UNIVERZITA KOMENSKÉHO V BRATISLAVE
FAKULTA MATEMATIKY, FYZIKY A INFORMATIKY

Využitie NCP-funkcií v lineárnom programovaní

BAKALÁRSKA PRÁCA

2012 Tomáš Rizman

UNIVERZITA KOMENSKÉHO V BRATISLAVE
FAKULTA MATEMATIKY, FYZIKY A INFORMATIKY

Využitie NCP-funkcií v lineárnom programovaní

BAKALÁRSKA PRÁCA

Študijný program:	Ekonomická a finančná matematika
Študijný odbor:	1114 Aplikovaná matematika
Školiace pracovisko:	Katedra aplikovanej matematiky a štatistiky
Vedúci práce:	Mgr. Peter Fúsek, PhD.

Bratislava 2012 **Tomáš Rizman**

Univerzita Komenského v Bratislave
Fakulta matematiky, fyziky a informatiky

ZADANIE ZÁVEREČNEJ PRÁCE

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

Názov: Využitie NCP-funkcií v lineárnom programovaní

Cieľ: Naštudovať si teóriu NCP-funkcií vrátane zodpovedajúcich vyhladzovacích metód, špeciálne sa zaoberať ich aplikáciou v lineárnom programovaní. Implementácia vhodnej metódy v MATLABe a jej aplikácia na lineárne programy.

Vedúci: Mgr. Peter Fúsek, PhD.

Dátum zadania: 26.10.2011

Dátum schválenia: 27.10.2011

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

.....
študent

.....
vedúci

Poďakovanie Touto cestou sa chcem poďakovať svojmu vedúcemu bakalárskej práce, Mgr. Petrovi Fúsekovi, PhD. za ochotu, pomoc, odborné rady a podnetné pripomienky, ktoré mi pomohli pri písaní tejto práce. Taktiež mu ďakujem za vedomosti, ktoré som počas písania tejto práce nadobudol. Ďakujem aj svojej mame za pomoc pri korekcií gramatických chýb a priateľom na internáte za pomoc pri pracovaní s LaTexom. Ďakujem Michalovi Hojčkovi za zapožičanie výkonnejšieho počítača, ktorý uľahčil prácu s MATLABom.

Abstrakt v štátnom jazyku

Rizman, Tomáš: Využitie NCP-funkcií v lineárnom programovaní [Bakalárska práca], Univerzita Komenského v Bratislave, Fakulta matematiky, fyziky a informatiky, Katedra aplikovanej matematiky a štatistiky; školiteľ: Mgr. Peter Fúsek, PhD., Bratislava, 2012, 42 s.

V našej práci sa zaoberáme NCP-funkciami, uvádzame niekoľko ich základných vlastností, príslušné vyhladzovacie metódy (aproximácie pomocou vyhladzovacieho parametra), ktoré z nich robia výborný nástroj na riešenie lineárnych programov. Venujeme sa podrobne odvodeniu a vysvetleniu dvoch algoritmov, ktoré využívajú práve NCP-funkcie. V záverečnej kapitole sme tieto algoritmy implementovali v MATLABe a porovnáli ich výsledky pri rôznych nastaveniach parametrov a pre rôzne lineárne programy.

Kľúčové slová: NCP-funkcie, Predikčno-Korekčná metóda, lineárne programovanie

Abstract

Rizman, Thomas: The use of NCP-functions in linear programming [Bachelor's thesis], Comenius University in Bratislava, Faculty of Mathematics, Physics and Informatics, Department of Applied Mathematics and Statistics, Supervisor: Mgr. Peter Fusek, PhD., Bratislava, 2012, 42 p.

In our work we deal with NCP-functions. We are writing about their basic properties, the smoothing methods (approximation with smoothing parameter), which makes them an excellent tool for solving linear programs. We are dedicated to a detailed explanation of the two algorithms ,which are using NCP function as a tool for solving linear program. In final chapter we implemented our algorithms in MATLAB and compared their numerical results with different parameter settings for a variety of linear programs.

Keywords: NCP-function, Predictor-Corrector smoothing method, linear programming

Obsah

Zoznam obrázkov

Zoznam tabuliek

Zoznam použitých symbolov

Niečo o použitej symbolike, notácií:

- Všetky normy, napríklad $\|F\|$, bližšie nešpecifikovaného vektora F sú euklidovské normy. Teda $\|F\| = \sqrt{\sum_{i=0}^{k} F(i)^2}$, kde k je dĺžka vektora F a $F(i)$ je jeho i-ta zložka.

- R^n označuje n-dimenzionálny reálny vektorový priestor.

- Horný index v kapitole 3 znamená poradové číslo iterácie. Teda napríklad x^k je vektor x pochádzajúci z k-tej iterácie.

- Symbolom $x \geq 0$, kde $x \in R^n$ myslíme nezápornosť každej zložky vektora x. Podobný význam majú symboly $x \leq 0, x < 0, x > 0$.

- Celkom často budeme používať symboliku $w = (x, \lambda, s)(x \in R^n, \lambda \in R^m, s \in R^n)$, aj keď matematicky by bolo viac správne: $w = (x^T, \lambda^T, s^T)^T$. Samozrejme, $w \in R^{n+m+n}$.

Úvod

"Riešenie lineárneho programu je jedna z najzákladnejších a napriek tomu stále vyzývavých problémov v numerickej matematike."([1])

Aj preto sme sa rozhodli vybrať si ako bakalársku prácu práve riešenie lineárneho programu pomocou NCP-funkcií. NCP-funkcia je funkcia dvoch premenných, ktorá má následovnú vlastnosť. Je nulová práve vtedy, keď obe vstupujúce premenné sú nezáporné a aspoň jedna z nich je nulová, čo sa dá, ako neskôr uvidíme, krásne využiť pre potreby lineárneho programovania.

Keď si v dnešnej dobe človek chce niečo zistiť o nejakej problematike, prvé čo urobí je to, že zadá danú problematiku Googlu. Keď do Googlu zadáme heslo "NCP- function", nájdeme množstvo článkov. Väčšina z nich je však od D. Suna, CH. Kanzowa a pár ďalších autorov ([4],[5],...). Keď však do Googlu zadáme heslo "NCP-funkcia", natrafíme len na jednu diplomovú prácu, kde sú NCP-funkcie len okrajovo spomenuté a medzi ďalšími odkazmi je napríklad skratka NCP, ktorá označuje Ninja Club Praha. Takže teoreticky sa dá povedať, že na slovenské pomery je naša problematika relatívne nepreskúmaná resp. nie veľmi zdokumentovaná. Preto si dávame za cieľ ukázať použitie NCP-funkcií v lineárnom programovaní a hlavne pokúsiť sa dosiahnuť primerané numerické výsledky pomocou algoritmu na riešenie lineárneho programu s využitím NCP-funkcií na nami zvolenom portfóliu úloh.

1 Cieľová úloha

V našej práci sa budeme zaoberať istými špeciálnymi metódami pre lineárne programovanie (LP). Budeme riešiť lineárny program s nezápornými premennými a ohraničeniami v tvare rovností. To znamená, že cieľom bude nájsť optimálne x^* pre túto optimalizačnú úlohu:

$$\begin{aligned} c^T x &\to min \\ Ax &= b \\ x &\geq 0 \end{aligned} \tag{1}$$

O matici A budeme predpokladať, že je rozmeru $m \times n$ a má plnú hodnosť. Môžeme to urobiť bez újmy na všeobecnosti. Ak by totižto hodnosť nebola plná, znamenalo by to, že v matici máme dva lineárne závislé riadky a to podľa príslušných pravých strán, alebo ekvivalentne vektora b (či sú tiež lineárne závislé alebo nie) znamená buď neprípustnosť riešenia alebo zdvojenie ohraničenia (ohraničenia by vyjadrovali to isté a jedno z nich je tam zbytočne). Z tohto predpokladu vyplýva pre rozmer matice že m, to znamená počet riadkov matice A alebo tiež počet ohraničení, je menší alebo rovný n, čiže počtu stĺpcov matice A alebo počtu premenných.
Vektor c bude mať rozmer n a vektor b bude mať rozmer m.
Pre našu úlohu budeme pojmom množina prípustných riešení rozumieť následujúcu množinu (P):

$$P = \{x \in R^n : Ax = b, x \geq 0\}$$

A množinou optimálnych riešení (P^*):

$$P^* = \left\{x^* \in P : c^T x^* \leq c^T x, \forall x \in P\right\}$$

Ako sme už uviedli, budeme sa zaoberať riešením lineárnych programov v tvare (1). Ak by chcel čitateľ použiť náš algoritmus na riešenie úlohy v nejakom inom tvare (rôzne tvary ohraničení, voľné premenné...), vieme ho previesť na tvar (1) následovne:

1. Maximalizáciu môžeme zmeniť na minimilizáciu rovnakého programu zmenou znamienka učelovej funkcie. Takže namiesto riešenia $c^T x \to max$ môžeme ekvivalentne riešiť $-c^T x \to min$.

2. Ak máme program s voľnými (reálnymi) premennými x_i, vieme ho previesť na program s nezápornými premennými následovnou substitúciou: každú voľnú premennú x_i nahradíme rozdielom nových nezáporných $x_i^{'}$ a $x_i^{''}$. Tým získame program s nezápornými premennými. Treba si však uvedomiť, že sme za to zaplatili zdvojnásobením rozmeru (dimenzie) programu. Ak by boli premenné aj voľné, aj nezáporné, substitúciu uskutočníme iba pre voľné premenné.

3. Ak máme niektoré ohraničenia v tvare nerovností, vieme z nich vyrobiť ohraničenia v tvare rovností pridaním nových pomocných premenných. Teda napríklad ak jedno z ohraničení bolo $x_1 + 78x_2 \leq 1000$, pridaním novej nezápornej premennej s získame rovnosť, teda získame $x_1 + 78x_2 + s = 1000$. Opäť sme zaplatili zvýšením rozmeru avšak nie na dvojnásobok ale iba o toľko, koľko ohraničení sme menili.

Takže sme si ukázali, ako si vieme lineárne programy upraviť na tvar, s ktorým sa nám bude dobre pracovať. Pre viac informácií ohľadne tvarov úloh LP, odkazujeme čitateľa na ([3]).
Jedným z často používaných nástrojov na riešenie optimalizačnej úlohy (1) je simplexová metóda([3]), ktorá môže byť alternatívne aplikovaná aj na duálnu optimalizačnú úlohu príslušnú k (1):

$$\begin{aligned} b^T\lambda &\rightarrow max \\ A^T\lambda &\leq c \end{aligned} \tag{2}$$

A tá môže byť prepísaná na:

$$\begin{aligned} b^T\lambda &\rightarrow max \\ A^T\lambda + s &= c \\ s &\geq 0 \end{aligned} \tag{3}$$

Zo Základnej vety lineárneho programovania ([3]) vieme, že pre každú úlohu LP môže nastať jeden z nasledujúcich troch prípadov:

1. Neprípustnosť: Množina prípustných riešení je prázdna.

2. Neohraničenosť: Účelová funkcia je na množine prípustných riešení v smere optimalizácie neohraničená.

3. Optimalita: Úloha má optimálne riešenie.

Keďže jeden z našich cieľov je aj implementácia nášho algoritmu v MATLABe, budeme sa zaoberať najmä tretím prípadom, teda úlohami, ktoré majú optimálne riešenie.
Vďaka silnej vete o dualite ([3]) vieme, že primárna a duálna úloha LP majú istú súvislosť a dokonca, v prípade existencie optimálneho riešenia jednej z nich, existuje aj optimálne riešenie druhej z nich a hodnoty účelových funkcií v týchto optimálnych riešeniach sa rovnajú.
Niečo podobné využívajú aj rôzne metódy vnútorných bodov, ktoré sú často využívanou alternatívou ku simplexovej metóde Georga Dantziga. Pre naše potreby zatiaľ uvedieme iba fakt, že najúspešnejšie metódy vnútorného bodu sú založené na riešení primárno-duálnych podmienok optimality ([1]):

$$\begin{aligned} Ax &= b \\ A^T\lambda + s &= c \\ s_i &\geq 0, i \in \{1, 2 \ldots, n\} \\ x_i &\geq 0, i \in \{1, 2 \ldots, n\} \\ x_i s_i &= 0, i \in \{1, 2 \ldots, n\} \end{aligned} \tag{4}$$

Keďže tieto podmienky optimality sú špeciálnym prípadom Karush-Kuhn-Tuckerových podmienok (ako si o chvíľu ukážeme) optimality obidvoch, primárnej aj duálnej úlohy, nasledujúce tvrdenie bude pravdivé:

$$(1) \text{ má riešenie} \iff (2),(3) \text{ má riešenie} \iff (4) \text{ má riešenie}$$

Z hore uvedeného získame oprávnenie riešiť systém (4) namiesto nášho cieľového programu (1). Budeme ho riešiť ako systém lineárnych a nelineárnych rovníc:

$$\Phi(x, \lambda, s) = 0 \tag{5}$$

Neskôr sa ukáže, že náš systém (5) je nehladký a pre potreby nášho algoritmu zavedieme vyhladzovací parameter $\tau \geq 0$ a k nemu ekvivalentne systém:

$$\Phi_\tau(x, \lambda, s) = 0 \tag{6}$$

Poďme si ešte ukázať, že podmienky (4) sú naozaj len špeciálnym prípadom podmienok

Karusha-Kuhna-Tuckera. Potrebujeme Lagrangeovú funkciu, ktorá pre náš prípad (1) bude vyzerať takto: $L(x, \lambda) = c^T x + \lambda^T (Ax - b)$, pričom $L : R^n \times R^m \to R$. Avšak ako je všeobecne známe, Lagrangeove multiplikátory, teda λ sú pre našu Lagrangeovu funkciu (príslušnú k (1)), totožné s premennými duálnej úlohy. Všeobecne by Karush-Kuhn-Tuckerove podmienky mali tvar:

- $\frac{\partial L}{\partial x} \geq 0$
- $(\frac{\partial L}{\partial x})^T x = 0$
- $x \geq 0$
- $\frac{\partial L}{\partial \lambda} = 0$

A v našom konkrétnom prípade to bude:

- $c - A^T \lambda \geq 0$, pričom vieme že $s = c - A^T \lambda$, a teda v tejto podmienke je aj $s = c - A^T \lambda$ aj $s_i \geq 0$, $i \in \{1, 2 \ldots, n\}$
- $(c - A^T \lambda)^T x = 0$, čo je to isté ako $s_i x_i = 0$, $i \in \{1, 2 \ldots, n\}$
- $x_i \geq 0$ $i \in \{1, 2 \ldots, n\}$
- $Ax = b$

Teraz už je jasné, že naše primárno-duálne podmienky optimality sú naozaj iba špeciálnym prípadom Karush-Kuhn-Tuckerových podmienok a teda môžeme pokračovať druhou kapitolou.

2 NCP-funkcie

2.1 Nelineárny problém komplementarity

Veľká časť článkov popisujúcich NCP-funkcie ich využíva na riešenie nelineárneho problému komplementarity (z anglického nonlinear complementarity problem, skrátene NCP, odtiaľ vyplýva aj názov NCP-funkcia). Poďme sa pozrieť na to, čo to je:

Definícia 1.: Daná je spojito diferencovateľná funkcia $F : R^n \to R^n$. Potom *Nelineárnym problémom komplementarity NCP(F)* nazveme úlohu nájsť vektor $y \in R^n$ taký, že:

$$y \geq 0,\ F(y) \geq 0,\ y^T F(y) = 0$$

Už na pohľad je jasné a literatúrou ([1]) je potvrdené, že nelineárny problém komplementarity je oveľa všeobecnejší problém, ako naša úloha (1). Je však medzi nimi následovný vzťah. Naša úloha (1), ktorá je ekvivalentná primárno-duálnym podmienkam optimality (4), je iba špeciálnym prípadom nelineárneho problému komplementarity. Stačí vhodne zvoliť vektor y a funkciu F. Nasledujúca voľba je jednou z možností:

$$y = \begin{pmatrix} x \\ \lambda' \\ \lambda'' \end{pmatrix} \text{ a } F(y) = \begin{pmatrix} c - A^T(\lambda' - \lambda'') \\ Ax - b \\ -Ax + b \end{pmatrix},$$

kde $y \in R^{n+2m}$; $\lambda', \lambda'' \in R^m_+$; $F : R^{n+2m} \to R^{n+2m}$, pričom λ' a λ'' sú nové nezáporné vektory premenných spĺňajúce $\lambda = \lambda' - \lambda''$. Je dôležité si uvedomiť, že ich voľba je nejednoznačná. Ak by sme chceli jednoznačnosť, museli by sme pridať napríklad podmienku komplementarity.

Poďme sa pozrieť na to, čo nám dajú vlastnosti nelineárneho problému komplementarity, pre nami zvolený vektor y a funkciu $F(y)$.:

$y \geq 0$ nám dáva $x \geq 0$, $F(y) \geq 0$ nám dáva $Ax - b \geq 0$ a $-b + Ax \geq 0$, z čoho hneď vyplýva prípustnosť v ohraničeniach primárnej úlohy((1)), teda $Ax = b$. Z nerovnosti $c - A^T(\lambda' - \lambda'') \geq 0$ vyplýva, že $c - A^T\lambda \geq 0$ a z toho hneď máme (keďže $s = c - A^T\lambda$), $s \geq 0$. V definícií nelineárneho problému komplementarity už ostáva iba $y^T F(y) = 0$. Z tejto vlastnosti pre náš prípad vyplýva: $(\lambda')^T(Ax - b) = 0$ a $(\lambda')^T(-Ax + b) = 0$, čo nám nedáva nič nové, pretože $Ax - b$ je, ako sme si ukázali nulové a teda tieto dve

rovnosti sú splnené. Posledné, čo sme získali vďaka $y^T F y = 0$ je $x^T(c - A^T(\lambda' - \lambda''))$ a to je ekvivalentné $x^T s = 0$.
Teda sme dostali

1. $x \geq 0$, čo je iba vektorovo napísané $x_i \geq 0, i \in \{1, 2 \ldots, n\}$
2. $Ax = b$
3. $A^T\lambda - c = s$ (premenná s je tak určená)
4. $s \geq 0$, čo je znova iba vektorový ekvivalent $s_i \geq 0, i \in \{1, 2 \ldots, n\}$
5. $x^T s = 0$, z čoho máme $x_i s_i = 0, i \in \{1, 2 \ldots, n\}$

Ukázali sme si teda, že vhodnou voľbou funkcie F a vektora y vieme zmeniť nelineárny problém komplementarity na našu úlohu (4).

2.2 NCP-funkcia

Podmienky optimality môžu byť preformulované na systém lineárnych a nelineárnych rovníc použitím NCP-funkcií:

Definícia 2.: Zobrazenie $\varphi : R^2 \to R$ budeme nazývať NCP-funkcia práve vtedy, ak spĺňa násedovné vlastnosti:

$$\varphi(a, b) = 0 \Leftrightarrow a \geq 0\,, b \geq 0\,, ab = 0,$$

to znamená že NCP-funkcia je nulová práve na oboch nezáporných poloosiach.

Nech φ je nejaká NCP funkcia, potom pravú stranu systému (5), teda funkciu $\Phi : R^n \times R^m \times R^n \to R^n \times R^m \times R^n$ definujeme následovne:

$$\Phi(w) = \Phi(x, \lambda, s) = \begin{pmatrix} A^T\lambda + s - c \\ Ax - b \\ \phi(x, s) \end{pmatrix},$$

kde $\phi : R^n \times R^n \to R^n$ je definovaná:

$$\phi(x, s) = (\varphi(x_1, s_1), \varphi(x_2, s_2), \varphi(x_3, s_3), ..., \varphi(x_n, s_n))^T$$

Z toho hneď vyplýva, že $w^* = (x^*, \lambda^*, s^*)$ je riešením podmienok optimality (4) vtedy a len vtedy, keď w^* je riešením systému rovníc:

$$\Phi(w) = 0 \tag{7}$$

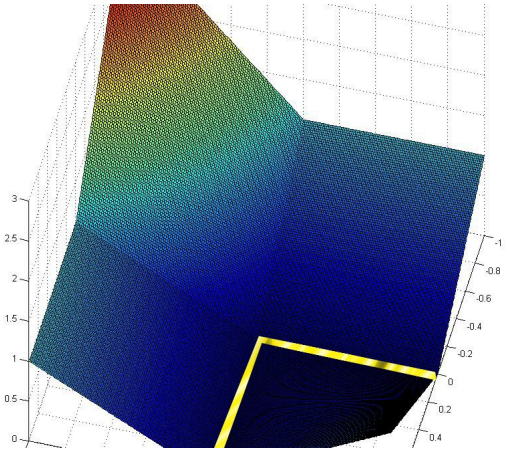
Poďme sa ďalej bližšie zaoberať NCP-funkciami, pre začiatok uveďme nejaké príklady:

1. $\varphi_1(a,b) := 2min\left\{a,b\right\}.$
2. $\varphi_2(a,b) := a + b - \sqrt{a^2+b^2}.$
3. $\varphi_3(a,b) := max(ab,0) + max\left\{-b,-a,0\right\}$
4. $\varphi_4(a,b) := \lambda(a+b-\sqrt{a^2+b^2}) + (1-\lambda)max\left\{0,a\right\}max\left\{0,b\right\}, \lambda \in (0,1).$
5. $\varphi_5(a,b) := \lambda(min\left\{a,b\right\}) + (1-\lambda)max\left\{0,a\right\}max\left\{0,b\right\}, \lambda \in (0,1).$
6. $\varphi_6(a,b) := \sqrt{\left[\varphi_2(a,b)\right]^2 + \alpha\left[max\left\{0,ab\right\}\right]^4}\ \alpha \geq 0$
7. $\varphi_7(a,b) := \sqrt{\left[\varphi_2(a,b)\right]^2 + \alpha\left[max\left\{0,ab\right\}\right]^2}\ \alpha \geq 0$
8. $\varphi_8(a,b) := \sqrt{\left[max\left\{\varphi_2(a,b),0\right\}\right]^2 + \alpha\left[max\left\{0,ab\right\}\right]^2}\ \alpha \geq 0$
9. $\varphi_9(a,b) := \sqrt[p]{|a|^p + |b|^p} - a - b,\ p \in \{1,2,...\infty\}.$

Funkcia φ_1 sa nazýva minimová funkcia (terminológia je iba voľným prekladom terminológie z ([1])) a φ_2 sa nazýva Fisherova-Burmeisterova funkcia. Dvojka pri funkcii φ_1 je tam preto, aby sme ju vedeli pekne aproximovať (ukážeme si v ďalšom odseku). Nie je veľmi ťažké si všimnúť, že funkcie φ_4 a φ_5 sú len zovšeobecnením funkcií φ_1 a φ_2 a preto sa im aj podobne bude hovoriť penalizovaná minimová funkcia a penalizovaná Fisher-Burmeisterova funkcia. Funkcia φ_3 je našou vlastnou úpravou jednej z triedy odvodených funkcií v článku ([6]). Funkcie φ_6, φ_7 a φ_8 sú podľa ([5]) len istou odvodeninou od Fisher-Burmeisterovej funkcie (φ_2). Tá má ešte jednu zaujímavú vlastnosť, ktorú by sme chceli spomenúť a to že jej druhá mocnina $(\varphi_2)^2$ je spojito diferencovateľná ([5]). Dokonca v tom istom článku sa môžeme dočítať, že pre naše príklady funkcií platí: $-\varphi_2$ 0 (mínus je tam kvôli kozmetickému odlíšeniu od ([5])) sa líši od funkcií φ_6, φ_7 a φ_8 iba v prvom alebo treťom kvadrante. Funkcia φ_9 pochádza

z článku([8]) a ako si možno všimnúť, je v istom zmysle generalizovanou, respektíve zovšeobecnenou verziou Fisher-Burmeisterovej-funkcie, kde parameter p vyjadruje druh p-normy, ktorú táto funkcia využíva.

Pre lepšiu predstavu o NCP-funkciách pripájame obrázok funkcie φ_3 (Obr. 1). Nie je ťažké uvidieť, že žltými čiarami sme zvýraznili nulové nezáporné poloosi a z obrázku je evidentné, že mimo nich funkcia rastie do kladných hodnôt. Teda spĺňa definíciu NCP-funkcií.

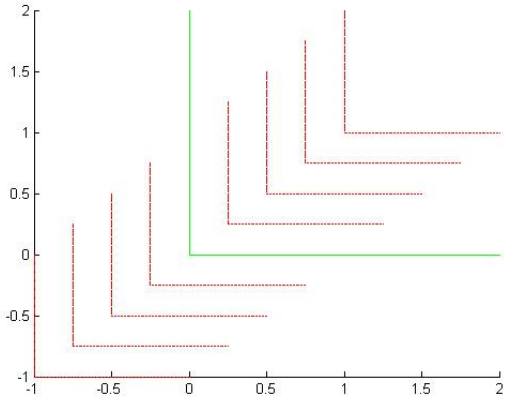


Obr. 1: $\varphi_3(x, y)$, pričom $x, y \in (-1, 1)$

Na následujúcom obrázku (Obr. 2) sme opäť, pre vylepšenie predstavy o NCP-funkciách načrtli graf vrstevníc pre funkciu φ_1. To znamená $\varphi_1(x, y) = c$, pre hodnoty c z následujúcej množiny: $\{-2, -1.5, -1, -0.5, 0, 0.5, 1, 1.5, 2\}$. Vrstevnica pre $c = 0$ je z hľadiska NCP-funkcií najdôležitejšia (zelená farba) a vidíme, že je konzistentná s dôsledkom definície NCP-funkcií (a teda tým, že NCP-funkcia je nulová práve na oboch nezáporných poloosiach).

2.3 Hladké aproximácie NCP-funkcií

Ako si možno všimnúť, či už z obrázkov alebo predpisou NCP-funkcií, tieto funkcie nie sú hladké. Našim cieľom je však použiť tieto funkcie na riešenie sústavy lineárnych a



Obr. 2: vrstevnice funkcie $\varphi_1(x, y)$, pričom $x, y \in (-1, 1)$

nelineárnych rovníc (5). Túto sústavu chceme riešiť aplikovaním Newtonovej metódy, čo pre nás znamená, že potrebujeme zabezpečiť hladkosť systému (5). Ako sme už v predchádzajúcej kapitole načrtli, urobíme to pomocou vyhladzovacieho parametra $\tau \geq 0$, s ktorým sa počas jednotlivých iterácií budeme blížiť k nule a ktorý bude slúžiť na predefinovanie NCP-funkcií. Potom už budeme vedieť aplikovať Newtonovu metódu na systém (6). Poďme si teda podrobnejšie prebrať hore spomenuté. Na začiatok uveďme niekoľko príkladov:

1. $\varphi_{\tau(1)}(a, b) := a + b - \sqrt{(a-b)^2 + 4\tau^2}$.
2. $\varphi_{\tau(2)}(a, b) := a + b - \sqrt{a^2 + b^2 + 2\tau^2}$.
3. $\varphi_{\tau(4)}(a, b) := \lambda\varphi_{\tau(1)} + (1-\lambda)\sqrt{max\{0, a\}^2 \, max\{0, b\}^2 + 4\tau^2}$, $\lambda \in (0, 1)$.
4. $\varphi_{\tau(5)}(a, b) := \lambda\varphi_{\tau(2)} + (1-\lambda)\sqrt{max\{0, a\}^2 \, max\{0, b\}^2 + 2\tau^2}$, $\lambda \in (0, 1)$.
5. $\varphi_{\tau(9)}(a, b) := \sqrt[p]{|a|^p + |b|^p + \tau^p} - a - b$, $p \in \{1, 2, ...\infty\}$.

Hore uvedené hladké aproximácie sú príslušné ku funkciám uvedeným v odseku 2.2. Teda napríklad $\varphi_{\tau(1)}$ je príslušnou hladkou aproximáciou funkcie φ_1. Konkrétne táto aproximácia sa nazýva Chen-Harker-Kanzow-Smaleová vyhladzovacia funkcia. Je to jedna z možností, ako hladko aproximovať minimovú funkciu. Podstatné je to, že ak zvolíme parameter $\tau = 0$, dostávame $\varphi_{0(1)}(a, b) = a+b-\sqrt{(a-b)^2} = a+b-|(a-b)| = 2min\{a, b\} = \varphi_1(a, b)$ a teda sme si vysvetlili napríklad aj význam konštanty 2 pred minimovou funkciou.

Ako sme v úvode spomínali, použijeme tieto hladké aproximácie NCP-funkcií na preformulovanie systému lineárnych a nelineárnych rovníc (5) na systém (6). Teda budeme riešiť $\Phi_\tau = 0$, kde $\Phi_\tau : R^n \times R^m \times R^n \to R^n \times R^m \times R^n$ je definované následovne:

$$\Phi_\tau(w) = \Phi_\tau(x, \lambda, s) = \begin{pmatrix} A^T\lambda + s - c \\ Ax - b \\ \phi_\tau(x, s) \end{pmatrix},$$

kde $\phi_\tau : R^n \times R^n \to R^n$ je definovaná:

$$\phi_\tau(x, s) = (\varphi_\tau(x_1, s_1), \varphi_\tau(x_2, s_2), \varphi_\tau(x_3, s_3), ..., \varphi_\tau(x_n, s_n))^T$$

S parametrom τ pôjdeme počas jednotlivých iterácií k nule, čo zabezpečí zhodu riešení sústav rovníc (5) a (6) (na určitú aproximačnú presnosť). Robíme teda niečo podobné ako robia metódy vnútorného bodu. Tie totižto riešia série bariérových úloh (podmienok):

$$\begin{aligned} Ax &= b \\ A^T\lambda + s &= c \\ s_i &\geq 0, i \in \{1, 2 \ldots, n\} \\ x_i &\geq 0, i \in \{1, 2 \ldots, n\} \\ x_i s_i &= \tau^2, i \in \{1, 2 \ldots, n\} \end{aligned} \tag{8}$$

kde $\tau \geq 0$. Na každú barierovú úlohu (8) aplikujú tak isto ako budeme aj my, pre našu sústavu (6), nejaký druh Newtonovej metódy, pomocou ktorej hľadajú centrálnu trajektóriu, ktorá zaručuje kladnosť primárnych a duálnych premenných(x, s).
Jeden z hlavných rozdielov medzi našim algoritmom a metódami vnútorného bodu bude aj to, že pozitívnosť primárnych a duálnych premenných už nebude počas jednotlivých iterácií zaručená. Čitateľa odkazujeme na ([9]), pre viac informácií o metódach vnútorného bodu.

Lema 1.: Pre každú z horeuvedených hladkých aproximácií NCP-funkcií existuje konštanta $c \geq 0$, nezávislá od τ, tak že platí:

$$\left|\varphi_{\tau(i)}(a, b) - \varphi_i(a, b)\right| \leq c\tau$$

Dôkaz: Keďže v praktickej časti budeme používať funkciu $\varphi_{\tau(1)}$, budeme uvádzať tento a aj nasledujúce dôkazy najmä pre túto funkciu.

Chceme dokázať: $\left|\varphi_{\tau(1)}(a,b) - \varphi_1(a,b)\right| \leq c\tau$, platí:

$$\begin{aligned}
& \left|\varphi_{\tau(1)}(a,b) - \varphi_1(a,b)\right| = \\
= & \left|a+b-\sqrt{(a-b)^2+4\tau^2} - (a+b-\sqrt{(a-b)^2})\right| = \\
= & \left|-\sqrt{(a-b)^2+4\tau^2} + \sqrt{(a-b)^2}\right| = \\
= & \sqrt{(a-b)^2+4\tau^2-2\sqrt{(a-b)^2((a-b)^2+4\tau^2)}+(a-b)^2} = \\
= & \sqrt{2(a-b)^2+4\tau^2-2\sqrt{4(a-b)^2\tau^2}-2(a-b)^2} = \\
= & \sqrt{4\tau^2-2\sqrt{4(a-b)^2\tau^2}} = \\
= & \sqrt{4\tau^2-4\left|a-b\right|\tau} \leq \sqrt{4\tau^2} = 2\tau
\end{aligned}$$

Z poslednej nerovnice je už zrejmé, že voľba $c = 2$ nám zaručí jej platnosť. Pre ostatné funkcie je dôkaz veľmi podobný, spomenieme však že napríklad pre funkciu $\varphi_{\tau(2)}$ by nám konštanta c vyšla $c = \sqrt{2}$ a rovnako by nám konštanty vyšli aj pre príslušné penalizované funkcie.

Podobnú lemu môžeme sformulovať aj pre náš hladký systém lineárnych a nelineárnych rovníc $(\Phi_\tau(w) = 0)$ využívajúci niektorú s NCP-funkcií :

Lema 2.: Pre systém lineárnych a nelineárnych rovníc $\Phi_\tau(w)(6)$, existuje konštanta $K \geq 0$ nezávislá od τ a w, tak že platí:

$$\|\Phi_\tau(w) - \Phi(w)\| \leq K\tau$$

Dôkaz: Dôkaz pre (6), využívajíci $\varphi_{\tau(1)}$ je len použitie lemy 1, a teda zvolenie $K = \sqrt{n}2$, konkrétne pre minimovú NCP-funkciu, pričom všeobecne by sme volili $K = \sqrt{n}c$, kde c je konštanta z lemy 1.

2.4 Jacobiho matica pre sústavu (6)

Keďže sme hladko aproximovali NCP-funkcie, Jacobiho matica pre systém lineárnych a nelineárnych rovníc $\Phi_\tau(w) = 0$, bude existovať pre ľubovolný vektor w a parameter $\tau \geq 0$. Chceme teda nájsť $\frac{\partial \Phi_\tau}{\partial w}$, čo budeme ďalej označovať Φ'_τ. Poďme na to však

postupne a urobme si potrebnú prípravu:

$$\frac{\partial \varphi_{\tau(1)}(x_i, s_i)}{\partial x_i} = 1 - \frac{x_i - si}{\sqrt{(x_i - s_i)^2 + 4\tau^2}}$$
$$\frac{\partial \varphi_{\tau(1)}(x_i, s_i)}{\partial x_j} = 0 \; a \; preto :$$
$$\frac{\partial \phi_{\tau(1)}(x, s)}{\partial x} = D_{\tau,x},$$

kde $D_{\tau,x}$ je diagonálna matica, pričom platí: $D_{\tau,x}(i,i) = 1 - \frac{x_i - s_i}{\sqrt{(x_i - s_i)^2 + 4\tau^2}}$, pre $i \in \{1, 2 \ldots, n\}$ a $D_{\tau,x}(i,j) = 0$, pre $\forall i \neq j$ (symbolikou $D(i,j)$ máme na mysli prvok matice D v i-tom riadku a j-tom stĺpci). Ďalej elementárnou matematikou a derivovaním získame:

$$\frac{\partial \varphi_{\tau(1)}(x_i, s_i)}{\partial s_i} = 1 - \frac{-x_i + si}{\sqrt{(x_i - s_i)^2 + 4\tau^2}}$$
$$\frac{\partial \varphi_{\tau(1)}(x_i, s_i)}{\partial s_j} = 0 \; a \; preto :$$
$$\frac{\partial \phi_{\tau(1)}(x, s)}{\partial s} = D_{\tau,s},$$

kde $D_{\tau,s}$ je diagonálna matica, pričom platí: $D_{\tau,s}(i,i) = 1 - \frac{-x_i + s_i}{\sqrt{(x_i - s_i)^2 + 4\tau^2}}$, pre $i \in \{1, 2 \ldots, n\}$ a $D_{\tau,s}(i,j) = 0$, pre $\forall i \neq j$.

Keďže ostatné čiastkové derivácie sú buď derivácie lineárnych funkcií alebo evidentne nulové, uvádzame rovno výsledok:

$$\Phi'_\tau(w) = \begin{pmatrix} 0 & A^T & I \\ A & 0 & 0 \\ D_{\tau,x} & 0 & D_{\tau,s} \end{pmatrix} \tag{9}$$

Pre niektorú inú funkciu z funkcií φ hore uvedených by sme postupovali podobne.

Lema 3: Jacobiho matica $\Phi'_\tau(w)$ je regulárna pre ľubovolný vektor $w = (x, \lambda, s) \in R^n \times R^m \times R^n$ a pre ľubovolný parameter $\tau \geq 0$.

Dôkaz: Všimnime si najprv, do akého intervalu patrí nasledujúca derivácia: $\frac{\partial \varphi_{\tau(1)}(x_i, s_i)}{\partial x_i} = 1 - \frac{x_i - si}{\sqrt{(x_i - s_i)^2 + 4\tau^2}}$. Pôjde o to, aké hodnoty môže nadobúdať nasledujúci člen: $\frac{x_i - si}{\sqrt{(x_i - s_i)^2 + 4\tau^2}}$.

Zjavne limita $\frac{x_i - si}{\sqrt{(x_i - s_i)^2 + 4\tau^2}}$ pre $\tau \to \infty$ bude nulová a limita pre $\tau \to 0$ dokonverguje do 1 alebo -1, v závislosti od x_i a s_i. Teda bude platiť, že celková hodnota $\frac{\partial \varphi_{\tau(1)}(x_i, s_i)}{\partial x_i} \in (0, 2)$, z čoho vyplýva že matica $D_{\tau,x}$ je kladne definitná. Rovnakým postupom by sme ukázali kladnú definitnosť aj pre $D_{\tau,s}$.

Zoberme teraz nejaký vektor $r = (r_1, r_2, r_3) \in R^n \times R^m \times R^n$, ktorý spĺňa $\Phi'_\tau(w)r = 0$. Ak sa nám podarí ukázať, že vektor r je nulový, dokázali sme regularitu $\Phi'_\tau(w)$.

Podľa (9) dostávame:

$$A^T r_2 + r_3 = 0 \tag{10}$$

$$Ar_1 = 0 \tag{11}$$

$$D_{\tau,x} r_1 + D_{\tau,s} r_3 = 0 \tag{12}$$

Keď prenásobíme (10) r_1^T sprava, dostaneme $r_1^T A^T r_2 + r_1^T r_3 = 0$. Keď si všimneme aj (11), dostaneme $r_1^T r_3 = 0$. Keďže $D_{\tau,x}$ je kladne definitná, môžeme (12) prenásobiť $D_{\tau,x}^{-1}$. Z toho dostaneme vzťah pre r_1, a keď ho skombinujeme s $r_1^T r_3 = 0$, dostaneme $r_3^T D_{\tau,x}^{-1} D_{\tau,s} r_3 = 0$. Spolu z tohto a faktu, že matice $D_{\tau,x}^{-1}$ a $D_{\tau,s}$ sú diagonálne a kladne definitné, máme hneď $r_3 = 0$. Z (12) potom dostaneme $r_1 = 0$ a z (10) spolu s predpokladom o plnej hodnosti matice A máme $r_2 = 0$. Teda sme ukázali, že $r = 0$ a tým aj regularitu matice $\Phi'_\tau(w)$ pre všetky $w \in R^n \times R^m \times R^n$ a pre ľubovolný parameter $\tau \geq 0$.

2.5 Ďalšie vlastnosti NCP-funkcií

V nasledujúcom odseku uvedieme iba zoznam ďalších vlastností bez dôkazu, ich dôkaz sa nachádza v článku ([1]).:

1. Pre NCP-fukcie φ_4 a φ_5, množinu L_c:

$$L_c := \left\{ w = (x, \lambda, s) \in R^n \times R^m \times R^n | Ax = b, A^T\lambda + s = c, \|\Phi(w)\| \leq c \right\}$$

a pre ľubovolnú konštantu $c \in R$ platí nasledovná vlastnosť:

Ak existuje striktne prípustný vektor pre primárno-duálne podmienky optimality (4), potom množina L_c je kompaktná.

2. Pre NCP-fukcie φ_4 a φ_5, množinu $L_{c,\tau}$:

$$L_{c,\tau} := \{w = (x, \lambda, s) \in R^n \times R^m \times R^n | Ax = b, A^T\lambda + s = c, \|\Phi_\tau(w)\| \leq c\},$$

pre ľubovolnú konštantu $c \in R$ a $\tau \geq 0$, platí nasledovná vlastnosť:
Ak existuje striktne prípustný vektor pre primárno-duálne podmienky optimality(4), potom množina $L_{c,\tau}$ je ohraničená.
Takisto pre množinu $L_{c,\tau}$ platí následovné:

$$\text{Množina } \bigcup_{\langle 0,\hat{\tau}\rangle} L_{c,\tau} \text{ je ohraničená pre každé } \hat{\tau} \geq 0 \text{ a každé } c \in R.$$

3. Uvažujme NCP-fukcie φ_4 a φ_5. Predpokladajme existenciu striktne prípustného vektoru pre primárno-duálne podmienky optimality (4). Potom systém lineárnych a nelineárnych rovníc (6) má riešenie pre každé $\tau \geq 0$. Zaujímavé je ešte spomenúť, že jednoznačnosť tohto riešenia je nejasná (v rámci horeuvedených viet bude určite izolované, avšak toto nevylučuje existenciu viacerých riešení). Naše algoritmy však budú s vyhladzovacím parametrom konvergovať k 0, preto sa týmto problémom ďalej zaoberať nemusíme.

Aby nám to všetko dávalo zmysel, poslednú vlastnosť si aj dokážeme. Predtým si ale spomenieme ešte pár faktov, s ktorými potrebujeme byť oboznámení. Pre zjednodušenie zápisu zavedieme funkciu $\Psi_\tau(w) : R^n \to R$, $\Psi_\tau(w) = \frac{1}{2}\|\Phi_\tau(w)\|^2$. A triviálnymi úpravami dôjdeme aj k pozorovaniu, že $\Psi'_\tau(w) = \Phi'_\tau(w)\Phi_\tau(w))$.

Dôkaz: Nech $\tau > 0$ je fixné, uvažujme optimalizačný problém:

$$\Psi_\tau(w) \to min$$

$$Ax = b$$

$$A^T\lambda + s = c$$

Keďže A má plnú hodnosť, množina $L_{c,\tau}$ je neprázdna (pre dostatočne veľkú konštantu $c > 0$) a kompaktná. Z toho dôvodu hore uvedený optimalizačný problém nadobúda svoje globálne minimum, vektor $w_\tau = (x_\tau, \lambda_\tau, s_\tau)$. Naším cieľom bude ukázať, že tento vektor je riešením systému lineárnych a nelineárnych rovníc (6). Horeuvedený optimalizačný problém je lineárny v ohraničeniach, ktoré sú v tvare rovností. Preto budú

existovať Lagrangeové multiplikátory $\mu \in R^n$ a $\eta \in R^m$, tak že w_τ, μ a η budú spĺňať Karush-Kuhn-Tuckerové podmienky pre horeuvedenú optimalizačnú úlohu:

$$\nabla \Psi_\tau(w_\tau) + \begin{pmatrix} A^T\eta \\ A\mu \\ \mu \end{pmatrix} = 0.$$

Keďže $\Psi'_\tau(w) = \Phi'_\tau(w)^T \Phi_\tau(w))$, môžeme tieto KKT podmienky prepísať následovne:

$$\begin{pmatrix} 0 & A & D_{\tau,x} \\ A^T & 0 & 0 \\ I & 0 & D_{\tau,s} \end{pmatrix} \begin{pmatrix} A^T\lambda_\tau + s_\tau - c \\ Ax_\tau - b \\ \phi_\tau(x_\tau, s_\tau) \end{pmatrix} + \begin{pmatrix} A^T\eta \\ A\mu \\ \mu \end{pmatrix} = 0.$$

Toto môžeme prepísať na tvar troch rovností, následovne (využijeme pri tom fakt, že w_τ je prípustný pre hore uvedenú optimalizačnú úlohu):

1. $D_{\tau,x}\phi_\tau(x_\tau, s_\tau) = -A^T\eta$
2. $A\mu = 0$
3. $D_{\tau,s}\phi_\tau(x_\tau, s_\tau) = -\mu$

Keď prenásobeníme tretiu rovnicu maticou A, vezmeme do úvahy druhú rovnicu a vyjadrením a dosadením $\phi_\tau(x_\tau, s_\tau)$ z prvej rovnice, dostaneme:

$$AD_{\tau,s}D_{\tau,x}^{-1}A^T\eta = 0.$$

Teraz už len spojením predpokladu o plnej hodnosti matice A a kladnej definitnosti matíc $D_{\tau,x}$ a $D_{\tau,x}$, získame $\eta = 0$. Potom z tretej rovnice vyplýva $\phi_\tau(x_\tau, s_\tau) = 0$. Nakoniec znova použijeme fakt, že w_τ je prípustné riešenie hore uvedeného optimalizačného problému a spolu s $\phi_\tau(x_\tau, s_\tau) = 0$ uvidíme, že je aj riešením sústavy lineárnych a nelineárnych rovníc (6).

3 Algoritmy používajúce NCP-funkcie

V tejto kapitole si uvedieme ako prakticky využiť NCP-funkcie definované v kapitole 2. Základnou myšlienkou je aplikácia globalizovanej Newtonovej metódy na systém lineárnych a nelineárnych rovníc (6). Vďaka vyhladzovaciemu parametru $\tau > 0$, ktorý bude konvergovať k nule, je Jacobiho matica regulárna a teda Newtonov krok vieme vyrátať.

3.1 Algoritmus 1

1. Voľba štartovacieho bodu $w^0 := (x^0, \lambda^0, s^0) \in R^n \times R^m \times R^n, \tau_0 \geq 0, \beta \geq \|\Phi_{\tau_0}(w^0)\| / \tau_0, \rho \in (0,1), \sigma \in (0,1), \epsilon \geq 0$ a nastaviť $k = 0$.

2. Ak $\|\Phi_{\tau_0}(w_0)\| \leq \epsilon$, koniec.

3. Výpočet kroku $\Delta w^k = (\Delta x^k, \Delta \lambda^k, \Delta s^k) \in R^n \times R^m \times R^n$, podľa nasledovného vzťahu:

$$\Phi'_{\tau_k}(w^k)\Delta w^k = -\Phi_{\tau_k}(w^k) \tag{13}$$

4. Výpočet pomocného parametra $t_k = max\left\{\rho^l | l = 0, 1, 2, 3, \ldots\right\}$, pričom dole uvedená podmienka musí ostať splnená:

$$\Psi_{\tau_k}(w^k + t_k \Delta w^k) \leq \Psi_{\tau_k}(w^k) + t_k \sigma \nabla \Psi_{\tau_k}(w^k)^T \Delta w^k \tag{14}$$

   A nastaviť $w^{k+1} = w^k + t_k \Delta w^k$

5. Vypočítať $\gamma_k = max\left\{\rho^l | l = 0, 1, 2, 3, \ldots\right\}$ tak, aby platilo:

$$\left\|\Phi_{(1-\gamma_k)\tau_k}(w^{k+1})\right\| \leq \beta(1-\gamma_k)\tau_k \tag{15}$$

   A voľba nového $\tau_{k+1} = (1-\gamma_k)\tau_k$.

6. Prechod na ďalšiu iteráciu, teda $k \leftarrow k+1$ a návrat do kroku 2.

Hore uvedený algoritmus pochádza z knihy C. Ciegeva a Ch. Kanzowa ([10]). Pozrime sa na to, ako funguje. V prvom bode zvolíme štartovací vektor w^0 a príslušný parameter τ_0, ktorým aproximujeme zvolenú NCP-funkciu. Takisto konštantu β, ktorá bude v

istom zmysle určovať kritérium príjmania nového kroku. Všimnime si fakt, že keby sme ju zvolili najmenšiu možnú, teda $\beta = \|\Phi_{\tau_0}(w^0)\| / \tau_0$, hodnota $\beta\tau_0$ bude rovná norme vektora $\Phi_{\tau_0}(w^0)$. Teda β určuje spolu s parametrom τ kritérium maximálnej normy, ktorú môže dosiahnuť w^k v k-tej iterácií. Vďaka tomu, že počas procesu iterovania bude τ_k konvergovať do nuly, bude aj $\beta\tau_k$ konvergovať k nule. Keďže norma vektora $\Phi_{\tau_k}(w^k)$ v k-tej iterácií je vždy menšia ako $\beta\tau_k$, bude aj norma $\Phi_{\tau_k}(w^k)$ konvergovať k nule, čo znamená, že sme našli optimálne riešenie. Ďalej volíme parameter ρ, ktorý určuje rýchlosť klesania dĺžky kroku pri jej optimalizácií (jej optimalizáciou máme na mysli najväčšiu možnú dĺžku, pri ktorej podmienka prijatia kroku ostane stále splnená). Teda ak sa napríklad rovná 0.5, znamená to že pri procese skúšania, ktorá dĺžka kroku je najlepšia, spôsobilosť dĺžok testujeme v tvare mocnín 0.5. Teda pri veľkej väčšine prípadov je lepšie zvoliť skôr väčšie ρ, aby sme vyskúšali viac možností, ale nie zas príliš veľké, aby tých možností nebolo príliš veľa. Napríklad keby sme zvolili $\rho = 0.001$, teda relatívne malé, budeme testovať vhodnosť najprv celého kroku Δw^k, potom $0,001\Delta w^k$ a potom $0,000001\Delta w^k$ a tak ďalej. Ak by bola optimálna dĺžka napríklad 0.7, príjmeme krok $0,001\Delta w^k$, čo je od optimálneho celkom vzdialené.

Podobnú funkciu má ρ aj pri zmenšovaní parametra τ_k. Podobne ako pri dĺžke kroku určuje, ako rýchlo budeme τ_k zmenšovať a popritom testovať, či sme neporušili podmienku prijatia kroku, teda či $\Phi_{\tau_k}(w^k) \leq \beta\tau_k$. Opäť keď je ρ veľmi malé, môže sa stať, že τ_k zmenšíme oveľa menej ako sme mohli, čo nie je dobré.

Parameter σ reprezentuje len akúsi konštantu v podmienke (14), pričom môže v podstate vytvoriť priestor na následné zrýchlenie procesu hľadania optimálnej dĺžky kroku, ale na druhú stranu, ak je príliš veľká, môže nastať situácia, že budeme príjmať kroky, ktoré sú veľmi vzdialené od optimálnych a tým vzniknú zbytočné iterácie. Posledné sa volí ϵ, ktoré určuje presnosť, akú chceme dosiahnuť počas nášho algoritmu.

Keď máme všetky parametre zvolené a vysvetlené, môžeme prejsť na druhý bod Algoritmu 1. Ten iba otestuje, či sme už dostatočne blízko k optimálnemu riešeniu a ak áno, ukončí algoritmus. Ak však nie, vypočíta sa Newtonov smer podľa (13) a hľadá sa jeho optimálna dĺžka t_k, ktorú chceme čo najväčšiu ale zároveň takú, aby (14) bolo splnené aj v novej iterácii. Potom sa snažíme čo najviac zmenšiť parameter τ_k, samozrejme opäť tak, aby sme neporušili kritérium príjmania nového kroku (15) pre novú iteráciu.

Keď sme toto zvládli, môžeme prejsť na ďaľšiu iteráciu, otestovať či už nie sme dostatočne blízko k optimálnemu riešeniu a ak nie tak znova vypočítame Newtonov krok podľa (13), nájdeme jeho optimálnu dĺžku a čo najviac zmenšíme parameter τ_k.
Toto opakujeme, až kým nedokonvergujeme k optimálnemu riešeniu (na ϵ-ovú presnosť).

3.2 Algoritmus 2

1. Voľba štartovacieho bodu $w^0 := (x^0, \lambda^0, s^0) \in R^n \times R^m \times R^n, \tau_0 \geq 0, \beta \geq \|\Phi_{\tau_0}(w^0)\| / \tau_0, \rho \in (0,1), \sigma \in (0,1), \epsilon \geq 0$ a nastaviť $k = 0$.

2. Ak $\|\Phi_{\tau_0}(w_0)\| \leq \epsilon$, koniec.

3. **Predikčný krok:**

   - Výpočet kroku $\Delta w^k = (\Delta x^k, \Delta \lambda^k, \Delta s^k) \in R^n \times R^m \times R^n$, podľa nasledovného vzťahu:

   $$\Phi'_{\tau_k}(w^k)\Delta w^k = -\Phi(w^k) \tag{16}$$

   - Ak $\left\|\Phi(w^k + \Delta w^k)\right\| \leq \epsilon$, koniec.
   - Ak $\left\|\Phi_{\tau_k}(w^k + \Delta w^k)\right\| \geq \beta\tau_k$, tak zvolíme:
     $\hat{w^k} := w^k$;
     $\hat{\tau^k} := \tau_k$;
   - Inak počítame $t_k = \rho^{l_k}$, kde l_k je nezáporné celé číslo spĺňajúce:

   $$\left\|\Phi_{\tau_k\rho^j}(w^k + \Delta w^k)\right\| \leq \beta\tau_k\rho^j, \text{ pre } \forall j \in \{0, 1, 2, \ldots, l_k\}$$
   $$\text{a } \left\|\Phi_{\tau_k\rho^{l_k+1}}(w^k + \Delta w^k)\right\| \geq \beta\tau_k\rho^{l_k+1}$$

   A potom zvolíme:

   $$\hat{\tau^k} = \tau^k t_k,$$
   $$\hat{w^k} = w^k, \text{ ak } l_k = 0,$$
   $$\text{inak } \hat{w^k} = w^k + \Delta w^k;.$$

4. **Korekčný krok:**

   - Výpočet kroku $\Delta\hat{w^k} = (\Delta\hat{x}^k, \Delta\hat{\lambda}^k, \Delta\hat{s}^k) \in R^n \times R^m \times R^n$, podľa nasledovného vzťahu:

$$\Phi'_{\hat{\tau_k}}(\hat{w^k})\Delta\hat{w^k} = -\Phi_{\hat{\tau_k}}(\hat{w^k}) \tag{17}$$

   - Výpočet pomocného parametra $\hat{t_k} = max\left\{\rho^l | l = 0, 1, 2, 3, \ldots\right\}$, pričom dole uvedená podmienka musí ostať splnená:

$$\Psi_{\hat{\tau_k}}(\hat{w^k} + \hat{t_k}\Delta\hat{w^k}) \leq \Psi_{\hat{\tau_k}}(\hat{w^k}) + \hat{t_k}\sigma\nabla\Psi_{\hat{\tau_k}}(\hat{w^k})^T\Delta\hat{w^k} \tag{18}$$

     A nastaviť $w^{k+1} = \hat{w^k} + \hat{t_k}\Delta\hat{w^k}$

   - Vypočítať $\gamma_k = max\left\{\rho^l | l = 0, 1, 2, 3, \ldots\right\}$, tak aby platilo:

$$\left\|\Phi_{(1-\gamma_k)\hat{\tau_k}}(w^{\hat{k}+1})\right\| \leq \beta(1-\gamma_k)\hat{\tau_k} \tag{19}$$

     A voľba nového $\tau_{k+1} = (1-\gamma_k)\hat{\tau_k}$.

5. Prechod na ďalšiu iteráciu, teda $k \leftarrow k+1$ a návrat do kroku 2.

Na prvý pohľad si môžeme všimnúť, že 4-tý krok a teda korekčný krok, robí presne to isté ako Algiritmus 1. Čo je však nové, to je krok 3 a teda predikčný krok. Najhlavnejší rozdiel medzi týmito dvoma krokmi je vo výpočtoch kroku a teda v rovniciach (16) a (17). Sústava rovníc (16) narozdiel od (17) nemôže byť interpretovaná ako obyčajná Newtonova rovnica. Totižto pre sústavu (17), aj pravá aj ľavá strana rovnice závisí od vyhladzovacieho parametra τ a v sústave (16) je pravá strana od vyhladzovacieho parametra nezávislá.

Pozrime sa na to, ako predikčný krok funguje. Tento krok najprv vypočíta vektor Δw podľa rovnice (16). Potom podľa veľmi podobného kritéria ako sme opisovali pri algoritme 1 rozhodne, či je krok prijatý alebo nie. V prípade, že by nebol prijatý ani jeden krát (teda v každej iterácií algoritmu), Algoritmy 1 a 2 sú totožné. Ak je krok Δw prijatý, snažíme sa čo najviac zmenšiť vyhladzovací parameter τ. Ak sa nám ho však zmenšiť nepodarí (teda nastane situácia kedy $l_k = 0$), krok neprijmeme aj napriek tomu, že podľa príjmacieho kritéria bol vyhovujúci.

Po tomto kroku prejdeme na krok, ktorý už dobre poznáme a to je krok 4, teda algoritmus 1. Po vykonaní kroku 4, teda korekčného kroku, prejdeme na krok 5, ktorý ukončí túto iteráciu a nastaví algoritmus na začiatok novej. Celý tento postup opakujeme až kým nezískame požadovanú presnosť.

3.3 Vlastnosti algoritmov

V nasledujúcom odseku uvedieme zoznam vlastností, ktoré platia pre naše algoritmy a môžu slúžiť ako ozrejmenie princípov ich fungovania. Dôkazy uvádzať nebudeme, čitateľa s prípadným záujmom odkazujeme na ([1]).

- Dĺžka kroku t_k algoritmu 1 (rovnako dĺžka predikčného kroku v algoritme 1) a tiež dĺžka $\hat{t_k}$ pre korekčný krok algoritmu 2 je jednoznačne definovaná.

- Iterácie w^k generované algoritmami 1 alebo 2 spĺňajú podmienku $\left\|\Phi_{\tau_k}(w^k)\right\| \leq \beta\tau_k$ pre každé $k \in \{0, 1, 2, \ldots\}$.

- Predpokladajme, že štartovací vektor $w^0 = (x^0, \lambda^0, s^0) \in R^n \times R^m \times R^n$ spĺňa rovnice $Ax^0 = b$ a $A^T\lambda^0 + s^0 = c$. Potom všetky iterácie generované algoritmami 1 alebo 2, spĺňajú tieto rovnice tiež. To znamená máme $Ax^k = b$ a $A^T\lambda^k + s^k = c$ pre každé $k \in \{0, 1, 2, \ldots\}$.

- Predpokladajme, že postupnosť $\left\{w^k = (x^k, \lambda^k, s^k) \in R^n \times R^m \times R^n\right\}$ generovaná iterovaním algoritmov 1 alebo 2 má aspoň jeden hromadný bod. Potom postupnosť vyhladzovacích parametrov $\{\tau_k\}$ príslušných k daným iteráciám konverguje k nule.

- Každý hromadný bod postupnosti $\left\{w^k = (x^k, \lambda^k, s^k) \in R^n \times R^m \times R^n\right\}$ generovanej algoritmami 1 alebo 2, je riešením primárno duálnych podmienok optimality (4).

4 Numerické výsledky

4.1 Generovanie lineárnych programov

Potom, ako sme implementovali naše algoritmy v MATLABe, vyskytol sa problém hľadania testovacieho portfólia. Potrebovali sme totiž lineárne programy v tvare (1), ktoré sú navyše prístupné. Keďže matica A má rozmer $m \times n$ a premenné majú byť nezáporné, na cieľovú úlohu (1) sa dá pozrieť ako na lineárny program s n premennými a $n+m$ ohraničeniami. Z toho vyplýva, že vo veľkom množstve náhodne vygenerovaných prípadov nastane neprípustnosť. Obzvlášť keby sme generovali iba nezáporné prvky elementou A, b a c. Keď sme však generovali náhodné prvky matice A (rozmery na úrovni okolo $n = 2m$) a takisto sme pre každý z nich vygenerovali náhodné znamienko, vo veľkej časti sme dospeli k prípustnému lineárnemu programu (príslušné vektory b, c stačilo vygenerovať nezáporne).

Je potrebné podotknúť, že sme predpokladali regularitu matice A a pri tomto postupe nie je regularita zaručená. Avšak možností ako vygenerovať jeden prvok matice je napríklad pri 16-cifrovej presnosti 10^{16}, z čoho hneď vyplýva, že pravdepodobnosť vzniku singularity určite nebude významná.

Nakoniec sme horeuvedeným spôsobom vygenerovali matice $A, B, C, D, E, F, G, H, I, J$, ich prístupnosť sme overili matlabovskou funkciu určenou na lineárne programovanie a spustili na nich naše algoritmy.

4.2 Štartovacie vektory w_0

Vygenerované programy sme testovali s použitím 4 rôznych štartovacích vektorov. Poďme sa pozrieť na to, aké su medzi nimi rozdiely:

- **Vektor 1.:** pre tento vektor platí následovné:
  1. $x(i) = 0, i \in \{0, 1, \ldots n\}$
  2. $s(i) = 0, i \in \{0, 1, \ldots n\}$
  3. $\lambda(i) = 1, i \in \{0, 1, \ldots m\}$

Tento štartovací vektor je veľmi jednoduchý, vo väčšine prípadov v ňom nie je splnené ani $Ax = b$, ani $A^T\lambda + s = c$. Ako však uvidíme neskôr, niekedy s ním dosiahneme prekvapivo dobré výsledky. To môže byť napríklad preto, že náhodné prvky matice A a vektorou b, c sú z inervalu $(0, 1)$ a my sme zvolili vektor, ktorý obsahuje hraničné čísla tohto intervalu, a teda niekedy môže byť celkom blízko k optimálnemu riešeiu.

- **Vektor 2.:**

  1. $x = A^T y$, kde y je riešením $AA^T y = b$
  2. λ je riešením $AA^T\lambda = Ac$
  3. $s = c - A^T\lambda$

  Tento štartovací vektor je o niečo komplikovanejší. Vďaka spôsobu voľby jednotlivých elementov štartovacieho vektora bude platiť $Ax = b$ a takisto aj $A^T\lambda + s = c$, a teda vektor $\Phi_{\tau_0}(w^0)$ bude mať na začiatku prvých $n + m$ elementov nulových.

- **Vektor 3.:**

  1. $x = A^T y$, kde y je riešením $AA^T y = b$
  2. $s(i) = c(i), i \in \{0, 1, \ldots n\}$
  3. $\lambda(i) = 1, i \in \{0, 1, \ldots m\}$

  Pri tomto spôsobe voľby štartovacieho vektora postupujeme o niečo komplikovanejšie ako pri vektore 1 ale jednoduchšie ako pri vektore 2. Vďaka spôsobu voľby jednotlivých elementov štartovacieho vektora bude platiť $Ax = b$ a teda vektor $\Phi_{\tau_0}(w^0)$ bude mať na začiatku prvých n elementov nulových.

- **Vektor 4.:**

  1. $x(i) = 10000, i \in \{0, 1, \ldots n\}$
  2. $s(i) = 10000, i \in \{0, 1, \ldots n\}$
  3. $\lambda(i) = 10000, i \in \{0, 1, \ldots m\}$

Tento vektor sme chceli zvoliť pokiaľ možno v celku vzdialený od optimálneho. Podobne ako pri vektore 1, rovnice $Ax = b$ a $A^T\lambda + s = c$ budú pravdepodobne nesplnené.

4.3 Získané výsledky a interpretácia

Posledná vec, ktorú treba určite spomenúť je fakt, že pri našej implementácií v MATLABe sme použili minimovú NCP-funkciu a jej príslušnú aproximáciu, obe uvedené v kapitole 2.

Na úvod sme algoritmy spustili na programe A. Jeho rozmer bol 10×5. Pre všetky 4 štartovacie vektory dosiahol algoritmus 2, ktorý je v podstate vylepšením algoritmu 1 na pohľad lepšie výsledky. Avšak algoritmus 2 je zložitejší a teda porovnávať počty iterácií nemusí byť vôbec relavantné. Predikčný krok, ktorý obohacuje algoritmus 2 oproti algoritmu 1, je jednoduchší a teda ak je počet iteracií v pomere (1:2) alebo lepšom, môžeme s určitosťou povedať, že bol algoritmus 2 úspešnejší. Tento fakt však neznamená to, že ak to tak nie je, algoritmus je horší.
Táto situácia v pomere iterácií nastala napríklad pri štartovacom vektore 1 (prvý riadok tabuľky 1).
Pri ďalšom rátaní sme zvolili parametre σ, ρ veľmi nevhodne (riadky 7-8 tabuľky 1). Počet iterácií pre oba algoritmy a štartovacie vektory vyšiel vysoký a identický. Identita sa dá vysvetliť tým, že predikčný krok nebol nikdy akceptovaný (aby bol pri takýchto parametroch akceptovaný, muselo by sa nám podariť zmenšiť vyhladzovací parameter pri každej vnútornej iterácií predikčného kroku aspoň na desatinu (hodnota ρ), čo je málo pravdepodobné). Vysoký počet iterácií zabezpečuje vysoká hodnota parametra σ a takisto hodnota ρ(opäť budeme príjmať v korekčnom kroku buď celý, desatinový, stotinový... krok, čo znamená, že sa budeme približovať k optmálnemu vektoru relatívne pomaly).
Pri lineárnom programe B sa začala vyskytovať situácia, kedy počet iterácií pri niektorých štartovacích vektoroch jednoduchšieho algoritmu (algoritmu 1), vyšiel nižší ako pri algoritme 2, ktorého jedna iterácia je zložitejšia. Dokopy sa to stalo v 20 experimentoch z celkových 42. Princíp, ako si to môžeme vysvetliť, je ukrytý vo fungovaní predikčného kroku. Ako si možno všimnúť, predikčný krok neoptimalizuje dĺžku kroku

Tabuľka 1: Výsledky experimentu

Problém	m	n	ρ	σ	ϵ	štart	Algoritmus 1	Algoritmus 2
A	5	10	0.9	0.0001	0.00001	1	11	5
A	5	10	0.9	0.0001	0.00001	2	10	6
A	5	10	0.9	0.0001	0.00001	3	7	5
A	5	10	0.9	0.0001	0.00001	4	20	17
A	5	10	0.9	0.0001	0.00000001	2	11	8
A	5	10	0.9	0.0001	0.00000001	3	9	7
A	5	10	0.1	0.9	0.0001	2	120	120
A	5	10	0.1	0.9	0.0001	3	119	119
B	10	19	0.9	0.0001	0.0001	1	9	8
B	10	19	0.9	0.0001	0.0001	2	9	11
B	10	19	0.9	0.0001	0.0001	3	9	10
B	10	19	0.9	0.0001	0.0001	4	17	16
B	10	19	0.99	0.00005	0.0001	2	9	9
B	10	19	0.99	0.00005	0.0001	3	8	9
C	25	50	0.9	0.0001	0.0001	1	15	9
C	25	50	0.9	0.0001	0.0001	2	11	16
C	25	50	0.9	0.0001	0.0001	3	15	10
C	25	50	0.9	0.0001	0.0001	4	21	16
D	48	99	0.9	0.0001	0.0001	1	20	11
D	48	99	0.9	0.0001	0.0001	2	12	17
D	48	99	0.9	0.0001	0.0001	3	16	14
D	48	99	0.9	0.0001	0.0001	4	31	21

a preto sa často krát môže stať, že príjmeme krok, ktorý sa síce priblíži k optimálnemu vektoru, ale keby sme ho neprijali, vďaka optimalizácii dĺžky kroku v korekčnom kroku by sme optimálne riešenie našli rýchlejšie.

Čo sa týka štartovacích vektorov, ukázalo sa, že štartovací vektor 4 je jednoznačne najhorší, čím sa naplnili naše očakávania (pri jeho voľbe sme sa snažili, aby bol čo

Tabuľka 2: Výsledky experimentu, časť druhá

Problém	m	n	ρ	σ	ϵ	štart	Algoritmus 1	Algoritmus 2
E	66	130	0.9	0.0001	0.0001	1	18	18
E	66	130	0.9	0.0001	0.0001	2	11	32
E	66	130	0.9	0.0001	0.0001	3	16	39
E	66	130	0.9	0.0001	0.0001	4	38	36
F	130	250	0.9	0.0001	0.0001	1	32	39
F	130	250	0.9	0.0001	0.0001	2	25	45
F	130	250	0.9	0.0001	0.0001	3	17	17
F	130	250	0.9	0.0001	0.0001	4	48	79
G	300	600	0.9	0.0001	0.0001	1	32	39
G	300	600	0.9	0.0001	0.0001	2	44	191
G	300	600	0.9	0.0001	0.0001	3	59	265
G	300	600	0.9	0.0001	0.0001	4	150	267
H	300	1000	0.9	0.0001	0.0001	1	38	17
H	300	1000	0.9	0.0001	0.0001	2	22	22
H	300	1000	0.9	0.0001	0.0001	3	28	23
H	300	1000	0.9	0.0001	0.0001	4	294	289
I	519	1154	0.9	0.0001	0.0001	2	33	33
I	519	1154	0.9	0.0001	0.0001	3	26	27
J	700	1500	0.9	0.0001	0.0001	2	24	35
J	700	1500	0.9	0.0001	0.0001	3	24	62
K	1000	2100	0.9	0.0001	0.0001	2	33	54
K	1000	2100	0.9	0.0001	0.0001	3	26	28

najhorší). Veľmi dobré výsledky dosiahli štartovacie vektory 2 a 3 a neiekedy aj vektor 1 (dôvody sme uviedli pri vysvetľovaní jeho voľby). Určiť celkovo najlepší štartovací vektor na základe nášho portfólia je veľmi náročné. Pre nižšie rozmery sme totiž dosiahli väčší úspech s vektorom 2 a pre vyššie rozmery s vektorom 3, avšak rozdiel v

iteráciách pri vyšších rozmeroch nebol veľmi signifikantný (konkrétne nenulový nastal len pri programoch K a I a bol rovný siedmym iteráciám). Teda by sme zvolili vektor 3.

Keď si pozrieme numerické výsledky v článku ([1]), na prvý pohľad uvidíme oveľa lepšie výsledky, nízke počty iterácií pre rádovo aj tri krát vyššie rozmery programov v porovnaní s našimi. Pokúsime sa teda zdôvodniť aj tento fakt v niekoľkých bodoch poukazujúcich na možné príčiny.

1. Použitie rozličnej NCP-funkcie. V článku ([1]) použili Penalizovanú-Fisher-Burmeisterovu funkciu a my sme použili minimovú NCP-funkciu.

2. Použitie rozličného terminačného kritéria, kým my sme ukončili iterovanie až keď norma vektora $\Phi_\tau(w_i)$ v i-tej ierácii bola menšia ako ϵ, v článku ([1]) ukončili iterovanie, keď $\tau_i < 10^{-4}$ (v i-tej iterácií).
   τ_0 sme volili rovnakým spôsobom, a to následovne: $\tau_0 = min\{10000, \|\Phi_\tau(w_0)\|\}$.

3. V článku ([1]) nepoužili presne tento algoritmus, ale jeho modifikáciu na trojkrokový. Prvý krok bol predikčný krok s tým, že ak celý vypočítaný krok teda Δw nevyhovoval príjmaciemu kritériu, brali aj polovičný, štvrtinový atď. Druhý krok bol predikčný krok ale s nahradením pravej strany v (16) za $\Phi_\tau(w)$, a tretím krokom bol korekčný krok tak ako ho poznáme.

4.4 Porovnanie s MATLABom

Čo môže byť ešte zaujímavé, je porovnanie hodnôt účelových funkcií, teda hodnôt $c^T x$, dosiahnutých pomocou jednotlivých algoritmov a pomocou MATLABovskej funkcie *linprog*. O tejto problematike nám hovorí nasledujúca tabuľka. V prvom stĺpci sú lineárne programy, na ktorých sme realizovali optimalizáciu, v druhom stĺpci je hodnota vyhladzovacieho parametra τ, dosiahnutá našim algoritmom 1. V treťom stĺpci je hodnota $c^T x$ dosiahnutá algoritmomom 1. Štvrtý a piaty stĺpec sú podobne ako druhý a tretí, len hovoria o algoritme 2, nie 1. V šiestom stĺpci je absolútna hodnota rozdielu medzi hodnotami výrazou $c^T x$, dosiahnutými algoritmomom 1 a MATLABovskou funkciou *linprog*. Siedmy je ekvivalentný ku šiestemu, len s použitím algoritmu 2. Všetky tieto hodnoty boli vypočítané s použitím nasledovných nastavení:

- $\rho = 0.9$
- $\sigma = 0.0001$
- Štartovací vektor: vektor 2.
- presnosť $\epsilon = 0.0001$

Tabuľka 3: Výsledky experimentu, porovnanie s MATLABom

Problém	τ-1 ($*10^{-4}$)	$c^T x$-1	τ-2 ($*10^{-3}$)	$c^T x$-2	*matlab*-1 ($*10^{-5}$)	*matlab*-2 ($*10^{-3}$)
A	0.7734	1.3632	0.0272	1.3632	0.2991	0.0004
B	0.5869	25.1555	0.4775	25.1557	0.3101	0.2052
C	0.3545	7.9141	0.0712	7.9141	0.3127	0.0126
D	0.1636	6.7800	0.0240	6.7800	0.1357	0.0029
E	0.2046	39.8409	0.0165	39.8409	0.2679	0.0017
F	0.2179	21.9234	0.0082	21.9234	0.5700	0.0008
G	0.0118	246.0581	0.0087	246.0581	0.0046	0.0023
H	0.0096	8.7898	0.0082	8.7899	0.0064	0.0045
I	0.0770	26.1467	0.0013	26.1467	0.3741	0.0001
J	0.0645	42.9619	0.0052	42.9619	0.3328	0.0022
K	0.0156	42.2006	0.0005	42.2006	0.0268	0.0000

Naše algoritmy bežali s presnosťou $\epsilon = 0.0001$. Preto sme aj dané ukazovatele uvádzali iba na štyri desatinné miesta. Ako vidíme, vyhladzovací parameter τ sa pohyboval v rozmedzí 10^{-4} až 10^{-6}. Jedine pri lineárnom programe B sme iterovanie algoritmu 2 ukončili už pri o niečo vyššej hodnote τ, čo sa potom aj prejavilo na najvyššej odchýlke od MATLABovskej funkcie. Keď porovnáme hodnoty $c^T x$ oboch algoritmov, sú takmer identické. Odchýlky výsledkov našich algoritmov od MATLABovskej funkcie sú opäť až na spomínaný lineárny program B algoritmu 2 nižšie ako presnosť, s ktorou sme počítali. Teda po tejto stránke fungujú naše algoritmy veľmi uspokojivo.

4.5 Vyvolanie implementovaných funkcií

V prílohe A uvádzame link, kde sa nachádza MATLABovský materiál ktorý sme použili. Pre záujemcov uvádzame krátky návod, ktorý popisuje správne volanie implementovaných funkcií.

1. Súbor *novinka.mat* obsahuje všetky matice a vektory, ktoré sme v tejto kapitole používali (teda matice $A, B, C, D, E, F, G, H, I, J, K$ a k ním príslušné vektory b, c, pričom vektory AA, AAA sú príslušné k matici A, BB, BBB k matici B atď.).

2. Algoritmus 1 reprezentuje funkcia:
   $function[x, iter, teta] = Hlavnyprogram2(A, b, c, epsilon)$, kde x je optimálne riešenie lineárneho programu, *iter* je počet iterácií, a *teta* je hodnota parametra τ_k v poslednej iterácii. Vstupné parametre sú príslušná matica, vektory lineárneho programu a požadovaná presnosť.

3. Pre algoritmus 2 máme funkciu:
   $function[x, iter, iter2, teta] = Hlavnyprogram33(A, b, c, epsilon)$, kde majú všetky vstupné aj výstupné parametre rovnaký význam, až na to že hodnota $iter2$ symbolizuje počet úspešných iterácií v predikčnom kroku a hodnota *iter* počet celkových iterácií.

4. Pre zmenu štartovacích vektorov máme jednotlivé funkcie *Generbakalarka* a ostatné parametre meníme priamo v hlavných funkciách algoritmov.

Záver

Cieľom našej práce bolo naštudovať teóriu NCP-funkcií a aplikovať tieto poznatky v praxi, implementáciou vhodnej metódy v MATLABe. Čo sa týka teoretickej časti, dokázali sme dať dokopy slušný balík teoretických poznatkov o NCP-funkciách, ich príslušných vlastnostiach a príslušných vyhladzovacích metódach. Vďaka tomu, že sme sa snažili vysvetlovať a dokazovať aj vety, ktoré v mnohých článkoch boli iba okrajovo spomenuté alebo ponechané bez dôkazu, môže táto práca poslúžiť ako kvalitný študijný materiál (najmä pre človeka, ktorý sa zaujíma o danú problematiku a nemá o nej žiadne vedomosti).

Čo sa týka numerických výsledkov a implementácie algoritmov, dosiahnuté výsledky boli o niečo horšie, ako sme očakávali. Či už v porovnaní s článkom ([1]), ktorý bol zdrojom inšpirácie, alebo jeho vylepšením, článkom ([2]). Avšak túto viditeľnú rozdielnosť numerických výsledkov sme dokázali racionálne vysvetliť v kapitole 4. Zároveň by sme chceli čitateľovu pozornosť upriamiť na priestor pri experimentovaní s implementovanými algoritmami. Pri našej implementácií sme použili iba najjednoduchšiu NCP-funkciu, vďaka čomu vzniká priestor pre experimentovanie a vyskúšanie niektorých ďalších NCP-funkcií z kapitoly 2 alebo nejakých iných. Ďalej je tu mnoho nastaviteľných parametrov a možných drobných úprav algoritmov, ktoré by mohli výsledky teoreticky vylepšiť.

Prínos našej práce je teda hlavne teoretický. Avšak aj po numerickej stránke sa nám podarilo ukázať, že uvedené metódy fungujú a aj keď zatiaľ nie až tak efektívne ako napríklad metódy vnútorného bodu, je tu stále priestor na vylepšovanie a napredovanie.

Zoznam použitej literatúry

[1] Engelke S., Kanzow Ch.: *PREDICTOR-CORRECTOR SMOOTHING METHODS FOR LINEAR PROGRAMS*, University of Hamburg, 2000

[2] Engelke S., Kanzow Ch.: *PREDICTOR-CORRECTOR SMOOTHING METHODS FOR LINEAR PROGRAMS WITH A MORE FLEXIBLE UPDATE OF THE SMOOTHING PARAMETER*, University of Hamburg , 2001

[3] Plesník J., Dupačová J., Vlach M.: *Lineárne programovanie*. Alfa, Bratislava, 1990

[4] Kanzow Ch., Yamashita N., Fukushima M.: *New NCP-Functions and Their Properties*, University of Hamburg, 1997 Dostupné na internete: http://citeseer.ist.psu.edu/viewdoc/summary?doi=10.1.1.44.5870

[5] Sun D., Qi L.: *Computational Optimization and Applications*, Kluwer Academic Publishers, Boston 1999,201-220

[6] Sun D., Qi L.: *Computational Optimization and Applications*, Kluwer Academic Publishers, Boston 1999

[7] Gui-Hua Liny, Xiaojun Chenz and Masao Fukushima.: *New Restricted NCP Functions and Their Applications to Stochastic NCP and Stochastic MPEC*, Kyoto university, September 8, 2006

[8] J. Chena, S. Panb, T.-C. Lin: *A smoothing Newton method based on the generalized Fischer Burmeister function for MCPs* Dostupné na internete: http://math.ntnu.edu.tw/∼jschen/Papers/Smoothing_MCP%28NA%29.pdf

[9] S.J. Wright: *Primal-Dual Interior-Point Methods.* SIAM, Philadelphia, PA, 1997

[10] C. Geiger, Ch. Kanzow: *Theorie und Numerik restringierter Optimierungsaufgaben*, Springer, 2002

Príloha A

Prístup k Matlabovským súborom

$http://uloz.to/xzDmHBE/bakalarka-rizman-rar$

Heslo: bakalarka1

Alebo alternatívne(avšak bez súboru novinka.mat):

$http://www.mbsoftworks.sk/dropbox/bak2.rar$