UNIVERZITA KOMENSKÉHO V BRATISLAVE
FAKULTA MATEMATIKY, FYZIKY A INFORMATIKY

Vlastnosti spektrahedrálnych množín a ich aplikácie v nelineárnej optimalizácii

DIPLOMOVÁ PRÁCA

2016 Bc. Andrej Iring

UNIVERZITA KOMENSKÉHO V BRATISLAVE
FAKULTA MATEMATIKY, FYZIKY A INFORMATIKY

Vlastnosti spektrahedrálnych množín a ich aplikácie v nelineárnej optimalizácii

DIPLOMOVÁ PRÁCA

Študijný program:	Ekonomická a finančná matematika a modelovanie
Študijný odbor:	9.1.9. Aplikovaná matematika
Školiace pracovisko:	Katedra aplikovanej matematiky a štatistiky
Vedúci práce:	prof. RNDr. Daniel Ševčovič, CSc

Bratislava 2016 **Bc. Andrej Iring**

60177325

Univerzita Komenského v Bratislave
Fakulta matematiky, fyziky a informatiky

ZADANIE ZÁVEREČNEJ PRÁCE

Meno a priezvisko študenta: Bc. Andrej Iring
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
Sekundárny jazyk: anglický

Názov: Vlastnosti spektrahedrálnych množín a ich aplikácie v nelineárnej optimalizácii.
Spectrahedral sets and their applications in nonlinear optimization.

Cieľ: Práca sa zameria na analýzu spektrahedrálnych množín. Tieto množiny su prirodzeným zovšeobecnením polyhedrálnych množín opísaných systémami lineárnych nerovností. Spektrahedrálne množiny sú charakterizované pomocou nerovností vyjadrených pomocou kladnej semidefinitnosti matíc. Cieľom práce bude preskúmať možnosti aproximácie spektrahedrálnych množín pomocou polyhedrálnych množín s možnosťou využitia v oblasti semidefinitného programovania a nelineárnej optimalizácie. V praktickej časti práce sa zameriame na aplikácie pochádzajúce z problematiky optimalizácie finančného portfólia.

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

Dátum zadania: 10.02.2015

Dátum schválenia: 11.02.2015 prof. RNDr. Daniel Ševčovič, CSc.
garant študijného programu

.....
študent

.....
vedúci práce

Poďakovanie Rád by som touto cestou poďakoval svojmu vedúcemu diplomovej práce prof. RNDr. Danielovi Ševčovičovi, CSc. za rady a podnetné pripomienky, pri písaní tejto práce.

Abstrakt v štátnom jazyku

IRING, Andrej: Vlastnosti spektrahedrálnych množín a ich aplikácie v nelineárnej optimalizácii [Diplomová práca], Univerzita Komenského v Bratislave, Fakulta matematiky, fyziky a informatiky, Katedra aplikovanej matematiky a štatistiky; školiteľ: prof. RNDr. Daniel Ševčovič, CSc, Bratislava, 2016, 58 s.

V našej práci sa venujeme spôsobu ako aproximovať spektrahedrálne množiny, t.j. množiny zadané pomocou maticových nerovností, pomocou polyhedrálnych množín. Jedným z cieľov našej práce je skonštruovať algoritmus, ktorý bude riešiť štandardnú úlohu semidefinitného programovania, pomocou série úloh lineárneho programovania, ďalším cieľom je navrhnúť algoritmus, ktorý by riešil úlohy so zložitejšou účelovou funkciou, a jeho implementácia.

Kľúčové slová: Spektrahedrálne množiny, Lineárne programovanie, Semidefinitné programovanie, Kladná semidefinitnosť matíc

Abstract

IRING, Andrej: Spectrahedral sets and their applications in nonlinear optimization [Master's thesis], Comenius University in Bratislava, Faculty of Mathematics, Physics and Informatics, Department of Applied Mathematics and Statistics; Supervisor: prof. RNDr. Daniel Ševčovič, CSc., Bratislava, 2016, 58 s.

In our work we propose a method to approximate spektrahedral set, i.e. a set which is specified by matrix inequalities, using just polyhedral sets. Our aim is to construct an algorithm that will solve standart task of semidefinite programming, using series of linear programming problems. Another objective of our work is to design an algorithm that would address optimization problem with a more complex objective function.

Keywords: Spectrahedron, Linear programming, Quadratic programming, Semidefinite programming, Positive-semidefinite matrix

Obsah

Zoznam obrázkov

Zoznam tabuliek

Zoznam použitých symbolov

$\|\boldsymbol{u}\|$	euklidovská norma vektora $\boldsymbol{u}$, $\|\boldsymbol{u}\| := \sum_i u_i^2$
$\boldsymbol{r}$	$\boldsymbol{r} := (x, y, z)^{\mathrm{T}}$, resp. $\boldsymbol{r} := (x_1, x_2, x_3)^{\mathrm{T}}$
$int(C)$	vnútro množiny C
∂C	hranica množiny C
$tr(A), \quad trace(A)$	stopa matice A
$Y \bullet H$	skalárny súčin nad maticami t.j. $tr(YH)$
k	počet premenných v úlohe SDP
n	rozmer symetrických matíc v úlohe SDP
$\mathbb{R}^n$	priestor vektorov dimenzie n
$\mathbb{R}^n_+$	množina realných vektorov dimenzie n s nezápornými členmi
$\mathbb{R}^n_{++}$	množina realných vektorov dimenzie n s kladnými členmi
S^n	priestor $n \times n$ symetrických matíc
S^n_+	priestor $n \times n$ symetrických kladne semidefinitných matíc
S^n_{++}	priestor $n \times n$ symetrických kladne definitných matíc
$\lambda_i(A)$	i-ta vlastná hodnota matice A
$\lambda_{min}(A)$	minimálna vlastná hodnota matice A
Q^{-T}	označenie pre inverziu a transpozíciu matice Q
I	identická matica
e_i	i-ty stĺpec matice I, resp. vektory štandardnej bázy $\mathbb{R}^n$
$Diag(a_1, a_2, \ldots, a_n)$	diagonálna matica s $a_1, \ldots, a_n$ na diagonále
$\mathcal{S}(A)$	stĺpcový priestor matice A,$t.j.\{Ax : x \in \mathbb{R}^n\}$
$\mathcal{N}(A)$	nulový priestor matice A, t.j. $\{x \in \mathbb{R}^n : Ax = 0\}$
$s^{(i)}_j$	j-tý prvok i-teho vektoru s

Úvod

Matematická optimalizácia a najmä semidefinitné programovanie v dnešnej dobe nadobúda stále viacej na významnosti. Tento druh matematickej optimalizácie je vo veľkom používaný v technologických odvetviach, štatistických oboroch a ekonómii. Hoci sú vytvorené značne efektívne metódy na riešenie úloh semidefinitného programovania, stále sú veľmi limitované zložitosťou úloh, a preto je potrebný výskum týchto úloh, aby sme boli schopní skonštruovať algoritmy na riešenie aj značne veľkých a zložitých úloh. V našej práci sa budeme venovať aproximácií množín prípustných riešení, v štandardnej úlohe semidefinitného programovania, pomocou jednoduchších množín, konkrétne pomocou polyhedrálnych množín. Touto problematikou sa už zaoberala dizertačná práca K. Sivaramakrishnana [11], na ktorú nadväzovali resp. odkazovali práce A. Galenka [15],[16] a práca J. Younga [17]. My budeme čerpať najmä z práce [11] a pokúsime sa o niekoľko modifikácií algoritmu, ktorý bol v tejto práci aj prezentovaný. Ďalej sa budeme zaoberať možnosťou využiť možnosti aproximácie prípustných množín v klasickej úlohe semidefinitného programovania na úlohy, s kvadratickými účelovými funkciami. Nakoniec uvedieme určité porovnanie nášho algoritmu s dostupným softwarom, ako aj niektoré možné aplikácie vo finančnej matematike.

1 Základné pojmy

V prvej časti úvodnej kapitoly uvedieme základné tvrdenia z lineárnej algebry a matematického programovania, ktoré budeme využívať v nasledujúcich kapitolách pri odvodzovaní jednotlivých metód. Všetky metódy, ktoré sú uvedené v nasledujúcich kapitolách, budú riešiť úlohu matematického programovania so spektrahedrálnou množinou prípustných riešení ako postupnosť úloh s rovnakou účelovou funkciou a polyhedrálnou množinou prípustných riešení.

1.1 Vety a definície z lineárnej algebry

Veľmi dôležitým pojmom v našej práci bude symetrická matica a priestor symetrických matíc. Symetrická matica je matica spĺňajúca $A = A^T$, ďalej podpriestor symetrických $n \times n$ matíc budeme označovať $S^n \subset \mathcal{M}^n$. Priestor S^n je izomorfný s vektorovým priestorom $\mathbb{R}^{\frac{n(n+1)}{2}}$. Toto možno vidieť, ak by sme zapísali všetky prvky dolnej trojuholníkovej časti matice do jedného dlhého vektoru. Nech $M \in S^n$ potom definujeme zobrazenie $vect(M) : S^n \to \mathbb{R}^{\frac{n(n+1)}{2}}$ ako zobrazenie, ktoré pokladá prvky matice M do vektora pozdĺž diagonály t.j.

$$vect(M) = \begin{pmatrix} M_{1,1} \\ M_{2,2} \\ \vdots \\ M_{n,n} \\ M_{2,1} \\ M_{3,2} \\ \vdots \\ M_{n,1} \end{pmatrix}$$

Zobrazenie $vect(M)$ priradí matici M jej prislúchajúci vektor z priestoru $\mathbb{R}^{\frac{n(n+1)}{2}}$. Zobrazenie $vect^{-1}(v)$, je inverzné zobrazenie, ktoré zas vektoru priradí prislúchajúcu symetrickú maticu. Ďalej je dôležité uviesť definíciu pojmov kladne definitná matica a kladne semidefinitná matica, kedže s nimi budeme značne veľa pracovať. Nasledujúce dve definície sú zjednodušením definícií z učebnice P. Zlatoša [9].

Definícia 1.1. *Matica A sa nazýva kladne definitná, ak $\forall x \in \mathbb{R}^n, x \neq 0$ platí $x^T Ax > 0$.*

Definícia 1.2. *Matica A sa nazýva kladne semidefinitná, ak $\forall x \in \mathbb{R}^n$ platí $x^T Ax \geq 0$.*

Podmnožinu symetrických matíc, ktoré sú kladne semidefinitných matíc budeme značiť S_+^n a jej vnútro, teda množinu kladne definitných symetrických matíc, budeme značiť S_{++}^n . Pri výpočtoch by bolo ťažkopádne zisťovať, či matice sú kladne definitné. Uvedieme preto postačujúce podmienky pre kladnú definitnosť a kladnú semidefinitnosť.

Veta 1.3 ([3, str.263] Sylvestrovo kritérium)**.** *Nech $A \in \mathbb{R}^{n\times n}$ je symetrická matica. Potom:*

(a) matica A je kladne definitná práve vtedy, keď všetky jej hlavné rohové minory (determinanty hlavných rohových podmatíc) sú kladné.

(b) matica A je kladne semidefinitná práve vtedy, keď všetky jej hlavné minory (determinanty hlavných podmatíc) sú nezáporné.

Jednou z veľmi dôležitých vlastností symetrických matíc je, že sú vždy diagonalizovateľné a navyše ich vlastné vektory sa dajú vybrať tak, aby tvorili ortogonálnu bázu priestoru. Toto tvrdenie uvedieme ako lemu. Nasledujúca lemma je prebratá z knihy M.Hamalu a M. Trnovskej Nelineárne programovanie [3].

Lema 1.4. *Pre každú štvorcovú symetrickú $n\times n$ maticu A existuje ortogonálna matica Q (t.j. $QQ^T = Q^TQ = I$) a diagonálna matica Λ s vlastnými číslami matice A na diagonále (ktoré sú reálne) tak, že*

$$A = Q\Lambda Q^T,$$

pričom stĺpce matice Q sú lineárne nezávislé ortogonálne vlastné vektory matice A.

Dôsledok 1.5.

- *Matica A je kladne semidefinitná ak má všetky vlastné hodnoty nezáporné.*
- *Špeciálne, ak matica A je kladne semidefinitná, potom existuje matica V taká, že $A = VV^T$.*

Veľmi dôležitá vlastnosť symetrických matíc je spektrálna dekompozícia.

$$A = \sum_{i=1}^{r} \lambda_i q_i q_i^T, \tag{1}$$

kde λ_i sú nenulové vlastné hodnoty a q_i sú prislúchajúce vlastné vektory. Ďalším dôležitým pojmom je stopa matice $A \in S^n$, ktorá je definovaná ako

$$tr(A) = \sum_{i=1}^{n} A_{ii},$$

a skalárny súčin pre matice $A, B \in S^n$, ktorý je definovaný

$$A \bullet B = tr(AB).$$

Ďalej definujme Frobeniovu normu pre symetrickú maticu $A \in S^n$, ako $\|A\|_F = \sqrt{A \bullet A}$. Jedným z ďalších užitočných kritérií kedy je matica kladne semidefinitná je

Veta 1.6. *Matica A je kladne semidefinitná práve vtedy, keď*

$$A \bullet B \geq 0, \quad \forall B \in S^n_+$$

Dôkaz tejto vety možno nájsť, už v úvode spomenutej dizertačnej práci [11, str. 10].

Veta 1.7. *Nech matice $A, B \in S^n_+$. Potom $tr(AB) = 0$ práve vtedy, keď $AB = 0$.*

Dôkaz. Ak $AB = 0$, tak potom je vidno, že $tr(AB) = 0$. Nech teda $tr(AB) = 0$. Keďže matice A, B sú kladne semidefinitné tak existujú matice V a U také, že $A = VV^T$, $B = UU^T$. Potom

$$\begin{aligned} tr(AB) &= tr(VV^T UU^T) \\ &= tr(U^T VV^T U) \\ &= tr\left(U^T V (U^T V)^T\right) = 0. \end{aligned}$$

V poslednej rovnosti máme na ľavej strane Frobeniovu normu matice $U^T V$ a z vlastností normy vyplýva, že $U^T V = 0$, a teda $AB = 0$. □

Toto tvrdenie bude neskôr zohrávať dôležitú úlohu pri podmienke komplementarity pre úlohu semidefinitného programovania. Ešte by sme radi uviedli dve tvrdenia, ktoré budeme neskôr využívať pri zmene ohraničení v príkladoch v poslednej kapitole. Prvé tvrdenie je známa veta o Schurovom komplemente.

Veta 1.8. *Nech $U = \left(\begin{smallmatrix} A & B \\ B^T & C \end{smallmatrix}\right)$, pričom $A \in S^n, C \in S^m, B$ $n \times m$ matica a $A \succ 0$. Potom*

$$U \succeq 0 \Leftrightarrow C - B^T A^{-1} B \succeq 0.$$

Dôkaz tejto vety možno nájsť v učebnici Matrix Analysis [19] od autorov Roger A. Horn a Charles R. Johnson.

Táto veta nesie meno po známom matematikovi Issaiovi Schurovi, ktorý sa narodil v roku 1875. Pracoval v mnohých odvetviach matematiky, napríklad v teórií grúp, kombinatorike, teórií čísel a ďalších.

Obr. 1: Issai Schur[1].

1.2 Semidefinitné programovanie

Semidefinitné programovanie je špeciálnou podmnožinou úloh konvexného programovania. Teória konvexného programovania sa začala pomaly rozvíjať od 20. rokov minulého storočia a záujem o túto problematiku stále nepoľavil. Najznámejšími priekopníkmi v oblasti teórie konvexnej optimalizácie resp. návrhov algoritmov na riešenie úloh boli John von Neumann, Narendra Karmarkar, Harold W. Kuhn, Albert W. Tucker, Morton L. Slater, Stephen Boyd a mnoho ďalších. Na úvod uvedieme tvar úlohy semidefinitného programovania, s ktorým budeme v našej práci naďalej pracovať.

$$\begin{aligned} \min_{x} \quad & c^T x \\ \text{s.t.} \quad & \mathcal{A}(x) \equiv H_0 + x_1 H_1 + \cdots + x_k H_k \succeq 0. \end{aligned} \tag{SDP}$$

Vidno, že úloha (SDP) nápadne pripomína úlohu lineárneho programovania až na ohraničenia $\mathcal{A}(x) \succeq 0$, ktoré sú v tvare lineárnej maticovej nerovnosti (LMI). Množina prípustných riešení tejto úlohy sa nazýva spektrahedrón.

Definícia 1.9. *Nech $H_0, H_1, \ldots, H_k \in S^n$. Potom spektrahedrálna množina je definovaná ako*

$$Sp = \{x \in \mathbb{R}^k | \mathcal{A}(x) \equiv H_0 + x_1 H_1 + \cdots + x_k H_k \succeq 0\}.$$

Množina Sp je množinou prípustných riešení úlohy (SDP).

Spektrahedrónom môžu byť rôzne množiny. Ako príklad najjednoduchšej spektra-

[1] zdroj https://en.wikipedia.org/wiki/File:Schur.jpg

hedrálnej množiny môžme uviesť kruh

$$\begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} + x \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix} + y \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix} \succeq 0 \Leftrightarrow$$

$$(1-x)(1+x) - y^2 \geq 0 \Leftrightarrow$$

$$x^2 + y^2 \leq 1.$$

Ďalšími jednoduchými spektrahedrónmi sú polyhedróny, vnútro paraboloidu a mnoho ďalších. Jednou zo základných vlastností spektrahedrónov je konvexnosť.

Veta 1.10. *Spektrahedrálne množiny sú konvexné a uzavreté.*

Dôkaz. Uvedieme dôkaz iba konvexnosti, uzavretosť vyplýva zo spojitosti charakteristického polynómu matice.
Nech $x, y \in Sp$ sú ľubovoľné. Kedže oba tieto body patria do množiny Sp tak platí:

$$\forall z \in \mathbb{R}^n : z^T \mathcal{A}(x) z \geq 0,$$

$$\forall z \in \mathbb{R}^n : z^T \mathcal{A}(y) z \geq 0.$$

Ukážeme, že $\alpha x + (1-\alpha)y$ tiež patrí do množiny Sp pre všetky $\alpha \in (0,1)$. Treba ukázať, že $\mathcal{A}(\alpha x + (1-\alpha)y) \succeq 0$. Keďže zobrazenie $\mathcal{A} : \mathbb{R}^k \to S^n$ je lineárne v premennej x tak platí $\mathcal{A}(\alpha x + (1-\alpha)y) = \alpha\mathcal{A}(x) + (1-\alpha)\mathcal{A}(y)$. Prenásobením poslednej rovnosti ľubovoľným vektorom z sprava a z^T zľava dostávame

$$z^T \mathcal{A}(\alpha x + (1-\alpha)y) z = \alpha z^T \mathcal{A}(x) z + (1-\alpha) z^T \mathcal{A}(y) z. \qquad \text{(I)}$$

Vidno, že výraz (I) je vždy nezáporný pre ľubovolné $z \in \mathbb{R}^n$ a teda matica $\mathcal{A}(\alpha x + (1-\alpha)y)$ je kladne semidefinitná, z čoho vyplýva, že bod $\alpha x + (1-\alpha)y$ patrí do množiny Sp. □

Ďalším vlastnostiam spektrahedrónov sa budeme venovať v nasledujúcej kapitole. V teórií konvexného programovania, a teda aj v semidefinitnom programovaní má nezanedbateľný význam teória duality. Skôr ako pristúpime ku konštrukcií duálnej úlohy pre úlohy (SDP) uvedieme Lagrangeovu funkciu pre úlohu (SDP). Definícia a vlastnosti Lagrangeovej funkcie sú podrobne rozpísané v knihe Nelineárne programovanie od M. Hamala a M. Trnovská[3] ako aj základy z teórie duality pre konvexné programovanie. My uvedieme iba niekoľko tvrdení z tejto publikácie v prípade väčšieho záujmu o túto

problematiku odkážeme čitateľa už na spomenutú knihu [3]. Lagrangeova funkcia k úlohe (SDP) má tvar:

$$L(x, Y) = c^T x - Y \bullet \left(H_0 + \sum_{i=1}^{k} x_i H_i \right)$$
$$Y \succeq 0.$$

Keďže funkcia $L(x, Y)$ je konvexná v premennej x, a teda pre pevne zvolené Y vždy nadobúda svoje minimum, môžme skonštruovať Wolfeovu duálnu úlohu k úlohe (SDP).

$$\begin{aligned} \max_{Y} \quad & -tr(H_0 Y) \\ \text{s.t.} \quad & tr(H_i Y) = c_i, \quad \text{pre} \quad i = 1, \ldots, k, \\ & Y \succeq 0. \end{aligned} \tag{SDD}$$

Ak maximalizujeme záporné hodnoty funkcie, tak maximalizáciu môžme nahradiť minimalizáciou a zbavíme sa mínusového znamienka v účelovej funkcií. Veľmi dôležitou vlastnosťou pri optimalizačných úlohách je dualita. Veta o slabej dualite platí skoro vždy. Avšak pre potreby väčšiny algoritmov je nutné, aby bola splnená aj silná veta o dualite. Táto veta avšak vyžaduje, aby úloha spĺňala tzv. Slaterovu podmienku. Pri tvorbe podmienky pre úlohu semidefinitného programovania vychádzame z jej všeobecného tvaru, ktorý možno nájsť v knihe Convex Optimization od autorov S. Boyda a L. Vandenbergheho [12].

Veta 1.11. *(Slaterova podmienka)*
Úloha (SDP) spĺňa Slaterovu podmienku, ak existuje $\bar{x} \in \mathbb{R}^k$ také, že

$$\mathcal{A}(\bar{x}) \succ 0$$

t.j. množina prípustných riešení Sp má aspoň jeden vnútorný bod.

Veľmi dôležitou vetou je slabá veta o dualite.

Veta 1.12. *(Slabá veta o dualite)*
Ak je x prípustné v (SDP) a Y je prípustné v (SDD), potom

$$c^T x - (-H_0 \bullet Y) = c^T x + H_0 \bullet Y \geq 0$$

Dôkaz. Kedže Y je prípustné, tak platí $c_i = Y \bullet H_i$, pre $i = 1, 2, \ldots, k$, a teda platí

$$\sum_{i=1}^{k} Y \bullet H_i x_i + H_0 \bullet Y = Y \bullet \left(H_0 + \sum_{i=1}^{k} H_i x_i \right)$$
$$= Y \bullet \mathcal{A}(x).$$

Kedže Y aj x sú prípustné tak podľa Vety 1.6 platí $Y \bullet \mathcal{A}(x) \geq 0$. □

Rozdiel hodnôt účelových funkcií sa zvykne označovať ako primárno-duálna medzera. Tento rozdiel je veľmi dôležitý pri tvorbe algoritmov, pretože sa využíva pri rozhodovaní o ukončení algoritmu z dôvodu blízkosti k optimálnemu riešeniu. Avšak na to, aby sme mohli primárno-duálnu medzeru použiť ako zastavovacie kritérium, je potrebné, aby bola splnená silná veta o dualite, ktorej znenie sme prebrali z článku M. Trnovskej Strong Duality Conditions in Semidefinite Programming [22].

Veta 1.13. *(Silná veta o dualite)*
Majme S-regulárnu úlohu semidefinitného programovania (SDP). Nech $\hat{x}$ je optimálnym riešením úlohy (SDP). Potom existuje $\hat{Y} \in S_+^n$ tak, že $\hat{Y}$ je optimálnym riešením duálnej úlohy (SDD) a platí:

$$c^T \hat{x} + H_0 \bullet \hat{Y} = 0.$$

Inak povedané, spomedzi všetkých prípustných riešení je iba pre optimálne riešenia úloh (SDP) a (SDD) primárno-duálna medzera rovná nule.

Jedným z najväčších prielomov v nelineárnom programovaní bolo odvodenie nutných podmienok optimality pre štandardnú úlohu nelineárneho programovania. Tieto podmienky boli v roku 1939 uvedené v diplomovej práci matematika William Karusha, avšak neboli publikované. Širšia matematická spoločnosť o nich nevedela. Neskôr v roku 1951 boli tieto podmienky nezávisle od práce Williama Karusha publikované dvojicou matematikov Harold Kuhn a Albert Tucker. Tieto podmienky určujú čo musí kandidát na optimálne riešenie úlohy spĺňať. V prípade konvexného programovania platí, že tieto podmienky sú zároveň aj postačujúcimi podmienkami, teda bod ktorý ich spĺňa je optimálnym riešením úlohy konvexného programovania. Bližšie informácie o Karush-Kuhn-Tuckerovych (K-K-T) podmienkach ako aj dôkaz o ich postačujúcosti v prípade

konvexného programovania môže čitateľ nájsť v publikácií [3]. My uvedieme iba K-K-T podmienky pre úlohu (SDP).

Veta 1.14. *Uvažujme úlohu (SDP), nech dvojica $(\hat{x}, \hat{Y})$ spĺňa Karush-Kuhn-Tuckerove podmienky:*

$$c_i - \hat{Y} \bullet H_i = 0, \quad i = 1, 2, \ldots, k, \tag{2}$$

$$H_0 + \sum_{i=1}^{k} \hat{x}_i H_i \succeq 0, \tag{3}$$

$$\hat{Y} \bullet \left(H_0 + \sum_{i=1}^{k} \hat{x}_i H_i \right) = \hat{Y} \bullet \mathcal{A}(\hat{x}) = 0, \tag{4}$$

$$\hat{Y} \succeq 0. \tag{5}$$

Potom $\hat{x}$ je optimálnym riešením úlohy (SDP).

Z týchto podmienok bude mať pre nás najväčší význam podmienka (4), najmä pri tvorbe zjednodušených podúloh úlohy (SDD). Na záver kapitoly uvedieme potrebný predpoklad a niekoľko tvrdení, ktoré budú potrebné najmä v kapitole (2).

Predpoklad 1. *Nech $Y \bullet H_i = c_i, \quad i = 1, 2, \ldots, k$. Potom $tr(Y) = a$ pre nejakú konštantu $a \geq 0$.*

Nasledujúce tvrdenie hovorí o tom, kedy je predchádzajúci predpoklad splnený.

Veta 1.15. *Nech $\mathcal{Y} = \{Y \succeq 0 : H_i \bullet Y = c_i, i = 1, 2, \ldots, k\}$ je ohraničená množina. Potom existuje regulárna škálovacia matica Q taká, že $tr(W) = a$ pre všetky $W \in \{W \succeq 0 : H_i \bullet (QWQ^T) = c_i, \quad i = 1, \ldots, k\} = \{Q^{-1}YQ^{-T} : Y \in \mathcal{Y}\}$.*

Dôkaz tejto vety možno nájsť v [13, str. 73].

Dôvod, prečo sme zaviedli predpoklad (1) a ukázali predchádzajúcu vetu je nasledujúca veta, ktorú preberáme z [11, str. 88], kde sa nachádza aj jej dôkaz.

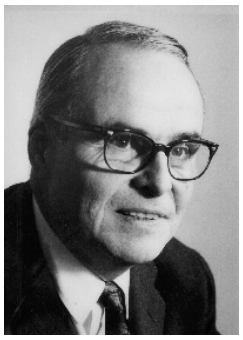
Veta 1.16. *Ak pre všetky prípustné riešenia Y pre (SDD) platí $tr(Y) = a$, potom*

$$\exists \hat{x} \in \mathbb{R}^k, \text{ také, že } \quad \sum_{i=1}^{k} \hat{x}_i H_i = I.$$

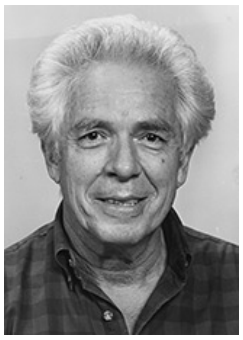
[3] zdroj https://en.wikipedia.org/wiki/File:Albert_W._Tucker.gif

[3] zdroj https://en.wikipedia.org/wiki/File:Harold_W._Kuhn.jpg

Túto vetu budeme využívať pri hľadaní vektorov z, ktoré budú generovať "dobrú" aproximáciu množiny Sp.



Obr. 2: Albert Tucker[2].



Obr. 3: Harold Kuhn[3].

2 Riešenie SPD pomocou LP

V tejto kapitole sa budeme venovať aproximácií množiny Sp pomocou jednoduchších množín a následne uvedieme algoritmus riešenia úlohy (SDP) pomocou série úloh využívajúcich iba lineárne programovanie. Základná myšlienka na aproximáciu vychádza priamo z definície kladnej semidefinitnosti matice (1.2).

2.1 Aproximácia množiny prípustných riešení

Definícia 2.1. *Nech $z \in \mathbb{R}^n$ je ľubovolne pevne zvolený vektor, potom definujeme polpriestor P_z nasledovne*

$$P_z = \{x| \quad z^T\mathcal{A}(x)z = z^TH_0z + x_1z^TH_1z + \cdots + x_kz^TH_kz \geq 0\}. \tag{6}$$

Označme vektor $p_z = (z^TH_1z, z^TH_2z, \ldots, z^TH_kz)^T$ a $h_z = z^TH_0z$. Budeme hovoriť, že vektor $z \in \mathbb{R}^n$ generuje nadrovinu $p_z^Tx + h_z = 0$.

Priamo z definície vidno, že ak $x \in Sp$, potom určite patrí aj do množiny P_z a teda dostávame, že $Sp \subseteq P_z$ pre ľubovolný vektor $z \in \mathbb{R}^n$. Avšak množina P_z vo všeobecnosti nemôže dobre aproximovať množinu Sp, keďže Sp môže mať ˝zaoblený˝ charakter, zatiaľ čo P_z je polyhedrálny polpriestor. Preto budú zaujímavé iba prieniky množín P_z pre rôzne vektory z. Z predchádzajúcich myšlienok je zrejmé, že $Sp \subseteq \bigcap_{i=1}^k P_{z_i}$, pre ľubovolné k. Pre jednoduchosť budeme zjednotenie $\bigcap_{i=1}^k P_{z_i}$ značiť P^k. Naozaj dokonalú aproximáciu množiny Sp, by sme dostali, ak by sme použili všetky vektory $z \in \mathbb{R}^n$. Hoci postačovali by aj iba vektory spĺňajúce $z^Tz = 1$. Avšak takáto aproximácia je v praxi nepoužiteľná. Navyše vzhľadom na to, že máme zreteľ na riešenie úlohy (SDP), tak nám postačuje mať dobrú aproximáciu, len na okolí optimálneho riešenia a teda nepotrebujeme aproximovať celú množinu Sp. Vzhľadom na už spomenutý "zaoblený" charakter spektrahedrálnych množín je zrejmé, že najlepšia aproximácia, ktorú môžme dostať pomocou len jednej množiny P_z je vtedy, keď nadrovina $p_z^Tx + h_z = 0$ je dotyková nadrovina množiny Sp pre aspoň jedno $x \in Sp$. Preto je namieste otázka ako voliť vektory z tak, aby nám generovali dotykové nadroviny $p_z^Tx + h_z = 0$. Pôvodne sme pracovali s myšlienkou hľadania bodov x, ktoré sa nachádzali na hranici množiny Sp a ako vektor generujúci množinu P_z sme používali vlastný vektor z_0 matice $\mathcal{A}(x)$

pre vlastnú hodnotu rovnú nule. Tento spôsob vždy generoval dotykovú nadrovinu v dotykovom bode, keďže bola splnená rovnosť $z_0^T \mathcal{A}(x) z_0 = 0$. Tento spôsob mal, ale plno nedostatkov, ktoré spomíname pri opise nášho algoritmu. Od hľadania bodov na hranici sme mohli upustiť vďaka poznatkom, ktoré sme nadobudli z dizertačnej práce K. K. Sivaramakrishnan [11]. Nasledujúce tvrdenie je prebraté z [11, str. 95]. Uvedieme aj jeho dôkaz, toto tvrdenie hovorí o tom, ako generovať dotykové nadroviny množine Sp.

Veta 2.2. *Ohraničenie* $dd^T \bullet \mathcal{A}(x) = d^T \left(H_0 + \sum_{i=1}^{k} x_i H_i \right) d \geq 0$, *kde* d *je vlastný vektor prislúchajúci najmenšej zápornej vlastnej hodnote matice* $\mathcal{A}(x)$, *je dotyková nadrovina množiny* Sp.

Dôkaz. Nech $\mathcal{A}(\hat{x})$ je naša indefinitná matica t.j. $\hat{x} \notin Sp$. Označme λ veľkosť najzápornejšej vlastnej hodnoty matice $\mathcal{A}(x)$ a d prislúchajúci vlastný vektor. Potom matica $S = \mathcal{A}(\hat{x}) + \lambda I$ je kladne semidefinitná matica. Z predpokladu 1 a Vety1.16 vyplýva, že existuje $\bar{x} \in \mathbb{R}^k$ také, že $\sum_{i=1}^{k} \bar{x}_i H_i = I$. Označme $x = \hat{x} + \lambda \bar{x}$ potom $S = \mathcal{A}(x)$. Keďže S je kladne semidefinitná, tak jej najmenšia vlastná hodnota je nula a prislúchajúci vlastný vektor je d. Dostávame $d^T S d = 0$. Bod x leží na hranici P_d, navyše bod x je prípustným riešením, teda patrí do množiny Sp. □

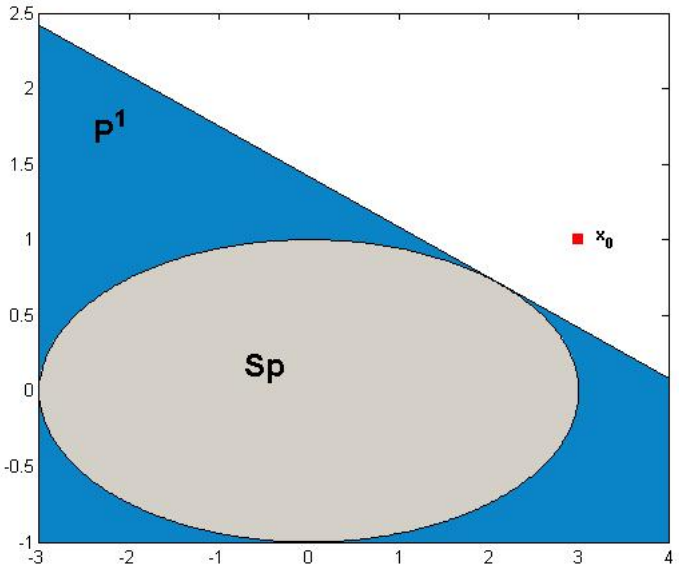
Takže dostávame, že ak je splnený predpoklad 1 a sme v ľubovoľnom bode x, tak najlepšia možnosť na voľbu generátora aproximácie množiny Sp je zobrať vlastný vektor $\mathcal{A}(x)$, ktorý prislúcha najzápornejšej vlastnej hodnote. Ešte pred tým, ako sa začneme venovať popisu tvorby podúloh, uvedieme geometrický význam spomenutej aproximácie.

Príklad 2.3. *Majme množinu* $Sp = \{x : \left(\frac{x_1}{3}\right)^2 + x_2^2 \leq 1.\}$ *Vidno, že sa jedná o elipsu so stredom v bode* $(0,0)^T$ *pričom hlavná os je* 3 *a vedľajšia* 1. *Túto nerovnosť môžme zapísať v tvare maticovej nerovnosti ako*

$$\mathcal{A}(x) = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} + x_1 \begin{pmatrix} \frac{1}{3} & 0 \\ 0 & -\frac{1}{3} \end{pmatrix} + x_2 \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix} \succeq 0.$$

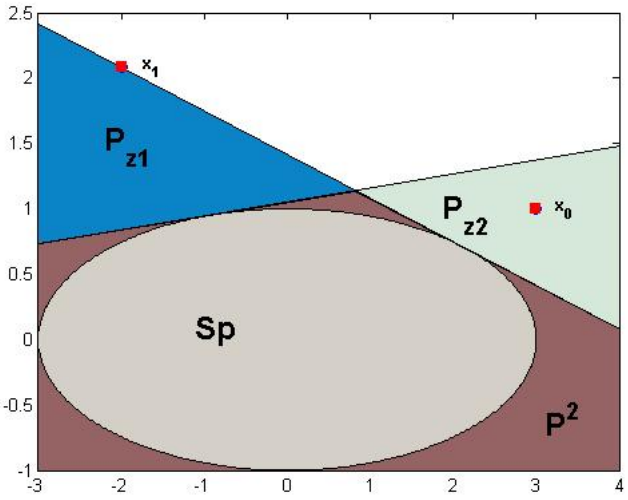
Nech náš štartovací bod je $x^0 = (3,1)^T$. *Potom* $\mathcal{A}(x^0) = \left(\begin{smallmatrix} 2 & 1 \\ 1 & 0 \end{smallmatrix}\right)$. *Najmenšia vlastná hodnota tejto matice je* $\lambda_{min} = 1 - \sqrt{2}$ *a prislúchajúci vlastný vektor je* $z_1 = (\frac{1}{2}, \frac{1}{2(1-\sqrt{2})})^T$,

ktorý generuje množinu $P^1 = \{x \in \mathbb{R}^2 : 2(2-\sqrt{2}) + \frac{2(1-\sqrt{2})}{3}x_1 + 2(1-\sqrt{2})x_2 \geq 0\}$. *Graficky to bude vyzerať nasledovne*



Obr. 4: Príklad aproximácie množiny *Sp* pomocou polyhedrálneho polpriestoru P^1

Vidno, že vektor z_1 *vygeneroval dotykovú nadrovinu. Graficky znázorníme množinu* P^2. *Nech náš ďalší bod* $x^1 = (-2, \frac{2}{3} + \sqrt{2})^T$, *tento bod sme zvolili tak, aby ležal na hranici množiny* P^1. *Obdobne by sme pokračovali ako pri bode* x_0. *Výpočty prenecháme na čitateľa a uvedieme už len, ako bude vyzerať aproximácia množiny Sp pomocou* $P^2 = P_{z_1} \bigcup P_{z_2}$.



Obr. 5: Príklad aproximácie množiny Sp pomocou množiny P^2

2.2 Tvorba série úloh

V predchádzajúcej časti sme uviedli najrozumnejší spôsob aproximácie množiny Sp. Teraz sa budeme venovať zostaveniu úlohy lineárneho programovania, ktorá bude využívať túto aproximáciu. Majme konečnú množinu vektorov $Z = \{z_1, z_2, \ldots, z_l\}$, pomocou ktorých vytvoríme aproximáciu množiny Sp vyššie popísaným spôsobom. Dostávame

$$z_i^T H_0 z_i + x_1 z_i^T H_1 z_i + \cdots + x_k z_i^T H_k z_i \geq 0, \quad \text{pre} \quad i = 1, 2, \ldots, l.$$

Tieto nerovnosti môžme zapísať v maticovom tvare ako $Px + b \geq 0$, kde zložky matice P sú $P_{i,j} = z_i^T H_j z_i$ a zložky vektora b sú $b_i = z_i^T H_0 z_i$.

2.2.1 Lineárne programovanie

$$\begin{aligned} \min_x \quad & c^T x \\ \text{s.t.} \quad & Px + b \geq 0. \end{aligned} \tag{LP}$$

Vzhľadom na to, že v úlohe (LP) optimalizujeme cez väčšiu množinu než v úlohe (SDP), tak platí vzťah pre optimálne hodnoty účelovej funkcie:

$$c^T \hat{x}_{LP} \leq c^T \hat{x}_{SDP}.$$

To znamená, že vyriešením úlohy LP dostávame dolný odhad úlohy SDP. Pred navrhnutím samotného algoritmu a jeho možných modifikácií ešte spomenieme veľmi dôle-

žité poznatky vzťahujúce sa ku komplementarite a k duálnej úlohe. V predchádzajúcej kapitole sme v rámci podmienok optimality uviedli aj podmienku komplementarity (4). S touto podmienkou súvisí aj nasledujúca veta, ktorá je prebratá z článku Complementarity and Nondegeneracy in Semidefinite Programming od autorov F. Alizadeh, J.P.A. Haeberly [14]. V spomenutom článku sú taktiež rozobraté kritéria, kedy sú úlohy (SDP) a (SDD) nedegenerované.

Veta 2.4. *Nech x a Y sú prípustné riešenia v (SDP) a (SDD). Potom sú optimálne vtedy a len vtedy, ak existuje $Q \in \mathbb{R}^{n\times n}$ spĺňajúca $Q^TQ = I$, a $\lambda, \omega \in \mathbb{R}^n_+$ také, že*

$$Y = QDiag(\lambda_1, \ldots, \lambda_n)Q^T, \tag{7}$$

$$\mathcal{A}(x) = QDiag(\omega_1, \ldots, \omega_n)Q^T, \tag{8}$$

$$\lambda_i\omega_i = 0, \quad i = 1, \ldots, n. \tag{9}$$

Nech $\hat{x}$ je optimálne riešenie úlohy (SDP) a r je násobnosť nuly ako vlastnej hodnoty matice $\mathcal{A}(\hat{x})$ a $z_1, \ldots, z_r$, sú prislúchajúce vlastné vektory. Potom z Vety 2.4 vyplýva, že optimálne riešenie duálnej úlohy (SDD) môžme hľadať v tvare $Y = y_1z_1z_1^T + \cdots + y_rz_rz_r^T$. Pre pripomenutie v úlohe (SDD) sme maximalizovali $-Y \bullet H_0$, čo je ekvivalentné s minimalizáciou $Y \bullet H_0$. Takže úloha (SDD) bude mať nasledujúci tvar:

$$\begin{aligned} \min_y \quad & \sum_{j=1}^{r} y_j z_j^T H_0 z_j \\ \text{s.t.} \quad & \sum_{j=1}^{r} y_j z_j^T H_i z_j = c_i, \quad \text{pre} \quad i = 1, \ldots, k, \\ & y_j \geq 0, \quad j = 1, 2, \ldots, r. \end{aligned} \tag{MSDD}$$

Úlohu (MSDD) budeme nazývať komplementárna duálna úloha. Teoreticky, by sme mohli vytvoriť úlohu (MSDD) aj pomocou iných vektorov ako z_i, avšak potom riešenie by nespĺňalo podmienku komplementarity. Vidno, že úloha (MSDD) je úlohou lineárneho programovania. Takže v prípade, že poznáme optimálne riešenie primárnej úlohy (SDP), tak vieme duálnu úlohu previesť na jednoduchý problém lineárnej optimalizácie. Z Vety 2.4 vyplýva, že optimálne riešenie úlohy (MSDD) je optimálnym riešením aj úlohy (SDD). Teraz, keď máme vybudovanú teóriu, budeme sa venovať zostaveniu samotného algoritmu.

2.3 Algoritmus

2.3.1 Myšlienky a opis

Pre začiatok predpokladajme, že platí predpoklad 1, ako aj Vety 1.11 to znamená, že platí Veta o silnej dualite. Náš algoritmus sa dá rozdeliť na tri hlavné časti. V prvej sa konštruuje aproximácia množiny Sp. Pre začiatok, ak nie je zadaný štartovací x^0 bod, tak naša metóda si ho sama náhodne vygeneruje. Majme bod x^i Následne sa skonštruuje matica $\mathcal{A}(x^i)$ a vypočítajú sa jej vlastné hodnoty a vektory z_j^i. Označíme z_m^i vlastný vektor, ktorý prislúcha najmenšej vlastnej hodnote. Vektory z_j^i nám vygenerujú n ohraničení. Avšak toto je relatívne malý počet ohraničení. Preto sme navrhli dva spôsoby ako získať ďalšie ohraničenia na lepšiu aproximáciu množiny Sp. Prvý spôsob je výpočtovo náročnejší, ale získame zaručene $2n$ dotykových ohraničení pre množinu Sp. Spočíva vo výpočte najmenšej vlastnej hodnoty a prislúchajúceho vlastného vektoru pre matice $\mathcal{A}(x^i + ke_j)$ a $\mathcal{A}(x^i - ke_j)$ pre $j = 1, \ldots, n$, kde k je konštanta, ktorú si určíme. Druhý spôsob, ako získať ďalšie ohraničenia je výpočtovo menej náročný a spočíva v tom, že narušíme pomocou náhodne vygenerovaných odchýlok, už vypočítaný vektor z_m^i, ktorý prislúcha najmenšej vlastnej hodnote matice $\mathcal{A}(x^i)$. Hlavná myšlienka za týmto spôsobom je nasledovná, keďže vektor z_m^i nám generuje dotykovú nadrovinu $p_{z_m^i} x + h_{z_m^i} \geq 0$, a keď ho jemne narušíme na $z_m + \epsilon$, tak zo spojitosti násobenia dostaneme, že nové ohraničenie pre $z_m^i + \epsilon$ bude mať jemne zmenený sklon, avšak neposunie sa oveľa od dotykovej nadroviny. Tento spôsob je, už ako sme spomínali, na výpočty menej náročný, avšak už nemáme garanciu, že dostaneme dotykové nadroviny k množine Sp. Voľnosť tohto spôsobu spočíva v nastavení generovania náhodných odchýlok ϵ. Po získaní všetkých vektorov vytvoríme aproximáciu množiny Sp pomocou množín P_z a vytvoríme maticu P a vektor b pre úlohu (LP) a pôjdeme na druhú časť nášho algoritmu. V druhej časti vyriešime úlohu (LP) pomocou metódy vnútorného bodu a získame x^{i+1}. Metódy vnútorného bodu sme zvolili z dôvodu, že sa jedná o najrýchlejší spôsob, ako vyriešiť úlohu (LP). Táto metóda však môže mať problémy, ak počet ohraničení vysoko presahuje počet premenných. Preto je dôležité, aby sme v matici P nemali nepotrebné ohraničenia, a preto po optimalizácií ponecháme v matici P iba aktívne ohraničenia. Pôvodne pri prvotnom návrhu nášho algoritmu sme pracovali so štartovacím bodom x^0 vo vnútri množiny Sp a pomocou bisekcie sme vypočítali nové

x^{i+1}, tak aby ležalo na hranici Sp. Dôvod bol, že na úplnom začiatku sme dokázali Vetu 2.2 iba pre $x \in \partial Sp$. Neskôr sme našli celé znenie Vety 2.2 v práci [11]. Vďaka Vete 2.2 sme mohli upustiť od značne obmedzujúceho predpokladu, že x^0 je z vnútra Sp, ako aj od bisekcie, ktorá celý náš algoritmus iba spomaľovala a robila nekompetitívnym voči algoritmom na riešenie úloh (SDP). Tretia časť nášho algoritmu sú zastavovacie kritéria. Používame konkrétne tri kritéria pre zastavenie algoritmu. Prvé klasické kritérium je, že algoritmus ukončíme ak v dvoch po sebe nasledujúcich iteráciách $\|x^i - x^{i+1}\| \leq \epsilon_1$. Druhé zastavovacie kritérium sme získali z publikácie Úvod do optimalizace od J. Dupačová a P. Lachout [10]. Podstatou tohto kritéria je, že zastavíme, ak hodnota účelovej funkcie m-krát neklesla o viac ako o určené ϵ_2, m je taktiež parameter. Toto kritérium je veľmi užitočné, ak optimálne riešenie sa nachádza na hranici množiny Sp, ktorá má lokálne lineárny charakter. Tretie kritérium spočíva v konštrukcií a výpočte komplementárnej duálnej úlohy a výpočte upravenej primárno-duálnej medzery. Máme bod x^{i+1}. Nech vektory $z_1^{i+1}, \ldots, z_r^{i+1}$ prislúchajú najmenšej vlastnej hodnote matice $\mathcal{A}(x^{i+1})$. Pomocou týchto vektorov vytvoríme pseudo-komplementárnu úlohu (MSDD) a získame riešenie $\hat{y^{i+1}}$ v prípade, že existuje. Vytvoríme maticu $\hat{Y}^{i+1} = \sum_{j=1}^{r} y_j^{i+1} z_j^{i+1} (z_j^{i+1})^T$ táto matica je určite prípustná v duálnej úlohe (SDD). Dôvod prečo vytvorená úloha je pseudo-komplementárna je ten, že výraz $\hat{Y}^{i+1} \bullet \mathcal{A}(x^{i+1})$ nie je rovný nule, ale výrazu $\lambda_m^{i+1} \left(\sum_{j=1}^{r} \hat{y}_j^{i+1} \right)$. Ako sa blížime k hranici množiny Sp, tak tento výraz ide k nule a teda matice $\hat{Y}^{i+1}$ začínajú spĺňať podmienku komplementarity (4). Samotné zastavovacie kritérium je postavené na Vete 1.12. Problém je, že predpoklady tejto vety vo všeobecnosti nemusia byť splnené pre naše x^{i+1}, lebo sa môže nachádzať mimo množiny Sp. Avšak pre všetky body $x \in \partial Sp$ veta o slabej dualite platí a zo silnej duality vyplýva, že pre všetky $\bar{x} \in \partial Sp$ okrem optimálnych riešení platí:

$$c^T \bar{x} + H_0 \bullet \hat{Y}^{i+1} > 0.$$

Dalo by sa ukázať, že existuje malé okolie bodov $\bar{x}$ také, že slabá veta o dualite stále platí aj mimo množiny Sp. Preto, v našom poslednom kritériu, budeme pracovať s výrazom $|c^T x^{i+1} + H_0 \bullet \hat{Y}^{i+1}|$. Avšak, ako už sme spomenuli mohlo, by sa stať, že ďalej od hranice množiny môže byť tento výraz nulový. Preto budeme mať aj penalizáciu za vzdialenosť od hranice a to rovnú $|\lambda_m^{i+1}|$, ktorá ak sme na hranici a teda v množine prípustných riešení úlohy (SDP), je rovná nule. Takže posledné zastavovacie kritérium má tvar

$|c^T x^{i+1} + H_0 \bullet \hat{Y}^{i+1}| + k|\lambda_m^{i+1}| \leq \epsilon_3$, kde ϵ_3, k volíme podľa toho akú veľkú presnosť chceme. Výhodou tohto postupu je to, že priebežne počítame aj hodnotu duálnej úlohy (SDD), ak konvergujeme k riešeniu úlohy (SDP) tak dostaneme aj riešenie duálnej úlohy. V prípade, že by sme nepotrebovali počítať duálnu úlohu, môžme toto kritérium nepoužiť, čím zrýchlime samotný algoritmus. Ale zrýchlenie nemusí byť veľmi veľké, keďže počítame navyše len jednu úlohu lineárneho programovania, ktorá ma zakaždým rovnaký počet ohraničení. Navyše, nie vždy musíme použiť algoritmy na výpočet úloh lineárneho programovania. Väčšinou násobnosť najmenšej vlastnej hodnoty $\mathcal{A}(x^{i+1})$ je rovná jednej a teda úloha (MSDD) sa zjednoduší na úlohu:

$$\begin{aligned} \min_y \quad & y(z_m^{i+1})^T H_0 z_m^{i+1} \\ \text{s.t.} \quad & y(z_m^{i+1})^T H_j z_m^{i+1} = c_j, \quad \text{pre} \quad j = 1, \ldots, k, \\ & y \geq 0. \end{aligned}$$

Táto úloha má riešenie iba ak x^i je optimálne riešenie (SDP) a je rovné $\hat{y} = \frac{\sum_{i=1}^k c_i}{(z_m^{i+1})^T \left(\sum_{i=1}^k H_i\right) z_m^{i+1}}$. Ak x^i nie je optimálnym riešením úlohy (SDP) tak vo všeobecnosti táto jednorozmerná úloha nemusí mať riešenie. Kvôli výpočtovým dôvodom v našom algoritme aproximujeme riešenie $\hat{y}$ pomocou výrazu $\hat{y} = \frac{\sum_{i=1}^k c_i}{z^T \left(\sum_{i=1}^k H_i\right) z}$.

Vstup: $x^0, \epsilon, maxiter$

1 **while** $i<=maxiter$ **do**
2 výpočet vlastných vektorov matice $\mathcal{A}(x^{(i)})$;
3 výpočet ohraničenia generujúcich vektorov ;
4 výpočet matice P a vektora b;
5 vyriešenie úlohy (LP) $\rightarrow x^{(i+1)}$;
6 overenie zastavovacích kriérií;
7 konštrukcia úlohy (MSDD) a jej výpočet;
8 **if** *primárno-duálna medzera<eps* **then**
9 koniec
10 **end**
11 **end**

Algorithm 1: Stručný popis algoritmu

2.3.2 Úloha s rovnosťami

Na koniec kapitoly, by sme radi rozobrali možnosť použitia nášho algoritmu v prípade, že v úlohe máme aj lineárne ohraničenia v tvare rovnosti.

$$\begin{aligned} \min_x \quad & c^T x \\ \text{s.t.} \quad & \mathcal{A}(x) = H_0 + x_1 H_1 + \cdots + x_k H_k \succeq 0 \\ & A_{eq} x = b_{eq}. \end{aligned} \tag{SDP_{eq}}$$

V takom prípade množina prípustných riešení bude mať tvar

$$Sp_{eq} = \left\{ x \in \mathbb{R}^k : \mathcal{A}(x) \succeq 0, \quad A_{eq} x = b_{eq} \right\}.$$

Existujú dva spôsoby ako sa vysporiadať s touto zmenou. Jedným je odvodenie novej duálnej úlohy a následné riešenie série úloh s rovnosťami. Druhý spôsob, ktorý sme sa rozhodli implementovať my, pozostáva z prevedenia úlohy s rovnosťami do tvaru úlohy (SDP). Budeme vychádzať z tvrdenia

Lema 2.5. *Nech $b \in \mathcal{S}(A)$, a nech $z^{(1)}, \ldots, z^{(p)}$ je báza priestoru $\mathcal{N}(A)$. Majme sústavu lineárnych rovníc $Ax = b$. Potom všetky riešenia tejto sústavy majú tvar:*

$$\hat{x} = x^p + \sum_{i=1}^{p} v_i z^{(i)}.$$

Na základe tejto lemy vidno, že ak budeme hľadať riešenie úlohy (2.3.2) v tvare $\hat{x} = x^p + \sum_{i=1}^{p} v_i z^i$, kde v_i sú parametre, tak všetky prípustné riešenia pre modifikovanú úlohu budú určite spĺňať podmienku $Ax = b$. Dostaneme teda úlohu

$$\begin{aligned} \min_v \quad & c^T (x^p + \sum_{i=1}^{p} v_i z^{(i)}) \\ \text{s.t.} \quad & \mathcal{A}(x^p + \sum_{i=1}^{p} v_i z^{(i)}) = H_0 + (x_1^p + \sum_{i=1}^{p} v_i z_1^{(i)}) H_1 + \cdots + (x_k^p + \sum_{i=1}^{p} v_i z_k^{(i)}) H_k \succeq 0. \end{aligned}$$

Vidno, že túto úlohu vieme previesť na ekvivalentnú úlohu v tvare klasickej úlohy (SDP).

$$\begin{aligned} \min_v \quad & \widetilde{c}^T v + c^T x^p \\ \text{s.t.} \quad & \widetilde{\mathcal{A}}(v) = \widetilde{H}_0 + v_1 \widetilde{H}_1 + \cdots + v_p \widetilde{H}_p \succeq 0, \end{aligned}$$

kde $\widetilde{c}^T = (c^T z^{(1)}, \ldots, c^T z^{(p)})$, $\widetilde{H}_0 = H_0 + \sum_{i=1}^{k} x_i^p H_i$ a $\widetilde{H}_j = \sum_{i=1}^{k} z_i^{(j)} H_i$ pre $j = 1, 2, \ldots, p$. Na túto úlohu, s pozmenenou účelovou funkciou a ohraničeniami, už môžeme použiť náš algoritmus, ktorý nám vypočíta vektor $\hat{v}$, na základe ktorého budeme vedieť zostaviť optimálne riešenie $\hat{x}$ v pôvodnej úlohe (2.3.2).

3 Semidefinitné programovanie s nelineárnou účelovou funkciou

3.1 Kvadratická účelová funkcia

Po predchádzajúcej kapitole, kde sme sa zaoberali úlohou s lineárnou účelovou funkciou a spektrahedrálnou množinou prípustných riešení, sme riešili iba aproximáciu množiny prípustných riešení. Samotná účelová funkcia vstupovala do nášho algoritmu až vo chvíli, keď sme chceli vypočítať sériu úloh s aproximovanou množinou Sp. Preto sa ponúka otázka či by sme poznatky, ktoré sme využívali v predchádzajúcej kapitole nevedeli zúžitkovať v prípade, keď máme rovnaký druh množiny prípustných riešení, ale komplikovanejšiu účelovú funkciu. Konkrétne sa budeme zaoberať návrhom algoritmu pre výpočet úlohy s kvadratickou účelovou funkciou a spektrahedrálnou množinou prípustných riešení. Takže tvar úlohy bude:

$$\begin{aligned} \min_{x} \quad & \frac{1}{2} x^T G x + c^T x \\ \text{s.t.} \quad & \mathcal{A}(x) \succeq 0. \end{aligned} \tag{KSP}$$

Pre začiatok uvedieme niekoľko predpokladov na účelovú funkciu, s ktorými budeme naďalej pracovať.

Predpoklad 2. *Matica $G \in S^k$ je kladne semidefinitná.*

Dôsledok 3.1. *Z predpokladu 2 vyplývajú nasledujúce vlastnosti:*

- *matica G má spektrálny rozklad $G = SDS^T$, označme s_i i-ty stĺpec matice S a r_i^T i-ty riadok matice S,*
- *účelová funkcia úlohy KSP je konvexná a teda jedná sa o úlohu konvexného programovania.*

Bez újmy na všeobecnosti, môžme predpokladať, že prvky na diagonále matice Λ sú zoradené od najmenšej po najväčšiu, vlastné vektory v matici S majú odpovedajúce poradie. Teda prvých $k-r$ stĺpcov matice S , kde r je hodnosť matice, odpovedá vlastným vektorom pre vlastné číslo nula.

Hlavná myšlienka nášho algoritmu je rovnaká ako v predchádzajúcej kapitole. Opäť sa budeme pomocou aproximácie množiny prípustných riešení snažiť nahradiť pôvodnú úlohu sériou jednoduchších úloh. Tieto úlohy budú mať tvar klasického problému kvadratického programovania. Tak, ako aj v Kapitole 2, na aproximáciu budeme používať vlastné vektory pre maticu $\mathcal{A}(x)$, pomocou ktorých vytvoríme množiny P_z. Úlohy s aproximovanou množinou prípustných riešení budú mať tvar

$$\begin{aligned} \min_x \quad & \frac{1}{2}x^T G x + c^T x \\ \text{s.t.} \quad & Px + b \geq 0. \end{aligned} \tag{QP}$$

Popis tvorby matice P bol rozobraný v časti (2.2). Náš algoritmus bude opäť rozdelený na tri časti, pričom prvá časť je skoro totožná s algoritmom, ktorý sme rozoberali v Kapitole 2. Menšie zmeny nastanú v druhej časti a najmä v tretej pri výpočte duálnej úlohy. Pred samotným popisom algoritmu sa budeme najskôr venovať možnosti zmeny súradníc. Dôvodom na zmenu súradníc je zjednodušenie účelovej funkcie a následne aj jednoduchšie odvodenia duálnej úlohy. Na zmenu bázy využijeme spektrálny rozklad matice G a to tak, že nové súradnice budú:

$$\begin{aligned} \tilde{x} &= S^T x, \\ x &= S\tilde{x}, \\ x_i &= r_i^T \tilde{x}. \end{aligned}$$

Keďže matica S je ortogonálna, tak táto zmena súradníc geometricky odpovedá otočeniu osí súradnicovej sústavy. Účelová funkcia v nových premenných bude mať tvar: $\frac{1}{2}\tilde{x}^T \Lambda \tilde{x} + c^T S\tilde{x}$. Pre jednoduchosť označme $\tilde{c}^T = c^T S$. Množina Sp sa zmenou súradníc nezmení, avšak bude mať v nových súradniciach iný predpis.

$$\begin{aligned}\mathcal{A}(x) &= H_0 + \sum_{i=1}^{k} x_i H_i \\ &= H_0 + \sum_{i=1}^{k} r_i^T \tilde{x} H_i \\ &= H_0 + \sum_{i=1}^{k} \left(\sum_{j=1}^{k} g_{i,j} \tilde{x}_j \right) H_i \\ &= H_0 + \sum_{j=1}^{k} \tilde{x}_j \left(\sum_{i=1}^{k} g_{i,j} H_i \right) \\ &= H_0 + \sum_{j=1}^{k} (\tilde{x}_j \sum_{i=1}^{k} s_i^{(j)} H_i).\end{aligned}$$

Označme $\tilde{H}_j = \sum_{j=1}^{k} s_i^{(j)} H_j$. Dostávame vyjadrenie ohraničení $\mathcal{A}(x)$ v nových súradniciach

$$\tilde{\mathcal{A}}(\tilde{x}) = H_0 + \sum_{i=1}^{k} \tilde{x}_i \tilde{H}_i.$$

Potom po zmene súradníc bude úloha (KSP) mať tvar

$$\begin{aligned} \min_{\tilde{x}} \quad & \frac{1}{2} \tilde{x}^T D \tilde{x} + \tilde{c}^T \tilde{x} \\ \text{s.t.} \quad & \tilde{\mathcal{A}}(\tilde{x}) \succeq 0. \end{aligned} \tag{KSP*}$$

Vidno, že úlohy (KSP) a (KSP*) sú ekvivalentné, pričom úloha (KSP*) má jednoduchšiu štruktúru. Preto odvodíme Lagrangeovu funkciu, Karush-Kuhn-Tuckerove podmienky, ako aj tvar duálnej úlohy pre (KSP*). Lagrangeova funkcia bude mať tvar

$$\begin{aligned} L(\tilde{x}, Y) =& \frac{1}{2} \tilde{x}^T \Lambda \tilde{x} + \tilde{c}^T \tilde{x} - Y \bullet \tilde{\mathcal{A}}(\tilde{x}) \\ =& \frac{1}{2} \sum_{i=1}^{k} \left(d_i \tilde{x}_i^2 + \tilde{c}_i \tilde{x}_i - \tilde{x}_i Y \bullet \tilde{H}_i \right) - Y \bullet H_0. \end{aligned} \tag{10}$$

Keď už máme tvar Lagrangeovej funkcie môžme skonštruovať Karush-Kuhn-Tuckerove

podmienky pre úlohu (KSP). K-K-T podmienky budú mať tvar

$$\tilde{c}_j - \hat{Y} \bullet H_j = 0, \quad \text{pre} \quad j = 1, \dots, k-r \tag{11}$$

$$d_i \tilde{x}_i + \tilde{c}_i - \hat{Y} \bullet H_i = 0, \quad i = n-r+1, \dots, k, \tag{12}$$

$$H_0 + \sum_{i=1}^{k} \hat{x}_i H_i \succeq 0, \tag{13}$$

$$\hat{Y} \bullet \left(H_0 + \sum_{i=1}^{k} \hat{x}_i H_i \right) = \hat{Y} \bullet \mathcal{A}(\hat{x}) = 0, \tag{14}$$

$$\hat{Y} \succeq 0. \tag{15}$$

Wolfeho duálna úloha má vo všeobecnosti tvar $Max\{L(x,y) | \nabla_x L(x,y) = 0\}$. Pre našu úlohu bude Wolfeho úloha mať tvar:

$$\begin{aligned} \max_{\tilde{x},Y} \quad & L(\tilde{x}, Y) \\ \text{s.t.} \quad & \tilde{c}_j - Y \bullet H_j = 0, \quad j = 1, \dots, k-r, \\ & d_i \tilde{x}_i + \tilde{c}_i - Y \bullet H_i = 0, \quad i = k-r+1, \dots, k, \\ & Y \succeq 0. \end{aligned} \tag{16}$$

Vidno, že ak $\tilde{x}_i$ prislúcha $d_i = 0$, tak pre tieto premenné úloha má lineárny charakter. Ak $d_i \neq 0$ tak môžeme, po úprave podmienok na $\tilde{x}_i = \frac{1}{d_i} Y \bullet H_i, \quad i = k-r+1, \dots, k$, eliminovať tak, že ich dosadíme do účelovej funkcie:

$$\begin{aligned} L(Y) &= \sum_{i>k-r}^{k} \left[\frac{(Y \bullet H_i - \tilde{c}_i)^2}{2d_i} + \frac{(Y \bullet H_i - \tilde{c}_i)}{d_i} \tilde{c}_i - \frac{(Y \bullet H_i - \tilde{c}_i)}{d_i} Y \bullet \tilde{H}_i \right] \\ &+ \sum_{i=1}^{k-r} \underbrace{\left[\tilde{c}_i \tilde{x}_i - \tilde{x}_i (Y \bullet \tilde{H}_i) \right]}_{0} - Y \bullet H_0 \\ &= \sum_{i>k-r}^{k} \left[-\frac{(Y \bullet \tilde{H}_i)^2}{2d_i} + \frac{(Y \bullet \tilde{H}_i)}{d_i} \tilde{c}_i - \frac{\tilde{c}_i^2}{d_i} \right] - Y \bullet H_0. \end{aligned}$$

Aby sme nemali mínusové znamienko, pri kvadratických členoch budeme minimalizovať $-L(Y)$. Potom Wolfeho úloha bude mať tvar:

$$\begin{aligned} \min_{\tilde{x},Y} \quad & \sum_{i>k-r}^{k} \left[\frac{(Y \bullet \tilde{H}_i)^2}{2d_i} - \frac{(Y \bullet \tilde{H}_i)}{d_i} \tilde{c}_i + \frac{\tilde{c}_i^2}{d_i} \right] + Y \bullet H_0 \\ \text{s.t.} \quad & \tilde{c}_j - Y \bullet H_j = 0, \quad j = 1, \dots, k-r, \\ & Y \succeq 0. \end{aligned} \tag{KSD}$$

V prípade, že by matica G bola kladne definitná, t.j všetky d_i by boli nenulové, tak Wolfeho duálna úloha k úlohe (KSP) bude mať tvar.

$$\begin{aligned} \min_{Y} \quad & \sum_{i=1}^{k} \left[\frac{1}{2d_i} \left(Y \bullet \tilde{H}_i \right)^2 - \frac{(Y \bullet \tilde{H}_i)\tilde{c}_i}{d_i} \right] + Y \bullet H_0 + \frac{1}{2}\tilde{c}^T D^{-1} \tilde{c} \\ \text{s.t.} \quad & Y \succeq 0. \end{aligned}$$

Tak ako v predošlej kapitole, by sme chceli nájsť spôsob, ako zjednodušiť duálnu úlohu resp. nájsť jednoduchú podúlohu, ktorej riešenie bude komplementárne s maticou $\mathcal{A}(x)$.

Veta 3.2. *Nech $\hat{x}$ je optimálne riešenie (KSP) a $\hat{Y}$ je optimálne riešenie úlohy KSD. Potom existuje $Q \in \mathbb{R}^{n\times n}$ spĺňajúca $Q^TQ = I$, a $\lambda, \omega \in \mathbb{R}^n_+$ také, že*

$$\hat{Y} = QDiag(\lambda_1, \ldots, \lambda_n)Q^T, \tag{17}$$

$$\mathcal{A}(\hat{x}) = QDiag(\omega_1, \ldots, \omega_n)Q^T, \tag{18}$$

$$\lambda_i\omega_i = 0, \quad i = 1, \ldots, n. \tag{19}$$

Skôr, ako dokážeme túto vetu, dokážeme pomocné tvrdenie:

Lema 3.3. *Nech $A, B \in S^n$ sú komutujúce matice. Potom A, B majú spoločné vlastné vektory.*

Dôkaz. Označme $s_1, s_2, \ldots, s_n$, vlastné vektory matice A. Keďže A je symetrická tak vytvárajú ortogonálnu bázu priestoru $\mathbb{R}^n$. Nech λ je ľubovoľná vlastná hodnota matice A. Označme $U_\lambda = \{x : (A-\lambda I)x = 0\}$ množinu všetkých vlastných vektorov prislúchajúcich k hodnote λ. Vidno, že U_λ je podpriestor v $\mathbb{R}^n$ dimenzie k, kde k je násobnosť vlastnej hodnoty λ. Ukážeme, že pre maticu B a pre všetky $x \in U_\lambda$ platí $Bx \in U_\lambda$. Nech $x \in U_\lambda$ potom

$$ABx = BAx = B\lambda x = \lambda Bx.$$

Úpravou dostaneme, že $(A - \lambda I)Bx = 0$ a teda $Bx \in U_\lambda \forall x \in U_\lambda$. Toto znamená, že podpriestor U_λ je invariantný vzhľadom na zobrazenie B. Zvoľme $s_1, \ldots, s_k$ ako bázu podpriestoru U_λ a nech $S_1 = (s_1, \ldots, s_k)$. Túto bázu doplníme vektormi $s_{k+1}, \ldots, s_n$ a usporiadame ich do matice $S_2 = (s_{k+1}, \ldots, s_n)$. Vidno, že matica $S = [S_1, S_2]$ je ortogonálna matica a teda platí $SS^T = I$, blokovo $S_1^TS_1 = I_k$, $S_2^TS_2 = I_{n-k}$, $S_1^TS_2 = 0$. Keďže U_λ je invariantný vzhľadom na zobrazenie B, tak platí $Bs_i \in U_\lambda$ pre $i = 1, \ldots, k$.

Takže Bs_i je lineárnou kombináciou $s_1, \ldots, s_k$ a teda $BS_1 = S_1C$ pre nejakú $k \times k$ maticu C. Ak $BS_1 = S_1C$ potom $BS = [BS_1 \quad BS_2] = [S_1C \quad BS_2]$ a teda

$$\begin{aligned} S^TBS =& [S^TBS_1 \quad S^TBS_2] \\ =& [S^TS_1C \quad S^TBS_2] \\ =& [\left[\begin{smallmatrix} I_k \\ 0 \end{smallmatrix}\right] C \quad S^TBS_2] = \begin{bmatrix} C & E \\ 0 & D \end{bmatrix}. \end{aligned}$$

Pričom blok $E = S_1^TBS_2$. Keďže B je symetrická, tak platí $S_1^TB = (BS_1)^T = (S_1C)^T = C^TS_1^T$. Dostávame $E = C^TS_1^TS_2 = 0$. Matica S^TBS má blokovo diagonálny tvar. Keďže matica B je symetrická, tak aj bloky na diagonále sú symetrické a teda diagonalizovateľné. Nech z je vlastný vektor matice C a μ je prislúchajúca vlastná hodnota. Potom $0 \neq S_1z \in U_\lambda$, a navyše $B(S_1z) = (BS_1)z = S_1Cz = \mu S_1z$. To znamená, že matica B má vlastný vektor v U_λ. Keďže z bol ľubovolný z k vlastných vektorov, tak matica B má k lineárne nezávislých vlastných vektorov v podpriestore U_λ. Dôvod, prečo sú lineárne nezávislé je ten, že vlastné vektory matice C sú lineárne nezávislé a nulový priestor matice S_1 je iba vektor samých núl. Takže dostávame, že matica B má k lineárne nezávislých vektorov v podpriestore U_λ, takže tvoria jeho bázu. Teda vektory $s_1, \ldots, s_k$ budú lineárnou kombináciou tejto bázy a teda dostávame, že vektory $s_1, \ldots, s_k$ sú taktiež vlastnými vektormi matice B. Keďže λ bola zvolená ľubovolná, tak dostávame, že matica B má rovnaké vlastné vektory ako matica A. □

Teraz dokážeme Vetu 3.2

Dôkaz. Keďže $\hat{x}, \hat{Y}$ sú optimálne riešenia tak spĺňajú (14) a teda $\hat{Y} \bullet \mathcal{A}(\hat{x}) = 0$. Z tvrdenia (1.7) vyplýva, že

$$\hat{Y}\mathcal{A}(\hat{x}) = 0.$$

Z podmienky komplementarity, $\hat{Y} \bullet \mathcal{A}(\hat{x}) = 0$ vyplýva, že matice $\hat{Y}, \mathcal{A}(\hat{x})$ spolu komutujú. Potom využitím lemy 3.3 dostávame, že matice $\hat{Y}$ a $\mathcal{A}(\hat{x})$ majú rovnaké vlastné vektory $q_1, \ldots, q_n$ a teda, keď vezmeme maticu Q takú, že jej stĺpce sú vlastné vektory q_i, tak platí

$$\begin{aligned} \hat{Y} =& Q\Lambda Q^T, \\ \mathcal{A}(\hat{x}) =& Q\Omega Q^T. \end{aligned}$$

Po dosadení do $\mathcal{A}(\hat{x}) = 0$ a úprave dostávame $\Lambda\Omega = 0 \Leftrightarrow \lambda_i\omega_i = 0$ pre $i = 1, \ldots, n$.

□

Vďaka tomuto tvrdeniu bude pre optimálne riešenie $\hat{x}$ úlohy (KSP) platiť, že môžme $\hat{Y}$ hľadať v tvare $Y = \sum_{j=1}^{p} y_j z_j z_j^T$ pričom $\mathcal{A}(\bar{x})z_j = 0$ pre $j = 1, \ldots, p$. Teda dostávame špeciálny typ podúlohy (KSD):

$$\begin{aligned}
\min_{y} \quad & \sum_{i>k-r}^{k} \left(\frac{\left[\sum_{j=1}^{p} y_j (z_j^T \tilde{H}_i z_j)\right]^2}{2d_i} - \frac{(\sum_{j=1}^{p} y_j z_j^T \tilde{H}_i z_j)}{d_i} \tilde{c}_i \right) + \sum_{j=1}^{p} y_j z_j^T H_0 z_j + \frac{1}{2} \tilde{c}^T D^{-1} \tilde{c} \\
\text{s.t.} \quad & \sum_{j=1}^{p} y_j z_j^T \tilde{H}_i z_j = \tilde{c}_i, \quad i = 1, \ldots, k-r \\
& y \geq 0.
\end{aligned} \tag{MKSD}$$

Úlohu (MKSD) vieme prepísať do tvaru úlohy kvadratického programovania:

$$\begin{aligned}
\min_{y} \quad & \frac{1}{2} y^T M y + u^T y + \frac{1}{2} \tilde{c}^T D^{-1} \tilde{c} \\
\text{s.t.} \quad & Ay = b, \\
& y \geq 0.
\end{aligned}$$

Označme $m_i^T = (z_1^T \tilde{H}_i z_1, \ldots, z_p^T \tilde{H}_i z_p)^T$, pre $i = 0, 1, \ldots, k$. Potom

$$\begin{aligned}
M &= \sum_{i>k-r}^{k} \frac{1}{d_i} m_i m_i^T, \\
u^T &= m_0^T - \sum_{i>k-r}^{k} \frac{\tilde{c}_i}{d_i} m_i^T, \\
A &= \begin{pmatrix} m_1^T \\ \vdots \\ m_{k-r}^T \end{pmatrix} \\
b &= \begin{pmatrix} c_1 \\ \vdots \\ c_{k-r} \end{pmatrix}.
\end{aligned}$$

Vidno, že matica M je kladne semidefinitná a teda sa jedná o úlohu konvexného programovania.

3.1.1 Popis celého algoritmu

Nami navrhnutý algoritmus pre riešenie úlohy (KSP) je opäť založený na aproximácií spektrahedrálnej množiny pomocou množín P_z, ktoré boli definované v predchádzajúcej kapitole. Preto jeho jednotlivé časti sú značne podobné algoritmu, ktorý sme navrhli pre riešenie úlohy (SDP). Vzhľadom na tieto podobnosti, ako aj na fakt, že oba algoritmy vychádzajú z rovnakej teórie, budeme používať označenie, ktoré sme zaviedli v predchádzajúcej kapitole a podrobnejšie budeme rozpisovať iba tie miesta algoritmu, kedy sa výraznejšie odlišuje. Náš algoritmus na riešenie úlohy (KSP) budeme odkazovať ako na kvadratický algoritmus, aby sme ho odlíšili od algoritmu, ktorý sme navrhli v kapitole (2). Keďže účelová funkcia úlohy už nie je lineárna, môže sa stať, že globálne minimum sa nachádza vo vnútri množiny Sp. V takom prípade, by duálna premenná Y musela byť nulová matica. Toto vyplýva z Vety 3.2, a teda úloha (KSP), by bola ekvivalentná úlohe na hľadanie voľného extrému. Preto pred samotným pristúpením k iteračnej časti nášho algoritmu vypočítame bod $\bar{x}$, ktorý rieši rovnicu $Gx - c = 0$. Tento bod existuje, ak $G \succ 0$. V prípade, že niektorá z vlastných hodnôt matice G je nulová, tak optimálne riešenie sa bude nachádzať na hranici množiny Sp. Ešte predtým, ako pristúpime k tvorbe úloh typu (QP) spravíme rozklad matice $G = SDS^T$ a následne spravíme substitúciu $\tilde{x} = Sx$, ktorej detaily sme popísali časti 3.1. Zvyšok kvadratického algoritmu rozdelíme na tri časti. Majme bod x^i. Potom prvá časť, totožná ako v algoritme z kapitoly (2), pozostáva z výpočtu vlastných vektorov matice $\mathcal{A}(x_i)$ a už popísanej aproximácie množiny Sp, pomocou polyhedrálnej množiny P_z^i. Zo získaných vlastných vektorov skonštruujeme maticu ohraničení P^i a vektor b^i. Následne prejdeme na druhú časť a tou je konštrukcia a výpočet úlohy (MKSD). Získame $\hat{y}^i$. Následne otestujeme, či je splnené zastavovacie kritérium založené na primárno-duálnej medzere. Rovnako, ako pre algoritmus z Kapitoly 2 nemáme splnené podmienky na platnosť vety o slabej dualite. Ako sme poznamenali v predchádzajúcej kapitole tento nedostatok kompenzujeme tým, že k vypočítanej primárno duálnej medzere pripočítame absolútnu hodnotu najmenšieho vlastného čísla matice $\mathcal{A}(x^i)$. V prípade, že kritérium $|\frac{1}{2}\tilde{x}^T D\tilde{x} + \tilde{c}^T\tilde{x} + \frac{1}{2}y^T My + u^T y + \frac{1}{2}\tilde{c}^T D^{-1}\tilde{c}| + |\lambda_{min}| < eps_3$ nie je splnené, prejdeme na tretiu časť nášho algoritmu a tou je riešenie úlohy kvadratického programovania (QP) s ohraničeniami $P^i x + b^i \geq 0$. Získame nový bod x^{i+1}, pre ktorý otestujeme ďalšie dve zastavovacie kritéria

ria. Prvé z je $||x^{(i)} - x^{(i+1)}|| \leq \epsilon_1$ a to druhé pracuje s rozdielom hodnôt účelovej funkcie.

Vstup: $x^0, \epsilon, maxiter$

1 výpočet matice S a matice D ;
2 substitúcia $\tilde{x} = Sx$;
3 výpočet $\tilde{H}_i$;
4 **while** $i<=maxiter$ **do**
5 výpočet vlastných vektorov matice $\mathcal{A}(x^{(i)})$;
6 konštrukcia úlohy (MKSD) a jej výpočet;
7 **if** */primárno-duálna medzera/*$+|\lambda_{min}| < \epsilon$ **then**
8 koniec
9 **end**
10 výpočet ohraničenia generujúcich vektorov ;
11 výpočet matice P^i a vektora b^i;
12 vyriešenie úlohy (QP) $\rightarrow x^{(i+1)}$;
13 overenie zastavovacích kritérií;
14 **end**
15 prevedenie výsledku do pôvodných súradníc;

Algorithm 2: Stručný popis algoritmu

3.1.2 Úloha s rovnosťami

Obdobne, ako pre (SDP) sme schopný náš algoritmus rozšíriť aj na úlohy s ohraničeniami v tvare $A_{eq}x = b_{eq}$. Toto rozšírenie bude mať veľký význam, lebo nám umožní použiť náš algoritmus na niektoré špeciálne úlohy, ktoré budeme rozoberať v nasledujúcej kapitole. Postup bude rovnaký ako v časti 2.3.2, preto na tomto mieste uvedieme už len skrátený postup. Majme úlohu:

$$\begin{aligned} \min_x \quad & \frac{1}{2}x^T G x + c^T x \\ \text{s.t.} \quad & \mathcal{A}(x) \succeq 0, \\ & A_{eq}x = b_{eq}. \end{aligned} \tag{KSP_{eq}}$$

Označme x^p riešenie rovnice $Ax = b$. Nech $z^{(1)}, \ldots, z^{(p)}$ je báza priestoru $\mathcal{N}(A) = \{z : Az = 0\}$ a označme maticu $Z = (z^{(1)}, \ldots, z^{(p)})$. Potom úloha (KSP_{eq}) je ekvivalentná

s úlohou:

$$\begin{aligned} \min_{v} \quad & \frac{1}{2}v^T\Sigma v + \zeta^T v + c^T x^p + \frac{1}{2}x_p^T G x_p \\ \text{s.t.} \quad & \mathcal{B}(v) = B_0 + \sum_{i=1}^{P} v_i B_i \succeq 0. \end{aligned}$$

Pričom

$$\begin{aligned} \Sigma &= Z^T G Z, \\ \zeta^T &= x_p^T G Z + c^T Z, \\ B_0 &= H_0 + \sum_{i=1}^{k} x_p^i H_i, \\ B_i &= \sum_{i=1}^{k} z_i^{(j)} H_i, \quad i = 1, \dots p. \end{aligned}$$

3.1.3 Úloha so špecifickou účelovou funkciou

V tejto časti sa budeme zaoberať zjednodušeniami, kedy vektor $c^T = \vec{0}$ a matica kvadratickej formy má špeciálny tvar:

$$G_{(n+m)\times(n+m)} = \left[\begin{array}{c|c} 0 & 0 \\ \hline 0 & K \end{array}\right].$$

Môžme predpokladať, že matica $K_{m\times m} \succ 0$. V praxi to znamená, že máme dva druhy premenných. Jedny, ktoré vystupujú v účelovej funkcií a druhé, ktoré vstupujú iba do ohraničení. Dôvod, pre ktorý sa budeme zaoberať takýmto druhom úlohy je obmedzenie numerických chýb, ktoré vznikajú najmä pri výpočte vlastných vektorov. V praxi sa môže stať, že hoci matica G je symetrická, tak z dôvodu numerických chýb Matlab vypočíta vlastné hodnoty, ako komplexné čísla, čo následne znefunkčni náš algoritmus. Preto ukážeme, že nie je potrebné počítať vlastné vektory matice G, ale stačí vypočítať vlastné vektory matice K.

Veta 3.4. *Nech matica A má blokový tvar $G = \left(\begin{smallmatrix} W & 0 \\ 0 & V \end{smallmatrix}\right)$. Nech $V = S\Lambda S^T$ a $W = Q\Sigma Q^T$. Potom matica G je diagonalizovateľná a platí:*

$$A = \left(\begin{array}{c|c} S & 0 \\ \hline 0 & Q \end{array}\right) \left(\begin{array}{c|c} \Lambda & 0 \\ \hline 0 & \Sigma \end{array}\right) \left(\begin{array}{c|c} S^T & 0 \\ \hline 0 & Q^T \end{array}\right)$$

Vďaka tomuto tvrdeniu nám stačí počítať vlastné vektory pre maticu K, keďže ako vlastné vektory nulovej matice môžme zobrať štandardnú bázu priestoru $\mathbb{R}^n$, čím čiastočne znížime numerické chyby.

Na koniec tejto kapitoly, by sme rozobrali prípad, kedy by účelová funkcia nebola konvexná. Vo všeobecnosti v takomto prípade nemáme zaručené platnosti mnohých tvrdení, ktoré sme v tejto práci uviedli, avšak keďže úloha, ktorú riešime v každej iterácií je úloha kvadratického programovania, na ktorej riešenie využívame jednú z naprogramovaných funkcií Matlabu, quadprog. Teoreticky teda vieme dostať lokálne optimum, keďže quadprog nemá vždy zaručenú konvergenciu ku globálnemu optimu, pre takýto druh úloh. Avšak ak by sme o množine Sp vedeli či je kompaktná, tak by náš algoritmus mohol teoreticky konvergovať k optimálnemu riešeniu. Avšak toto je len domnienka, ktorú sme nadobudli na základe niektorých príkladov. Ďalší výskum tohto problému by bol potrebný.

4 Výstupy a aplikácia metód

Pred konkrétnymi aplikáciami zadefinujeme jednú špeciálnu lineárnu maticovú funkciu, ktorá sa bude viackrát objavovať. Nech I je identická matica s rozmerom $\frac{n(n+1)}{2} \times \frac{n(n+1)}{2}$, ktorej stĺpce označíme e_i. Potom matice $M_i = vect^{-1}(e_i)$ tvoria ortogonálnu bázu priestoru S^n. Potom označíme

$$\mathcal{P}(x) = \sum_{i=1}^{\frac{n(n+1)}{2}} x_i M_i.$$

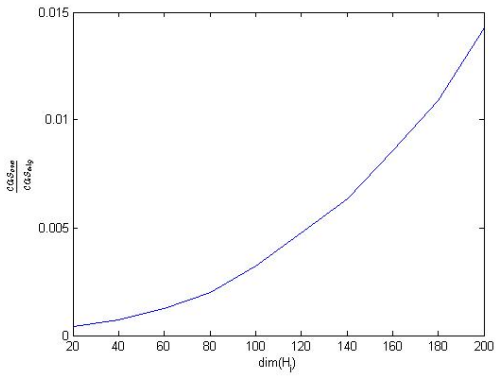
Táto maticová funkcia sa bude vyskytovať viackrát v aplikáciach a budeme ju využívať na to, aby sme prešli od podmienky $X \succeq 0$ k tvaru podmienky $\mathcal{P}(x) \succeq 0$, kde vektor $x = vect(X)$.

4.1 Analýza rýchlosti algoritmu

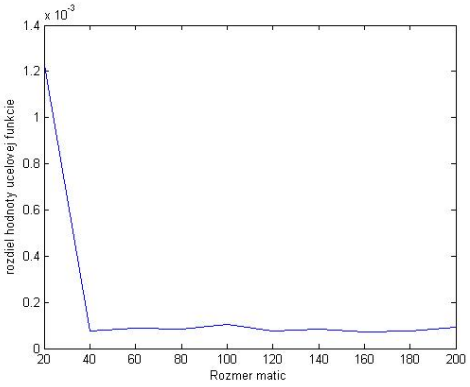
Obidva algoritmy sú schopné riešiť aj úlohy s rovnosťami sme implementovali v programovacom prostredí Matlab. Skôr, ako sa budeme venovať využitiu semidefinitného programovania, a teda aj možnostiam využitia našich algoritmov, radi by sme porovnali rýchlosť a presnosť nášho algoritmu z Kapitoly 2 so solverom SeDuMi [4]. Používali sme rozhranie CVX [1, 2], ktoré je určené pre program Matlab. Náš algoritmus sme testovali spôsobom, že sme vygenerovali náhodnú úlohu, vyriešili sme ju pomocou CVX a potom pomocou nášho algoritmu. Získali sme x_{cvx}, x_a a hodnoty trvania výpočtu $\text{čas}_{cvx}, \text{čas}_a$ porovnávali sme časy trvania výpočtov a rozdiel v hodnote účelovej funkcie. V tabuľke (1) uvádzame priemerné ukazovatele nazbieraných dát pre rôzne rozmery úloh.

$dim(x)$	$dim(H_i)$	$mean(\frac{\text{čas}_{cvx}}{\text{čas}_a})[10^{-4}]$	$mean(c^T(x_{cvx} - x_a))$
20	20	0.0004.355	0.001236877
20	40	0.0007156	0.00123687705
20	80	0.001995	0.0000858852
20	120	0.004745	0.0000752873
20	160	0.008570	0.0000726554
20	200	0.014274	0.0000911683
30	30	0.000049	0.0002499
30	60	0.000216	0.000055966
40	40	0.000024	0.0076

Tabuľka 1: Porovnanie s CVX



Obr. 6: Podiel časov pre rôzne $dim(H_i)$ a $dim(x) = 20$



Obr. 7: Priemerné hodnoty $c^T(x_{cvx} - x_a)$, pre rôzne $dim(H_i)$ a $dim(x) = 20$

Z tabuľky (1) je jasne vidieť, že náš algoritmus bol zakaždým pomalší, avšak časť tohto rozdielu môže byť spôsobená našou implementáciou, snažili sme sa aby bola pokiaľ možno čo najoptimálnejšia vrámci našich schopností. Ak by sme chceli aby náš algoritmus bol pokiaľ možno najviac konkurencie schopný, bolo by treba náš algoritmus optimálne implementovať priamo do jazyku C++ resp. Python. Avšak rozdiel časov bol až príliš veľký aby spočívala len v implementácií. Preto možno predpokladať, že náš algoritmus je vo všeobecnosti pomalší než implementované metódy v solvery SeDuMi. Veľkou výhodou nášho algoritmu ale je menšia pamäťová zložitosť a najmä jednoduchosť. Náš algoritmus je teoreticky implementovateľný aj v matematicky nie

tak zameraných jazykoch ako je Matlab. Zaujímavá možnosť je jeho implementácia v Microsoft Excel. Všetky kódy, ktoré sme používali sú uvedené v prílohe A.

4.2 Ekonómia

Využitie semidefinitného programovania v ekonómií je v dnešnej dobe značne rozsiahle, my sa budeme zaoberať problémom optimalizácie portfólia, konkrétne budeme vychádzať z článku Robust portfolio optimization via solution to the Hamilton–Jacobi–Bellman equation od S. Kilianovej a M. Trnovskej [18]. Riešime úlohu maximalizácie očakávaných výnosov pri minimalizácií rizika. Riešime úlohu tvaru:

$$\begin{aligned} \min_{\theta,\Sigma,t} \quad & t - \mu_0^T \theta \\ \text{s.t.} \quad & \theta^T \Sigma \theta \leq t, \\ & diag(\Sigma^{-1}) = d, \\ & \Sigma \succeq 0. \end{aligned} \tag{20}$$

Kde μ_0 je známy vektor, konštánt. Pomocou Vety (1.8) môžme ohraničenie $\theta^T \Sigma \theta \leq t$ previesť na ohraničenie tvaru:

$$\begin{aligned} \begin{pmatrix} \Sigma^{-1} & \theta \\ \theta^T & t \end{pmatrix} \succeq 0, \\ \Sigma^{-1} \succeq 0. \end{aligned} \tag{21}$$

Podmienka $\Sigma \succeq 0$ je ekvivalentná s $\Sigma^{-1} \succeq 0$. Preto môžme prejsť od pôvodnej matice Σ k hľadaniu matice Σ^{-1}. Dostávame úlohu tvaru:

$$\begin{aligned} \min_{\theta,\Sigma^{-1},t} \quad & t - \mu_0^T \theta \\ \text{s.t.} \quad & diag(\Sigma^{-1}) = d, \\ & \begin{pmatrix} \Sigma^{-1} & \theta \\ \theta^T & t \end{pmatrix} \succeq 0, \\ & \Sigma^{-1} \succeq 0. \end{aligned} \tag{22}$$

Pomocou Vety 3.4, vieme všetky ohraničenia, v tvare nerovností, úlohy (25) vyjadriť ako jednu maticovú nerovnosť a to

$$\left(\begin{array}{cc|c} \Sigma^{-1} & \theta & \\ \theta^T & t & 0 \\ \hline 0 & & \Sigma^{-1} \end{array}\right) \succeq 0. \tag{23}$$

Majme teda úlohu o rozdelení nášho portfólia pričom, máme vektor μ_0, ktorý odpovedá priemerným výnosom jednotlivých aktív. Označme $\omega = vect(\Sigma^{-1})$. Potom maticu Σ^{-1} môžme zapísať ako $\Sigma^{-1} = \mathcal{P}(\omega)$. Dostaneme

$$\left(\begin{array}{cc|c} \mathcal{P}(\omega) & \theta & \\ \theta^T & t & 0 \\ \hline 0 & & \mathcal{P}(\omega) \end{array}\right) \succeq 0. \tag{24}$$

Správnou voľbou matíc $H_1, \ldots, H_{\frac{n(n+1)}{2}}, Z_1, \ldots, Z_n, R$ vieme toto ohraničenie previesť do tvaru $\mathcal{A}(\omega, \theta, t) = \sum_{i=1}^{\frac{n(n+1)}{2}} \omega_i H_i + \sum_{j=1}^{n} \theta_j Z_j + tR$. Nakoniec dostávame úlohu tvaru:

$$\begin{aligned} \min_{\theta,\omega,t} \quad & t - \mu_0^T \theta \\ \text{s.t.} \quad & A\omega = d, \\ & \mathcal{A}(\theta, \omega, t) \succeq 0. \end{aligned} \tag{25}$$

Táto úloha je úlohou semidefinitného programovania, a je možné ju riešiť naším algoritmom z Kapitoly 2.

4.3 Fourierove rady

Jedným z možných využití nášho kvadratického algoritmu je riešenie úloh minimalizácie energie funkcie $f(x)$

$$\begin{aligned} \min_{f} \quad & \frac{1}{2\pi} \int_{-1}^{1} \left[f^2(x)dx + (f'(x))^2 \right] dx \\ \text{s.t.} \quad & f(x) \geq \varphi(x), \quad \forall x \in [-1, 1], \\ & f(x), \varphi(x) \quad \text{sú párne.} \end{aligned} \tag{26}$$

Jedná sa o funkcionálny problém, ktorý má využitie v spracovaní signálov. Tento problém pojednáva o minimalizácií energie funkcie f a jej derivácie f', pričom nesmie byť menšia než zvolená bariéra zvolená funkciou $\varphi(x)$. Podmienka na párnosť funkcií $f(x), \varphi(x)$ sa objasní o niečo neskôr. Táto úloha na prvý pohlaď nevyzerá ako úloha tvaru (KSP), avšak pomocou niekoľkých pomocných tvrdení ju prevedieme do tvaru

úlohy (KSP), a potom na túto úlohu už budeme môcť použiť náš algoritmus. Ako prvé pomocné tvrdenie uvedieme známu Parsevalovu vetu, ktorá pojednáva o súvislosti medzi hodnotou integrálu $\int_{-1}^{1} f^2(x)dx$ a koeficientami fourierovho radu, pre 2π-periodickú funkciu f. Dôkaz tejto vety môžno nájsť v knihe [20, str. 158]

Veta 4.1. *Nech f je funkcia integrovateľná na $\langle a, a+l\rangle$, kde je $l > 0$. Nech $a_0, a_1, \ldots, a_n \ldots b_1, b_2 \ldots, b_n \ldots$ sú Fourierove koeficienty funkcie f pre interval $\langle a, a+l\rangle$. Potom pre funkciu f platí Parsevalova rovnosť t.j.*

$$\frac{a_0^2}{2} + \sum_{n=1}^{\infty}(a_n^2 + b_n^2) = \frac{2}{l}\int_a^{a+l} f^2(x)dx.$$

Keďže funkcia f je párna, tak jej Fourierov rozvoj bude iba kosínusový rad t.j. $b_n = 0$, pre $\forall n \in \mathbb{N}$. Z Parsevalovej rovnosti vyplýva, že účelovú funkciu úlohy (26) možno prepísať do tvaru $\sum_{n=0}^{\infty}(1+n^2)a_n^2$. Budeme ďalej predpokladať, že funkcia f má konečný Fourierov rozvoj dĺžky K. Potom účelová funkcia bude mať tvar $\frac{a_0^2}{2} + \sum_{n=1}^{K}(1+n^2)a_n^2$. Komplikovanejší problém je prevedenie podmienky $f(x) \geq \varphi(x), \quad \forall x \in [-1,1]$. Budeme predpokladať, že φ bude mať tiež konečný Fourierov rozvoj rovnakej dĺžky ako f, rovný $\varphi(x) = \frac{\varphi_0}{2} + \sum_{n=1}^{K}\varphi_n \cos(n\pi x)$. Toto ohraničenie je ekvivalentné s $f(x) - \varphi(x) \geq 0, \quad \forall x \in [-1,1]$. Môžme využiť podmienku pre nezápornosť konečného Fourierovho radu. Táto podmienka bola dokázaná dvojicou J.W. McLean s H. J. Woerdeman v článku [21].

Veta 4.2 (McLean-Woerdeman)**.** *Konečný Fourierov rad $\sigma(x) = \sum_{n=-N+1}^{N-1}\sigma_k e^{ikx}$, je nezáporný pre $\forall x \in \mathbb{R}$, vtedy a len vtedy ak množina $\mathcal{M} \subset \mathcal{H}^N$ je neprázdna, kde :*

$$\mathcal{M} = \left\{ F \in \mathcal{H}^N | F \succeq 0, \sum_{p=n+1}^{N} F_{p,p-n} = \sigma_n, \quad pre \quad n = 0, 1, \ldots, N-1 \right\}.$$

Na základe Vety 4.2, môžme podmienku $f(x) \geq \varphi(x), \quad \forall x \in [-1,1]$ previesť na existenciu matice F takej, že bude platiť

$$\begin{aligned} &\sum_{p=n+1}^{N} F_{p,p-n} = \frac{a_n - \varphi_n}{2}, \quad \text{pre} \quad n = 0, \ldots, K-1 \\ &F \succeq 0. \end{aligned} \tag{27}$$

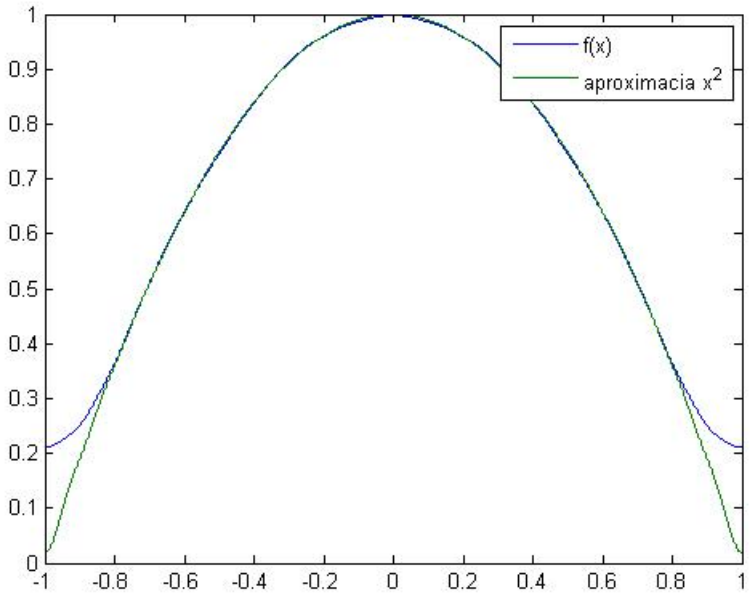
Dôvod, prečo máme požiadavku na to, aby funkcia f bola párna je ten, že ak by nebola, tak matica F z Vety 4.2 je vo všeobecnosti komplexná hermitovská a s týmto druhom matíc náš algoritmus zatiaľ nevie pracovať. Takže po zmene účelovej funkcie bude úloha (26) mať tvar:

$$\begin{aligned} \min_{a,F} \quad & \frac{a_0^2}{2} + \sum_{n=1}^{K} (1+n^2) a_n^2 \\ \text{s.t.} \quad & \sum_{p=n+1}^{K} F_{p,p-n} = \frac{a_n}{2} - \frac{\varphi_n}{2}, \quad \text{pre} \quad n = 0, \ldots, K-1 \\ & F \succeq 0. \end{aligned} \tag{28}$$

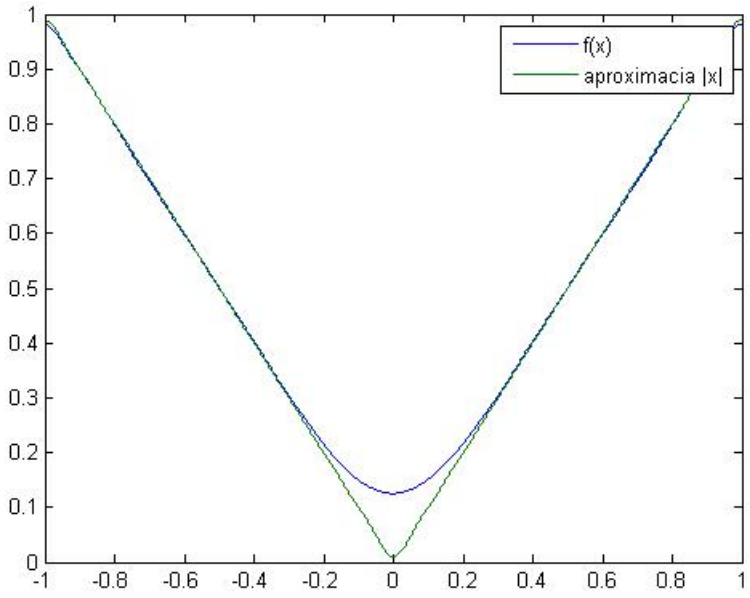
Túto úlohu už vieme veľmi jednoducho previesť na úlohu tvaru (KSP), využitím toho, že maticu F vieme zapísať pomocou maticovej funkcie $\mathcal{P}(f)$, kde $f = vect(F)$. Potom dostávame, že úlohu (26) vieme previesť na úlohu tvaru

$$\begin{aligned} \min_{a,f} \quad & \frac{a_0^2}{2} + \sum_{n=1}^{K} (1+n^2) a_n^2 \\ \text{s.t.} \quad & \sum_{p=1}^{K-n} f_{Kn+p} = \frac{a_n}{2} - \frac{\varphi_n}{2}, \quad \text{pre} \quad n = 0, \ldots, K-1 \\ & \mathcal{P}(f) \succeq 0. \end{aligned} \tag{29}$$

Táto úloha už je v tvare (KSP) a teda môžme použiť náš kvadratický algoritmus.Teraz uvedieme niekoľko grafických zobrazení výslednej funkcie $f(x)$ pre rôzne bariéry $\varphi(x)$.



Obr. 8: Riešenie $f(x)$ pre aproximáciu bariéry $1 - x^2$ na [-1,1] a $K = 20$



Obr. 9: Riešenie $f(x)$ pre aproximáciu bariéry $|x|$ na [-1,1] a $K = 20$

Jednou z ďalších možných aplikácií nášho algoritmu môže byť riešenie inverzného Wulffovho problému. Týmto problémom ako aj jeho riešením pomocou prevedenia na relaxovanú úlohu semidefinitného programovania sa zaoberali D. Ševčovič a M. Trnov-

ská v publikáciach [5, 6]. Jedna z úloh, ktoré boli odvodené v tejto publikácií, bola značne podobná úlohe (26), avšak účelová funkcia nebola konvexná, preto použitie nášho algoritmu na túto úlohu, by nemuselo dať správny výsledok.

4.4 Aplikácia pri odhade matíc

Jednou z možných oblastí využitia semidefinitného programovania je matematická štatistika, konkrétne pri vytváraní rôznych odhadov na kovariančnú maticu. Pri týchto odhadoch je veľmi dôležité, aby zohľadnili to, že kovariančná matica je vždy kladne semidefintná a symetrická. Jednou z ľahších úloh je hľadať kovariančnú maticu Σ takú, že má minimálnu Frobeniovu normu, pričom poznáme varianciu, teda diagonála matice Σ je rovná konštantnému vektoru $d > 0$. Táto úloha patrí pod konvexné programovanie, keďže Frobeniova norma je rýdzo konvexné zobrazenie. Riešime úlohu:

$$\begin{aligned} \min_{\Sigma} \quad & \|\Sigma\|_F^2 + \mu^T \Sigma \mu \\ \text{s.t.} \quad & diag(\Sigma) = d \\ & \Sigma \succeq 0. \end{aligned} \tag{30}$$

Kde vektor μ je vektor konštánt. Označme $\sigma = vect(\Sigma)$ potom matica Σ môže byť vyjadrená ako:

$$\begin{aligned} \Sigma &= \sum_{i=1}^{\frac{n(n+1)}{2}} \sigma_i M_i \\ &= \mathcal{P}(\sigma). \end{aligned}$$

Podmienku $\Sigma \succeq 0$ vieme teda previesť na $\mathcal{P}(\sigma) \succeq 0$, pre výraz $\|\Sigma\|_F^2$ platí

$$\begin{aligned} \|\Sigma\|_F^2 &= trace(\Sigma\Sigma) \\ &= trace(\mathcal{P}(\sigma)\mathcal{P}(\sigma)) \\ &= \sum_{i}^{\frac{n(n+1)}{2}} \sum_{j}^{\frac{n(n+1)}{2}} \sigma_i \sigma_j trace(M_i M_j). \end{aligned}$$

Z ortogonality matíc M_i platí $trace(M_i M_j) = 0$, pre $i \neq j$, preto $\|\Sigma\|_F^2 = \sum_{i=1}^{\frac{n(n+1)}{2}} \sigma_i^2 \|M_i\|_F^2$. Dostávame, že úlohu (30) môžme zapísať v tvare:

$$\begin{aligned} \min_{\sigma} \quad & \sum_{i}^{\frac{n(n+1)}{2}} \left[\sigma_i^2 \|M_i\|_F^2 + (\mu^T M_i \mu)\sigma_i\right] \\ \text{s.t.} \quad & \sigma_j = d_j, \quad \text{pre} \quad j = 1, \ldots, n, \\ & \mathcal{P}(\sigma) \succeq 0. \end{aligned} \tag{31}$$

Vidno, že táto úloha už je tvaru ako (KSP), a preto môžme na jej riešenie použiť náš algoritmus.

Záver

V našej práci sme vytvorili dva algoritmy na riešenie úloh s lineárnou resp. kvadratickou účelovou funkciou a spektrahedrálnou množinou prípustných riešení. Oba algoritmy vychádzajú z poznatkov z kapitoly(2), kde sme rozobrali aproximáciu spektrahedrálnej množiny pomocou polyhedrálnej množiny. Táto množina je generovaná pomocou vlastných vektorov, prislúchajúcich najmenšej vlastnej hodnote matice $\mathcal{A}(x)$. Pre oba problémy sme skonštruovali duálne úlohy a ukázali sme, využitím podmienky komplementarity, možnosť ako počítať aj duálnu úlohu popri počítaní primárnej úlohy. Ďalej sme v našej práce sme uviedli porovnanie nášho algoritmu pre úlohu (SDP) so solverom SeDuMi. Z tohto porovnania sme dostali, že vo všeobecnosti je náš algoritmus pomalší. Na určenie, ale presnej rýchlosti by bola potrebná optimálna implementácia nášho algoritmu. Výhodou našich algoritmov je ich jednoduchá štruktúra a možnosť ich implementovať aj v programovacích jazykoch, ktoré nie sú až tak veľmi matematicky zamerané ako Matlab. Na koniec našej práce sme použili naše algoritmy na riešenie problémov z rôznych oblastí matematiky.

Zoznam použitej literatúry

[1] M. Grant, S. Boyd. CVX: Matlab software for disciplined convex programming, version 2.0 beta. http://cvxr.com/cvx, September 2013.

[2] M. Grant, S. Boyd. Graph implementations for nonsmooth convex programs, Recent Advances in Learning and Control (a tribute to M. Vidyasagar), V. Blondel, S. Boyd, and H. Kimura, editors, pages 95-110, Lecture Notes in Control and Information Sciences, Springer, 2008. http://stanford.edu/~boyd/graph_dcp.html.

[3] M. Hamala, M. Trnovská: Nelineárne programovanie, EPOS, Bratislava, 2013.

[4] J. F. Sturm, Using SeDuMi 1.02, A Matlab toolbox for optimization over symmetric cones, Optimization Methods and Software 11 (1999), 625–653.

[5] D. Ševčovič, M. Trnovská: Solution to the Inverse Wulff Problem by Means of the Enhanced Semidefinite Relaxation Method, Journal of Inverse and Ill-posed Problems 23(3) 2015, 263-285.

[6] D. Ševčovič, M. Trnovská:Application of the Enhanced Semidefinite Relaxation Method to Construction of the Optimal Anisotropy Function, IAENG International Journal of Applied Mathematics 45(3) (2015), 227-234.

[7] K.C. Toh, M.J. Todd, R.H. Tutuncu: SDPT3 — a Matlab software package for semidefinite programming, Optimization Methods and Software, 11 (1999), pp. 545–581.

[8] R.H Tutuncu, K.C. Toh, M.J. Todd: Solving semidefinite-quadratic-linear programs using SDPT3, Mathematical Programming Ser. B, 95 (2003), pp. 189–217.

[9] P. Zlatoš: Lineárna algebra a geometria, Marenčin PT, Bratislava, 2011.

[10] J. Dupačová, P. Lachout: Úvod do optimalizace, MATFYZPRESS, Praha 2011.

[11] K. K. Sivaramakrishnan: Linear programming approaches to semidefinite programming problems, Rensselaer Polytechnic Institute, New York, 2002.

[12] S. Boyd, L. Vandenberghe: Convex Optimization, Cambridge University Press, New York, 2004.

[13] C. Helmberg: Semidefinite Programming for Combinatorial Optimization, ZIB-Report ZR-00-34, Konrad-Zuse-Zentrum Berlin, October 2000.

[14] F. Alizadeh, J.P.A. Haeberly, M.L. Overton: Complementarity and Nondegeneracy in Semidefinite Programming, Mathematical Programming, 77(1997), pp. 111-128.

[15] A. Y. Galenko: Modeling stochastic dependencies and their impact on optimal decision making. ProQuest, 2008.

[16] A. Galenko: "Simulating cointegrated time series."Winter Simulation Conference, 2009.

[17] J. G. Young: Implementation of a cutting plane method for semidefinite programming. Diss. New Mexico Institute of Mining and Technology, 2004.

[18] S. Kilianová, M. Trnovská: Robust portfolio optimization via solution to the Hamilton–Jacobi–Bellman equation. International Journal of Computer Mathematics, Volume 93, Issue 5, 2016.

[19] R.A. Horn, Ch.R. Johnson: Matrix analysis, Cambridge University Press, Cambridge 1985.

[20] I. Kluvánek, L. Mišík, M. Švec: Matematika II, Slovenské vydavateľstvo technickej literatúry, Bratislava, 1961.

[21] J.W. McLean, H.J. Woerdeman. Spectral factorizations and sums of squares representations via semidefinite programming. SIAM J. Matrix Anal. Appl., 23(3):646–655, 2001.

[22] M. Trnovská: Strong Duality Conditions in Semidefinite Programming, Journal of Electrical Engineering, Vol 56, No 12/s, 2005.

Príloha A

Imlementácia algoritmu na riešenie úlohy (SDP)

```
function[vysledok,hodnota,j,x_vonkajsie,cas]=linearna_metoda(c,x0,M,Aeq,beq,opt)
% popis funkcie
% popis vstupnych velicin opt je cell zadavany v tvare {maxiter krok eps1 eps2 eps3}
% popi vytupu

numvarargs = length(opt);
optargs = {1000 1 10^-5 10^-8 10^-3};
optargs(1:numvarargs) = opt;
[maxiter, sigma, eps1, eps2, eps3] = optargs{:};
function[ohr1]=ohranicenia(M,z,h1)
     ohr1=zeros(1,h1);
     parfor i1=1:h1
         ohr1(1,i1)=z'*M(:,:,i1)*z;
     end
end

function[H2]=matica(M,x0,h2,pociatok)
     if nargin <=3
       pociatok=1;
     end
     x0=[1;x0];
     H2=0;
     for i2=pociatok:h2
         H2=H2+x0(i2,1)*M(:,:,i2);
     end
end
x=x0;
exit_flag=0;
vysledok=[];
hodnota=[];
cas=[];
x_vonkajsie=[];
hodnoty_ucelovej_funkcie=[];
j=1;
[~,n,h]=size(M);
konstanta=0;
if  ~isempty(Aeq)
    x_partikularne=Aeq\beq;
    pomocna_premenna=sum(abs(Aeq*x_partikularne-beq)>10^-5);
    if pomocna_premenna~=0
        warning('sustava rovnic Ax=b nema riesenie')
        return
    end
    x_homogenne=null(Aeq);
    k=h;
    [~,h]=size(x_homogenne);
    [~,p]=chol(matica(M,x_partikularne,h));
    if h==0
        if p==0
          vysledok=x_partikularne;
          hodnota=c*x_partikularne;
          return
        else
          x=[];
          warning('mnozina pripustnych rieseni je prazdna')
        end
    else
        M_stare=M;
        c_stare=c;
        c=[];
        M=[];
        x_povodna_uloha=@(x) x_partikularne+ x_homogenne*x;
        M=matica(M_stare,x_partikularne,k);
        c=c_stare*x_homogenne;
        konstanta=c_stare*x_partikularne;
        for i=1:h
            M(:,:,i+1)=matica(M_stare,x_homogenne(:,i),k,2);
        end
        h=h+1;
    end
    if ~isempty(A)
        A_stare=A;
        b_stare=b;
        A=[];
        b=[];
        for i=1:h
            A(:,i)=A_stare*x_homogenne(:,i);
        end
        b=b_stare-A_stare*x_partikularne;
    end
end



tic
if isempty(x)
   x=rand(h-1,1);
end
x_vonkajsie=x;
hodnoty_ucelovej_funkcie=c*x;
ohran=[];
G=matica(M,ones(h-1,1),h)-M(:,:,1);
I=eye(h-1);
options=optimset('LargeScale','off','Display', 'off');
while (or(j<maxiter+1,maxiter==0))
    V=[];
    H=matica(M,x,h);
    [Vp,lambdy]=eig(H);
    minimalna_lambda=min(diag(lambdy));
    pom=1:n;
    pom=pom(diag(lambdy)<=0);
    index_min_lambda=find((diag(lambdy)-minimalna_lambda)<10^-5);
    Z=Vp(:,index_min_lambda);
    if minimalna_lambda>0
      Vp=Vp(:,index_min_lambda);
    else
      Vp=Vp(:,pom);
    end
   [~,index_min_lambda]=size(Vp);
    V=repmat(Vp,1,100) +sigma*randn(n,index_min_lambda*100);
    V=[V,Vp];
    %%%%tretie zastavovacie kriterium%%%%%
    riesenie_dualnej_ulohy=zeros(n);
    [~,k]=size(Z);
```

```
    A_dualna_uloha=[];
        for p=1:h
            d=0;
            parfor i=1:k
                d(1,i)=Z(:,i)'*M(:,:,p)*Z(:,i);
            end
          if p==1
            c_dualna_uloha=d;
          else
            A_dualna_uloha=[A_dualna_uloha;d];
          end
        end
       if k==1
            y=sum(c)/(Z'*(G)*Z);
            hodnota_dualnej_ulohy=c_dualna_uloha*y;
            riesenie_dualnej_ulohy=y*Z*Z';
       else
            [y,hodnota_dualnej_ulohy,ef]=linprog(c_dualna_uloha,[],[],...
              A_dualna_uloha,c',zeros(k,1),[],[],options);
             if ef==1
               parfor i=1:k
                 riesenie_dualnej_ulohy=riesenie_dualnej_ulohy+y(i)*...
                     Z(:,i)*Z(:,i)';
               end
             else
                y=[];
             end
       end
        %tretie zastavovacie pravidlo
    if ~isempty(y)
        if abs(c*x+hodnota_dualnej_ulohy)+abs(minimalna_lambda)<eps3
           disp('Metoda zastala lebo p-d medzera je mensia nez eps3')
           exit_flag=1;
           break
        end
    end
    [~,s]=size(V);
    for i=1:s
        ohr=ohranicenia(M,V(:,i),h);
        ohran=[ohran;ohr];
    end
    [x,~,exitflag]=linprog(c,[-ohran(:,2:end);A],[ohran(:,1);b],[],[],...
        [],[],[],options);
    hodnoty_ucelovej_funkcie=[hodnoty_ucelovej_funkcie,c*x];
    x_vonkajsie=[x_vonkajsie,x];
    %%%%%%prve zastavovacie kriterium %%%%
    if (abs(hodnoty_ucelovej_funkcie(:,end)-hodnoty_ucelovej_funkcie(:,end-1))<eps1) & (j>3)
      pomocna_premenna=  pomocna_premenna+1;
    else
        pomocna_premenna=0;
    end
    if pomocna_premenna>5
        disp('Metoda zastala lebo hodnota ucelovej funkcie sa uz nezmensovala')
        exit_flag=1;
        break
    end
    %%%%%druhe zastavovacie kriterium%%%%
    if norm(x_vonkajsie(:,end-1)-x)<eps2
       disp('Metoda zastala lebo sa pohybujeme uz v malom okoli jedneho bodu')
       exit_flag=1;
        break
    end
 j=j+1;
end
if ~isempty(Aeq)
  vysledok= x_povodna_uloha(x);
  hodnota= c_stare*vysledok;
  hodnoty_ucelovej_funkcie=hodnoty_ucelovej_funkcie+konstanta;
else
  hodnota=c*x;
  vysledok=x;
end
cas=toc;
end
```

Imlementácia algoritmu na riešenie úlohy (KSP)

```
function[vysledok,hodnota,eF,j,x_vonkajsie,Y,cas]=kvadraticka_metoda_v3...
    (K,c,x0,M,Aeq,beq,opt)
%metoda pre ulohu s kvadratickou ucelovou funkciou
%K musi byt kladne definitna matica, c riadkovy vektor, M je matica
%rozmerov n x n x m+1 kde su ulozene v kazdej vrstve matice Hi rozmerov nxn
%maxiter je pocet maximalnych iteracii
% eps3 a eps4 su presnosti pre zastavovacie kriteria

%tvar riesenej ulohy: min 0.5xT*G*x+cT*x
%                     s.t. A(x,M)>=0
eF=0;
vysledok=[];
hodnota=[];
j=[];
x_vonkajsie=[];
cas=[];
Y=[];

numvarargs = length(opt);
%nastavenie tolerancii
optargs = {1000 1 10^-5 10^-5 10^-2};
optargs(1:numvarargs) = opt;
[maxiter, sigma, eps1, eps2, eps3] = optargs{:};


    function[ohr1]=ohranicenia(M,z,h1)
     ohr1=zeros(1,h1);
     [~,l]=size(z);
     if l==1
        parfor i1=1:h1
          ohr1(1,i1)=z'*M(:,:,i1)*z;
        end
     else
         parfor i1=1:h1
           ohr1(1,i1)=trace(M(:,:,i1)*z);
         end

     end
```

```
    end

    function[H2]=matica(M,x0,h2,pociatok)
        if nargin <=3
            pociatok=1;
        end
        x0=[1;x0];
        H2=0;
      for i2=pociatok:h2
          H2=H2+x0(i2,1)*M(:,:,i2);
     end
    end

tic;
ohran=[];
[~,n,h]=size(M);
konstanta=0;

if  ~isempty(Aeq)
    x_partikularne=Aeq\beq;
    pomocna_premenna=sum(abs(Aeq*x_partikularne-beq)>10^-5);
    if pomocna_premenna~=0
        warning('sustava rovnic Ax=b nema riesenie')
        return
    end
    pomocna_premenna=[];
    x_homogenne=null(Aeq);
    k=h;
    [~,h]=size(x_homogenne);
    [~,p]=chol(matica(M,x_partikularne,h));
    if h==0
        if p==0
          vysledok=x_partikularne;
          hodnota=c*x_partikularne;
          return
        else
          x=[];
          warning('mnozina pripustnych rieseni je prazdna')
        end
    else %uprava ohraniceni a ucelovej funkcie
        M_stare=M;
        M=[];
        konstanta=x_partikularne'*K*x_partikularne/2+c*x_partikularne;
        M=matica(M_stare,x_partikularne,k);
        c=(c+x_partikularne'*K)*x_homogenne;
        K=x_homogenne'*K*x_homogenne;
        x_povodna_uloha=@(x,pomocna_premenna) repmat(x_partikularne,1,...
            pomocna_premenna)+ x_homogenne*x;
        for i=1:h
            M(:,:,i+1)=matica(M_stare,x_homogenne(:,i),k,2);
        end
        h=h+1;
    end
    if ~isempty(x0)
        x0=x_homogenne\x0;
    end
end




[ST,D]=eig(K);
D=diag(D);
indexy=1:h-1;
pocet_nvlh=sum(abs(D)<10^-9);
indexy=[indexy(abs(D)<10^-9),indexy(abs(D)>=10^-9)];
overenie pripustnosti K
if sum(D<0)~=0
    warning('ucelova funkcia musi byt konvexna')
    return
end
D=D(indexy,1);
ST=ST(:,indexy);
D=diag(D);
M_stare=M;
M=[];
%zmena ohraniceni ulohy
M=M_stare(:,:,1);
for i=2:h
    M(:,:,i)=matica(M_stare,ST(:,i-1),h,2);
end
K_stare=K;
K=D;
c_stare=c;
c=c*ST;
x_povodne_suradnice= @(x) ST*x;
hodnota_povodnej_ulohy= @(x) x'*K*x/2+c*x+ konstanta;

j=0;
options = optimset('Algorithm','interior-point-convex','Display','off',...
    'TolX',10^-17);
%prva iteracia zistovanie ci optimum nie je vo vnutri mnoziny prip ries
x_hat=D\c';
if  isempty(Aeq)
  pomocna_premenna=sum(abs(D*x_hat-c')>10^-10);
else
    pom=Aeq*x_povodna_uloha(x_hat,1)-beq;
    pomocna_premenna=sum(abs(D*x_hat-c')>10^-10)+sum(abs(pom)>10^-10);
end
  if pomocna_premenna==0
    H=matica(M,x_hat,h);
    [~,p]=chol(H);
    if p==0
      vysledok=x_povodne_suradnice(x_hat);
      hodnota=hodnota_povodnej_ulohy(x_hat);
      if ~isempty(Aeq)
        vysledok= x_povodna_uloha(vysledok,1);
      end
      eF=1;
      disp('metoda zastala lebo riesenie je vo vnutri mnoziny')
      return
    end
  end
pomocna_premenna=0;


if isempty(x0)
    if sum(isnan(x_hat))==0
    x0=x_hat;
    else
        x0=-10+20*rand(h-1,1);
    end
end
x_vonkajsie=x0;
hodnoty_ucelovej_funkcie=hodnota_povodnej_ulohy(x0);
x=x0;
konstanta2=sum((c(1,pocet_nvlh+1:end)'.^2)./diag(2*D(pocet_nvlh+1:end,...
    pocet_nvlh+1:end)));
I=eye(n);
```

```
while (j<=maxiter)
    V=[];
    G=0;
    H=matica(M,x,h);
    [Vp,lambdy]=eig(H,I);
    minimalna_lambda=min(diag(lambdy));
    pom=1:n;
    index_min_lambda=find(diag(lambdy)-minimalna_lambda<10^-4);

    pom=pom(diag(lambdy)<=0);

    Z=Vp(:,index_min_lambda);

    if minimalna_lambda>0
      Vp=Vp(:,index_min_lambda);
    else
      Vp=Vp(:,pom);
    end
    [~,index_min_lambda]=size(Vp);
    V=repmat(Vp,1,100) +sigma*randn(n,index_min_lambda*100);
    V=[V,Vp];
    [~,k]=size(Z);
    G_dualna_uloha=0;
    Aeq_dualna_uloha=[];
    beq_dualna_uloha=c(1,1:pocet_nvlh)';
    for p=1:h
        d=0;
        for i=1:k
            d(i,1)=Z(:,i)'*M(:,:,p)*Z(:,i);
        end
    if p==1
        c_dualna_uloha=d';
    elseif (p>pocet_nvlh+1)
        G_dualna_uloha=G_dualna_uloha+d*d'./D(p-1,p-1);
        c_dualna_uloha=c_dualna_uloha-d'*c(1,p-1)/D(p-1,p-1);
%         konstanta=konstanta+(c(1,p-1)^2)/D(p-1,p-1);
    else
        Aeq_dualna_uloha(p-1,:)=d';
    end
    end
    [y,hodnota_dualnej_ulohy]=quadprog(G_dualna_uloha,c_dualna_uloha,[],...
        [],Aeq_dualna_uloha,beq_dualna_uloha,zeros(k,1),[],[],options);
    hodnota_dualnej_ulohy=hodnota_dualnej_ulohy+konstanta2-konstanta;
    if isempty(y)&& (k==1)
        y=max(0,sum(beq_dualna_uloha)/sum(Aeq_dualna_uloha));
        hodnota_dualnej_ulohy=0.5*G_dualna_uloha*y*y+c_dualna_uloha*y...
            +konstanta2-konstanta;
    end
%riesenie dualnej ulohy
  Y=0;
  for i=1:k
    Y=Y+y(i,1)*Z(:,i)*Z(:,i)';
  end
% tretie zastavovacie pravidlo
  if abs(hodnoty_ucelovej_funkcie(1,end)+hodnota_dualnej_ulohy)...
          +abs(minimalna_lambda)<eps3
     disp('Metoda zastala lebo primarno-dualna medzera je mensia nez eps4')
     eF=1;
    break
  end
  [~,s]=size(V);
  for i=1:s %parfor
    ohr=ohranicenia(M,V(:,i),h);
    ohran=[ohran;ohr];
  end
  x=quadprog(K,c,-ohran(:,2:end),ohran(:,1),[],[],[],[],[],options);
  x_vonkajsie=[x_vonkajsie,x_povodne_suradnice(x)];
  hodnoty_ucelovej_funkcie=[hodnoty_ucelovej_funkcie,...
      hodnota_povodnej_ulohy(x)];
  j=j+1;
%prve zastavovacie kriterium
  if abs(hodnoty_ucelovej_funkcie(:,j)-hodnoty_ucelovej_funkcie(:,j+1))...
          <eps1
    pomocna_premenna=  pomocna_premenna+1;
  else
    pomocna_premenna=0;
  end

  if pomocna_premenna>5
    disp('Metoda zastala lebo hodnota ucelovej funkcie sa uz nezmensovala')
    eF=1;
    break
  end

%druhe zastavovacie kriterium
  if norm(x_vonkajsie(:,end-1)-x_vonkajsie(:,end))<eps2
    disp('Metoda zastala lebo sa pohybujeme uz v malom okoli jedneho bodu')
    eF=1;
    break
  end
  ohran=ohran(abs(ohran(:,2:end)*x+ohran(:,1))<10^-4,:);
end
vysledok=x_povodne_suradnice(x);
hodnota=hodnota_povodnej_ulohy(x);
if ~isempty(Aeq)
  vysledok= x_povodna_uloha(vysledok,1);
  x_vonkajsie=x_povodna_uloha(x_vonkajsie,j+1);
end
cas=toc;
end
```