UNIVERZITA KOMENSKÉHO V BRATISLAVE
FAKULTA MATEMATIKY, FYZIKY A INFORMATIKY

ÚLOHY KONVEXNÉHO PROGRAMOVANIA A MODELOVACÍ SYSTÉM CVX

BAKALÁRSKA PRÁCA

2012 Jakub KOVÁCS

UNIVERZITA KOMENSKÉHO V BRATISLAVE
FAKULTA MATEMATIKY, FYZIKY A INFORMATIKY

ÚLOHY KONVEXNÉHO PROGRAMOVANIA A MODELOVACÍ SYSTÉM CVX

BAKALÁRSKA PRÁCA

Študijný program:	Ekonomická a finančná matematika
Študijný odbor:	1114 Aplikovaná matematika
Školiace pracovisko:	Katedra aplikovanej matematiky a štatistiky
Vedúci práce:	RNDr. Mária Trnovská, PhD.

Bratislava 2012 **Jakub KOVÁCS**

Univerzita Komenského v Bratislave
Fakulta matematiky, fyziky a informatiky

ZADANIE ZÁVEREČNEJ PRÁCE

Meno a priezvisko študenta: Jakub Kovács
Študijný program: ekonomická a finančná matematika (Jednoodborové štúdium, bakalársky I. st., denná forma)
Študijný odbor: 1114 aplikovaná matematika
Typ záverečnej práce: bakalárska
Jazyk záverečnej práce: slovenský

Názov: Úlohy konvexného programovania a modelovací systém CVX

Cieľ: Vytvorenie praktickej príručky modelovacieho systému CVX pre úlohy konvexného programovania.

Vedúci: RNDr. Mária Trnovská, PhD.

Dátum zadania: 16.10.2011

Dátum schválenia: 27.10.2011

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

.....
študent

.....
vedúci

Poďakovanie Touto cestou by som sa chcel poďakovať svojej školiteľke RNDr. Márie Trnovskej, PhD. za ochotu, pomoc a podnetné rady pri konzultáciách. Taktiež ďakujem aj ostatným učiteľom za vedomosti nadobudnuté počas štúdia a rodine za trpezlivosť a podporu.

Abstrakt v štátnom jazyku

KOVÁCS, Jakub: Úlohy konvexného programovania a modelovací systém CVX [Bakalárska práca], Univerzita Komenského v Bratislave, Fakulta matematiky, fyziky a informatiky, Katedra aplikovanej matematiky a štatistiky; školiteľ: RNDr. Mária Trnovská, PhD., Bratislava, 2012, 48s.

CVX ako nadstavba Matlabu, je modelovací systém na konvexnú optimalizáciu. Zámerom tejto práce je vytvoriť praktickú príručku pre úlohy konvexného programovania a poukázať na možnosť širokého použitia systému. Na splnenie úlohy sme preriešili osem príkladov. Problémy sa týkajú rôznych pracovných oblastí dôležitých pre spoločnosť a život človeka, ako sú financie, numerika, medicína, inžinierstvo. Ako v práci ukážeme, modelovací systém CVX vie rýchlo a jednoducho vyriešiť aj zložitejšie optimalizačné úlohy. Na jeho použitie je však nutné správne naformulovať problém pomocou Matlabovskej syntaxe. V práci sa zoznámime so základnou skladbou špecifického syntaxu, ktoré následne rozvinieme pri príkladoch. Pri väčšine z nich je nutná aj teoretická analýza úlohy a jej následná úprava.

Kľúčové slová: Modelovací systém CVX, Matlab, Konvexná optimalizácia, Interpolácia funkcie

Abstract

KOVÁCS, Jakub: Convex optimization problems and modeling system CVX [Bachelor Thesis], Comenius University in Bratislava, Faculty of Mathematics, Physics and Informatics, Department of Applied Mathematics and Statistics; Supervisor: RNDr. Mária Trnovská, PhD., Bratislava, 2012, 48p.

CVX is Matlab based modeling system for convex optimization. Intention of this work is to create a practical guide for convex optimization problems and point to possibility of wide-spread usage of the system. To fulfil the task, we solved eight problems. These problems are related to different fields of study important for society and human life, like finance, numerical analysis, medicine, engineering. In the thesis we will show, how simply and quickly can CVX solve even more complex optimization problems. However it is necessary , to formulate problem in the right syntax. The paper explains base composition of specific syntax, which will expand with solved problems. In most of them theoretical analysis and modification is required.

Keywords: Modeling system CVX, Matlab, Convex Optimization, Interpolation

Obsah

Zoznam obrázkov

Úvod

Konvexné programovanie je podtrieda nelineárneho programovania, ktoré zjednocuje a zovšeobecňuje metódu najmenších štvorcov, lineárne programovanie a kvadratické programovanie. Numerické algoritmy, ktoré riešia konvexné úlohy, poskytujú čoraz väčšiu spoľahlivosť, presnosť a efektivitu.

Štúdiu matematiky schovanej za konvexným programovaním a algoritmov na riešenie sa ľudia venujú približne už storočie, ale v nedávnej dobe stúpla popularita tohoto odvetvia. Zaslúžil sa o to hlavne Narendra Karmarkar, keď v roku 1984 publikoval jeho metódu vnútorného bodu na riešenie problémov lineárneho programovania. Ako sa neskôr zistilo, táto nová metóda sa dá použiť aj na riešenie problémov konvexného programovania, a teda problémy programovania nad kuželmi druhého rádu, alebo semidefinitné programy možno riešiť takmer tak jednoducho ako lineárne programy.

Ďalším činiteľom bol objav väčšej miery prítomnosti konvexných optimalizačných problémov v praxi, ako sa pôvodne očakávalo. Odvtedy sa objavilo veľké množstvo aplikácií v oblastiach ako automatizované kontrolné systémy, komunikácie a siete, design elektrických obvodov, analýza dát a modelovanie, štatistika a financie. Verím, že ešte mnoho ďalších aplikácií konvexného programovania stále čaká na objavenie.

Pre široké použitie konvexného programovania, tu stále zostáva jedna prekážka, a to nutnosť odborných poznatkov potrebných na použitie. S relatívne vyspelými algoritmami pri lineárnom alebo kvadratickom programovaní môže byť problém špecifikovaný a vyriešený s menším úsilím či so základnými vedomosťami prítomných operácií. Pri všeobecnom konvexnom programovaní, to už nie je také jednoduché a pre transformáciu problému do jedného z mnohých limitovaných foriem je potrebné hlbšie poznanie problematiky. Ľudia, ktorých zameranie je aplikácia v praxi, môže byť tento proces prekážkou.

O prekonanie tejto prekážky sa ľudia snažia vytvorením modelovacích jazykov. Do jedného z nich sa pokúsime nahliadnuť pomocou riešenia viacerých úloh a ukážkou jeho univerzálnosti a relatívnej jednoduchosti rozšíriť popularitu konvexného programovania.

Zadania úloh pochádzajú z [6]. K ich riešeniu nám dopomáhala teória [7] a používateľský manuál [2]. Modelovací jazyk cvx je vybudovaný na základe špecifického jazyka, ktorý umožnuje zápis problému v matematickej forme a ponúka ich overenie, konverziu do riešiteľnej podoby a následné numerické riešenie. Autori cvx neustále zdokonaľujú.

1 Teória konvexného programovania

1.1 Optimalizácia

Úloha matematického programovania, alebo len optimalizačný problém má formu

$$\begin{aligned} \min \quad & f_0(x) \\ & f_i(x) \leq b_i, \qquad i = 1, \cdots, m. \end{aligned} \tag{1}$$

Kde vektor $x = x_1, \cdots, x_n$ je **optimalizačná premenná** problému, funkcia $f_0 : \mathbb{R}^n \longrightarrow \mathbb{R}$ je **účelová funkcia**, funkcie $f_i : \mathbb{R}^n \longrightarrow \mathbb{R},\ i = 1, \cdots, m$ sú **funkcie ohraničení** a konštanty $b_i,\ i = 1, \cdots, m$ sú ohraničenia pre tieto funkcie. Vektor x^* nazývame **optimálnym riešením** ak pre zadanú úlohu dáva najmenšiu možnú hodnotu zo všetkých možných vektorov, ktoré spĺňajú ohraničenia: pre akýkoľvek vektor z ktorý spĺňa nerovnosti $f_i \leq b_i,\ i = 1, \cdots, m$ platí $f_0(z) \geq f_0(x^*)$.

Úlohy programovania rozdeľujeme do viacerých kategórií podľa vlastností účelovej funkcie a funkcií ohraničenia. Ak všetky zúčastnené funkcie sú afínne, tak (1) nazývame úlohou lineárneho programovania. Keď problém nie je lineárny, nazývame ho **úlohou nelineárneho programovania**. [7]

1.2 Konvexná optimalizácia

Modelovací systém cvx vznikol hlavne na riešenie konvexných optimalizačných problémov. Sú to také, kde účelová funkcia aj funkcie ohraničenia sú konvexné, teda funkcie $f_0, \cdots, f_m$ spĺnajú nerovnosť

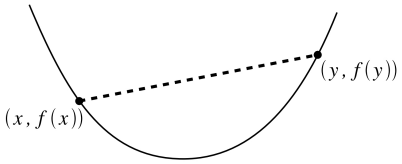
$$f_i(\alpha x + (1 - \alpha)y) \leq \alpha f_i(x) + (1 - \alpha) f_i(y) \tag{2}$$

pre všetky $x,\ y \in \mathbb{R}^n,\ \alpha \in \mathbb{R}$, kde $0 \leq \alpha \leq 1$.

Ako si môžeme všimnúť, podmienka konvexnosti je všeobecnejšia ako podmienka linearity. Keďže akýkoľvek lineárny problém je zároveň aj konvexným problémom, konvexnú optimalizáciu môžme považovať za zovšeobecnenie lineárneho programovania.

Intuitívne, keď si vyberieme 2 rôzne body na grafe konvexnej funkcie, a nakreslíme medzi nimi rovnú čiaru, časť funkcie medzi týmito bodmi zostane ležať pod čiarou. (Obr. 1)

Funkciu nazývame **rýdzo konvexnú** ak (2) platí s ostrou nerovnosťou pre $x \neq y$ a $0 < \alpha < 1$. Funkciu f nazývame **konkávnu** ak funkcia $-f$ je konvexná. [7]



Obr. 1: Graf konvexnej funkcie

Príklady konvexných funkcí:

- **Exponenxiálna funkcia**. $f(x) = e^{ax},\ a \in \mathbb{R}$.
- **Mocninové funkcie**. $f(x) = x^a,\ x \in \mathbb{R}_{++},\ a \geq 1$ alebo $a \leq 0$.
- **Záporný logaritmus**. $f(x) = -\log x,\ x \in \mathbb{R}_{++}$.
- **Afínne funkcie**. $f(x) = a^T x + b,\ a \in \mathbb{R}^n,\ b \in \mathbb{R}$. Funkcie takejto formy sú konvexné aj konkávne zároveň.
- **Kvadratické funkcie**. $f(x) = \frac{1}{2}x^T Ax + b^T x + c$ pre symetrickú maticu $A \in \mathbb{S}^n,\ b \in \mathbb{R}^n,\ c \in \mathbb{R}$. Ak je A kladná semidefinitná matica, $f(x)$ je konvexné (ak je A kladne definitná, $f(x)$ je rýdzo konvexná).
- **Normy**. $\|x\|_p = (\sum_{i=1}^n |x|^p)^{1/p}$ pre $p \geq 1$; $\|x\|_\infty = max_k|x_k|$.

1.3 Lagrangeova dualita a podmienky optimality

Uvažujme o všeobecnej úlohe matematického programovania(nemusí byť konvexná) [7]

$$\begin{aligned} \min \quad & f_0(x) \\ & f_i(x) \leq 0, \qquad i = 1, \cdots, m \\ & h_i(x) = 0, \qquad i = 1, \cdots, p \end{aligned} \tag{3}$$

kde $x \in \mathbb{R}$ je premenná, úloha má neprázdny definičný obor D a optimálne riešenie p^*. Následne definujeme Lagrangeovu funkciu $L : \mathbb{R}^n \times \mathbb{R}^m_+ \times \mathbb{R}^p \longrightarrow \mathbb{R}$

$$L(x, \lambda, \nu) = f_0(x) + \sum_{i=1}^{m} \lambda_i f_i(x) + \sum_{i=1}^{p} \nu_i h_i(x)$$

kde λ_i je Lagrangeov multiplikátor pridružený ku $f_i(x) \leq 0$ a ν_i je Lagrangeov multiplikátor pridružený ku $h_i(x) = 0$. Vektory λ a ν sa nazývajú **duálne premenné problému**. [7]

Lagrangeovu duálnu funkciu $g : \mathbb{R}^m \times \mathbb{R}^p \longrightarrow \mathbb{R}$ definujeme ako najmenšiu hodnotu Lagrangeovej funkcie podľa x pre $\lambda \in \mathbb{R}^m, \nu \in \mathbb{R}^p$

$$g(\lambda, \nu) = \inf_{x \in D} L(x, \lambda, \nu) = \inf_{x \in D} \left(f_0(x) + \sum_{i=1}^{m} \lambda_i f_i(x) + \sum_{i=1}^{p} \nu_i h_i(x) \right)$$

Ak by teda bola Lagrangeova funkcia L zdola neohraničená v premennej x, duálna funkcia g by nadobudla hodnotu $-\infty$. Duálna funkcia g je konkávna, aj keď pôvodná úloha nie je konvexná.

Lagrangeova duálna úloha má formu

$$\begin{aligned} \max \quad & g(\lambda, \nu) \\ & \lambda \succeq 0 \end{aligned}$$

Kuhn-Tuckerove podmienky optimality (autori Harold W. Kuhn a Albert W. Tucker):

- **optimality(stacionarity)**

  $$\nabla f(x^*) + \sum_{i=1}^{m} \lambda_i \nabla f_i(x^*) + \sum_{i=1}^{p} \nu_i \nabla h_i(x^*) = 0$$

- **prípustnosti**

  $$f_i(x^*) \leq 0, \qquad i = 1, \cdots, m$$
  $$h_i(x^*) = 0, \qquad i = 1, \cdots, p$$

- **duality**

  $$\lambda_i \geq 0, \qquad i = 1, \cdots, m$$

- **komplementarity**

  $$\lambda_i f_i(x^*) = 0 \qquad i = 1, \cdots, m$$

Za predpokladu Slaterovej regularity sú Kuhn-Tuckerove(KT) podmienky nutnými podmienkami optimality pre všeobecnú úlohu (3).

Keď je primárna úloha konvexná (f_i sú konvexné a h_i sú afínne), sú KT podmienky zároveň aj postačujúcimi. Ak x^*, λ^*, ν^* spĺňajú KT podmienky, x^* a (λ^*, ν^*) sú primárne a duálne optimálne riešenia úlohy. [7]

2 Začíname

Nadstavba Matlabu cvx je modelovací systém pre problémy konvexného programovania. Jeho autorom je Michael C. Grant, Ph.D. Cvx vie riešiť problémy lineárneho programovania, kvadratického programovania, programovania nad kuželmi druhého rádu, či úlohy semidefinitného programovania [3].

Inštaláciou cvx, premeníte Matlab na silný modelovací jazyk optimalizácií. Môžete ho používať v skripte, funkcii alebo priamo v príkazovom riadku. Dôležité však je odlíšiť príkazy určené pre cvx od ostatného kódu. Tieto špecifické príkazy začneme písať až za cvx_begin, čo je znak ktorý dá Matlabu vedieť, že riešenie má prenechať systému cvx. Súbor príkazov zakončíme povelom cvx_end.

Kód pre cvx môže obsahovať okrem vlastných špeciálnych príkazov aj príkazy obyčajné, známe i Matlabu. V rámci riešenia sady určenej pre cvx, sa v premenných nenachádza číselná hodnota. Vyskytujú sa tu však ako špeciálne matlabovské objekty pomocou ktorých sa skladá štruktúra optimalizačnej úlohy. Pokiaľ sa počas procesu riešenia vyskytne chyba, systém nám ju zobrazí. Po dosiahnutí koncového príkazu cvx_end sa úloha vyrieši a do premenných sa uloží výsledok už v normálnej číselnej forme.

2.1 Metóda najmenších štvorcov

Ukážme si riešenie konvexnej optimalizačnej úlohy, ktorá je považovaná za najzákladnejšiu. Takou úlohou je metóda najmenších štvorcov v ktorej hľadáme $x \in \mathbb{R}^n$ ktorý minimalizuje $\|Ax - b\|_2$, kde $A \in \mathbb{R}^{m \times n}$ má plnú hodnosť a $m \geq n$, tak hodnosť matice $h(A) = n$.

Vytvorme si teda základné dáta m, n, A, b v Matlabe:

```
1 m = 16; n = 8;
2 A = randn(m,n);
3 b = randn(m,1);
```

Štandardne možno riešenie $x = (A^T A)^{-1} A^T b$ získať pomocou opačnej lomky:

```
4 xls = A\b;
```

použitím cvx úlohu vyriešime nasledovne:

```
5 cvx_begin
6     variable x(n);
7     minimize( norm(A*x-b) );
8 cvx_end
```

- riadok č. 5 oznamuje Matlabu začiatok sekcie s použitím príkazov špecifických pre systém cvx
- riadok č. 6 deklaruje x ako premennú s dimenziou n (takto je nutné zadeklarovať všetky premenné ktoré sa budú používať pri riešení úlohy)
- riadok č. 7 špecifikuje funkciu na minimalizovanie (prípadne maximalizovanie), tu konkrétne $\|Ax$ - $b\|_2$
- riadok č. 8 ukončuje súbor príkazov určených pre cvx a dáva signál na vyriešenie úlohy.

Ak by sme sa skúsili nazrieť do neznámeho vektora x po riadku č. 5 no pred riadkom č. 8, nezískali by sme žiadnu číselnú hodnotu, len údaj o špeciálnej premennej:

```
x = cvx affine expression (8x1 vector)
```

V podobnom tvare sa nachádzajú všetky premenné, či funkcie ktoré chceme vypočítať. Je to však vždy tvar nenumerického objektu v Matlabe. Pokúsme sa urobiť chybu v deklarovaní funkcie a zameňme riadok č.7 za

```
maximize( norm(A*x-b) );
```

a získame chybu, ktorá nás upozorní na chýbajúcu konkávnosť pri funkcii ktorú chceme zmaximalizovať.

Môže sa však stať, že matica $A \in \mathbb{R}^{m \times n}$ nebude mať plnú hodnosť, t.j. $h(A) = r$, kde $r < n$ a $r < m$. V takomto prípade je nutné rozdeliť maticu $A = U\Sigma V^T$. Potom sa dá táto úloha vyriešiť pomocou pseudo-inverznej matice $A^\dagger = V\Sigma^{-1}U^T \in \mathbb{R}^{n \times m}$.

Ak te teda hodnosť matice $h(A) = n$, tak $A^{\dagger} = (A^T A)^{-1} A^T$. Ak $h(A) = m$, $A^{\dagger} = A^T (A A^T)^{-1}$. Pri štvorcovej, nesingulárnej je to potom $A^{\dagger} = A^{-1}$.

Vytvorme si teda príklad:

```
A = magic(8); A = A(:,1:6);
b = 260*ones(8,1);
```

Vytvorili sme maticu $A_{8\times6}$ ktorá však má hodnosť $h(A) = 3$. Vo vektore b, 260 zodpovedá súčtu riadku(alebo stĺpca) 8×8 magickej matice. Ak by sme jej neodobrali dva stĺpce, riešením by bol jednotkový vektor. Úloha je ale stále riešiteľná, no má viac riešení. Ak teda chceme získať výsledok $x = A^{\dagger} * b$, treba použiť príkaz:

```
cvx_begin
    variable x(6);
    minimize( norm(A*x-b)+norm(x) );
cvx_end
```

čo nám ponúkne riešenie:

$$x = [1.15 \quad 1.46 \quad 1.38 \quad 1.38 \quad 1.46 \quad 1.15]'$$

Rovnaké riešenie sa dá získať aj Matlabovým príkazom

```
x=pinv(A)*b
```

Toto riešenie je však unikátne, pretože norma x ($\|x\|_2$) je menšia ako norma ktoréhokoľvek iného riešenia, preto bolo nutné túto podmienku pridať do (13).

Ak by sme použili len postup ako v (7), získali by sme výsledok len s 3-mi nenulovými členmi(pre toto riešenie cvx dosiahol najmenšiu možnú hodnotu):

$$x = [4 \quad 8 \quad -4 \quad 0 \quad 0 \quad 0]'$$

Aj tu by sa dal použiť príkaz s opačnou lomkou `xls = A\b;` čo nám pri varovnom vyhlásení, `Warning: Rank deficient, rank = 3`, ponúkne ďalšie možné riešenie

$$x = [4 \quad 5 \quad 0 \quad 0 \quad 0 \quad -1]'.$$

2.2 Ohraničenia

Predpokladajme, že do úlohy chceme pridať nejaké ohraničenia $l \preceq x \preceq u$, kde $l \preceq x$ znamená, že vektor l je po zložkách menší ako vektor x t.j. $l_i \leqq x_i$ pre všetky i. Takúto úlohu už však pomocou lomky riešiť nejde. Zaplňme si teda vektory l a u číselnými hodnotami:

```
ohr = randn(n, 2);
l = min( ohr, [], 2);
u = max( ohr, [], 2);
```

tu funkcia *min(ohr,* [] *,* 2 *)* vyberie najmenšie prvky z druhej dimenzie matice *ohr*. [] tu predstavuje žiadnu maticu.

S týmito údajmi, nám stačí riadky č.5 až 8 pozmeniť na:

```
18 cvx_begin
19     variable x(n);
20     minimize ( norm(A*x-b) );
21     subject to
22       l <= x <= u;
23 cvx_end
```

- riadok č. 21 nemá žiaden vplyv na chod ani výpočet minima. Vloženie takéhoto riadku do kódu je dobrovoľné a slúži čisto na lepšiu čitateľnosť.
- riadok č. 22 predstavuje naše ohraničenie $l \preceq x \preceq u$.

Ako ohraničenie môžeme chápať akúkoľvek podmienku. Napr. $Cx = d$, kde C je matica s rozmerom *(k,n)* a d je vektor s dĺžkou k. Dvojité rovná sa predstavuje priradenie podmienky a neudáva pravdivostnú hodnotu, ako by tomu bolo mimo cvx.

2.3 Výhody

Jedna z najväčších výhod používania cvx, je prehľadnosť. Nech máme akokoľvek jednoduchý, alebo zložitý optimalizačný problém, zápis pre cvx sa veľmi nemení. Tento spôsob je teda stále rovnako prehľadný

- Deklarované premenné sa vždy nachádzajú pri príkaze *variable* a môžu byť reálne alebo komplexné. Každej jednej sa musí priradiť ich veľkosť (či je to skalár, vektor, matica alebo rad). Premenným maticiam sa dá špecifikovať štruktúra, čo nám uľahčuje prácu a zvyšuje efektivitu. Príkazmi

  ```
  variable x(20) complex;
  variable Y(20,5);
  variable Z(20,20) symetric;
  ```

  vznikne komplexný 20 členný vektor x, reálna 20×5matica Y, symetrická 20×20 matica Z.

  Ďaľšie podporované špecifikácie sú [2]:

  - `banded(lb,ub)` hovorí o "pruhovanosti"matice. Ak je $lb = lu = 0$ vznikne diagonálna matica(dosiahnuteľé aj s `diagonal`), v prípade $lb = lu = 1$ je to trojdiagonálna matica (rovnako s `tridiagonal`).

    `banded(1,2)` vytvára maticu s jedným pruhom pod diagonálou a dvomi pruhmi nad

    $$\begin{bmatrix} a & b & c & 0 \\ d & e & f & g \\ 0 & h & i & j \\ 0 & 0 & k & l \end{bmatrix}$$

  - `toeplitz` deklaruje maticu ktorá má rovnaké hodnoty na diagonálach (medzi diagonálami sa môžu líšiť)
  - `hankel` je podobná ako `toeplitz`, no rovnaké hodnoty sa nachádzajú na antidiagonálach
  - `scaled_identity` deklaruje rovnakú maticu ako `diagonal` a `toeplitz` spolu.

– Ostatné možné príkazy:

`hermitian` `lower_bidiagonal` `lower_hessenberg`
`lower_triangular` `skew_symmetric` `upper_bidiagonal`
`upper_hankel` `upper_hessenberg` `upper_triangular`

- Ľahko viditeľná účelová funkcia a jej podmienky

Pri riešení akéhokoľvek problému, sa stačí sústrediť hlavne na samotné teoretické riešenie, namiesto tvorenia kódu na výpočet úlohy.

3 Príklady a riešenia

V tejto kapitole si ukážeme využitie cvx pri riešení praktických úloh.

3.1 Výmenná krivka(Trade-off curve)

Toto je veľmi častá úloha, ktorej princíp spočíva v hľadaní Paretovho optima. Toto optimum predstavuje napríklad stav ekonomiky, keď už ziaden obyvateľ nedokáže zvýšiť svoje bohatstvo, bez toho aby niekto iný nezchudobnel. Týchto optím je väčšinou viac ako len jedno.

Predstavme si príklad, kde bod x je Paretovo optimum pre ciele $F_1(x)$ a $F_2(x)$. Naša otázka teraz znie, o koľko by sme mali zvýšiť $F_2(y)$, aby sme získali prijateľný bod y s $F_1(y) \leq F_1(x)$ - a kde konštanta $a>0$. Pýtame sa teda koľko musíme zaplatiť v druhom cieli F_2 aby sme dosiahli vylepšenie o a v prvom cieli F_1. Ak musíme akceptovať veľký nárast v F_2 pre malý pokles v F_1, voláme to *silnou výmenou* medzi cieľmi pri Paretovom optime. Naopak, ak vieme získať veľký pokles v F_1 pri malom náraste F_2, voláme túto výmenu *slabou*

Vo všeobecnosti, môžme mať viac ako iba dva ciele ktoré chceme naplniť. Vtedy sa množina Paretových optím nazýva *optimálna výmenná plocha*. Pri dvoch cieľoch sa klasicky používa výraz *optimálna výmenná krivka*.

3.1.1 Príklad: Zmena rizika a výnosu v optimalizácii portfólia [6(13.2)]

Majme portfólio s dátami:

$$\bar{p} = \begin{bmatrix} 0.12 \\ 0.10 \\ 0.07 \\ 0.03 \end{bmatrix}, \qquad \Sigma = \begin{bmatrix} 0.0064 & 0.0008 & -0.0011 & 0 \\ 0.0008 & 0.0025 & 0 & 0 \\ -0.0011 & 0 & 0.0004 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix},$$

kde $\bar{p}$ je priemerný výnos aktíva a Σ je variančná matica aktív.

vyriešime úlohu:

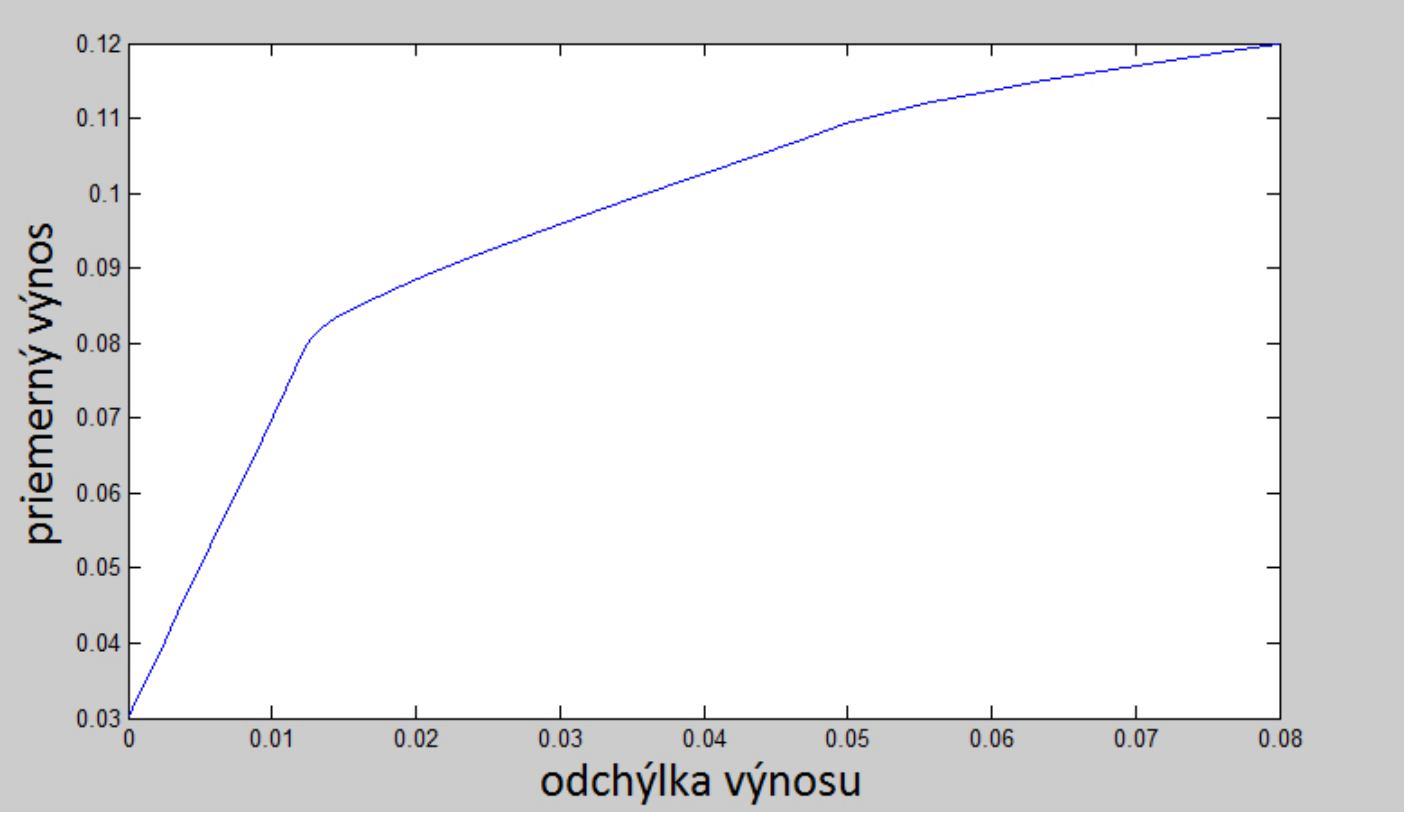
$$\begin{aligned} \min \qquad & -\bar{p}^T x + \mu x^T \Sigma x \\ & \mathbf{1}^T x = 1,\ x \preceq 0 \end{aligned}$$

pre veľké množstvo kladných čisel μ (napríklad 100 logaritmicky rozdelených hodnôt od 1 po 10^7). Potom nakreslíme optimálne hodnoty očakávaného výnosu $\bar{p}^T x$ od štandardnej odchýlky $(x^T \Sigma x)^{1/2}$, a optimálne hodnoty x od štandardnej odchýlky $(x^T \Sigma x)^{1/2}$.

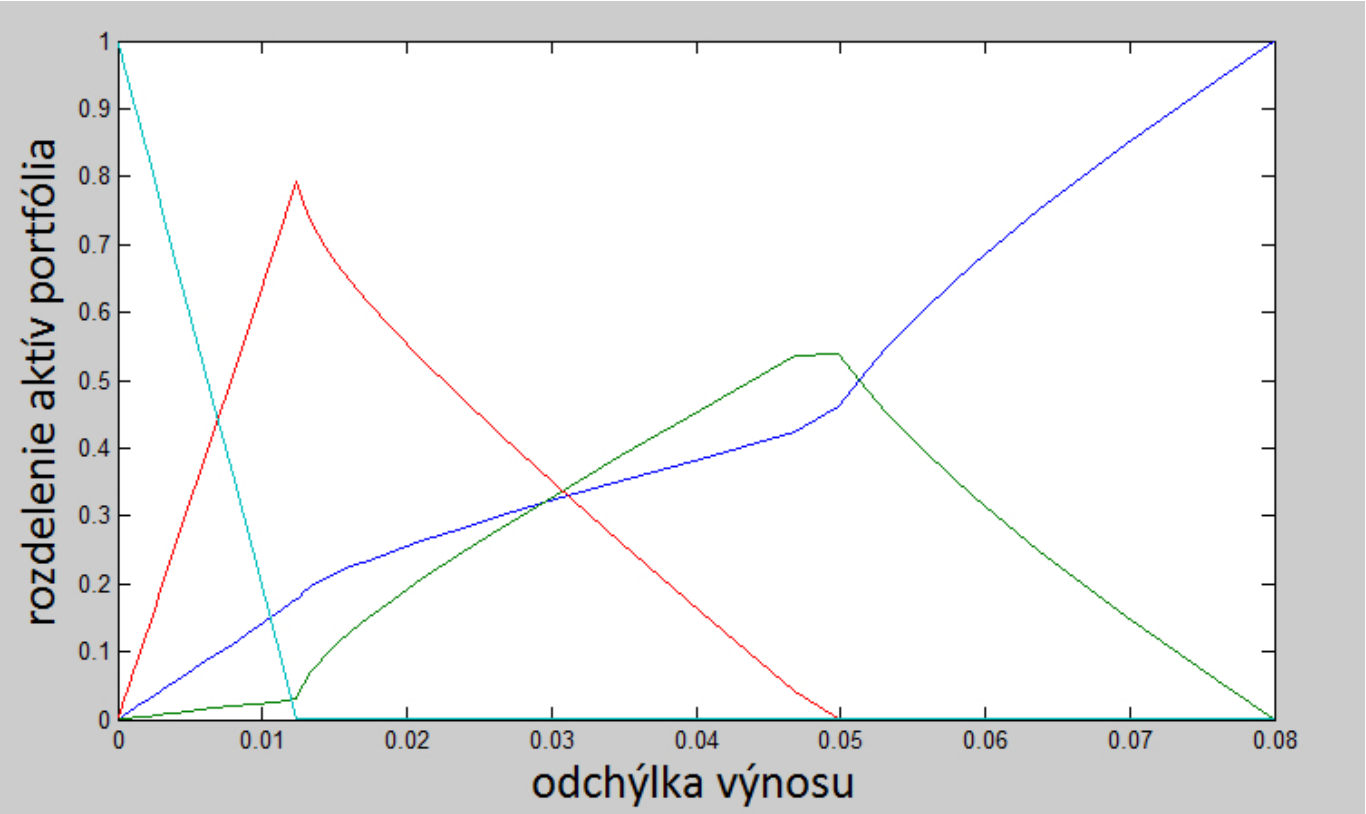
Treba si uvedomiť, že premennou μ určujeme svoj pomer preferencií ku priemernému alebo náhodnému výnosu, . Ak by sme teda zvolili $\mu \to 0$, portfólio koncentrujeme ku aktívu s najvyšším priemerným ziskom. (Zvačšovaním hodnoty μ, prenášame váhu na menej rizikové aktíva, čím zvyšujeme ich podiel v portfóliu, t.j. znižujem varianciu portfólia).

```
24 p=[0.12; 0.1; 0.07; 0.03];
25 SIG=[0.0064 0.0008 -0.0011 0; 0.0008 0.0025 0 0; -0.0011 0 0.0004 0; 0 0 0 0];
26 mik=logspace(0,7,100);
27 for i=1:length(mik)
28     cvx_begin quiet
29         variable x(4);
30         minimize(-p'*x+mik(i)*quad_form(x,SIG))
31         subject to
32           sum(x)==1;
33           0<=x;
34     cvx_end
35     Rpx(i)=p'*x;
36     Rx(:,i)=x;
37     RSIG(i)=sqrt(x'*SIG*x);
38 end
39 plot(RSIG, Rpx)
40 plot(RSIG, Rx(1,:), RSIG, Rx(2,:), RSIG, Rx(3,:), RSIG, Rx(4,:))
```

- funkcia $logspace(a, b, n)$ vytvorí logaritmicky rozložený vektor veľkosti n s hranicami $[10^a,\ 10^b]$ (26).
- pri niekoľkonásobnom spúšťaní cvx, by všetky výstupy procesov pôsobili rušivo. použitím *quiet* zrušíme toto vypisovanie (28).
- funkcia $quad\_form(x, SIG)$ nám slúži ako náhrada za $x'SIGx$. Tento príkaz umožňuje cvx-u priamo rozpoznať kvadratickú formu. (ak cvx nájde zápis $x'SIGx$, najprv zistí jeho kokávnosť či konvexnosť a následne ho preloží do ekvivalentného volania funkcie $quad\_form(x, SIG)$). (30).
- po každej optimalizácií si hodnotu priemerného výnosu uložíme do Rpx a rozloženie aktív v portfóliu do Rx. Odchýlky výnosu sa nachádzajú v $RSIG$.



Obr. 2: priemerný výnos od štandardnej odchýlky



Obr. 3: prerozdelenie zložiek portfólia podľa odchýlky

Ako môžeme vidieť z (Obr. 1) naša výmenná krivka sa začína na hodnote výnosu 3% s odchýlkou 0%. Rozdelenie aktív do tohoto bezrizikového portfólia získame pozretím sa na (Obr. 2). Keď zvyšujeme svoj záujem o dodatočný náhodný zisk, úloha bezrizikového aktíva klesá v portfóliu, až ho po určitom čase nevyužívame. Zamenili sme vyššie riziko, za možný vyšší zisk.

3.2 Aproximácia a interpolácia funkcí

Interpolácia je metóda pomocou ktorej sa vytvára nový súbor dát(väčšinou v podobe funkcie) na určitom intervale z už známych dátových bodov.

V inžinierstve a vo vede, sa väčšinou získa množstvo údajov pozorovaním či testovaním. Tieto údaje sú reprezentáciou hodnôt funkcie pre určitý počet nezávyslej premennej. Tu sa potom odhadujú (interpolujú) daľšie body z rovnakej, no neznámej funkcie.

Veľmi blízky problém k interpolácii, je aproximácia zložitých funkcií pomocou jednoduchších. Predpokladajme, že poznáme predpis funkcie, ktorá je však príliš zložitá na efektívne vyhodnotenie. Môžme si však vybrať niekoľko známych bodov z tejto funkcie a následne pomocou interpolácie vytvoriť jednoduchšiu funkciu. Z tejto vytvorenej funkcie pravdaže nezískame úplne rovnaké výsledky, no zameranie problému či typ použitej interpolačnej metódy môžu vykompenzovať vzniknutú chybu. Aproximáciou a interpoláciou funkcií sa zaoberá numerická analýza.

3.2.1 Príklad: Aproximácia(fitovanie) k polynómu [7(6.7)]

Majme dáta $x_1, ..., x_m \in \mathbf{R}$ a $y_1, ..., y_m \in \mathbf{R}$.
Tieto dáta chceme aproximovať do podoby

$$p(x) = a_1 + a_2 x + a_3 x^2 + ... + a_n x^{n-1}$$

Pre každé a si vytvoríme vektor chýb

$$e = (p(x_1) - y_1, ..., p(x_m) - y_m).$$

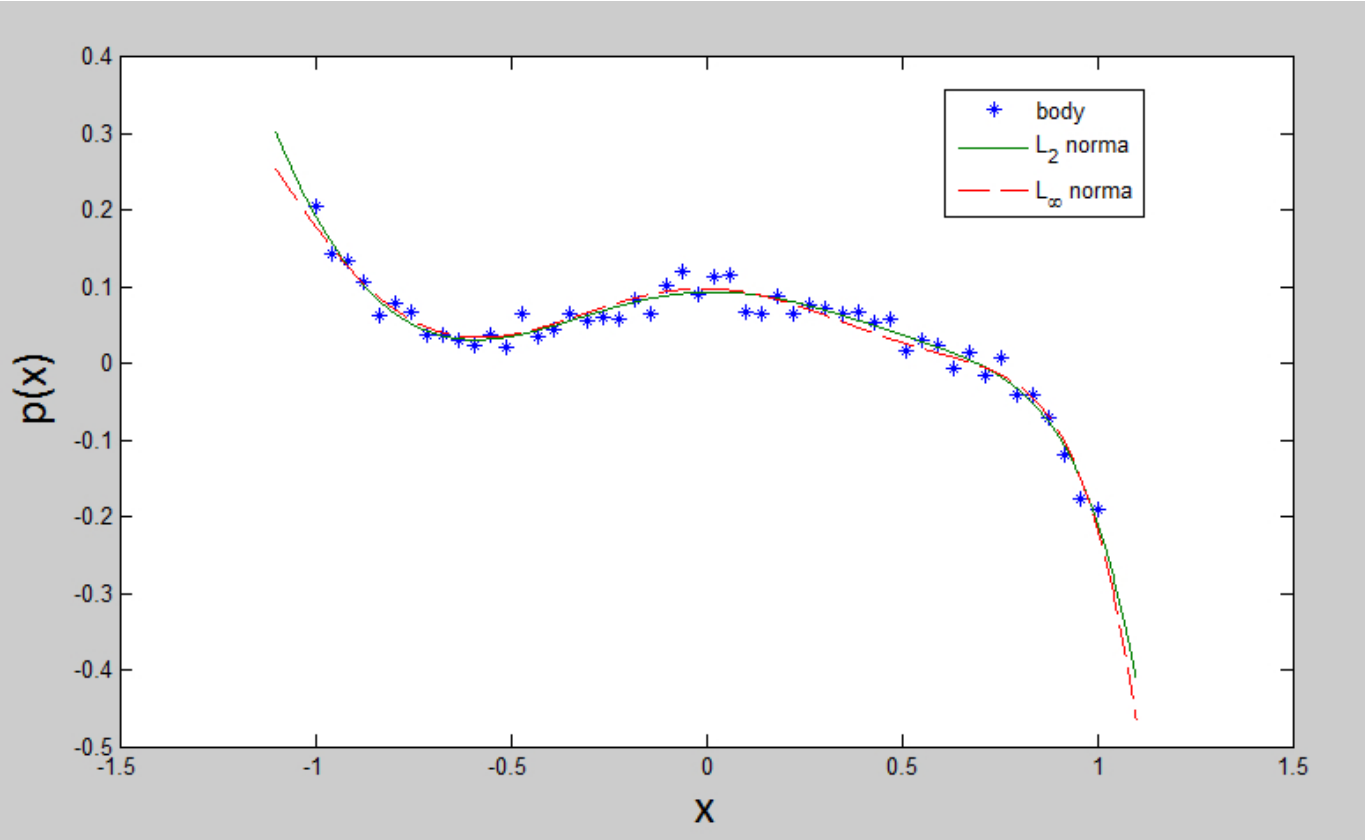
Pre nájdenie polynómu, čo minimalizuje normu chýb, nám stačí vyriešiť úlohu

$$\min \qquad \|e\| = \|Xa - y\|$$

s premennou $a \in \mathbf{R}^n$, kde $X_{ij} = x_i^{j-1};\ i = 1, ..., m;\ j = 1, ..., n.$

```
41 n=8; m=50; randn('state',0);
42 x = linspace(-1, 1, m);
43 y = 1./(10+80*x.^2) + 0.1*x.^3 - 0.3*x.^5 + 0.015*randn(1,m);
44 X = vander(x');
45 X=X(:,m-n+[1:n]);
46 cvx_begin quiet
47     variable a(n)
48     minimize (norm(X*a - y'))
49 cvx_end
```

```
cvx_begin quiet
    variable ai(n)
    minimize (norm(X*ai - y', inf))
cvx_end
x2 = linspace(-1.1, 1.1, 1000);
yp = a(1)*ones(1,1000);
ypi = ai(1)*ones(1,1000);
for i = 2:n
    yp = yp.*x2 + a(i);
    ypi = ypi.*x2 + ai(i);
end;
plot(x,y,'*' ,x2,yp,'-', x2, ypi,'--')
```



Obr. 4: vygenerované body $[x,\ y]$ a aproximované funkcie L_2 *a* L_∞ *normou*

- zadeklarujeme si stupeň polynómu $(n-1)$, počet interpolačných bodov a nastavíme si stav generátora na počiatok. Tak môžeme pri každom spustení pozorovať rovnaké vygenerované body. (41).

- funkcia $linspace(-1,\ 1,\ m)$ vytvorí lineárne rozložený vektor x veľkosti m s hranicami $[a,\ b]$ (42), z ktorého si následne vytvoríme body y podľa nejakej funkcie, aby sme mali čo aproximovať(funkcia je vymyslená tak aby medzi bodmi neboli veľké vzdialenosti)(43). Ku funkcii si je vhodné pridať náhodnú odchýlku, aby sme mohli pozorovať rozdiel medzi dvomi typmi fitovania.

- funkcia $X = vander(x)$ vytvorí Vandermondovu maticu, ktorej stĺpce sú mocniny vektora x ($X_{ij} = x_i^{m-j}$ kde m je dĺžka vektora x). (44)

- Vezmeme si len podsledných n potrebných stĺpcov. Riadky matice X, teda budú vyzerať takto: $[x_i^{n-1}, ...,\ x_i,\ 1]$, pričom vyriešené vektory a a ai budú mať usporiadanie $[a_n, ...,\ a_2,\ a_1]$ (45).

- Následne vykreslíme obe aproximované funkcie spolu s vygenerovanými bodmi. Na toto vykreslenie si vytvoríme nový set $x2$ 1000 bodov, ktorých funkčné yp a ypi vypočítame Hornerovou metódou.

3.2.2 Príklad: Prispôsobenie racionálnej funkcie k exponenciálnej funkcii [6(5.2)]

Uvažujme o probléme s dátami:

$$t_i = -3 + 6\frac{i-1}{k-1}, \quad y_i = ^{t_i}, \quad i = 1,\ 2, ...,\ k,$$

kde $k = 201$. Inak povedané, body nám uniformne opisujú exponenciálnu funkciu na intervale $[-3,\ 3]$.

Pokúsime sa nájsť funkciu:

$$f(t) = \frac{a_0 + a_1 t + a_2 t^2}{1 + b_1 t + b_2 t^2}$$

ktorá minimalizuje $max_{i=1,...,k}\ |f(t_i) - y_i|$. Požadujeme podmienku $(1 + b_1 t + b_2 t^2) > 0\ pre\ i = 1,\ 2, ...,\ k$. Dopočítame optimálne hodnoty $a_0,\ a_1,\ a_2,\ b_1,\ b_2$ s presnosťou 0,001.

Nakoniec vykreslíme opisujúce body a výslednú optimálnu funkciu a na ďaľší obrázok odchýlky $f(t_i) - y_i$.

Funkcia v zadanej úlohe nie je konvexná ale kvázikonvexná na zadanom definičnom obore $[-3,\ 3]$. Naša úloha $max_{i=1,...,k}\ |f(t_i) - y_i| \leq \alpha$ platí práve vtedy keď

$$\left|\frac{a_0 + a_1 t_i + a_2 t_i^2}{1 + b_1 t_i + b_2 t_i^2} - y_i\right| \leq \alpha, \qquad i = 1,\ 2, ...,\ k.$$

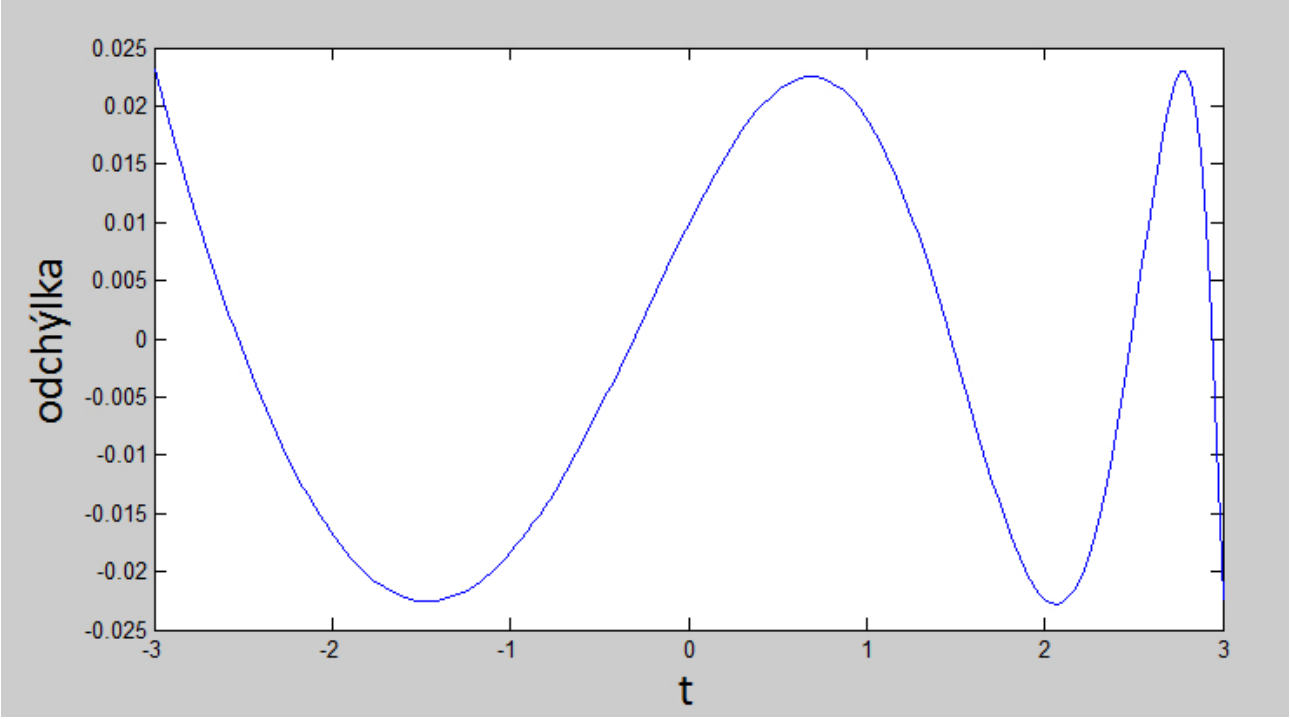
Vďaka kladnému menovateľovi možno úlohu upraviť na

$$\left|a_0 + a_1 t_i + a_2 t_i^2 - y_i(1 + b_1 t_i + b_2 t_i^2)\right| \leq \alpha(1 + b_1 t_i + b_2 t_i^2), \qquad i = 1, , ...,\ k,$$

čo je súbor $2k$ lineárnych nerovníc s premennými a a b pre jednu hodnotu α.
Na vyriešenie môžeme použiť metódu bisekcie, pokiaľ rozdiel hornej u a dolnej hranice l nebude menší ako α. Výsledok získame príslušným kódom

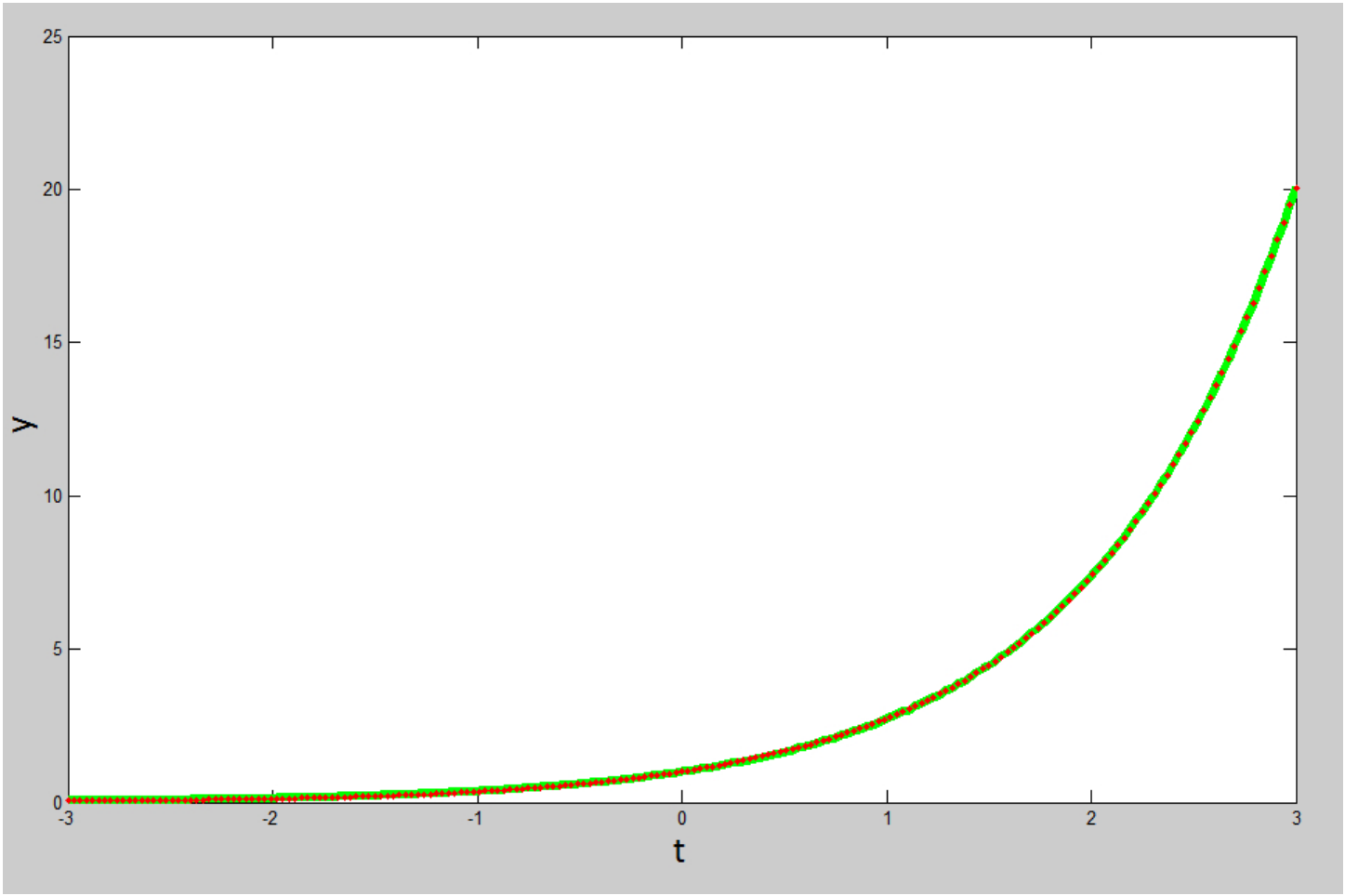
```
62 k=201;
63 t=(-3:6/(k-1):3)';
64 y=exp(t);
65 T=[ones(k,1) t t.^2];
66 u=exp(3); l=0;
67 alpha=0.001;
68 while u-l>= alpha
69     al2=(l+u)/2;
70     cvx_begin quiet
71         variables a(3) b(2);
72         subject~to
73           abs(T*a-y.*(T*[1;b])) <= al2*T*[1;b];
74     cvx_end
75     if strcmp(cvx_status,'Solved')
76         u=al2;
77         aopt=a; bopt=b;
```

```
78      else
79          l=al2;
80      end
81  end
82  yf=T*aopt./(T*[1;bopt]);
83  figure(1);
84      P=plot(t, y, 'g', t, yf, 'r.');
85      set(P,'LineWidth',5,'MarkerSize', 5);
86  xlabel('t'); ylabel('y');
87  figure(2);
88      plot(t, yf-y);
89      xlabel('t'); ylabel('odchylka');
```



Obr. 5: odchýlky funkčných hodnôt aproximovanej racionálnej od exponenciálnej funkcie

- Vytvoríme si požadované body $[t, y]$. (63).
- Pre zjednodušenie ďaľšieho zápisu, si vytvoríme maticu T do ktorej uložíme požadované mocniny vektora t. (65).



Obr. 6: exponenciálna funkcia(zelenou) a body z aproximovanej funkcie(červenou)

- Naprogramujeme cyklus, kde budeme metódou bisekcie postupne zmenšovať hornú u a dolnú l hranicu, až kým ich rozdiel nebude menší ako požadovaná chyba α
- Treba však vždy sledovať, či cvx dokázal nájsť prípustné riešenie, a keď áno, uložíme si ho vedľa. (75).
- Optimálne hodnoty:

$$a_1 = 1.0098,\ a_2 = 0.6119,\ a_3 = 0.1135,\ b_1 = -0.4146,\ b_2 = 0.0485.$$

3.3 Dualita

Väčšinou optimálna hodnota duálnej funkcie d^* má nižšiu hodnotu ako optimálna hodnota p^* primárnej funkcie, t.j. $d^* \leq p^*$(Slabá veta o dualite, ktorá platí vždy). Ak je primárna úloha konvexná (vo všeobecnosti stačí platnosť Slaterovej regularity), optimálne hodnoty primárnej a duálnej úlohy sa rovnajú $d^* = p^*$(Silná veta o dualite).

3.3.1 Príklad: Numerická analýza ruchu [6(4.1)]

Máme kvadratický problém

$$\begin{aligned} \min \quad & x_1^2 + 2x_2^2 - x_1x_2 - x_1 \\ & x_1 + x_2 \leq u_1 \\ & x_1 - 4x_2 \leq u_2 \\ & 5x_1 + 76x_2 \leq 1 \end{aligned}$$

s premennými $x_1,\ x_2$ a parametrami $u_1,\ u_2$.

(a) Vyriešime tento problém s hodnotou parametrov $u_1 = -2,\ u_2 = -3$ a nájdeme optimálne hodnoty primárnych premenných x_1^* a x_2^* a duálnych premenných $\lambda_1^*,\ \lambda_2^*$ a λ_3^*. Nech p^* predstavuje optimálnu hodnotu účelovej funkcie. Overíme platnosť Kuhn-Tuckerových podmienok pre nájdené optimálne hodnoty primárnych a duálnych premenných (pri rozumnej presnosti).

(b) Následne vyriešime rozrušenú verziu tohoto problému:

$$u_1 = -2 + \delta_1 \qquad\qquad u_2 = -3 + \delta_2$$

kde δ_1 a δ_2 nadobúdajú hodnoty z $\{-0.1,\ 0,\ 0.1\}$. (Spolu je tu teda 9 kombinácií, vrátane pôvodného problému pri $\delta_1 = \delta_2 = 0$). Pre každú kombináciu δ_1 a δ_2 vytvoríme predpoveď p^*_{pred} optimálnej hodnoty rozrušeného kvadratického problému. Túto hodnotu porovnáme s p^*_{exact}, presnou optimálnou hodnotou rozrušeného KP a Vytvoríme tabuľku s výsledkami. Ukážeme, že nerovnosť $p^*_{pred} \leq p^*_{exact}$ platí.

(a)

```
90  Q = [1 -1/2; -1/2 2];
91  h = [-1; 0];
92  A = [1 2; 1 -4; 5 76];
93  b = [-2; -3; 1];
94  cvx_begin quiet
95      variable x(2)
96      dual variable lambda
97      minimize(quad_form(x,Q)+h'*x)
98      subject to
99        lambda: A*x <= b
100 cvx_end
101 phv = cvx_optval;
```

- Deklarácia v (96) hovorí cvx, že *lambda* bude reprezentovať duálnu premennú.
- Príkaz v (99) spája túto premennú s ohraničeniami. Rozmer premennej *lambda* nedeklarujeme, je jej totiž automaticky priradený v tomto príkaze.
- Pri dosiahnutí cvx_end sa dopočítajú optimálne hodnoty primárnych aj duálnych premenných $x,\ \lambda$.
- Po spustení, zistíme optimálne hodnoty, pričom optimálne hodnoty x_1^* a x_2^* sú vďaka striktne konvexnej účelovej funkcií unikátne.

$$p^* = 8.22, \qquad x_1^* = -2.33,\ x_2^* = 0.17$$
$$\lambda_1^* = 1.61,\ \lambda_2^* = 3.67,\ \lambda_3^* = 0.11$$

Kuhn-Tuckerove podmienky:

$$x_1^* + 2x_2^* - u_1 \leq 0, \qquad x_1^* - 4x_2^* - u_2 \leq 0, \qquad 5x_1^* + 76x_2^* - 1 \leq 0 \tag{4}$$

$$\lambda_1^* \geq 0, \qquad \lambda_2^* \geq 0, \qquad \lambda_3^* \geq 0 \tag{5}$$

$$\lambda_1^*(x_1^* + 2x_2^* - u_1) = 0, \qquad \lambda_2^*(x_1^* - 4x_2^* - u_2) = 0, \qquad \lambda_3^*(5x_1^* + 76x_2^* - 1) = 0 \tag{6}$$

$$2x_1^* - x_2^* - 1 + \lambda_1^* + \lambda_2^* + 5\lambda_3^* = 0, \tag{7}$$

$$4x_2^* - x_1^* + 2\lambda_1^* - 4\lambda_2^* + 76\lambda_3^* = 0 \tag{8}$$

- (4) sú podmienky prípustnosti. Overíme ich príkazom

  ```
  A*x-b
  ```

- (5) sú podmienky duality. Ich platnosť je vidno z výsledku.

- (6) sú podmienky komplementarity. Overíme ich príkazom

  ```
  lambda.*(A*x-b)
  ```

- (7) a (8) sú podmienky optimality(stacionarity).

  Overíme ich príkazom

  ```
  2*Q*x+f+A'*lambda
  ```

Všetky podmienky môžeme vyhlásiť za splnené, keďže overované hodnoty sú veľmi malé.

(b) Predpokladáme platnosť vety o silnej dualite a $\lambda_1^*,\ \lambda_2^*$ sú optimálne duálne premenné pre nerozrušený problém. Použitím vety o slabej dualite, vieme optimálnu hodnotu rozrušeného problému predpovedať [8]:

$$p_{pred}^* \geq g(\lambda_1^*,\ \lambda_2^*) - \lambda_1^*\delta_1 - \lambda_2^*\delta_2$$
$$p_{pred}^* = p^* - \lambda_1^*\delta_1 - \lambda_2^*\delta_2$$

a všetky hodnoty získame pomocou kódu:

```
102 delta = [0 -0.1 0.1];
103 tab = [];
104 for i = delta
105     for j = delta
106         pp = phv - [lambda(1) lambda(2)]*[i; j];
107         cvx_begin quiet
108             variable x(2)
109             minimize(quad_form(x,Q)+h'*x)
110             subject to
111               A*x <= b+[i;j;0]
```

```
112         cvx_end
113         tab = [tab; i j pp cvx_optval];
114     end
115 end
116 tab
```

δ_1	δ_2	p^*_{pred}	p^*_{exact}
0	0	8.22	8.22
0	-0.1	8.58	8.71
0	0.1	7.86	7.98
-0.1	0	8.38	8.57
-0.1	-0.1	8.75	8.82
-0.1	0.1	7.02	8.32
0.1	0	8.06	8.22
0.1	-0.1	8.43	8.71
0.1	0.1	7.70	7.75

Tabuľka 1: Predpokladané a dopočítané optimálne hodnoty rozrušeného KP

- Vytvoríme si vektor požadovaných zmien (102) a prázdnu maticu pre uloženie výsledkov (103).
- (113) Pre každú hodnotu *delta* dopočítame predpokladanú optimálnu hodnotu p^*_{pred} a presnú optimálnu hodnotu p^*_{exact}.
- V každom jednom prípade platí nerovnosť $p^*_{pred} \leq p^*_{exact}$.

3.4 Rôzne aplikácie

3.4.1 Príklad: Plánovanie rádioterapie (liečby ožarovaním) [6(17.2)]

Pri liečbe ožarovaním, je pacient ožarovaný za cieľom zničiť bunky tumoru, avšak pri snahe čo najmenej poškodiť okolité tkanivo. Žiarenie sa tkanivu dodáva v lúčoch

podľa známeho vzoru, pričom dávka žiarenia v lúči sa dá meniť. (Väčšinou sa ožaruje viacerými lúčmi naraz. Celá liečba potom pozostáva z niekoľých takýchto sérií).

Nech teda b_j predstavuje dávku žiarenia v lúči j, kde $j = 1, ..., n$. Toto musí spĺňať $0 \leq b_j \leq B^{max}$, kde B^{max} je maximálna možná úroveň nastavenia jedného lúča. Ožiarená oblasť je rozdelená na m voxelov [1], s označením $i = 1, ..., m$. Dávka d_i ktorú dostane voxel i je lineárne závyslá od úrovne lúča, t.j. $d_i = \sum_{j=1}^{n} A_{ij} b_j,\ A \in \mathbb{R}_+^{m \times n}$. Matica A je známa a charakterizuje vzor lúčov.

Známa podmožina voxelov, $\tau \subset \{1, ..., m\}$, predstavuje tumor alebo cieľovú oblasť. Každý nádorový voxel chceme ožiariť minimálnou dávkou D^{target}, t.j. $d_i \geq D^{target}$ pre $i \in \tau$. Všetky ostatné voxely chceme ožiariť čo najmenej, teda $d_i \leq D^{other}$, kde D^{other} je maximálna chcená dávka pre necielené voxely. Toto je však vo všeobecnosti neuskutočniteľné, a preto budeme riešiť problém minimalizovaním nadbytočného ožiarenia

$$E = \sum_{i \notin \tau} ((d_i - D^{other})_+)^2,$$

kde $(.)_+$ predstavuje nezápornú časť. E sa teda dá vyložiť ako súčeť štvorcov nadbytočného ožiarenia aplikované na necielené tkanivo.

Vyriešime problém pre dáta [A1]:

```
117 rand('state', 0);
118 n = 200;                        %počet lúčov
119 mtarget = 100;                  %počet cieľených voxelov
120 mother = 400;                   %počet ostatných voxelov
121 Atarget = sprand(mtarget, n, 0.2);
122 Atarget = Atarget + [5*sprand(mtarget,mtarget,0.2) zeros(mtarget,n-mtarget)];
123 Aother= sprand(mother, n, 0.2);
124 Bmax = 10;
125 Dtarget = 1;
126 Dother = 0.2;
```

[1] Zloženie anglických slov *volumetric* (objemový) a *pixel*. Voxel je teda častica objemu, predstavujúca hodnotu v pravidelnej mriežke trojdimenzionálneho priestoru počítačovej grafiky. Voxely sa používajú najčastejšie pri vizualizácií a analýze lekárskych a vedeckých dát.

Tu máme maticu A rozdelenú na $Atarget$, ktorá obsahuje riadky zodpovedajúce voxelom s nechceným tkanivom, a $Aother$, obsahujúca riadky predstavujúcim ostatným voxelom. [2]

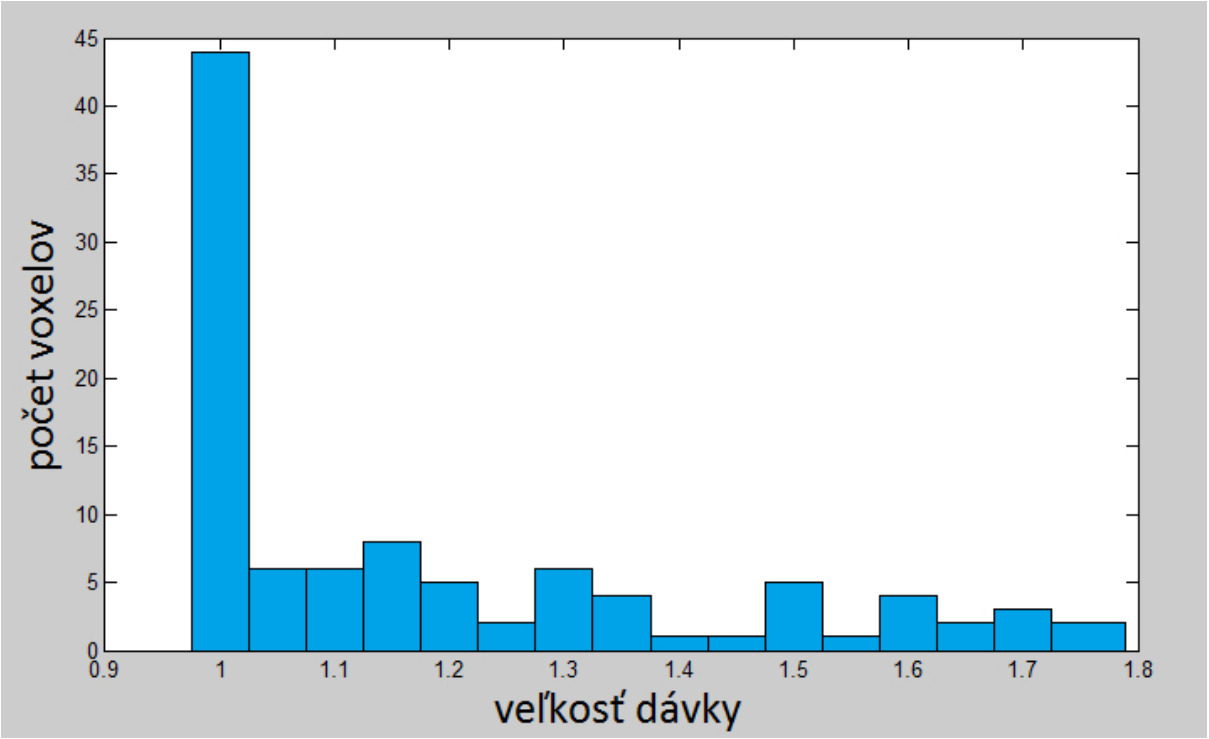
Získame optimálne riešenie, vykresl9me dávky histogramom pre mierené i pre ostatné voxely.

```
127 cvx_begin quiet
128     variable b(n)
129     minimize(sum_square_pos(Aother*b-Dother))
130     subject to
131       Atarget*b>=Dtarget
132       0<=b<=Bmax
133 cvx_end
134 cvx_optval
135 figure(1)
136     hist(Atarget*b,[min(Atarget*b):0.05:max(Atarget*b)])
137     xlabel('veľkost davky');
138     ylabel('počet voxelov');
139 figure(2)
140     hist(Aother*b,[min(Aother*b):0.01:max(Aother*b)])
141     xlabel('veľkost dávky');
142     ylabel('počet voxelov');
```

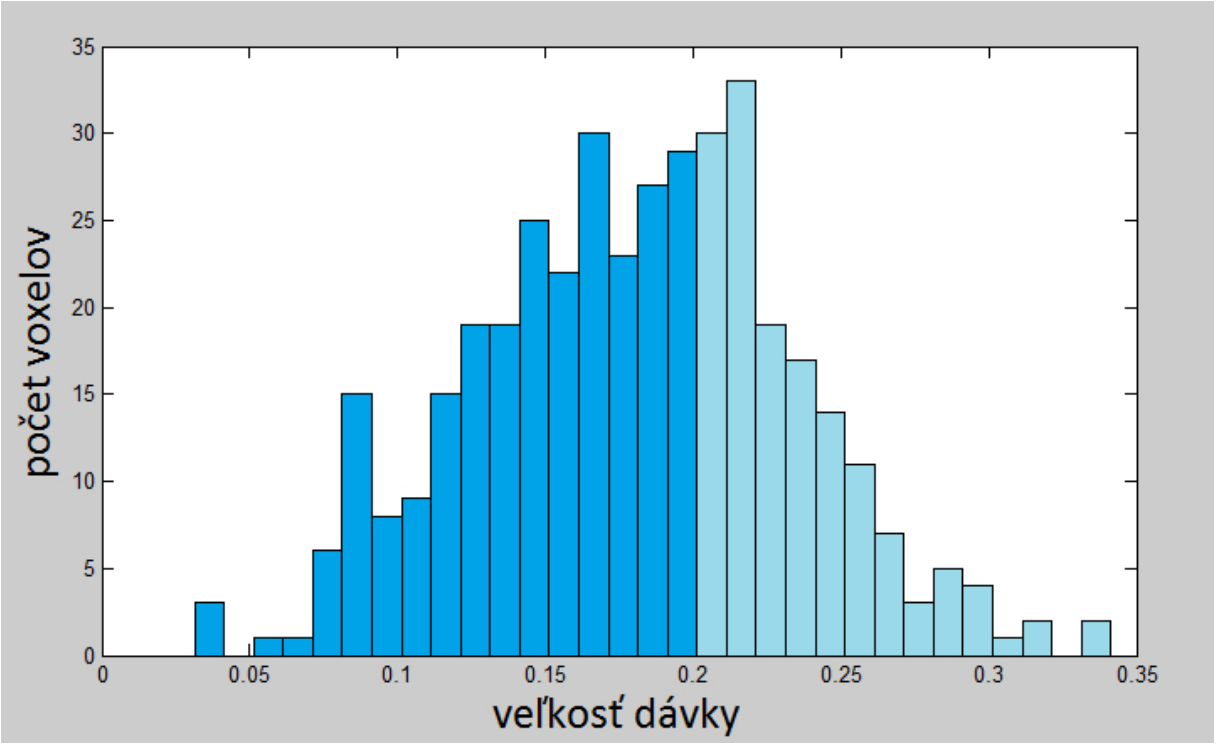
- Príkaz $sum\_square\_pos(x)$ (129) sa dá zameniť za funkciu $sum(square\_pos(x))$. Kde $square\_pos(x) = max\{x, 0\}^2, x \in \mathbb{R}$. Znamená to, že automaticky získame súčet štvorcov iba kladných členov vektora x. (v našom prípade $Aother * b - Dother$).

- Optimálna hodnota dosahuje 0.31, čo ukazuje, že okolité tkanivo je ožiarené viac ako by sme chceli.

[2] Obe matice sa generujú príkazom $sprand(m, n, hustota)$ ktorý vytvorí náhodnú $m \times n$ riedku maticu s počtom nenulových prvkov približne $hustota * m * n$, kde $0 \leq hustota \leq 1$. Vzorová matica pre lúče je tu síce náhodne vygenerovaná, no podobné výsledky by sa dali získať reálnymi dátami.



Obr. 7: Histogram veľkosti dávok dodané chorým voxelom



Obr. 8: Histogram veľkosti dávok dodané ostatným voxelom

(svetlejšie stĺpce predstavujú hodnotu $Aother * b \geq Dother$)

- Z (Obr. 7) je vidno, že takmer polovica voxelov s tumorom dostane minimálne chcené žiarenie $D^{target} = 1$. Zbytok je potom priblizne rovnomerne rozdelený v hraniciach $\langle 1.1,\ 1.8\rangle$

- Na (Obr. 8) môžme pozorovať nadbytočné ožarovanie okolitého tkaniva, kde znova niečo okolo polovice voxelov dostane prijateľné žiarenie. Tesne nad limit je ich počet ešte dosť vysoký no následne rapídne klesá. Niekoľko voxelov je však ožiarených takmer dvojnásobnou dávkou.

- Predpokladám, že tumor by bol zlikvidovaný, no za obeť by padlo aj nejaké tkanivo naviac.

3.4.2 Príklad: Maximalizácia zisku v hazarde a hypotetická pravdepodobnosť [6(13.9)]

Máme n účastníkov, ktorí vsádzajú na m možných výsledkov. Účastník i vytvorí ponuku na kúpu maximálne $q_i > 0$ hazardných lístkov za cenu $p_i > 0$ s určitým súborom možných výsledkov $S_i \subset \{1, ..., m\}$. Kancelária potom účastníkovi predá x_i $(0 \leq x_i \leq q_i)$ lístkov. Ak potom nastane udalosť j a pritom sa táto možnosť nachádza v S_i, potom účastník i dostane vyplatené 1€ za každý lístok. V opačnom prípade nedostane nič. Kancelária pozbiera dokopy $x_1p_1 + ... + x_np_n$ a vyplatí podľa udalosti j:

$$\sum_{i:\ j \in S_i} x_i.$$

Rozdiel medi týmito hodnotami, je zisk kancelárie.

(a) *Optimálna stratégia.* Ako by mala kancelária určiť x tak, aby maximalizovala svoj zisk pri najhoršom možnom scenári? (Kancelária určuje x až po prezkúmaní všetkých ponúk.)

(b) *Hypotetická pravdepodobnosť.* Predpokladajme, že x^* maximalizuje zisk kancelárie pri najhoršom možnom výsledku. Ukážeme, že existuje vektor pravdepodobností π $(\pi \in \mathbb{R}^m_+, \mathbf{1}^T\pi = 1)$, pre ktoré x^* maximalizuje aj predpokladaný zisk kancelárie.

Poznámka. Naformulujeme úlohu ako problém lineárneho programovania, potom sa π dá zostaviť z optimálnych duálnych premenných.

Poznámka. Pri známom π, vieme určiť "férovú"cenu ponuky i ako $p_i^{fer} = \sum_{j \in S_i} \pi_j$. Všetky ponuky s $p_i > p_i^{fer}$ budú kompletne prijaté ($x_i = q_i$). všetky ponuky s $p_i < p_i^{fer}$ budú odmietnuté ($x_i = 0$).

Poznámka. Táto úloha ukazuje ako možno odhadovať výsledky (napr. volieb) z ponúk hráčov.

(c) *Číselný príklad.* Vytvorenú metódu použijeme na jednoduchom príklade s $n = 5$ účastníkmi, $m = 5$ možných výsledkov a ponukami účastníkov

účastník i	p_i	q_i	S_i
1	0.5	10	$\{1,\ 2\}$
2	0.6	5	$\{4\}$
3	0.6	5	$\{1,\ 4,\ 5\}$
4	0.6	20	$\{2,\ 5\}$
5	0.2	10	$\{3\}$

Porovnáme aj optimálny najhorší scenár, s najhorším scenárom pri prijatí všetkých ponúk ($x_i = q_i$)a nájdeme pravdepodobnosti.

(a) Zisk kancelárie je

$$p^T x - \sum_{i:\ j \in S_i} x_i.$$

V najhoršom možnom prípade má kancelária zisk

$$p^T x - \max_{j=1,\dots,m} \sum_{i: j \in S_i} x_i.$$

Ak teda chceme maximalizovať zisk v takejto situácií, musíme riešiť problém:

$$\begin{aligned} \max \quad & p^T x - max(Ax) \\ & 0 \preceq x \preceq q \end{aligned}$$

kde x je premenná a matica A je definovaná takto:

$$A_{ji} = \begin{cases} 1 & j \in S_i, \\ 0 & \text{inak.} \end{cases}$$

(b) Tento problém môžeme prepísať do tvaru

$$\begin{aligned} \max \quad & p^T x - b \\ & b\mathbf{1} \preceq Ax \qquad\qquad (P1) \\ & 0 \preceq x \preceq q, \end{aligned}$$

kde b je nová skalárna veličina, ktorá pri riešení nadobudne hodnotu blízku hodnote maximálneho prvku vektoru Ax. Z úlohy vieme vytvoriť Lagrangeovu funkciu:

$$L(x,\ b,\ \lambda_1,\ \lambda_2,\ \lambda_3) = b - p^T x + \lambda_1^T(Ax - b\mathbf{1}) - \lambda_2^T x + \lambda_3^T(x - q),$$

z ktorého získavame podmienky ku duálnemu problému

$$\begin{aligned} \max \quad & -q^T\lambda_3 \\ & \mathbf{1}^T\lambda_1 = 1 \\ & A^T\lambda_1 - \lambda_2 + \lambda_3 = p \\ & \lambda_1 \succeq 0, \quad \lambda_2 \succeq 0, \quad \lambda_3 \succeq 0, \end{aligned}$$

kde $\lambda_1,\ \lambda_2,\ \lambda_3$ sú premenné. Ak sa pozrieme na podmienky ktoré musí spĺňať λ_1, t.j. $\mathbf{1}^T\lambda_1 = 1$ a $\lambda_1 \succeq 0$, zistíme, že λ_1 predstavuje pravdepodobnostné rozdelenie. Riešením prbolému získame optimálne primálne a duálne hodnoty $x^*,\ b^*,\ \lambda_1^*,\ \lambda_2^*$ a λ_3^*, pričom si môžeme povedať $\pi = \lambda_1^*$.

Ak teraz chceme maximalizovať predpokladaný zisk, musíme vyriešiť problém

$$\begin{aligned} \max \quad & p^T x - \pi^T Ax \qquad\qquad (P2) \\ & 0 \preceq x \preceq q, \end{aligned}$$

z ktorého získame optimálnu hodnotu x_2^*. Označme si riadky matice A ako $a_1^T, ..., a_n^T$, potom vieme jednoducho zapísať férovú cenu pre každého hráča ako $a_i^T\pi$.

- ak $p_i - a_i^T \pi > 0$, potom $x_{2i}^* = q_i$. (kancelária predá všetky možné lístky)
- ak $p_i - a_i^T \pi = 0$, potom $0 \preceq x_{2i}^* \preceq q_i$. (kancelária predá nejaké lístky)
- ak $p_i - a_i^T \pi < 0$, potom $x_{2i}^* = 0$. (kancelária nepredá žiadne lístky)

Na ukázanie rovnosti $x^* = x_2^*$ sa pozrime na Kuhn-Tuckerovu podmienku z problému (P1)

$$A^T \lambda_1 - \lambda_2 + \lambda_3 = p,$$

ktorú vieme upraviť na

$$p - A^T \pi = \lambda_3^* - \lambda_2^*$$

- ak $p_i - a_i^T \pi > 0$, potom z podmienky komplementarity $\lambda_{3i}^*(x_i^* - q_i)$, získavame $x_i^* = q_i$, keďže $\lambda_{3i}^* > 0$. Okrem toho z $-\lambda_{2i}^* x_i^* = 0$ získame $\lambda_{2i}^* = 0$.
- ak $p_i - a_i^T \pi = 0$, potom $\lambda_{2i}^* = 0,\ \lambda_{3i}^* = 0$ a $0 \preceq x_i^* \preceq q_i$.
- ak $p_i - a_i^T \pi < 0$, potom $\lambda_{2i}^* > 0,\ \lambda_{3i}^* = 0$ a $x_i^* = 0$.

(c) Nasledujúcim kódom vyriešime úlohu

```
143 A = [1 0 1 0 0; 1 0 0 1 0; 0 0 0 0 1; 0 1 1 0 0; 0 0 1 1 0];
144 p = [0.5; 0.6; 0.6; 0.6; 0.2];
145 q = [10; 5; 5; 20; 10];
146 n = 5; m = 5;
147 cvx_begin quiet
148     variables x(n) t
149     dual variable pie
150     maximize (p'*x-t)
151     subject to
152       pie: A*x <= t
153       0 <= x <= q
154 cvx_end
155 najhorz = cvx_optval
156 najhorz2 = p'*q-max(A*q)
```

```
157 pie
158 pfer = A'*pie
159 xopt = x
```

- $najhorz = 3.5$ predstavuje optimálny zisk pri najhoršom možnom výsledku.
- $najhorz2 = -5$ predstavuje zisk ak by kancelária prijala všetky ponuky(predala by všetky lístky a skončila by v strate).
- pie sú pravdepodobnosti nastania udalosti j, podľa účastníkov.

$$\pi = (0.1124, 0.3876, 0.1202, 0.1674, 0.2124)$$

- $pfer$ sú férové ceny ponúk.
- $xopt$ sú optimálne množstvá lístkov.

účastník i	p_i	p_i^{fer}	q_i	x_i	S_i
1	0.5	0.5	10	5	$\{1,\ 2\}$
2	0.6	0.1674	5	5	$\{4\}$
3	0.6	0.4923	5	5	$\{1,\ 4,\ 5\}$
4	0.6	0.6	20	5	$\{2,\ 5\}$
5	0.2	0.1202	10	10	$\{3\}$

3.4.3 Príklad: Minimalizácia spotreby paliva [6(3.17)]

Majme lineárny dynamický systém so stavom $x(t) \in \mathbb{R}^n, t = 0,,,,,N$ a pohon alebo vstupný signál $u(t) \in \mathbb{R}, t = 0,,,,,N-1$. Dynamika systému je daná lineárnou rekurziou

$$x(t+1) = Ax(t) + bu(t), \qquad t = 0, ..., N-1,$$

kde $A \in \mathbb{R}^{n\times n}$ a $b \in \mathbb{R}^n$ sú známe, pričom $x(0) = 0$.

Našou úlohou je vybrať vhodné vstupy $u(0), ..., u(N-1)$ tak aby sme minimalizovali spotrebu paliva danú ako

$$F = \sum_{t=0}^{N-1} f(u(t)).$$

Ohraničením je nám zadaný koncový stav $x(N) = x_{des}$, kde $x_{des} \in \mathbb{R}^n$ a N je daný časový horizont. Funkcia $f : \mathbb{R} \longrightarrow \mathbb{R}$ predstavuje spotrebu paliva pohonom ako funkciu amplitúdy signálu pohonu. Tu použijeme

$$f(\alpha) = \begin{cases} |\alpha| & |\alpha| \leq 1, \\ 2\,|\alpha| - 1 & |\alpha| > 1. \end{cases}$$

Toto znamená že spotreba paliva je úmerná absolútnej hodnote signálu pohonu, pre signál v rozmedzí $< -1,\ 1 >$. Pri vyššej amplitúde signálu je účinnosť pohonu približne polovica.

Sformulujeme problém ako úlohu lineárneho programovania a vyriešime pre dáta

$$A = \begin{bmatrix} -1 & 0.4 & 0.8 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix}, \qquad b = \begin{bmatrix} 1 \\ 0 \\ 0.3 \end{bmatrix}, \qquad x_{des} = \begin{bmatrix} 7 \\ 2 \\ -6 \end{bmatrix}, \qquad N = 30.$$

Riešenie:

Náš systém si vieme prepísať ako:

$$\begin{aligned} x(0) &= 0 \\ x(1) &= bu(0) \\ x(2) &= Abu(0) + bu(1) \\ x(3) &= A^2bu(0) + Abu(1) + bu(2) \\ &\vdots \\ x(N) &= A^{N-1}bu(0) + A^{N-2}bu(1) + \cdots + Abu(N-2) + bu(N-1) = x_{des} \end{aligned}$$

Podľa posledného riadku si zadefinujem

$$\begin{aligned} \text{maticu} \qquad & C = \begin{bmatrix} A^{N-1}b & A^{N-2}b & \cdots & Ab & b \end{bmatrix} \\ \text{a vektor} \qquad & u^T = \begin{bmatrix} u(0) & u(1) & \cdots & u(N-1) \end{bmatrix}. \\ \text{a teda} \qquad & Cu = x_{des} \end{aligned}$$

Pridaním jednej premennej t vieme f zmeniť na lineárne obmedzenie

$$|\alpha| \leq t$$

$$2\,|\alpha| - 1 \leq t$$

Lineárny problém znie:

$$\begin{aligned} \min \quad & \mathbf{1}^T t \\ & -t \leq u \leq t \\ & -t \leq 2u - 1 \leq t \\ & Cu = x_{des}, \end{aligned}$$

kde u a t sú premenné.

Nasledujúci kód rieši úlohu

```
160 A=[-1 0.4 0.8; 1 0 0; 0 1 0];
161 b=[1;0;0.3];
162 xdes=[7; 2; -6];
163 N=30; C=[];
164 for i=0:N-1
165     C=[A^i*b C];
166 end
167 cvx_begin quiet
168     variables t(N) u(N)
169     minimize ones(1,N)*t
170     subject to
171       -t<=u<=t
172       -t-ones(N,1)<=2*u<=t+ones(N,1)
173       C*u==xdes
174 cvx_end
175 F=cvx_optval
```

Optimálna hodnota spotreby paliva je $F = 17.3$,

Záver

Hlavným cieľom práce bolo vytvoriť praktickú príručku k modelovaciemu systému cvx. Pre lepšie pochopenie problematiky, sme v kapitole č. 1 stručne vysvetlili základ matematického a konvexného programovania, t.j. akú má daný problém formu a aké musí spĺňať podmienky. Hlavný zdroj, ktorý nám dopomáhal k efektívnemu zhrnutiu dôležitých faktov bol [7] a [8].

Podobne sme pokračovali v kapitole č.2 objasnením fundamentálnej stavby príkazov [2], ktoré sme použili na vyriešenie dvoch najzákladnejších problémov konvexného programovania. Sú to úlohy minimalizácie normy $Ax - b$ pre maticu A s plnou hodnosťou (5-8) a bez nej (9-14). Pri každej z možností sme výsledky cvx porovnali riešením z Matlabu (4) a (15).

Následne sme riešili niekoľko praktických úloh z rôznych oblastí života. Väčšina úloh si vyžadovala odlišný spôsob riešenia a nutnosť použiť inú schopnosť systému cvx.
I keď sa to možno nezdá, cvx je vytvorený tak, aby zvládol aj omnoho náročnejšie úlohy, ako tie, na ktorých sme vysvetľovali jeho fungovanie. Som si však istý, že táto práca dokáže zaujať každodennou tématikou úloh, a poskytnúť poznatky ku konvexnému programovaniu a podnietiť k ich riešeniu pomocou cvx. Je to totiž systém ktorý dokáže efektívne a rýchlo vyriešiť zadaný konvexný problém a ušetriť čas vo vede či vo výučbe.

Zoznam použitej literatúry

[1] Belk J.: *Convexity, Inequalities, and Norms*, dostupné na internete (30.05.2012): http://math.bard.edu/belk/math461/

[2] Boyd S., Grant M.: *cvx Users' Guide*, dostupné na internete (01.02.2012): http://cvxr.com/cvx/download/

[3] Boyd S., Grant M.: *CVX: Matlab software for disciplined convex programming, version 1.21*, dostupné na internete (30.05.2012): http://cvxr.com/cvx/

[4] Boyd S., Grant M.: *Graph implementations for nonsmooth convex programs*, Recent Advances in Learning and Control, Springer, 2008, 95-110, dostupné na internete (30.05.2012): http://stanford.edu/~boyd/graph_dcp.html

[5] Boyd S., Grant M., Ye Y.: *Disciplined Convex Programming*, dostupné na internete (29.05.2012): http://www.stanford.edu/~boyd/papers/disc_cvx_prog.html

[6] Boyd S., Vandenberghe L.: *Additional Exercises for Convex Optimization*, dostupné na internete (30.03.2012): http://www.stanford.edu/~boyd/cvxbook/

[7] Boyd S., Vandenberghe L.: *Convex Optimization*, Cambridge University Press, Cambridge, 2004, dostupné na internete (20.05.2012): http://www.stanford.edu/~boyd/cvxbook/

[8] Boyd S., Vandenberghe L.: *Lecture slides*, dostupné na internete (30.05.2012): http://www.stanford.edu/~boyd/cvxbook/

[9] Kotler Z.: *Convex Optimization Overview*, dostupné na internete (30.05.2012): http://people.csail.mit.edu/kolter/doku.php?id=other

Príloha A

[A1] Generátor údajov ku (3.4.1)

treatment_planning_data.m, dostupné na internete(30.5.2012):

http://www.stanford.edu/∼boyd/cvxbook/cvxbook_additional_exercises/