SLOVENSKÁ TECHNICKÁ UNIVERZITA V BRATISLAVE

FAKULTA CHEMICKEJ A POTRAVINÁRSKEJ TECHNOLÓGIE

Evidenčné číslo: FCHPT-5414-76933

Metódy dvojúrovňovej optimalizácie

DIPLOMOVÁ PRÁCA

2019 **Bc. Daniel Boroš**

SLOVENSKÁ TECHNICKÁ UNIVERZITA V BRATISLAVE

FAKULTA CHEMICKEJ A POTRAVINÁRSKEJ TECHNOLÓGIE

Evidenčné číslo: FCHPT-5414-76933

Metódy dvojúrovňovej optimalizácie

DIPLOMOVÁ PRÁCA

Študijný program:	automatizácia a informatizácia v chémii a potravinárstve
Študijný odbor:	5.2.14. Automatizácia
Školiace pracovisko:	Ústav informatizácie, automatizácie a matematiky
Vedúci práce:	doc. Ing. Radoslav Paulen, PhD.

2019 **Bc. Daniel Boroš**

Slovenská technická univerzita v Bratislave
Oddelenie informatizácie a riadenia procesov

Fakulta chemickej a potravinárskej technológie
Akademický rok: 2018/2019
Evidenčné číslo: FCHPT-5414-76933

STU
FCHPT

ZADANIE DIPLOMOVEJ PRÁCE

Študent: **Bc. Daniel Boroš**
ID študenta: 76933
Študijný program: automatizácia a informatizácia v chémii a potravinárstve
Študijný odbor: 5.2.14. automatizácia
Vedúci práce: doc. Ing. Radoslav Paulen, PhD.

Názov práce: **Metódy dvojúrovňovej optimalizácie**

Jazyk, v ktorom sa práca vypracuje: slovenský jazyk

Špecifikácia zadania:

Dvojúrovňová optimalizácia predstavuje dôležitú triedu optimalizačných problémov širokovyužiteľných pre potreby moderného riadenia procesov. Vo všeobecnosti je však tento problém náročný na riešenie. Cieľom projektu je skúmať výpočtové metódy pre špeciálny prípad dvojúrovňovej optimalizácie, tzv. min-max optimalizácie. Vyvinuté metódy budú aplikované na príkladoch návrhu optimálnych experimentov.

Zoznam odbornej literatúry:

1. Mitsos, A. & Tsoukalas, A. J Glob Optim (2015) 61: 1. https://doi.org/10.1007/s10898-014-0146-6

Riešenie zadania práce od: 11. 02. 2019
Dátum odovzdania práce: 12. 05. 2019

Bc. Daniel Boroš
študent

doc. Ing. Michal Kvasnica, PhD.
vedúci pracoviska

prof. Ing. Miroslav Fikar, DrSc.
garant študijného programu

Poďakovanie

Ďakujem môjmu vedúcemu diplomovej práce doc. Ing. Radoslavovi Paulenovi, PhD. za ochotu, pomoc, odborné vedenie a cenné rady. Rovnako ďakujem všetkým pracovníkom katedry informatizácie a riadenia procesov za ich ochotu a pomoc.

Abstract

Aim of this master thesis is to provide the reader with theoretical basis and resources for solving certain types of bi-level optimization problems as, for example, the problem of optimal design of experiment. In this thesis are implemented methods based on Karush-Kuhn-Tucker optimality conditions, Blankenship-Falk's and Mitsos-Tsoukalas' algorithms. Functions used to solve these problems were created in MATLAB environment with the use of global solver BARON and are available as a toolbox for MATLAB. Optimal design of experiments is achieved by appropriate excitation of system using available inputs. Prepared functions are able to design optimal experiment based on A-design criteria for nonlinear static and discrete-time models. For solving optimal design of experiments problems, two strategies are prepared. First one is based on Karush-Kuhn-Tucker optimality conditions and is suitable mostly for linear systems. Second strategy employs Mitsos-Tsoukalas algorithms, which is able to solve even problems based on nonlinear models. Thesis also deals with solving more generalized types of bi-level problems for which three strategies are available, application of Karush-Kuhn-Tucker optimality conditions, Blankenship-Falk's algorithm and Mitsos-Tsoukalas' algorithm. In conclusion review of used methods is given.

keywords: bi-level optimization problems, Blankenship-Falks algorithm, Mitsos-Tsoukalas algorithm, optimal design of experiments

Abstrakt

Cieľom diplomovej práce je poskytnúť čitateľom teoretický základ a prostriedky na riešenie konkrétnych typov dvojúrovňových optimalizačných problémov ako je napríklad problém optimálneho návrhu experimentu. V tejto práci sú implementované metódy založené na Karush-Kuhn-Tuckerových podmienkach optimality, Blankenship-Falkovom a Mitsos-Tsoukalasovom algoritme. Funkcie použité na riešenie týchto problémov boli vytvorené v prostredí MATLAB s použitím globálneho riešiteľa BARON a sú voľne dostupné ako balík pre MATLAB. Optimálny dizajn experimentu je dosiahnutý vhodným vybudením systému použitím dostupných akčných zásahov. Pripravené funkcie sú schopné navrhnúť optimálny dizajn experimentu použitím A-dizajn kritéria aj pre nelineárne diskrétne a statické modely. Na riešenie problému optimálneho dizajnu experimentu sú pripravené dve stratégie. Prvá je založená na Karush-Kuhn-Tuckerových podmienkach optimality a je vhodná prevažne pri lineárnych systémoch. Druhá stratégia používa Mitsos-Tsoukalasov algoritmus, ktorý dokáže riešiť aj problémy založené na nelineárnych modeloch. Práca sa zaoberá aj riešením všeobecnejších typov dvojúrovňových problémov na ktoré sú dostupné tri stratégie, aplikácia Karush-Kuhn-Tuckerových podmienok optimality, Blankenship-Falkov algoritmus a Mitsos-Tsoukalasov algoritmus. Na záver je uvedené vyhodnotenie jednotlivých metód.

kľúčové slová: dvojúrovňové optimalizačné problémy, Blankenship-Falkov algoritmus, Mitsos-Tsoukalasov algoritmus, optimálny návrh experimentov

Obsah

Zoznam obrázkov

Zoznam tabuliek

Kapitola 1

Úvod

Dvojúrovňové optimalizačné problémy boli prvýkrát predstavené v roku 1934 H.F. v. Stackelbergom [10], [11]. Do dnes sa v ekonomickej teórii pracuje aj so špeciálnym typom dvojúrovňových problémov tzv. Stackelbergovou hrou. Výraznejší rozvoj metód riešenia týchto problémov začal v 80. rokoch 20. storočia hnaný najmä matematikmi, ekonómmi a inžiniermi, keďže problémy tohto typu sa často vyskytujú v ekonómii, ale aj v priemysle.

Výskum metód, schopných dosiahnuť globálne optimálne výsledky je dôležitý, pretože nám to umožňuje efektívne riešiť problémy ako manažment výnosov [3], manažment nebezpečných materiálov [4], problémy návrhu cestných sietí [6], [7], [2] a aj problémy, v ktorých sa nachádzajú fyzikálne alebo chemické rovnováhy.

Jedným zo spôsobov, ktorým sa dajú riešiť dvojúrovňové optimalizačné problémy je redukcia problému na jednu úroveň. Redukcia je možná pomocou transformácie problému nižšej úrovne Karush-Kuhn-Tuckerovými podmienkami optimality. Tento postup je vhodný najmä vtedy, keď problém nižšej úrovne je konvexný. Tejto metóde budú venované samostatné sekcie.

Na riešenie dvojúrovňových problémov bolo navrhnutých niekoľko gradientových metód, kde smer zostupu vedie k minimalizácii účelovej funkcie problému vyššej úrovne. Súčasne je nutné udržiavať nový bod prípustným. Nový bod je považovaný za prípustný, iba vtedy ak je optimálny z pohľadu problému nižšej úrovne. Za účelom splnenia týchto požiadaviek na nový bod vedci skúmali spôsoby ako aproximovať gradient funkcie vyššej úrovne [5] a aj pridávali pomocné [9], [12] optimalizačné úlohy, ktoré spresňujú určovanie smeru zostupu.

Ďalším zo spôsobov riešenia dvojúrovňových optimalizačných problémov je vnorený prístup. Pri vnorenom prístupe je dvojúrovňový problém rozdelený na dve časti. Na problém nižšej úrovne, ktorý je nutné vyriešiť globálne a problém vyššej úrovne na

ktorý stačí aj lokálny riešiteľ. Metóda sa nazýva vnorená, pretože problém nižšej úrovne je vnorený do vyššej úrovne, čo znamená, že keď riešiteľ vyššej úrovne nájde potenciálne optimálny bod, tak pre tento bod musí byť vyriešený celý problém nižšej úrovne aby sa zistilo, či je daný bod prípustný. Táto metóda je veľmi efektívna na malé optimalizačné problémy, ale s počtom optimalizovaných premenných jej výpočtová náročnosť výrazne rastie.

Jednou z alternatív sú metódy založené na pridávaní ohraničení do problému vyššej úrovne. S rastúcim počtom ohraničení zmenšujú prípustný region, v ktorom sa nachádza optimálne riešenie, až kým výsledok nieje dosiahnutý. V tejto práci sa budeme zaoberať dvoma metódami, ktoré patria do tejto skupiny, Blankenship-Falkovým algoritmom [1] a jeho upravenou verziou Mitsos-Tsoukalasovým algoritmom [8, 13].

Dokonca aj v dnešnej dobe s výpočtovým výkonom, ktorý bol v minulosti len veľmi ťažko predstaviteľný, sú dvojúrovňové problémy s väčším počtom premenných často riešiteľné len veľmi ťažko alebo dokonca vôbec.

V tejto práci sa budeme venovať teórii, implementácii a niekoľkým príkladom aplikácie vybraných metód riešenia dvojúrovňových optimalizačných problémov v programovacom prostredí MATLAB. Výsledkom tejto práce je vytvorenie funkcií používajúcich optimalizačný riešiteľ BARON, ktoré budú schopné riešiť vybrané typy týchto problémov. Vytvorené funkcie sú dostupné ako balík pre programovacie prostredie MATLAB. Ako bude v práci ukázané ďalej, je častým javom, že algoritmy, ktoré riešia dvojúrovňové optimalizačné problémy si vyžadujú dodatočnú transformáciu riešeného problému na ďalšie podproblémy. Pri použití poskytnutých funkcií, ale transformácia problému nieje nutná keďže to funkcie spravia automaticky.

V druhej kapitole sú predstavené optimalizačné problémy, ktoré sa snažíme riešiť a jednotlivé použité metódy na ich riešenie. V tretej kapitole sú popísané možnosti funkcií vytvorené v MATLAB-e za účelom riešenia týchto problémov. Štvrtá kapitola je venovaná príkladom a ich riešeniam použitím poskytnutých funkcií.

Kapitola 2

Dvojúrovňové optimalizačné problémy a spôsoby ich riešenia

2.1 Dvojúrovňové optimalizačné problémy

Dvojúrovňová optimalizácia je špeciálny prípad optimalizácie, kde jeden optimalizačný problém je argumentom druhého. Vrchná časť optimalizačného problému sa štandardne nazýva problém vyššej úrovne a na spodnú časť sa odkazujeme ako na problém nižšej úrovne. Tento typ problému sa dá previesť na všeobecný semi-infinitný problém (z angl. general semi-infinite problem, ďalej GSIP).

V rámci tejto diplomovej práce a vytvoreného balíka funkcií pre MATLAB sa zaoberáme problémami v nasledovnom tvare:

$$\min_{\boldsymbol{x}} \; f(\boldsymbol{x},\boldsymbol{y}) \tag{2.1a}$$

$$\text{s.t. } \arg \max_{\boldsymbol{y}} g_u(\boldsymbol{x},\boldsymbol{y}) \tag{2.1b}$$

$$s.t. \; \boldsymbol{g_l}(\boldsymbol{x},\boldsymbol{y}) \leq 0 \tag{2.1c}$$

kde

- $\boldsymbol{x}$ sú premenné vyššej úrovne.
- $\boldsymbol{y}$ sú premenné nižšej úrovne.
- $f(\boldsymbol{x},\boldsymbol{y})$ je účelová funkcia vyššie úrovne.
- $g_u(\boldsymbol{x},\boldsymbol{y})$ je účelová funkcia nižšej úrovne.
- $\boldsymbol{g_l}(\boldsymbol{x},\boldsymbol{y})$ je vektor ohraničení nižšej úrovne v tvare nerovnosti.

Pri riešení dvojúrovňových optimalizačných problémov uvažujeme spojité premenné a spojité, diferencovateľné účelové funkcie a funkcie ohraničení.

2.2 Optimálny návrh experimentu

Optimálny návrh experimentu sa zaoberá spôsobmi ako čo najlepšie odhadnúť neznáme parametre v danom modeli. Výsledkom optimálneho návrhu experimentu je región spoľahlivosti (ďalej RS).

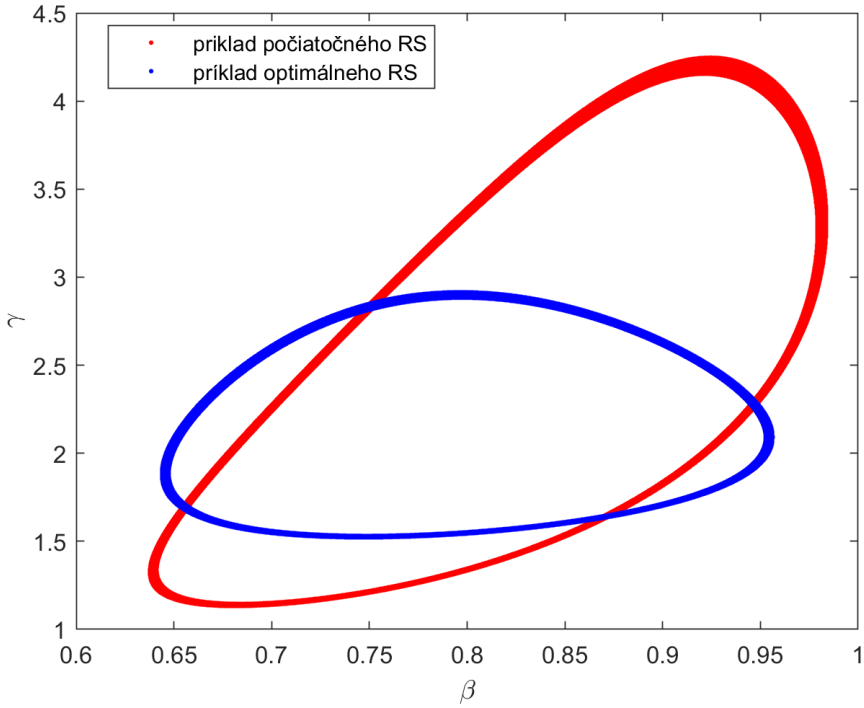
2.2.1 Región spoľahlivosti

Predstavuje množinu všetkých hodnôt parametrov, pri ktorých výsledky získané meraním a výsledky nadobudnuté použitím modelu s odhadovanými parametrami sa líšia maximálne o nami zvolenú hodnotu, ktorá zohľadňuje štatistickú významnosť odhadu. Aby dvojica parametrov patrila do RS musí spĺňať nasledovnú podmienku:

$$\sum_{k=1}^{n} \frac{(\boldsymbol{y}(\boldsymbol{\pi}_j, \boldsymbol{u}_k) - \boldsymbol{y_m}(\boldsymbol{u}_k))^2}{\sigma^2} \leq \chi^2_{\alpha, n_p} \tag{2.2}$$

kde

- $\boldsymbol{\pi}_j$ je bod parametrov.
- $\boldsymbol{u}_k$ je vektor akčných zásahov.
- $\boldsymbol{y}(\boldsymbol{\pi}_j, \boldsymbol{u}_k)$ odhady hodnôt výstupu v jednotlivých krokoch.
- $\boldsymbol{y_m}(\boldsymbol{u}_k)$ merania hodnôt výstupu v jednotlivých krokoch
- σ je štandardná odchýlka šumu merania.
- χ^2_{α, n_p} je horný α kvantil χ^2 rozdelenia pravdepodobnosti s n_p stupňami voľnosti.
- n je počet meraní experimentu .



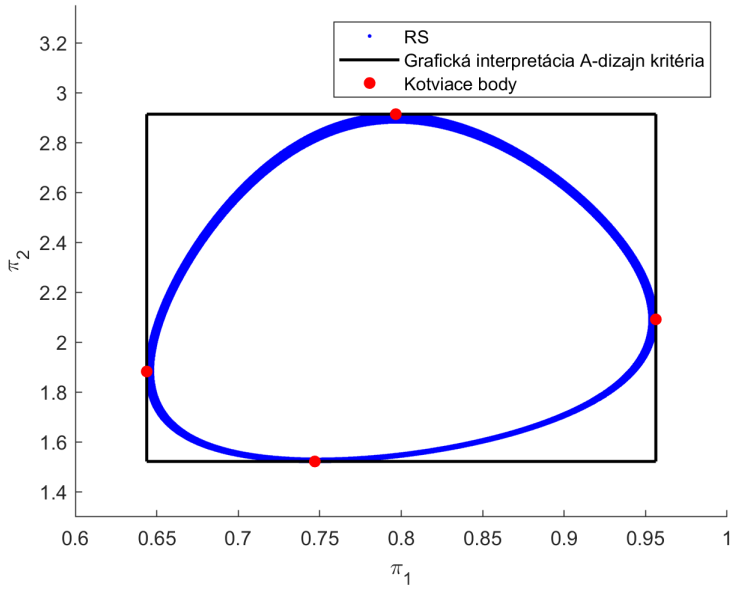
Obr. 2.1: Príklad porovnania suboptimálneho a optimálneho RS

2.2.2 Spôsoby optimalizácie regiónu spoľahlivosti

V literatúre sa uvádzajú mnohé kritéria, ktoré môžu byť použité na získanie optimálneho regiónu spoľahlivosti. Napríklad A, D, E, modifikované E, V, Q, M a ďalšie. Každé z týchto kritérií sa zameriava na zlepšenie určitej vlastnosti RS. V tejto práci sa budeme zaoberať aplikáciou A-kritéria, čo ale neznamená, že použitie iných kritérií by nebolo výhodné.

2.2.3 A-optimálny dizajn

Jeden zo spôsobov určovania RS je A-optimálny dizajn. Pri A-optimálnom dizajne sa snažíme o nasledovné:



Obr. 2.2: Príklad interpretácie A-dizajn kritéria

Na obrázku 2.2 je zobrazený príklad grafickej interpretácie A-dizajn kritéria v prípade, že sa snažíme vytvoriť RS pre dvojicu hľadaných parametrov π_1 a π_2. Pokúšame sa modrú množinu, ktorá predstavuje RS ohraničiť, čo najmenším obdĺžnikom, ktorý reprezentuje grafickú interpretáciu A-dizajn kritéria označenú čiernou farbou na základe kotviacich bodov označených červenými krúžkami.

Všeobecne sa dá identifikovať $2n_p$ kotviacich bodov π, kde každý bod predstavuje hodnotu konkrétneho parametra v skutočnom RS. Počet hľadaných parametrov je vyjadrený premennou n_p.

$$\pi = \begin{bmatrix} \pi_1^L & \pi_2 & \dots & \dots & \pi_{np} \\ \pi_1^U & \pi_2 & \dots & \dots & \pi_{np} \\ \pi_1 & \pi_2^L & \dots & \dots & \pi_{np} \\ \pi_1 & \pi_2^U & \dots & \dots & \pi_{np} \\ \vdots & \vdots & \ddots & & \vdots \\ \vdots & \vdots & & \ddots & \vdots \\ \pi_1 & \pi_2 & \dots & \dots & \pi_{np}^U \\ \pi_1 & \pi_2 & \dots & \dots & \pi_{np}^L \end{bmatrix} \tag{2.3}$$

Kotviace body sa dajú získať vyriešením nasledovného optimalizačného problému:

$$\max_{\boldsymbol{\pi}} \sum_{j=1}^{n_p} \pi_j^U - \pi_j^L \tag{2.4a}$$

$$\text{s.t.} \sum_{k=1}^{n} \frac{(\boldsymbol{y}(\boldsymbol{\pi}_j, \boldsymbol{u}_k) - \boldsymbol{y_m}(\boldsymbol{u}_k))^2}{\sigma^2} \leq \chi^2_{\alpha, n_p} \tag{2.4b}$$

kde

- π_j^L sú spodné hodnoty jednotlivých parametrov.
- π_j^U sú horné hodnoty jednotlivých parametrov.

Celkový dvojúrovňový optimalizačný problém optimálneho návrhu experimentu je potom definovaný nasledovne:

$$\min_{\boldsymbol{u}} \sum_{j=1}^{n_p} \pi_j^U - \pi_j^L \tag{2.5a}$$

$$\arg\max_{\boldsymbol{\pi}} \sum_{j=1}^{n_p} \pi_j^U - \pi_j^L \tag{2.5b}$$

$$\text{s.t.} \sum_{k=1}^{n} \frac{(\boldsymbol{y}(\boldsymbol{\pi}_j, \boldsymbol{u}_k) - \boldsymbol{y_m}(\boldsymbol{u}_k))^2}{\sigma^2} \leq \chi^2_{\alpha, n_p} \tag{2.5c}$$

Problém nižšej úrovne 2.5b zabezpečuje získanie kotviacich bodov na základe funkcií príslušnosti 2.5c, ktoré limitujú prípustné hodnoty parametrov na také, ktoré patria do RS. Problém vyššej úrovne 2.5a sa snaží pomocou vektora akčných zásahov $\boldsymbol{u}$ nájsť čo najmenší RS.

2.3 Prehľad použitých stratégii riešenia

Na nasledujúcich stranách uvádzam spôsoby riešenia dvojurovňových problémov, ktoré boli použité pri tvorbe balíka pre MATLAB.

2.3.1 Karush-Kuhn-Tuckerové podmienky optimality

Karush-Kuhn-Tuckerove podmienky optimality sú štyri nutné, ale nie postačujúce podmienky pre získanie optimálneho riešenia problému. Pri aplikácii na dvojúrovňový

optimalizačný problém má zmysel použiť KKT-podmienky len na problém nižšej úrovne. Problém vyššej úrovne je potom vyriešený globálne pomocou riešiteľa BARON.

2.3.1.1 Podmienka stacionarity

Podmienka stacionarity pre tvar problému, ktorý budeme v tejto práci používať vyzerá nasledovne:

$$\nabla g_u(\boldsymbol{x},\boldsymbol{y}^*) = \sum_{i=1}^{m} \lambda_i \nabla g_{l,i}(\boldsymbol{x},\boldsymbol{y}^*) \tag{2.6}$$

kde

- λ $(1,\ldots,m)$ adjungované pre ohraničenia v tvare rovnosti.
- m je počet obmedzení v tvare nerovnosti.
- * sú označené optimálne hodnoty.

2.3.1.2 Podmienka primárnej zlučiteľnosti

Podmienka primárnej zlučiteľnosti hovorí, že pôvodné ohraničenia musia v optimálnom riešení problému byť splnené.

$$g_{l,i}(\boldsymbol{x},\boldsymbol{y}^*) \leq 0,\ pre\ i = 1,...,m \tag{2.7}$$

2.3.1.3 Podmienka duálnej zlučiteľnosti

Podmienka duálnej zlučiteľnosti hovorí, že ak obmedzenie je aktívne, tak optimálne riešenie bude v bode, kde $\nabla g_u(\boldsymbol{x},\boldsymbol{y}^*)$ a $\nabla g_l(\boldsymbol{x},\boldsymbol{y}^*)$ budú paralelne ($\lambda \geq 0$). Ak nie sú paralelne v optimálnom riešení, tak ohraničenie nie je aktívne.

$$\lambda_i \geq 0,\ pre\ i = 1,...,m \tag{2.8}$$

2.3.1.4 Podmienka doplnkovej voľnosti

$$\lambda_i g_{l,i}(\boldsymbol{x},\boldsymbol{y}^*) = 0 \; pre \; i = 1,...,m \tag{2.9}$$

V prípade, že $m = 0$, teda neexistujú žiadne ohraničenia v tvare nerovnosti, tak sa KKT-podmienky menia na Lagrangeove podmienky a adjungované premenné sa potom nazývajú Lagrangeove násobiče.

2.3.2 Riešenie GSIP pomocou KKT-podmienok

Spodnú úroveň dvojúrovňového optimalizačného problému sme schopný transformovať pomocou KKT-podmienok na sústavu obmedzení pre problém vyššej úrovne. Tento postup je výhodný najmä vtedy, ak problém nižšej úrovne je konvexný. Ak nie je konvexný, tak napriek tomu, že použijeme na vyriešenie GSIP-u globálny riešiteľ, nemusíme dosiahnuť globálne optimálny výsledok. Optimalizačná úloha po transformácii vyzerá nasledovne:

$$\min_{\boldsymbol{x},\boldsymbol{y}} \; f(\boldsymbol{x},\boldsymbol{y}) \tag{2.10a}$$

$$\text{s.t. } \nabla g_u(\boldsymbol{x},\boldsymbol{y}) - \sum_{i=1}^{m} \lambda_i \nabla g_{l,i}(\boldsymbol{x},\boldsymbol{y}) = 0 \tag{2.10b}$$

$$g_{l,i}(\boldsymbol{x},\boldsymbol{y}) \leq 0 \tag{2.10c}$$

$$\lambda_i g_{l,i}(\boldsymbol{x},\boldsymbol{y}) = 0 \tag{2.10d}$$

$$\lambda_i \geq 0 \tag{2.10e}$$

2.3.3 Použitie KKT-podmienok na problém optimálneho návrhu experimentu

Podobne ako v prípade GSIP-u tak aj pri optimálnom dizajne experimentu je možné transformovať problém spodnej úrovne pomocou KKT-podmienok. Častým problémom týchto úloh je nekonvexnosť problému nižšej úrovne, čo spôsobuje aj pri použití globálneho riešiteľa nájdenie iba lokálne optimálnych výsledkov. Transformovaný

problém optimalneho návrhu experimentu je definovaný nasledovne:

$$\min_{\boldsymbol{u},\boldsymbol{\pi}} \sum_{j=1}^{n_p} \pi_j^U - \pi_j^L \tag{2.11a}$$

$$\text{s.t. } \nabla \sum_{j=1}^{n_p} (\pi_j^U - \pi_j^L) - \sum_{i=1}^{m} \lambda_i \nabla (\sum_{k=1}^{n} \frac{(\boldsymbol{y}(\boldsymbol{\pi}_j, \boldsymbol{u}_k) - \boldsymbol{y_m}(\boldsymbol{u}_k))^2}{\sigma^2} - \chi^2_{\alpha,n_p}) = 0 \tag{2.11b}$$

$$\sum_{k=1}^{n} \frac{(\boldsymbol{y}(\boldsymbol{\pi}_j, \boldsymbol{u}_k) - \boldsymbol{y_m}(\boldsymbol{u}_k))^2}{\sigma^2} - \chi^2_{\alpha,n_p} \leq 0 \tag{2.11c}$$

$$\lambda_i (\sum_{k=1}^{n} \frac{(\boldsymbol{y}(\boldsymbol{\pi}_j, \boldsymbol{u}_k) - \boldsymbol{y_m}(\boldsymbol{u}_k))^2}{\sigma^2} - \chi^2_{\alpha,n_p}) = 0 \tag{2.11d}$$

$$\lambda_i \geq 0 \tag{2.11e}$$

Počet ohraničení 2.11c je závislý od počtu parametrov modelu. V prípade jedného parametra budú ohraničenia 2.11c dve, jedno pre hornú hranicu parametra a jedno pre spodnú hranicu parametra, teda počet ohraničení 2.11c je rovný dvojnásobku počtu parametrov.

2.3.4 Blankenship-Falkov algoritmus

V poradí druhá skúmaná metóda riešenia dvojúrovňových problémov je Blankenship-Falkov algoritmus. Jedná sa o iteračnú metódu veľmi podobnú šachovej partii, kde minimalizácia začína svojím prvým návrhom riešenia, "ťahom", na ktorý maximalizácia odpovedá vlastním "ťahom", ktorý predstavuje pridanie nových ohraničení do problému vyššej úrovne. S hromadiacimi sa ohraničeniami sa prípustný region riešení zmenšuje, až kým algoritmus nedosiahne výsledok.

2.3.4.1 Riešenie GSIP Blankenship-Falkovým algoritmom

Pri aplikácii Blankenship-Falkovho algoritmu na GSIP riešime optimalizačný problém nasledovne:

1. Najprv si potrebujeme inicializovať počiatočné hodnoty premenných nižšej úrovne $\boldsymbol{y}$ ako $\boldsymbol{y_0}$. S $\boldsymbol{y}$ definovanými vo vyššej úrovni ako konštanty $\boldsymbol{y_0}$ získame vektor premenných vyššej úrovne $\boldsymbol{x}$, vyriešením optimalizačného problému v našom

prípade minimalizácie v tvare:

$$\min_{\boldsymbol{x}} \; f(\boldsymbol{x},\boldsymbol{y_0}) \tag{2.12a}$$

$$\text{s.t. } g_u(\boldsymbol{x},\boldsymbol{y_0}) \leq 0 \tag{2.12b}$$

2. Na základe výsledného vektora $\boldsymbol{x}^*$, zadefinujeme premenné $\boldsymbol{x}$ v nižšej úrovni(maximalizácia) ako konštanty.

$$\max_{\boldsymbol{y}} g_u(\boldsymbol{x}^*,\boldsymbol{y}) \tag{2.13a}$$

$$\text{s.t. } \boldsymbol{g_l}(\boldsymbol{x}^*,\boldsymbol{y}) \leq 0 \tag{2.13b}$$

Skontrolujeme či je dané riešenie prípustné. Na to, aby bolo dané riešenie prípustné musí platiť:

$$g_u(\boldsymbol{x}^*,\boldsymbol{y}^*) \leq \text{Tol} \tag{2.14}$$

kde Tol je veľmi malé číslo, ktoré je možné nastaviť v poskytnutých funkciách.

3. Ak riešenie je prípustné, tak sme dosiahli optimálny výsledok. V prípade, že riešenie nie je prípustné, tak do vyššej úrovne pridáme nové ohraničenie s hodnotami $\boldsymbol{y}$ získanými maximalizáciou. Vyriešime nasledovný optimalizačný problém a pokračujeme bodom 2.

$$\min_{\boldsymbol{x}} \; f(\boldsymbol{x},\boldsymbol{y_1}) \tag{2.15a}$$

$$\text{s.t. } g_u(\boldsymbol{x},\boldsymbol{y_0}) \leq 0 \tag{2.15b}$$

$$g_u(\boldsymbol{x},\boldsymbol{y_1}) \leq 0 \tag{2.15c}$$

2.3.5 Mitsos-Tsoukalasov algoritmus

Vylepšenie algoritmu Blankenship-Falka ponúkli autori A. Mitsos a A. Tsoukalas [8]. Tento algoritmus rozdelí všeobecný semi-infinitný optimalizačný problém na štyri podproblémy. V nasledovnom delení bolo zanechané značenie, ktoré používali autori algoritmu:

1. GSIP-LBD (z ang. lower bounds of general semi-infinite problem, spodné hranice všeobecného semi-infinitného problému)
2. GSIP-UBD (z ang. upper bounds of general semi-infinite problem, horné hranice všeobecného semi-infinitného problému)
3. LLP (z ang. lower level problem, problém nižšej úrovne)
4. LLP-AUX (z ang. auxiliary lower level problem, doplnkový problém nižšej úrovne)

2.3.5.1 Riešenie GSIP Mitsos-Tsoukalasovým algoritmom

V tejto sekcii si ukážeme ako sú definované jednotlivé podproblémy pri riešení GSIP-u Mitsos-Tsoukalasovým algoritmom.

GSIP-LBD

Výsledkom GSIP-LBD je nájdenie optimálnych parametrov $\boldsymbol{x}^*$ a tým získanie spodnej hranice GSIP. Nadobudnutie optimálneho vektora $\boldsymbol{x}^*$ sa realizuje cez vyriešenie nasledovného optimalizačného problému:

$$\min_{\boldsymbol{x}} \; f(\boldsymbol{x},\boldsymbol{y_{fix}}) \tag{2.16a}$$

$$s.t. \; \min\left\{g_u(\boldsymbol{x},\boldsymbol{y_{fix}}), \; \min_j -g_{l,j}(x,y_{fix})\right\} \leq 0 \quad \forall \; \boldsymbol{y_{fix}} \in Y^{LBD} \tag{2.16b}$$

kde

- Y^{LBD} je množina hodnôt premenných nižšej úrovne $\boldsymbol{y}$ získaná programom LLP.
- $\boldsymbol{y_{fix}}$ sú hodnoty premenných optimalizovaných v probléme nižšej úrovne, ktoré v GSIP-LBD vystupujú ako konštanty.

Pred riešením je nutné poskytnúť programu vhodný nástrel parametrov $\boldsymbol{y_{fix}}$, ktoré budú tvoriť začiatočnú množinu $Y^{LBD,0}$. Pri riešení problému si globálny riešiteľ vyberie minimálne jedno alebo aj viacej ohraničení, ktoré dokáže v danej iterácii splniť, za účelom dosiahnutia optimálnej hodnoty účelovej funkcie $f(\boldsymbol{x})$. V každej iterácii pribudne nová množina ohraničení $g_u(\boldsymbol{x},\boldsymbol{y_{fix}})$ a $-g_{l,j}(\boldsymbol{x},\boldsymbol{y_{fix}})$, ktoré sú generované na základe nových hodnôt $\boldsymbol{y_{fix}}$ získaných vyriešením LLP alebo LLP-AUX.

Funkcia $g_u(\boldsymbol{x},\boldsymbol{y_{fix}})$ predstavuje súčasne ohraničenie pre GSIP – LBD a účelovú funkciu pre problém nižšej úrovne. Funkcie $g_{l,j}(\boldsymbol{x},\boldsymbol{y_{fix}})$ predstavujú dodatočné ohraničenia, ktoré musia byť splnené. Možnosť výberu ohraničenia, ktoré je splnené je užitočná z toho dôvodu, že LLP-AUX môže vygenerovať také $\boldsymbol{y}$, ktoré nie sú prípustné.

Keďže nie každý globálny riešiteľ podporuje funkciu min, tak bolo nutné ohraničenia preformulovať pomocou binárnych premenných $\boldsymbol{z}$, ktoré nadobúdajú hodnotu 1, keď je ohraničenie splnené a 0, keď nie je splnené. Počet pomocných binárnych premenných

n_b sa rovná súčtu počtu $g_{l,j}$ a g_u ohraničení. Ohraničenia v optimalizačnom probléme vyzerajú po transformácii nasledovne:

$$z_j - 1 - \frac{g_{l,j}(\boldsymbol{x},\boldsymbol{y_{fix}})}{g_{l,j}^{max}} \leq 0 \tag{2.17a}$$

$$z_u - 1 + \frac{g_u(\boldsymbol{x},\boldsymbol{y_{fix}})}{g_u^{max}} \leq 0 \tag{2.17b}$$

$$1 - \sum_{i=1}^{n_b} \leq 0 \tag{2.17c}$$

Menovatele $g_{l,j}^{max}$ a g_u^{max} predstavujú nadhodnotenie maximálnej hodnoty funkcie $g_{l,j}$ a g_u. Posledné ohraničenie zabezpečuje, že aspoň jedno ohraničenie je splnené.

GSIP-UBD

GSIP-UBD má za úlohu získanie optimálnych parametrov $\boldsymbol{x}^*$ pre hornú hranicu skúmaného problému. Pre nájdenie optimálneho vektora $\boldsymbol{x}^*$ je nutné vyriešiť nasledovnú optimalizačnú úlohu:

$$\min_{\boldsymbol{x}} \; f(\boldsymbol{x},\boldsymbol{y_{fix}}) \tag{2.18a}$$

$$s.t. \; \min\left\{g_u(\boldsymbol{x},\boldsymbol{y_{fix}}) + \varepsilon_u, \; \min_j -g_{l,j}(\boldsymbol{x},\boldsymbol{y_{fix}}) + \varepsilon_{l,j}\right\} \leq 0 \quad \forall \; \boldsymbol{y_{fix}} \in Y^{UBD} \tag{2.18b}$$

kde

- Y^{UBD} je množina hodnôt $\boldsymbol{y}$ získaná z problému nižšej úrovne (LLP).
- $\varepsilon_{l,j}$ a ε_u sú hodnoty reštrikčných parametrov.

Podobne ako v prípade GSIP-LBD aj pre GSIP-UBD môže byť nutné prerobiť ohraničenia tak, aby sme simulovali funkciu min operátora.

$$z_j - 1 - \frac{g_{l,j}(\boldsymbol{x},\boldsymbol{y_{fix}}) - \varepsilon_{l,j}}{g_{l,j}^{max}} \leq 0 \tag{2.19a}$$

$$z_u - 1 + \frac{g_u(\boldsymbol{x},\boldsymbol{y_{fix}}) + \varepsilon_u}{g_u^{max}} \leq 0 \tag{2.19b}$$

$$1 - \sum_{i=1}^{n_b} \leq 0 \tag{2.19c}$$

Úlohou GSIP-UBD je nájsť hornú hranicu GSIP. Pred riešením je nutné poskytnúť programu vhodný nástrel parametrov $\boldsymbol{y_{fix}}$, ktoré budú tvoriť počiatočnú množinu $Y^{UBD,0}$. Na dosiahnutie tohto cieľa sme do ohraničení pridali pomocné premenné ε_u a $\varepsilon_{l,j}$, ktoré predstavujú reštrikciu pravej strany ohraničení. Ich úlohou je vytvoriť aproximáciu pôvodného GSIP-u, ktorý bude predstavovať hornú hranicu celkového problému. Je možné ich definovať jednotlivo pre každé ohraničenie alebo použiť jednu hodnotu pre všetky. Pri stúpajúcom počte iterácii hodnota týchto premenných klesá a pri dosiahnutí optimálnej hodnoty účelovej funkcie GSIP – UBD a GSIP – LBD dosiahnu rovnaké výsledky.

LLP

Úlohou LLP je získať vektor optimalizovaných premenných problému nižšej úrovne $\boldsymbol{y}$. LLP je definovaný nasledovne:

$$\max_{\boldsymbol{y}} \; g_u(\boldsymbol{x_{fix}},\boldsymbol{y}) \tag{2.20a}$$

$$\text{s.t. } g_{l,j}(\boldsymbol{x_{fix}},\boldsymbol{y}) \leq 0 \tag{2.20b}$$

LLP-AUX

Výsledkom LLP-AUX je získanie Slaterovho bodu, čo predstavuje relaxáciu LLP, ale zároveň reštrikciu pre GSIP – LBD.

$$\min_{\boldsymbol{y}} \max_{j} \; g_{l,j}(\boldsymbol{x_{fix}},\boldsymbol{y}) \tag{2.21a}$$

$$\text{s.t. } \xi g_u(\boldsymbol{x_{fix}},\boldsymbol{y_{fix}}) - g_u(\boldsymbol{x_{fix}},\boldsymbol{y}) \leq 0 \tag{2.21b}$$

Keďže BARON nepodporuje ani funkciu max, tak problém minimalizácie maximálnej hodnoty funkcií ohraničení $g_{l,j}(\boldsymbol{x_{fix}},\boldsymbol{y})$ je nutné transformovať do nasledovnej podoby:

$$\min_{\boldsymbol{y},\omega} \; \omega \tag{2.22a}$$

$$\text{s.t. } g_{l,j}(\boldsymbol{x_{fix}},\boldsymbol{y}) - \omega \leq 0 \tag{2.22b}$$

$$\xi g_u(\boldsymbol{x_{fix}},\boldsymbol{y_{fix}}) - g_u(\boldsymbol{x_{fix}},\boldsymbol{y}) \leq 0 \tag{2.22c}$$

Premenná ω je pomocná premenná, ktorej funkcia je simulácia funkcie max. Parameter ξ nadobúda kladné hodnoty menšie ako 1. Autori algoritmu odporúčajú hodnotu 0,5.

Algoritmus

$F^{LBD} = -\infty$, $F^{UBD} = +\infty$, $Y^{LBD} = Y^{LBD,0}$, $Y^{UBD} = Y^{UBD,0}$,$\varepsilon_u = \varepsilon_{u,0}$, $\varepsilon_{l,j} = \varepsilon_{l,j,0}$ pre každé j
Pokiaľ $(F^{UBD} - F^{LBD}) > \varepsilon_f$
Vyrieš globálne GSIP-LBD.
Ak je problém neriešiteľný, **tak** priraď $F^{U,*} = F^{LBD} = \infty$ a **ukonči program, koniec**
F^{LBD} sa rovná optimálnej hodnote účelovej funkcie (najlepšia nájdená hodnota).
$\boldsymbol{x_{fix,LBD}}$ sa rovná optimálnemu riešeniu (najlepšie nájdené hodnoty).
S hodnotami $\boldsymbol{x_{fix,LBD}}$ získanými v aktuálnej iterácii globálne vyrieš LLP.
Vyriešením LLP sa získa optimálna hodnota účelovej funkcie $g_u^*(\boldsymbol{x_{fix,LBD}},\boldsymbol{y^*})$ a optimálne hodnoty parametrov $\boldsymbol{y}^*$ (najlepšie nájdené hodnoty).
Ak $g_u^*(\boldsymbol{x_{fix,LBD}},\boldsymbol{y^*}) \leq 0$ **tak**
$F^{UBD} = f(\boldsymbol{x_{fix}},\boldsymbol{y^*}), \boldsymbol{x^*} = \boldsymbol{x_{fix}}$, **Ukonči program**
Alebo $g_{l,j}(\boldsymbol{x_{fix,LBD}},\boldsymbol{y^*}) < 0$
Pridaj $\boldsymbol{y}$ do Y^{LBD}.
Inak
Vyrieš LLP – AUX s **y** získanými z LLP a **x** získanými z GSIP – LBD v danej iterácii a pridaj optimálne hodnoty **y** získané z LLP-AUX do Y^{LBD}.
Koniec
Vyrieš globálne GSIP- UBD
Ak je problém riešiteľný tak
$\boldsymbol{x_{fix,UBD}}$ je rovné optimálnemu riešeniu pre UBD.
S hodnotami $\boldsymbol{x_{fix,UBD}}$ získanými v aktuálnej iterácii globálne vyrieš LLP.
Pomocná premenná v sa rovná hodnote účelovej funkcie LLP, $g_u^*(\boldsymbol{x_{fix,UBD}},\boldsymbol{y^*_{UBD}})$.
Vyriešením LLP sa získajú aj optimálne hodnoty $\boldsymbol{y^*_{UBD}}$.
Ak $v < 0$ **potom**
Ak $f(\boldsymbol{x_{fix,UBD}},\boldsymbol{y^*_{UBD}}) < F^{UBD}$ **tak**
$F^{UBD} = f(\boldsymbol{x_{fix,UBD}},\boldsymbol{y^*_{UBD}}),\boldsymbol{x^*} = \boldsymbol{x_{fix,UBD}}$
koniec
Priraď $\varepsilon_u = \varepsilon_u / r_u,\ \varepsilon_{l,j} = \varepsilon_{l,j}/r_j$
Inak
Pridaj $\boldsymbol{y_{fix,UBD}}$ do Y^{UBD}
Koniec
Inak
Priraď $\varepsilon_u = \varepsilon_u / r_u,\ \varepsilon_{l,j} = \varepsilon_{l,j}/r_j$
Koniec
Koniec

2.3.5.2 Použitie Mitsos-Tsoukalasovho algoritmu na problém optimálneho návrhu experimentu

Pred použitím Mitsos-Tsoukalasovho algoritmu na problém optimálneho návrhu experimentu bolo nutné jednotlivé podproblémy upraviť. Potrebnú úpravu navrhli Walzová, Djelassi, Caspari a Mitsos [13]. Potreba získavania hornej hranice optimalizačného problému pomocou riešenia GSIP-UBD nie je nutná, pretože hornú hranicu riešenia optimalizačného problému v danej iterácii získame riešením LLP. Riešenie sa nám potom zjednodušilo na 3 problémy a to GSIP-LBD, LLP a LLP-AUX.

GSIP-LBD

Výsledkom GSIP-LBD je nájdenie optimálnych hodnôt vektora akčných zásahov $\boldsymbol{u}^*$ a získanie spodnej hranice GSIP v podobe pomocného parametra μ. Nadobudnutie optimálneho vektora $\boldsymbol{u}^*$ sa realizuje cez vyriešenie nasledovného optimalizačného problému:

$$\min_{\boldsymbol{u}} \ \mu \tag{2.23a}$$

$$s.t. \ \min\left\{g_u(\boldsymbol{u},\boldsymbol{\pi_{fix}} - \mu) \ \min_j -g_{l,j}(\boldsymbol{u},\boldsymbol{\pi_{fix}})\right\} \leq 0 \quad \forall \ \boldsymbol{\pi_{fix}} \in \Pi^{LBD} \tag{2.23b}$$

$$g_{l,j} = \sum_{k=1}^{n} \frac{(\mathbf{y}(\boldsymbol{\pi}_j,\mathbf{u}_k) - \mathbf{y_m}(\mathbf{u}_k))^2}{\sigma^2} - \chi^2_{\alpha,n_p} \leq 0 \tag{2.23c}$$

$$g_u(\boldsymbol{u},\boldsymbol{\pi_{fix}}) = \sum_{i=1}^{n_p} \pi_i^U - \pi_i^L \tag{2.23d}$$

Rovnako ako pri riešení pomocou KKT-podmienok aj v tomto prípade je počet obmedzení $g_{l,j}$ závislý od počtu odhadovaných parametrov. Počet obmedzení g_l je rovný dvojnásobku počtu parametrov.

Aj v prípade problému optimálneho návrhu experimentu môže byť nutné implementovať tento optimalizačný problém bez použitia min operátora, v tom prípade sa nám

ohraničenia zmenia nasledovne:

$$z_j - 1 - \frac{\sum_{k=1}^{n} \frac{(\mathbf{y}(\boldsymbol{\pi}_j,\mathbf{u}_k)-\mathbf{y_m}(\mathbf{u}_k))^2}{\sigma^2} - \chi^2_{\alpha,n_p}}{g^{max}_{l,j}} \leq 0 \tag{2.24a}$$

$$z_u - 1 + \frac{\sum_{i=1}^{n_p}(\pi_i^U - \pi_i^L) - \mu}{g^{max}_{u}} \leq 0 \tag{2.24b}$$

$$1 - \sum_{i=1}^{n_b} \leq 0 \tag{2.24c}$$

LLP

Úlohou LLP je v rámci možného regiónu spoľahlivosti, ktorý je definovaný vektorom akčných zásahov $\boldsymbol{u}$, nájsť maximálne a minimálne hodnoty jednotlivých parametrov $\boldsymbol{\pi}$.

$$\max_{\boldsymbol{\pi}} \sum_{i=1}^{n_p} \pi_i^U - \pi_i^L \tag{2.25a}$$

$$s.t.\ g_{l,j}(\boldsymbol{u_{fix}},\boldsymbol{\pi}) \leq 0 \tag{2.25b}$$

$$g_{l,j} = \sum_{k=1}^{n} \frac{(\mathbf{y}(\boldsymbol{\pi}_j,\mathbf{u}_k) - \mathbf{y_m}(\mathbf{u}_k))^2}{\sigma^2} - \chi^2_{\alpha,n_p} \tag{2.25c}$$

LLP-AUX

Výsledkom LLP-AUX je získanie Slaterovho bodu, čo predstavuje relaxáciu LLP, ale zároveň reštrikciu pre GSIP – LBD. LLP-AUX získa hodnoty maximálnych a minimálnych hodnôt parametrov, ktoré nie sú na hranici regiónu spoľahlivosti, ale v jeho vnútri.

$$\min_{\boldsymbol{\pi},\boldsymbol{\omega}} \omega \tag{2.26a}$$

$$s.t.\ \xi(\sum_{i=1}^{n_p} \pi_{i,fix}^U - \pi_{i,fix}^L) - \sum_{i=1}^{n_p} \pi_i^U - \pi_i^L \leq 0 \tag{2.26b}$$

$$g_{l,j} = \sum_{k=1}^{n} \frac{(\mathbf{y}(\boldsymbol{\pi}_j,\mathbf{u}_k) - \mathbf{y_m}(\mathbf{u}_k))^2}{\sigma^2} - \chi^2_{\alpha,n_p} \leq 0 \tag{2.26c}$$

Parameter ξ nadobúda kladné hodnoty menšie ako 1. Autori algoritmu odporúčajú hodnotu 0,5.

Upravený algoritmus pre optimálny návrh experimentu

$F^{LBD} = -\infty$, $F^{UBD} = +\infty$, $\Pi^{LBD} = \Pi^{LBD,0}$,$\varepsilon_u = \varepsilon_{u,0}$, $\varepsilon_{l,j} = \varepsilon_{l,j,0}$ pre každé j
Pokiaľ $(F^{UBD} - F^{LBD}) > \varepsilon_f$
Vyrieš globálne GSIP-LBD
Ak je problém neriešiteľný, **tak** priraď $F^{U,*} = F^{LBD} = \infty$ a **ukonči program, koniec**
F^{LBD} sa rovná optimálnej hodnote účelovej funkcie (najlepšia nájdená hodnota).
$\boldsymbol{u_{fix}}$ sa rovná optimálnemu riešeniu (najlepšie nájdené hodnoty).
S hodnotami $\boldsymbol{u_{fix}}$ získanými v aktuálnej iterácii globálne vyrieš LLP.
Vyriešením LLP sa získa optimálna hodnota účelovej funkcie $g_u^*(\boldsymbol{u_{fix}}, \boldsymbol{\pi})$ a optimálne hodnoty parametrov $\mathbf{y}$ (najlepšie nájdené hodnoty)
Ak $F^{UBD} > g_u^*(\boldsymbol{u_{fix}}, \boldsymbol{\pi})$ **tak**
$F^{UBD} = g_u^*(\boldsymbol{u_{fix}}, \boldsymbol{\pi})$
Koniec
Ak $g_{l,j}(\boldsymbol{u_{fix}}, \boldsymbol{\pi^*}) < 0$
Pridaj $\boldsymbol{\pi}$ do Π^{LBD}
Inak
Vyrieš LLP – AUX s $\boldsymbol{\pi}$ získanými z LLP a $\boldsymbol{u}$ získanými z GSIP – LBD v danej iterácii
Pridaj $\boldsymbol{\pi}$ do Π^{LBD}
Koniec
Koniec

Kapitola 3

Implementácia funkcií bi_level a bi_level_OED

Za účelom umožnenia používania opísaných algoritmov boli zostrojené dve funkcie. Funkcia bi_level je schopná riešiť dvojúrovňové optimalizačné problémy vo všeobecnom tvare, definovanom v 2.1a pomocou Karush-Kuhn-Tuckerových podmienok optimality, Blankenship-Falkovým a Mitsos-Tsoukalasovým algoritmom. Funkcia bi_level_OED je schopná riešiť problém optimálneho návrh experimentu 2.5 transformáciou na jednoúrovňový problém KKT-podmienkami alebo Mitsos-Tsoukalasovým algoritmom. Repozitár so všetkými funkciami ju možné si stiahnuť cez git:

```
git clone https://Dany8246@bitbucket.org/Dany8246/bi_level.git
```

Súčasťou balíka funkcií je aj krátka príkladová dokumentácia vo formáte html, kde sú v krátkosti popísané jednotlivé funkcie a ich možnosti.

3.1 Výpočtové prostredie MATLAB

Pre riešenie skúmaných problémov sme si vybrali programovacie prostredie MATLAB. Toto prostredie sme si zvolili najmä kvôli dostupnosti globálneho riešiteľa BARON, pre ktorý existuje rozhranie medzi vlastným BARON-ovským prostredím a MATLAB-om. Ďalšou výhodou MATLAB-u je jemu vlastný symbolický balík, ktorý nám umožňuje efektívne a intuitívne narábať s funkciami pri tvorbe účelových funkcií a ohraničení pre optimalizačné problémy. Symbolický balík nám ďalej umožňuje intuitívnejšie definovanie vstupov do funkcií ako keby bola tvorba vstupných funkcií riešená pomocou iných dostupných metód.

3.1.1 BARON

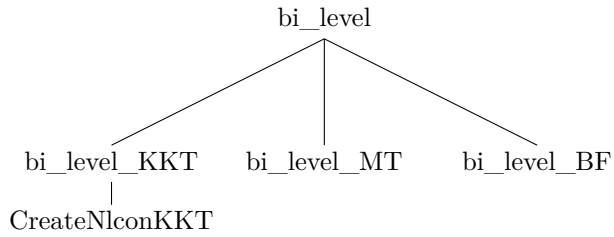
BARON je všeobecný riešiteľ, ktorý používa na riešenie optimalizačných problémov metódu vetiev a hraníc(z ang. Branch and Bound). Je schopný riešiť kontinuálne, celočíselné(z ang. integer problems) a zmiešané-celočíselné nelineárne problémy(z ang. mixed-integer nonlinear problems). BARON môže byť rozšírený o IBM ILOG CPLEX, ktorý rieši lineárne problémy a zmiešané-celočíselné problémy.

3.2 Štruktúra funkcií bi_level a bi_level_OED

Z dôvodu jednoduchšej, rýchlejšej obsluhy a väčšieho pohodlia pre užívateľa sú funkcie, ktoré používajú jednotlivé implementované algoritmy zakomponované do jednej nadradenej funkcie. Takto je možné po zadefinovaní všetkých potrebných parametrov jednoduchou zmenou premennej prepnúť na algoritmus, ktorý chceme v danom momente použiť. V nasledovných sekciách je uvedený kompletný strom funkcií, ktoré nadradené programy bi_level a bi_level_OED používajú pri tvorení a riešení optimalizačných problémov.

3.2.1 Strom a popis volaných funkcií programu bi_level

Strom volaných funkcií bi_level programu vyzerá nasledovne:



KKT stratégia

bi_level_KKT používa na riešenie Karush-Kuhn-Tuckerové podmienky optimality. Po zadefinovaní všetkých vstupov funkcia pretvorí štandardný tvar optimalizačného problému 2.1a na jednoúrovňový problém 2.10a použitím funkcie CreateNlconKKT.

CreateNlconKKT je funkcia zodpovedná za vytvorenie ohraničení pre optimalizovaný problém, táto funkcia rieši najmä vytvorenie gradientu Lagrangeovej funkcie, čo

v prípade zložitejších problémov môže byť veľmi výpočtovo náročná operácia. Pri tvorbe gradientu funkcia CreateNlconKKT využíva symbolický balík a MATLAB-u natívnu funkciu jacobian, ktorá je schopná vypočítať gradient pre vektor definovaných premenných. Po získaní gradientu funkcia vytvorí stĺpcový vektor ohraničení skladajúci sa z jednotlivých Karush-Kuhn-Tuckerových podmienok optimality a spolu s účelovou funkciou ich transformuje zo symbolického balíka do formátu function_handle, ktorý je vyžadovaný BARON-om ako vstupný formát. Funkcia CreateNlconKKT vytvorí aj horné a spodné hodnoty jednotlivých ohraničení, ktoré sú vyžadované BARON-om ako vstupné argumenty.

Po vytvorení všetkých funkcií ohraničení, účelovej funkcie, definovaní horných a spodných hodnôt jednotlivých premenný a ohraničení vo vhodných formátoch bi_level_KKT zavolá BARON. Po skončení behu BARON-u je ďalej výsledok spracovaný do požadovaného formátu a následne je poskytnutý užívateľovi.

BF stratégia

bi_level_BF používa na riešenie Blankenship-Falkov algoritmus. Táto funkcia nevyžaduje vytvorenie gradientu, ale v každej iterácii sa zvyšuje množstvo ohraničení pre problém vyššej úrovne.

Stále je nutná transformácia vstupov do funkcie prostredníctvo symbolického balíka a následné pretvorenie účelových funkcií a funkcií ohraničení do formátu function_handle v každej iterácii. Blankenship-Falkov algoritmus je iteračná stratégia riešenia a teda práca s funkciami musí prebiehať v každej iterácii, tento proces bol zlepšený tak aby nebolo nutné vytvárať celé funkcie ohraničení v každej iterácií od začiatku, ale si program udržuje v pamäti ohraničenia z predchádzajúcej iterácie a nové ohraničenia sa do množiny pridajú.

Po každom vytvorení funkcií ohraničení, účelovej funkcie a definovaní hraníc pre premenné a ohraničenia je zavolaný BARON, ktorý získa výsledky pre danú iteráciu. Na základe prípustnosti výsledkov sa potom algoritmus buď ukončí alebo pokračuje ďalej pridaním ohraničení. V prípade najdenia prípustného výsledku je výstup spracovaný do preddefinovaného formátu a poskytnutý užívateľovi.

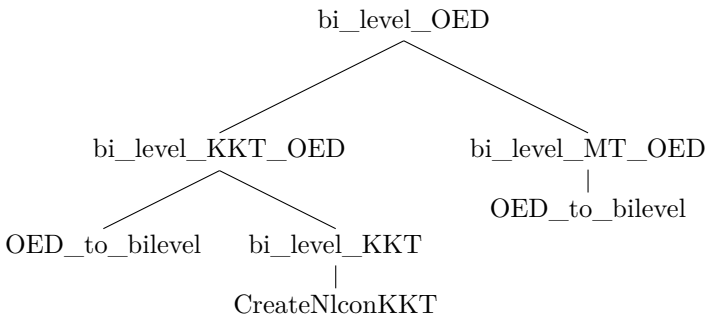
MT stratégia

bi_level_MT aplikuje Mitsos-Tsoukalasov algoritmus 2.3.5.1, ktorý je vylepšenou verziou Blankenship-Falkovho algoritmu. Táto stratégia je najnáročnejšia na implementáciu, pretože na riešenie dvojúrovňového optimalizačného problému je potrebné v symbolickom balíku vytvoriť 5 podproblémov a to GSIP-LBD 2.16a, LLP 2.20, LLP-AUX 2.3.5.1 pre získanie spodnej hranice problému a GSIP-UBD 2.18a a LLP 2.20 pre získanie hornej hranice problému. Po vytvorení podproblémov je samozrejme nutné ich dodať BARON-u vo formáte function_handle.

LLP je v oboch prípadoch rovnako definované, ale vstupné hodnoty optimalizovaných premenných vyššej úrovne označené ako $\boldsymbol{x_{fix}}$ sú rozdielne pre LLP v UBD a LBD časti. Po vytvorení všetkých podproblémov je možné začať problém riešiť. Keďže sa jedná o algoritmus podobný Blankenship-Falkovmu algoritmu aj v tomto prípade dochádza k pridávaniu ohraničení do funkcie vyššej úrovne (GSIP-LBD a GSIP-UBD). Funkcia rozšíri a upraví všetky ohraničenia automaticky. Na konci behu funkcie sa výsledky uložia do preddefinovaného formátu a poskytnú užívateľovi.

3.2.2 Strom a popis volaných funkcií programu bi_level_OED

Štruktúra volaných funkcií nadradeného programu bi_level_OED je nasledovná:



KKT stratégia pre optimálny návrh experimentu

bi_level_KKT_OED má za úlohu riešenie problémov optimálneho návrhu experimentu redukciou dvojúrovňového problému na problém jednej úrovne pomocou Karush-Kuhn-Tuckerových podmienok optimality.

Funkcia na základe stavového a výstupného modelu a informácie o počte meraní vytvorí najprv dvojúrovňový optimalizačný problém, túto časť má za úlohu funkcia OED_to_bilevel. Optimalizačný problém je vytvorený v takom tvare aby mohla byť využitá už existujúca funkcia bi_level_KKT 3.2.1, ktorá vytvorí finálny problém v tvare 2.11. Po skončení riešenia problému sú výsledky naformátované do preddefinované formátu a poskytnuté používateľovi.

MT stratégia pre optimálny návrh experimentu

bi_level_MT_OED je funkcia používajúca Mitsos-Tsoukalasov algoritmus pre optimálny návrh experimentu 2.3.5.2. Na rozdiel od pôvodného algoritmu 2.3.5.1 získavanie hornej hranice problému sa uskutočňuje už v LLP a tak je celkový problém zjednodušený na 3 podproblémy a to GSIP 2.23, LLP 2.25 a LLP-AUX 2.26, ktoré je nutné v symbolickom balíku vytvoriť a v každej iterácii upraviť, rozšíriť a previesť na function_handle.

Aj v tejto funkcii využívame OED_to_bilevel funkciu, ktorá z informácii o počte meraní, stavového a výstupného modelu vytvorí optimalizačný problém, ktorý je následné upravený na jednotlivé podproblémy. Po vyriešení problému sú výsledky uložené, do preddefinovaného formátu a poskytnuté užívateľovi.

3.3 Volanie funkcií a popis jednotlivých prvkov

V tejto sekcii je popísaný spôsob akým sa používajú pripravené funkcie a ako sa nastavujú ich povinné a voliteľné parametre.

3.3.1 Argumenty funkcie bi_level

Funkciu bi_level je možné po vložení do MATLAB-u zavolať nasledovným príkazom:

```
[J_opt, lv_opt, uv_opt, Log] = bi_level(obj_fun, obj_LLP, const, lv, uv,...
lb, ub, start, strategy, 'MaxIter', MaxIter, 'Tol', Tol, 'Xi', Xi,...
'Gl_max', Gl_max, 'Gu_max', Gu_max, 'Eps_L', Eps_L, 'Eps_R', Eps_R ,...
'Eps_U', Eps_U, 'BarRuntime', BarRuntime, 'Baronset', Baronset)
```

Výsledky získané funkciou sa ukladajú do štyroch premenných:

- J_opt v ktorej sa nachádza optimálna hodnota účelovej funkcia problému vyššej úrovne.

- lv_opt je stĺpcový vektor hodnôt optimalizovaných premenných nižšej úrovne, kde premenné sú usporiadané v poradí v akom boli zadané programu.

- uv_opt je stĺpcový vektor hodnôt optimalizovaných premenných vyššej úrovne, kde premenné sú usporiadané v poradí v akom boli zadané programu.

- Log je štruktúra v ktorej sú uložené dodatočné informácie získané jednotlivými stratégiami riešenia. Premennej Log je venovaná samostatná sekcia.

Povinné aj voliteľné vstupné argumenty a ich formát vstupu je popísaný v nasledovných tabuľkách:

Tabuľka povinných argumentov		
Názov	Príklad formátu vstupu	Popis
obj_fun	$syms\ x1,\ x2$ $obj\_fun = x2$	Definuje účelovú funkciu vyššej úrovne. (symbolická funkcia)
obj_LLP	$syms\ x1,\ x2,\ y1,\ y2$ $obj\_LLP = (x1+y1)^2 - (x2+y2);$	Definuje účelovú funkciu pre nižšiu úroveň. (symbolická funkcia)
const	$Q = [0.8,\ -0.6;\ -0.6,\ 0.8]$ $const\ =\ [[v,\ w] * inv(Q) * [v;\ w] - 1]$	Ohraničenia problému nižšej úrovne v tvare nerovnosti $const \leq 0$ (symbolický stlpcový vektor)
lv	$lv\ =\ [y1;\ y2]$	Definuje programu premenné pre nižšiu úroveň (symbolický stĺpcový vektor)
uv	$uv\ =\ [x1;\ x2]$	Definuje programu premenné pre vyššiu úroveň (symbolický stĺpcový vektor)
lb	$lb\ =\ [-10,\ -10,\ -10,\ -10]$	Definuje spodnú hranicu pre premenné vo formáte: $[uv(1),uv(2),...,uv(n),lv(1),lv(2),...,lv(m)]$ (riadkový vektor)
ub	$ub\ =\ [10,\ 10,\ 10,\ 10]$	Definuje hornú hranicu pre premenné vo formáte: $[uv(1),uv(2),...,uv(n),lv(1),lv(2),...,lv(m)]$ (riadkový vektor)
start	$start\ =\ [0;\ 0]$	Určuje nástrely premenných spodnej úrovne pre Blankenship-Falkov a Mitsos-Tsoukalasov algoritmus (stĺpcový vektor)
strategy	$strategy = "BF"$	Vyberá stratégiu použitú na riešenie problému možnosti $"KKT"/"BF"/"MT"$ (reťazec)

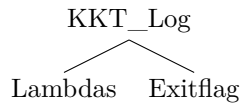
Tabuľka 3.1: Tabuľka povinných argumentov pre bi_level funkciu

Tabuľka voliteľných argumentov		
Názov	Príklad formátu vstupu	Popis
MaxIter	$MaxIter = 30$	Definuje maximálny počet cyklov pre Blankenship-Falkov a Mitsos-Tsoukalasov algoritmus prednastavené na 30 (celé číslo)
Tol	$Tol = 1e-3$	Definuje toleranciu pre konečnú podmienku Blankenship-Falkovho a Mitsos-Tsoukalasovho algoritmu (číslo)
Xi	$Xi = linspace(0.5,0.9,30)$	Definuje koeficient pre LLP-AUX 2.3.5.1) v jednotlivých iteráciách Riadkový vektor s prvkami medzi 0 a 1 o dlžke MaxIter(prednastavené 30) (riadkový vektor)
Gl_max	$Gl\_max = 30$	Definuje maximálnu pozitívnu hodnotu ohraničení nižšej úrovne 2.17 (číslo)
Gu_max	$Gu\_max = 35$	Definuje maximálnu pozitívnu hodnotu účelovej funkcie nižšej úrovne 2.17 (číslo)
Eps_L	$Eps\_L = 2$	Definuje hodnotu použitú na reštrikciu pravej strany pre ohraničenia nižšej úrovne v UBD časti Mitsos-Tsoukalasovho algoritmu2.19 (číslo)
Eps_R	$Eps\_R = 2$	Definuje parameter, ktorý zmenšuje reštrikciu pravej strany (v algoritme r_u) 2.3.5.1 (číslo)
Eps_U	$Eps\_U = 2$	Definuje hodnotu použitú na reštrikciu pravej strany pre účelovú funkciu nižšej úrovne v UBD časti Mitsos-Tsoukalasovho algoritmu2.19 (číslo)
BarRuntime	$BarRuntime = linspace(300,3600,30)$	BarRuntime definuje maximálny čas behu BARON-u v jednotlivých iteráciach. Je to vektor časov v sekundách o dĺžke práve MaxIter(prednastavené na 30) (riadkový vektor)
Baronset	$Baronset = baronset('maxtime',150,...$ $'threads',8)$	Cez Baronset je možné definovať všetky voliteľné možnosti BARON-u, pre ďalšie informácie o premennej Baronset je odporúčané nahliadnuť do dokumentácie BARON-u. (baronset-štruktúra)

Tabuľka 3.2: Tabuľka voliteľných argumentov pre bi_level funkciu

3.3.2 Štruktúra logov funkcie bi_level

3.3.2.1 Log pre KKT stratégiu riešenia



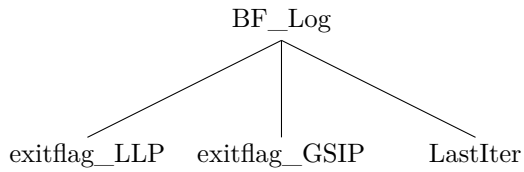
Stratégia založená na Karush-Kuhn-Tuckerových podmienkach optimality, má v premennej Log uložené hodnoty Karush-Kuhn-Tuckerových multiplikátorov pod názvom "Lambdas", ktoré vyjadrujú ako sa približne zmení hodnota účelovej funkcie v prípade zmeny ohraničenia ku ktorému daná lambda prislúcha. Jednotlivé lambdy sú priradené k ohraničeniam podľa poradia v ktorom boli ohraničenia zadané do programu. Druhá premenná je Exitflag, ktorá dáva informáciu o úspešnosti vyriešenia optimalizačného problému riešiteľom BARON v podobe čísla bez desatinnej čiarky jednotlivé možné výsledky a ich význam sú popísané v nasledovnej tabuľke:

Exitflag	Popis
1	Optimum v rámci tolerancie
2	Neriešiteľný problém
3	Neohraničený problém
4	Prechodne riešiteľný problém
5	Neznáma chyba

Tabuľka 3.3: Tabuľka možných hodnôt premennej Exitflag

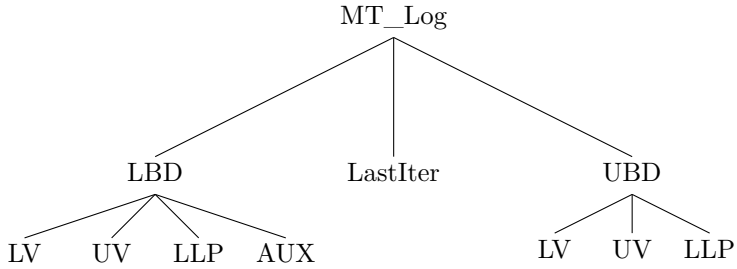
V prípade že Exitflag dosiahne status 4 alebo 5 tak niektoré premenné nemajú vhodne nastavené hranice a BARON nie je schopný garantovať globálne optimálny výsledok.

3.3.2.2 Log pre Blankenship-Falkovu stratégiu riešenia



Blankenship-Falkov algoritmus poskytuje v premennej Log prístup k dvom exitflag premenným, kde LLP sa vzťahuje na riešenie problému nižšej úrovne a GSIP sa odkazuje na problém vyššej úrovne. Exitflag premenené sú v rovnakom tvare ako v prípade KKT. LastIter poskytuje informáciu o počte iterácií (počte cyklov pridávania ohraničení) potrebných na vyriešenie problému.

3.3.2.3 Log pre Mitsos-Tsoukalasovu stratégiu riešenia



Mitsos-Tsoukalasov algoritmus poskytuje robustný Log s premennou LastIter poskytujúcou informáciou o počte cyklov potrebných na riešenie a dvoma štruktúrami LBD a UBD. Do štruktúry LBD sa ukladajú všetky hodnoty optimalizovaných premenných v rámci spodného ohraničujúceho cyklu, teda LBD časti algoritmu 2.3.5.1.

- LV je matica hodnôt všetkých premenných $\boldsymbol{y_{fix}}$ použitých na vytváranie $g_l(\boldsymbol{x},\boldsymbol{y_{fix}})$ ohraničení v jednotlivých cykloch. V algoritme je táto premenná označená ako Y^{LBD}
- UV je matica hodnôt všetkých premenných vyššej úrovne získaných pomocou riešenia GSIP-LBD. V jednotlivých stĺpcov sú premenné zoradené rovnako ako boli zadané do programu.
- LLP je matica hodnôt všetkých premenných vyprodukovaných pri riešení LLP programu. Z týchto hodnôt sa môže napĺňať LV.
- AUX je matica hodnôt všetkých premenných vyprodukovaných pri riešení LLP-AUX programu. Z týchto hodnôt sa môže napĺňať LV v prípade, že LLP hodnoty sú nevyhovujúce.

Pre UBD existuje podobný Log s tým rozdielom, že v UBD sa nevyskytuje AUX, pretože v UBD programe LLP-AUX riešený nie je.

3.3.3 Argumenty funkcie bi_level_OED

```
[J_opt, U_opt, Param_opt, Log] = bi_level_OED(model_x, model_y, u,...
parameters, parameters_values, x0, x0_value, lb, ub, start, model_type,...
strategy, 'MaxIter', MaxIter, 'Alpha', Alpha, 'Sigma', Sigma,...
'Tol', Tol, 'Xi', Xi 'Gl_max', Gl_max, 'Gu_max', Gu_max,...
'BarRuntime', BarRuntime, 'Baronset', Baronset)
```

Výsledky získané funkciou sa ukladajú do štyroch premenných:

- J_opt v ktorej sa nachádza optimálna hodnota účelovej funkcia problému vyššej úrovne.
- U_opt je stĺpcový vektor hodnôt optimalizovaných premenných vyššej úrovne. Jedná sa o vektor akčných zásahov pri ktorých bola dosiahnutá daná hodnota účelovej funkcie.
- Param_opt je matica v ktorej prvom riadku sú najmenšie hodnoty parametrov a v druhom riadku sú najväčšie hodnoty parametrov. Parametre sú zoradené tak ako boli programu zadané.
- Log je štruktúra v ktorej sú uložené dodatočné informácie získané jednotlivými stratégiami riešenia. Premennej Log je venovaná samostatná sekcia.

Povinné aj voliteľné vstupné argumenty a ich formát vstupu je popísaný v nasledovných tabuľkách:

Tabuľka povinných argumentov		
Názov	Príklad formátu vstupu	Popis
model_x	$u = sym('u',[4\ 1])$ $syms\ gama,\ x0$ $model\_x = \frac{1}{(gama+x0^2)} + u(1)$	Definuje stavový model skúmaného procesu. Musí obsahovať prvý prvok vektora akčných zásahov. (symbolická funkcia)
model_y	$syms\ beta$ $model\_y = beta * model\_x$	Definuje výstupný model skúmaného procesu. (symbolická funkcia)
u	$u = sym('u',[4\ 1])$	Vektor akčných zásahov, musí byť súčasťou modelov (symbolický stlpcový vektor)
parameters	$parameters = [gama,\ beta]$	Definuje programu parametre modelu (symbolický riadkový vektor)
parameters_values	$parameters\_values = [2,0.8]$	Definuje programu ideálne hodnoty parametrov modelu (riadkový vektor)
x0	$syms\ x0$	Definuje premennú ktorá predstavuje počiatočnú podmienku pre diskrétne modely (symbolická premenná)
lb	$lb = [-1\ 0\ 0]$	Definuje spodnú hranicu pre premenné vo formáte: $[u,\ param1,\ param2,\ ...\ ,param(n)]$ (riadkový vektor)
ub	$ub = [1\ 10\ 10]$	Definuje hornú hranicu pre premenné vo formáte: $[u,\ param1,\ param2,\ ...\ ,param(n)]$ (riadkový vektor)
start	$start =$ [3.3167 1.3256 4.2460 1.1322 0.9837 0.6377 0.9244 0.6811]	Výber nástrelových bodov parametrov pre Mitsos-Tsoukalasov algoritmus vo formáte: 4.3 (matica)
strategy	$strategy =$ "MT"	Vyberá stratégiu použitú na riešenie problému možnosti: "KKT"/"MT" (reťazec)
model_type	$model\_type =$ "$discrete$"	Určuje typ modelu. možnosti "$discrete$"/"$static$" (reťazec)

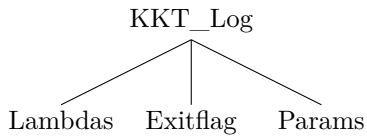
Tabuľka 3.4: Tabuľka povinných argumentov pre bi_level_OED funkciu

Tabuľka voliteľných argumentov		
Názov	Príklad formátu vstupu	Popis
MaxIter	$MaxIter = 30$	Definuje maximálny počet cyklov pre Mitsos-Tsoukalasov algoritmus (celé číslo)
Alpha	$Alpha = 0.95$	Alpha je pravdepodobnosť daná funkcii chi2inv, ktorá vypočíta χ^2_{α,n_p} (0<Alpha<1)
Sigma	$Sigma = 0.1$	Predstavuje štandardnú odchýlku šumu merania (číslo)
Tol	$Tol = 1e-3$	Definuje toleranciu pre konečnú podmienku Mitsos-Tsoukalasovho algoritmu (číslo)
Xi	$Xi = linspace(0.5,0.9,30)$	Definuje koeficient pre LLP-AUX 2.3.5.1) v jednotlivých iteráciách Riadkový vektor s prvkami medzi 0 a 1 o dlžke MaxIter(prednastavené 30) (riadkový vektor)
Gl_max	$Gl\_max = 30$	Definuje maximálnu pozitívnu hodnotu ohraničení nižšej úrovne 2.24 (číslo)
Gu_max	$Gu\_max = 35$	Definuje maximálnu pozitívnu hodnotu účelovej funkcie nižšej úrovne 2.24 (číslo)
BarRuntime	$BarRuntime = linspace(300,3600,30)$	BarRuntime definuje maximálny čas behu BARON-u v jednotlivých iteráciach. Je to vektor časov v sekundách o dĺžke práve MaxIter(prednastavené na 30) (riadkový vektor)
Baronset_U	$Baronset = baronset('maxtime',150,...$ $'threads',8)$	Cez Baronset je možné definovať všetky voliteľné možnosti BARON-u, pre ďalšie informácie o premennej Baronset je odporúčané nahliadnuť do dokumentácie BARON-u. (baronset-štruktúra)

Tabuľka 3.5: Tabuľka voliteľných argumentov pre bi_level_OED funkciu

3.3.4 Štruktúra logov bi_level_OED funkcie

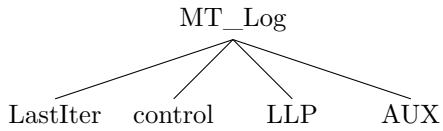
3.3.4.1 Log pre KKT stratégiu riešenia



Log v prípade použitia KKT stratégie bi_level_OED funkcie je podobný ako v prípade bi_level programu. Rozdiely sú v tom, že jednotlivé lambdy sa odvolávajú na funkcie príslušnosti 2.11c, ktoré si bi_level_OED poskladá na základe modelu sám a “Params”

predstavuje vektor všetkých optimalizovaných parametrov, teda aj doplnkových.

3.3.4.2 Log pre Mitsos-Tsoukalasovu stratégiu riešenia



Pri použití Mitsos-Tsoukalasovej stratégie premenná Log obsahuje číslo poslednej iterácie v premennej LastIter a matice control, LLP a AUX

- control obsahuje maticu s hodnotami akčných zásahov získaných v jednotlivých iteráciách.
- LLP obsahuje maticu s hodnotami parametrov v jednotlivých iteráciách získaných programom LLP. Každý stĺpec LLP matice predstavuje maticu π transformovanú do jedného stĺpca, tak, že každý ďalší stĺpec matice π je uložený pod prvý stĺpec. Z týchto hodnôt sa môže napĺňať množina Π v algoritme2.3.5.2.
- AUX je matica hodnôt parametrov vyprodukovaných pri riešení LLP-AUX programu. Z týchto hodnôt sa môže napĺňať množina Π v algoritme2.3.5.2 v prípade, že LLP hodnoty sú nevyhovujúce.

V nasledujúcej kapitole sú uvedené príklady použitia aj s úplným návodom na zápis daných problémov.

Kapitola 4

Príklady použitia funkcií bi_level a bi_level_OED

V nasledovných sekciách sa dajú nájsť kompletné príklady použitia vyššie spomínaných funkcií. Sekcie s príkladmi riešenia problémov sú rozdelené na definíciu problému, prípravu vstupných argumentov a výsledky dosiahnuté všetkými stratégiami.

4.1 Problém riešený bi_level funkciou

V tejto sekcii sa venujeme príkladu problému vo všeobecnom formáte popísanom 2.1a.

4.1.1 Problém minimalizácie y-súradnice elipsy

V rámci tohoto problému sa snažíme umiestniť elipsu daných rozmerov čo najnižšie nad ohraničenie v tvare paraboly, tak aby ich elipsa nepretla.

4.1.1.1 Definovanie problému

Optimalizačný problém je definovaný nasledovne:

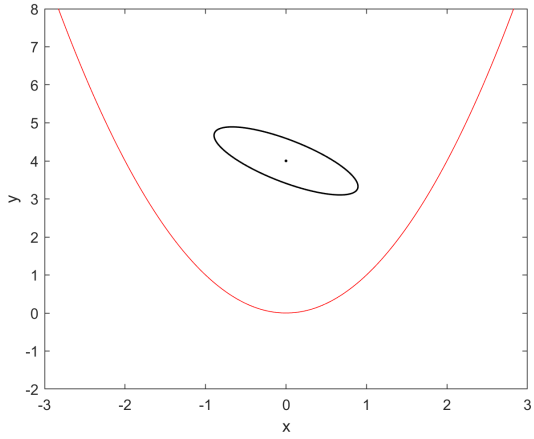
$$\min_{x,y} y \tag{4.1a}$$

$$arg\max_{v,w}(x+v)^2 - (y+w) \tag{4.1b}$$

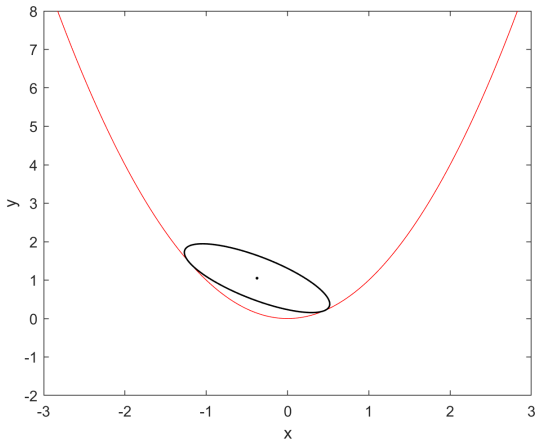
$$s.t.\ [v,w]Q^{-1}\begin{bmatrix} v \\ w \end{bmatrix} \leq 0 \tag{4.1c}$$

kde

- $Q = \begin{pmatrix} 0.8 & -0.6 \\ -0.6 & 0.8 \end{pmatrix}$ je kovariančná matica, ktorá definuje rozmery a natočenie elipsy.
- x,y sú súradnice stredu elipsy.
- v,w sú odchylkové premenné pomocou, ktorých hľadáme bod ktorý najviac porušuje kvadratické ohraničenie.



Obr. 4.1: Počiatočná pozícia elipsy



Obr. 4.2: Optimálna pozícia elipsy

4.1.1.2 Definovanie vstupných argumentov

V nasledovnej ukážke kódu sú definované všetky povinné a aj voliteľné argumenty aj s popisom ich účelu. Ako prvé je nutné zadefinovať symbolické premenné pre optimalizačný problém.

```
syms x y v w
```

Keď máme zadefinované symbolické premenné tak s ich pomocou vytvoríme účelovú funkciu vyššej úrovne, účelovú funkciu nižšej úrovne a ohraničenia.

```
% Ucelova funkcia problemu vyssej urovne
% (symbolicka funkcia)
obj_fun = y;
% Ucelova funkcia problemu nizsej urovne
% (symbolicka funkcia)
obj_LLP = (x+v)^2-(y+w);
% Ohranicenia pre problem nizsej urovne v tvare:
% const <= 0
% (symbolicky stlpcovy vektor)
const  = [[v,w]*inv([0.8 -0.6;-0.6 0.8])*[v;w]-1];
```

BARON potrebuje mať zadefinované maximálne a minimálne hranice všetkých optimalizovaných premenných.

```
% Spodne a horne hranice hodnot optimalizovanych premennych
% format:[uv(1),uv(2),...,uv(n),lv(1),lv(2),...,lv(m)]
% (riadkovy vektor)
lb = [-10 -10 -10 -10];
ub = [10 10 10 10];
```

Následne je nutné definovať ktoré symbolické premenné patria vyššej úrovni a ktoré nižšej úrovni.

```
% Premenne problemu vyssej urovne
% (symbolicky stlpcovy vektor)
uv = [x;y];
% Premenne problemu nizsej urovne
% (symbolicky stlpcovy vektor)
lv = [v;w];
```

Stratégie riešenie BF a MT potrebujú počiatočné hodnoty premenných nižšej úrovne, preto je nutné ich definovať.

```
% Nastrelove hodnoty optimalizovanych premennych nizsej urovne
% pre strategie "BF" a "MT"
% (stlpcovy vektor)
start = [0;0];
```

Posledným povinným argumentom je stratégia riešenia, ktorú si môžeme zvoliť napríklad “BF”.

```
% Definuje, ktora strategia riesenia problemu bude pouzita
% moznosti su "KKT", "BF" a "MT"
strategy = "BF"
```

Následne si môžeme zvoliť voliteľné argumenty. Voliteľné argumenty môžu byť zadané v ľubovolnom poradí pokiaľ sa dodrží párovanie názov argumentu čiarka hodnota argumentu. Prvým voliteľným argumentom je MaxIter, ktorý tvorí stopovacie kritériu na počet cyklov pre MT a BF stratégiu, predvolená hodnota je 30.

```
% Maximalny pocet iteracii pre "MT" a "BF" strategie)
% (kladne cislo)
MaxIter = 30;
```

Prísnosť kritéria ktoré rozhoduje o dosiahnutej presnosti optimálneho riešenia pre algoritmi BF 2.14 a MT 2.3.5.1 je možné nastaviť zmenou hodnoty premennej Tol, ktorej predvolená hodnota je $1e-6$.

```
% Definuje prisnost koncovej podmienky pre "BF" a "MT" algoritmus
% (male kladne cislo)
Tol = 1e-4;
```

Pri použití MT algoritmu je v podprobléme LLP-AUX 2.3.5.1 možné meniť parameter ξ. Autori odporúčajú hodnotu 0.5. Zadávanie tohoto parametra je prispôsobené tak, že uživateľ má možnosť zadať inú hodnotu parametra v každej iterácii. Pre správne fungovanie tohoto parametra je potrebné aby uživateľ zadal riadkový vektor o dĺžke maximálneho počtu iterácii(MaxIter, predvolené 30) s hodnotami jednotlivých elementov v rozmedzí 0 a 1. Pozícia prvku vo vektore zodpovedá iterácii v ktorej bude daná hodnota použitá. V príklade bolo použité na vytvorenie vektora ξ funkcia linspace, ktorá vytvorí 30 hodnôt, ktoré sú od seba rovnomerne vzdialené medzi hodnotami 0.5 a 0.9. Predvolená hodnota je linspace(0.5,0.9,MaxIter)

```
% Predstavuje hodnotu koeficienta Xi v kazdej iteracii (MT=>LLP-AUX)
% Je odporucane zacat na hodnote 0.5 a postupne ju zvysovat po hodnotu 0.9
% (vektor s dlzkou MaxIter, ktora je prednastavena na 50)
Xi = linspace(0.5,0.9,30);
```

MT algoritmus reformuluje ohraničenia pomocou binárnych premenných z pre podproblémy GSIP-LBD 2.17 a GSIP-UBD 2.19. V tejto transformácii je možné určiť nadhodnotenie maximálnej pozitívnej hodnoty funkcií ohraničení $g_{l,j}^{max}$ a nadhodnotenie maximálnej pozitívnej hodnoty účelovej funkcie problému nižšej úrovne g_u^{max}. Predvolené hodnoty sú $g_{l,j}^{max} = 30$ a $g_u^{max} = 35$

```
% Nadhodnotenie maximalnej pozitivnej hodnoty funkcii ohraniceni
% (kladne cislo)
Gl_max  = 25;
% Nadhodnotenie maximalnej pozitivnej hodnoty ucelovej funkcie
% problemu nizsej urovne
% (kladne cislo)
Gu_max  = 20;
```

Pri použití MT algoritmu je možné nastavovať reštrikcie pravej strany ohraničení ε_L a ε_U , ktoré GSIP-UBD 2.18a pomáhajú vytvárať hornú ohranicu optimalizovaného problému. Tieto ohraničenia sa zmenšujú s počtom iterácii a faktor ktorým sú delené je tiež možné nastaviť pod premennou ε_R . Predvolené hodnoty týchto parametrov sú

rovné 2.

```
% Kladna hodnota restrikcie pravej strany ohraniceni pouzita "MT"
    strategiou
% pri ziskavani hornej hranice GSIP
% (kladne cislo)
Eps_L   = 1.5;
Eps_U   = 1;
% Eps_R je faktor, ktorym sa Eps_L a Eps_U znizuju
% (1<Eps_R)
Eps_R   = 2;
```

Pre iteračné algoritmy ako je BF a MT je výhodné mať možnosť zadefinovať rozdielny maximálny čas behu BARON-u v jednotlivých iteráciách. Pomocou definovania premennej BarRuntime je možné funkciu informovať koľko času má BARON na riešenie jednotlivých problémov v konkrétnej iterácii. BarRuntime je riadkový vektor o dĺžke MaxIter(predvolené 30), kde jednotlivé elementy vektora predstavujú časy v sekundách vyhradené na riešenie problémov. Pozícia elementu určuje v ktorej iterácii bude daná hodnota použitá. V prípade nesprávneho definovanie(vektor je kratší alebo dlhší ako MaxIter) bude použitá hodnota z premennej Baronset, MaxTime. V prípade že baronset nebol definovaný je použité štandardné nastavenie BARON-u, ktoré vo verzii BARON-u použitej pri tvorbe týchto funkcií má hodnotu MaxTime rovné 1000 sekúnd na jeden beh.

```
% Maximalny cas behu BARON-u pre kazdu iteraciu, ak nie je nastaveny
% pouzije sa cas z Baronset-u. BarRuntime je vektor s dlzkou
% MaxIter(zakladna hodnota 50). Elementy tohoto vektora definuju
    maximalny cas behu
% BARON-u v sekundach v jednotlivych iteraciach.
% (riadkovy vektor)
BarRuntime = [ones(1,8)*200,ones(1,10)*600,ones(1,12)*1800]
```

Posledným voliteľným argumentom je Baronset. Baronset predstavuje štruktúru s mnohými nastavenia pre riešiteľ BARON. Keďže týchto nastavení je veľa je odporučané prečítať si dokumentáciu o jednotlivých možnostiach BARON-u.

```
% Priklad definicie Baronset struktury
% (pre viacej informacii o nastaveniach baronset
%  je vhodne nahliadnut do dokumentacie BARON-u,
%  alebo napisat baronset v MATLABE-e )
Baronset = baronset('MaxTime',300,'threads',8);
```

Po zadefinovaní všetkých povinných a v tomto prípade aj voliteľných argumentom nasleduje zavolanie funkcie:

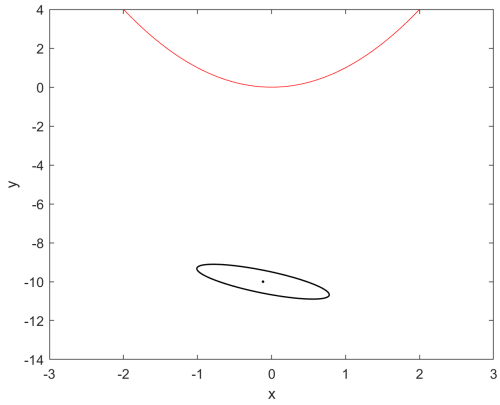
```
% Volanie funkcie
[J_opt, lv_opt, uv_opt, Log] = bi_level(obj_fun, obj_LLP, const,...
lv, uv, lb, ub, start, strategy, 'MaxIter', MaxIter, 'Tol', Tol,...
'Xi', Xi,'Gl_max', Gl_max, 'Gu_max', Gu_max, 'Eps_L', Eps_L,...
'Eps_R', Eps_R , 'Eps_U' ,Eps_U,'BarRuntime', BarRuntime,...
 'Baronset', Baronset)
```

4.1.1.3 Výsledky

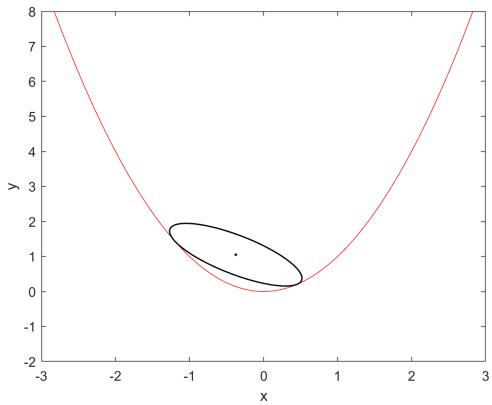
V nasledovnej tabuľke sú uvedené výsledky získané použitím jednotlivých stratégii.

Stratégia:	KKT	BF	MT
$\begin{pmatrix} x \\ y \end{pmatrix}$	$\begin{pmatrix} -0.1147 \\ -10 \end{pmatrix}$	$\begin{pmatrix} -0.3750 \\ 1.05 \end{pmatrix}$	$\begin{pmatrix} -0.3750 \\ 1.0499 \end{pmatrix}$
$\begin{pmatrix} v \\ w \end{pmatrix}$	$\begin{pmatrix} -0.4567 \\ -0.1661 \end{pmatrix}$	$\begin{pmatrix} 0.8310 \\ -0.8420 \end{pmatrix}$	$\begin{pmatrix} -0.8311 \\ 0.4046 \end{pmatrix}$
počet iterácii	—	5	18
celkový čas riešenia [s]	0.32	2.8	52.5

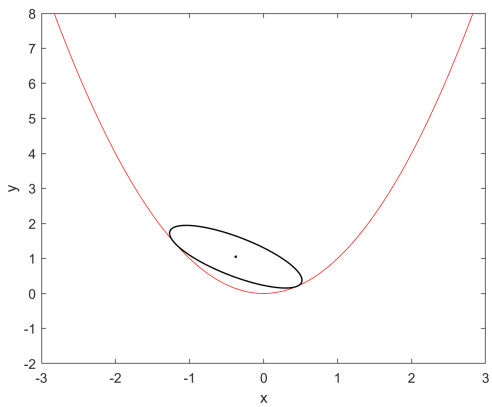
Tabuľka 4.1: Výsledky pre problém umiestnenia elipsy



(a) Umiestnenie elipsy pomocou KKT



(b) Umiestnenie elipsy pomocou BF



(c) Umiestnenie elipsy pomocou MT

Obr. 4.4: Grafické znázornenie výsledkov pre problém umiestnenia elipsy

Pomocou jednotlivých stratégii sme získali výsledky, ktoré sú zobrazené na skupine obrázkov 4.4. Algoritmy BF a MT dosiahli skoro rovnaké výsledky v ktorých úspešne umiestnili elipsu presne nad kvadratické ohraničenie. Na prvý pohľad sa javí že metódy BF a MT majú rozdielne výsledky v premenných nižšej úrovne, ale v prípade, že sa pozrieme do logov MT stratégie konkrétne Log.LBD.LLP tak zistíme, že v predchádzajúcej iterácii boli hodnoty premenných nižšej úrovne:

$$\begin{pmatrix} v \\ w \end{pmatrix} = \begin{pmatrix} 0.8310 \\ -0.8420 \end{pmatrix}$$

Čo sú rovnaké hodnoty, aké dosiahla BF stratégia v poslednej iterácii. Tento jav je spôsobený tým, že ako BF tak MT si v probléme hornej úrovne (pre MT GSIP) držia v rámci ohraničení všetky hodnoty premenných nižšej úrovne a teda po niekoľkých iteráciách problém nižšej úrovne(pre MT LLP alebo LLP-AUX) už dáva len výsledky mierne odlišné od predchádzajúcich.

Metóda založená na KKT-podmienkach nedosiahla vyhovujúce výsledky, keďže elipsa je evidentne umiestnená pod ohraničením a teda ho porušuje.

4.2 Problémy riešené bi_level_OED funkciou

Na nasledujúcich stranách je možné nájsť príklady riešenia problémov optimálneho návrhu experimentu, založených na diskrétnych a statických modeloch.

4.2.1 Optimálny návrh experimentu založený na statickom modely

Na ukážku riešenia optimálneho návrhu experimentu pre statické modely sme si vybrali jednoduchý model s dvoma parametrami p_1 a p_2.

$$x = (1 - e^{-p_2 u}) \tag{4.2a}$$
$$y = p_1 x \tag{4.2b}$$

Na základe tohoto modelu funkcia bi_level_OED najprv vytvorí maticu spodných a horných hraníc parametrov Π 4.3.

$$\Pi = \begin{bmatrix} p_1^L & p_{2,p_1^L} \\ p_1^U & p_{2,p_1^U} \\ p_{1,p_2^L} & p_2^L \\ p_{1,p_2^L} & p_2^U \end{bmatrix} \tag{4.3}$$

Zaujímavé sú len hodnoty parametrov p_1^L,p_1^U,p_2^L a p_2^U, ktoré predstavujú horné a spodné hranice. Pridružené premenné p_{2,p_1^L},p_{2,p_1^U},p_{1,p_2^L} a p_{1,p_2^U} sú len doplnkové. V modely sa nachádza aj premenná u, ktorá predstavuje akčný zásah, aj keď v tomto konkrétnom statickom modely môže byť interpretovaná ako čas merania.

Následne sa definuje vyššia a nižšia úroveň optimalizačného problému a ohraničenia. Riešený optimalizačný problém má potom nasledovný tvar:

$$\min_{\boldsymbol{u}} \; p_1^U - p_1^L + p_2^U - p_2^L \tag{4.4a}$$

$$\max_{\boldsymbol{\Pi}} \; p_1^U - p_1^L + p_2^U - p_2^L \tag{4.4b}$$

$$s.t. \sum_{k=1}^{n} \frac{(\mathbf{y}(p_1^L, p_{2,p_1^L}, \mathbf{u}_k) - \mathbf{y_m}(\mathbf{u}_k))^2}{\sigma^2} \leq \chi^2_{\alpha,n_p} \tag{4.4c}$$

$$\sum_{k=1}^{n} \frac{(\mathbf{y}(p_1^U, p_{2,p_1^U}, \mathbf{u}_k) - \mathbf{y_m}(\mathbf{u}_k))^2}{\sigma^2} \leq \chi^2_{\alpha,n_p} \tag{4.4d}$$

$$\sum_{k=1}^{n} \frac{(\mathbf{y}(p_{1,p_2^L}, p_2^L, \mathbf{u}_k) - \mathbf{y_m}(\mathbf{u}_k))^2}{\sigma^2} \leq \chi^2_{\alpha,n_p} \tag{4.4e}$$

$$\sum_{k=1}^{n} \frac{(\mathbf{y}(p_{1,p_2^L}, p_2^U, \mathbf{u}_k) - \mathbf{y_m}(\mathbf{u}_k))^2}{\sigma^2} \leq \chi^2_{\alpha,n_p} \tag{4.4f}$$

Ohraničenia spodnej úrovne sú tvorené funkciami príslušnosti, ktoré keď platia tak daná hodnota parametra sa pri danom vektore časov $\boldsymbol{u}$ nachádza vo vnútri alebo na hranici regiónu spoľahlivosti. y a y_m sú vektory reprezentujúce odhadovaný model a merania simulované pomocou dodaných hodnôt parametrov. V prípade, že vektor akčných zásahov $\boldsymbol{u}$ má dĺžku 4 vyzerá $\boldsymbol{y_m}$ nasledovne:

$$\boldsymbol{y_m} = \begin{bmatrix} p_1(1 - e^{-p_2 u_1}) \\ p_1(1 - e^{-p_2 u_2}) \\ p_1(1 - e^{-p_2 u_3}) \\ p_1(1 - e^{-p_2 u_4}) \end{bmatrix} \tag{4.5a}$$

kde $p_1 = 2.5$ a $p_2 = 0.5$.

Hodnoty výstupu $\boldsymbol{y}$ získané pomocou odhadovaných parametrov p_1^L a p_{2,p_1^L} v prípade, že vektor akčných zásahov $\boldsymbol{u}$ má dĺžku 4 sú definované nasledovne:

$$\boldsymbol{y} = \begin{bmatrix} p_1^L(1 - e^{-p_{2,p_1^L} u_1}) \\ p_1^L(1 - e^{-p_{2,p_1^L} u_2}) \\ p_1^L(1 - e^{-p_{2,p_1^L} u_3}) \\ p_1^L(1 - e^{-p_{2,p_1^L} u_4}) \end{bmatrix} \tag{4.6a}$$

4.2.1.1 Definovanie vstupných parametrov

V nasledujúcej ukážke programu je kompletná definícia všetkých povinných aj voliteľných argumentov potrebných pre riešenie optimálneho návrhu experimentu založeného na statickom modely 4.4a.

Najprv je nutné zadefinovať symbolické premenné. Pre model definujeme dva parametre p_1 a p_2 a napriek tomu, že chceme použiť funkciu na statický optimalny dizajn experimentu je nutné definovať počiatočnú podmienku v podobe premennej x_0 a jej hodnoty x0_value = 0 aj keď nebude v tomto prípade použitá.

```
% Definuj symbolicke premenne
syms p1 p2 x0
% Definuj hodnotu pociatocnej podmienky
x0_value = 0;
```

Následne si zadefinujeme vektor akčných zásahov $\boldsymbol{u}$ ako symbolický vektor, stavový model procesu v premennej model_x (môže byť použitý aj iný názov), ktorý musí obsahovať prvý prvok vektora akčných zásahov $\boldsymbol{u}$ a výstupný model v premennej model_y, ktorá musí obsahovať odkaz na stavovú rovnicu v tomto prípade model_x.

```
% Definuj pocet krokov vybratim dlzky vektora casu u
% (stlpcovy vektor)
u = sym('u',[4 1]);
% Definuj stavovy model
% (symbolicka funkcia)
model_x = 1-exp(-p2*u(1));
% Definuj vystupny model
% (symbolicka funkcia)
model_y = p1*model_x;
```

Po definovaní modelu je nutné zadefinovať parametre do premennej parameters pre ktoré sa bude hľadať región spoľahlivosti a ich ideálnu hodnotu do premennej parameters_values na základe ktorej sa bude počítať $\boldsymbol{y_m}$ 4.5

```
% Definuj parametrov modelu
% (urcuje poradie v ktorom su zobrazene vysledky)
% (riadkovy symbolicky vektor)
parameters = [p1,p2];
% Definuj hodnoty parametrov modelu
% (hodnoty su naviazane na premenne na zaklade poradia)
% (riadkovy vektor)
parameters_values = [2.5,0.5];
```

Podobne ako vo funkcii bi_level aj tu je nutné definovať spodné(lb) a horné(ub) hranice všetkých parametrov. Prvý element vektorov definuje hranice akčných zásahov, nasledujú hranice pre parametre. Program priraďuje hranice k parametrom na základe pozície vo vektore parameters.

```
% Definuj spodne hranice hodnot pre vektor u a parametre
% format lb=[lb_u,lb_parameter1,lb_parameter2,...,lb_parameter(last)]
% (riadkovy vektor)
lb = [0 0 0];
% Definuj horne hranice hodnot pre vektor u a parametre
% format ub=[ub_u,ub_parameter1,ub_parameter2,...,ub_parameter(last)]
% (riadkovy vektor)
ub = [20 10 10];
```

Pre algoritmus MT sú potrebné štartovacie body pre hodnoty jednotlivých dvojíc parametrov. V prípade dvoch parametrov vznikajú 4 dvojice, horná a dolná hranica p_1 a k nim patriace p_2 a horná a dolná hranica p_2 a k nim patriace p_1. Pri definícii štartovacích bodov sa neodporúča používať 0, pretože môže dôjsť ku deleniu nulou čo spôsobí pád programu.

```
% Urci startovacie hodnoty parametrov, pri inicializacii parametrov na
% nulove hodnoty moze vyskocit chyba delenie nulov, preto je
% odporucane nepouzivat nulu. (MT)
% format:
%   [parameter1_low parameter_2 ... parameter_last   ]
%   [parameter1_up parameter_2  ... parameter_last   ]
%   [parameter1   parameter2_low ... parameter_last ]
%   [parameter1   parameter2_up ... parameter_last  ]
%   [    .              .                .          ]
%   [    .                 .             .          ]
%   [    .                     .         .          ]
%   [    .                        .      .          ]
%   [parameter1   parameter2    ... parameter_last_low ]
%   [parameter1   parameter2    ... parameter_last_up ]
start = [ 0.1 0.1
        10   0.1
         0.1 0.1
        10   0.1];
```

V tomto príklade pracujeme so statickým modelom, preto do premennej model_type vložíme reťazec “static”.

```
% Vyber typ modelu diskretny alebo staticky (discrete/static)
% (retazec)
model_type = "static";
```

Posledným povinným argumentom je výber stratégie, ktorú si môžeme zvoliť napríklad “MT”.

```
% Definuj strategiu riesenia problemu
% (retazec)
Strategy = "MT"
```

Prvým voliteľným argumentom je podobne ako pri bi_level funkcii možnosť určiť maximálny počet iterácií pre MT stratégiu. Predvolená hodnota je 30.

```
% Urci maximalny pocet iteracii (MT)
% (cislo)
MaxIter = 30;
```

Definíciou α je možné meniť hodnotu horného α kvantilu χ^2_{α,n_p} rozdelenia pravdepodobnosti, kde α predstavuje pravdepodobnosť danú natívnej MATLAB funkcii chi2inf na výpočet χ^2_{α,n_p}. χ^2_{α,n_p} a tým pádom aj α sa vyskytuje vo funkciách príslušnosti všetkých podproblémov 2.23, 2.25 a 2.26. Predvolená hodnota je 0.95.

```
% Alpha je pravdepodobnost dana funkcii chi2inf na vypocet horneho alfa
% kvantilu Chi^2 rozdelenia pravdepodobnosti
% (cislo)
Alpha   = 0.95;
```

Funkcia ponúka aj možnosť meniť štandardnú odchýlku šumu merania pod premennou σ, ktorej predvolená hodnota je 0.1. Podobne ako α aj σ sa vyskytuje vo funkciách príslušnosti všetkých podproblémov 2.23, 2.25 a 2.26. Predvolená hodnota je 0.1.

```
% Standardna odchylka sumu merania
% (cislo)
Sigma   = 0.1;
```

Premenná Tol predstavuje prísnosť ukončovecej podmienky pre MT algoritmus. Predvolená hodnota je $1e-3$

```
% Tol urcuje prisnost uknocovacej podmienky pre algoritmi MT
% (male cislo)
Tol     = 1e-4;
```

Vektor ξ a premenná BarRuntime majú rovnakú funkciu, formát a aj predvolenú hodnotu ako v prípade bi_level funkcie.

```
% Predstavuje hodnotu koeficienta Xi v kazdej iteracii (MT=>LLP-AUX)
% Je odporucane zacat na hodnote 0.5 a postupne ju zvysovat po hodnotu 0.9
% (vektor s dlzkou MaxIter, ktora je prednastavena na 50)
% to value of 0.9
Xi      = linspace(0.5,0.9,30);
% Maximalny cas behu BARON-u pre kazdu iteraciu, ak nie je nastaveny
% pouzije sa cas z Baronset-u. BarRuntime je vektor s dlzkou
% MaxIter(zakladna hodnota 50). Elementy tohoto vektora definuju
    maximalny cas behu
% BARON-u v sekundach v jednotlivych iteraciach.
% (riadkovy vektor)
BarRuntime = [ones(1,8)*200,ones(1,10)*1800,ones(1,12)*3600]
```

Pre MT algoritmus je podobne ako v prípade bi_level funkcie, možné definovať hodnoty maximálneho pozitívneho nadhodnotenia účelovej funkcie nižšej úrovne g_u^{max} a funkcií ohraničení $g_{l,j}^{max}$ 2.24. Predvolené hodnoty sú $g_{l,j}^{max} = 30$ a $g_u^{max} = 35$

```
% Nadhodnotenie maximalnej pozitivnej hodnoty ohraniceni
% problemu nizsej urovne (MT)
% (cislo)
Gl_max  = 5;
% Nadhodnotenie maximalnej pozitivnej hodnoty ucelovej funkcie obj_LLP
    (MT)
% (cislo)
Gu_max  = 20;
```

Posledným voliteľným argumentom je štruktúra Baronset.

```
% Priklad definicie Baronset struktury
% (pre viacej informacii o nastaveniach baronset
%  je vhodne nahliadnut do dokumentacie BARON-u,
%  alebo napisat baronset v MATLABE-e )
Baronset = baronset('MaxTime',10800,'threads',8,'EpsA',1e-2 );
```

Po definovaní všetkých povinných a v tomto prípade aj voliteľných argumentov je možné funkciu bi_level_OED zavolať nasledovným príkazom:

```
[J_opt, U_opt, Param_opt, Log] = bi_level_OED(model_x,model_y,...
u, parameters, parameters_values, x0, x0_value, lb, ub, start,...
model_type, strategy,'MaxIter', MaxIter, 'Alpha', Alpha,...
'Sigma', Sigma, 'Tol', Tol, 'Xi', Xi, 'Gl_max', Gl_max,...
'Gu_max', Gu_max, 'BarRuntime', BarRuntime,'Baronset', Baronset)
```

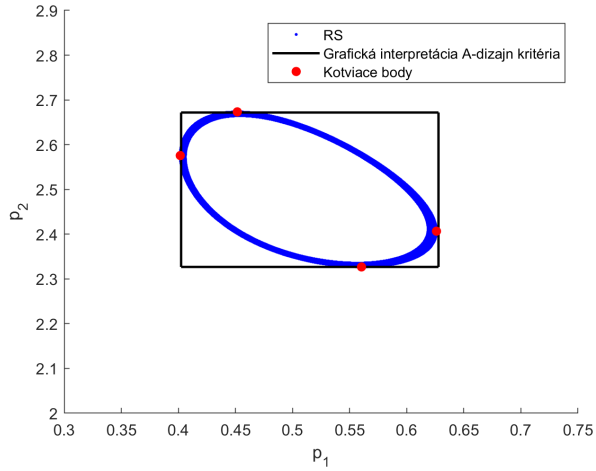
4.2.1.2 Výsledky

V nasledujúcej tabuľke je možné vidieť dosiahnuté výsledky.

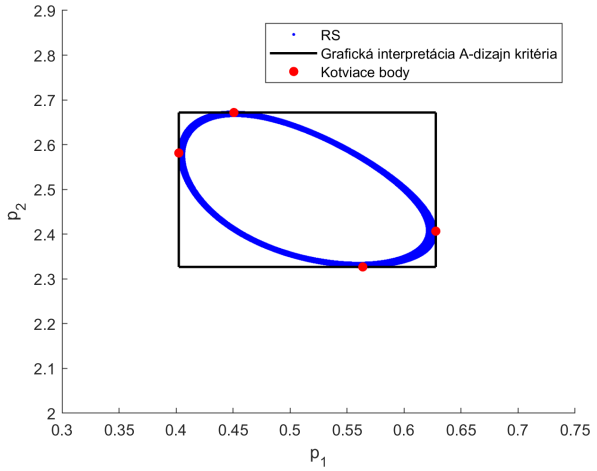
Stratégia:	KKT	MT
Hodnota účelovej funkcie	0.5708	0.5704
$\boldsymbol{u}$	$\begin{pmatrix} 1.6289 \\ 20.0000 \\ 1.6289 \\ 20.0000 \end{pmatrix}$	$\begin{pmatrix} 1.7620 \\ 1.8223 \\ 19.8706 \\ 19.8141 \end{pmatrix}$
$\begin{pmatrix} p_1^L \\ p_1^U \\ p_2^L \\ p_2^U \end{pmatrix}$	$\begin{pmatrix} 2.3268 \\ 2.6733 \\ 0.4017 \\ 0.6260 \end{pmatrix}$	$\begin{pmatrix} 2.3268 \\ 2.6718 \\ 0.4024 \\ 0.6278 \end{pmatrix}$
počet iterácii	—	14
celkový čas riešenia	120 min čas vypršal	61.95 min výsledok nájdený

Tabuľka 4.2: Výsledky optimálneho navrhu experimentu s statickým modelom

Ako si môžeme všimnúť v prípade statického modelu rozdiel v hodnotách účelovej funkcie je veľmi malý. Zaujímavé je, že vektory časov $\boldsymbol{u}$ nadobudnuté jednotlivými funkciami sa výrazne líšia a napriek tomu ich vplyv na hodnoty parametrov je približne rovnaký, to nám podáva informáciu o tom že tento problém je numericky veľmi citlivý.



Obr. 4.5: Región spoľahlivosti pre statický model získaný pomocou KKT programu



Obr. 4.6: Región spoľahlivosti pre statický model získaný pomocou MT algoritmu

Výsledkom riešenia problému optimálneho návrhu experimentu je región spoľahlivosti. Získané regióny spoľahlivosti pomocou KKT metódy a MT algoritmu je možné vidieť na obrázkoch 4.5 a 4.6. Kotviace body získané programom sú zaznačené červenou farbou, región spoľahlivosti pri danom vektore časov $\boldsymbol{u}$ je označený modrou a na základe kotviacich bodov je zostrojená grafická reprezentácia A-dizajn kritéria, ktorá je označená čiernou farbou.

4.2.2 Optimálny návrh experimentu založený na diskrétnom modely

Diskrétny model použitý na predvedenie schopností funkcie bi_level_OED vyzerá nasledovne:

$$x = \frac{1}{\gamma + x_0^2} + u \tag{4.7a}$$

$$y = \beta x \tag{4.7b}$$

Na základe tohoto modelu funkcia bi_level_OED rovnako ako v prípade statického

modelu vytvorí maticu spodných a horných hraníc parametrov Π 4.3.

$$\Pi = \begin{bmatrix} \gamma^L & \beta_{\gamma^L} \\ \gamma^U & \beta_{\gamma^U} \\ \gamma_{\beta^L} & \beta^L \\ \gamma_{\beta^L} & \beta^U \end{bmatrix} \tag{4.8}$$

Na získanie regiónu spoľahlivosti sú potrebné parametre γ^L, γ^U, β^L a β^U, ktoré ohraničujú región spoľahlivosti. Doplnkové premenné β_{γ^L}, β_{γ^U}, γ_{β^L} a γ_{β^U} môžu by použité na vykreslenie jednotlivých kotviacich bodov.

Program následne automaticky transformuje poskytnutý model na dvojúrovňový optimalizačný problém, ktorý vyzerá nasledovne:

$$\min_{\boldsymbol{u}} \ \gamma^U - \gamma^L + \beta^U - \beta^L \tag{4.9a}$$

$$\max_{\boldsymbol{\Pi}} \ \gamma^U - \gamma^L + \beta^U - \beta^L \tag{4.9b}$$

$$s.t. \ \sum_{k=1}^{n} \frac{(\mathbf{y}(\gamma^L, \beta_{\gamma^L}, \mathbf{u}_k) - \mathbf{y_m}(\mathbf{u}_k))^2}{\sigma^2} \leq \chi^2_{\alpha,n_p} \tag{4.9c}$$

$$\sum_{k=1}^{n} \frac{(\mathbf{y}(\gamma^U, \beta_{\gamma^U}, \mathbf{u}_k) - \mathbf{y_m}(\mathbf{u}_k))^2}{\sigma^2} \leq \chi^2_{\alpha,n_p} \tag{4.9d}$$

$$\sum_{k=1}^{n} \frac{(\mathbf{y}(\gamma_{\beta^L}, \beta^L, \mathbf{u}_k) - \mathbf{y_m}(\mathbf{u}_k))^2}{\sigma^2} \leq \chi^2_{\alpha,n_p} \tag{4.9e}$$

$$\sum_{k=1}^{n} \frac{(\mathbf{y}(\gamma_{\beta^L}, \beta^U, \mathbf{u}_k) - \mathbf{y_m}(\mathbf{u}_k))^2}{\sigma^2} \leq \chi^2_{\alpha,n_p} \tag{4.9f}$$

Ohraničenia spodnej úrovne sú tvorené funkciami príslušnosti, ktoré keď platia tak daná hodnota parametra sa pri danom vektore akčných zásahov $\boldsymbol{u}$ nachádza vo vnútri alebo na hranici regiónu spoľahlivosti. $\boldsymbol{y}$ a $\boldsymbol{y_m}$ sú vektory reprezentujúce odhadovaný model a merania simulované pomocou dodaných hodnôt parametrov. V prípade, že vektor akčných zásahov $\boldsymbol{u}$ má dĺžku 4 vyzerá $\boldsymbol{y_m}$ nasledovne:

$$\boldsymbol{y_m} = \begin{bmatrix} \beta(\frac{1}{\gamma+x_0^2} + u) \\ \beta(\frac{1}{\gamma+(\frac{1}{\gamma+x_0^2}+u_1)^2} + u_2) \\ \beta(\frac{1}{\gamma+(\frac{1}{\gamma+(\frac{1}{\gamma+x_0^2}+u_1)^2}+u_2)^2} + u_3) \\ \beta(\frac{1}{\gamma+(\frac{1}{\gamma+(\frac{1}{\gamma+(\frac{1}{\gamma+x_0^2}+u_1)^2}+u_2)^2}+u_3)^2} + u_4) \end{bmatrix} \tag{4.10a}$$

kde $\gamma = 2$ a $\beta = 0.8$

$\boldsymbol{y}$ pre parametre γ^L a β_{γ^L} má potom podobu:

$$\boldsymbol{y} = \begin{bmatrix} \beta_{\gamma^L}(\frac{1}{\gamma^L + x_0^2} + u) \\ \beta_{\gamma^L}(\frac{1}{\gamma^L + (\frac{1}{\gamma^L + x_0^2} + u_1)^2} + u_2) \\ \beta_{\gamma^L}(\frac{1}{\gamma^L + (\frac{1}{\gamma^L + (\frac{1}{\gamma^L + x_0^2} + u_1)^2} + u_2)^2} + u_3) \\ \beta_{\gamma^L}(\frac{1}{\gamma^L + (\frac{1}{\gamma^L + (\frac{1}{\gamma^L + (\frac{1}{\gamma^L + x_0^2} + u_1)^2} + u_2)^2} + u_3)^2} + u_4) \end{bmatrix} \tag{4.11a}$$

4.2.2.1 Definovanie vstupných parametrov

V nasledujúcej ukážke programu je kompletná definícia všetkých povinných aj voliteľných argumentov potrebných pre riešenie optimalneho návrhu experimentu založeného na diskrétnom modeli 4.9a. Pre diskrétne modely je odporúčané používať MT stratégiu, keďže vytvorenie gradientu Lagrangeovej funkcie 2.11b je výpočtovo náročné a aj pri úspešnom vytvorení gradientu, môže dôjsť ku chybe pri transformácii týchto ohraničení do pracovného formátu BARON-u, čo môže spôsobiť pád programu.

Mnohé premenné sú zadefinované úplne rovnakým spôsobom ako v prípade optimalneho návrhu experimentu so statickým modelom 4.2.1.1.

Najprv je nutné definovať symbolické premenné parametrov a počiatočnej podmienky x_0, keďže ideme použiť diskrétny model tak počiatočná podmienka a jej hodnota je riadne použitá programom v rovniciach 4.10 a 4.11.

```
% Definuj symbolicke premenne
syms gama, beta, x0
% Definuj hodnotu pociatocnej podmienky
x0_value = 0;
```

Vektor akčných zásahov $\boldsymbol{u}$ a výstupný model model_y sú definované podobný spôsobom ako pri statickom modely. Stavovový model model_x v sebe musí mať prvý prvok vektora akčných zásahov $\boldsymbol{u}$ a počiatočnú podmienky x_0

```
% Definuj pocet krokov vybratim dlzky vektora casu u
% (stlpcovy vektor)
u = sym('u',[4 1]);
% Definuj stavovy model
% Vo stavovom modely sa musi nachadzat prvy prvok vektora u
% (symbolicka funkcia)
model_x = 1/(gama+x0^2)+u(1);
% Definuj vystupny model
% (symbolicka funkcia)
model_y = beta*model_x;
```

Rovnako je nutné definovať programu, ktoré premenné sú parametrami a ich hodnotu pre y_m 4.10.

```
% Definuj parametrov modelu
% (urcuje poradie v ktorom su zobrazene vysledky)
% (riadkovy symbolicky vektor)
parameters = [beta,gama];
% Definuj hodnoty parametrov modelu
% (hodnoty su naviazane na premenne na zaklade poradia)
% (riadkovy vektor)
parameters_values = [2,0.8];
```

BARON si vyžaduje mať definované horné a dolné hranice všetkých optimalizovaných parametrov.

```
% Definuj spodne hranice hodnot pre vektor u a parametre
% format lb=[lb_u,lb_parameter1,lb_parameter2,...,lb_parameter(last)]
% (riadkovy vektor)
lb = [-1 0 0];
% Definuj horne hranice hodnot pre vektor u a parametre
% format ub=[ub_u,ub_parameter1,ub_parameter2,...,ub_parameter(last)]
% (riadkovy vektor)
ub = [1 10 10];
```

MT stratégia si vyžaduje štartovacie hodnoty dvojíc parametrov fo formáte 2.3

```
% Urci startovacie hodnoty parametrov, pri inicializacii parametrov na
% nulove hodnoty moze vyskocit chyba delenie nulov, preto je
% odporucane nepouzivat nulu. (MT)
% format:
%   [parameter1_low parameter_2 ... parameter_last   ]
%   [parameter1_up parameter_2  ... parameter_last   ]
%   [parameter1    parameter2_low ... parameter_last ]
%   [parameter1    parameter2_up ... parameter_last  ]
%   [     .              .                .          ]
%   [     .                 .             .          ]
%   [     .                    .          .          ]
%   [     .                       .       .          ]
%   [parameter1    parameter2    ... parameter_last_low ]
%   [parameter1    parameter2    ... parameter_last_up ]
start = [1.1322 0.6811
        4.2460 0.9244
        1.3256 0.6377
        3.3167 0.9837];
```

V tomto príklade pracujeme s diskrétnym modelom a tak model_type má hodnotu "discrete"

```
% Vyber typ modelu diskretny alebo staticky (discrete/static)
% (retazec)
model_type = "discrete";
```

Pre diskrétne modely je odporúčané používať MT stratégiu, pretože KKT môže mať problémy spôsobené tvorbou a prekladom gradientu Lagrangeovej funkcie do BARON-u.

```
% Definuj strategiu riesenia problemu
% (retazec)
Strategy = "MT"
```

Voliteľné argumenty sú rovnaké ako v prípade statického modelu.

```
% Urci maximalny pocet iteracii (MT and BF)
% (cislo)
MaxIter = 30;
% Alpha je pravdepodobnost dana funkcii chi2inf na vypocet horneho alfa
% kvantilu Chi^2 rozdelenia pravdepodobnosti
% (cislo)
Alpha   = 0.95;
% Standardna odchylka sumu merania
% (cislo)
Sigma   = 0.1;
% Tol urcuje prisnost uknocovacej podmienky pre algoritmi BF a MT
% (male cislo)
Tol     = 1e-4;
% Predstavuje hodnotu koeficienta Xi v kazdej iteracii (MT=>LLP-AUX)
% Je odporucane zacat na hodnote 0.5 a postupne ju zvysovat po hodnotu 0.9
% (vektor s dlzkou MaxIter, ktora je prednastavena na 50)
Xi      = linspace(0.5,0.9,30);
% Nadhodnotenie maximalnej pozitivnej hodnoty
% funkcii ohraniceni (MT)
% (cislo)
Gl_max  = 5;
% Nadhodnotenie maximalnej pozitivnej hodnoty
% ucelovej funkcie obj_LLP (MT)
% (cislo)
Gu_max  = 20;
% Elementy tohoto vektora definuju maximalny cas behu
% BARON-u v sekundach v jednotlivych iteraciach, ak nie je nastaveny
% pouzije sa cas z Baronset-u.
% (riadkovy vektor s dlzkou MaxIter(prednastavena 50))
BarRuntime = [ones(1,8)*200,ones(1,10)*300,ones(1,12)*1200]
% Priklad definicie Baronset struktury
% (pre viacej informacii o nastaveniach baronset
%  je vhodne nahliadnut do dokumentacie BARON-u,
%  alebo napisat baronset v MATLABE-e )
Baronset = baronset('MaxTime',10800,'threads',8,'EpsA',1e-2 );
```

Po zadefinovaní všetkých povinných a v našom príklade aj voliteľných argumentov môžeme funkciu zavolať nasledovným príkazom.

```
[J_opt, U_opt, Param_opt, Log] = bi_level_OED(model_x,model_y,u,...
parameters, parameters_values,x0,x0_value,lb,ub,start,model_type,...
strategy,'MaxIter',MaxIter,'Alpha',Alpha,'Sigma',Sigma,'Tol',Tol,...
'Xi',Xi,'Gl_max',Gl_max,'Gu_max',Gu_max,'BarRuntime',BarRuntime,...
'Baronset',Baronset)
```

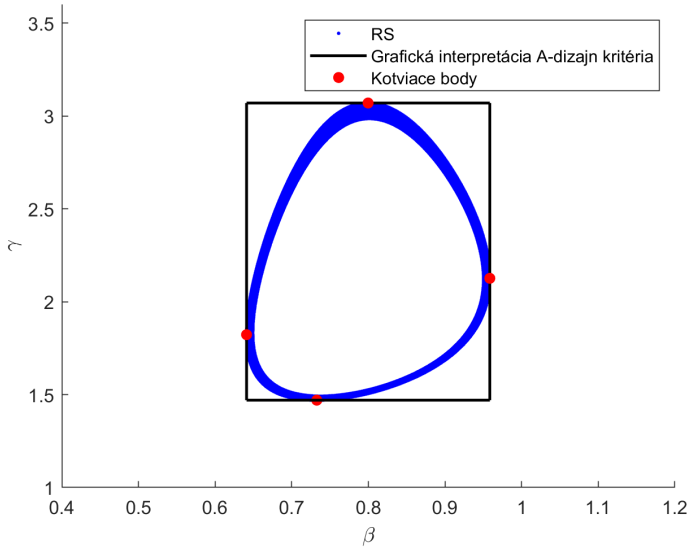
4.2.2.2 Výsledky

Napriek tomu že použitý diskrétny model je silne nelineárny Mitsos-Tsoukalasov algoritmus bol schopný v 15 iteráciách, dosiahnuť optimálny návrh experimentu.

Stratégia:	MT
Hodnota účelovej funkcie	1.9175
$\boldsymbol{u}$	$\begin{pmatrix} -0.3395 \\ -0.9243 \\ -0.8965 \\ 1 \end{pmatrix}$
$\begin{pmatrix} \gamma_L \\ \gamma_U \\ \beta_L \\ \beta_U \end{pmatrix}$	$\begin{pmatrix} 1.4701 \\ 3.0702 \\ 0.6411 \\ 0.9585 \end{pmatrix}$
počet iterácii	15
celkový čas riešenia [min]	53.8

Tabuľka 4.3: Výsledky optimálneho navrhu experimentu s diskrétnym modelom

Výsledky získané MT-algoritmom sú zhrnuté v tabuľke 4.3. Na nasledovnom obrázku si môžeme všimnúť grafickú reprezentáciu regiónu spoľahlivosti, kotviacich bodov A-dizajn kritéria.



Obr. 4.7: Región spoľahlivosti pre diskrétny model získaný pomocou MT programu

Bohužiaľ výsledky dosiahnuté KKT-metódou na porovnanie nie sú k dispozícii, pretože pri transformovaní gradientu Lagrangeovej funkcie do natívneho BARON-ovho formátu dôjde ku chybe, ktorá môže byť spôsobená náročnosťou parsovania funkcií ohraničení.

Kapitola 5

Záver

Cieľom tejto práce bolo zostrojiť funkcie bi_level a bi_level_OED v programovacom prostredí MATLAB s využitím rozhrania s globálnym riešiteľom optimalizačných problémov BARON, ktoré sú schopné riešiť vybrané všeobecné typy optimalizačných problémov a problém optimalneho návrhu experimentu založenom na diskrétnom alebo statickom modeli systému.

Funkcia bi_level poskytuje tri stratégie riešenia optimalizačných problémov a to prístup založený na pretvorení dvojúrovňového problému na jednu úroveň pomocou Karush-Kuhn-Tuckerových podmienok optimality, Blankenship-Falkov algoritmus založený na pridávaní dodatočných ohraničení do problému vyššej úrovne a Mitsos-Tsoukalasov algoritmus, ktorý predstavuje vylepšenú verziu Blankenship-Falkovho algoritmu.

Na riešenie problému optimálneho návrhu experimentu je k dispozícii funkcia bi_level_OED, ktorá na základe poskytnutého modelu vytvorí celý optimalizačný problém, ktorý je možné riešiť buď transformáciou na jednoúrovňový program Karush-Kuhn-Tuckerovými podmienkami optimality, alebo Mitsos-Tsoukalasovým algoritmom, upraveným pre potreby optimálneho návrhu experimentu.

Súčasťou práce je aj kompletná dokumentácia k poskytnutým funkciám, kde je podrobne rozobraté aké sú vstupné argumenty do jednotlivých programov a čo predstavujú pre danú funkciu, v akom formáte sú vyprodukované výsledky a čo predstavujú. V dokumentácii je možné nájsť aj stromovú štruktúru jednotlivých funkcií, ktorá poukazuje na podfunkcie, ktoré sú nevyhnutné pre fungovanie daných programov. Pre obe hlavné funkcie bola aj zvlášť vytvorená dokumentácia v angličtine vo formáte html, v ktorej je na príkladoch ukázané ako sa dajú poskytnuté funkcie použiť.

V poslednej časti sú riešené 3 ukážkové príklady dvojúrovňových problémov. Jeden sa venuje všeobecnému tvaru a dva riešia problém optimálneho návrhu experimentu s diskrétnym a statickým modelom. V prvom riešenom príklade sa snažíme umiestniť elipsu

určitých rozmerov a naklonenia nad kvadratické ohraničenie. Úspešne sa to podarilo stratégiám založeným na Blankenship-Falkovom a Mitsos-Tsoukalasovom algoritme. Použitie Karush-Kuhn-Tuckerových podmienok optimality a následná transformácia problému na jednu úroveň neviedla ku vhodným výsledkom.

Druhý príklad sa venoval optimálnemu návrhu experimentu založenom na statickom modeli procesu. Problém bol vyriešený obomi dostupnými stratégiami s veľmi podobnými výsledkami účelových funkcií aj keď s veľmi rozdielnymi výslednými vektormi akčných zásahov $\boldsymbol{u}$. Toto mohlo byť spôsobené veľkou numerickou citlivosťou problému.

V poslednom príklade bol riešený optimálny návrh experimentu založený na nelineárnom diskrétnom modely. Tento príklad bol úspešne vyriešený iba Mitsos-Tsoukalasovým algoritmom. Metóda založená na Karush-Kuhn-Tuckerových podmienkach optimality, narazila na problém pri preklade funkcie z MATLAB-u do BARON-u a teda problém nebol riešiteľný.

Výpočtová náročnosť jednotlivých metód stúpa nasledovne KKT, BF a MT avšak toto poradie je skôr orientačné než, že by malo byť pravidlom. Ako si môžeme všimnúť v prípade optimálneho návrhu experimentu so statickým modelom tam KKT metóda potrebovala výrazne viacej času ako MT, ale v prípade minimalizácie y súradnice elipsy MT metóda potrebovala viacej času ako KKT.

Balík funkcií v aktuálnom stave má stále priestor pre niekoľko zlepšení. Napríklad rozšírenie balíka o ďalšie stratégie riešenia dvojúrovňových optimalizačných problémov. Rovnako by mohlo byť zaujímavé rozšírenie štandardnej definície optimalizačných problémov použitej v tejto práci a vo funkciách o ohraničenia v tvare rovnosti pre problém nižšej úrovne a o ohraničenia v tvare nerovnosti a rovnosti pre problém vyššej úrovne. Rozšírenie problému optimálneho návrhu experimentu aj na systémy s dvoma a viacerými stavmi a výstupmi je ďalšia s možností vylepšenia tohoto balíka.

Literatúra

[1] J. W. Blankenship and J. E. Falk. Infinitely constrained optimization problems. *Journal of Optimization Theory and Applications 19 (2)*, pages 261–281, 1976.

[2] I. Constantin and M. Florian. Optimizing frequencies in a transit network: a nonlinear bi-level programming approach. *International Transactions in Operational Research*, pages 149–164, 1995.

[3] J.-P. Côté, P. Marcotte, and G. Savard. A bilevel modeling approach to pricing and fare optimization in the airline industry. journal of revenue and pricing management. *Journal of Revenue and Pricing Management*, pages 23–36, 2003.

[4] B. Y. Kara and V. Verter. Designing a road network for hazardous materials transportation. *Transportation Science*, pages 188–196, 2004.

[5] C. Kolstad and L. Lasdon. Derivative evaluation and computational experience with large bilevel mathematical programs. *Journal of Optimization Theory and Applications*, pages 485–499, 1990.

[6] L. J. LeBlanc. Mathematical programming algorithms for large scale network equilibrium and network design problems. phd thesis. *Northwestern University, Evanston, Illinois*, 1973.

[7] P. Marcotte. Network design problem with congestion effects: a case of bilevel programming. *Mathematical Programming*, pages 23–36, 1986.

[8] A. Mitsos and A. Tsoukalas. Global optimization of generalized semi-infinite programs via restriction of the right hand side. *Journal Global Optimization 61 (1)*, pages 1–17, 4 March 2018.

[9] G. Savard and J. Gauvin. The steepest descent direction for the nonlinear bilevel programming problem. *Operations Research Letters, 15*, pages 275–282, 1994.

[10] H. F. v. Stackelberg. *Market Structure and Equilibrium*. 1934.

[11] H. F. v. Stackelberg. *The theory of market economy.* Oxford University Press, 1952.

[12] L. Vicente, G. Savard, and J. Júdice. Descent approaches for quadratic bilevel programming. *Journal of Optimization Theory and Applications 81*, pages 379–399, 1994.

[13] O. Walz, H. Djelassi, A. Caspari, and A. Mitsos. Bounded-error optimal experimental design via global solution of constrained min-max program. *Computers and Chemical Engineering Volume 111*, pages 92–101, 4 March 2018.