ŘEŠENÍ LINEÁRNÍCH MATICOVÝCH NEROVNOSTÍ V MATLABU[1]

Zdeněk Hurák, Michael Šebek

Centrum aplikované kybernetiky
České vysoké učení technické, Praha, ČR

e-mail: z.hurak@c-a-k.cz, m.sebek@c-a-k.cz

Abstrakt: Článek nabízí velmi stručný úvod do problematiky lineárních maticových nerovností a semidefinitního programování a podává přehled dostupných softwarových solverů pro pracujících v prostředí Matlab, včetně základního popisu komerčního produktu LMI Control Toolbox for Matlab.

Klíčová slova: konvexní optimalizace, lineární maticové nerovnosti, semidefinitní programování, LMI Control Toolbox for Matlab, SeDuMi.

I. LINEÁRNÍ MATICOVÉ NEROVNOSTI

Lineární nerovnost obyčejná je nerovnost typu $3x + 5y \geqslant 6$. Proměnnými jsou v tomto případě skaláry x a y. Lineární maticová nerovnost, anglicky *linear matrix inequality (LMI)* je definovaná vlastně úplně stejně, pouze místo konstant 3, 5 a 6 budou matice.

Lineární maticová nerovnost je definovaná jako

$$F(x) \geqslant 0$$

kde

$$F(x) = F_0 + \sum_{i=0}^{m} F_i x_i$$

Přitom $x = [x_0\, x_1 \ldots x_m]^T$ je m-rozměrná promměnná, zadáno je $m+1$ symetrických matic $F_0, F_1, \ldots, F_m \in R^{n \times n}$ a znaménko nerovnosti je interpretováno jako positivní semidefinitnost matice.

Běžná je i ostrá nerovnost.

II. SEMIDEFINITNÍ PROGRAMOVÁNÍ

Lineární maticové nerovnosti mají jednu mimořádně užitečnou vlastnost pro konvexní optimalizaci – definují konvexní množinu. A tak je výhodné použít je pro popis množiny přípustných řešení pro optimalizační úlohy.

[1] Tato práce byla podpořena Ministerstvem školství a tělovýchovy ČR v rámci projektu LN00B096.

Semidefinitní program (SDP) je konvexní optimalizační úloha zadaná jako

$$min\, c^T x$$

za omezujících podmínek

$$F_0 + \sum_{i=0}^{m} F_i x_i \geqslant 0$$

přičemž c je vektor a F_i jsou konstatní symetrické matice.

III. Vlastnosti a příklady semidefinitních programů

a) Lineární program jako speciální případ semidefinitního programu

Běžný LP zapsaný jako, $min\, c^T x$, za omezujících podmínek $Ax + b \geq 0$
kde nerovnost je chápána jako nerovnost *po prvcích,* můžeme snadno přepsat jako SDP s $F(x) = diag(Ax + b)$, neboli, $F_0 = diag(b), F_i = diag(a_i), i = 1, \ldots m$, kde a_i jsou sloupce matice A.

b) Nelineární konvexní optimalizace jako semidefinitní program

Příkladem nelineární, ale konvexní optimalizační úlohy, která může být formulována jako SDP (nikoliv však jako LP), je

$$min \frac{(c^T x)^2}{d^T x}$$

za omezujících podmínek $Ax = b \geq 0$ je možné [1] vyjádřit jako $min\, t$ za omezujících podmínek

$$\begin{bmatrix} diag(Ax + b) & 0 & 0 \\ 0 & t & c^T x \\ 0 & c^T x & d^T x \end{bmatrix} \geq 0$$

c) Maximalizace vlastních čísel

Mějme symetrickou maticovou funkci $A(x)$, která afinně závisí na vektorové proměnné $x \in R^k$. Problém nalezení maximálního vlastního čísla může být formulován jako SDP

$$min\, t$$

za omezujících podmínek daných LMI

$$\begin{bmatrix} tI & A(x) \\ A(x)^T & I \end{bmatrix} \geq 0$$

IV. Aplikace v teorii řízení

Užitečnější forma lineární maticové nerovnosti v teorii řízení a zpracování signálů je

$$F_0 + F_1 X_1 + \ldots + F_m X_m \geq 0$$

kde F_i jsou zadané symetrické matice a X_i jsou promměnné symetrické matice. Je triviální vidět, že původní (kanonickou) verzi LMI lze snadno převést na tuto maticovou a obráceně. Při výběru software je dobré zjistít, jako formu podporuje, neboť již pro vetší než

jen školní příklady je převod z jednoho tvaru do druhého mnohdy náročnější než samotná optimalizace.

Typickým příkladem využití LMI v řízení je Ljapunovova věta

$$A^T X + XA < 0, X > 0$$

Tu lze snadno formulovat jako problém existence (jakéhokoliv) řešení lineární maticové nerovnosti

$$\begin{bmatrix} -A^T X - XA & 0 \\ 0 & X \end{bmatrix} > 0$$

Zajímavé možná je, že povědomí o tom, ze mnoho úloh z teorie automatické řízení je možno formulovat jako LMI je v řídicí komunitě již přes sto let (již od doktorské práce Lyapunova), ale až teprve rozvoj numerických algoritmů pro řešení SDP na konci osmdesátých let zůsobil skutečný boom v teorii řízení [2].

V. Přehled dostupných solverů pro Matlab

Kromě známého LMI Control Toolboxu vyvíjeného a distribuovaného firmou The Mathworks existuje celá řada volně dostupných a vysoce výkonných kódů. Namátkou: SDPA, SeDuMi, CSDP, DSDP, SDPT3, MOSEK, LOQO, BMPR, BMZ, BUNDLE. Jejich srovnání lze najít na webové stránce [12], další stručné a přehledné srovnání lze najít v poznámkách k přednášce [11].

V komunitě automatického řízení je v posledních letech mimořádně oblíbený balík *SeDuMi* [8] Jose Sturma. Ve spojení s objektově orientovaným matlabských rozhraním nazvaným *SeDuMi Interface* [9] vytvořeným Dimitri Peacellem, jde o velice silný nástroj, který se vhodně doplňuje s *LMI Control Toolboxem* [10].

VI. LMI Control Toolbox for Matlab

(Tato část byla zčásti vytvořena podle manuálu LMI Control Toolbox [10], který až donedávna ale nebyl k dispozici v elektronické podobě)

LMI Control Toolbox for Matlab je velice komplexní a silný nástroj pro řešení SDP a obsahuje již i sadu aplikačních funkcí pro návrh optimální a robustních regulátorů. Pro samotné řešení lineárních maticových nerovností z tohoto toolboxu ale využijeme nástroj, kterému se říka *LMI Lab.* Ten umožňuje

- zadat systém lineárních maticových nerovností buď symbolicky s použitím pohodlného LMI Editoru, nebo z příkazové řádky příkazy **`lmivar`** a **`lmiterm`**.
- získat informace o existujícím systému lineárních maticových nerovností.modifikovat existující systém lineárních maticových nerovností.
- řešit tři základní LMI problémy
  1. existence přípustného řešení (feasibility problem) - **`feasp`**
  2. minimalizace lineární nákladové funkce (linear objective minimization) - **`mincx`**
  3. minimalizace zobecněného vlastního čísla (generalized eigenvalue minimization) – **`gevp`**
- vyhodnotit výsledek optimalizace

Veškeré informace o daném LMI systému, jak byly uvedeny v předchozí kapitole, jsou uchovávány v tzv. *interní reprezentaci* v jediném vektoru označeném genericky jako

LMISYS. Není ale potřeba snažit se porozumět, jakým způsobem jsou do něj tyto informace zadány. K získání jakékoliv informace, či k případné modifikaci můžeme použít příkazy LMI Labu. Tyto nástroje je možné rozdělit do následujících skupin

Zadávání systému lineárních maticových nerovností
Buď jako symbolické maticové výrazy v interaktivním prostředí **lmiedit**, nebo z příkazové řádky použitím příkazů **lmivar** a **lmiterm**. První volba je intuitivnější a jednodušší, druhá volba však nabízí větší fexibilitu a rozšiřující možnosti.

Získání informací o LMI systému
K získání informací o systému lineárních maticových nerovností použijeme funkci **lmiinfo**, případně můžeme celý systém vizualizovat v interaktivním prostředí **lmiedit**.

Řešení LMI optimalizačních úloh
V jazyce C napsané optimalizační rutiny pro tři základní LMI optimalizační problémy (**feasp, mincx, gevp**). Jako výstupní parametry vrací vektor x^* přípustných nebo optimálních proměnných. Odpovídající hodnoty maticových proměnných X_1^*, …, X_k^* lze získat funkcí **dec2mat**.

Analýza a validace výsledků
Výsledek x^* dodaný LMI řešitelem (solverem) lze jednoduše otestovat funkcemi **evallmi** a **showlmi**. Toto umožňuje rychlou kontrolu a/nebo analýzu výsledků. Funkce **evallmi** dosadí do všech proměnných v zadaném LMI systému hodnoty x^* vypočtené solverem. Levá i pravá strana LMI se tak stávají konstantními maticemi, které mohou být zobrazeny funkcí **showlmi**.

Modifikace LMI systému
Existující systém lineárních maticových nerovností může být modifikován dvěma způsoby
- LMI může být odstraněna ze systému příkazem **dellmi**
- maticová proměnná může být vymazána příkazem **delmvar**. Kromě toho, příkazem **setmvar** můžeme nastavit některé z maticových proměnných na nějakou konstantní hodnotu a řešit systém vzhledem ke zbývajícím maticovým proměnným.
- Kromě toho samozřejmě můžeme opět použít pohodlný editor **lmiedit**

Zadávání systému lineárních maticových nerovností

Při zadávání systému lineárních maticových nerovností postupujeme ve dvou krocích

[1] deklarace rozměrů a struktury maticových proměnných $X_1, X_2, \ldots, X_k$
[2] zadání elementárních členů v jednotlivých blocích LMI

Jak už bylo zmíněno v úvodu, toto je možné udělat dvěma způsoby: (1) specifikovat celý system z příkazové řádky pomocí příkazů **lmivar** a **lmiterm**. (2) zadat LMI systém pomocí symbolických výrazů v interaktivním GUI , které se vyvolá příkazem **lmiedit**. První způsob je sice velice flexibilní a poskytuje při zadávání spoustu možností, je však pro začátečníky poněkud nepřehledný. Proto jeje nebudeme v tomto stručném úvodu více rozebírat. Zájemce odkazujeme na manuál k *LMI Control Toolbox*u, či on-line help. Velice užitečné a dostatečně vysvětlující je rovněž demo vyvolané příkazem **lmidem**. Doporučujeme každému si jej zběžně prohlédnout.

Zadávání LMI systému pomocí interaktivního GUI rozhraní **lmiedit** je intuitivní a pro počáteční seznámení dokonale postačí. Uvedeme teď příklad, který můžete sledovat i v již zmíněném demu.

Příklad 1
Tento příklad ilustruje zadávání systému lineárních maticových nerovností v interaktivním prostředí **lmiedit**. Doporučujeme pustit si i demo **lmidem**, kde můžete celý proces zadávání sledovat, včetně používání příkazů **lmivar** a **lmiterm** pro zadávání z příkazové řádky.

Uvažujme stabilní přenosovou funkci

$$G(s) = C(sI - A)^{-1} B$$

se 4 vstupy, 4 výstupy a 6 stavy. Dále uvažujme množinu váhovacích matic D s blokově diagonální strukturou

$$D = \begin{pmatrix} d_1 & 0 & 0 & 0 \\ 0 & d_1 & 0 & 0 \\ 0 & 0 & d_2 & d_3 \\ 0 & 0 & d_4 & d_5 \end{pmatrix}$$

Následující problém je potřeba řešit při analýze robustní stability systémů s časově proměnnými nejistotami:

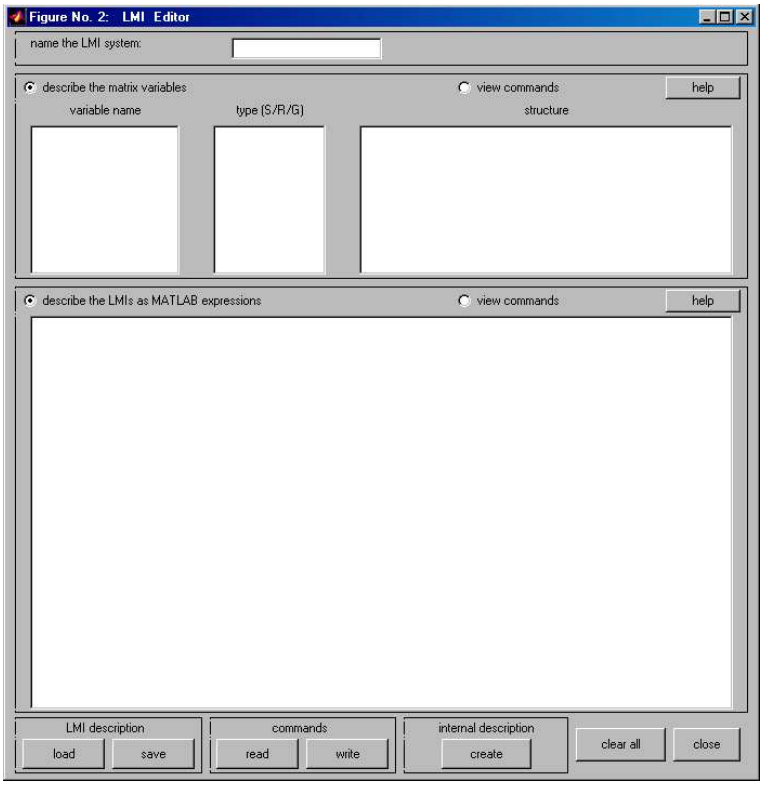
> *Najděte, pokud existuje, matici D se strukturou (3.1) takovou, že maximální zisk systému $DG(s)D^{-1}$ na celém frekvenčním rozsahu je menší než jedna.*

Problém má jednoduchou LMI formulaci: matice D existuje tehdy a jedině tehdy, když následující problém má řešení (tzv. feasibility problem; feasibility = přípustnost, uskutečnitelnost, proveditelnost)

> *Najděte dvě symetrické matice X a $S = D^T D$ takové, že platí*

$$\begin{pmatrix} A^T X + XA + C^T SC & XB \\ B^T X & -S \end{pmatrix} < 0$$
$$X > 0$$
$$S > I$$

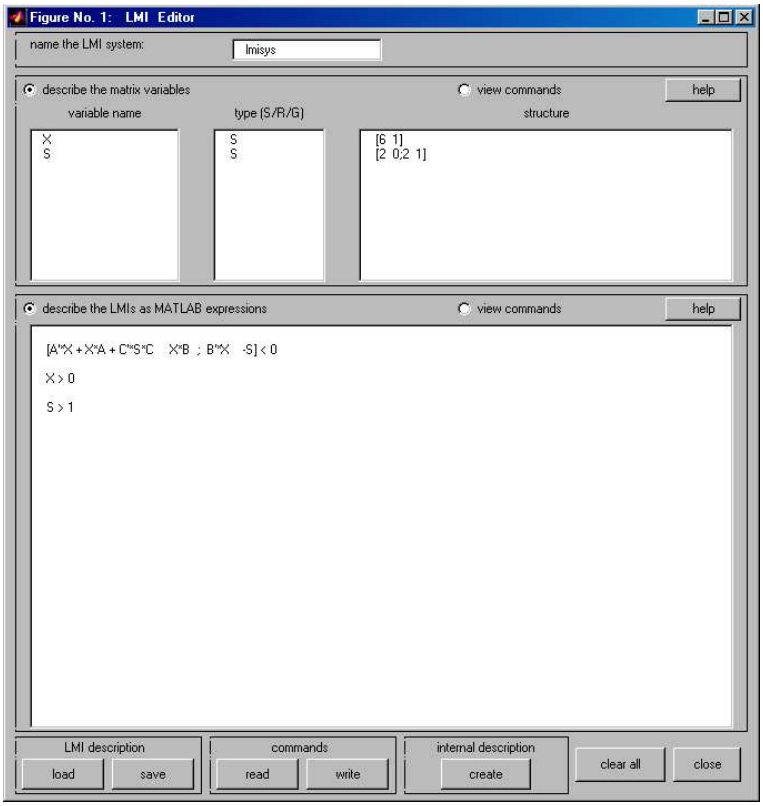
Napsáním příkazu **lmiedit** do příkazové řádky Matlabu vyvoláme interaktivní GUI



Bráno shora dolů a zleva doprava, zadáváme postupně:

1. jméno pod kterým se budeme při dalších výpočtech na náš LMI systém odvolávat.
2. jméno maticové proměnné. V našem případě to bude *X* a *S*
3. typ maticové proměnné: S = symetrická matice, R = nestrukturovaná matice, G = jiný typ.
4. informace o strukture maticové proměnné. Syntaxe této položky pojmenované *structure* odpovídá syntaxi položky *struct* v příkazu **`lmivar`**. Tedy v našem případě víme, že *X* je symetrická matice 6x6 a píšeme [6 1]. Matice *S* je opět symetrická matice, ovšem obsahující dva bloky: první blok je 2x2 diagonální matice a druhý blok je 2x2 synetrická matice. Píšeme tedy do druhého řádku v poli *structure* [2 0;21]. Je zřejmé, že každý řádek odpovídá jednomu bloku a první prvek v řádku popisuje rozměr bloku, zatímco druhý prvek v řádku popisuje typ bloku (1 = symetrický, 2 = skalární, -1 = nulový). Více ale najdete v on-line nápovědě pro příkaz **`lmivar`**. Především se tam dozvíte, jak specifikovat složitější struktury maticových proměnných.
5. ve spodní části GUI vkládáme symbolické vyjádření systému lineárních maticových nerovností. V našem případě tedy píšeme [A'*X+X*A+C'*S*C X*B; B'*X –S]<0. Na další řádek pak X>0 a na další řádek S>1.

Při zadávání systému lineárních maticových nerovností pomocí lmiedit platí můžeme využívat následujících schopností editoru: nulové matice, stejně jako i jednotkové matice můžeme jednoduše psát jako 0 a 1. Nemusíme se tedy obtěžovat hledáním jejich správných rozměrů. Stejně tak si můžeme ulehčit práci při zadání symetrických matic, když zadáme pouze prvky nad hlavní diagonálou a pod ní na adekvátní místa píšeme pouze *. Celý systém tedy vypadá následovně



Nyní máme celý systém zadán, zbývá ještě vytvořit vnitřní reprezentaci systému, čili přiřadit vytvořený systém proměnné s námi zadaným jménem. Toho dosáhneme kliknutím na tlačítko create. Máme tedy vytvořen vnitřní popis systému a na ten se od této chíle můžeme odvolávat při používání LMI optimalizačních rutin.

Celý popis systému si můžeme uložit příkazem save pro pozdější opětovné nahrání do editoru příkazem load.

Zatržením volby view commands získáme posloupnost příkazů pro vytvoření totožného LMI systému ale z příkazové řádky. Jedno z možných využití této volby je, že se takto můžeme učit, jak používat příkazy **`lmivar`** a **`lmiterm`**. Celou posloupnost příkazů uložíme příkaze write. Příkazem read můžeme symbolicky zobrazit jinou námi vytvořenou posloupnost příkazů.

Získání informací o LMI systému

Jak již bylo zmíněno, uchovává LMI toolbox veškeré informace o námi zadaném systému lineárních maticových nerovností v jediném vektoru nazývaném tzv. *internal representation* (tyto pojmy uvádíme jen pro Vaši rychlejší orientaci v manuálu a on-line nápovědách). Místo abychom se pokoušeli číst přímo tento vektor, použijeme funkce **`lmiinfo`**, **`lminbr`** a **`matnbr`** k extrahování a zobrazení požadované informace.

`lmiinfo` je interaktivní nástroj pro získání informace o počtu lineárních maticových nerovností, počtu a struktuře maticových proměnných, o členech v jednotlivých blocích.

`lminbr` vrací počet lineárních maticových nerovností v zadaném LMI systému.

`matnbr` vrací počet maticových proměnných v zadaném LMI systému.

LMI solvery

LMI toolbox obsahuje pokročilé optimalizační rutiny pro řešení tří základních LMI optimalizačních úloh:

- nalezení přípustného řešení (feasibility problem):

  Nalezněte $x \in R^N$ (nebo ekvivalentně matice $X_1, \ldots, X_K$ s předepsanou strukturou), které vyhovují systému lieárních maticových nerovností

  $$A(x) < B(x)$$

  Odpovídající solver spustíme příkazem **`feasp`**.

- minimalizace lineární nákladové funkce s LMI omezujícími podmínkami

  Minimalizuj $c^T x$ nad $x \in R^N$ při omezujících podmínkách $A(x) < B(x)$

  Odpovídající solver spustíme příkazem **`mincx`**.

- minimalizace zobecněného vlastního čísla

  Minimalizuj nad $x \in R^N$ při omezujících podmínkách

  $$\begin{aligned} C(x) &< D(x) \\ 0 &< B(x) \\ A(x) &< B(x) \end{aligned}$$

  Odpovídající solver spustíme příkazem **`gevp`**.

Tři výše uvedené optimalizační rutiny berou jako vstupní argument interní reprezentaci LMISYS systému lineárních maticových nerovností a vrací jako výstup hodnotu x^* přípustného či optimálního řešení. Odpovídající hodnoty maticových proměnných jsou nalezeny z x^* použitím konvertoru **`dec2mat`**.

Poznámky k použití **`gevp:`**
Je nutno rozlišovat mezi standartní LMI omezující podmínkou $C(x) < D(x)$ a lineární frakcionální LMI omezující podmínkou $A(x) < B(x)$. Při používání **`gevp`** musíme dodržovat následující pravidla:

- podmínku $A(x) < B(x)$ zadáváme bez λ. Tedy jako $A(x) < B(x)$.
- a zadáváme tuto podmínku jako poslední v soustavě lineárních maticových nerovností. **`gevp`** potom automaticky předpokládá, že posledních L LMI jsou lineární fractionální, kde L je počet LMI nerovností obsahujících λ.
- přidejte podmínku $0 < B(x)$ či jakoukoliv jinou podmínku, která tuto podmínku zahrnuje. Tato podmínka je nutná pro *správnou definici* problému a není automaticky dosazována rutinou **`gevp.`**

Počáteční odhad může být dodán funkcím **mincx** a **gevp.** Použijte **mat2dec** pro získání vektoru x_{init} z počátečních matic $X_1, \ldots, X_K$. Kromě toho, v manuálu najdete spostu možností jak řídit optimalizační proces a chování solverů.

Následující příklad ilustruje použití **mincx** solveru.

Příklad 2

Minimalizujte *trace(X)* za omezující podmínky $A^T X + XA + XBB^T X + Q < 0$
(5.1)
přičemž

$$A = \begin{pmatrix} -1 & -2 & 1 \\ 3 & 2 & 1 \\ 1 & -2 & -1 \end{pmatrix}; \quad B = \begin{pmatrix} 1 \\ 0 \\ 1 \end{pmatrix}; \quad Q = \begin{pmatrix} 1 & -1 & 0 \\ -1 & -3 & -12 \\ 0 & -12 & -36 \end{pmatrix}$$

Lze ukázat, že matice X_{opt} je stabilizujícím řešením algebraické Ricattiho rovnice

$$A^T X + XA + XBB^T X + Q = 0$$

Můžeme toto řešení také najít s využitím standartního příkazu **care** a porovnat s výsledkem optimalizace **mincx.**

My ale nyní problém formulujeme jako LMI optimalizační úlohu. S využitím známého vztahu pro *Schurův doplněk* můžeme psát

Minimalizujte *trace(X)* za omezujících podmínek daných

$$\begin{pmatrix} A^T X + XA + Q & XB \\ B^T X & -I \end{pmatrix} < 0$$

Stopa matice (zde používáme označení běžné v zahraniční literatuře – trace) je lineární funkcí prvků matice. Můžeme tedy použít LMI solver **mincx.**

1. Nejdříve ale potřebujeme vytvořit interní reprezentaci LMI systému. K tomu použijeme interaktivní GUI **lmiedit**. Nezapomeňte, že sice zadáváte matice *A, B a Q* v symbolické formě, ale předtím už musí být jejich hodnoty známy v matlabském pracovním prostoru.
2. Nyní potřebujeme vyjádřit stopu matice ve tvaru $c^T x$. Jelikož *c* má vybírat diagonální prvky matice *X*, získáme jej pomocí funkce **mat2dec** pro převod maticové proměnné do vektorové proměnné (inverzní funkce k fci **dec2mat**) pro matici $X = I$. Tedy

   ```
   c = mat2dec(riccati_sol,eye(3));
   ```

   Poznámka: příkaz **defcx** nabízí systematičtější přístup ke specifikaci lineárních nákladových funkcí.
3. Použijte **mincx** k nalezení minimalizujícího řešení x^* a globálního minimuma nákladové funkce $c^* = c^{*T} x^*$. Na matlabskou příkazovou řádku tedy píšeme

   ```
   options = [1e-5,0,0,0,0];       %specifikace relativni presnosti
   [copt,xopt] = mincx(riccati_sol,c,options)
   ```

na obrazovce se nam objeví následující hlášení o průběhu i výsledku optimalizace

```
Solver for linear objective minimization under LMI constraints

 Iterations   :    Best objective value so far

     1
     2                  -8.511476
     3                 -13.063640
***                  new lower bound:   -34.023978
     4                 -15.768450
***                  new lower bound:   -25.005604
     5                 -17.123012
***                  new lower bound:   -21.306781
     6                 -17.882558
***                  new lower bound:   -19.819471
     7                 -18.339853
***                  new lower bound:   -19.189417
     8                 -18.552558
***                  new lower bound:   -18.919668
     9                 -18.646811
***                  new lower bound:   -18.803708
    10                 -18.687324
***                  new lower bound:   -18.753903
    11                 -18.705715
***                  new lower bound:   -18.732574
    12                 -18.712175
***                  new lower bound:   -18.723491
    13                 -18.714880
***                  new lower bound:   -18.719624
    14                 -18.716094
***                  new lower bound:   -18.717986
    15                 -18.716509
***                  new lower bound:   -18.717297
    16                 -18.716695
***                  new lower bound:   -18.716873

 Result:  feasible solution of required accuracy
          best objective value:   -18.716695
          guaranteed relative accuracy: 9.50e-006
          f-radius saturation:  0.000% of R = 1.00e+009

copt =

  -18.7167

xopt =

   -6.3542
   -5.8895
   -6.2855
    2.2046
    2.2201
   -6.0771
```

V levém sloupci dostáváme pořadí iterace a hodnotu $c^T x$ v pravém sloupci. Všimněte si, že žádná hodnota není zobrazena pro první iteraci, jelikož přípustné řešení, tedy řešení splňující omezující LMI podmínku (5.2), bylo nalezeno až ve druhé iteraci. Po celkem 16 iteracích dostáváme hodnotu globálního minima *trace(X)* = -18.716695 s relativní

přesností 9.50e-006.

4. **mincx** také vrací hodnotu vektoru x = xopt, pro který bylo dosaženo minimální nákladové funkce. Odpovídající hodnotu optimální maticové proměnné lze získat jednoduše

   ```
   Xopt = dec2mat(riccati_sol,xopt,X)
   ```

   Tento výsledek můžeme porovnat s Riccatiho stabilizujícím řešením vypočítaným funkcí **care**:

   ```
   Xst = care(A,B,Q,-1);
   norm(Xopt-Xst)
   ans =
    6.5389e-005
   ```

Převod z vektorového formátu do maticového a naopak

Zatímco v teorii automatického řízení je pro nás výhodné zadávat lineární maticové nerovnosti s maticovými proměnnými $X_1, \ldots, X_k$, LMI solvery vrací jako výsledek optimalizace vektor x skalárních prvků. Pro převod mezi těmito dvěma formáty použijeme příkazy **mat2dec** a **dec2mat** (mat = matrix, dec = decision variable).

Mějme například LMI systém se třemi proměnnými maticemi X_1, X_2, X_3 . Pro konkrétní hodnoty těchto proměnných nalezneme odpovídající hodnotu ve vektorovém tvaru následovně

```
xdec = mat2dec(LMISYS,X1,X2,X3)
```

A naopak – máme-li jako výsledek optimalizace vektor xdec, potom například druhou matici v maticovém formátu získáme

```
X2 = dec2mat(LMISYS,xdec,2)
```

Počet maticových proměnných získáme příkazem **matnbr** a velikost vektorové proměnné (počet skalárních prvků) získáme příkazem **decnbr**. Kromě toho, příkaz **decinfo** poskytuje přesnou informaci o vztahu mezi zmíněnými dvěma formáty.

Analýza a validace výsledků LMI optimalizace

LMI Lab poskytuje dva nástroje k analýze a validaci výsleků LMI optimalizace.

evallmi najde hodnoty všech proměnných členů v LMI systému při dané hodnotě vektorové proměnné získané například jako přípustné nabo optimální řešení LMI optimalizace. Následně levá i pravá strana mohou být zobrazeny funkcí **showlmi**

V *Příkladu 2* jsme dostali jako výsledek minimalizace vektor xopt. Hledáme li hodnotu levé i pravé strany lineární maticové nerovnosti označené **riccati_sol**, postupujeme následovně

```
evlmi =  evallmi(riccati_sol,xopt);
[lhs,rhs] = showlmi(evlmi,1)
```

První příkaz vypočítá hodnotu LMI systému označeného riccari_sol pro vektor xopt a uloží ji do proměnné **evlmi**. Druhý příkaz vrací levou a pravou stranu první (a v tomto případě jediné) lineární maticové nerovnosti. Podmínka negativní definitnosti pak může být otestována jednoduše:

```
eig(lhs-rhs)

ans =
      -2.0387e-04
      -3.9333e-05
      -1.8917e-07
      -4.6680e+01
```

Další rady je možné najít manuálu k LMI Control Toolboxu [7] či v česky psaném stručném úvodu do LMI Labu [7].

VII. Závěr

V příspěvku byly uvedeny základní definice a vlastnosti lineárních maticových nerovností a semidefinitních programů a nastíněny jejich aplikace v teorii řízení a zpracování signálů. Byl uveden přehled dostupných sofwarových solverů, včetně jejich základních parametrů a odkazů na jejich webové stránky. Dále byl proveden stručný úvod do části jednoho softwarového balíků – komerčního LMI Control Toolboxu, který je volitelnou součástí Matlabské distribuce.

Literatura a odkazy na webové stránky

[3] Vandeberghe, L., Boyd, S. *Semidefinite programming*. In SIAM Review, 38(1): 49-95, March 1996. Též ke stažení na http://www.stanford.edu/~boyd/sdp.html

[4] Boyd, S., El Ghaoui, L., Feron, E., Balakrishnan, V. *Linear matrix inequalities in system and control theory*. Published by SIAM, volume 15 of Studies in Applied Mathematics, June 1994, ISBN 0-89871-334-X.

[5] Weiland, S., Scherer, C. *LMIs in Controller Analysis and Synthesis*. A graduate course at TU Delft. Ke stažení na http://www.er.ele.tue.nl/SWeiland/LMI.HTM

[6] 4. Dullerud, G.E., Paganini, F. *A course in robust control theory - a convex approach*. Springer- Verlag NewY ork, 2000.

[7] Hurak, Z. *LMI Lab - stručný úvod do používaní*. Ke stažení na http://ar.c-a-k.cz/hurak/LMI_Lab.pdf.

[8] Sturm, J., *SeDuMi* - Matlab toolbox for solving optimization problems over symmetric cones. Volně ke stažení na http://fewcal.kub.nl/sturm/software/sedumi.html

[9] Peaucelle, D., Taitz, K. *Sedumi Interface:* A user-friendly free Matlab package for defining optimisation problems over Linear Matrix Constraints (LMCs) http://www.laas.fr/~peaucell/SeDuMiInt.html

[10] The Mathworks. *LMI Control Toolbox*. Informace a manual na http://www.mathworks.com/access/helpdesk/help/toolbox/lmi/lmi_product_page.shtml

[11] Arzelier, D. Some notes on standard LMI solvers, a lecture at Department of Control Engineering, April 2002. Ke stažení na http://www.laas.fr/~arzelier/publis.html

[12] Mittelman, H. An Independent Benchmarking of SDP and SOCP Solvers. http://plato.asu.edu/dimacs/