoja do druhého rádu. Aby bolo možné formulovať tento model funkcia f musí byť aspoň dvakrát spojito diferencovateľná.
Predpokladajme, že v k-tej iterácii máme k dispozícii bod $x_k \in \Re^n$. V okolí uvedeného bodu x_k nahradíme minimalizovanú funkciu $f(x)$ jej kvadratickou aproximáciou :

$$f(x) \approx M_N(x_k + s) = f(x_k) + g_k^T s + \frac{1}{2} s^T H_k s \tag{2.1}$$

kde x_k je aktuálny bod, g_k je hodnota gradientu $\nabla f(x_k)$, H_k je Hessova matica $\nabla^2 f(x_k)$, prípadne jej aproximácia v bode x_k, a $s \in \Re^n$ je "dostatočne krátky" vektor (napr.

$\|s\| \leq \delta$). Nasledovnú, (k+1)-iteráciu určíme ako minimum kvadratickej funkcie M_N v príslušnom okolí.

2.2. Newtonova a modifikovaná Newtonova metóda.

V Newtonovej metóde minimalizujeme kvadratickú funkciu M_N "globálne", t.j. zanedbávame fakt, že príslušná aproximácia funkcie $f(x)$ funkciou M_N má len lokálny charakter. Potom aktuálny bod x_{k+1} pre nasledujúcu iteráciu určíme takto :

$$x_{k+1} = x_k - H_k^{-1} g_k, \tag{2.2}$$

kde

$$s_k = -H_k^{-1} g_k$$

je Newtonovský smer. Práve to, že zanedbávame lokálny charakter aproximácie funkcie $f(x)$, má za následok, že Newtonova metóda vo všeobecnosti konverguje len pri "dobrej" voľbe štartovacieho bodu ("dostatočne" blízko k optimálnemu riešeniu). Tento nedostatok je eliminovaný v modifikovanej Newtonovej metóde, keď rešpektujeme lokálny charakter aproximácie funkcie $f(x)$ funkciou M_N, a teda minimum funkcie M_N použijeme len na určenie "smeru postupu". V tomto smere potom dodatočne určíme "dĺžku postupu", t.j. krok $\lambda_k > 0$. Najprirodzenejšie je voliť λ_k ako hodnotu, v ktorej funkcia

$$\varphi(\lambda) = f(x_k + \lambda s_k)$$

dosahuje minimum. Potom (k+1)-iterácia v modifikovanej Newtonovej metóde má tvar :

$$x_{k+1} = x_k + \hat{\lambda}_k s_k$$

2.3. Kvázinewtonovské metódy.

Základnou myšlienkou Kvázinewtonovských metód je postupná aproximácia inverznej Hessovej matice pomocou gradientov g_k. Problémom je teda určenie tejto aproximácie, označíme ju H_k. Predpokladajme, že v k-tej iterácii poznáme bod x_k, gradient g_k a maticu H_k. Po vypočítaní spádového smeru

$$s_k = -H_k g_k$$

a optimálneho kroku λ_k vypočítame nový bod x_{k+1} podľa vzťahu (2.2). Ďalej vypočítame gradient g_{k+1} v tomto novom bode x_{k+1} a vektory

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

a

$$s_k = x_{k+1} - x_k$$

Od novej matice H_{k+1} potom vyžadujeme splnenie nasledujúcich podmienok :

$$H_{k+1} y_k = s_k$$

a

$$H_{k+1} r = H_k r,$$

kde prvá podmienka sa označuje ako kvázinewtonovská podmienka a vyjadruje požiadavku zúžitkovania novej informácie. Druhá podmienka vyjadruje požiadavku zachovania pôvodnej informácie obsiahnutej v starej matici H_k pre všetky vektory r nejakého podpriestoru, ktorý neobsahuje vektor y_k. Z možných volieb týchto priestorov potom vyplýva rozmanitosť kvázinewtonovských formúl. Maticu H_{k+1} potom reprezentujeme v tvare aditívnej korekcie :

$$H_{k+1} = H_k + \Delta H_k,$$

kde korekčná matica ΔH_k sa konštruuje pomocou známych vektorov s_k a y_k tak, aby boli splnené požiadavky kladené na maticu H_{k+1} a teda pre maticu ΔH_k musí platiť :

$$(\Delta H_k) y_k = s_k - H_k y_k$$

$$(\Delta H_k)r = 0$$

Pre odštartovanie kvázinewtonovskej metódy tak potrebujeme okrem začiatočného bodu x_0 aj začiatočnú maticu H_0, ktorá sa spravidla volí ako $H_0 = I$.

KAPITOLA 3

TENZOROVÁ METÓDA RIEŠENIA ÚLOHY (1.1)

3.1. Základné definície operácie s tenzormi.

DEFINÍCIA 1 Tenzorom T tretieho stupňa rozumieme objekt s usporiadanými n^3 prvkami T_{ijk} (i=1,...,n; j=1,...,n; k=1,...n).

Poznamenajme, že analogicky ako matica (čo je vlastne tenzor druhého stupňa) obsahuje riadky a stĺpce, tenzor tretieho stupňa možno interpretovať ako n-ticu po vrstvách usporiadaných matíc, pričom existujú tri možné spôsoby tvorby "vrstiev" podľa jednotlivých indexov.

Pomocou tenzora tretieho stupňa môžeme utvoriť "trilineárnu" funkciu premenných $u, v, w \in \Re^n$:

$$t(u, v, w) = T \cdot uvw = \sum_{i=1}^{n}\sum_{j=1}^{n}\sum_{k=1}^{n} T_{ijk}u_i v_j w_k,$$

alebo kubickú formu premennej $x \in \Re^n$:

$$t(x, x, x) = T \cdot x^3 = \sum_{i=1}^{n}\sum_{j=1}^{n}\sum_{k=1}^{n} T_{ijk}x_i x_j x_k.$$

Súčin tenzora tretieho stupňa s vektorom $u \in \Re^n$ dáva maticu :

$$(T \cdot u)_{j,k} = \sum_{i=1}^{n} T_{ijk}u_i, \quad j = 1, ..., n; k = 1, ..., n.$$

Súčin tenzora T tretieho stupňa s maticou A dáva vektor :

$$(T \cdot A)_k = \sum_{i=1}^{n} \sum_{j=1}^{n} T_{ijk} A_{ij}, \;\; k = 1, ..., n.$$

Analogicky súčin tenzora T tretieho stupňa s dvoma vektormi u, $v \in \Re^n$ dáva vektor :

$$(T \cdot uv)_k = \sum_{i=1}^{n} \sum_{j=1}^{n} T_{ijk} u_i v_j, \;\; k = 1, ..., n.$$

DEFINÍCIA 2 Tenzorom V štvrtého stupňa rozumieme objekt s usporiadanými n^4 prvkami V_{ijkl} (i=1,...,n; j=1,...,n; k=1,...n; l=1,...,n).

Analogicky ako vyššie, môžeme si tenzor štvrtého stupňa predstaviť ako n-ticu po vrstvách usporiadaných tenzorov tretieho stupňa, pričom existujú štyri možné spôsoby tvorby "vrstiev" podľa jednotlivých indexov.

Pomocou tenzora štvrtého stupňa môžeme utvoriť "štyrilineárnu" funkciu premenných $r, u, v, w \in \Re^n$:

$$v(r, u, v, w) = V \cdot ruvw = \sum_{i=1}^{n} \sum_{j=1}^{n} \sum_{k=1}^{n} \sum_{l=1}^{n} V_{ijkl} r_i u_j v_k w_l,$$

alebo "bikvadratickú" formu premennej $x \in \Re^n$:

$$v(x, x, x, x) = V \cdot x^4 = \sum_{i=1}^{n} \sum_{j=1}^{n} \sum_{k=1}^{n} \sum_{l=1}^{n} V_{ijkl} x_i x_j x_k x_l.$$

Súčin tenzora štvrtého stupňa V s vektorom $u \in \Re^n$ dáva tenzor tretieho stupňa :

$$(V \cdot u)_{jkl} = \sum_{i=1}^{n} V_{ijkl} u_i, \;\; j = 1, ..., n; k = 1, ..., n; l = 1, ..., n.$$

Súčin tenzora štvrtého stupňa V s maticou dáva maticu :

$$(V \cdot A)_{kl} = \sum_{i=1}^{n} \sum_{j=1}^{n} V_{ijkl} A_{ij}, \;\; k = 1, ..., n; l = 1, ..., n.$$

Súčin tenzora štvrtého stupňa V s tenzorom tretieho stupňa dáva vektor :

$$(V \cdot T)_l = \sum_{i=1}^{n}\sum_{j=1}^{n}\sum_{k=1}^{n} V_{ijkl}T_{ijk}, \;\; l = 1, ..., n.$$

A analogicky súčin tenzora V s troma vektormi $u, v, w \in \Re^n$ dáva vektor :

$$(V \cdot uvw)_l = \sum_{i=1}^{n}\sum_{j=1}^{n}\sum_{k=1}^{n} V_{ijkl}u_i v_j w_k, \;\; l = 1, ..., n.$$

V zmysle týchto operácií vlastne definujeme "tenzorový zápis" v snahe zjednodušiť algebraický zápis pri práci s tenzormi. Na záver ešte definujme operátor "tenzorového súčinu" v snahe vyhnúť sa zložkovému zápisu. Potom tenzor štvrtého stupňa hodnosti jedna určený po zložkách :

$$v_{ijkl} = s_i s_j s_k s_l$$
$$\big(i = 1, ..., n; \;\; j = 1, ..., n; \;\; k = 1, ..., n; \;\; l = 1, ..., n\big),$$

kde $s \in \Re^n$, zapíšeme takto :

$$V = s \otimes s \otimes s \otimes s.$$

3.2. Tenzorový model (štvrtého rádu).

Predpokladajme, že $f \in C^4$, čo umožňuje utvoriť Taylorov polynóm štvrtého stupňa v okolí bodu $x_k \in \Re^n$:

$$M_T(x_k + d) = f(x_k) + g_k \cdot d + \frac{1}{2}H_k \cdot d^2 + \frac{1}{6}T_k \cdot d^3 + \frac{1}{24}V_k \cdot d^4, \tag{3.2}$$

kde $d \in \Re^n$ je smer, x_k je aktuálny bod, g_k a H_k sú gradient a Hessova matica, a tenzorové členy v x_k, $T_k \in \Re^{n\times n\times n}$ a $V_k \in \Re^{n\times n\times n\times n}$ sú symetrické. Tenzorovému zápisu, ako je uvedený vyššie, sme prispôsobili aj zápis členov prvého a druhého rádu a teda $g_k \cdot d$ a $H_k \cdot d^2$ znamenajú $g_k^T d$ a $d^T H_k d$.

Podobne ako pri štandardných metódach, nasledujúca, (k+1)-iterácia je určená ako minimum funkcie M_T. Pri tenzorovej metóde môžeme vidieť určitú analógiu s kvázi-newtonovskými metódami, ktoré v snahe vyhnúť sa náročnému výpočtu (n^2 funkčných hodnôt) presnej Hessovej matice používajú len jej aproximáciu. Výpočet tretej derivácie (n^3 funkčných hodnôt) a štvrtej derivácie (n^4 funkčných hodnôt) v tenzorovej metóde by bolo priveľmi náročné na čas aj pamäť. Práve to je dôvodom pre aproximáciu týchto členov "vhodnými" tenzormi. Tieto sa snažíme určiť tak, aby ich vnútorná štruktúra umožňovala dostatočné zjednodušenie pri minimalizácii funkcie M_T. Ako ukážeme ďalej, je možné vybrať tenzorové členy hodnosti dva a jedna.

3.3. Bouarichova metóda.

V tejto časti naznačíme odvodenie tenzorového modelu. Výber tenzorov T_k a V_k je vedený požiadavkou na jednoduchosť členov $T_k \cdot d_k^3$ a $V_k \cdot d_k^4$. Tieto tenzorové členy vyberáme tak, aby model interpoloval cez funkčné hodnoty a hodnoty gradientu v bodoch x_{k-1} a x_k. Interpolačné podmienky v predchádzajúcom bode x_{k-1} sú dané :

$$f(x_{k-1}) = f(x_k) + g_k \cdot s + \frac{1}{2} H_k \cdot s^2 + \frac{1}{6} T_k \cdot s^3 + \frac{1}{24} V_k \cdot s^4 \tag{3.3}$$

a

$$g_{k-1} = g_k + H_k \cdot s + \frac{1}{2} T_k \cdot s^2 + \frac{1}{6} V_k \cdot s^3, \tag{3.4}$$

kde

$$s = x_{k-1} - x_k$$

je smer použitý v predchádzajúcej iterácii. Ukážeme, že interpolačné podmienky (3.3) a (3.4) jednoznačne určujú skalárne hodnoty členov $T_k \cdot s^3$ a $V_k \cdot s^4$. Prenásobením (3.4) sprava vektorom s, dostávame :

$$g_{k-1} \cdot s = g_k \cdot s + H_k \cdot s^2 + \frac{1}{2} T_k \cdot s^3 + \frac{1}{6} V_k \cdot s^4 \tag{3.5}$$

Kôli prehľadnosti zavedieme zjednodušenú symboliku a skalárne hodnoty skúmaných členov označíme symbolicky $\alpha, \beta \in \Re$, t.j.

$$\alpha = T_k \cdot s^3, \quad \beta = V_k \cdot s^4$$

Potom z (3.3) a (3.4) dostávame systém dvoch lineárnych rovníc o dvoch neznámych α a β :

$$\frac{1}{2}\alpha + \frac{1}{6}\beta = \mu \tag{3.6}$$

$$\frac{1}{6}\alpha + \frac{1}{24}\beta = \nu, \tag{3.7}$$

kde $\mu, \nu \in \Re$ sú definované ako

$$\mu = g_{k-1} \cdot s - g_k \cdot s - H_k \cdot s^2$$

$$\nu = f(x_{k-1}) - f(x_k) - g_k \cdot s - \frac{1}{2} H_k \cdot s^2$$

Systém (3.6)-(3.7) je regulárny a preto hodnoty α, β sú jednoznačne určené. Nemáme však dostatok informácií na určenie samotných tenzorov T_k a V_k. Schnabel a Chow [2] vybrali T_k a V_k najmenšie v zmysle Frobeniovej normy a teda riešili úlohu :

$$\min_{V_k \in \Re^{n \times n \times n \times n}} \|V_k\|_F \tag{3.8}$$
$$\text{za podmienky } V_k \cdot s^4 = \beta \text{ a } V_k \text{ je symetrický tenzor,}$$

ktorá už obsahuje dostatok podmienok na jednoznačné určenie tenzora V_k. Podobne potom riešením úlohy :

$$\min_{T_k \in \Re^{n \times n \times n}} \|T_k\|_F \tag{3.9}$$
$$\text{za podmienky } T_k \cdot s^2 = a, \text{ kde } T_k \text{ je symetrický tenzor}$$
$$\text{pričom } \quad a = 2(\nabla f(x_{k-1}) - \nabla f(x_k) - \nabla^2 f(x_k)s - \frac{1}{6} V_k s^3)$$

jednoznačne určíme tenzor T_k. Obe úlohy prepíšeme do štandardného tvaru úlohy na viazaný extrém. Potom úloha (3.8) je ekvivalentná s úlohou

$$Min\Big\{\frac{1}{2}\|V_k\|_F^2 = \frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n}\sum_{k=1}^{n}\sum_{l=1}^{n} v_{ijkl}^2 \;\Big|\; \sum_{i=1}^{n}\sum_{j=1}^{n}\sum_{k=1}^{n}\sum_{l=1}^{n} v_{ijkl} s_i s_j s_k s_l = \beta\Big\}, \tag{3.10}$$

pričom Lagrangeova funkcia pre úlohu (3.10) vyzerá takto :

$$L(v,\gamma) = \frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n}\sum_{k=1}^{n}\sum_{l=1}^{n} v_{ijkl}^2 - \gamma\Big(\sum_{i=1}^{n}\sum_{j=1}^{n}\sum_{k=1}^{n}\sum_{l=1}^{n} v_{ijkl}s_i s_j s_k s_l - \beta\Big),$$

kde γ je Lagrangeov multiplikátor. Položením praciálnych derivácií funkcie $L(v,\gamma)$ podľa všetkých premenných rovnými nule, dostávame nutné podmienky pre extrém :

$$\frac{\partial L}{\partial v_{ijkl}} = v_{ijkl} - \gamma s_i s_j s_k s_l = 0 \tag{3.11}$$

$$\big(i = 1,...,n, \quad j = 1,...,n, \quad k = 1,...,n, \quad l = 1,...,n\big)$$

$$V_k \cdot s^4 = \beta. \tag{3.12}$$

Úpravou podmienok (3.11) a využitím tenzorového zápisu dostaneme :

$$V_k = \gamma s \otimes s \otimes s \otimes s,$$

čo po dosadení do podmienky (3.12) vedie k vzťahu :

$$\big[\gamma s \otimes s \otimes s \otimes s\big] \cdot s^4 = \beta.$$

Vzhľadom na vnútornú štruktúru symbolického zápisu spojíme členy obsahujúce zložky vektora s do skalárneho súčinu a z toho potom vyjadríme Lagrangeov multiplikátor γ :

$$\gamma = \frac{\beta}{(s^T s)^4}. \tag{3.13}$$

Potom riešením úlohy (3.8) je :

$$V_k = \gamma(s \otimes s \otimes s \otimes s) \text{ , kde } \gamma = \frac{\beta}{(s^T s)^4}$$

a tenzor štvrtého stupňa $V_k \in \Re^{nxnxnxn}$ má hodnosť jedna.

Tenzor tretieho stupňa T_k odvodíme rovnakým spôsobom, aj keď samotná realizácia bude o čosi zložitejšia, keďže ide o tenzor hodnosti 2. Podobne ako v prípade tenzora V_k môžeme zapísať úlohu ekvivalentnú s úlohou (3.9) ako :

$$Min\Big\{\frac{1}{2}\|T_k\|_F^2 = \frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n}\sum_{k=1}^{n} t_{ijk}^2 \;\Big|\; \sum_{i=1}^{n}\sum_{j=1}^{n} t_{ijk}s_i s_j = a_k, \text{ k=1,...,n } \Big\} \tag{3.14}$$

Lagrangeova funkcia pre úlohu (3.14) potom vyzerá takto :

$$L(t,b) = \frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n}\sum_{k=1}^{n} t_{ijk}^2 - \sum_{k=1}^{n} b_k\Big(\sum_{i=1}^{n}\sum_{j=1}^{n} t_{ijk}s_i s_j - a_k\Big),$$

kde b je vektor Lagrangeových multiplikátorov. Položením parciálnych derivácií funkcie $L(t,b)$ rovnými nule dostávame nutné podmienky pre extrém :

$$\frac{\partial L}{\partial t_{ijk}} = t_{ijk} - b_k s_i s_j = 0,\ \text{k=1,...,n} \tag{3.15}$$

$$T_k \cdot s^2 = a. \tag{3.16}$$

Potom využitím tenzorového zápisu a prihliadnuc na symetriu T_k dostávame z (3.15) vzťah pre tenzor T_k :

$$T_k = b \otimes s \otimes s + s \otimes b \otimes s + s \otimes s \otimes b \tag{3.17}$$

Keďže vektor s je známy, ostáva ešte určiť vektor b. Po dosadení (3.17) do (3.16) dostávame

$$\big[b \otimes s \otimes s + s \otimes b \otimes s + s \otimes s \otimes b\big]s^2 = a$$

Opäť môžeme prihliadnuc na vnútornú štruktúru tenzorového súčinu "spojiť" zložky vektorov s a b do skalárnych súčinov, čo potom vedie k :

$$b(s^T s)^2 + 2(s^T s)(b^T s)s = a \tag{3.18}$$

Keďže neznámy je vektor b, prenásobíme (3.18) vektorom s,čo vedie k :

$$b^T s = \frac{a^T s}{3(s^T s)^2}. \tag{3.19}$$

Po dosadení (3.19) do (3.18) a niekoľkých úpravách už získavame samotný vektor b :

$$b = \frac{3(s^T s)a - 2(a^T s)s}{3(s^T s)^2}.$$

Potom riešením úlohy (3.9) je

$$T_k = b \otimes s \otimes s + s \otimes b \otimes s + s \otimes s \otimes b,$$

kde tenzor tretieho stupňa $T_k \in \Re^{n\times n\times n}$ má hodnosť dva a $b \in \Re^n$ je jednoznačne určený vektor, daný

$$b = \frac{3a(s^T s) - 2s(s^T a)}{3(s^T s)^3}$$

Nízka hodnosť oboch tenzorových členov je kľúčom k efektívnemu formovaniu, skladovaniu a riešeniu modelu. Celý proces formovania tenzorového modelu si vyžaduje iba $O(n^2)$ operácií, čo je však spôsobené nevyhnutnosťou rátať Hessovu maticu. Výpočet tenzorových členov T_k a V_k si vyžaduje iba $O(n)$ operácií. Pamäť potrebná na formovanie a skladovanie modelu je iba 3n.

Po dosadení tenzorov T_k a V_k do (3.2) teda dostávame model

$$M_T(x_k + d) = f(x_k) + g_k \cdot d + \frac{1}{2}H_k \cdot d^2 + \frac{1}{2}(b^T d)(s^T d)^2 + \frac{\gamma}{24}(s^T d)^4 \tag{3.20}$$

Ďalej budeme minimalizovať funkciu M_T danú vzťahom (3.20), avšak musíme rozlišovať dva prípady a to prípad regulárnej Hessovej matice a prípad, keď Hessova matica je singulárna.

3.4. Výpočet tenzorového smeru v prípade regulárnej Hessovej matice.

Ukážeme, že v prípade, keď Hessova matica je regulárna, minimalizácia modelu štvrtého rádu (3.20) spočíva vo vyriešení kubickej rovnice jednej premennej a v riešení troch systémov lineárnych rovníc s rovnakou maticou H_k. Nutnou podmienkou pre d ako lokálne minimum funkcie M_T je, že derivácia tenzorového modelu (3.20) podľa d sa musí rovnať nule, teda :

$$\nabla M_T(x_k + d) = g_k + H_k d + (b^T d)(s^T d)s + \frac{1}{2}(s^T d)^2 b + \frac{\gamma}{6}(s^T d)^3 s = 0,$$

čo vedie k

$$d = -H_k^{-1}\left(g_k + (b^T d)(s^T d)s + \frac{1}{2}(s^T d)^2 b + \frac{\gamma}{6}(s^T d)^3 s\right), \tag{3.21}$$

pričom vektor d vystupuje aj na pravej strane "skrytý" v skalárnych súčinoch s vektormi s a b. Preto najprv prenásobíme (3.21) zľava vektorom s^T, čím dostaneme kubickú rovnicu s neznámymi $\psi = s^T d$ a $\omega = b^T d$, :

$$s^T H^{-1} g + \psi + s^T H^{-1} s\omega\psi + \frac{1}{2} s^T H^{-1} b\psi^2 + \frac{\gamma}{6} s^T H^{-1} s\psi^3 = 0 \tag{3.22}$$

Potom prenásobíme (3.21) zľava vektorom b^T, čím dostaneme druhú kubickú rovnicu s neznámymi ψ a ω :

$$b^T H^{-1} g + \psi + b^T H^{-1} s\omega\psi + \frac{1}{2} b^T H^{-1} b\psi^2 + \frac{\gamma}{6} b^T H^{-1} s\psi^3 = 0 \tag{3.23}$$

Máme teda systém dvoch kubických rovníc o dvoch neznámych ψ a ω. Ďalej ukážeme, že riešenia tohto systému môžeme nájsť riešením jedinej kubickej rovnice s neznámou ψ. Kvôli prehľadnosti označíme členy obsahujúce maticu H^{-1} ako $\alpha = s^T H^{-1} g$, $\delta = s^T H^{-1} b$, $\sigma = s^T H^{-1} s$, $\eta = b^T H^{-1} g$ a $\beta = b^T H^{-1} b$. V novej symbolike sústava kubických rovníc s neznámymi ψ a ω vyzerá takto :

$$\alpha + \psi + \sigma\omega\psi + \frac{1}{2}\delta\psi^2 + \frac{\gamma}{6}\sigma\psi^3 = 0 \tag{3.22'}$$

$$\eta + \psi + \delta\omega\psi + \frac{1}{2}\beta\psi^2 + \frac{\gamma}{6}\delta s\psi^3 = 0 \tag{3.23'}$$

Pričom konkrétne hodnoty skalárov α, δ, σ, η a β získame riešením dvoch sústav lineárnych rovníc s rovnakou maticou. Označme $y_1 = H^{-1}s$, $y_2 = H^{-1}b$ a $y_3 = H^{-1}g$. Zavedením premennej y_3 sme do systému dvoch sústav lineárnych rovníc pridali tretiu sústavu, čo je predprípravou na riešenie rovnice (3.21). Potom nám ostáva vyriešiť nasledujúce tri sústavy :

$$\begin{aligned} Hy_1 &= s \\ Hy_2 &= b \\ Hy_3 &= g. \end{aligned} \tag{3.24}$$

Potom z riešení y_1 a y_2 systému (3.24) získame skaláre α, δ, σ, η a β takto :

$$\alpha = y_1^T g \qquad \delta = y_1^T b$$
$$\sigma = y_1^T s \qquad \eta = y_2^T g$$
$$\beta = y_2^T b$$

Poznamenajme ešte, že systém (3.24) môžeme efektívne riešiť Choleského rozkladom matice H, $H = LL^T$, čo vedie k postupnému riešeniu sústav :

$$\begin{aligned} Lz_1 &= s & L^T y_1 &= z_1 \\ Lz_2 &= b & L^T y_2 &= z_2 \\ Lz_3 &= g & L^T y_3 &= z_3. \end{aligned}$$

Výhodou takéhoto postupu je, že môžeme naraz riešiť v prvej fáze prvú trojicu a v druhej fáze druhú trojicu sústav. Teraz, keď máme vypočítané hodnoty skalárov α, δ, σ, η a β, môžeme pristúpiť k riešeniu samotného systému kubických rovníc. Predpokladáme, že skalár $\psi \neq 0$, pretože inak by došlo k redukcii tenzorového modelu na Newtonov model. Ďalšie riešenie rozčleníme do dvoch prípadov.

V prípade, keď $\boxed{\sigma = 0}$,

rovnica (3.22') sa redukuje na kvadratickú rovnicu jednej premennej ψ a teda platí :

$$\psi = \frac{-1 \pm \sqrt{1 - 2\alpha\delta}}{\delta}.$$

Potom neznámu omega vypočítame z (3.23') :

$$\omega = \frac{1}{1 + \delta\psi}(-\eta - \frac{1}{2}\beta\psi^2 + \frac{\gamma}{6}\delta\psi^3).$$

V tomto prípade existuje reálne riešenie ψ, ak $1 - 2\alpha\delta \geq 0$, a hodnota ω je definovaná, ak $1 + \delta\psi \neq 0$.

V prípade, keď $\boxed{\sigma \neq 0}$,

sa nevyhneme riešeniu kubickej rovnice. Najprv vyjadríme ω z (3.22') :

$$\omega = -\frac{\alpha + \psi + \frac{1}{2}\delta\psi^2 + \frac{\gamma}{6}\sigma\psi^3}{\sigma\psi}. \tag{3.25}$$

A potom po dosadení výrazu pre ω do (3.23') dostávame kubickú rovnicu s jednou neznámou ψ, ktorá vyzerá takto :

$$-\alpha + (\gamma\sigma - 2\alpha\delta - 1)\psi - \frac{3}{2}\delta\psi^2 + (\frac{1}{2}\sigma\beta - \frac{\gamma}{6} - \frac{1}{2}\delta^2)\psi^3 = 0.$$

Po vypočítaní koreňov kubickej rovnice, dosadíme hodnoty ψ do (3.25) a vypočítame zodpovedajúce hodnoty ω Potom jednoducho dosadíme skaláre ψ a ω do (3.21) a vypočítame im zodpovedajúce smery d. Z týchto smerov potom vyberáme ten, ktorý je najbližšie k aktuálnemu bodu x_k v zmysle Euklidovej normy, pričom však musí garantovať spádovosť v smere od x_k k $x_k + d$. V prípade, keď nebolo nájdené minimum tenzorového modelu, použijeme pre aktuálnu iteráciu Newtonovský smer. Pozrime sa teraz bližšie na správanie sa kubickej rovnice.

Analýza kubiky

Máme kubickú rovnicu, ktorá vo všeobecnosti vyzerá takto :

$$f(x) = a_3x^3 + a_2x^2 + a_1x + a_0 = 0.$$

Položením gradientu funkcie $f(x)$ rovnému nule, vypočítame stacionárne body. Nech teda :

$$\nabla f(x) = 3a_3x^2 + 2a_2x + a_1 = 0.$$

Ďalej riešime kvadratickú rovnicu, pričom môžu nastať tri prípady :

1.Diskriminant $D < 0$ − neexistujú reálne stacionárne body a kubika má jeden koreň

2.Diskriminant $D = 0$ − existuje jeden stacionárny bod a kubika má jeden koreň

3.Diskriminant $D > 0$ − existujú dva stacionárne body a kubika má jeden koreň alebo tri korene

V prípadoch $D < 0$ a $D = 0$ sme jediný koreň určili Newtonovou metódou na riešenie nelineárnych rovníc s presnosťou 10^{-5}.

V prípade, keď existujú dva stacionárne body, x_{min} - bod minima a x_{max} - bod maxima, sme vypočítali funkčné hodnoty $f(x_{min})$ a $f(x_{max})$.

Ak $f(x_{min}) > 0$ alebo $f(x_{max}) < 0$, t.j. funkčné hodnoty v x_{min} a v x_{max} majú rovnaké znamienko, existuje jediný koreň a ten sme určili rovnako, ako v prípadoch 1 a 2.

Ak $f(x_{min}) < 0$ a $f(x_{max}) > 0$, t.j. kubika medzi x_{min} a x_{max} mení znamienko, existujú tri korene. Prostredný koreň (medzi x_{min} a x_{max}) sme vypočítali Newtonovou metódou na riešenie nelineárnych rovníc so štartovacím bodom $\frac{x_{min}+x_{max}}{2}$ a presnosťou 10^{-5} a ďalšie dva sme určili analyticky, ako korene kvadratickej rovnice, ktorá vznikla predelením pôvodnej kubickej rovnice koreňovým činiteľom.

3.5. Výpočet tenzorového smeru v prípade singulárnej Hessovej matice.

V prípade, keď Hessova matica je singulárna, by postup ako v predchádzajúcom prípade neviedol k jednoznačnému riešeniu. Keďže naším cieľom bolo porovnať tenzorovú a modifikovanú Newtonovu metódu, týmto prípadom sme sa podrobnejšie nezaoberali. Pre úplnosť však uvádzame aspoň hlavnú myšlienku riešenia modelu v tomto prípade. Ukázalo sa [1], že je vhodné transformovať model (3.20) nasledujúcim spôsobom. Nech $d = \hat{d} + \delta$, pre fixované $\hat{d}$, kde δ je nová neznáma. Dosadením tohto výrazu do (3.20) dostávame nasledujúci model, ktorý je funkciou δ :

$$M_T(x_k + d) = f(x_k) + g_k\hat{d} + \frac{1}{2}H_c\hat{d}^2 + \frac{1}{2}(b^T\hat{d})(s^T\hat{d})^2 + \frac{\gamma}{24}(s^T\hat{d})^4 +$$
$$+(g_k + H_k\hat{d} + (b^T\hat{d})(s^T\hat{d})s + \frac{1}{2}(s^T\hat{d})^2 b + \frac{\gamma}{6}(s^T\hat{d})^3 s)\delta +$$
$$+\frac{1}{2}(H_k + (b^T\hat{d})ss^T + \frac{\gamma}{2}(s^T\hat{d})^2 ss^T)\delta^2 +$$
$$+(s^T\hat{d})(b^T\delta)(s^T\delta) + \frac{1}{2}(b^T\delta)(s^T\delta)^2 + \frac{\gamma}{6}(s^T\hat{d})(s^T\delta)^3 + \frac{\gamma}{24}(s^T\delta)^4$$

Ak označíme $\hat{\psi} = s^T\hat{d}$, $\hat{\omega} = b^T\hat{d}$, $\hat{g} = g_k + H_k\hat{d} + \hat{\omega}\hat{\psi}s + \frac{1}{2}\hat{\psi}^2 b + \frac{\gamma}{6}\hat{\beta}^3 s$, $c = b^T\hat{d} + \frac{\gamma}{2}(s^T\hat{d})^2$ a $\hat{H} = H_k + css^T$, dostávame modifikovaný model :

$$M_T(x_k + d) = M_T(x_k + \hat{d}) + \hat{g}\delta +$$
$$+\frac{1}{2}\hat{H}\delta^2 + \hat{\psi}(b^T\delta)(s^T\delta) + \frac{1}{2}(b^T\delta)(s^T\delta)^2 + \frac{\gamma}{6}\hat{\psi}(s^T\delta)^3 + \frac{\gamma}{24}(s^T\delta)^4$$

Výhodou tejto transformácie je, že matica $\hat{H}$ je pravdepodobne regulárna, ak hodnosť matice H_k je aspoň $n-1$. Ďalej už postupujeme podobne ako v kapitole 3.4 až po výber vhodného δ. Z toho potom tenzorový smer $d = \hat{d} + \delta$, pričom vhodnou voľbou $\hat{d}$ je smer použitý v predchádzajúcej iterácii. Tento postup je však použiteľný iba v prípade, keď Hessova matica má hodnosť $n-1$. V praxi sa taktiež ukázalo [2], že práve v tomto prípade, je konvergencia tenzorovej metódy lepšia, ako lineárna konvergencia Newtonovej metódy. V prípadoch, keď hodnosť Hessovej matice je menšia ako $n-1$, tenzorová metóda nemá dostatok informácií na dôkaz lepšej ako lineárnej konvergencie. Pre úplnosť ešte uvádzame dôkaz regularity matice $\hat{H} = H_k + css^T$ [1]. *Lemma 1* formuluje nutnú a postačujúcu podmienku pre regularitu matice $\hat{H}$.

Lemma 1 *Nech $H \in \Re^{n\times n}$, $s \in \Re^n$, $c \neq 0$. Matica $H + css^T$ je regulárna vtedy a len vtedy keď matica*

$$M = \begin{bmatrix} H & cs \\ cs^T & -c \end{bmatrix}$$

je regulárna.

Dôkaz: Ukážeme, že existuje nenulový vektor $v \in \Re^n$, pre ktorý $(H + css^T)v = 0$ vtedy a len vtedy keď existujú vektory $\overline{v} \in \Re^b$ a $w \in \Re^n$, pre ktoré platí :

$$\begin{bmatrix} H & cs \\ cs^T & -c \end{bmatrix} \begin{bmatrix} \overline{v} \\ w \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \end{bmatrix}, \text{ pričom } \begin{bmatrix} \overline{v} \\ w \end{bmatrix} \neq \begin{bmatrix} 0 \\ 0 \end{bmatrix}. \tag{3.26}$$

Predpokladajme najprv, že $(H + css^T)v = 0$, $v \neq 0$. Potom vektory $\overline{v} = v$ a $w = s^T v$ spĺňajú podmienku (3.26). Opačne, ak existujú vektory $(\overline{v}, w)$, ktoré spĺňajú podmienku (3.26), tak $s^T\overline{v} = w$ a teda $(H + css^T)\overline{v} = 0$ a $\overline{v} \neq 0$, inak by $w = 0$, čo je spor s podmienkou (3.26). □

Dôsledok *Nech $H \in \Re^{n\times n}$, $s \in \Re^n$. Ak je matica $H + css^T$ regulárna, potom matica $[\,H \quad cs\,]$ má plnú riadkovú hodnosť.*

Lemma 2 *Nech $H \in \Re^{n\times n}$, pričom hodnosť H je $n-1$, $s \in \Re^n$. Matica $H + css^T$ je regulárna vtedy a len vtedy, keď matica $[\,H \quad cs\,]$ má plnú riadkovú hodnosť.*

Dôkaz: Predpokladajme teraz, že matica $[\,H \quad cs\,]$ má plnú riadkovú hodnosť. Keďže $\text{hod}(H) = n-1$, môžeme maticu H napísať ako $H = H_1 H_2^T$, kde matice $H_1, H_2 \in \Re^{n\times(n-1)}$ majú plnú stĺpcovú hodnosť. Z toho, že matica $[\,H \quad cs\,]$ má plnú riadkovú hodnosť vyplýva, že

$$\left(v^T H = 0 \text{ a } v^T s = 0\right) \Rightarrow v = 0 \tag{3.27}$$

Ďalej z toho, že $H = H_1 H_2^T$, pričom matica H_2 má plnú stĺpcovú hodnosť vyplýva, že (3.27) je ekvivalentné s:

$$\left(v^T H_1 = 0 \text{ a } v^T s = 0\right) \Rightarrow v = 0$$

A teda matica $[\,H_1 \quad cs\,]$ je regulárna. Analogicky sa dá ukázať, že aj matica $[\,H_2 \quad s\,]$ je regulárna. Preto

$$[\,H_1 \quad cs\,]\begin{bmatrix} H_2^T \\ s^T \end{bmatrix} = H_1 H_2^T + css^T = H + css^T$$

je regulárna.

Opačná implikácia je priamym dôsledkom *Lemmy 1.* □

3.6. Optimálna dĺžka kroku.

Algoritmus výpočtu nového bodu x_+

Nech
x_k je aktuálny bod
d_t je tenzorový smer
d_n je Newtonovský smer
g_k, f_k sú aktuálny gradient a funkčná hodnota

sklon=$g^T d_t$
$\alpha = 10^{-4}$
$x_+^t = x_k + d_t$
$f_p = f(x_+^t)$

if (bolo nájdené minimum tenzorového modelu) **then**
  if($f_p < f_k + \alpha.$*sklon*) **then** $x_+ = x_+^t$
    else
      1) Nájdi x_+^n v Newtonovskom smere d_n s optimálnou dĺžkou kroku
      2) Nájdi x_+^t v tenzorovskom smere d_t s optimálnou dĺžkou kroku
        if ($f(x_+^n) < f(x_+^t)$) **then** $x_+ = x_+^n$
                else $x_+ = x_+^t$
        endif
  endif
  else
      1) Nájdi x_+^n v Newtonovskom smere d_n s optimálnou dĺžkou kroku
      2) $x_+ = x_+^n$
endif

Vyššie uvedená schéma zobrazuje algoritmus výpočtu nového bodu x_+. Ako sme už spomenuli, ani samotná Newtonova metóda nemusí konvergovať s jednotkovou dĺžkou kroku. Podobne aj v tenzorovej metóde sa ukazuje vhodné optimalizovať dĺžku kroku. V každej iterácii tak po výbere vhodného smeru, najprv testujeme plný krok, t.j. $\lambda = 1$.

Tento je použitý v prípade, že poskytuje dostatočný pokles účelovej funkcie. V opačnom prípade hľadáme metódou minimalizácie na lúči optimálne dĺžky krokov v tenzorovom aj Newtonovom smere. Potom sa použije ten smer, ktorý vykazuje väčší pokles účelovej funkcie.

3.7. Metóda Zlatého rezu.

Predpokladajme teda, že máme k dispozícii bod x_k a smerový vektor $0 \neq d \in \Re^n$. Podobne ako v modifikovanej Newtonovej metóde, v tomto smere teraz určíme dĺžku kroku, t.j. smer λ_k a to tak, že riešime úlohu :

$$Min\{\varphi(\lambda) = f(x_k + \lambda d) \big| \lambda \geq 0\}, \tag{3.28}$$

pričom funkcia φ spĺňa : $\varphi^{'}(0) = d^T g_k < 0$. Na hľadanie optimálnej dĺžky kroku sme sa rozhodli použiť Metódu zlatého rezu. Vstupom pre metódu zlatého rezu je okrem funkcie $\varphi(x)$ aj interval $< a, b >$, ktorý obsahuje hľadané minimum. Potom metódou zlatého rezu minimalizujeme funkciu $\varphi(\lambda)$ na tomto intervale. Spôsob voľby začiatočného intervalu $< a, b >$ pre úlohu (3.28) je popísaný v nasledujúcej schéme.

Algoritmus na určenie intervalu <a,b>

$a=0$
$\lambda_0 = 1$
$\lambda_1 = 2$
$\varphi_0 = \varphi(\lambda_0) = f(x_k + \lambda_0 d)$
$\varphi_1 = \varphi(\lambda_1) = f(x_k + \lambda_1 d)$

while $(\varphi_0 > \varphi_1)$
  $a = \lambda_0$
  $\lambda_0 = \lambda_1$
  $\lambda_1 = 2\lambda_1$
  $\varphi_0 = \varphi_1$
  $\varphi_1 = \varphi(\lambda_1)$
end of while
$b = \lambda_1$

Po určení začiatočného intervalu$< a, b >$ môžeme pristúpiť k samotnej metóde zlatého rezu, ktorej výstupom je optimálna dĺžka kroku λ_k. Myšlienka metódy zlatého rezu spočíva v postupnom skracovaní intervalu $< a, b >$ z pravej alebo ľavej strany, o čom sa rozhoduje na základe porovnávania funkčných hodnôt. Algoritmus končí pri skrátení intervalu na požadovanú presnosť a ako minimum vracia ľavý okraj posledného intervalu.

Algoritmus metódy zlatého rezu na intervale <a,b>

VSTUP: funkcia $\varphi(x)$ definovaná na intervale <a,b>
požadovaná presnosť $\varepsilon>0$

VÝPOČET: 0) $t=\frac{\sqrt{5}+1}{2}$, $z_1=2-t$, $z_2=t-1$

1) $c_1=a+z_1(b-a)$ $\varphi_1=\varphi(c_1)=f(x_k+c_1 d)$
$c_2=a+z_2(b-a)$ $\varphi_2=\varphi(c_2)=f(x_k+c_2 d)$

2) **if** $(\varphi_1<\varphi_2)$ **then** GOTO 4

3) (prípad $\varphi_1\leq\varphi_2$)
Položíme $a=c_1$ $c_1=c_2$ $\varphi_1=\varphi_2$
if $((b-a)<\varepsilon)$ **then** GOTO 5
else $c_2=a+z_2(b-a)$
$\varphi_2=\varphi(c_2)$
GOTO 2

4) (prípad $\varphi_1<\varphi_2$)
Položíme $b=c_2$ $c_2=c_1$ $\varphi_2=\varphi_1$
if $((b-a)<\varepsilon)$ **then** GOTO 5
else $c_1=a+z_1(b-a)$
$\varphi_1=\varphi(c_1)$
GOTO 2

VÝSTUP: 5) $x_0=c_1$ $\varphi_0=\varphi_1$
6) STOP

Rýchlosť redukcie intervalu neurčitosti metódou zlatého rezu je $d_n=\frac{(b-a)}{t^{n-1}}$, kde n je počet výpočtov funkčných hodnôt $\varphi(x)$ a $t=\frac{\sqrt{5}+1}{2}$.

NUMERICKÉ EXPERIMENTY

4.1. Voľba testovacích funkcií.

Cieľom predkladanej práce je aj experimentálne porovnanie tezorovej metódy s modifikovanou Newtonovou metódou. Možnosti výberu testovacích funkcií sú potom zúžené predpokladmi oboch metód o minimalizovaných funkciách. Napríklad tenzorová metóda používa na lokálnu aproximáciu minimalizovanej funkcie v okolí riešenia model štvrtého rádu oproti modelu druhého rádu v Newtonovej metóde. Preto aby bolo možné použiť obidve metódy, testovacia funkcia musí byť triedy C^4. Newtonova metóda predpokladá regularitu Hessovaj matice, preto sa aj pri tezorovej metóde zameriame len na prípad regulárnej Hessovej matice. Ďalej požadujeme, aby funkcia $f(x)$ bola konvexná, čo zaručí, že nájdené lokálne minimum bude zároveň globálnym minimom funkcie $f(x)$. Dôležité je, aby sme mohli ľahko meniť jednak konkrétne parametre funkcie $f(x)$, ale aj rozmer úlohy n.

Bikvadratická funkcia

Vzhľadom na vyššie uvedené požiadavky sme sa rozhodli otestovať tenzorovú metódu na bikvadratickej funkcii tvaru :

$$f(x) = \frac{1}{4\pi}\left(x^T G_1 x\right)^2 + \frac{1}{2}\left(x^T G_2 x\right) + h^T x, \tag{4.1}$$

kde člen

$$\pi = \|G_1\|_F^2 = \sum_{i=1}^{n}\sum_{j=1}^{n}(g_{ij}^1)^2$$

je normujúcim členom. Keďže chceme, aby Hessova matica funkcie $f(x)$ bola regulárna, volíme matice G_1 a G_2 symetrické, kladne definitné a to nasledovným spôsobom :

$$G = A^T A + Diag$$

kde matica A je matica vygenerovaná generátorom náhodných čísel. Prenásobenie matice A zľava jej transponovanou maticou A^T zaručí symetriu a pripočítanie diagonálnej, kladne definitnej matice zaručí regularitu konečnej matice G. Potom gradient tejto funkcie môžeme symbolicky zapísať takto :

$$\nabla f(x) = \frac{1}{\pi}\big(x^T G_1 x\big) G_1 x + G_2 x + h \tag{4.2}$$

Otázkou ešte ostáva určenie vektora h. Keďže je pre naše ďalšie účely vhodné vopred poznať optimálne riešenie $\hat{x}$, najprv vygenerujeme vektor $\hat{x} \in \Re^n$ a z neho potom dopočítame vektor $h \in \Re^n$ tak, aby bola splnená nutná podmienka pre minimum, t.j. aby $\nabla f(\hat{x}) = 0_n$. Teda vektor h spĺňa :

$$h = -\left[\frac{1}{\pi}\big(\hat{x}^T G_1 \hat{x}\big) G_1 \hat{x} + G_2 \hat{x}\right]$$

Ešte ostáva ukázať, že takáto funkcia je naozaj konvexná. Budeme sa opierať o nasledovnú vetu [4] :

Veta : Nech $f : \Re^n \to \Re$ má spojité druhé parciálne derivácie. Potom f je konvexná $\Leftrightarrow \forall x : \nabla^2 f(x)$ je kladne semidefinitná.

Keďže matica druhých derivácií funkcie $f(x)$ vyzerá takto :

$$\nabla^2 f(x) = \frac{2}{\pi}\big(G_1 x\big)\big(G_1 x\big)^T + \frac{1}{\pi}\big(x^T G_1 x\big) G_1 + G_2$$

pričom matice G_1 a G_2 sú kladne definitné, je zrejmé, že funkcia $f(x)$ definovaná vzťahom (4.1) má dokonca kladne definitnú Hessovu maticu a teda je dokonca rýdzokonvexná. Teda riešime úlohu :

$$Min\left\{ f(x) = \frac{1}{4\pi}\big(x^T G_1 x\big)^2 + \frac{1}{2}\big(x^T G_2 x\big) + h^T x \;\middle|\; x \in \Re^n \right\},$$

kde $\pi = \|G_1\|_F^2$, $G_1 \succ 0$ a $G_2 \succ 0$.

4.2. Metodika experimentu.

Každú úlohu sme najprv vyriešili tenzorovou metódou, potom sme sa vrátili do štartovacieho bodu a riešili sme ju modifikovanou Newtonovou metódou. Úlohy sme riešili v sériách po sto úloh pre rôzne rozmery a rôzne štartovacie vzdialenosti, pričom sme sledovali počet iterácií a čas do dosiahnutia optimálneho riešenia s požadovanou presnosťou u obidvoch metód. Keďže sme dopredu poznali optimálne riešenie, mohli sme sledovať nielen normu gradientu funkcie $f(x)$, ale aj vzdialenosť aktuálneho bodu od optimálneho riešenia. Potom podmienkou pre zastavenie algoritmu je dosiahnutie relatívnej presnosti:

$$\frac{\|x^k - \hat{x}\|_2}{\|\hat{x}\|_2 + 1} < \varepsilon.$$

Kvôli možnosti vrátiť sa k už riešeným príkladom sme na generovanie matíc G_1, G_2 a optimálneho riešenia $\hat{x}$ použili generátor pseudonáhodných čísel. Po vygenerovaní optimálneho riešenia $\hat{x}$ a ďalšieho náhodného bodu y_0 sme určili štartovací bod x_{-1} tak, aby ležal na polpriamke určenej vektorom $y_0 - \hat{x}$ a aby jeho vzdialenosť od bodu $\hat{x}$ bola zadaná štartovacia vzdialenosť ρ. Teda :

$$x_{-1} = \hat{x} + \tau(y_0 - \hat{x}),$$
$$\text{kde } \tau = \frac{\rho}{\sqrt{(y_0 - \hat{x})^T (y_0 - \hat{x})}}$$

Tento prístup umožnil sledovať a porovnávať efektívnosť obidvoch metód pri zadaní rôznych štartovacích vzdialeností.

Považujeme ešte za potrebné poznamenať, že tenzorová metóda má "predprípravnú" nultú iteráciu, v ktorej sa z bodu x_{-1} použitím Newtonovského smeru vypočíta bod x_0. Tenzorová metóda totiž v každej iterácii potrebuje na interpoláciu modelom štvrtého rádu dva body. Modifikovaná Newtonova metóda však začína v bode x_{-1} a prechod do bodu x_0 je už prvou iteráciou.

4.3. Výsledky experimentov.

4.3.1. Ilustratívne príklady

V tejto časti by sme chceli ilustrovať ako sa počas riešenia úlohy z iterácie na iteráciu menia jednotlivé sledované charakteristiky. Budeme pre bikvadratickú funkciu postupne riešiť úlohy rozmeru $n = 3, 5, 7, 10$, pričom sledujeme absolútnu aj relatívnu vzdialenosť aktuálneho bodu od optimálneho riešenia, funkčnú hodnotu a jej odchýlku od optimálnej funkčnej hodnoty, normu aktuálneho bodu a normu gradientu v aktuálnom bode, ktorá v prípade, že nepoznáme optimálne riešenie je jedinou charakteristikou, ktorú môžeme použiť na testovanie optimality v aktuálnom bode. Máme teda funkciu:

$$f(x) = \frac{1}{4\pi}\left(x^T G_1 x\right)^2 + \frac{1}{2}\left(x^T G_2 x\right) + h^T x,$$

kde $\pi = \|G_1\|_F^2$. V nasledujúcich štyroch príkladoch uvedenú funkciu postupne napĺňame nasledovnými údajmi :

Príklad 1

$$G_1 = \begin{bmatrix} 9 & -2 & -2 \\ -2 & 10 & 2 \\ -2 & 2 & 2 \end{bmatrix} \qquad G_2 = \begin{bmatrix} 3 & 0 & 1 \\ 0 & 4 & 1 \\ 1 & 1 & 2 \end{bmatrix} \qquad h = \begin{bmatrix} -2.555024 \\ 2.717703 \\ 3.717703 \end{bmatrix},$$

kde príslušný bod minima je $\hat{x} = (1, 0, -2)^T$
a minimálna funkčná hodnota je $f(\hat{x}) = -5.7428$.
Štartovací bod $x_{-1} = (-7.0178, 2.6726, 3.3452)^T$ vedie k nasledujúcim výsledkom:

iterácia	$\|x_k - \hat{x}\|$	$\frac{\|x_k - \hat{x}\|}{\|x_k\| + 1}$	$f(x_k)$	$f(x_k) - f(\hat{x})$	$\|\nabla f(x_k)\|$	$\|x_k\|$
0.	7,1819	2,2193	175,906	181,6488	108,1539	5,3479
1.	1,5,8892	1,8199	47,6594	53,4022	33,4432	3,776
2.	1,3,5774	1,1055	10,4518	16,1946	10,6575	1,6983
3.	0,2384	0,0737	-5,6093	0,1335	1,1525	2,3298
4.	0,0031	0,001	-5,7428	0	0,0171	2,2389

pričom $x_4 = (1.002403, -0.000424, -2.001982)^T$.

Príklad 2

$$G_1 = \begin{bmatrix} 58 & 8 & 32 & 20 & -13 \\ 8 & 13 & 4 & 2 & -16 \\ 32 & 4 & 27 & 11 & -2 \\ 20 & 2 & 11 & 18 & -8 \\ -13 & -16 & -2 & -8 & 35 \end{bmatrix} \qquad G_2 = \begin{bmatrix} 20 & 16 & -8 & -6 & -4 \\ 16 & 35 & -11 & -6 & -9 \\ -8 & -11 & 47 & 17 & 29 \\ -6 & -6 & 17 & 20 & 7 \\ -4 & -9 & 29 & 7 & 36 \end{bmatrix}$$

$$h = \begin{bmatrix} -128.049332 \\ -106.948189 \\ 246.030426 \\ 140.357727 \\ 221.896378 \end{bmatrix},$$

kde príslušný bod minima je $\hat{x} = (4, -1, -1, -4, -4)^T$
a minimálna funkčná hodnota je $f(\hat{x}) = -1074.2219$.
Štartovací bod $x_{-1} = (-5.6699, 0.2087, -3.4175, 4.4611, 3.2524)^T$ vedie k nasledujúcim výsledkom :

iterácia	$\|x_k - \hat{x}\|$	$\frac{\|x_{k+1} - \hat{x}\|}{\|x_{k+1}\| + 1}$	$f(x_k)$	$f(x_k) - f(\hat{x})$	$\|\nabla f(x_k)\|$	$\|x_k\|$
0.	6,5833	0,8157	-732,4617	341,7602	123,486	5,4626
1.	2,0748	0,2571	-1032,3989	41,823	42,3308	6,2864
2.	0,6491	0,0804	-1068,7035	5,5184	19,9688	7,4765
3.	0,0785	0,0097	-1074,1652	0,0568	1,6058	7,118
4.	0,0626	0,0078	-1074,1886	0,0333	1,1593	7,0273
5.	0,0467	0,0058	-1074,2051	0,0168	0,7394	7,1037
6.	0,0474	0,0059	-1074,2056	0,0164	0,7048	7,1063
7.	0,0083	0,001	-1074,2216	0,0004	0,0792	7,0738
8.	0,0007	0,0001	-1074,2219	0	0,0109	7,0716

pričom $x_8 = (4.000456, -1.00047, -0.999832, -4.000269, -4.000165)^T$.

Príklad 3

$$G_1 = \begin{bmatrix} 116 & 33 & 49 & -19 & -19 & 25 & -20 \\ 33 & 128 & 74 & -14 & 46 & -40 & -32 \\ 49 & 74 & 79 & 29 & 28 & 5 & 18 \\ -19 & -14 & 29 & 92 & 21 & 54 & 68 \\ -19 & 46 & 28 & 21 & 76 & 37 & 17 \\ 25 & -40 & 5 & 54 & 37 & 149 & 51 \\ -20 & -32 & 18 & 68 & 17 & 51 & 75 \end{bmatrix}$$

$$G_2 = \begin{bmatrix} 126 & 13 & 71 & 14 & -16 & 0 & -16 \\ 13 & 81 & 37 & 76 & 39 & 3 & -21 \\ 71 & 37 & 110 & 34 & 14 & -10 & -7 \\ 14 & 76 & 34 & 96 & 58 & -14 & -9 \\ -16 & 39 & 14 & 58 & 113 & -21 & -7 \\ 0 & 3 & -10 & -14 & -21 & 43 & -5 \\ -16 & -21 & -7 & -9 & -7 & -5 & 20 \end{bmatrix} \qquad h = \begin{bmatrix} 303.894684 \\ -476.629822 \\ -205.699677 \\ -406.221924 \\ -621.225953 \\ 69.43378 \\ 58.83212 \end{bmatrix},$$

kde príslušný bod minima je $\hat{x} = (-3, 6, 2, -3, 4, 0, 2)^T$

a minimálna funkčná hodnota je $f(\hat{x}) = -2840.0774$.

Štartovací bod $x_{-1} = (-3.0000, -9.9111, -1.1822, 8.1378, 5.5911, 3.1822, 2.0000)$ vedie k nasledujúcim výsledkom :

iterácia	$\|x_k - \hat{x}\|$	$\frac{\|x_k - \hat{x}\|}{\|x_k\| + 1}$	$f(x_k)$	$f(x_k) - f(\hat{x})$	$\|\nabla f(x_k)\|$	$\|x_k\|$
0.	18,082	1,8391	-1290,9104	1549,167	362,1848	13,6279
1.	9,8992	1,0069	-2335,8506	504,2268	144,798	7,8559
2.	1,5865	0,1614	-2805,4131	34,6643	70,7761	9,7103
3.	0,1495	0,0152	-2839,9534	0,124	3,4737	8,9234
4.	0,0034	0,0003	-2840,0776	-0,0002	0,0376	8,83

pričom $x_4 = (-3.000593, 5.998522, 2.000529, -2.999006, 3.999006, -0.000234, 1.997217)^T$.

Príklad 4

$$G_1 = \begin{bmatrix} 246 & -51 & -61 & 54 & 81 & -199 & -4 & -13 & 80 & -90 \\ -51 & 174 & 118 & 161 & -19 & -60 & 1 & 0 & -21 & 148 \\ -61 & 118 & 281 & 63 & -94 & 19 & 79 & -33 & 16 & 109 \\ 54 & 161 & 63 & 325 & 180 & -145 & -85 & -19 & 36 & 95 \\ 81 & -19 & -94 & 180 & 328 & -79 & -144 & 7 & 92 & -123 \\ -199 & -60 & 19 & -145 & -79 & 509 & 131 & 43 & -205 & 178 \\ -4 & 1 & 79 & -85 & -144 & 131 & 248 & 106 & -8 & 133 \\ -13 & 0 & -33 & -19 & 7 & 43 & 106 & 270 & 12 & -39 \\ 80 & -21 & 16 & 36 & 92 & -205 & -8 & 12 & 210 & -169 \\ -90 & 148 & 109 & 95 & -123 & 178 & 133 & -39 & -169 & 377 \end{bmatrix}$$

$$G_2 = \begin{bmatrix} 279 & -197 & 78 & 20 & -8 & -6 & -182 & 67 & 114 & -170 \\ -197 & 360 & 30 & -827 & -101 & 6 & 218 & -131 & -18 & 192 \\ 78 & 30 & 262 & 58 & -84 & -33 & -48 & -207 & 103 & 23 \\ 20 & -82 & 58 & 442 & -28 & 35 & 63 & 10 & -193 & 144 \\ -8 & -101 & -84 & -28 & 216 & -22 & -74 & 61 & 46 & 0 \\ -6 & 6 & -33 & 35 & -22 & 113 & 84 & -19 & -44 & 32 \\ -182 & 218 & -48 & 63 & -74 & 84 & 266 & -52 & -159 & 159 \\ 67 & -131 & -207 & 10 & 61 & -19 & -52 & 368 & -90 & -126 \\ 114 & -18 & 103 & -193 & 46 & -44 & -159 & -90 & 304 & -146 \\ -170 & 192 & 23 & 144 & 0 & 32 & 159 & -126 & -146 & 385 \end{bmatrix}$$

$h = (-2151.566162, 1746.028687, -1107.134521, 5285.330566, -1797.276123, 1754.445312,$

$3276.059570, 1743.218872, -5188.976562, 4752.085938)^T,$

kde príslušný bod minima je $\hat{x} = (4, -5, 2, -9, 8, -9, 7, -8, 3, -7)^T$
a minimálna funkčná hodnota je $f(\hat{x}) = -72331.8125$.
Štartovací bod $x_{-1} = (4.9600, 1.2402, 2.9600, -3.7198, 6.0799, -8.0400, 2.6799,$
$-8.4800, 4.4400, -4.5999)^T$ vedie k nasledujúcim výsledkom :

iterácia	$\|x_k - \hat{x}\|$	$\frac{\|x_k - \hat{x}\|}{\|x_k\| + 1}$	$f(x_k)$	$f(x_k) - f(\hat{x})$	$\|\nabla f(x_k)\|$	$\|x_k\|$
0.	2,072	0,0941	-72178,6562	153,1562	303,8971	22,3238
1.	0,1662	0,0075	-72331,1719	0,6406	11,8926	21,1073
2.	0,0008	0	-72331,8047	0,0078	0,0329	21,024

pričom $x_2 = (3.999923, -5.000445, 2.000037, -9.000074, 7.999842, -9.000198,$
$7.000439, -7.999915, 3.000265, -6.999770)^T$.

Vo všetkých uvedených príkladoch môžeme vidieť, že relatívna norma klesá rýchlejšie ako absolútna. Relatívnu presnosť považujeme za objektívnejšiu charakteristiku, keďže zohľadňuje aj veľkosť vektora x_k. Funkčná hodnota v aktuálnom bode sa pomerne rýchlo stabilizuje na hodnote blízkej k funkčnej hodnote v optimálnom bode. Taktiež môžeme povedať, že norma gradientu sa blíži k nule "pomalšie", ako sa aktuálny bod x_k blíži k optimálnemu bodu $\hat{x}$. Stabilizuje sa aj norma aktuálneho bodu x_k.

4.3.2. Testovanie na sériách po 100 úloh

V tejto časti uvádzame výsledky z testovaní na sériách po sto úloh rozmerov $n = 3, 5, 7, 10, 20, 30$. Sledovali sme počet iterácií a čas výpočtu. Každú úlohu sme riešili najprv tenzorovou metódou, potom sme sa "vrátili" do štartovacieho bodu a tú istú úlohu sme riešili modifikovanou Newtonovou metódou.

Séria č.1

$$n = 3,\ \rho = 10,\ seed = 2,\ \text{presnosť} : \frac{\|x_k - \hat{x}\|}{\|\hat{x}\| + 1} = 10^{-3}$$

Výsledky :

	tenzorová metóda	Newtonova metóda
priemerný počet iterácií :	3.67	3.07
priemerný čas výpočtu :	0.381	0.625

číslo úlohy	tenzorová metóda		newtonova metóda		číslo úlohy	tenzorová metóda		newtonova metóda	
	iterácií	čas	iterácií	čas		iterácií	čas	iterácií	čas
1.	5	0,3846	3	0,6040	51.	5	0,7143	4	0,8240
2.	2	0,1648	3	0,6590	52.	3	0,2198	3	0,6040
3.	3	0,2198	3	0,6040	53.	4	0,6044	4	0,8790
4.	2	0,1099	2	0,4400	54.	5	0,3846	3	0,6040
5.	2	0,1648	3	0,6040	55.	3	0,2198	3	0,6040
6.	3	0,1648	3	0,6040	56.	3	0,2198	3	0,6040
7.	5	1,2090	3	0,6040	57.	4	0,3297	3	0,6040
8.	3	0,2198	3	0,6040	58.	3	0,6044	3	0,6590
9.	4	0,2747	3	0,6040	59.	5	0,3297	4	0,8240
10.	3	0,2198	3	0,6040	60.	6	0,9341	3	0,6040
11.	2	0,1648	3	0,6040	61.	3	0,2747	3	0,6040
12.	3	0,2747	3	0,6040	62.	3	0,2198	4	0,8240
13.	4	0,3297	5	1,0400	63.	4	0,2747	3	0,6040
14.	4	0,3846	3	0,6040	64.	2	0,1099	2	0,3850
15.	4	0,3297	3	0,6040	65.	8	0,6593	3	0,6040
16.	2	0,1648	2	0,4400	66.	5	0,3846	3	0,6040
17.	3	0,2198	3	0,6040	67.	7	0,5495	4	0,7690
18.	3	0,2198	3	0,6040	68.	3	0,6044	3	0,6040
19.	2	0,1099	3	0,6590	69.	3	0,6044	3	0,6040
20.	2	0,1648	3	0,6040	70.	3	0,2747	3	0,6040
21.	2	0,1648	3	0,6040	71.	5	0,3297	3	0,6040
22.	3	0,2198	2	0,4400	72.	5	0,7143	4	0,8240
23.	4	0,3297	4	0,7690	73.	3	0,2198	4	0,8240
24.	4	0,2747	3	0,6590	74.	4	0,7143	3	0,6040
25.	6	0,8791	6	1,2100	75.	6	0,8242	4	0,7690
26.	1	0,1099	2	0,3850	76.	3	0,2198	3	0,6040
27.	4	0,2747	3	0,6040	77.	3	0,1648	3	0,6040
28.	1	0,1099	2	0,4400	78.	2	0,1648	2	0,4950
29.	2	0,1099	2	0,3850	79.	5	0,3297	4	0,8240
30.	7	0,5495	4	0,7690	80.	4	0,6044	3	0,6040
31.	6	1,0440	3	0,6040	81.	6	0,8791	3	0,6040
32.	4	0,3846	3	0,6040	82.	2	0,1099	3	0,6040
33.	6	0,4945	3	0,6040	83.	7	0,4945	4	0,7690
34.	3	0,5495	3	0,6590	84.	3	0,1648	3	0,6040
35.	2	0,1648	3	0,6590	85.	2	0,1648	3	0,6040
36.	3	0,2198	3	0,6040	86.	7	0,6044	3	0,6040
37.	6	0,4396	3	0,6040	87.	2	0,1648	3	0,6040
38.	2	0,1648	2	0,3850	88.	2	0,1648	3	0,6040
39.	5	0,7692	4	0,8240	89.	3	0,1648	3	0,6040
40.	4	0,7692	3	0,6040	90.	4	0,6593	3	0,6040
41.	4	0,7143	3	0,6040	91.	2	0,1099	3	0,6040
42.	6	0,4396	4	0,8240	92.	2	0,1648	3	0,6040
43.	2	0,1648	2	0,3850	93.	2	0,1648	3	0,6590
44.	3	0,2198	3	0,6040	94.	4	0,6593	3	0,6040
45.	2	0,1648	2	0,3850	95.	2	0,1099	2	0,3850
46.	5	0,4396	3	0,6040	96.	5	0,8242	3	0,6040
47.	3	0,6044	3	0,6040	97.	3	0,2198	2	0,3850
48.	1	0,0550	3	0,6040	98.	3	0,2747	3	0,6040
49.	6	0,7692	4	0,8240	99.	3	0,2198	3	0,6040
50.	5	0,8791	3	0,6040	100.	8	1,0440	3	0,6590
Porovnanie podľa počtu iterácií *		20		43	Porovnanie podľa času *		81		19

* Čísla udávajú počet príkladov,v ktorých sa daná metóda ukázala ako lepšia. **TAB 1**

Ako môžeme vidieť z tabuľky TAB 1 tenzorová metóda bola lepšia, čo sa týka počtu iterácií iba v 20 príkladoch zo 100. Taktiež priemerný počet iterácií tenzorovej metódy bol vyšší, ako priemerný počet iterácií modifikovanej Newtonovej metódy. 37 úloh vyriešili obidve metódy s rovnakým počtom iterácií. Čo sa týka času, tenzorová metóda sa javí ako podstatne rýchlejšia, čo je pomerne prekvapivý výsledok, keďže sme očakávali, že jedna iterácia tentorovej metódy, je minimálne tak časovo náročná, ako jedna iterácia modifikovanej Newtonovej metódy. Presnosť merania času v systéme je však diskutabilná.

Séria č.2

$$n = 5,\ \rho = 10,\ seed = 17,\ \text{presnosť} : \frac{\|x_k - \hat{x}\|}{\|\hat{x}\| + 1} = 10^{-3}$$

Výsledky :

	tenzorová metóda	Newtonova metóda
priemerný počet iterácií :	3.03	3.04
priemerný čas výpočtu :	0.508	1.169

Ako môžeme vidieť z nasledujúcej tabuľky TAB 2, Tenzorová metóda bola lepšia, čo sa týka počtu iterácii v 36 príkladoch zo 100, čo predstavuje určité zlepšenie, oproti sérii príkladov rozmeru $n = 3$. Rovnako priemerný počet iterácií klesol, dokonca je už nižší, ako priemerný počet iterácií v modifikovanej Newtonovej metóde, aj keď len veľmi tesne. Počet úloh, ktoré vyriešila za menšieho počtu iterácií modifikovaná Newtonova metóda je 23 a počet úloh, ktoré riešili obe metódy za rovnaký počet iterácií je 41. Teda úspešnosť modifikovanej Newtonovej metódy poklesla, čo sa prejavilo jednak na vyššej úspešnosti tenzorovej metódy a jednak na náraste počtu úloh, keď boli obe úlohy rovnako úspešné. Čo sa týka času, opäť je úspešnejšia tenzorová metóda.

číslo úlohy	tenzorová metóda		newtonova metóda		číslo úlohy	tenzorová metóda		newtonova metóda	
	iterácií	čas	iterácií	čas		iterácií	čas	iterácií	čas
1.	2	0,2198	3	1,1500	51.	4	0,5495	3	1,1500
2.	3	0,3297	3	1,3700	52.	2	0,2747	3	1,1500
3.	2	0,2747	2	0,7140	53.	3	1,0990	2	0,7690
4.	4	0,4945	3	1,1500	54.	2	0,2747	3	1,1500
5.	3	0,3846	3	1,1500	55.	3	0,4396	3	1,1500
6.	3	0,3846	3	1,1500	56.	2	0,2747	3	1,1500
7.	3	0,4396	3	1,1500	57.	2	0,3297	3	1,1500
8.	3	0,4396	3	1,1500	58.	4	0,5495	3	1,1500
9.	6	1,5930	3	1,1500	59.	2	0,2747	3	1,1500
10.	4	1,2090	4	1,5400	60.	2	0,2198	3	1,1500
11.	3	0,3297	3	1,1500	61.	4	1,2640	3	1,2100
12.	2	0,2747	3	1,1500	62.	1	0,1099	2	0,7690
13.	4	0,4396	4	1,4800	63.	4	0,6044	4	1,5400
14.	2	0,2198	3	1,2100	64.	2	0,2747	3	1,1500
15.	2	0,2198	3	1,1500	65.	2	0,2747	3	1,1500
16.	2	0,2198	3	1,2100	66.	2	0,2747	3	1,1500
17.	3	0,4396	3	1,1500	67.	2	0,2198	3	1,1500
18.	3	0,3846	3	1,1500	68.	3	0,3846	3	1,1500
19.	2	0,2198	3	1,2100	69.	6	0,8242	3	1,1500
20.	3	0,4396	3	1,1000	70.	3	1,1540	3	1,1500
21.	3	0,3846	3	1,1500	71.	4	0,5495	3	1,1500
22.	3	0,3846	3	1,1000	72.	2	0,2198	2	0,7690
23.	3	0,3846	3	1,1500	73.	4	0,4945	3	1,1500
24.	1	0,1099	2	0,7140	74.	3	0,3297	3	1,1500
25.	2	0,2747	3	1,1500	75.	3	1,2090	3	1,1500
26.	2	0,2198	3	1,2100	76.	2	0,2198	3	1,1500
27.	4	0,5495	3	1,1500	77.	4	0,6044	4	1,5400
28.	3	0,4396	3	1,1500	78.	4	0,5495	3	1,1500
29.	4	2,2530	3	1,1500	79.	5	0,7143	4	1,5400
30.	2	0,2747	2	0,7140	80.	3	0,4396	4	1,5400
31.	10	2,1980	4	1,4800	81.	2	0,2747	3	1,1500
32.	5	1,4290	3	1,1500	82.	2	0,2747	3	1,1500
33.	3	0,3297	3	1,1500	83.	3	0,3846	3	1,2100
34.	4	0,4945	3	1,1500	84.	3	0,4396	3	1,1500
35.	3	1,1540	4	1,5400	85.	4	0,6044	3	1,1500
36.	4	0,4945	4	1,5400	86.	3	0,4396	3	1,1500
37.	3	0,3846	3	1,2100	87.	2	0,2198	3	1,1500
38.	2	0,2747	3	1,1500	88.	3	0,3297	3	1,1500
39.	3	0,3846	3	1,1500	89.	2	0,2747	3	1,1500
40.	3	0,4396	3	1,1500	90.	3	0,3846	3	1,1500
41.	1	0,1099	2	0,8240	91.	3	0,4396	3	1,1500
42.	2	0,2747	4	1,4800	92.	2	0,2747	3	1,1500
43.	3	0,3846	3	1,1500	93.	4	1,9230	3	1,2100
44.	3	0,3297	3	1,1500	94.	3	0,4396	3	1,1500
45.	4	0,6044	3	1,1000	95.	2	0,2747	3	1,1000
46.	6	0,8242	3	1,1500	96.	4	0,5495	4	1,5400
47.	2	0,2198	3	1,2100	97.	1	0,1099	2	0,7690
48.	3	1,0990	3	1,1500	98.	2	0,2747	3	1,1500
49.	5	0,7143	4	1,5400	99.	5	0,6593	3	1,1500
50.	4	0,6044	3	1,1500	100.	2	0,2747	3	1,1500
Porovnanie podľa počtu iterácií *		36		23	Porovnanie podľa času *		91		9

* Čísla udávajú počet príkladov,v ktorých sa daná metóda ukázala ako lepšia.

TAB 2

	tenzorová metóda	Newtonova metóda
priemerný počet iterácií :	3.03	3.04
priemerný čas výpočtu :	0.508	1.169

Séria č.3

$$n = 7,\ \rho = 10,\ seed = 7,\ \text{presnosť} : \frac{\|x_k - \hat{x}\|}{\|\hat{x}\| + 1} = 10^{-3}$$

Výsledky :

	tenzorová metóda	Newtonova metóda
priemerný počet iterácií :	2.55	2.98
priemerný čas výpočtu :	0.592	1.883

Ako môžeme vidieť z nasledujúcej tabuľky TAB 3, počet úloh, keď bola úspešnejšia tenzorová metóda, čo sa týka počtu iterácií sa opäť zväčšil, v tejto sérii úloh je to 52 oproti 9 úlohám, ktoré vyriešila s menším počtom iterácií modifikovaná Newtonova metóda. Priemerný počet iterácií tenzorovej metódy klesol ešte hlbšie pod priemerný počet iterácií modifikovanej Newtonovej metódy. Počet úloh, ktoré vyriešili obe úlohy za rovnaký počet iterácií trochu klesol. Čo sa týka času, stále je úspešnejšia tenzorová metóda.

Séria č.4

$$n = 10,\ \rho = 30,\ seed = 17,\ \text{presnosť} : \frac{\|x_k - \hat{x}\|}{\|\hat{x}\| + 1} = 10^{-3}$$

Výsledky :

	tenzorová metóda	Newtonova metóda
priemerný počet iterácií :	3.38	3.82
priemerný čas výpočtu :	1.699	4.377

číslo úlohy	tenzorová metóda		newtonova metóda		číslo úlohy	tenzorová metóda		newtonova metóda	
	iterácií	čas	iterácií	čas		iterácií	čas	iterácií	čas
1.	3	1,70300	3	1,87000	51.	3	1,75800	3	1,92000
2.	2	0,38460	3	1,92000	52.	2	0,38460	3	1,87000
3.	1	0,16480	2	1,26000	53.	3	0,65930	3	1,87000
4.	3	0,71430	3	1,87000	54.	5	1,09900	4	2,53000
5.	3	0,71430	3	2,09000	55.	3	0,65930	3	1,92000
6.	3	0,71430	3	1,87000	56.	2	0,38460	3	1,92000
7.	2	0,49450	3	1,92000	57.	2	0,43960	3	1,92000
8.	2	0,38460	2	1,26000	58.	5	1,04400	3	1,92000
9.	2	0,38460	3	1,87000	59.	3	0,60440	3	1,87000
10.	2	0,49450	3	1,92000	60.	3	1,86800	4	2,53000
11.	3	0,71430	3	1,87000	61.	4	0,93410	4	2,53000
12.	2	0,38460	3	1,87000	62.	4	0,82420	3	1,87000
13.	2	0,38460	3	1,87000	63.	4	0,93410	3	1,92000
14.	6	1,26400	4	2,53000	64.	2	0,43960	3	1,92000
15.	2	0,38460	3	1,87000	65.	2	0,49450	3	1,87000
16.	4	0,82420	4	2,53000	66.	4	0,82420	4	2,53000
17.	3	0,65930	3	1,92000	67.	2	0,49450	3	1,92000
18.	2	0,43960	3	1,92000	68.	2	0,49450	3	1,92000
19.	4	0,82420	3	1,87000	69.	2	0,43960	3	1,98000
20.	2	0,43960	3	1,87000	70.	3	0,60440	3	1,87000
21.	3	0,60440	3	1,92000	71.	3	0,71430	3	1,87000
22.	3	0,65930	3	1,87000	72.	2	0,38460	2	1,21000
23.	2	0,49450	2	1,26000	73.	3	0,65930	3	1,92000
24.	2	0,38460	3	1,87000	74.	2	0,38460	3	1,87000
25.	2	0,38460	3	1,92000	75.	2	0,43960	3	1,87000
26.	4	0,82420	3	1,87000	76.	3	0,71430	3	1,87000
27.	2	0,38460	3	1,92000	77.	2	0,43960	3	1,81000
28.	2	0,38460	3	1,92000	78.	3	0,65930	3	1,92000
29.	2	0,43960	2	1,26000	79.	2	0,38460	2	1,26000
30.	1	0,27470	2	1,26000	80.	2	0,38460	3	1,87000
31.	2	0,43960	3	1,87000	81.	2	0,38460	3	1,87000
32.	2	0,38460	3	1,87000	82.	2	0,38460	3	1,87000
33.	1	0,21980	2	1,26000	83.	2	0,43960	3	1,81000
34.	3	0,65930	3	1,87000	84.	3	0,65930	3	1,98000
35.	2	0,38460	3	1,87000	85.	3	0,60440	3	1,87000
36.	4	0,93410	4	2,58000	86.	3	0,60440	4	2,58000
37.	2	0,38460	2	1,26000	87.	2	0,38460	3	1,87000
38.	2	0,43960	3	1,92000	88.	2	0,49450	3	1,92000
39.	3	0,65930	3	1,92000	89.	3	0,60440	3	1,92000
40.	3	0,60440	3	1,87000	90.	2	0,38460	3	1,81000
41.	3	0,65930	3	1,87000	91.	2	0,38460	3	1,87000
42.	1	0,27470	2	1,32000	92.	3	0,60440	3	1,92000
43.	3	0,65930	3	1,87000	93.	2	0,38460	3	1,98000
44.	3	0,65930	3	1,87000	94.	2	0,38460	3	1,92000
45.	4	0,98900	3	1,92000	95.	1	0,16480	2	1,26000
46.	3	1,81300	3	1,92000	96.	2	0,43960	3	1,87000
47.	2	0,38460	3	1,92000	97.	2	0,38460	4	2,47000
48.	2	0,38460	3	1,92000	98.	1	0,21980	2	1,26000
49.	4	0,87910	3	1,87000	99.	1	0,27470	2	1,26000
50.	4	0,87910	4	2,58000	100.	2	0,43960	4	2,53000
Porovnanie podľa počtu iterácií *	52		9		Porovnanie podľa času *	100		0	

* Čísla udávajú počet príkladov,v ktorých sa daná metóda ukázala ako lepšia.

TAB 3

číslo úlohy	tenzorová metóda		newtonova metóda		číslo úlohy	tenzorová metóda		newtonova metóda	
	iterácií	čas	iterácií	čas		iterácií	čas	iterácií	čas
1.	3	1,319	4	4,560	51.	4	1,648	4	4,560
2.	3	3,242	4	4,560	52.	4	1,648	4	4,620
3.	3	1,209	4	4,620	53.	3	1,154	4	4,560
4.	4	1,648	4	4,620	54.	3	1,209	3	3,460
5.	3	1,209	4	4,670	55.	3	1,264	4	4,620
6.	3	1,099	4	4,560	56.	3	1,264	4	4,560
7.	4	1,593	4	4,670	57.	4	3,516	4	4,620
8.	3	1,099	3	3,410	58.	3	1,154	4	4,620
9.	4	3,736	5	5,600	59.	3	1,209	4	4,510
10.	4	1,648	4	4,560	60.	3	1,209	3	3,410
11.	3	1,154	4	4,560	61.	4	1,593	5	5,660
12.	4	1,648	4	4,670	62.	4	3,681	4	4,560
13.	7	4,835	5	5,770	63.	4	1,484	4	4,620
14.	4	1,484	4	4,510	64.	3	1,099	3	3,460
15.	2	0,824	3	3,460	65.	3	3,297	4	4,560
16.	3	3,132	4	4,560	66.	3	1,264	4	4,560
17.	4	1,484	4	4,620	67.	3	1,209	3	3,410
18.	4	1,484	4	4,620	68.	3	1,264	3	3,410
19.	4	1,648	5	5,600	69.	3	1,209	3	3,410
20.	3	1,154	3	3,350	70.	4	1,648	4	4,560
21.	3	1,209	4	4,560	71.	3	1,209	4	4,620
22.	3	1,209	4	4,560	72.	2	0,769	3	3,460
23.	3	1,209	4	4,620	73.	3	1,099	3	3,410
24.	4	1,648	4	4,560	74.	4	3,571	4	4,620
25.	3	3,132	4	4,560	75.	4	1,648	4	4,620
26.	4	1,648	4	4,620	76.	2	0,769	3	3,460
27.	4	1,538	4	4,560	77.	3	1,209	4	4,560
28.	3	1,209	4	4,620	78.	6	6,319	4	4,560
29.	3	1,154	4	4,620	79.	3	1,209	3	3,460
30.	3	1,319	3	3,350	80.	3	1,209	4	4,560
31.	3	1,099	3	3,350	81.	4	3,516	4	4,560
32.	4	1,484	4	4,670	82.	3	1,209	4	4,560
33.	2	0,879	3	3,460	83.	3	1,209	4	4,560
34.	4	1,484	4	4,620	84.	3	1,209	5	5,600
35.	3	1,209	3	3,410	85.	4	1,593	4	4,670
36.	2	0,769	3	3,460	86.	4	1,648	4	4,560
37.	3	1,264	3	3,520	87.	4	1,593	4	4,670
38.	3	1,099	3	3,460	88.	3	3,187	4	4,560
39.	4	1,484	4	4,670	89.	2	0,769	3	3,350
40.	2	0,769	3	3,350	90.	4	1,538	4	4,620
41.	4	3,681	4	4,620	91.	4	1,538	4	4,510
42.	3	1,099	3	3,460	92.	3	1,099	4	4,620
43.	4	1,648	4	4,620	93.	3	1,154	3	3,460
44.	4	1,648	4	4,560	94.	3	1,154	4	4,670
45.	4	3,571	4	4,560	95.	3	3,132	3	3,520
46.	3	1,209	5	5,710	96.	4	1,648	4	4,510
47.	3	1,099	4	4,620	97.	4	3,571	4	4,620
48.	4	1,648	4	4,560	98.	3	1,099	4	4,620
49.	3	3,022	4	4,560	99.	4	1,484	5	5,710
50.	3	1,209	4	4,620	100.	3	1,209	4	4,620
Porovnanie podľa počtu iterácií *		46		2	Porovnanie podľa času *		99		1

* Čísla udávajú počet príkladov,v ktorých sa daná metóda ukázala ako lepšia.

TAB 4

Z predchádzajúcej tabuľky TAB 4 môžeme vidieť, že počet úloh, kde bola modifikovaná Newtonova metóda lepšia, čo sa týka počtu iterácií klesol oproti sérii úloh rozmeru $n = 7$. Ale napriek očakávaniam klesol aj počet úloh, pri ktorých bola úspešnejšia tenzorová metóda. Zvýšil sa počet úloh, kde mali obidve úlohy rovnaký počet iterácií. Priemerný počet iterácií stúpol pre obidve metódy. Môžeme pozorovať aj výraznejší nárast času výpočtu, čo je spôsobené väčším rozmerom úlohy. Stále je však úspešnejšia tenzorová metóda.

Séria č.5

$$n = 20,\ \rho = 10,\ seed = 1,\ \text{presnosť} : \frac{\|x_k - \hat{x}\|}{\|\hat{x}\| + 1} = 10^{-3}$$

Výsledky :

	tenzorová metóda	Newtonova metóda
priemerný počet iterácií :	1.94	2.78
priemerný čas výpočtu :	2.815	11.035

Ako môžeme vidieť z nasledujúcej tabuľky TAB 5, výrazne vzrástol počet úloh, pri ktorých bola tenzorová metóda úspešnejšia, čo sa týka počtu iterácií, na 73. Modifikovaná Newtonova metóda bola úspešnejšia len v 1 úlohe zo 100. Obidve metódy boli rovnako úspešné v 26 úlohách. Priemerný počet iterácií tenzorovej metódy je už o vyše 1 iteráciu nižší ako priemerný počet iterácií modifikovanej Newtonovej metódy. Takisto aj značne narástol rozdiel v časoch výpočtu, v prospech tenzorovej metódy. Teraz sa na tomto rozdiele podieľa aj pokles priemerného počtu iterácií tenzorovej metódy.

Séria č.6

$$n = 30,\ \rho = 10,\ seed = 1,\ \text{presnosť} : \frac{\|x_k - \hat{x}\|}{\|\hat{x}\| + 1} = 10^{-3}$$

Výsledky :

	tenzorová metóda	Newtonova metóda
priemerný počet iterácií :	1.89	2.78
priemerný čas výpočtu :	8.651	24.174

číslo úlohy	tenzorová metóda		newtonova metóda		číslo úlohy	tenzorová metóda		newtonova metóda	
	iterácií	čas	iterácií	čas		iterácií	čas	iterácií	čas
1.	2	2,527	3	12	51.	1	1,484	2	7,86
2.	2	2,967	3	11,7	52.	2	2,747	3	11,7
3.	3	4,231	3	12	53.	1	1,264	3	11,8
4.	2	9,835	2	7,91	54.	2	2,747	3	12
5.	1	1,264	2	7,91	55.	2	2,527	2	7,91
6.	2	2,802	3	12	56.	2	2,802	3	11,8
7.	2	9,835	2	7,91	57.	1	1,319	2	7,91
8.	1	1,264	3	11,8	58.	1	1,264	3	11,7
9.	2	2,747	3	12	59.	2	2,747	4	16,1
10.	2	2,527	3	12	60.	2	2,747	4	15,8
11.	3	4,286	3	12	61.	2	2,747	3	11,8
12.	2	2,802	2	7,91	62.	2	2,747	2	7,91
13.	2	2,747	2	7,91	63.	2	2,747	4	15,8
14.	3	0,1132	3	12	64.	2	9,835	3	12
15.	2	2,747	3	12	65.	2	2,802	3	11,7
16.	1	1,264	2	7,91	66.	2	2,747	3	12
17.	2	2,527	3	12	67.	2	0,1005	2	7,91
18.	1	1,264	2	7,91	68.	2	9,835	2	8,19
19.	2	2,527	2	7,86	69.	2	2,527	3	12
20.	2	2,747	3	11,7	70.	1	1,264	4	15,8
21.	2	2,527	3	12	71.	2	2,747	3	11,8
22.	2	2,582	3	12	72.	2	2,747	3	12
23.	3	4,011	3	12	73.	2	2,747	3	12
24.	1	1,264	2	7,91	74.	2	2,747	2	7,91
25.	2	2,802	3	12	75.	2	2,747	2	7,91
26.	3	0,1126	3	12	76.	2	0,1005	4	15,5
27.	4	0,2648	2	7,91	77.	2	9,615	2	7,91
28.	2	2,747	3	12	78.	2	2,802	3	12
29.	1	1,264	2	7,91	79.	3	4,011	3	12
30.	2	2,747	3	11,7	80.	2	2,582	3	12
31.	1	1,264	2	7,91	81.	2	2,527	4	15,8
32.	2	2,527	3	12	82.	2	2,747	3	12
33.	2	2,967	2	7,91	83.	2	2,527	3	11,7
34.	3	4,231	3	12	84.	2	2,747	3	11,7
35.	2	2,802	2	7,91	85.	2	2,747	3	11,7
36.	1	1,264	3	11,7	86.	2	2,527	3	11,8
37.	2	2,747	3	12	87.	1	1,264	2	7,91
38.	2	2,527	3	12	88.	2	2,527	3	12
39.	2	2,747	3	12	89.	2	2,747	3	11,8
40.	1	1,264	2	7,91	90.	2	2,747	3	12
41.	2	2,527	3	12	91.	2	2,527	3	12
42.	3	4,066	3	12	92.	2	9,341	2	7,91
43.	2	2,747	3	12	93.	1	1,264	2	7,91
44.	3	1,077	3	12	94.	1	1,264	4	15,8
45.	1	1,264	2	7,91	95.	2	2,802	3	12
46.	2	2,747	3	11,8	96.	2	2,582	3	12
47.	2	2,527	2	7,86	97.	2	2,527	3	12
48.	2	2,582	3	12	98.	3	4,231	3	12
49.	2	2,527	3	12	99.	2	2,747	3	12
50.	2	2,747	3	12	100.	2	2,8	3	12
Porovnanie podľa počtu iterácií *		73		1	Porovnanie podľa času *		95		5

* Čísla udávajú počet príkladov,v ktorých sa daná metóda ukázala ako lepšia.

TAB 5

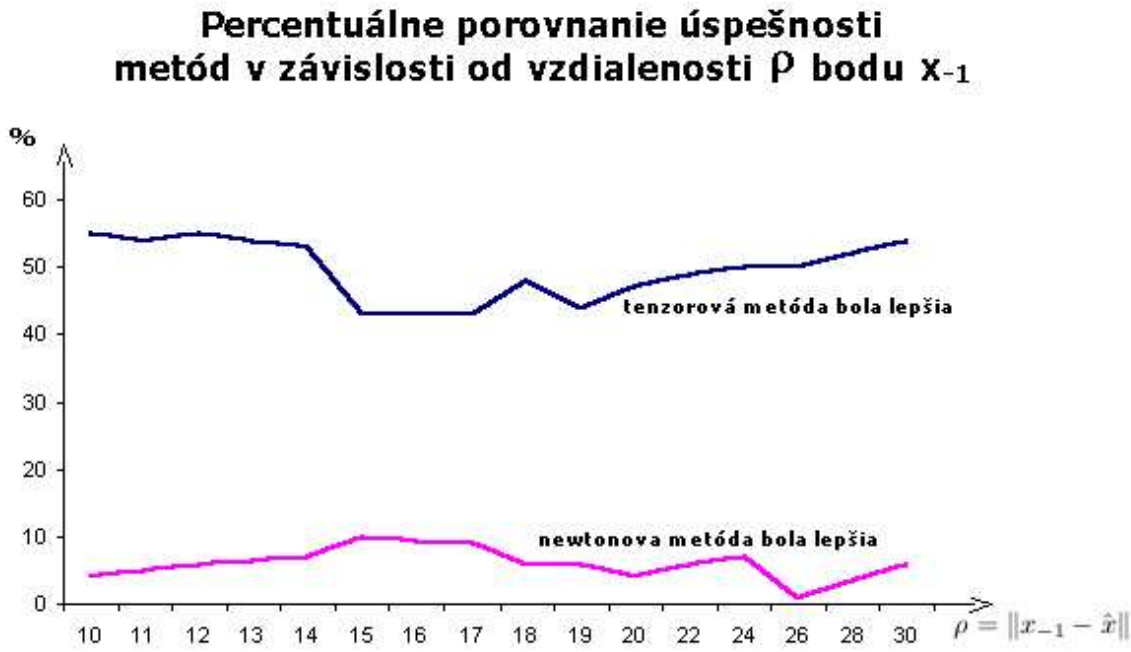
číslo úlohy	tenzorová metóda		newtonova metóda		číslo úlohy	tenzorová metóda		newtonova metóda	
	iterácií	čas	iterácií	čas		iterácií	čas	iterácií	čas
1.	1	2,91209	3	26,31868	51.	2	5,87912	3	26,31868
2.	2	5,87912	3	25,71429	52.	2	5,87912	4	34,72527
3.	2	5,87912	3	26,31868	53.	2	6,31868	3	26,37363
4.	2	6,31868	3	26,37363	54.	2	5,87912	3	26,37363
5.	2	21,42857	4	34,12088	55.	1	2,91209	2	17,36264
6.	2	6,31868	3	25,76923	56.	2	6,26374	4	35,27473
7.	1	2,91209	3	25,76923	57.	2	6,7033	3	26,31868
8.	2	5,87912	2	17,41758	58.	2	6,31868	2	17,36264
9.	2	21,48352	2	17,36264	59.	2	5,82418	3	26,31868
10.	2	5,87912	3	26,31868	60.	2	6,26374	4	34,72527
11.	2	5,87912	3	26,37363	61.	2	6,26374	4	34,72527
12.	2	6,26374	2	17,36264	62.	2	5,87912	3	25,71429
13.	2	5,87912	2	17,36264	63.	2	6,26374	3	26,31868
14.	3	38,95604	3	26,31868	64.	1	2,91209	3	25,71429
15.	1	2,96703	2	17,36264	65.	1	2,91209	3	26,37363
16.	2	6,26374	2	17,36264	66.	2	20,8791	3	25,76923
17.	4	27,08791	2	17,36264	67.	2	20,4396	2	17,41758
18.	2	6,31868	2	17,36264	68.	2	6,31868	3	25,76923
19.	2	5,87912	3	25,76923	69.	2	5,87912	3	25,76923
20.	2	6,31868	3	26,37363	70.	3	38,9011	2	17,36264
21.	1	2,96703	4	34,17582	71.	2	5,87912	3	26,31868
22.	2	6,75824	3	26,31868	72.	2	21,4835	2	17,41758
23.	2	5,87912	3	25,76923	73.	3	38,9011	2	17,36264
24.	2	6,7033	3	26,31868	74.	1	2,91209	2	17,36264
25.	1	2,91209	2	17,36264	75.	1	2,96703	3	25,76923
26.	2	5,87912	3	25,76923	76.	2	6,26374	4	34,72527
27.	2	5,87912	2	17,36264	77.	2	6,31868	2	17,36264
28.	2	6,31868	3	25,76923	78.	2	5,87912	4	34,12088
29.	1	2,96703	2	17,36264	79.	2	21,4835	3	25,76923
30.	3	23,79121	3	26,37363	80.	2	5,87912	3	26,31868
31.	3	39,34066	2	17,36264	81.	2	6,31868	4	34,78022
32.	1	2,91209	2	17,36264	82.	1	2,91209	3	25,76923
33.	2	6,26374	3	26,31868	83.	2	6,26374	2	17,36264
34.	2	6,31868	3	26,31868	84.	2	5,87912	3	26,37363
35.	2	5,87912	2	17,36264	85.	2	6,31868	3	26,37363
36.	2	5,87912	2	17,36264	86.	2	6,26374	3	26,31868
37.	2	6,31868	3	26,37363	87.	2	6,26374	3	26,31868
38.	1	2,91209	2	17,36264	88.	3	9,23077	4	34,72527
39.	2	6,26374	4	34,17582	89.	2	6,75824	3	26,31868
40.	2	6,75824	3	25,76923	90.	2	6,31868	3	26,37363
41.	2	6,75824	2	17,36264	91.	1	2,91209	2	17,36264
42.	2	6,26374	3	26,31868	92.	2	6,31868	2	17,36264
43.	2	21,48352	2	17,36264	93.	1	2,91209	3	26,37363
44.	2	5,82418	4	34,72527	94.	2	5,87912	2	17,36264
45.	1	2,91209	4	34,72527	95.	2	6,31868	2	17,36264
46.	1	2,91209	2	17,30769	96.	2	6,26374	3	26,31868
47.	2	20,98901	2	17,36264	97.	2	6,31868	3	26,31868
48.	1	2,91209	3	26,31868	98.	2	5,87912	2	17,36264
49.	2	6,31868	2	17,41758	99.	2	21,8681	3	26,31868
50.	2	6,26374	3	25,76923	100.	2	6,75824	3	26,37363
Porovnanie podľa počtu iterácií *		72		4	Porovnanie podľa času *		90		10

* Čísla udávajú počet príkladov,v ktorých sa daná metóda ukázala ako lepšia.

TAB 6

Z predchádzajúcej tabuľky TAB 6 môžeme vidieť, že úspešnosť tenzorovej metódy v tejto sérii úloh, čo sa týka počtu iterácií trochu poklesla oproti sérii úloh rozmeru $n = 20$ a zvýšila sa úspešnosť modifikovanej Newtonovej metódy, čo sme neočakávali. Priemerný počet iterácií tenzorovej metódy klesol, priemerný počet iterácií modifikovanej Newtonovej metódy sa nezmenil. Čas výpočtu vzrástol podľa očakávaní u obidvoch metód.

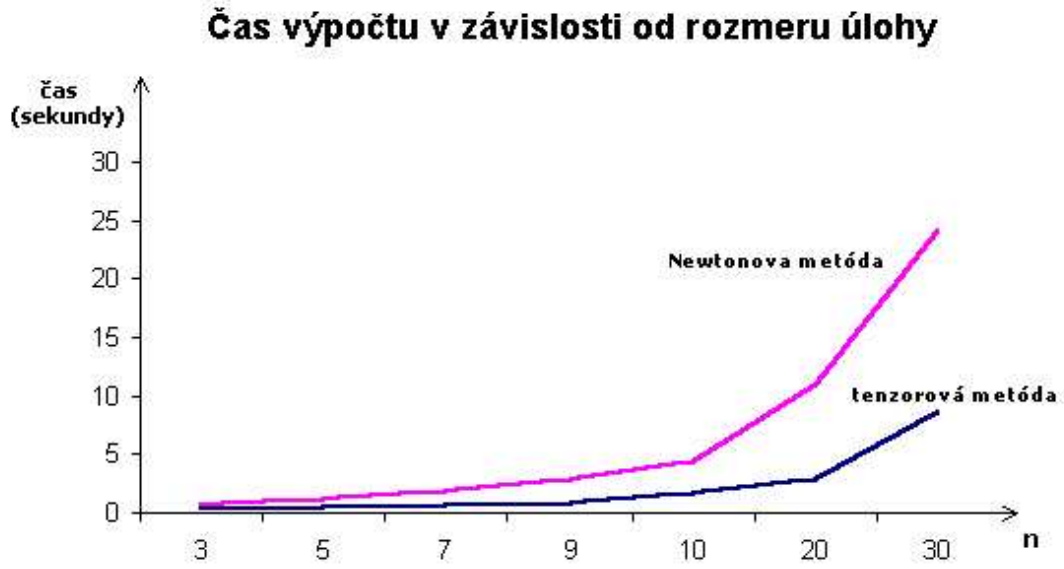
V ďalších experimentoch sme skúmali vplyv zmeny štartovacej vzdialenosti na efektívnosť oboch metód pre úlohu rozmeru $n = 10$. Keďže meranie času v systéme nie je celkom objektívne, nakoľko sa nedajú vylúčiť iné procesy prebiehajúce v systéme počas riešenia úloh, považujeme počet iterácií za relevantnejšie kritérium ako čas výpočtu. Nasledujúci graf zobrazuje závislosť úspešnosti jednotlivých metód v závislosti od voľby štartovacej vzdialenosti. Každá z kriviek zobrazuje, v koľkých úlohách zo sto, potrebovala daná metóda menší počet iterácií na dosiahnutie optimálneho riešenia s požadovanou presnosťou (v ďalšom kvôli stručnosti budeme hovoriť, že metóda A bola úspešnejšia ako metóda B, ak metóda A skonvergovala za menší počet iterácií).



Z grafu môžeme vidieť, že efektívnosť tenzorovej metódy oproti modifikovanej Newtonovej metóde s rastúcou štartovacou vzdialenosťou neklesá, naopak vykazuje stabilitu.

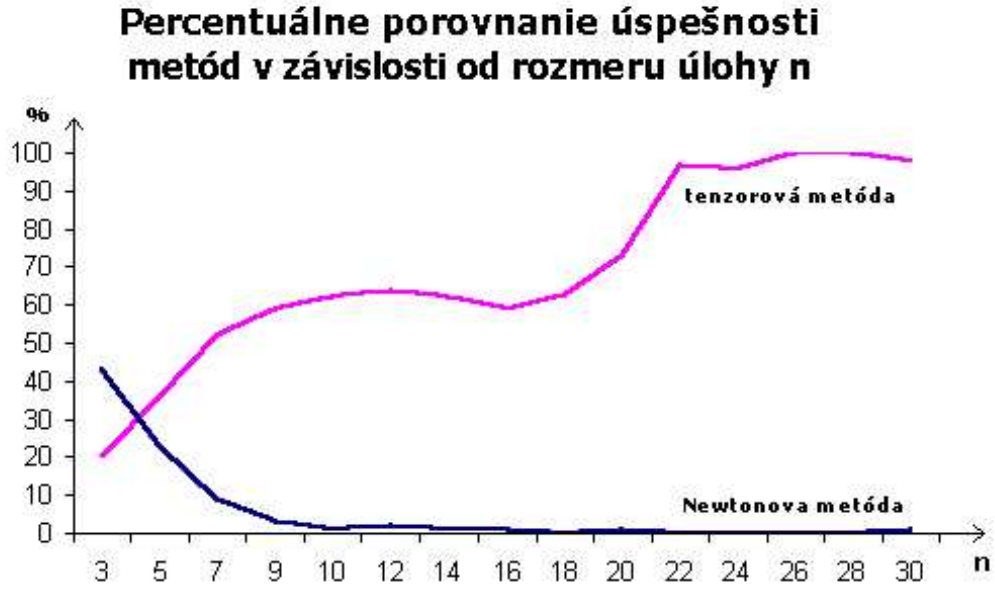
Minimálny počet úloh zo sto, ktoré vyriešila tenzorová metóda za menší počet iterácií ako modifikovaná Newtonova metóda je 43 a maximálny počet úloh, v ktorých bola úspešnejšia modifikovaná Newtonova metóda je 10.

Napriek tomu, že čas nepovažujeme za veľmi objektívnu charakteristiku, predsa len sa mu nedá odoprieť určitá výpovedná vlastnosť. Nasledujúci graf zobrazuje závislosť priemerného času riešenia úlohy od rozmeru úlohy.



Z grafu môžeme vidieť nielen to, že priemerný čas pre tenzorovú metódu je kratší, ale aj to, že priemerný čas riešenia úlohy modifikovanou Newtonovou metódou rastie s rastúcim rozmerom úlohy rýchlejšie, ako priemerný čas riešenia úlohy tenzorovou metódou.

Ďalší graf znázorňuje ako sa menila percentuálna úspešnosť obidvoch metód (vzhľadom na počet iterácií) v závislosti od zmeny rozmeru úlohy n. Štartovacia vzdialenosť bola $\rho = 10$ (riešili sme 100 úloh pre každý rozmer).



Z grafu môžeme vidieť, že už od rozmeru $n = 10$ je minimálne percento úloh, ktoré riešila úspešnejšie modifikovaná Newtonova metóda. Úspešnosť tenzorovej metódy je vyše 50%-ná, ostatné úlohy riešili obidve metódy za rovnaký počet iterácií. Od rozmeru úlohy $n = 22$ máme takmer 100%-nú úspešnosť tenzorovej metódy.

4.3.3. Vyhodnotenie na opakovaných sériách úloh

Keďže niektoré charakteristiky, ktoré sme sledovali v sériách po 100 úloh sa nesprávali celkom podľa našich očakávaní (ako sme už popísali vyššie), rozhodli sme sa opakovať experimenty vo viacerých sériách pre jednotlivé rozmery. V tomto rozhodnutí nás utvrdilo aj pozorovanie, že v sérii úloh rozmeru $n = 30$ poklesol počet úloh, ktoré riešila úspešnejšie tenzorová metóda (aj keď len o 1, očakávali sme však nárast). Uvedomili sme si totiž, že táto charakteristika vypovedá len o jedinej sérii a výsledok teda nemôžeme zovšeobecniť. Nasledujúce tabuľky teda zobrazujú 20 sérií po 100 úloh rozmerov $n = 3, 5, 7, 10$.

n=3	tenzorová metóda		newtonova metóda		porovnanie počtu iterácií	
číslo série	priem.počet iterácií	priem.čas výpočtu	priem.počet iterácií	priem.čas výpočtu	lepšia T	lepšia N
1.	3,26	0,380220	3,05	0,723630	33	32
2.	3,47	0,443960	3,03	0,719230	29	38
3.	3,37	0,421980	3,04	0,719780	28	32
4.	3,20	0,387910	3,13	0,737910	35	29
5.	3,36	0,462090	3,04	0,718130	25	37
6.	3,32	0,396150	2,99	0,706040	30	33
7.	3,31	0,383520	3,00	0,707690	23	36
8.	3,23	0,414840	2,95	0,700000	26	32
9.	3,29	0,454681	3,05	0,724135	38	22
10.	3,36	0,443960	3,13	0,742860	34	34
11.	3,49	0,492310	3,03	0,714290	25	41
12.	3,04	0,381870	3,02	0,717030	39	25
13.	3,46	0,428020	3,07	0,726370	23	31
14.	3,65	0,442310	3,05	0,723080	19	39
15.	3,47	0,431320	3,04	0,723080	28	37
16.	3,61	0,450000	3,03	0,714290	20	42
17.	3,36	0,441210	2,95	0,697800	27	38
18.	3,07	0,386260	2,98	0,707140	33	29
19.	3,36	0,442310	3,12	0,741210	33	33
20.	3,59	0,459340	3,04	0,723080	26	36
priemerné hodnoty	3,3635	0,42721	3,037	0,719339	28,7	33,8

n=5	tenzorová metóda		newtonova metóda		porovnanie počtu iterácií	
číslo série	priem.počet iterácií	priem.čas výpočtu	priem.počet iterácií	priem.čas výpočtu	lepšia T	lepšia N
1.	3,07	0,569780	3,15	1,373630	37	25
2.	2,86	0,506590	2,97	1,296700	32	14
3.	2,72	0,445050	3,02	1,314290	50	12
4.	2,82	0,465380	2,98	1,296150	41	15
5.	2,69	0,487360	2,91	1,263190	38	14
6.	2,92	0,619780	2,98	1,297250	34	21
7.	2,93	0,530220	3,11	1,353850	42	16
8.	2,83	0,513190	3,01	1,314290	41	19
9.	2,81	0,489010	2,99	1,301100	41	17
10.	2,83	0,492860	3,05	1,326370	38	12
11.	3,00	0,567580	2,97	1,291210	38	22
12.	2,68	0,488460	2,94	1,280220	44	12
13.	2,91	0,531320	3,03	1,314290	40	19
14.	2,96	0,557140	3,03	1,312640	34	21
15.	2,77	0,553850	2,98	1,297800	40	13
16.	2,88	0,537910	3,06	1,338460	41	19
17.	2,90	0,554400	3,05	1,328570	45	19
18.	2,96	0,540110	3,02	1,316480	34	19
19.	3,00	0,579670	3,07	1,335160	38	21
20.	2,98	0,535710	3,05	1,330220	34	22
priemerné hodnoty	2,876	0,52827	3,0185	1,314094	39,1	17,6

n=7	tenzorová metóda		newtonova metóda		porovnanie počtu iterácií	
číslo série	priem.počet iterácií	priem.čas výpočtu	priem.počet iterácií	priem.čas výpočtu	lepšia T	lepšia N
1.	2,45	0,641760	2,90	2,039560	52	6
2.	2,64	0,687910	2,95	2,071430	41	8
3.	2,62	0,707690	2,96	2,085710	45	10
4.	2,58	0,665930	2,94	2,065930	48	10
5.	2,56	0,624730	2,95	2,078570	51	11
6.	2,54	0,634070	2,92	2,052750	46	7
7.	2,74	0,725820	3,04	2,130220	40	8
8.	2,80	0,725270	3,01	2,115380	44	15
9.	2,78	0,712090	3,05	2,144510	43	42
10.	2,53	0,656590	2,93	2,056040	48	7
11.	2,57	0,634620	2,96	2,080220	44	5
12.	2,62	0,675270	2,97	2,084070	44	7
13.	2,50	0,632970	2,99	2,096700	54	5
14.	2,53	0,611540	2,99	2,101650	49	3
15.	2,53	0,643960	2,94	2,067580	51	7
16.	2,51	0,608790	2,96	2,082970	54	9
17.	2,70	0,664290	2,92	2,050000	31	6
18.	2,44	0,635160	2,91	2,048350	54	6
19.	2,71	0,767580	3,04	2,132970	46	11
20.	2,75	0,749450	3,00	2,105490	42	13
priemerné hodnoty	2,605	0,67027	2,9665	2,084505	46,35	9,8

n=10	tenzorová metóda		newtonova metóda		porovnanie počtu iterácií	
číslo série	priem.počet iterácií	priem.čas výpočtu	priem.počet iterácií	priem.čas výpočtu	lepšia T	lepšia N
1.	2,39	1,00549	2,91	3,60055	54	2
2.	2,42	0,98187	2,87	3,55385	46	1
3.	2,32	0,9956	2,86	3,54505	56	2
4.	2,33	0,94121	2,84	3,51209	54	3
5.	2,4	0,97088	2,85	3,53187	47	2
6.	2,5	1,0456	2,94	3,64505	49	5
7.	2,47	1,03901	3	3,71758	54	1
8.	2,35	0,94286	2,87	3,55	54	2
9.	2,38	0,98187	2,93	3,62527	57	2
10.	2,37	0,98132	2,9	3,58462	53	0
11.	2,35	0,94835	2,87	3,5511	52	0
12.	2,32	0,93516	2,93	3,62527	61	0
13.	2,26	0,92582	2,81	3,48132	56	1
14.	2,39	0,96319	2,87	3,55714	51	6
15.	2,48	1,0033	2,91	3,60604	45	2
16.	2,5	1,02143	2,95	3,65	51	6
17.	2,5	1,09176	2,98	3,69286	50	1
18.	2,39	1,00495	2,89	3,57857	53	3
19.	2,49	1,02527	2,93	3,62308	48	4
20.	2,4	0,96923	2,87	3,55604	51	4
priemerné hodnoty	2,4005	0,98871	2,899	3,589368	52,1	2,35

Z týchto tabuliek môžeme vidieť, že pre úlohy rozmeru $n = 3$ je priemerný počet iterácií tenzorovej metódy vyšší, ako priemerný počet iterácií modifikovanej Newtonovej metódy. Už od rozmeru $n = 5$ je však výsledok stále priaznivejší pre tenzorovú metódu. Čo sa týka úspešnosti metód vzhľadom na počet úloh zo 100 vyriešených za menší počet iterácií, pre $n = 3$ je tenzorová metóda menej úspešná ako modifikovaná Newtonova metóda. Už pre $n = 5$ je však úspešnejšia tenzorová metóda a s rastúcim rozmerom jej úspešnosť rastie. Čo sa týka času výpočtu, potvrdili sa výsledky z predchádzajúcej časti a teda čas výpočtu tenzorovej metódy je kratší ako čas výpočtu modifikovanej Newtonovej metódy.

ZÁVER

V predloženej práci sme sa zaoberali tenzorovou metódou na riešenie úlohy na voľný extrém. Cieľom práce bolo : 1) Podrobne popísať algoritmus tenzorovej metódy na základe článkov [1] a [2], 2) Experimentálne porovnať efektívnosť tenzorovej metódy a modifikovanej Newtonovej metódy.

Pri napĺňaní prvého cieľa sme podrobne študovali prácu [1], v ktorej sme objavili niekoľko nepresností vo vzorcoch. Tieto chyby sme opravili a príslušný algoritmus podrobne vysvetlili.

Pri napĺňaní druhého cieľa sme naprogramovali (v jazyku C++) príslušnú tenzorovú metódu spolu s modifikovanou Newtonovou metódou. S vytvoreným programom sme potom riešili úlohy do 30 premenných s regulárnou Hessovou maticou. Rozsiahle numerické experimenty sme robili na bikvadratickej funkcii, pričom sme pri zadanej požadovanej presnosti sledovali počet iterácií a čas výpočtu. Čo sa týka počtu iterácií, zistili sme, že percentuálna úspešnosť tenzorovej metódy oproti modifikovanej Newtonovej metóde rastie s rastúcim rozmerom úlohy. Ďalej sme zistili, že zmena vzdialenosti štartovacieho bodu x_{-1} od optimálneho riešenia $\hat{x}$ nemá vplyv na percentuálnu úspešnosť tenzorovej metódy v porovnaní s úspešnosťou modifikovanej Newtonovej metódy. Pomerne prekvapivo pôsobia výsledky meraní času výpočtu, ktoré hovoria jednoznačne v prospech tenzorovej metódy. Pri úlohách rozmeru $n \geq 7$, pri ktorých je už väčší rozdiel v počte iterácii tenzorovej a modifikovanej Newtonovej metódy (v prospech tenzorovej metódy) je pochopiteľné, že sa tento priaznivý rozdiel prejaví aj na čase výpočtu.

Pri úlohách menšieho rozmeru $n \leq 7$, sa javí ako úspešnejšia (čo do počtu iterácií) modifikovaná Newtonova metóda. V tomto prípade je tiež vysoké percento úloh, ktoré riešia obidve metódy za rovnaký počet iterácií. Tento výsledok pôsobí na prvý pohľad nepriaznivo pre tenzorvú metódu, ale vzhľadom na "geometrickú príbuznosť" bikvadratickej funkcie

s kvadratickou je odôvodniteľný. Pridanie tenzora tretieho a štvrtého stupňa (hodnosti 2 a 1) môže totiž znamenať aj opačný efekt na stupeň aproximácie danej funkcie.

Obe metódy sme ešte testovali na funkciách :

$$f_1(x) = ln\big(\sum_{j=1}^{n} c_j e^{x_j}\big) - h^T x + \frac{\varepsilon}{2} x^T I x$$

a

$$f_2(x) = (\sum_{j=1}^{n} x_j) ln\big(\sum_{j=1}^{n} x_j\big) - \sum_{j=1}^{n} \big(x_j ln x_j\big) - h^T x + \frac{\varepsilon}{2} x^T I x.$$

Pre funkciu f_1 sa ukázala ako veľmi úspešná modifikovaná Newtonova metóda. V prípadoch, keď boli parametre c_j volené ako čísla 10 až 10^3 krát menšie ako boli hodnoty $\hat{x}_j$ sme zaznamenali väčší výskyt prípadov, keď obidve metódy skonvergovali k optimálnemu riešeniu $\hat{x}$ za rovnaký počet iterácií. Percento úloh, pri riešení ktorých bola úspešnejšia tenzorová metóda bolo nulové pri všetkých voľbách parametrov c_j, ktoré sme vyskúšali.

Pre funkciu f_2 bolo vysoké percento úloh (okolo 75% a viac), ktoré riešili obidve metódy za rovnaký počet iterácií, pričom percento úloh, pri ktorých bola úspešnejšia tenzorová metóda bolo tiež nulové.

Na záver možno konštatovať, že v prípade regulárnej Hessovej matice je tenzorová metóda lepšia v porovnaní s modifikovanou Newtonovou metódou len pre niektoré funkcie a väčšie dimenzie. Jej význam je zrejme nespochybniteľný v prípade singulárnej Hessovej matice, keď Newtonovu metódu nemožno priamo použiť.

Literatúra

[1] A. BOUARICHA, *Tensor methods for large, sparse unconstrained optimization*, SIAM J. Optim. **7** (1997), 732-756.

[2] R. B. SCHNABEL AND T. CHOW, *Tensor methods for unconstrained optimization using second derivatives*, SIAM J. Optim. **1** (1991), 293-315.

[3] R. B. SCHNABEL AND P. D. FRANK, *Tensor methods for nonlinear equations*, SIAM J. Optim. **21** (1984), 815-843.

[4] M. HAMALA, *Prednášky z Nelineárneho programovania.*

PRÍLOHA

TENZOR.CPP

```
#include <stdio.h>
#include <conio.h>
#include <stdlib.h>
#include <string.h>
#include <math.h>
#include <time.h>

typedef float matica[100][100];
typedef float vektor[100];
typedef struct{float x1[100];float x2[100];float x3[100];} riesenie;
typedef float korene[3];

matica G1,G2,A,H_k,pomM;
vektor x_opt,x_start,y_0,h,g_k1,g_k,x_k1,x_k,s,pomV;
riesenie ries;
float pi,tau,ro=10,nor,nor1;
float f_k1,f_k,x0=0,x2=0,x3=0;

int i,j,k,n=10,info,pocitadlo1,pocitadlo2,lepsiT,rovnako,index;

void vloz_maticu(matica A,matica B) //uloli maticu A do matice B
{
    int i,j;
for(i=0;i<n;i++)
for(j=0;j<n;j++) B[i][j]=A[i][j];
}

void vloz_vektor(vektor a,vektor b)                //uloli vektor ado vektora b
{
int i;
for(i=0;i<n;i++) b[i]=a[i];
}

float abshod(float x)    // vracia absolutnu hodnotu cisla x
{
if(x>=0) return(x);
else return(-1*x);
}

float norma(vektor a) // vracia normu vektora a
{
```

```
   int i;
float suma=0;
for(i=0;i<n;i++) suma=suma+a[i]*a[i];
suma=sqrt(suma);
return suma;
}

float MTM(matica A, matica B)     // vracia sucin 2 matic AB
{
int k,i,j;
float suma;
for(k=0;k<n;k++)
    for(j=0;j<n;j++)
{
          suma=0;
for(i=0;i<n;i++)
suma=suma+A[i][k]*B[i][j];
pomM[k][j]=suma;
}
return(pomM[n][n]);
}

float Mv(matica A,vektor a)      // vracia sucin matice a vektora Aa
{
    int i,j;
float suma;
for(i=0;i<n;i++)
    {
suma=0;
for(j=0;j<n;j++) suma=suma+A[i][j]*a[j];
pomV[i]=suma;
}
return(pomV[n]);
}

float vTv(vektor a,vektor b)   // vracia skalarny suciin 2 vektorov
{
int i;
float suma=0;
for(i=0;i<n;i++) suma=suma+a[i]*b[i];
return(suma);
}

float vvT(vektor a,vektor b) // nasobi 2 vektory ab
{
int l,m;
```

```
    for(m=0;m<n;m++)
    for(l=0;l<n;l++)
       pomM[m][l]=a[m]*b[l];
    return(pomM[n][n]);
    }

float alphav(float alpha,vektor a)  // nasobi vektor kontantou
{
    int i;
    for(i=0;i<n;i++) pomV[i]=alpha*a[i];
    return(pomV[n]);
    }

float cholesky(matica A)  //pocita Choleskeho rozklad matice A
{
    float suma;
    int s,i,j,k;
    for(k=0;k<n;k++)
{
    suma=0;
    for(s=0;s<=k-1;s++) suma=suma+pomM[k][s]*pomM[k][s];
    if((A[k][k]-suma)<0) {printf("chyba: CHOLESKY");return(A[n][n]);}
    pomM[k][k]=sqrt(A[k][k]-suma);
    for(i=k+1;i<n;i++)
    {
          suma=0;
          for(s=0;s<=k-1;s++) suma=suma+pomM[i][s]*pomM[k][s];
          pomM[i][k]=(A[i][k]-suma)/pomM[k][k];
          }
       }
return(pomM[n][n]);
    }

float sustava1(matica A,vektor b)  // riei sustavu linearnych rovnic
{
    float pom;
    int i,j;
    vektor z;
    for(i=0;i<n;i++)
       {
       pom=0;
for(j=0;j<i;j++)
          pom=pom+A[i][j]*z[j];
      z[i]=(1/A[i][i])*(b[i]-pom);
       }
    for(i=n-1;i>=0;i--)
```

```
       {
       pom=0;
for(j=i+1;j<n;j++)
          pom=pom+A[j][i]*pomV[j];
     pomV[i]=(1/A[i][i])*(z[i]-pom);
       }
    return(pomV[n]);
}

riesenie sustava3(matica A,vektor a,vektor b,vektor c)      //naraz riei tri sustavy lin
{
    int i,j;
    float pom1,pom2,pom3;
    vektor z1,z2,z3;
    for(i=0;i<n;i++) {z1[i]=0;z2[i]=0;z3[i]=0;}
    for(i=0;i<n;i++)
       {
       pom1=0;pom2=0;pom3=0;
for(j=0;j<i;j++)
          {
          pom1=pom1+A[i][j]*z1[j];
          pom2=pom2+A[i][j]*z2[j];
          pom3=pom3+A[i][j]*z3[j];
}
      z1[i]=(1/A[i][i])*(a[i]-pom1);
       z2[i]=(1/A[i][i])*(b[i]-pom2);
       z3[i]=(1/A[i][i])*(c[i]-pom3);
       }
    for(i=n-1;i>=0;i--)
       {
       pom1=0;pom2=0;pom3=0;
for(j=i+1;j<n;j++)
          {
          pom1=pom1+A[j][i]*ries.x1[j];
          pom2=pom2+A[j][i]*ries.x2[j];
          pom3=pom3+A[j][i]*ries.x3[j];
          }
     ries.x1[i]=(1/A[i][i])*(z1[i]-pom1);
       ries.x2[i]=(1/A[i][i])*(z2[i]-pom2);
       ries.x3[i]=(1/A[i][i])*(z3[i]-pom3);
       }
    return(ries);
}

int kubika(float a0,float a1,float a2,float a3)  //riei kubicku rovnicu
{
```

```
float epsilon=1,D;
float f,g,x1,pom1,pom2,xmin,xmax,fmax,fmin;
    pom1=pow(a2,2);
    pom2=3*a1*a3;
    if(pom1<=pom2)
    {
x0=0;
f=a0+a1*x0+a2*pow(x0,2)+a3*pow(x0,3);
   while(epsilon>0.00001)
{
g=3*a3*pow(x0,2)+2*a2*x0+a1;
       x1=x0-f/g;
f=a0+a1*x1+a2*pow(x1,2)+a3*pow(x1,3);
         epsilon=abshod(x1-x0);
epsilon=epsilon/(1+abshod(x0));
          x0=x1;
   }
    return(1);
}
else
       {
          xmin=(-a2+sqrt(pow(a2,2)-3*a1*a3))/(3*a3);
          xmax=(-a2-sqrt(pow(a2,2)-3*a1*a3))/(3*a3);
fmin=a0+a1*xmin+a2*pow(xmin,2)+a3*pow(xmin,3);
      fmax=a0+a1*xmax+a2*pow(xmax,2)+a3*pow(xmax,3);
          if((fmin>0)||(fmax<0))
          {
x0=0;                                    f=a0+a1*x0+a2*pow(x0,2)+a3*pow(x0,3);
            while(epsilon>0.00001)
  {
g=3*a3*pow(x0,2)+2*a2*x0+a1;
x1=x0-f/g;
                f=a0+a1*x1+a2*pow(x1,2)+a3*pow(x1,3);
               epsilon=abshod(x1-x0);
          epsilon=epsilon/(1+abshod(x0));
          x0=x1;
}
return(1);
}
          if((fmin<0)&&(fmax>0))
             {
x0=-a2/(3*a3);
f=a0+a1*x0+a2*pow(x0,2)+a3*pow(x0,3);
                            while(epsilon>0.00001)
{
g=3*a3*pow(x0,2)+2*a2*x0+a1;
```

```
                    x1=x0-f/g;
                    f=a0+a1*x1+a2*pow(x1,2)+a3*pow(x1,3);
                              epsilon=abshod(x1-x0);
          epsilon=epsilon/(1+abshod(x0));
          x0=x1;
}
                pom1=a2+a3*x0;
                pom1=pow(pom1,2);
                pom2=a1+a2*x0+a3*pow(x0,2);
                D=pom1-4*a3*pom2;
                if(D<0) return(1);
                if(D==0)
                {
                   x2=(-1)*a2-a3*x0;
                   return(2);
                   }
                x2=(-1)*a2-a3*x0+sqrt(D);
                x2=x2/(2*a3);
                x3=(-1)*a2-a3*x0-sqrt(D);
                x3=x3/(2*a3);
            return(3);
           }
          }
    }

float f_x(vektor x)  //pocita funkcnu hodnotu
    {
    int i;
    float pom1,FI,pom3,pom4;
    vektor pom2;
    for(i=0;i<n;i++) pom2[i]=x[i];
    pom1=1/(2*sqrt(pi));
    alphav(pom1,pom2);
    vloz_vektor(pomV,pom2);
    Mv(G1,pom2);
    vloz_vektor(pomV,pom2);
    FI=vTv(x,pom2);
    FI=pow(FI,2);
Mv(G2,x);
vloz_vektor(pomV,pom2);
    pom3=vTv(pom2,x);
    pom3=0.5*pom3;
    pom4=vTv(h,x);
    pom1=FI+pom3+pom4;
    return(pom1);
    }
```

```
float gradient_f(vektor x,int info)  //pocita gradient
{
    int i;
float pom1;
    vektor pom2,pom3;
    Mv(G1,x);
vloz_vektor(pomV,pom2);
    Mv(G2,x);
vloz_vektor(pomV,pom3);
    pom1=(1/pi)*vTv(x,pom2);
    if(info==1)
       {
    for(i=0;i<n;i++)
    pomV[i]=pom1*pom2[i]+pom3[i]+h[i];
       return(pomV[n]);
       }
    if(info==0)
       {
    for(i=0;i<n;i++)
    pomV[i]=-pom1*pom2[i]-pom3[i];
       return(pomV[n]);
       }
    }

float Hess(vektor x)            //pocita Hessovu maticu
    {
    int i,j;
    float pom1;
    matica pom;
    vektor pom2;
    Mv(G1,x);
    vloz_vektor(pomV,pom2);
    vvT(pom2,pom2);
    vloz_maticu(pomM,pom);
    pom1=vTv(x,pom2);
    pom1=pom1/pi;
    for(i=0;i<n;i++)
      for(j=0;j<n;j++)
     pomM[i][j]=(2/pi)*pom[i][j]+pom1*G1[i][j]+G2[i][j];
    return(pomM[n][n]);
    }

float newton_step(matica A,vektor grad)  //pocita Newtonovsky smer
{
    vektor pom;
```

```
    alphav(-1,grad);
vloz_vektor(pomV,pom);
sustava1(A,pom);
return(pomV[n]);
    }

float zlaty_rez(vektor x,vektor d)     //metoda zlateho rezu
{
    float fi_1,fi_2,a,b,c1,c2,pom,t,z1,z2,nor;
    vektor pom1;
    t=(sqrt(5)+1)/2;
    z1=2-t;
    z2=t-1;
    b=1;
    for(i=0;i<n;i++) pom1[i]=x[i]+b*d[i];
    gradient_f(pom1,1);
    pom=vTv(pomV,d);
    while(pom<0)
    {
       b=2*b;
       for(i=0;i<n;i++) pom1[i]=x[i]+b*d[i];
    gradient_f(pom1,1);
    pom=vTv(pomV,d);
       }
    a=0;
    c1=a+z1*(b-a);
    c2=a+z2*(b-a);
    for(i=0;i<n;i++) pom1[i]=x[i]+c1*d[i];
    fi_1=f_x(pom1);
    for(i=0;i<n;i++) pom1[i]=x[i]+c2*d[i];
    fi_2=f_x(pom1);
    nor=b-a;
    nor=abshod(nor);
    while(nor>0.0001)
    {
       if(fi_1<fi_2)
          {
          b=c2;
          c2=c1;
       fi_2=fi_1;
          nor=b-a;
   nor=abshod(nor);
          if(nor<0.0001) break;
    else
          {
                c1=a+z1*(b-a);
```

```
                for(i=0;i<n;i++) pom1[i]=x[i]+c1*d[i];
        fi_1=f_x(pom1);
           }
           }
        else
        {
           a=c1;
           c1=c2;
     fi_1=fi_2;
           nor=b-a;
    nor=abshod(nor);
           if(nor<0.0001) break;
     else
        {
                c2=a+z2*(b-a);
                 for(i=0;i<n;i++) pom1[i]=x[i]+c2*d[i];
           fi_2=f_x(pom1);
           }
           }
        nor=b-a;
    nor=abshod(nor);
     }
     return(c1);
     }

void GENERUJ(int rozsah) // generuje ulohu
{
float max,pom1;
     int n1,n2;
     vektor pom2;
     for(i=0;i<n;i++)
for(j=0;j<n;j++)
           {
           n1=random(rozsah);
     n2=n1%2;
     n1=2*n2-1;
     A[i][j]=n1* random(rozsah);
           }
MTM(A,A);
vloz_maticu(pomM,A);
     for(i=0;i<n;i++)
     {
        G1[i][i]=A[i][i]+1;
for(j=0;j<n;j++)
        if(i!=j) G1[i][j]=A[i][j];
        }
```

```
    for(i=0;i<n;i++)
for(j=0;j<n;j++)
          {
          n1=random(rozsah);
    n2=n1%2;
    n1=2*n2-1;
    A[i][j]=n1* random(rozsah);
          n1=random(rozsah);
    n2=n1%2;
    n1=2*n2-1;
    x_opt[i]=n1*random(rozsah);
          n1=random(rozsah);
n2=n1%2;
    n1=2*n2-1;
    y_0[i]=n1*random(rozsah);
          }
    MTM(A,A);
vloz_maticu(pomM,A);
   for(i=0;i<n;i++)
    {
       G2[i][i]=A[i][i]+1;
for(j=0;j<n;j++)
       if(j!=i)G2[i][j]=A[i][j];
       }
pi=0;
    max=0;
for(i=0;i<n;i++) for(j=0;j<n;j++)
if(abshod(G1[i][j])>max) max=abshod(G1[i][j]);

    for(i=0;i<n;i++) {
    for(j=0;j<n;j++)
{
          pom1=G1[i][j]/max;
pi=pi+pow(pom1,2);
}
}
pi=pi*pow(max,2);
gradient_f(x_opt,0);
vloz_vektor(pomV,h);
for(i=0;i<n;i++) pom2[i]=y_0[i]-x_opt[i];
tau=ro/sqrt(vTv(pom2,pom2));
for(i=0;i<n;i++) x_k1[i]=x_opt[i]+tau*pom2[i];
vloz_vektor(x_k1,x_start);
f_k1=f_x(x_k1);
gradient_f(x_k1,1);
vloz_vektor(pomV,g_k1);
```

```
Hess(x_k1);
vloz_maticu(pomM,H_k);
    cholesky(H_k);
vloz_maticu(pomM,A);
    newton_step(A,g_k1);
for(i=0;i<n;i++) x_k[i]=x_k1[i]+pomV[i];
f_k=f_x(x_k);
gradient_f(x_k,1);
vloz_vektor(pomV,g_k);
Hess(x_k);
    vloz_maticu(pomM,H_k);
}

void iteracia() //jedna iteracia tenzorovej metody
     {
     float pom0,pom1,pom3;
     float mi,eta,gama,beta,alpha,delta,sigma,ni,f_p,f_t,f_n;
     vektor a,b,pom2,x_kt,x_kn,x_p;
     int infoR;
     korene psi,omega;
     float d[3][100];
     riesenie y;
     cholesky(H_k);
vloz_maticu(pomM,A);
     for(i=0;i<n;i++) s[i]=x_k1[i]-x_k[i];
     pom0=vTv(g_k,s);
     Mv(H_k,s);
     vloz_vektor(pomV,pom2);
     pom1=vTv(s,pom2);
     pom3=vTv(g_k1,s);
     mi=pom3-pom0-pom1;
ni=f_k1-f_k-pom0-0.5*pom1;
beta=24*mi-72*ni;
pom0=vTv(s,s);
gama=beta/pow(pom0,4);
pom3=pow(pom0,3)/6;
pom1=pom3*gama;
for(i=0;i<n;i++) a[i]=(g_k1[i]-g_k[i]-pom2[i]-pom1*s[i])*2;
pom1=1/pow(pom0,2);
pom3=2*vTv(s,a)/(3*pow(pom0,3));
for(i=0;i<n;i++)  b[i]=pom1*a[i]-pom3*s[i];
y=sustava3(A,s,b,g_k);
alpha=vTv(y.x1,g_k);
delta=vTv(y.x1,b);
sigma=vTv(y.x1,s);
eta=vTv(y.x2,g_k);
```

```
beta=vTv(y.x2,b);
infoR=0;
if(sigma==0)
     {
          pom0=1-2*alpha*delta;
if(pom0<0)
{
printf("\nNo minimizer -> psi: 1-2*alpha*delta<0");   infoR=0;
}
if(pom0>=0)
                {
                if(pom0==0)
                   {
                psi[0]=-1/delta;
                   infoR=1;
                   }
                else
                {
          psi[0]=(-1+sqrt(pom0))/delta;
                psi[1]=(-1-sqrt(pom0))/delta;
                if(psi[0]!=0) infoR=1;
if(psi[1]!=0)
{
infoR=2;
if(psi[0]==0)
                                    {
                                       psi[0]=psi[1];
                                       infoR=1;
                                       psi[1]=0;
                                       }
 }
                  }
                if(infoR==2)
                   {
pom0=1+delta*psi[0];
          pom1=1+delta*psi[1];
          if((pom0==0)&&(pom1==0))
                     {
printf("\nNo minimizer -> omega: 1+delta*psi=0");
infoR=0;
}
if(pom0!=0)
omega[0]=(1/pom0)*(-eta-0.5*beta*pow(psi[0],2)+(gama/6)*delta*pow(psi[0],3));
                   else
                      {
omega[0]=(1/pom1)*(-eta-0.5*beta*pow(psi[1],2)+(gama/6)*delta*pow(psi[1],3));
```

```
                        infoR=1;
                        }
if(pom1!=0)
omega[1]=(1/pom1)*(-eta-0.5*beta*pow(psi[1],2)+(gama/6)*delta*pow(psi[1],3));
                  else infoR=1;
}
if(infoR==1)
{
                  pom0=1+delta*psi[0];
                  if(pom0==0)
{
printf("\nNo minimizer -> omega: 1+delta*psi=0");
infoR=0;
}
          if(pom0!=0)
omega[0]=(1/pom0)*(-eta-0.5*beta*pow(psi[0],2)+(gama/6)*delta*pow(psi[0],3));

                  }
               }
          }
 if(sigma!=0)
{
info=kubika(-1*alpha,gama*sigma-2*alpha*delta-1,-1.5*delta,0.5*sigma*beta-gama/6-0.5*del
psi[0]=x0;
       psi[1]=x2;
       psi[2]=x3;
       i=0;
       if(info==1)
{
          omega[0]=alpha+psi[0]+0.5*delta*pow(psi[0],2);
          pom1=gama*sigma*pow(psi[0],3);
          pom1=pom1/6;
          omega[0]=pom1+omega[0];
          pom1=sigma*psi[0];
          omega[0]=omega[0]/pom1;
          omega[0]=omega[0]*(-1);
       infoR=1;
          }
       if(info==3)
          {
          for(i=0;i<3;i++)
{
             omega[i]=alpha+psi[i]+0.5*delta*pow(psi[i],2);
             pom1=gama*sigma*pow(psi[i],3)/6;
          omega[i]=omega[i]+pom1;
             pom1=sigma*psi[i];
```

```
            omega[i]=omega[i]/pom1;
            omega[i]=omega[i]*(-1);
            }
         infoR=3;
         }
       }
     if(infoR!=0)
{
        for(j=0;j<infoR;j++)
        {
for(i=0;i<n;i++) pom2[i]=-g_k[i]-omega[j]*psi[j]*s[i]-0.5*pow(psi[j],2)*b[i]-(gama/6)*pow
       sustava1(A,pom2);
    for(i=0;i<n;i++) d[j][i]=pomV[i];
          }
       }
     vloz_vektor(x_k,x_k1);
     if(infoR==0)
       {
printf("\nnebol najdeny minimalizator tenzoroveho modelu, bude pouzity newtonovsky krok.
       f_k1=f_k;
      newton_step(A,g_k);
       vloz_vektor(pomV,pom2);
       pom1=zlaty_rez(x_k,pom2);
for(i=0;i<n;i++) x_k[i]=x_k1[i]+pom1*pom2[i];
       f_k=f_x(x_k);
       }
    if(infoR!=0)
        {
       pom0=1000000;
       for(i=0;i<infoR;i++)
       {
            for(j=0;j<n;j++) pom2[j]=d[i][j];
            pom1=norma(pom2);
            if(pom1<pom0) {index=i;pom0=pom1;}
            }
for(i=0;i<n;i++) pom2[i]=d[index][i];
       pom0=vTv(g_k,pom2);
       for(i=0;i<n;i++) x_p[i]=x_k[i]+d[index][i];
       f_k1=f_k;
       f_p=f_x(x_p);
       if(f_p<f_k+0.0001*pom0) {vloz_vektor(x_p,x_k);f_k=f_p;}
       else
       {
          pom1=zlaty_rez(x_k,pom2);
       for(i=0;i<n;i++) d[index][i]=pom1*d[index][i];
       for(i=0;i<n;i++) pom2[i]=d[index][i];
```

```
           for(i=0;i<n;i++) x_kt[i]=x_k1[i]+d[index][i];
          newton_step(A,g_k);
       vloz_vektor(pomV,pom2);
        pom1=zlaty_rez(x_k,pom2);
for(i=0;i<n;i++) x_kn[i]=x_k1[i]+pom1*pom2[i];
          f_n=f_x(x_kn);
          f_t=f_x(x_kt);
          if(f_n<f_t) {vloz_vektor(x_kn,x_k);f_k=f_n;}
          else {vloz_vektor(x_kt,x_k);f_k=f_t;}
          }
       }
    vloz_vektor(g_k,g_k1);
    gradient_f(x_k,1);
    vloz_vektor(pomV,g_k);
    Hess(x_k);
    vloz_maticu(pomM,H_k);
    pocitadlo1++;
   }

void main()
{
float krok,pom1,/*pom2,*/presnost=0.001;
vektor rozdiel,sN;
clock_t start1,start2, end1,end2;
int STOP;
clrscr();
srand(123);
lepsiT=0;
rovnako=0;
STOP=0;
for(k=0;k<100;k++)
{
pocitadlo1=0;
GENERUJ(10);
nor=norma(g_k);
for(i=0;i<n;i++) rozdiel[i]=x_k[i]-x_opt[i];
nor1=norma(rozdiel);
pom1=norma(x_opt);
pom1=pom1+1;
nor1=nor1/pom1;
start1=clock();
while(nor1>presnost)
{
iteracia();
nor=norma(g_k);
    for(i=0;i<n;i++) rozdiel[i]=x_k[i]-x_opt[i];
```

```
    nor1=norma(rozdiel);
    pom1=norma(x_opt);
pom1=pom1+1;
nor1=nor1/pom1;
    if(pocitadlo1>249)
    {
          STOP=1;
          break;
          }
    }
end1 = clock();
if(k<9) printf("%d.   %d ",k+1,pocitadlo1);
if((k>8)&&(k<99)) printf("%d.  %d ",k+1,pocitadlo1);
if(k>98) printf("%d. %d ",k+1,pocitadlo1);
if(pocitadlo1<10) printf("  ");
if((pocitadlo1>9)&&(pocitadlo1<100)) printf(" ");
if(STOP==1) printf(" stopped  ");
else printf(" %.3E s",(end1-start1)/CLK_TCK);
pocitadlo2=0; //tu zacina modifikovana Newtonova metoda
STOP=0;
vloz_vektor(x_start,x_k);
gradient_f(x_k,1);
vloz_vektor(pomV,g_k);
nor=norma(g_k);
for(i=0;i<n;i++) rozdiel[i]=x_k[i]-x_opt[i];
nor1=norma(rozdiel);
pom1=norma(x_opt);
pom1=pom1+1;
nor1=nor1/pom1;
start2=clock();
while(nor1>presnost)
{
   pocitadlo2++;
    Hess(x_k);
    vloz_maticu(pomM,H_k);
    cholesky(H_k);
vloz_maticu(pomM,A);
    newton_step(A,g_k);
    vloz_vektor(pomV,sN);
    krok=zlaty_rez(x_k,sN);
for(i=0;i<n;i++)
    {
    x_k[i]=x_k[i]+krok*sN[i];
rozdiel[i]=x_k[i]-x_opt[i];
       }
    gradient_f(x_k,1);
```

```
vloz_vektor(pomV,g_k);
    nor=norma(g_k);
    nor1=norma(rozdiel);
    pom1=norma(x_opt);
pom1=pom1+1;
nor1=nor1/pom1;
if(pocitadlo2>249)
    {
          STOP=1;
         break;
          }
}
end2=clock();
printf("    %d ",pocitadlo2);
if(pocitadlo2<10) printf("  ");
if((pocitadlo2>9)&&(pocitadlo2<100)) printf(" ");
if(STOP==1) printf(" stopped  ");
else printf(" %.2E s\n",(end2-start2)/CLK_TCK);
if(pocitadlo1<pocitadlo2) lepsiT++;
if(pocitadlo1==pocitadlo2) rovnako++;
}
printf("\nTenzorova metoda mala mensi pocet iteracii v %d pripadoch",lepsiT);
printf("\nRovnaky pocet iteracii mali obe metody v %d pripadoch",rovnako);
getch();
}
```