**UNIVERZITA KOMENSKÉHO V BRATISLAVE**
**FAKULTA MATEMATIKY, FYZIKY A INFORMATIKY**

Evidenčné číslo: f95e1338-0635-46f7-a9ae-a35b7a13fb44

# Metódy riešenia úloh o sedlovom bode

Bc. Alena Hrabovská

Bratislava, 2011

**UNIVERZITA KOMENSKÉHO V BRATISLAVE**
**FAKULTA MATEMATIKY, FYZIKY A INFORMATIKY**

# Metódy riešenia úloh o sedlovom bode

(Diplomová práca)

Bc. Alena Hrabovská

**Študijný odbor:** Ekonomická a finančná matematika
**Vedúci práce:** doc. RNDr. Milan Hamala, CSc.

Bratislava, 2011

Univerzita Komenského v Bratislave
Fakulta matematiky, fyziky a informatiky

# ZADANIE ZÁVEREČNEJ PRÁCE

**Meno a priezvisko študenta:** Bc. Alena Hrabovská
**Študijný program:** ekonomická a finančná matematika (Jednoodborové štúdium, magisterský II. st., denná forma)
**Študijný odbor:** 9.1.9. aplikovaná matematika
**Typ záverečnej práce:** diplomová
**Jazyk záverečnej práce:** slovenský

**Názov :** Metódy riešenia úloh o sedlovom bode

**Cieľ :** 1. Oboznámiť sa s aplikáciami metód sedlového bodu v algoritmoch nelineárneho programovania 2. Naprogramovať niektoré metódy sedlového bodu. 3. Numerickým experimentom porovnať ich efektívnosť.

**Vedúci :** doc. RNDr. Milan Hamala, CSc.

**Dátum zadania:** 27.11.2008

**Dátum schválenia:** 20.04.2011

.....
prof. RNDr. Marek Fila, DrSc.
garant študijného programu

.....
študent

.....
vedúci práce

Dátum potvrdenia finálnej verzie práce, súhlas s jej odovzdaním (vrátane spôsobu sprístupnenia)

.....
vedúci práce

Čestne vyhlasujem, že som diplomovú prácu vypracovala samostatne, na základe svojich vedomostí s pomocou odbornej literatúry a cenných rád vedúceho diplomovej práce.

V Bratislave 8. augusta 2011

_____

podpis študenta

**Poďakovanie**

Ďakujem vedúcemu diplomovej práce, Doc. RNDr. Milanovi Hamalovi, CSc. za neoceniteľné rady a za čas a ochotu pri konzultáciách. Vďaka patrí aj mojej rodine a priateľom, ktorí mi boli oporou počas celého vysokoškolského štúdia.

# Abstrakt

HRABOVSKÁ, Alena: *Metódy riešenia úloh o sedlovom bode.* Diplomová práca - Univerzita Komenského v Bratislave; Fakulta matematiky, fyziky a informatiky; Katedra aplikovanej matematiky a štatistiky. Školiteľ: doc. RNDr. Milan HAMALA, CSc., Bratislava 2011.

Táto diplomová práca sa zaoberá metódami riešenia úloh o sedlovom bode. Ústrednou témou je rozšírená Lagrangeova funkcia pre úlohu konvexného programovania. Hlavným cieľom práce je naprogramovanie algoritmu využívajúceho túto funkciu. Posledná časť práce je venovaná numerickému experimentu, ktorý zahŕňa porovnanie výsledkov pre rôzne funkcie.

**Kľúčové slová:** nelineárne programovanie, úloha konvexného programovania, rozšírená Lagrangeova funkcia.

# Abstract

HRABOVSKÁ, Alena: *About different ways to solve saddlepoint problem.* Diploma work - Comenius University, Bratislava; Faculty of Mathematics, Physiscs and Informatics; Department of applied mathematics and statistics. Adviser: doc. RNDr. Milan HAMALA, CSc., Bratislava 2011.

This diploma work discusses different approaches to solution of saddlepoint problem. Key topic is augmented Lagrangian function for convex programming. The goal of this work is to create algorithm utilizing that function. Last part of the work is dedicated to numerical experiment which involves comparison of results for different functions.

**Keywords:** nonlinear programming, problem of convex programming, augmented Lagrangian function

# Obsah

# Úvod

V posledných rokoch sme svedkami narastania záujmu o problém optimalizácie v mnohých oblastiach ľudskej činnosti. Nelineárne programovanie je jedným z odvetví optimalizácie. Svoje začiatky má v polovici minulého storočia, keď po rozvoji lineárneho programovania už nestačilo opisovať rôzne javy iba pomocou lineárnych rovníc, ale bolo potrebné riešiť zložitejšie úlohy. V roku 1963 sa Everett [3] pokúšal riešiť tieto úlohy hľadaním sedlového bodu Lagrangeovej funkcie, no ukázalo sa, že táto metóda má viacero nedostatkov a je nevhodná na riešenie úloh na viazaný extrém. Už o pár rokov nato, v roku 1969, Hestenes [4] odstránil tieto nedostatky a prišiel s prvou rozšírenou Lagrangeovou funkciou, ktorá je dodnes známa ako Hestenesova funkcia. Táto funkcia je vhodná na riešenie úloh s ohraničeniami v tvare rovností, no už v roku 1973, Rockafellar [5] predstavil ďalšiu rozšírenú Lagrangeovu funkciu, ktorá rieši úlohy na viazaný extrém. Následne v sedemdesiatych rokoch minulého storočia vypukol veľký "boom" spojený s týmito metódami. Ten neskôr ustal, no záujem o túto problematiku sa ešte úplne nestratil, čoho dôkazom sú aj články [12] a [13].

Práve problematikou rozšírených Lagrangeových funkcií sa budeme zaoberať v tejto diplomovej práci. V úvode si zadefinujeme základné pojmy nelineárneho programovania. V ďalšej kapitole sformulujeme zovšeobecnenú Lagrangeovu funkciu pre úlohu konvexného programovania a skonštruujeme algoritmus na jej využitie. Následne budeme realizovať numerický experiment na porovanie vplyvu zmeny rôznych vstupných parametrov na hodnoty výstupov a zároveň vykonáme porovnanie pre rôzne funkcie.

# Kapitola 1

# Základné pojmy

V tejto kapitole zadefinujeme základné pojmy týkajúce sa nelineárneho programovania, potrebné pre lepšie pochopenie skúmanej problematiky. [1]

## 1.1 Úloha nelineárneho programovania

Majme funkciu $f(\mathbf{x})$: $R^n \to R$, ktorú budeme ďalej nazývať **účelovou funkciou** a funkcie $g_i(\mathbf{x})$: $R^n \to R \quad (i = 1, \ldots, m)$, ktoré budeme nazývať **funkciami ohraničení** určujúcimi tzv. **množinu prípustných riešení**
$K = \{\mathbf{x} | g_i(\mathbf{x}) \geq 0, \quad i = 1, \ldots, m\}$.

**Definícia 1.1.1** ***Úlohou nelineárneho programovania*** *nazývame úlohu nájsť minimum účelovej funkcie $f(\mathbf{x})$ na množine prípustných riešení $K$ a budeme ju zapisovať nasledovne:*

$$Min\{f(\mathbf{x}) \mid g_i(\mathbf{x}) \geq 0; \quad i = 1, \ldots, m\}, \tag{1.1}$$

Prípustné riešenie $\hat{\mathbf{x}} \in K$, v ktorom účelová funkcia $f(\mathbf{x})$ nadobúda svoje minimum (vzhľadom na množinu prípustných riešení $K$) nazývame **optimálnym riešením** úlohy (1.1).

V teórii nelineárneho programovania sú známe dva spôsoby vyjadrenia nutných a postačujúcich podmienok pre optimálne riešenie úlohy (1.1).

Prvý spôsob používa diferenciálny počet a je známy ako Kuhn-Tuckerove podmienky. V takom prípade o úlohe (1.1) musíme predpokladať, že jej funkcie $f$, $g_i$ sú diferencovateľné. Príslušné Kuhn-Tuckerove podmienky reprezentujú nutné podmienky optimality. (Kuhn-Tuckerove podmienky sa stanú postačujúcimi podmienkami optimality, ak úloha (1.1) je navyše úlohou konvexného programovania, o čom pojednáva časť 1.2.).

[1] Všetky definície a vety sú prevzaté z [1] a [2]

**Veta 1.1.1** *(**Kuhn-Tucker**) Nech funkcie $f(\mathbf{x}), g_i(\mathbf{x}) \quad (i = 1, \ldots, m)$ sú diferencovateľné, t.j. $f(\mathbf{x}), g_i(\mathbf{x}) \in C^1$. Nech $\hat{\mathbf{x}}$ je optimálnym riešením úlohy* (1.1) *a nech je v určitom zmysle "regulárnym bodom"*[2].
*Potom existuje $\hat{\mathbf{u}} \in R^m$ tak, že platia tzv. Kuhn-Tuckerove podmienky (ďalej ich budeme skrátene označovať K-T podmienky):*

$$\nabla f(\hat{\mathbf{x}}) - \sum_{i=1}^{m} \hat{\mathbf{u}}_\mathbf{i} \nabla g_i(\hat{\mathbf{x}}) = 0_n \tag{1.2}$$

$$g_i(\hat{\mathbf{x}}) \geq 0 \qquad i = 1, \ldots, m \tag{1.3}$$

$$\hat{\mathbf{u}}_i g_i(\hat{\mathbf{x}}) = 0 \qquad i = 1, \ldots, m \tag{1.4}$$

$$\hat{\mathbf{u}}_\mathbf{i} \geq 0 \qquad i = 1, \ldots, m \tag{1.5}$$

**Poznámka:** V teórii nelineárneho programovania sa obvykle predpokladá, že popri (1.4) platí aj

$$\hat{u_i} + g_i(\hat{\mathbf{x}}) > 0 \qquad i = 1, \ldots, m \tag{1.6}$$

(tzv. "strictly complementarity" predpoklad).

Druhý spôsob vyjadrenia nutných a postačujúcich podmienok pre optimálne riešenie úlohy (1.1) je pomocou sedlového bodu tzv. Lagrangeovej funkcie. Pomocou sedlového bodu dostaneme postačujúce podmienky optimality. Tieto podmienky sa stanú nutnými, ak úloha (1.1) bude úloha konvexného programovania (viac v časti 1.2.).

**Definícia 1.1.2** ***Lagrangeovou funkciou*** $L\colon R^n \times R^m_+ \to R$ *úlohy* (1.1) *nazývame funkciu*

$$L(\mathbf{x}, \mathbf{u}) = f(\mathbf{x}) - \sum_{i=1}^{m} \mathbf{u}_\mathbf{i} g_i(\mathbf{x}). \tag{1.7}$$

*Premenné $\mathbf{u} \in R^m_+$ sa nazývajú Lagrangeove premenné.*

**Definícia 1.1.3** *Bod $(\mathbf{x^0}, \mathbf{u^0})$ sa nazýva **sedlovým bodom Larangeovej funkcie**, ak platí:*

$$\forall \boldsymbol{x} \in R^n \;\; \forall \mathbf{u} \in R^m_+ \colon L(\mathbf{x^0}, \mathbf{u}) \leq L(\mathbf{x^0}, \mathbf{u^0}) \leq L(\mathbf{x}, \mathbf{u^0}) \tag{1.8}$$

*Zložky vektora $\mathbf{u^0} \in R^m_+$ sa nazývajú Lagrangeove multiplikátory, vektor $\mathbf{u^0} \in R^m_+$ sa nazýva vektor Lagrangeových multiplikátorov.*

**Veta 1.1.2** *Nech $(\hat{\mathbf{x}}, \hat{\mathbf{u}}) \in R^n \times R^m_+$ je sedlový bod Lagrangeovej funkcie* (1.7).
*Potom $\hat{\mathbf{x}}$ je optimálnym riešením úlohy nelineárneho programovania* (1.1).

[2]Podrobnosti o regularite sú uvedené v [2]

Poznamenávame, že Lagrangeova funkcia (1.7) umožňuje zapísať K-T podmienky (1.2) – (1.5) v kompaktnom tvare:

$$\nabla_x L(\hat{\mathbf{x}}, \hat{\mathbf{u}}) = 0 \tag{1.9}$$

$$\nabla_u L(\hat{\mathbf{x}}, \hat{\mathbf{u}}) \leq 0 \tag{1.10}$$

$$\hat{\mathbf{u}}^T \nabla_u L(\hat{\mathbf{x}}, \hat{\mathbf{u}}) = 0 \tag{1.11}$$

$$\hat{\mathbf{u}} \geq 0 \tag{1.12}$$

Vzťah (1.9) vyjadruje podmienku stacionarity bodu $\hat{\mathbf{x}}$ funkcie $L(\mathbf{x}, \hat{\mathbf{u}})$ vzhľadom na premennú $\mathbf{x} \in R^n$, ďalšie tri vzťahy (1.10) - (1.12) sú nutnými podmienkami pre maximum funkcie $L(\hat{\mathbf{x}}, \mathbf{u})$ vzhľadom na premennú $\mathbf{u} \in R^m_+$.

## 1.2 Úloha konvexného programovania

Ak spĺňa úloha (1.1) vlastnosti konvexnosti, tak K-T podmienky sú zároveň aj postačujúcimi podmienkami optimality a sedlový bod Lagrangeovej funkcie sa stáva nutnou podmienkou optimality. Tieto skutočnosti sme už spomenuli v predchádzajúcej časti. Teraz podrobnejšie charakterizujeme úlohu konvexného programovania.

**Definícia 1.2.1** *Nech $\Omega \subseteq R^n$ je konvexná množina.*
*Potom funkciu $f(\mathbf{x})\colon \Omega \to R$ nazývame konvexnou, ak platí :*

$$\forall \mathbf{x}, \mathbf{y} \in \Omega, \mathbf{x} \neq \mathbf{y} \quad a \quad \forall \lambda \in (0,1): \qquad f(\lambda x + (1-\lambda)y) \leq \lambda f(x) + (1-\lambda) f(y). \tag{1.13}$$

*Funkciu $g\colon \Omega \to R$ nazývame konkávna, ak $(-g)$ je konvexná.*

**Definícia 1.2.2** *Úlohu nelineárneho programovania* (1.1) *nazývame* ***úlohou konvexného programovania****, ak účelová funkcia $f$ je konvexná a funkcie ohraničení $g_i$ sú konkávne. Zapisovať ju budeme nasledovne:*

$$Min\{f(\mathbf{x}) \mid g_i(\mathbf{x}) \geq 0; \quad i = 1, \ldots, m\}. \tag{1.14}$$

Úlohou konvexného programovania (1.14) má nasledovné užitočné vlastnosti:
1. Množina prípustných riešení je konvexná.
2. Každé lokálne minimum funkcie je zároveň aj globálnym minimom.
3. Množina všetkých miním je konvexná množina.

**Veta 1.2.1** *Nech platia Kuhn-Tuckerove podmienky* (1.2) – (1.5).
*Potom $\hat{\mathbf{x}}$ je optimálnym riešením úlohy* (1.14).

**Veta 1.2.2** *(Veta Slatera) Majme úlohu konvexného programovania* (1.14)*, ktorá má nasledovnú vlastnosť (S-regularita):*

$$\exists \bar{\mathbf{x}} \in R^n : g_i(\bar{\mathbf{x}}) > 0, \qquad i = 1, \ldots, m$$

*Nech* $\hat{\mathbf{x}}$ *je optimálnym riešením takejto S-regulárnej úlohy konvexného programovania. Potom existuje* $\hat{\mathbf{u}} \in R^m_+$ *tak, že* $(\hat{\mathbf{x}}, \hat{\mathbf{u}})$ *je sedlovým bodom Lagrangeovej funkcie* (1.7)*.*

## 1.3 Metódy riešenia úlohy konvexného programovania

K-T podmienky (1.2)−(1.5) pre úlohu konvexného programovania (1.14) sú nielen nutnými podmienkami optimality (Veta 1.1.2.), ale aj postačujúcimi podmienkami optimality (Veta 1.2.2).

Úlohu (1.14) by sme teda mohli riešiť vyriešením systému nelineárnych rovníc a nerovníc (1.2)−(1.5). Na tomto postupe sú založené mnohé algoritmy, ktorými sa však v tejto diplomovej práci nezaoberáme.

Charakteristika optimálneho riešenia úlohy konvexného programovania (1.14) pomocou sedlového bodu Lagrangeovej funkcie vytvára inú možnosť na jej riešenie. Úloha o sedlovom bode má totiž pomerne jednoduchú množinu prípustných riešení $R^n \times R^m_+$, a preto možno očakávať, že v niektorých prípadoch nájsť sedlový bod Lagrangeovej funkcie môže byť kvalitatívne jednoduchšie ako priame riešenie úlohy (1.14).

Hľadanie sedlového bodu možno realizovať striedavou minimalizáciou funkcie $L(\mathbf{x}, \bar{\mathbf{u}})$ pri fixovanom $\bar{\mathbf{u}} \geq 0_m$ a maximalizáciou funkcie $L(\bar{\mathbf{x}}, \mathbf{u})$ pri fixovanom $\bar{\mathbf{x}} \in R^n$.
Takýto prístup k riešeniu úlohy (1.14) navrhol Everett [3] už v roku 1963. Narazil však na určité technické ťažkosti (minimum Larangeovej funkcie nemusí existovať pre každé $\mathbf{u} \geq 0_m$). Preto neskôr boli navrhnuté určité zovšeobecnenia Lagrangeovej funkcie, ktoré odstránili spomenuté ťažkosti. *Týmto technikám je venovaná predložená diplomová práca.*

# Kapitola 2

# Zovšeobecnená Lagrangeova funkcia

Ako sme už spomenuli, klasická Lagrangeova funkcia nie je práve najvhodnejším nástrojom na riešenie úloh nelineárneho programovania, nakoľko nemusí existovať minimum tejto funkcie pre každé $\mathbf{u} \geq 0_m$. Tento fakt podnietil vznik rozšírených Lagrangeových funkcií (tzv. *"Augmented Lagrangian"*). Rozšírená Lagrangeova funkcia vznikne z klasickej Lagrangeovej funkcie pridaním ďalšieho výrazu (tzv. penalizácie). Ako prvý predstavil takúto funkciu Hestenes [4] v roku 1969 - funkcia bola skonštruovaná pre úlohu s ohraničeniami v tvare rovností. Pre úlohy s ohraničeniami v tvare nerovností navrhol ďalšiu funkciu Rockafellar [5] v roku 1973.

## 2.1 Rozšírená Lagrangeova funkcia Rockafellara

Ako sme už spomenuli, pre úlohu konvexného programovania (1.14) prvú zovšeobecnenú funkciu navrhol Rockafellar [5] v roku 1973 . Má tvar:

$$R(\mathbf{x}, \mathbf{y}; \eta) = f(\mathbf{x}) + \frac{1}{2\eta} \sum_{i=1}^{m} (\max[0, \eta g_i(\mathbf{x}) - y_i]^2 - y_i^2), \tag{2.1}$$

alebo ekvivalentne

$$R(\mathbf{x}, \mathbf{y}; \eta) = \begin{cases} f(\mathbf{x}) - \sum_{i=1}^{m} y_i g_i(\mathbf{x}) + \frac{\eta}{2} \sum_{i=1}^{m} g_i^2(\mathbf{x}) & \text{ak } \eta g_i(\mathbf{x}) - y_i \geq 0 \\ f(\mathbf{x}) - \frac{1}{2\eta} \sum_{i=1}^{m} y_i^2 & \text{ak } \eta g_i(\mathbf{x}) - y_i < 0, \end{cases} \tag{2.2}$$

kde $\mathbf{x} \in R^n, \mathbf{y} \in R^m$ a $\eta > 0$ je parameter.

Rockafellarova funkcia je oproti klasickej Lagrangeovej funkcii rozšírená o tzv. penalizačný výraz, ktorý penalizuje prípadné porušenie podmienok prípustnosti.

Rockafellarova funkcia zachováva kľúčovú vlastnosť klasickej Lagrangeovej funkcie uvedenú vo vetách (1.1.2) a (1.2.2), tj. jej sedlový bod jednoznačne korešponduje s optimálnym riešením príslušnej úlohy konvexného programovania (1.14) ( a zároveň nemá nedostatok klasickej Lagrangeovej funkcie spomenutý v úvode).

## 2.2 Zovšeobecnená Lagrangeova funkcia s lineárnymi multiplikátormi

**Definícia 2.2.1** *Zovšeobecnenou Lagrangeovou funkciou s lineárnymi multiplikátormi* $\mathcal{L} : R^n \times R^m_+ \to R$ *úlohy konvexného programovania* (1.14) *nazývame funkciu tvaru*

$$\mathcal{L}(\mathbf{x}, \mathbf{y}) = f(\mathbf{x}) + \sum_{i=1}^{m} y_i \psi[g_i(\mathbf{x})], \tag{2.3}$$

*kde tzv. transformačná funkcia* $\psi : R \to R$ *má nasledovné vlastnosti:*

*(i)* $\psi \in C^2$,

*(ii)* $\psi''(\xi) > 0 \ \ \forall \xi \in R$, *tj.* $\psi$: $R \to R$ *je konvexná,*

*(iii)* $\psi'(\xi) < 0$, *tj.* $\psi$ *klesajúca,*

*(iv)* $\psi(0) = 0$ , *teda podľa (iii)* $\psi(\xi) > 0$ *pre* $\xi < 0$ *a* $\psi(\xi) < 0$ *pre* $\xi > 0$,

*(v)* $\psi'(0) = -1$.

*Premenné* $\mathbf{y} \in R^m_+$ *sa nazývajú zovšeobecnené Lagrangeove premenné.*

**Veta 2.2.1** *Nech* $(\hat{\mathbf{x}}, \hat{\mathbf{y}}) \in R^n \times R^m_+$ *je sedlovým bodom funkcie* (2.3).
*Potom* $\hat{\mathbf{x}} \in R^n$ *je optimálnym riešením úlohy* (1.14).

Dôkaz: Nech platí :

$$\forall \mathbf{x} \in R^n \ \ \forall \mathbf{y} \geq 0_m : \ \ \mathcal{L}(\hat{\mathbf{x}}, \mathbf{y}) \leq \mathcal{L}(\hat{\mathbf{x}}, \hat{\mathbf{y}}) \leq \mathcal{L}(\mathbf{x}, \hat{\mathbf{y}}) \tag{2.4}$$

**I.** Z prvej nerovnosti dostávame

$$\forall \mathbf{y} \geq 0_m : \ \ f(\hat{\mathbf{x}}) + \sum_{i=1}^{m} y_i \psi[g_i(\hat{\mathbf{x}})] \leq f(\hat{\mathbf{x}}) + \sum_{i=1}^{m} \hat{y}_i \psi[g_i(\hat{\mathbf{x}})] \tag{2.5}$$

z čoho ďalej vyplýva

$$\forall \mathbf{y} \geq 0_m : \sum_{i=1}^{m} y_i \psi[g_i(\hat{\mathbf{x}})] \leq \sum_{i=1}^{m} \hat{y}_i \psi[g_i(\hat{\mathbf{x}})] \tag{2.6}$$

Na ľavej strane máme lineárnu funkciu a na pravej strane konštantný výraz. Preto koeficienty $\psi[g_i(\hat{\mathbf{x}})]$ lineárnej funkcie nemôžu byť kladné.
Teda,

$$\forall i = 1, \ldots, m : \ \ \psi[g_i(\hat{\mathbf{x}})] \leq 0 \tag{2.7}$$

Z vlastnosti (iii) vyplýva, že (2.7) implikuje

$$\forall i = 1, \ldots, m: \quad g_i(\hat{\mathbf{x}}) \leq 0 \tag{2.8}$$

Tým sme dokázali, že $\hat{\mathbf{x}} \in R^n$ je prípustným riešením úlohy (1.14).
Ďalej ukážeme, že

$$\hat{y}_i \psi[g_i(\hat{\mathbf{x}})] = 0 \qquad (i = 1, \ldots, m) \tag{2.9}$$

Máme $g_i(\hat{\mathbf{x}}) \geq 0$. Z vlastnosti (iii) vyplýva, že $\psi[g_i(\hat{\mathbf{x}})] \leq 0$, a teda aj $\hat{y}_i \psi[g_i(\hat{\mathbf{x}})] \leq 0$. Pripusťme, že $\hat{y}_i \psi[g_i(\hat{\mathbf{x}})] < 0$. Potom pre $\bar{\mathbf{y}} = 0_m$ nerovnosť (2.6) by neplatila. Tým je (2.9) dokázené.
**II.** Z druhej nerovnosti dostávame

$$\forall \mathbf{x} \in R^n \quad : \mathcal{L}(\hat{\mathbf{x}}, \hat{\mathbf{y}}) \leq \mathcal{L}(\mathbf{x}, \hat{\mathbf{y}}) \tag{2.10}$$

čo na základe vzťahu (2.9) možno zjednodušiť na tvar

$$\forall \mathbf{x} \in R^n \quad : f(\hat{\mathbf{x}}) \leq f(\mathbf{x}) + \sum_{i-1}^{m} \hat{y}_i \psi[g_i(\mathbf{x})] \tag{2.11}$$

Ak sa obmedzíme na prípustné riešenia $\mathbf{x} \in K$ úlohy (1.14), tak $g_i(\mathbf{x}) \geq 0$ a $\psi[g_i(\mathbf{x})] \leq 0$. Teda $\sum_{i=1}^{m} \hat{y}_i \psi[g_i(\mathbf{x})] \leq 0$. Potom dostávame

$$\forall \mathbf{x} \in K \quad : f(\hat{\mathbf{x}}) \leq f(\mathbf{x}) + \sum_{i=1}^{m} \hat{y}_i \psi[g_i(\mathbf{x})] \leq f(\mathbf{x}) \tag{2.12}$$

Teda

$$\forall \mathbf{x} \in K \quad : f(\hat{\mathbf{x}}) \leq f(\mathbf{x}) \tag{2.13}$$

čím sme dokázali, že $\hat{\mathbf{x}} \in K$ je optimálnym riešením (1.14). □

**Definícia 2.2.2** *Zložka $\hat{\mathbf{y}} \in R_+^m$ z dvojice $(\hat{\mathbf{x}}, \hat{\mathbf{y}}) \in R^n \times R_+^m$ tvoriacej sedlový bod funkcie* (2.3) *sa nazýva vektor zovšeobecnených Lagrangeových multiplikátorov.*

**Veta 2.2.2** *Majme S-regulárnu úlohu konvexného programovania* (1.14).
*Nech $\hat{\mathbf{x}} \in R^n$ je optimálnym riešením úlohy* (1.14).
*Potom existuje $\hat{\mathbf{y}} \geq 0_m$ tak, že dvojica $(\hat{\mathbf{x}}, \hat{\mathbf{y}}) \in R^n \times R_+^m$ je sedlový bod funkcie* (2.3).

Dôkaz: Vlastnosti (iii), (iv) funkcie $\psi$ umožňujú množinu prípustných riešení úlohy (1.14)

$$K = \{\mathbf{x} \in R^n \mid g_i(\mathbf{x}) \geq 0, \;\; i = 1, \ldots, m\}$$

zapísať v ekvivalentnom tvare (so zachovaním S=regularity)

$$K = \{\mathbf{x} \in R^n | \psi[g_i(\mathbf{x})] \leq 0, \;\; i = 1, \ldots, m\}$$

Z toho vyplýva, že úloha (1.14) je ekvivalentná úlohe

$$\text{Min}\{f(\mathbf{x})|\psi[g_i(\mathbf{x})] \leq 0, \quad i = 1, \ldots, m\}, \tag{2.14}$$

ktorá je tiež úlohou konvexného programovania, pretože vlastnosti (ii), (iii) implikujú konvexnosť funkcií $\psi[g_i(\mathbf{x})]$. No klasická Lagrangeova funkcia pre (2.14) sa zhoduje s (2.3), a teda tvrdenie tejto vety vyplýva zo Slaterovej vety (1.2.2). □

K-T podmienky pre úlohu (2.14) majú tvar:

$$\nabla_x \mathcal{L}(\hat{\mathbf{x}}, \hat{\mathbf{y}}) = \nabla f(\hat{\mathbf{x}}) + \sum_{i=1}^{m} \hat{y}_i \psi'[g_i(\hat{\mathbf{x}})] \nabla g_i(\hat{x}) = 0_n \tag{2.15}$$

$$\psi[g_i(\hat{\mathbf{x}})] \leq 0 \qquad i = 1, \ldots, m \tag{2.16}$$

$$\hat{y}_i \psi[g_i(\hat{\mathbf{x}})] = 0 \qquad i = 1, \ldots, m \tag{2.17}$$

$$\hat{y}_i \geq 0 \qquad i = 1, \ldots, m \tag{2.18}$$

Tieto sú nutnými a zároveň postačujúcimi podmienkami optimality, a v našom prípade aj "sedlovosti" bodu $(\hat{\mathbf{x}}, \hat{\mathbf{y}})$.

Z vlastností (iii) a (iv) vyplýva, že (2.16) a (2.17) môžeme zjednodušiť na tvar

$$g_i(\hat{\mathbf{x}}) \geq 0 \tag{2.19}$$

$$\hat{y}_i g_i(\hat{\mathbf{x}}) = 0 \tag{2.20}$$

a) Ak $g_i(\hat{\mathbf{x}}) > 0$, tak (2.20) implikuje $\hat{y}_i = 0$ (a vtedy aj $\hat{u}_i = 0$).

b) Ak $g_i(\hat{\mathbf{x}}) = 0$, tak podľa (iv) aj $\psi[g_i(\hat{\mathbf{x}})] = 0$ a podľa (v) platí $\psi'[g_i(\hat{\mathbf{x}})] = -1$.

Na základe "strictly complementarity" predpokladu (1.6) platí tiež $\hat{\mathbf{u}} > 0$.
Z hore uvedených vzťahov vzťah (2.15) implikuje

$$\nabla_x \mathcal{L} = \nabla f(\hat{\mathbf{x}}) - \sum_{i=1}^{m} \hat{u}_i \nabla g_i(\hat{\mathbf{x}}) \tag{2.21}$$

Teda platí, že

$$\hat{\mathbf{y}} = \hat{\mathbf{u}} \tag{2.22}$$

## 2.3 Metódy výpočtu sedlového bodu zovšeobecnenej Lagrangeovej funkcie

Našou úlohou je nájsť sedlový bod zovšeobecnenej Lagrangeovej funkcie. Výpočet sedlového bodu $(\hat{\mathbf{x}}, \hat{\mathbf{y}})$ by sa dal realizovať striedavou minimalizáciou tejto funkcie podľa $\mathbf{x} \in R^n$ a

maximalizáciou podľa $\mathbf{y} \in R_+^m$, čiže generovaním postupnosti bodov $\{\mathbf{x^k},\mathbf{y^k}\}_{k=1}^{\infty} \subset R^n \times R_+^m$, kde

$$\mathbf{x^k} = \underset{x \in R^n}{\operatorname{argmin}}\, \mathcal{L}(\mathbf{x}, \mathbf{y^k}) \tag{2.23}$$

a

$$\mathbf{y^{k+1}} = \underset{y \geq 0_m}{\operatorname{argmax}}\, \mathcal{L}(\mathbf{x^k}, \mathbf{y}) \tag{2.24}$$

Pri praktickej realizácii uvedenej iteračnej schémy (2.23), (2.24) sa však musíme uspokojiť s konečnou postupnosťou, ktorá dáva len určitú aproximáciu $(\overline{\mathbf{x}}, \overline{\mathbf{y}}) \in R^n \times R_+^m$ sedlového bodu $(\hat{\mathbf{x}}, \hat{\mathbf{y}})$, charakterizovaného podmienkami (2.15) – (2.18).
Zrejme podmienka (2.23) ja ekvivalentná podmienke

$$\nabla_x \mathcal{L}(\mathbf{x^k}, \mathbf{y^k}) = 0_n, \tag{2.25}$$

ktorú v reálnom algoritme musíme nahradiť podmienkou

$$||\nabla_x \mathcal{L}(\mathbf{x^k}, \mathbf{y^k})||_2 \leq \epsilon \tag{2.26}$$

pre nejaké vopred zadané $\epsilon > 0$.
Analogicky podmienka (2.24) je ekvivalentná s podmienkami

$$\frac{\partial \mathcal{L}}{\partial y_i}(\mathbf{x^k}, \mathbf{y^{k+1}}) \leq 0 \qquad i = 1, \ldots, m \tag{2.27}$$

$$y_i^{k+1} \frac{\partial \mathcal{L}}{\partial y_i}(\mathbf{x^k}, \mathbf{y^{k+1}}) = 0 \qquad i = 1, \ldots, m \tag{2.28}$$

$$y_i^{k+1} \geq 0 \qquad i = 1, \ldots, m, \tag{2.29}$$

ktoré po zderivovaní nadobúdajú tvar

$$\psi[g_i(\mathbf{x^k})] \leq 0 \qquad i = 1, \ldots, m \tag{2.30}$$

$$y_i^{k+1} \psi[g_i(\mathbf{x^k})] = 0 \qquad i = 1, \ldots, m \tag{2.31}$$

$$y_i^{k+1} \geq 0 \qquad i = 1, \ldots, m. \tag{2.32}$$

Ďalej, ak zohľadníme vlastnosti (iii) a (iv) funkcie $\psi$, tak podmienky (2.30) - (2.32) sa ešte zjednodušia:

$$g_i(\mathbf{x^k}) \leq 0 \qquad i = 1, \ldots, m \tag{2.33}$$

$$y_i^{k+1} g_i(\mathbf{x^k}) = 0 \qquad i = 1, \ldots, m \tag{2.34}$$

$$y_i^{k+1} \geq 0 \qquad i = 1, \ldots, m. \tag{2.35}$$

Takto však vznikajú komplikácie, a to hneď z dvoch dôvodov:
(i) podmienka (2.33) nezávisí od premennej $\mathbf{y} \geq 0_m$, a teda ju nevieme zabezpečiť,
(ii) funkcia $G_k(\mathbf{y}) = \mathcal{L}(\mathbf{x^k}, \mathbf{y})$ je lineárna, a preto jej maximum na $R_+^m$ nemusí existovať.

Z týchto dôvodov by bolo vhodnejšie namiesto riešenia úlohy (2.24) konštruovať bod $\mathbf{y^{k+1}} \in R_+^m$, pre ktorý $\mathcal{L}(\mathbf{x^k}, \mathbf{y^{k+1}}) > \mathcal{L}(\mathbf{x^k}, \mathbf{y^k})$.
V praxi sa však tento postup ukázal ako pomaly konvergujúci, a preto na určenie hodnoty $\mathbf{y^{k+1}} \in R_+^m$ sa používa iný "princíp ". Využíva sa fakt, že $\mathbf{y^{k+1}} \approx \mathbf{u^{k+1}}$.
V algoritme v iteračnom procese, pri zafixovanom $\mathbf{y^k} \in R_+^m$ nájdeme bod $\mathbf{x^k} \in R^n$ ako minimum funkcie

$$\mathcal{L}(\mathbf{x}, \mathbf{y^k}),$$

teda platí

$$\nabla f(\mathbf{x^k}) + \sum y_i^k \psi'[g_i(\mathbf{x^k})] \nabla g_i(\mathbf{x^k}) = 0_n. \tag{2.36}$$

Ak položíme

$$y_i^k \psi'[g_i(\mathbf{x^k})] = -u_i^k, \tag{2.37}$$

tak vzťah (2.36) nadobúda tvar

$$\nabla f(\mathbf{x^k}) + \sum u_i^k \nabla g_i(\mathbf{x^k}) = 0_n, \tag{2.38}$$

t.j. spĺňa podmienku $\nabla_x L(\mathbf{x^k}, \mathbf{u^k}) = 0_n$. Inak povedané, pre dané

$$\hat{\mathbf{u}}_\mathbf{i}^\mathbf{k} = -\hat{\mathbf{y}}_\mathbf{i}^\mathbf{k} \psi'[g_i(\hat{\mathbf{x}}^\mathbf{k})] \tag{2.39}$$

je bod $\mathbf{x^k} \in R^n$ tiež bodom minima klasickej Lagrangeovej funkcie $L(\mathbf{x}, \mathbf{u^k})$. To nás privádza k myšlienke použiť vzťah (2.39) na definovanie bodu

$$y_i^{k+1} = -y_i^k \psi'[g_i(\mathbf{x^k})]. \tag{2.40}$$

Skôr než sformulujeme náš algoritmus, musíme ešte zadefinovať podmienky jeho ukončenia.

**Definícia 2.3.1** *Nech sú dané dve konštanty ("malé čísla") $\epsilon > 0$ a $\delta > 0$.*
*Bod $(\bar{\mathbf{x}}, \bar{\mathbf{y}}) \in R^n \times R_+^m$ nazveme $(\epsilon, \delta)$-aproximáciou sedlového bodu $(\hat{\mathbf{x}}, \hat{\mathbf{y}})$ funkcie (2.3), ak sú splnené podmienky:*

$$||\nabla_x \mathcal{L}(\bar{x}, \bar{y})||_2 \leq \epsilon \tag{2.41}$$

$$g_i(\bar{\mathbf{x}}) \geq -\delta \quad i = 1, \ldots, m \tag{2.42}$$

$$g_i(\bar{\mathbf{x}}) > \delta \Rightarrow \bar{y}_i = 0 \quad i = 1, \ldots, m \tag{2.43}$$

$$y_i \geq 0 \quad i = 1, \ldots, m. \tag{2.44}$$

**Definícia 2.3.2** *Ak $(\bar{\mathbf{x}}, \bar{\mathbf{y}}) \in R^n \times R_+^m$ je $(\epsilon, \delta)$-aproximácia sedlového bodu $(\hat{\mathbf{x}}, \hat{\mathbf{y}})$, tak bod $\bar{x} \in R^n$ nazývame $(\epsilon, \delta)$-aproximáciou optimálneho riešenia $\hat{\mathbf{x}} \in R^n$ úlohy (1.14).*

Teraz možno sformulovať ideu nášho algoritmu hľadania $(\epsilon, \delta)$-aproximácie sedlového bodu funkcie (2.3).

1. Vstup: $x^0 \in R^n$, $y^1 \in R^m_+$, $\epsilon > 0$, $\delta > 0$
2. Iteračný cyklus $k = 1, 2, \ldots, k_{max}$

   2 a) Štartujúc z bodu $x^{k-1} \in R^n$ minimalizujeme funkciu $\mathcal{L}(\mathbf{x}, \mathbf{y^k})$ a nájdeme bod $\mathbf{x^k} \in R^n$ spĺňajúci vzťah (2.26).

   2 b) Ak bod $(\mathbf{x^k}, \mathbf{y^{k+1}})$ spĺňa vzťahy (2.42)−(2.44), tak tento bod je $(\epsilon, \delta)$-aproximáciou sedlového bodu $(\hat{\mathbf{x}}, \hat{\mathbf{y}})$ funkcie (2.3). Tým, podľa Definície 2.3.2 sme našli $(\epsilon, \delta)$-aproximáciu optimálneho riešenia úlohy (1.14). (STOP)

   2 c) $y_i^{k+1} = -y_i^k \psi'[g_i(\mathbf{x^k})], \quad i = 1, \ldots, m$

V závislosti od voľby transformačnej funkcie $\psi$ dostávame rôzne konkretizácie uvedeného algoritmu.

## 2.4 Príklady transformačných funkcií $\psi$

Rozšírenou Lagrangeovou funkciou sa zaoberalo mnoho autorov. V článku [12] je uvedený zoznam transformačných funkcií, ktoré používali pri svojich výpočtoch napríklad A. Ben Tal [7], [8], A. Auslender [6], D.P. Bertsekas [9], [10], R.A. Polyak [11] a ďalší. Uvádzame niektoré z nich (rovnako aj ich prvú a druhú deriváciu, odkiaľ ľahko vidieť splnenie spomínaných vlastností funkcie $\psi$):

$$\psi_1(\xi) = \begin{cases} 1 - \sqrt{2\xi + 1} & \text{ak } \xi \geq 0 \\ \frac{1}{2}\xi^2 - \xi & \text{ak } \xi \leq 0 \end{cases} \tag{2.45}$$

$$\psi_1'(\xi) = \begin{cases} \frac{-1}{\sqrt{2\xi+1}} & \text{ak } \xi \geq 0 \\ \xi - 1 & \text{ak } \xi \leq 0 \end{cases}$$

$$\psi_1''(\xi) = \begin{cases} (2\xi + 1)^{-\frac{3}{2}} & \text{ak } \xi \geq 0 \\ 1 & \text{ak } \xi \leq 0 \end{cases}$$

. . . . . .

$$\psi_2(\xi) = \begin{cases} -\frac{\xi}{1+\xi} & \text{ak } \xi \geq 0 \\ \xi^2 - \xi & \text{ak } \xi \leq 0 \end{cases} \tag{2.46}$$

$$\psi_2'(\xi) = \begin{cases} -(1 + \xi)^{-2} & \text{ak } \xi \geq 0 \\ 2\xi - 1 & \text{ak } \xi \leq 0 \end{cases}$$

$$\psi_2''(\xi) = \begin{cases} 2(1+\xi)^{-3} & \text{ak } \xi \geq 0 \\ 2 & \text{ak } \xi \leq 0 \end{cases}$$

. . . . . .

$$\psi_3(\xi) = \begin{cases} -\frac{\xi}{1+\xi} & \text{ak } \xi \geq -\frac{1}{2} \\ (8\xi^2 + 4\xi + 1) & \text{ak } \xi \leq -\frac{1}{2} \end{cases} \tag{2.47}$$

$$\psi_3'(\xi) = \begin{cases} -(1+\xi)^{-2} & \text{ak } \xi \geq -\frac{1}{2} \\ 16\xi + 4 & \text{ak } \xi \leq -\frac{1}{2} \end{cases}$$

$$\psi_3''(\xi) = \begin{cases} 2(1+\xi)^{-3} & \text{ak } \xi \geq -\frac{1}{2} \\ 16 & \text{ak } \xi \leq -\frac{1}{2} \end{cases}$$

. . . . . .

$$\psi_4(\xi) = \begin{cases} -\log(1+\xi) & \text{ak } \xi \geq -\frac{1}{2} \\ (2\xi^2 + \log(2) - \frac{1}{2}) & \text{ak } \xi \leq -\frac{1}{2} \end{cases} \tag{2.48}$$

$$\psi_4'(\xi) = \begin{cases} -\frac{1}{1+\xi} & \text{ak } \xi \geq -\frac{1}{2} \\ 4\xi & \text{ak } \xi \leq -\frac{1}{2} \end{cases}$$

$$\psi_4''(\xi) = \begin{cases} (1+\xi)^{-2} & \text{ak } \xi \geq -\frac{1}{2} \\ 4 & \text{ak } \xi \leq -\frac{1}{2} \end{cases}$$

. . . . . .

$$\psi_5(\xi) = \mathrm{e}^{-\xi} - 1 \tag{2.49}$$

$$\psi_5'(\xi) = -\mathrm{e}^{-\xi}$$

$$\psi_5''(\xi) = \mathrm{e}^{-\xi}$$

. . . . . .

$$\psi_6(\xi) = \begin{cases} -\log(1+\xi) & \text{ak } \xi \geq -\frac{1}{2} \\ \mathrm{e}^{-2\xi-1} + \ln(2) - 1 & \text{ak } \xi \leq -\frac{1}{2} \end{cases} \tag{2.50}$$

$$\psi_6'(\xi) = \begin{cases} -\frac{1}{1+\xi} & \text{ak } \xi \geq -\frac{1}{2} \\ -2\mathrm{e}^{-2\xi-1} & \text{ak } \xi \leq -\frac{1}{2} \end{cases}$$

$$\psi_6''(\xi) = \begin{cases} (1+\xi)^{-2} & \text{ak } \xi \geq -\frac{1}{2} \\ 4\mathrm{e}^{-2\xi-1} & \text{ak } \xi \leq -\frac{1}{2} \end{cases}$$

. . . . . .

$$\psi_7(\xi) = \begin{cases} -\frac{\xi}{1+\xi} & \text{ak } \xi \geq -\frac{1}{2} \\ \mathrm{e}^{-4\xi-2} & \text{ak } \xi \leq -\frac{1}{2} \end{cases} \tag{2.51}$$

$$\psi_7'(\xi) = \begin{cases} -(1+\xi)^{-2} & \text{ak } \xi \geq -\frac{1}{2} \\ -4\mathrm{e}^{-4\xi-2} & \text{ak } \xi \leq -\frac{1}{2} \end{cases}$$

$$\psi_7''(\xi) = \begin{cases} 2(1+\xi)^{-3} & \text{ak } \xi \geq -\frac{1}{2} \\ 16\mathrm{e}^{-4\xi-2} & \text{ak } \xi \leq -\frac{1}{2} \end{cases}$$

. . . . . .

$$\psi_8(\xi) = \begin{cases} -\frac{1}{4}\log 2\xi - \frac{3}{8} & \text{ak } \xi \geq \frac{1}{2} \\ \frac{\xi^2}{2} - \xi & \text{ak } \xi \leq \frac{1}{2} \end{cases} \tag{2.52}$$

$$\psi_8'(\xi) = \begin{cases} -\frac{1}{4\xi} & \text{ak } \xi \geq \frac{1}{2} \\ \xi - 1 & \text{ak } \xi \leq \frac{1}{2} \end{cases}$$

$$\psi_8''(\xi) = \begin{cases} \frac{1}{4\xi^2} & \text{ak } \xi \geq \frac{1}{2} \\ 1 & \text{ak } \xi \leq \frac{1}{2} \end{cases}$$

# Kapitola 3

# Experimentálna časť

V tejto kapitole aplikujeme navrhnutý algoritmus (str. 20) na riešenie troch konkrétnych typov úloh konvexného programovania. Najprv budeme riešiť klasickú úlohu kvadratického programovania (3.1). Potom ku kvadratickej účelovej funkcii pridáme špeciálny logaritmický výraz, čím dostaneme úlohu (3.6). Nakoniec v úlohe (3.7) pridáme do účelovej funkcie exponenciálny výraz.

Uvedené tri typy úloh budeme riešiť pre rôzne rozmery: počet premenných $n$ od 40 do 150, počet ohraničení $m$ od 50 do 200. Úlohy vytvárame pomocou generátora pseudonáhodných čísel.

Naším cieľom je analyzovať numerické vlastnosti navrhnutého algoritmu. Je vhodné riešiť také úlohy, ktorých optimálne riešenie vopred poznáme. Na tento účel (v prípade úlohy konvexného programovania) nám poslúžia K-T podmienky. Pomocou nich špeciálne skonštruujeme tri algoritmy na generovanie úloh (3.1), (3.6), (3.7) s vopred zadaným optimálnym riešením.

## 3.1 Generátory úloh

### 3.1.1 Generátor 1

Chceme generovať úlohy kvadratického programovania tvaru

$$\mathrm{Min}\left\{\frac{1}{2}\mathbf{x^T}G\mathbf{x}+\mathbf{h^T}\mathbf{x} \mid A\mathbf{x}-\mathbf{b}\leq 0\right\}, \tag{3.1}$$

kde $G = G^T$ je kladne definitná matica $n \times n$, $A$ je matica ohraničení $m \times n$, $\mathbf{h} \in R^n$, $\mathbf{b} \in R^m$.

Zároveň chceme vopred zadať optimálne riešenie $\hat{\mathbf{x}} \in R^n$ aj vektory Lagrangeových multiplikátorov $\hat{\mathbf{u}} \geq 0_m$.

K-T podmienky pre túto úlohu vyzerajú nasledovne:

$$G\hat{\mathbf{x}} + \mathbf{h} - A^T\hat{\mathbf{u}} = 0_n \tag{3.2}$$

$$A\hat{\mathbf{x}} - \mathbf{b} \geq 0_m \tag{3.3}$$

$$\hat{\mathbf{u}}^T(A\hat{\mathbf{x}} - \mathbf{b}) = 0 \tag{3.4}$$

$$\hat{\mathbf{u}} \geq 0 \tag{3.5}$$

Z tvaru K-T podmienok vidíme, že matice $G$, $A$ a vektory $\hat{\mathbf{x}}$, $\hat{\mathbf{u}}$ môžeme zvoliť ľubovoľne a potom príslušné vektory $\mathbf{h}$, $\mathbf{b}$ dopočítať.

Takýto postup však ešte nezaručuje jednoznačnosť optimálneho riešenia. Aby sme zaručili jednoznačnosť sedlového bodu $(\hat{\mathbf{x}}, \hat{\mathbf{u}})$ Lagrangeovej funkcie, musíme ešte zabezpečiť splnenie vlastnosti "stictly complementarity" (1.6).

**Definícia 3.1.1** *Ohraničenie $g_i(\mathbf{x}) \geq 0$ v bode $\bar{x} \in R^n$ sa nazýva **aktívnym ohraničením**, ak $g_i(\bar{x}) = 0$.*

Generátor sme konštruovali nasledovne:

1. Vstupom do generátora je rozmer úlohy, čiže počet premenných $n$ a počet ohraničení $m$, a zároveň aj počet aktívnych ohraničení $m_a$ v optimálnom riešení $\hat{\mathbf{x}} \in R^n$.
2. Náhodne vygenerujeme maticu $A_{m\times n}$ ako celočíselnú maticu s prvkami $-5 \leq a_{ij} \leq 5$.
3. Symetrickú kladne definitnú maticu $G_{n\times n}$ utvoríme podľa vzorca $G = BB^T + I$, kde $B_{n\times n}$ je ľubovoľná celočíselná matica s prvkami $-3 \leq b_{ij} \leq 3$ a $I_{n\times n}$ je jednotková matica.
4. Náhodne vygenerujeme optimálne riešenie $\hat{\mathbf{x}} \in R^n$, s celočíselnými prvkami $-9 \leq \hat{x_i} \leq 9$.
5. Následne vygenerujeme celočíselný vektor Lagrangeových multiplikátorov $\hat{\mathbf{u}} \geq 0$ s prvkami $0 \leq \hat{u_i} \leq 9$, pričom počet kladných $\hat{u_i}$ sa rovná číslu $m_a$ (tj. počet nulových $\hat{u_i}$ sa rovná číslu $m - m_a$).
6. Dopočítame vektor $\mathbf{b} \in R^m$ podľa vzorca

$$b_i = \begin{cases} (A\hat{\mathbf{x}})_i & \text{ak } \hat{u_i} > 0 \\ (A\hat{\mathbf{x}})_i - \beta_i & \text{ak } \hat{u_i} = 0 \end{cases}$$

   kde $\beta_i$ sú náhodné celé čísla $0 < \beta_i \leq 3$.
7. Dorátame vektor $\mathbf{h} = A^T\hat{\mathbf{u}} - G\hat{\mathbf{x}}$

Poznamenávame, že podľa (2.22) platí $\hat{\mathbf{y}} = \hat{\mathbf{u}}$.

### 3.1.2 Generátor 2

V tomto prípade sme konštruovali úlohu konvexného programovania v nasledovnom tvare:

$$\text{Min}\left\{\frac{1}{2}\mathbf{x^T}G\mathbf{x}+\mathbf{h^T}\mathbf{x}+\sum_{j=1}^{n}x_j\ln(x_j)\mid A\mathbf{x}-\mathbf{b}\geq 0\right\}. \tag{3.6}$$

Postup bol podobný ako v prípade Generátora 1, s nasledovnými rozdielmi:

- na základe definičného oboru účeovej funkcie prvky optimálneho riešenia $\hat{\mathbf{x}}$ sú celé kladné čísla: $0 < x_i < 9$
- druhá K-T podmienka (3.2) má tvar $G\hat{\mathbf{x}}+\mathbf{h}+\mathbf{e}+\log(\hat{\mathbf{x}})-A^T\hat{\mathbf{u}}=0_n$, kde $\mathbf{e}$ je jednotkový vektor $n\times 1$ a $\log(\hat{\mathbf{x}})=(\log(x_1),\log(x_2),\ldots,\log(x_n))^T$. Vektor $\mathbf{h}$ teda dorátame ako $\mathbf{h}=A^T\hat{\mathbf{u}}-(G\hat{\mathbf{x}}+\mathbf{e}+\log(\hat{\mathbf{x}}))$ .

### 3.1.3 Generátor 3

Ako ďalšiu sme si sformulovali nasledovnú úlohu:

$$\text{Min}\left\{\frac{1}{2}\mathbf{x^T}G\mathbf{x}+\mathbf{h^T}\mathbf{x}+e^{\mathbf{c^T}\mathbf{x}}\mid A\mathbf{x}-\mathbf{b}\geq 0\right\}. \tag{3.7}$$

Generátor 3 sme konštruovali podobne ako v prípade Generátora 1, s nasledovnými odlišnosťami:

- ako súčasť účelovej funkcie si musíme vygenerovať ešte vektor $\mathbf{c}, -5\leq c_i\leq 5$
- druhá K-T podmienka (3.2) má tvar $G\hat{\mathbf{x}}+\mathbf{h}+\mathbf{c}e^{\mathbf{c^T}\mathbf{x}}-A^T\hat{\mathbf{u}}=0_n$, a teda $\mathbf{h}=A^T\hat{\mathbf{u}}-(G\hat{\mathbf{x}}+\mathbf{c}e^{\mathbf{c^T}\mathbf{x}})$

Po sformulovaní a vygenerovaní úloh pokračujeme ich riešením.

## 3.2 Popis algoritmov

### 3.2.1 Algoritmus hľadania sedlového bodu

Idea algoritmu bola sformulovaná na strane 20. Teraz upresníme niektoré detaily:

**1.** Vstupom do algoritmu budú štartovacie body $\mathbf{x}^0 \in R^n$, $\mathbf{y}^1 \geq 0$ (tieto si vygenerujeme pomocou konštánt $\rho > 0$ a $\sigma > 0$, tak, aby platilo: $||\hat{\mathbf{x}} - \mathbf{x^0}|| = \rho$, $||\hat{\mathbf{y}} - \mathbf{y^1}|| = \sigma$) a tolerančné konštanty $\epsilon > 0$ a $\delta > 0$. Zároveň nastavíme "počítadlo" makroiterácií $k = 1$.

**2.** Spustíme cyklus makroiterácií:

2a) Riešime pomocnú úlohu

$$\text{Min}\{F_k(\mathbf{x}) = \mathcal{L}(\mathbf{y}^k, \mathbf{x}) | \mathbf{x} \in R^n\} \tag{3.8}$$

Ako štartovací bod použijeme bod $\mathbf{x^{k-1}} \in R^n$ a úlohu budeme riešiť pomocou modifikovanej Newtonovej metódy so stopovacím kritérim $||\nabla F(\mathbf{x}^k)|| \leq \epsilon$.

2b) Testujeme dosiahnutú $\delta$-presnosť "sedlového bodu"$(\mathbf{x}^k, \mathbf{y}^k)$:
Ak sú splnené podmienky (2.42), (2.43) a (2.44), bod $\mathbf{x^k}, \mathbf{y^k}$ je $(\epsilon, \delta)$-aproximáciou sedlového bodu a algoritmus zastavíme.

2c) Utvoríme novú aproximáciu vektora $\hat{\mathbf{y}} \geq 0_m$:
$y_i^{k+1} = -y_i^k \psi'(g_i(\mathbf{x}^k))$.

2d) $k = k + 1$.

### 3.2.2 Algoritmus modifikovanej Newtonovej metódy

Newtonova metóda slúži na nájdenie minima funkcie nasledovným spôsobom (riešime úlohu (3.8)):

**1.** Do Newtonovej metódy vstupujú štartovací bod ($\mathbf{x}^0 \in R^n$), tolerančná konštanta $\epsilon$ a "počítadlo"Newtonovych iterácií $l = 0$.

**2.** Spustíme cyklus Newtonových iterácií:

2a) Vypočítame gradient $\nabla F_k(\mathbf{x}^l)$ účelovej funkcie $F_k(\mathbf{x})$.

2b) Test presnosti: Ak $||\nabla F_k(\mathbf{x}^l)|| < \epsilon$, Newtonovu metódu zastavíme, a $\mathbf{x}^l$ je $\epsilon$-približným riešením úlohy.

2c) Vypočítame Hessovu maticu $\nabla^2 F_k(\mathbf{x}^l)$funkcie $F_k(\mathbf{x})$.

2d) Riešime sústavu lineárnych rovníc $\nabla^2 F_k(\mathbf{x}^l)\mathbf{s} = -\nabla F_k(\mathbf{x}^l)$. Riešenie tejto sústavy označíme $\mathbf{s}^l \in R^n$.

2e) Metódou zlatého rezu[1] minimalizujeme funkciu $\phi(\lambda) = \mathcal{L}(\mathbf{x}^l + \lambda \mathbf{s}^l, \mathbf{y}^k)$ a výsledné minimum označíme $\lambda_l$.

2f) $\mathbf{x}^{l+1} = \mathbf{x}^l + \lambda_l \mathbf{s}^l$.

2g) $l = l + 1$.

## 3.3 Ilustratívne riešenie

V našich numerických experimentoch sme používali štyri rôzne transformačné funkcie $\psi(\xi)$ - (2.45), (2.46), (2.47) a (2.48). V tomto poradí ich budeme označovať ako funkcie 1 - 4. Všetky pozorovania rôznych funkcií 1 - 4 prebiehali na rovnakých úlohách konvexného programovania, aby bolo možné objektívne zhodnotiť výsledky. Porovnávali sme počet makroiterácií v algoritme, celkový počet iterácií Newtonovej metódy v celom algoritme a čas trvania výpočtu. Pozorovali sme vplyv počtu aktívnych ohraničení na vývoj spomínaných hodnôt pri úlohách rôznych rozmerov.

Funkcie sme vybrali tak, aby sme mohli porovnať rôzne typy - funkcia 1 obsahuje v jednej vetve odmocninu, druhá vetva je parabola, spojené sú v bode 0. Funkcia 2 spája v bode 0 parabolu a hyperbolu. Funkcia 3 je rovnako spojením paraboly a hyperboly, no v bode $-\frac{1}{2}$, čo nie je práve najtypickejší bod, a teda budeme môcť pozorovať, ako tento netradičný "bod zlepenia" ovplyvní správanie nášho algoritmu. Funkcia 4 spája kvadratickú funkciu (parabolu) a zápornú logaritmickú funkciu, opäť v netradičnom bode $-\frac{1}{2}$.

V ilustratívnom riešení sme sa podrobnejšie zaoberali výstupom algoritmu. V každej makroiterácii sme sledovali vzdialenosť bodov $x^k$ a $y^k$ od optimálneho riešenia, rozdiel hodnôt Lagrangeovej funkcie vo vypočítanom bode a v optimálnom riešení, počet Newtonových iterácií a počty aktívnych a neaktívnych ohraničení (presnejšie ohraničení, ktorých hodnota je menšia ako tolerančná konštanta $-\delta$, väčia ako $\delta$ a hodnoty pohybujúce sa medzi týmito dvoma hranicami). Uvádzame podrobný výstup z algoritmu, ktorý riešil rovnakú úlohu kvadratického programovania (vygenerovanú Generátorom 1, tj. kvadratickú funkciu) s použitím každej penalizačnej funkcie zvlášť. V tomto pozorovaní sme zadali počet premenných $n = 50$, počet ohraničení $m = 70$, počet aktívnych ohraničení $m_a = 25$, vzdialenosť počiatočného bodu $\mathbf{x^0}$ od optimálneho riešenia $\rho = 10$, vzdialenosť $\mathbf{y^1}$ od optimálneho riešenia $\sigma = 10$, tolerančné konštanty $\epsilon = 10^{-3}$ a $\delta = 10^{-3}$.

[1] Podrobnejší popis metódy zlatého rezu je možné nájsť v [1]

Tabuľka 3.1: Tabuľka výstupov pre transformačnú funkciu 1 - ilustratívne riešenie

| číslo makroit. | $\|\|x^k - \hat{x}\|\|$ | $\|\|y^k - \hat{y}\|\|$ | $\mathcal{L}(x^k, y^k) - \mathcal{L}(\hat{x}, \hat{y})$ | Počet Newt. iterácií | Počet i: $g(x_i) > \delta$ | Počet i: $g(x_i) < \delta$ | Počet i: $g(x_i) < -\delta$ |
|---|---|---|---|---|---|---|---|
| 1 | 3,87E-01 | 1,00E+01 | -2,32E+02 | 6 | 57 | 1 | 12 |
| 2 | 5,08E-02 | 2,48E+00 | -4,25E+01 | 4 | 57 | 0 | 13 |
| 3 | 1,32E-02 | 6,57E-01 | -8,15E+00 | 4 | 56 | 0 | 14 |
| 4 | 4,78E-03 | 1,95E-01 | -1,57E+00 | 4 | 57 | 6 | 7 |
| 5 | 1,88E-03 | 6,69E-02 | -3,05E-01 | 4 | 54 | 1 | 6 |
| 6 | 9,98E-04 | 2,58E-02 | -5,95E-02 | 4 | 49 | 19 | 2 |
| 7 | 3,09E-04 | 9,81E-03 | -1,16E-02 | 4 | 45 | 24 | 1 |
| 8 | 9,58E-05 | 3,48E-03 | -2,28E-03 | 4 | 45 | 25 | 0 |
| 9 | 8,15E-05 | 1,32E-03 | -4,48E-04 | 4 | 45 | 25 | 0 |

Tabuľka 3.2: Tabuľka výstupov pre transformačnú funkciu 2 - ilustratívne riešenie

| číslo makroit. | $\|\|x^k - \hat{x}\|\|$ | $\|\|y^k - \hat{y}\|\|$ | $\mathcal{L}(x^k, y^k) - \mathcal{L}(\hat{x}, \hat{y})$ | Počet Newt. iterácií | Počet i: $g(x_i) > \delta$ | Počet i: $g(x_i) < \delta$ | Počet i: $g(x_i) < -\delta$ |
|---|---|---|---|---|---|---|---|
| 1 | 5,45E-02 | 1,00E+01 | -5,12E+01 | 6 | 55 | 2 | 13 |
| 2 | 2,68E-03 | 1,71E-01 | -2,60E-01 | 4 | 53 | 7 | 10 |
| 3 | 5,70E-04 | 3,17E-02 | -1,48E-03 | 4 | 49 | 19 | 2 |
| 4 | 1,39E-04 | 7,53E-03 | -1,10E-05 | 4 | 45 | 25 | 0 |

Tabuľka 3.3: Tabuľka výstupov pre transformačnú funkciu 3 - ilustratívne riešenie

| číslo makroit. | $\|\|x^k - \hat{x}\|\|$ | $\|\|y^k - \hat{y}\|\|$ | $\mathcal{L}(x^k, y^k) - \mathcal{L}(\hat{x}, \hat{y})$ | Počet Newt. iterácií | Počet i: $g(x_i) > \delta$ | Počet i: $g(x_i) < \delta$ | Počet i: $g(x_i) < -\delta$ |
|---|---|---|---|---|---|---|---|
| 1 | 5,40E-02 | 1,00E+01 | -5,12E+01 | 10 | 55 | 2 | 13 |
| 2 | 2,66E-03 | 1,74E-01 | -2,60E-01 | 7 | 52 | 9 | 9 |
| 3 | 5,59E-04 | 3,38E-02 | -1,48E-03 | 6 | 48 | 20 | 2 |
| 4 | 4,18E-04 | 1,75E-02 | -2,76E-06 | 7 | 47 | 23 | 0 |
| 5 | 1,18E-04 | 2,05E-03 | 4,26E-06 | 7 | 45 | 25 | 0 |

Tabuľka 3.4: Tabuľka výstupov pre transformačnú funkciu 4 - ilustratívne riešenie

| číslo makroit. | $\|\|x^k - \hat{x}\|\|$ | $\|\|y^k - \hat{y}\|\|$ | $\mathcal{L}(x^k, y^k) - \mathcal{L}(\hat{x}, \hat{y})$ | Počet Newt. iterácií | Počet i: $g(x_i) > \delta$ | Počet i: $g(x_i) < \delta$ | Počet i: $g(x_i) < -\delta$ |
|---|---|---|---|---|---|---|---|
| 1 | 1,81E-01 | 1,00E+01 | -1,50E+02 | 8 | 56 | 0 | 14 |
| 2 | 2,29E-02 | 1,22E+00 | -1,06E+01 | 7 | 59 | 0 | 11 |
| 3 | 4,95E-03 | 2,08E-01 | -7,87E-01 | 7 | 55 | 5 | 10 |
| 4 | 1,48E-03 | 5,13E-02 | -5,92E-02 | 6 | 54 | 14 | 2 |
| 5 | 3,47E-04 | 1,57E-02 | -4,50E-03 | 6 | 47 | 23 | 0 |
| 6 | 1,53E-04 | 5,51E-03 | -3,42E-04 | 6 | 45 | 25 | 0 |

Z priložených tabuliek je vidieť, že náš algoritmus výsledky vrámci požadovanej presnosti. V poslednom riadku každej tabuľky je uvedená vzdialenosť konečného vypočítaného riešenia od optimálneho riešenia $\hat{\mathbf{x}}$, ako aj presnosti vypočítanehého vektora zovšeobecnených Lagrangeových multiplikátorov a funkčnej hodnoty zovšeobecnenej Lagrangeovej funkcie.

Pri porovnaní výsledkov v Tab. 3.1 - Tab. 3.4 je vidieť, že druhá funkcia pracovala najrýchlejšie čo sa počtu makroiterácií týka (Tab. 3.2 má iba štyri riadky, prebehli iba štyri makroiterácie), za ňou nasleduje tretia, štvrtá a nakoniec prvá funkcia (sledovali sme prvý stĺpec pri všetkých štyroch uvedených tabuľkách). Čo sa týka počtu Newtonových iterácií, najpomalšia bola funkcia 3. Tieto intuitívne odhady sme overovali v nasledujúcej časti, kde sme sledovali už spomínané parametre a ich vplyv na jednotlivé typy funkcií.

## 3.4 Experimentálne výsledky

Všetky výsledky v nasledujúcich tabuľkách boli získané ako aritmetický priemer hodnôt výsledkov z 50 úloh (pri porovnávaní transformačných funkcií pre jednu účelovú funkciu išlo vždy o rovnaké úlohy). Menili sme počet premenných, počet ohraničení a pozorovali sme vplyv zmeny počtu aktívnych ohraničení. Ďalšie vstupné hodnoty sme zadávali v každom pozorovaní rovnako: $\rho = 10$, $\sigma = 10$, $\epsilon = 10^{-3}$ a $\delta = 10^{-3}$. Pozorovanie sme vykonávali pri troch rôznych rozmeroch úloh (počet premenných $n$ a počet ohraničení $m$):

- $n = 40$, $m = 50$
- $n = 80$, $m = 100$
- $n = 150$, $m = 200$

### 3.4.1 Riešenie úloh (3.1)

#### Úlohy s rozmermi $n = 40$, $m = 50$

Pri všetkých štyroch transformačných funkciách sme menili počet aktívnych ohraničení ($m_a = 10, 20, 30$). Do tabuliek sme zaznamenali vplyv na pozorované hodnoty výstupov.

Tabuľka 3.5: Tabuľka výstupov pre transformačnú funkciu 1 pri rozmeroch úlohy $n = 40$, $m = 50$ - riešenie úloh (3.1)

| Počet aktívnych ohraničení $m_a$ | Počet makroiterácií | Celkový počet Newtonových iterácií | Čas výpočtu v sekundách |
|---|---|---|---|
| 10 | 8,00 | 17,32 | 0,19 |
| 20 | 8,36 | 17,48 | 0,21 |
| 30 | 14,28 | 24,62 | 0,28 |

Tabuľka 3.6: Tabuľka výstupov pre transformačnú funkciu 2 pri rozmeroch úlohy $n = 40$, $m = 50$ - riešenie úloh (3.1)

| Počet aktívnych ohraničení $m_a$ | Počet makroiterácií | Celkový počet Newtonových iterácií | Čas výpočtu v sekundách |
|---|---|---|---|
| 10 | 5,74 | 19,84 | 0,12 |
| 20 | 6,18 | 19,18 | 0,12 |
| 30 | 10,18 | 25,24 | 0,16 |

Tabuľka 3.7: Tabuľka výstupov pre transformačnú funkciu 3 pri rozmeroch úlohy $n = 40$, $m = 50$ - riešenie úloh (3.1)

| Počet aktívnych ohraničení $m_a$ | Počet makroiterácií | Celkový počet Newtonových iterácií | Čas výpočtu v sekundách |
|---|---|---|---|
| 10 | 4,32 | 13,54 | 0,08 |
| 20 | 4,48 | 14,40 | 0,09 |
| 30 | 7,52 | 18,06 | 0,11 |

Tabuľka 3.8: Tabuľka výstupov pre transformačnú funkciu 4 pri rozmeroch úlohy $n = 40$, $m = 50$ - riešenie úloh (3.1)

| Počet aktívnych ohraničení $m_a$ | Počet makroiterácií | Celkový počet Newtonových iterácií | Čas výpočtu v sekundách |
|---|---|---|---|
| 10 | 8,02 | 16,86 | 0,18 |
| 20 | 7,74 | 17,10 | 0,18 |
| 30 | 11,82 | 21,30 | 0,24 |

Na základe vzájomného porovnania uvedených tabuliek sú časovo najkratšie výpočty s použitím transformačnej funkcie 3, a to nielen z pohľadu času, ale aj vzhľadom na počet iterácií (makroiterácií aj Newtonových iterácií). Najvyšší čas výpočtu sme dosiahli pri použití funkcie 1. Druhá funkcia zaznamenala najvyšší počet Newtonových iterácií. Zároveň môžme pozorovať, že pri zvyšovaní počtu aktívnych ohraničení sa zväčšuje aj počet makroiterácií a čas výpočtu. Nakoľko však rozpätie menených hodnôt bolo dosť malé, vykonali sme experiment s väčšími rozmermi, kde si tieto predpoklady môžme lepšie overiť.

### Úlohy s rozmermi $n = 80$, $m = 100$

Tabuľka 3.9: Tabuľka výstupov pre transformačnú funkciu 1 pri rozmeroch úlohy $n = 80$, $m = 100$ - riešenie úloh (3.1)

| Počet aktívnych ohraničení $m_a$ | Počet makroiterácií | Celkový počet Newtonových iterácií | Čas výpočtu v sekundách |
|---|---|---|---|
| 10 | 9,02 | 16,72 | 0,29 |
| 20 | 9,24 | 17,44 | 0,30 |
| 30 | 9,74 | 18,30 | 0,32 |
| 40 | 9,28 | 17,32 | 0,30 |
| 50 | 9,90 | 18,16 | 0,31 |
| 60 | 13,58 | 20,78 | 0,38 |
| 70 | 25,24 | 28,28 | 0,60 |

Tabuľka 3.10: Tabuľka výstupov pre transformačnú funkciu 2 pri rozmeroch úlohy $n = 80$, $m = 100$ - riešenie úloh (3.1)

| Počet aktívnych ohraničení $m_a$ | Počet makroiterácií | Celkový počet Newtonových iterácií | Čas výpočtu v sekundách |
|---|---|---|---|
| 10 | 5,64 | 24,08 | 0,16 |
| 20 | 5,50 | 21,24 | 0,15 |
| 30 | 6,42 | 19,42 | 0,14 |
| 40 | 6,54 | 20,72 | 0,17 |
| 50 | 7,20 | 23,46 | 0,18 |
| 60 | 8,24 | 26,38 | 0,18 |
| 70 | 9,00 | 23,00 | 0,19 |

Tabuľka 3.11: Tabuľka výstupov pre transformačnú funkciu 3 pri rozmeroch úlohy $n = 80$, $m = 100$ - riešenie úloh (3.1)

| Počet aktívnych ohraničení $m_a$ | Počet makroiterácií | Celkový počet Newtonových iterácií | Čas výpočtu v sekundách |
|---|---|---|---|
| 10 | 4,66 | 14,72 | 0,11 |
| 20 | 4,40 | 14,06 | 0,10 |
| 30 | 4,22 | 13,56 | 0,10 |
| 40 | 5,10 | 15,36 | 0,11 |
| 50 | 4,68 | 15,42 | 0,11 |
| 60 | 7,80 | 17,72 | 0,14 |
| 70 | 12,68 | 21,36 | 0,18 |

Tabuľka 3.12: Tabuľka výstupov pre transformačnú funkciu 4 pri rozmeroch úlohy $n = 80$, $m = 100$ - riešenie úloh (3.1)

| Počet aktívnych ohraničení $m_a$ | Počet makroiterácií | Celkový počet Newtonových iterácií | Čas výpočtu v sekundách |
|---|---|---|---|
| 10 | 6,36 | 15,32 | 0,25 |
| 20 | 6,24 | 15,32 | 0,24 |
| 30 | 6,58 | 16,24 | 0,26 |
| 40 | 6,50 | 16,54 | 0,26 |
| 50 | 8,06 | 17,88 | 0,29 |
| 60 | 12,86 | 21,46 | 0,38 |
| 70 | 21,62 | 26,86 | 0,53 |

Pri tomto pozorovaní najrýchlejšie prebehol výpočet pri použití transformačnej funkcie 3 (vzhľadom na výpočtový čas aj na počet makroiterácií). Naopak, najhoršie výsledky sme zaznamenali pri použití funkcie 1. Počet Newtonových iterácií bol najmenší pri výpočtoch s použitím funkcie 3, najväčší pri funkcii 2. Tieto pozorovania sme overili na úlohách s ešte väčšími rozmermi.

Ako sme predpokladali, zvýšením počtu aktívnych ohraničení sa zvyšuje aj čas výpočtu, rovnako ako počet makroiterácií. Tieto závery sa nám potrvdili vo všetkých experimentoch, preto sme sa ďalej zamerali na vzájomné porovnávanie transformačných funkcií, ako aj na porovnávanie výsledkov pre rôzne účelové funkcie.

### Úlohy s rozmermi $n = 150$, $m = 200$

Tabuľka 3.13: Tabuľka výstupov pre transformačnú funkciu 1 pri rozmeroch úlohy $n = 150$, $m = 200$ - riešenie úloh (3.1)

| Počet aktívnych ohraničení $m_a$ | Počet makroiterácií | Celkový počet Newtonových iterácií | Čas výpočtu v sekundách |
|---|---|---|---|
| 10 | 9,00 | 16,74 | 0,52 |
| 20 | 9,58 | 16,52 | 0,52 |
| 30 | 9,28 | 15,68 | 0,50 |
| 40 | 9,74 | 16,02 | 0,52 |
| 50 | 10,22 | 17,50 | 0,55 |
| 60 | 10,38 | 17,76 | 0,56 |
| 70 | 10,36 | 17,48 | 0,56 |
| 80 | 10,28 | 17,40 | 0,55 |
| 90 | 10,32 | 17,70 | 0,57 |
| 100 | 10,46 | 17,78 | 0,56 |
| 110 | 12,14 | 19,20 | 0,62 |
| 120 | 15,56 | 21,40 | 0,73 |
| 130 | 21,68 | 24,48 | 0,91 |
| 140 | 56,80 | 45,40 | 2,02 |

Pri porovnaní výsledkov použitia transformačnej funkcie 1 v úlohách s rôznymi rozmermi (Tab. 3.4, Tab. 3.9 a Tab. 3.13) vidíme, že pri malom počte aktívnych ohraničení je počet makroiterácií 8, resp. 9, pri zvyšovaní počtu aktívnych ohraničení sa zväčšuje aj počet makroiterácií. Čas výpočtu závisí od rozmeru úlohy (dalo sa predpokladať), ale aj od počtu aktívnych ohraničení, čo však súvisí aj s nárastom počtu iterácií.

Tabuľka 3.14: Tabuľka výstupov pre transformačnú funkciu 2 pri rozmeroch úlohy $n = 150$, $m = 200$ - riešenie úloh (3.1)

| Počet aktívnych ohraničení $m_a$ | Počet makroiterácií | Celkový počet Newtonových iterácií | Čas výpočtu v sekundách |
|---|---|---|---|
| 10 | 4,30 | 26,14 | 0,25 |
| 20 | 5,40 | 22,60 | 0,23 |
| 30 | 5,50 | 24,00 | 0,25 |
| 40 | 6,00 | 24,60 | 0,27 |
| 50 | 5,78 | 24,86 | 0,27 |
| 60 | 6,12 | 25,10 | 0,28 |
| 70 | 5,80 | 24,78 | 0,27 |
| 80 | 5,76 | 24,68 | 0,27 |
| 90 | 6,18 | 25,56 | 0,28 |
| 100 | 6,92 | 25,84 | 0,29 |
| 110 | 8,94 | 27,38 | 0,34 |
| 120 | 11,78 | 29,20 | 0,42 |
| 130 | 16,98 | 32,14 | 0,55 |
| 140 | 40,04 | 41,30 | 1,23 |

Ak porovnáme výsledky z tabuliek, kde sme použili transformačnú funkciu 2 (Tab. 3.5, Tab. 3.10 a Tab. 3.14), je viditeľné, že pri zväčšení rozmerov úlohy sa zvýši výpočtový čas. Počet makroiterácií sa však pri rovnakých počtoch aktívnych ohraničení výrazne nemení (ani pri rôznych rozmeroch úlohy), rastie s narastajúcim počtom $m_a$.

Tabuľka 3.15: Tabuľka výstupov pre transformačnú funkciu 3 pri rozmeroch úlohy $n = 150$, $m = 200$ - riešenie úloh (3.1)

| Počet aktívnych ohraničení $m_a$ | Počet makroiterácií | Celkový počet Newtonových iterácií | Čas výpočtu v sekundách |
|---|---|---|---|
| 10 | 4,28 | 14,30 | 0,15 |
| 20 | 4,76 | 14,54 | 0,16 |
| 30 | 4,74 | 14,36 | 0,16 |
| 40 | 4,64 | 14,46 | 0,16 |
| 50 | 4,86 | 14,38 | 0,16 |
| 60 | 4,48 | 13,92 | 0,15 |
| 70 | 4,40 | 14,26 | 0,15 |
| 80 | 4,18 | 14,34 | 0,15 |
| 90 | 4,68 | 15,68 | 0,16 |
| 100 | 5,40 | 15,88 | 0,17 |
| 110 | 6,92 | 17,16 | 0,19 |
| 120 | 8,60 | 18,14 | 0,22 |
| 130 | 12,66 | 21,14 | 0,27 |
| 140 | 17,36 | 23,60 | 0,33 |

Aj pri použití transformačnej funkcie 3 (Tab. 3.6, Tab. 3.11 a Tab. 3.15) môžme pozorovať, že so zväčšujúcimi sa rozmermi úlohy sa zvyšuje výpočtový čas. Počet makroiterácií pre $m_a = 10$ je pri všetkých rozmeroch úlohy približne rovnaký. So zvyšovaním počtu aktívnych ohraničení však počet makroiterácii už pre všetky ostatné prípady rastie.

Tabuľka 3.16: Tabuľka výstupov pre transformačnú funkciu 4 pri rozmeroch úlohy $n = 150$, $m = 200$ - riešenie úloh (3.1)

| Počet aktívnych ohraničení $m_a$ | Počet makroiterácií | Celkový počet Newtonových iterácií | Čas výpočtu v sekundách |
|---|---|---|---|
| 10 | 6,04 | 15,10 | 0,41 |
| 20 | 6,22 | 15,36 | 0,42 |
| 30 | 6,64 | 15,54 | 0,43 |
| 40 | 7,76 | 16,90 | 0,47 |
| 50 | 6,44 | 16,30 | 0,44 |
| 60 | 7,62 | 17,20 | 0,48 |
| 70 | 6,74 | 16,16 | 0,44 |
| 80 | 6,92 | 15,84 | 0,44 |
| 90 | 7,58 | 16,84 | 0,46 |
| 100 | 9,00 | 17,42 | 0,50 |
| 110 | 11,74 | 19,32 | 0,58 |
| 120 | 15,24 | 21,58 | 0,69 |
| 130 | 20,68 | 24,30 | 0,84 |
| 140 | 48,26 | 42,20 | 1,72 |

Rovnako ako v predošlých prípadoch, aj pri používaní trnasformačnej funkcie 4 (Tab. 3.7, Tab. 3.12 a Tab. 3.16) môžeme pozorovať nárast času výpočtu pri rastúcich rozmeroch úlohy.

Pri porovnaní výsledkov jednotlivých transformačných funkcií pre úlohy s rozmerom $n = 150$, $m = 200$ (Tab. 3.13 - Tab. 3.16) je najvýhodnejšia opäť funkcia 3 a najpomalšia (pri zohľadnení času a počtu makroiterácií) funkcia 1. Počet Newtonových iterácií bol najvyšší pri použití funkcie 2, najnižší pri použití funkcie 3. Počet Newtonových iterácií bol najnižší pri funkcii 3, najvyšší pri funkcii 2.

Pri riešení úloh (3.1) sa teda ako najvýhodnejšie javí použitie transformačnej funkcie 3, pretože aj pri väčších počtoch aktívnych ohraničení a väčších hodnotách $n$ a $m$ dosahuje najrýchlejšie výsledky pri najkratšom počte vykonaných makroiterácií spomedzi použitých transformačných funkcií.

### 3.4.2 Riešenie úloh (3.6)

Pre úlohy (3.6) sme vykonali rovnaké pozorovania ako v predošlom prípade, s rovnakými vstupnými údajmi. Výsledky uvádzame v tabuľkách. Keďže nárast času a počtu iterácií sa nám potvrdil vo všetkých prípadoch, zameriame sa hlavne na porovnanie výhodnosti použitia tej-ktorej transformačnej funkcie a na zhodnotenie výsledkov v porovnaní s výsledkami z časti 3.4.1.

#### Úlohy s rozmermi $n = 40$, $m = 50$

Tabuľka 3.17: Tabuľka výstupov pre transformačnú funkciu 1 pri rozmeroch úlohy $n = 40$, $m = 50$ - riešenie úloh (3.6)

| Počet aktívnych ohraničení $m_a$ | Počet makroiterácií | Celkový počet Newtonových iterácií | Čas výpočtu v sekundách |
|---|---|---|---|
| 10 | 9,42 | 38,22 | 0,88 |
| 20 | 9,16 | 36,16 | 0,76 |
| 30 | 14,46 | 57,66 | 1,11 |

Tabuľka 3.18: Tabuľka výstupov pre transformačnú funkciu 2 pri rozmeroch úlohy $n = 40$, $m = 50$ - riešenie úloh (3.6)

| Počet aktívnych ohraničení $m_a$ | Počet makroiterácií | Celkový počet Newtonových iterácií | Čas výpočtu v sekundách |
|---|---|---|---|
| 10 | 4,58 | 20,32 | 0,41 |
| 20 | 4,14 | 17,96 | 0,34 |
| 30 | 7,92 | 33,20 | 0,57 |

Tabuľka 3.19: Tabuľka výstupov pre transformačnú funkciu 3 pri rozmeroch úlohy $n = 40$, $m = 50$ - riešenie úloh (3.6)

| Počet aktívnych ohraničení $m_a$ | Počet makroiterácií | Celkový počet Newtonových iterácií | Čas výpočtu v sekundách |
|---|---|---|---|
| 10 | 4,40 | 26,72 | 0,49 |
| 20 | 4,88 | 30,26 | 0,56 |
| 30 | 9,40 | 52,24 | 0,96 |

Tabuľka 3.20: Tabuľka výstupov pre transformačnú funkciu 4 pri rozmeroch úlohy $n = 40$, $m = 50$ - riešenie úloh (3.6)

| Počet aktívnych ohraničení $m_a$ | Počet makroiterácií | Celkový počet Newtonových iterácií | Čas výpočtu v sekundách |
|---|---|---|---|
| 10 | 6,74 | 36,06 | 1,07 |
| 20 | 7,36 | 38,58 | 1,06 |
| 30 | 14,24 | 72,66 | 2,09 |

Pre riešenie úloh (3.6) s malými rozmermi vyplýva z tabuliek Tab 3.17 - Tab 3.20, že funkcia 2 je najrýchlejšia vzhľadom na čas aj počet makroiterácií. Naopak, najpomalšou funkciou z hľadiska trvania celkového výpočtu je funkcia 4 a z hľadiska potrebného počtu makroiterácii pre výpočet je funkcia 1. Z pohľadu Newtonových iterácií bola najrýchlejšia funkcia 2, najpomalšia funkcia 1.

### Úlohy s rozmermi $n = 80$, $m = 100$

Overíme si predchádzjúce výsledky pri riešení úloh s vačšími rozmermi.

Tabuľka 3.21: Tabuľka výstupov pre transformačnú funkciu 1 pri rozmeroch úlohy $n = 80$, $m = 100$ - riešenie úloh (3.6)

| Počet aktívnych ohraničení $m_a$ | Počet makroiterácií | Celkový počet Newtonových iterácií | Čas výpočtu v sekundách |
|---|---|---|---|
| 10 | 9,04 | 35,90 | 1,17 |
| 20 | 9,20 | 35,56 | 1,31 |
| 30 | 9,40 | 34,28 | 1,08 |
| 40 | 9,28 | 35,80 | 1,09 |
| 50 | 11,06 | 40,60 | 1,19 |
| 60 | 12,96 | 50,64 | 1,51 |
| 70 | 26,54 | 101,58 | 2,96 |

Tabuľka 3.22: Tabuľka výstupov pre transformačnú funkciu 2 pri rozmeroch úlohy $n = 80$, $m = 100$ - riešenie úloh (3.6)

| Počet aktívnych ohraničení $m_a$ | Počet makroiterácií | Celkový počet Newtonových iterácií | Čas výpočtu v sekundách |
|---|---|---|---|
| 10 | 4,06 | 18,86 | 0,46 |
| 20 | 4,58 | 20,06 | 0,60 |
| 30 | 4,24 | 17,50 | 0,45 |
| 40 | 4,50 | 18,94 | 0,51 |
| 50 | 5,90 | 23,70 | 0,60 |
| 60 | 7,56 | 29,42 | 0,75 |
| 70 | 13,82 | 54,66 | 1,37 |

Tabuľka 3.23: Tabuľka výstupov pre transformačnú funkciu 3 pri rozmeroch úlohy $n = 80$, $m = 100$ - riešenie úloh (3.6)

| Počet aktívnych ohraničení $m_a$ | Počet makroiterácií | Celkový počet Newtonových iterácií | Čas výpočtu v sekundách |
|---|---|---|---|
| 10 | 4,14 | 27,08 | 0,63 |
| 20 | 4,46 | 28,16 | 0,70 |
| 30 | 4,32 | 28,44 | 0,65 |
| 40 | 5,10 | 32,70 | 0,83 |
| 50 | 5,68 | 36,74 | 0,83 |
| 60 | 7,82 | 50,34 | 1,07 |
| 70 | 13,90 | 84,44 | 1,90 |

Tabuľka 3.24: Tabuľka výstupov pre transformačnú funkciu 4 pri rozmeroch úlohy $n = 80$, $m = 100$ - riešenie úloh (3.6)

| Počet aktívnych ohraničení $m_a$ | Počet makroiterácií | Celkový počet Newtonových iterácií | Čas výpočtu v sekundách |
|---|---|---|---|
| 10 | 6,30 | 34,08 | 1,45 |
| 20 | 6,36 | 34,34 | 1,48 |
| 30 | 6,26 | 34,46 | 1,41 |
| 40 | 6,58 | 35,88 | 1,52 |
| 50 | 8,60 | 44,72 | 1,95 |
| 60 | 12,38 | 65,84 | 2,69 |
| 70 | 25,34 | 128,24 | 5,32 |

V tomto numerickom experimente (Tab. 3.21 - Tab. 3.24) sa nám potvrdilo, že pre samotný výpočet je časovo najrýchlejším použitie transformačnej funkcie 2, no vzhľadom na počet makroiterácií sa funkcie 2 a 3 javia rovako výhodné (túto skutočnosť si overíme

v experimente s ešte väčšími rozmermi úloh). Najpomalšou metódou vzhľadom na počet makroiterácií je funkcia 1, vzhľadom na čas zasa funkcia 4. Podľa počtu Newtonových iterácií je najvýhodnejšou funkcia 2, na opačnej strane stojí funkcia 1.

### Úlohy s rozmermi $n = 150$, $m = 200$

Overíme si predošlé výsledky a zároveň porovnáme výsledky s výsledkami pre úlohy (3.1).

Tabuľka 3.25: Tabuľka výstupov pre transformačnú funkciu 1 pri rozmeroch úlohy $n = 150$, $m = 200$ - riešenie úloh (3.6)

| Počet aktívnych ohraničení $m_a$ | Počet makroiterácií | Celkový počet Newtonových iterácií | Čas výpočtu v sekundách |
|---|---|---|---|
| 10 | 9,24 | 36,50 | 2,61 |
| 20 | 9,16 | 35,04 | 2,61 |
| 30 | 9,26 | 34,94 | 2,55 |
| 40 | 9,34 | 34,44 | 2,59 |
| 50 | 9,38 | 34,42 | 2,52 |
| 60 | 9,88 | 35,00 | 2,59 |
| 70 | 11,16 | 38,16 | 2,76 |
| 80 | 10,90 | 40,18 | 2,99 |
| 90 | 10,44 | 37,30 | 2,73 |
| 100 | 10,54 | 37,12 | 2,73 |
| 110 | 11,78 | 44,34 | 3,25 |
| 120 | 16,22 | 58,34 | 4,28 |
| 130 | 22,40 | 78,90 | 5,87 |
| 140 | 33,68 | 127,38 | 9,56 |

Riešili sme komplikovanejšiu úlohu, než v predchádzajúcom prípade (úloha kvadratického programovani (3.1) (Tab. 3.13), čoho dôkazom sú aj nasledovné výsledky. Vyriešenie úlohy trvá celkovo dlhšie a zároveň aj potrebný počet Newtonových iterácií je väčší. Počet makroiterácií zostal približne rovnaký, z čoho vyplýva, že komplikovanejší je len výpočet optimálneho riešenia $\hat{\mathbf{x}}$.

Tabuľka 3.26: Tabuľka výstupov pre transformačnú funkciu 2 pri rozmeroch úlohy $n = 150$, $m = 200$ - riešenie úloh (3.6)

| Počet aktívnych ohraničení $m_a$ | Počet makroiterácií | Celkový počet Newtonových iterácií | Čas výpočtu v sekundách |
|---|---|---|---|
| 10 | 4,06 | 18,94 | 0,91 |
| 20 | 4,06 | 18,44 | 0,87 |
| 30 | 4,44 | 18,92 | 0,93 |
| 40 | 4,22 | 17,42 | 0,74 |
| 50 | 4,44 | 18,08 | 0,88 |
| 60 | 4,12 | 16,82 | 0,77 |
| 70 | 4,40 | 17,48 | 0,83 |
| 80 | 5,28 | 20,06 | 0,92 |
| 90 | 5,74 | 21,26 | 1,03 |
| 100 | 8,92 | 31,34 | 1,49 |
| 110 | 9,00 | 33,46 | 1,64 |
| 120 | 10,48 | 39,30 | 1,71 |
| 130 | 15,04 | 55,06 | 2,38 |
| 140 | 18,76 | 75,68 | 3,43 |

Vzhľadom na riešenie zložitejšej ulohy aj pri použití transformačnej funkcie 2 sme dostali väčší výpočtový čas ako v prípade riešenia úlohy (3.1) (Tab.3.14). Avšak počet Newtonových iterácii bol pri menšom počte aktívnych ohraničení $m_a$ dokonca nižší ako v predchádzajúcom prípade, čo naznačuje výhodnosť použitia tejto transformačnej funkcie pre danú úlohu.

Tabuľka 3.27: Tabuľka výstupov pre transformačnú funkciu 3 pri rozmeroch úlohy $n = 150$, $m = 200$ - riešenie úloh (3.6)

| Počet aktívnych ohraničení $m_a$ | Počet makroiterácií | Celkový počet Newtonových iterácií | Čas výpočtu v sekundách |
|---|---|---|---|
| 10 | 4,16 | 26,64 | 1,15 |
| 20 | 4,04 | 26,68 | 1,17 |
| 30 | 4,26 | 27,50 | 1,29 |
| 40 | 4,60 | 30,50 | 1,26 |
| 50 | 4,16 | 28,90 | 1,31 |
| 60 | 4,50 | 30,62 | 1,35 |
| 70 | 4,32 | 29,44 | 1,38 |
| 80 | 4,22 | 29,28 | 1,16 |
| 90 | 4,90 | 33,12 | 1,38 |
| 100 | 5,26 | 34,76 | 1,79 |
| 110 | 6,84 | 44,30 | 1,96 |
| 120 | 8,66 | 54,88 | 2,29 |
| 130 | 11,50 | 71,86 | 2,72 |
| 140 | 16,64 | 96,36 | 4,08 |

Pre výpočet tejto úlohy s použitím transformačnej funkcie 3 je potrebný dlhší čas ako aj väčší počet Newtonových iterácii, než v príprade výpočtov pre funkcie (3.1) (Tab. 3.15).

Tabuľka 3.28: Tabuľka výstupov pre transformačnú funkciu 4 pri rozmeroch úlohy $n = 150$, $m = 200$ - riešenie úloh (3.6)

| Počet aktívnych ohraničení $m_a$ | Počet makroiterácií | Celkový počet Newtonových iterácií | Čas výpočtu v sekundách |
|---|---|---|---|
| 10 | 6,16 | 32,82 | 2,63 |
| 20 | 6,10 | 32,68 | 2,54 |
| 30 | 6,16 | 33,68 | 2,66 |
| 40 | 6,46 | 34,86 | 2,46 |
| 50 | 6,20 | 34,16 | 2,62 |
| 60 | 6,38 | 34,40 | 2,60 |
| 70 | 6,30 | 33,98 | 2,63 |
| 80 | 6,60 | 35,44 | 2,69 |
| 90 | 7,50 | 40,22 | 3,11 |
| 100 | 8,44 | 43,74 | 3,34 |
| 110 | 11,64 | 60,58 | 4,82 |
| 120 | 15,24 | 77,32 | 5,83 |
| 130 | 21,62 | 108,12 | 8,00 |
| 140 | 31,10 | 150,14 | 11,44 |

Potreba vyššieho počtu Newtonových iterácií a celkovo dlhšieho času pre výpočet úlohy kvadratického programovania (Tab. 3.16) sa potvrdila aj pri použití transformačnej funkcie 4.

Ak porovnáme navzájom transformačné funkcie v tomto experimente (Tab. 3.25 - Tab. 3.28), najvýhodnejšou sa javí funkcia 2 (v počte makroiterácií aj výpočtovom čase). Využitie transformačnej funkcie 1 pre výpočet úlohy sa zdá byť najhorším riešením spomedzi pozorovaných funkcií. Počet Newtonoých iterácií ukazuje ako najvhodnejšie použitie funkcie 2, najhoršie výsledky z tohto pohľadu dosiahla opäť funkcia 1.

Z toho vyvodzujeme, že pri riešení úloh typu (3.6) je najvhodnejšie použiť transformačnú funkciu 2, pretože aj pri väčších počtoch aktívnych ohraničení a väčších hodnotách $n$ a $m$ dosahuje najrýchlejšie výsledky pri najmenšom počte vykonaných makroiterácií spomedzi použitých transformačných funkcií. Prípadne by bolo možné použiť funkciu 3, nakoľko jej výstupné hodnoty sa výrazne neodlišujú od funkcie 2.

### 3.4.3 Riešenie úloh (3.7)

Vykonali sme experimenty s úlohami (3.7), s rovnakými vstupnými hodnotami ako v predošlých prípadoch. Opäť sme porovnávali výhodnosť použitia jednotlivých transformačných funkcií. Porovnanie výsledkov sme realizovali aj pre rôzne účelové funkcie ((3.1) a (3.6)).

#### Úlohy s rozmermi $n = 40$, $m = 50$

Tabuľka 3.29: Tabuľka výstupov pre transformačnú funkciu 1 pri rozmeroch úlohy $n = 40$, $m = 50$ - riešenie úloh (3.7)

| Počet aktívnych ohraničení $m_a$ | Počet makroiterácií | Celkový počet Newtonových iterácií | Čas výpočtu v sekundách |
|---|---|---|---|
| 10 | 6,00 | 41,40 | 0,53 |
| 20 | 7,58 | 51,68 | 0,65 |
| 30 | 14,56 | 88,80 | 1,13 |

Tabuľka 3.30: Tabuľka výstupov pre transformačnú funkciu 2 pri rozmeroch úlohy $n = 40$, $m = 50$ - riešenie úloh (3.7)

| Počet aktívnych ohraničení $m_a$ | Počet makroiterácií | Celkový počet Newtonových iterácií | Čas výpočtu v sekundách |
|---|---|---|---|
| 10 | 5,24 | 39,74 | 0,33 |
| 20 | 5,70 | 39,80 | 0,33 |
| 30 | 8,46 | 49,32 | 0,42 |

Tabuľka 3.31: Tabuľka výstupov pre transformačnú funkciu 3 pri rozmeroch úlohy $n = 40$, $m = 50$ - riešenie úloh (3.7)

| Počet aktívnych ohraničení $m_a$ | Počet makroiterácií | Celkový počet Newtonových iterácií | Čas výpočtu v sekundách |
|---|---|---|---|
| 10 | 4,28 | 34,28 | 0,35 |
| 20 | 4,58 | 38,18 | 0,30 |
| 30 | 8,38 | 59,06 | 0,48 |

Tabuľka 3.32: Tabuľka výstupov pre transformačnú funkciu 4 pri rozmeroch úlohy $n = 40$, $m = 50$ - riešenie úloh (3.7)

| Počet aktívnych ohraničení $m_a$ | Počet makroiterácií | Celkový počet Newtonových iterácií | Čas výpočtu v sekundách |
|---|---|---|---|
| 10 | 6,14 | 45,12 | 0,58 |
| 20 | 7,74 | 52,96 | 0,68 |
| 30 | 14,56 | 88,82 | 1,16 |

Z výstupných tabuliek (Tab. 3.29 - Tab. 3.32) možno pozorovať, že časovo najvýhodnejšou je funkcia 2, zatiaľ čo vzhľadom na počet makroiterácií sa najlepšou javí funkcia 3. Horšie výsledky dosiahli funkcie 1 a 4. Počet Newtonových iterácií ukazuje ako najrýchlejšiu funkciu 3, najhoršie výsledky dosiahla funkcia 4. Vzhľadom na malý rozmer pozorovaných úloh sme tieto výsledky overovali pri riešení úloh s väčšími rozmermi.

### Úlohy s rozmermi $n = 80$, $m = 100$

Tabuľka 3.33: Tabuľka výstupov pre transformačnú funkciu 1 pri rozmeroch úlohy $n = 80$, $m = 100$ - riešenie úloh (3.7)

| Počet aktívnych ohraničení $m_a$ | Počet makroiterácií | Celkový počet Newtonových iterácií | Čas výpočtu v sekundách |
|---|---|---|---|
| 10 | 6,20 | 41,52 | 0,97 |
| 20 | 6,00 | 40,52 | 0,94 |
| 30 | 6,32 | 47,66 | 1,14 |
| 40 | 6,28 | 42,14 | 0,98 |
| 50 | 8,20 | 57,14 | 1,33 |
| 60 | 11,64 | 82,36 | 1,89 |
| 70 | 29,34 | 175,12 | 4,13 |

Tabuľka 3.34: Tabuľka výstupov pre transformačnú funkciu 2 pri rozmeroch úlohy $n = 80$, $m = 100$ - riešenie úloh (3.7)

| Počet aktívnych ohraničení $m_a$ | Počet makroiterácií | Celkový počet Newtonových iterácií | Čas výpočtu v sekundách |
|---|---|---|---|
| 10 | 4,14 | 35,76 | 0,50 |
| 20 | 4,50 | 28,50 | 0,41 |
| 30 | 4,92 | 35,28 | 0,53 |
| 40 | 5,08 | 32,08 | 0,48 |
| 50 | 6,00 | 42,86 | 0,63 |
| 60 | 7,64 | 50,90 | 0,75 |
| 70 | 16,62 | 107,00 | 1,58 |

Tabuľka 3.35: Tabuľka výstupov pre transformačnú funkciu 3 pri rozmeroch úlohy $n = 80$, $m = 100$ - riešenie úloh (3.7)

| Počet aktívnych ohraničení $m_a$ | Počet makroiterácií | Celkový počet Newtonových iterácií | Čas výpočtu v sekundách |
|---|---|---|---|
| 10 | 4,24 | 33,88 | 0,48 |
| 20 | 4,18 | 32,32 | 0,45 |
| 30 | 4,30 | 38,84 | 0,54 |
| 40 | 4,22 | 34,22 | 0,47 |
| 50 | 5,68 | 51,50 | 0,71 |
| 60 | 7,72 | 73,32 | 1,02 |
| 70 | 15,58 | 109,78 | 1,55 |

Tabuľka 3.36: Tabuľka výstupov pre transformačnú funkciu 4 pri rozmeroch úlohy $n = 80$, $m = 100$ - riešenie úloh (3.7)

| Počet aktívnych ohraničení $m_a$ | Počet makroiterácií | Celkový počet Newtonových iterácií | Čas výpočtu v sekundách |
|---|---|---|---|
| 10 | 6,24 | 41,74 | 1,00 |
| 20 | 6,18 | 39,10 | 0,93 |
| 30 | 6,32 | 47,66 | 1,17 |
| 40 | 6,28 | 42,14 | 1,00 |
| 50 | 8,20 | 57,14 | 1,38 |
| 60 | 11,64 | 82,36 | 1,93 |
| 70 | 29,44 | 175,44 | 4,23 |

Pozorovaním výsledkov riešenia úloh (3.7) s rozmermi $n = 80$, $m = 150$ (Tab. 3.33 - Tab. 3.36) sa ukazuje, že použitie transformačných funkcií 2 a 3 je rovnako výhodné (z hladiska potrebného času výpočtu a počtu makroiterácií). Menej výhodným sa zdá

byť použitie funkcií 1 a 4, ktoré rovnako dosahujú podobné výsledky (sledovanými ukazovateľmi sú opäť potrebný čas a počet makroiterácií). Najnižší počet Newtonových iterácií sme zaznamenali pri použití transformačnej funkcie 3, najvyššie počty dosiahli funkcie 1 a 4.

### Úlohy s rozmermi $n = 150$, $m = 200$

Tabuľka 3.37: Tabuľka výstupov pre transformačnú funkciu 1 pri rozmeroch úlohy $n = 150$, $m = 200$ - riešenie úloh (3.7)

| Počet aktívnych ohraničení $m_a$ | Počet makroiterácií | Celkový počet Newtonových iterácií | Čas výpočtu v sekundách |
|---|---|---|---|
| 10 | 6,46 | 42,38 | 2,46 |
| 20 | 6,32 | 45,56 | 2,55 |
| 30 | 6,26 | 42,90 | 2,36 |
| 40 | 6,20 | 44,32 | 2,41 |
| 50 | 6,64 | 45,00 | 2,47 |
| 60 | 6,34 | 45,52 | 2,50 |
| 70 | 6,20 | 42,74 | 2,37 |
| 80 | 6,62 | 46,13 | 2,56 |
| 90 | 7,88 | 52,56 | 2,95 |
| 100 | 8,92 | 60,58 | 3,40 |
| 110 | 10,74 | 68,42 | 3,91 |
| 120 | 14,40 | 93,60 | 5,37 |
| 130 | 21,54 | 135,00 | 7,71 |
| 140 | 32,82 | 203,90 | 11,72 |

Nakoľko sa jedná o zložitejšiu účelovú funkciu než je funkcia (3.1), opať sa potvrdila skutočnosť, že čas potrebný na výpočet ako aj počet Newtonových iterácií je pre túto funkciu vyšší (v porovnaní s Tab. 3.13) .

Pri porovnaní výsledkov výpočtov s použitím účelovej funkcie (3.6), sú spomínané hodnoty naopak nižšie (v porovnaní s Tab. 3.25), z čoho usudzujeme, že účelová funkcie (3.7) je výpočtovo jednoduchšia.

Tabuľka 3.38: Tabuľka výstupov pre transformačnú funkciu 2 pri rozmeroch úlohy $n = 150$, $m = 200$ - riešenie úloh (3.7)

| Počet aktívnych ohraničení $m_a$ | Počet makroiterácií | Celkový počet Newtonových iterácií | Čas výpočtu v sekundách |
|---|---|---|---|
| 10 | 4,58 | 38,24 | 1,45 |
| 20 | 5,38 | 41,16 | 1,58 |
| 30 | 5,50 | 35,32 | 1,36 |
| 40 | 4,86 | 30,32 | 1,14 |
| 50 | 5,12 | 33,18 | 1,25 |
| 60 | 5,24 | 31,50 | 1,23 |
| 70 | 5,36 | 35,14 | 1,36 |
| 80 | 5,44 | 36,82 | 1,43 |
| 90 | 5,60 | 39,90 | 1,59 |
| 100 | 6,78 | 43,16 | 1,72 |
| 110 | 6,72 | 42,52 | 1,68 |
| 120 | 8,04 | 50,56 | 2,16 |
| 130 | 11,20 | 75,60 | 3,02 |
| 140 | 16,32 | 103,32 | 4,20 |

Počet makroiterácií, počet Newtonových iterácií potrebných pre výpočet optimálneho $\hat{\mathbf{x}}$ a celkový čas výpočtu úlohy s účelovou funkciou (3.7) je v porovnaní s výpočtami pre účelové funkcie (3.1) (Tab. 3.14) a (3.6) (Tab. 3.26) najväčší (čo vyvrátilo náš predpoklad o výpočtovo jednoduchšej úlohe).

Tabuľka 3.39: Tabuľka výstupov pre transformačnú funkciu 3 pri rozmeroch úlohy $n = 150$, $m = 200$ - riešenie úloh (3.7)

| Počet aktívnych ohraničení $m_a$ | Počet makroiterácií | Celkový počet Newtonových iterácií | Čas výpočtu v sekundách |
|---|---|---|---|
| 10 | 4,22 | 33,38 | 1,28 |
| 20 | 4,42 | 37,78 | 1,41 |
| 30 | 4,44 | 37,06 | 1,34 |
| 40 | 4,38 | 38,12 | 1,35 |
| 50 | 4,24 | 35,60 | 1,25 |
| 60 | 4,36 | 37,06 | 1,30 |
| 70 | 4,20 | 35,20 | 1,25 |
| 80 | 4,86 | 41,86 | 1,51 |
| 90 | 4,84 | 41,92 | 1,52 |
| 100 | 5,76 | 48,84 | 1,79 |
| 110 | 6,92 | 53,84 | 2,01 |
| 120 | 7,58 | 62,86 | 2,38 |
| 130 | 11,56 | 88,52 | 3,34 |
| 140 | 17,64 | 128,56 | 4,90 |

Spomedzi všetkých troch sledovaných účelových funkcií, sú výpočty s použitým funkcie (3.7) najnáročnejšie, a to nie len z časového hľadiska samotného výpočtu, ale aj z pohľadu potrebného počtu Newtonových iterácií pre zistenie optimálnej hodnoty $\hat{\mathbf{x}}$.

Tabuľka 3.40: Tabuľka výstupov pre transformačnú funkciu 4 pri rozmeroch úlohy $n = 150$, $m = 200$ - riešenie úloh (3.7)

| Počet aktívnych ohraničení $m_a$ | Počet makroiterácií | Celkový počet Newtonových iterácií | Čas výpočtu v sekundách |
|---|---|---|---|
| 10 | 6,14 | 40,62 | 1,14 |
| 20 | 6,42 | 44,12 | 1,26 |
| 30 | 6,44 | 41,06 | 1,10 |
| 40 | 6,18 | 40,22 | 1,04 |
| 50 | 6,36 | 40,56 | 1,06 |
| 60 | 6,34 | 40,66 | 1,08 |
| 70 | 6,28 | 40,32 | 1,06 |
| 80 | 6,66 | 44,22 | 1,24 |
| 90 | 7,12 | 47,40 | 1,43 |
| 100 | 8,16 | 53,46 | 1,71 |
| 110 | 9,14 | 57,52 | 1,94 |
| 120 | 11,02 | 71,60 | 2,71 |
| 130 | 15,78 | 101,66 | 4,10 |
| 140 | 23,28 | 147,20 | 6,35 |

V porovnaní s oboma predošlými experimentami je pri výpočte s použitím učelovej funkcie (3.7) potrebný približne rovnaký počet makroiterácii. V príprade zamerania sa na hodnoty celkového počtu Newtonových iterácií, táto skuočnosť neplatí. Práve naopak. Na vyriešenie úlohy je potrebný najväčší počet týchto iterácií. Z časového hľadiska je trvanie výpočtov pre funkciu (3.7) dlhšie než pre funkciu (3.6) (Tab. 3.28) a naopak kratšie ako pri funkcii (3.1) (Tab. 3.16).

V tomto experimente (Tab. 3.37 - Tab. 3.40) sa javí ako najvýhodnejšie použiť funkciu 2 alebo 3 - dosahovali podobné výsledky, výhodné vo všetkých ohľadoch. Nevýhodnym sa zdá byť hlavne použitie transformačnej funkcie 1. Počet Newtonových iterácií poukazuje na nevýhodnosť použitia funkcie 1.

Môžme teda konštatovať, že na riešenie úloh typu (3.7) je najvhodnejšie použiť transformačnú funkciu 2 alebo 3, keďže dosahovali podobné a vo všetkých ohľadoch najlepšie výsledky.

# Záver

V tejto práci sme sa venovali metóde rozšírených Lagrangeových funkcií z teoretického ako aj z praktického hľadiska. Príbližili sme si ich zostrojovanie a skonštruovali sme algoritmus na hľadanie sedlového bodu práve pomocou týchto funkcií. Ako sme ukázali pri ilustratívnych riešeniach, vytvorený algoritmus poskytoval riešenia vrámci požadovanej presnosti.

Následne sme realizovali sériu experimentov (experimenty sme vykonali na úlohách rôznych rozmerov - počet premenných $n$ a počet ohraničení $m$), kde sme sa zamerali hlavne na porovnávanie výsledkov dosiahnutých pri použití troch typov účelových a štyroch typov transformačných funkcií. Spolu sme vygenerovali približne 3 600 úloh, z ktorých každú sme riešili štyrikrát (spolu teda približne 15 000 riešení). Výsledkom našich pozorovaní bolo zistenie, že počet makroiterácií, počet Newtonových iterácií a čas výpočtu závisia od počtu aktívnych ohraničení.

Zároveň sme porovnávali štyri rôzne transformačné funkcie.
Z našich experimentov vyplynulo, že pri riešení úloh (3.1) je najvhodnejšie použiť transformačnú funkciu 3 - (2.47), keďže aj pri väčších počtoch aktívnych ohraničení a väčších hodnotách $n$ a $m$ dosahuje najrýchlejšie výsledky pri najmenšom počte vykonaných makroiterácií.
Z rovnakého dôvodu je najvhodnejšie pri riešení úloh (3.6) použiť transformačnú funkciu 2 - (2.46), alebo funkciu 3 - (2.47). Metódy s použitím oboch týchto tansformačných funkcií dosahovali približne rovnaké hodnoty.
Najvhodnejšou metódou pre riešenie funkcií (3.7) je v rozšírenej Lagrangeovej funkcii použiť transformačnú funkciu 2 - (2.46), alebo funkciu 3 - (2.47). Obe funkcie dosahovali približne rovnaké výsledky.

Efektívne a rýchle riešenie rôznych typov úloh si teda vyžaduje použitie rôznych transformačných funkcií.

# Literatúra


[1] HAMALA M. 2007. *Nelineárne programovanie*, Prednášky 3. ročník, Fakulta matematiky, fyziky a informatiky UK, Bratislava.

[2] HAMALA M. 1972. *Nelineárne programovanie*, ALFA, Bratislava.

[3] EVERETT, H. 1963. Generalized Lagrange Multiplier Method for Solving Problems of Optimum Allocation of Resources. In *Operations Research*. Vol. 11, No.3, p. 300-417.

[4] HESTENES M.R. 1969. Multiplier and Gradient Methods. In *Journal of Optimization Theory and Applications*. Vol. 4, No. 5, p. 303-320.

[5] ROCKAFELLAR R.T. 1970. *Convex Analysis*, Princeton Univ. Press, Princ., N.J.

[6] AUSLENDER A. - TEBOULLE M. - BEN-TIBA S. 1999. Interior proximal point and multiplier methods based on second order homogenous kernels. In *Mathematics of Operations Research*. Vol.24, No.3, p. 645-668.

[7] BEN TAL A. - YUZEFOVICH I. - ZIBULEVSKY M. 1992. Penalty/barier multiplier methods for minimax and constrained smooth programs. In *Research report 9/92*, Optimization Laboratory, Faculty of Industrial Engeneering Management, Technion, Haifa, Israel.

[8] BEN TAL. A - ZIBULEVSKY M. 1997. Penalty/barier multilier methods for convex programming problems. In *SIAM Journal on Optimization*, Vol. 7, No. 2, p. 347-366.

[9] BERTSEKAS D.P. 1982. *Constrained optimization and Lagrange multiplier methods*, Academic Press, New York

[10] KORT B.W. - BERTSEAKS D.P. 1973. Multiplier methods for convex programming. In *Proceedings of the IEEE Decision and Control Conference*, San Diego, CA, p. 260-264.

[11] POLYAK R.A. - TEBOULLE M. 1997. Nonlinear rescaling and proximal-like methods in convex optimization. In *Mathematical programming*, Vol. 76, p. 265-284.

[12] BIRGIN E. G. - CASTILLO R.A. - MARTINEZ J.M. 2005. Numerical comparison of Augmented Lagrangian algorithms for noncovex problems. In *Computational Optimization and Aplications*. Vol. 31, No. 1, p. 31-55.

[13] SUN X.L. - LI D. - MCKINNON K.I.M. 2005. On saddle points of Augmented Lagrangian for constrained optimization. In *SIAM J. Optim.*. Vol. 15, No. 4, p. 1228-1146.

# Príloha

V tejto časti pripájame zdrojový kód (Matlab 7.9.0) algoritmu a ďalších pomocných funkcií, ktoré sme používali pri našich výpočtoch.

```
function [it,pocitN,time] = algoritmus(m,ma,n,r,ro,sigma, eps, delta)
[A, G, xopt,yopt, b,h,f, g]=generator(m,ma, n,r);
[x0, y0]=pomocne(xopt, yopt, ro, sigma,m,n);
t1=clock;
format short e
it=1;
x=x0;
yk=y0;
pocitN=0;
poc=0;
while poc~=2*m
   poc=0;
   [xN, itN] = Newton(x, yk, A, G, h,r, eps,g,b,f,m);
   pocitN=pocitN+itN;
   x=xN;
   ghod=eval(g);
   poc1=0;
   poc2=0;
   poc3=0;
   for i1=1:m
      if ghod(i1) > delta
         poc1=poc1+1;
      end
      if abs(ghod(i1)) <= delta
         poc2=poc2+1;
      end
      if ghod(i1)< -delta
         poc3=poc3+1;
      end
   end
   for i=1:m
      if ghod(i)>= -delta
         poc=poc+1;
      end
      if ghod(i) > delta
         if yk(i)> 10^(-5)
            poc=poc-1;
         end
      end
      if yk(i)>0
         poc=poc+1;
      end
   end
   if poc==2*m
```

```
        break
    end
    for i=1:m
        gg=ghod(i);
        yk(i)=-(yk(i)*prvaderpsig(gg,1));
    end
   it=it+1;
end
t2=clock;
time=etime(t2,t1);
end



function cislo = druhaderpsig(gg,r)
if gg>= -r/2
%if gg>=0
    %cislo=(2*gg + r^2)^(-1.5);
    %cislo=(2*r^2)/(r+gg)^3;
    cislo=r/(r+gg)^2;
else
    %cislo = 1/r^3;
    %cislo=2/r;
    %cislo=16/r;
    cislo=4/r;
end
end



function [A, G, xopt,yopt, b,h,f, g]=generator1(m,ma, n,r)
A=-5+randi(10,m,n);
B=-3+randi(6,n,n);
C=diag(ones(n,1),0);
G=B*B' + C;
xopt=-9+randi(18,n,1);
b=zeros(m,1);
u=zeros(m,1);
temp=A*xopt;
for i=1:ma
    u(i)=1+randi(8,1,1);
    b(i)=temp(i);
end
beta=1 + randi(2,m,1);
for i=(ma+1):m
    u(i)=0;
    b(i)=temp(i)-beta(i);
end
uopt=u;
h= A'*uopt-G*xopt;
```

```
f=sprintf('0.5*x''*G*x + h''*x');
g=sprintf('A*x - b');
x=xopt;
ghod=eval(g);
yop=zeros(m,1);
for i=1:m
    gg=ghod(i);
    yop(i)=uopt(i)/-prvaderpsig(gg,r);
end
yopt=yop;
end



function gradient=gradL(xl,yk, A, G, h,r,g,b,m)
x=xl;
gg=eval(g);
v=zeros(m,1);
psi=zeros(m,1);
for i=1:m
   psi(i)=prvaderpsig(gg(i),r);
    v(i)=yk(i)*psi(i);
end
gradient=((G*xl + h) + A'*v)';
end



function hesian=hesL(xl,yk, A, G, h,r,g,b,m)
x=xl;
ghod=eval(g);
d=zeros(m,1);
for i=1:m
    gg=ghod(i);
    d(i)=yk(i)*druhaderpsig(gg,r);
end
D=diag(d);
hesian = G + A'*D*A;
end



function  fi = L(xl,yk,f, G, h, r, A, b, g,m)
x=xl;
ff = eval(f);
gg = eval(g);
suma = 0;
psi=zeros(m,1);
for i = 1:m
    psi(i)=psignove(gg(i),r);
```

```
    suma = suma + yk(i)*psi(i);
end
fi = ff + suma;
end
```

```
function [xk, itN]= Newton(x0, y1, A, G, h,r, eps,g,b,f,m)
x=x0;
y=y1;
itN=0;
Ll=L(x0,y1,f, G, h, r, A, b, g,m);
for i=1:5000
   H=hesL(x,y,A,G,h,r,g,b,m);
   Gr=gradL(x,y,A,G,h,r,g,b,m);
   if (norm(Gr) <= eps)
     break
   end
   sk=-(H\Gr');
   pk=sk/norm(sk);
   Lambda = novygolden(x, y,A, b,f,g,h,G,r,pk,m);
   x1=x+Lambda*pk;
   Ll1=L(x1,y1,f, G, h, r, A, b, g,m);
   if Ll1>=Ll
      break
   end
   itN=itN+1;
   x=x1;
   Ll=Ll1;
end
xk=x;
end
```

```
function lam = novygolden(xl, y,A, b,f,g,h,G,r,pk,m)
lambda=0;
x=xl+lambda*pk;
fipred=L(x,y,f, G, h, r, A, b, g,m);
t=(sqrt(5)+1)/2;
tau=t;
ax=0;
bx=0;
    for i=1:500
       lambda=tau^(i-1);
       x=xl+lambda*pk;
       fi=L(x,y,f,G,h,r,A,b,g,m);
       if fi>= fipred
          ax=0;
          bx=lambda;
```

```
          break
        else fipred=fi;
        end
    end
z1=2-t;
z2=t-1;
c1=ax+z1*(bx-ax);
x1=xl+c1*pk;
f1 = L(x1,y,f,G,h,r,A,b,g,m);
c2=ax+z2*(bx-ax);
x2=xl+c2*pk;
f2 = L(x2,y,f,G,h,r,A,b,g,m);
eps=0.001;
while ((bx-ax)*norm(pk))>eps
  if f1<f2
     bx=c2;
     c2=c1;
     f2=f1;
     c1=ax+z1*(bx-ax);
     x1=xl+c1*pk;
     f1=L(x1,y,f,G,h,r,A,b,g,m);
  else
     ax=c1;
     c1=c2;
     f1=f2;
     c2=ax+z2*(bx-ax);
     x2=xl+c2*pk;
     f2=L(x2,y,f,G,h,r,A,b,g,m);
  end
end
lam=c2;
end



function [x0, y0]=pomocne(xopt, yopt, ro, sigma,m,n)
xz=-9+randi(18,n,1);
x0=xopt+((xz-xopt)/(norm(xz-xopt)))*ro;
yz=1+randi(8,m,1);
y0=yopt+((yz-yopt)/(norm(yz-yopt)))*sigma;
end



function cislo = prvaderpsig(gg,r)
if gg>= -r/2
%if gg>=0
   %cislo= -((2*gg+r^2)^(-0.5));
   %cislo=-(r/(r+gg)^2);
   %cislo=-(r^2/(gg+r))^2;
```

```
   cislo= -r/(r+gg);
else
   %cislo=(gg/r^3) - 1/r;
   %cislo=((2*gg)/r)-1;
   %cislo=16*gg/r +4;
   cislo=4*gg/r;
end
end



function cislo = psignove(gg,r)
if gg>= -r/2
%if gg>=0
   %cislo=r-sqrt(2*gg + r^2);
   %cislo=-(gg*r)/(r+gg);
   %cislo= -(gg*r)/(gg+r);
   cislo=-r*log(gg/r +1);
else
   %cislo = (gg^2)/(2* (r^3)) - (gg/r);
   %cislo=((gg^2)/r) - gg;
   %cislo=8*(gg^2/r) + 4*gg +r;
   cislo=r*(2*(gg/r)^2 + log(2) -0.5);
end



function [A, G, xopt,yopt, b,h,f, g]=generator2(m,ma, n,r)
A=-5+randi(10,m,n);
B=-3+randi(6,n,n);
C=diag(ones(n,1),0);
G=B*B' + C;
xopt=randi(9,n,1);
b=zeros(m,1);
u=zeros(m,1);
temp=A*xopt;
for i=1:ma
   u(i)=1+randi(8,1,1);
   b(i)=temp(i);
end
beta=1 + randi(2,m,1);
for i=(ma+1):m
   u(i)=0;
   b(i)=temp(i)-beta(i);
end
uopt=u;
h= A'*uopt-(G*xopt+ones(n,1)+log(xopt));
f=sprintf('0.5*x''*G*x + h''*x + logsuma(x)');
g=sprintf('A*x - b');
x=xopt;
```

```
ghod=eval(g);
yop=zeros(m,1);
for i=1:m
   gg=ghod(i);
   yop(i)=uopt(i)/-prvaderpsig(gg,r);
end
yopt=yop;
end



function gradientf = gradf(x,G,h)
gradientf=(G*x +h+ones(length(x),1)+log(x))';
end



function gradient=gradL(xl,yk, A, G, h,r,g,b,m,m1)
x=xl;
gg=eval(g);
v=zeros(m,1);
psi=zeros(m,1);
for i=1:m
  psi(i)=prvaderpsig(gg(i),r);
   v(i)=yk(i)*psi(i);
end
gradient=gradf(x,G,h)+(A'*v)';
end



function [A, G, xopt,yopt, b,h,c,f, g]=generator3(m,ma, n,r)
A=-5+randi(10,m,n);
B=-3+randi(6,n,n);
C=diag(ones(n,1),0);
G=B*B' + C;
xopt=-9+randi(18,n,1);
b=zeros(m,1);
u=zeros(m,1);
temp=A*xopt;
for i=1:ma
   u(i)=1+randi(8,1,1);
   b(i)=temp(i);
end
beta=1 + randi(2,m,1);
for i=(ma+1):m
   u(i)=0;
   b(i)=temp(i)-beta(i);
end
uopt=u;
```

```
c=-3+randi(6,n,1);
h= A'*uopt-(G*xopt + c*exp(c'*xopt));
f=sprintf('0.5*x''*G*x + h''*x+exp(c''*x)');
g=sprintf('A*x - b');
x=xopt;
ghod=eval(g);
yop=zeros(m,1);
for i=1:m
    gg=ghod(i);
    yop(i)=uopt(i)/-prvaderpsig(gg,r);
end
yopt=yop;
end



function gradient=gradL(x,yk, A, G, h,c,r,g,b,m)
gg=eval(g);
v=zeros(m,1);
psi=zeros(m,1);
for i=1:m
   psi(i)=prvaderpsig(gg(i),r);
    v(i)=yk(i)*psi(i);
end
gradient=((G*x + h + c*exp(c'*x)) + A'*v)';
end
```