Linear and Nonlinear Semidefinite Optimization by PENNON

Michal Kočvara

School of Mathematics, University of Birmingham, UK

Workshop
Robotics and Mathematics
Birmingham, June 2019

Semidefinite optimization (SDP)

"generalized" mathematical optimization problem

$$\begin{aligned} &\min f(x) \\ &\text{subject to} \\ &\quad g(x) \geq 0 \\ &\quad \mathcal{A}(x) \succeq 0 \end{aligned}$$

$\mathcal{A}(x)$ — (non)linear matrix operator $\mathbb{R}^n \to \mathbb{S}^m$

Linear SDP: primal-dual pair

$$\inf_X \langle C, X\rangle := \mathrm{tr}(CX) \tag{P}$$
$$\text{s.t.} \quad \langle A_i, X\rangle = b_i,\ i = 1, \ldots, n$$
$$X \succeq 0$$

$$\sup_{y,S} \langle b, y\rangle := \sum b_i y_i \tag{D}$$
$$\text{s.t.} \quad \sum y_i A_i + S = C$$
$$S \succeq 0$$

Linear Semidefinite Programming

Vast area of applications...

- LP and CQP is SDP
- eigenvalue optimization
- robust programming
- control theory
- relaxations of integer optimization problems
- approximations to combinatorial optimization problems
- structural optimization
- chemical engineering
- machine learning
- many many others...

Why nonlinear SDP?

Problems from

- Structural optimization
- Control theory
- Nearest correlation matrix
- Conic geometric programming
- and more...

PENNON collection

PENNON (PENalty methods for NONlinear optimization)
a collection of codes for NLP, (linear) SDP and BMI

– one algorithm to rule them all –

- PENNON (NLP+SDP) extended AMPL, MATLAB, C/For
- PENSDP MATLAB/YALMIP, SDPA, C/Fortran
- PENBMI MATLAB/YALMIP, C/Fortran

- PENLAB (NLP+SDP) open source MATLAB

- PENCOR C++, multithread

PENNON focuses on:

Large-scale linear SDP

– using iterative solvers, we can solve problems with $\sim 10^5$ variables

Nonlinear SDP

– bilinear and polynomial matrix inequalities

Polynomial optimization

– robust SDP solutions, very large scale problems, new algorithms

H2020 ITN POEMA (2019–2022)

PENSDP with an iterative solver

$$\min_{X \in \mathbb{S}^m} \langle C, X \rangle \qquad \text{(P)}$$
$$\text{s.t. } \langle A_i, X \rangle = b_i, \ i = 1, \ldots, n$$
$$X \succeq 0$$

Large n, small m (Kim Toh)

problem	n	m	PENSDP CPU	PEN-PCG CPU	PEN-PCG CG/it
theta12	17979	600	27565	254	8
theta162	127600	800	mem	13197	13

Large m, small/medium n (robust QCQP, Martin Andersen)

problem	n	m	PENSDP CPU	PEN-PCG CPU	PEN-PCG CG/it
rqcqp1	101	806	324	20	4
rqcqp2	101	3206	6313	364	4

PENNON: the problem

Optimization problems with nonlinear objective subject to nonlinear inequality and equality constraints and semidefinite bound constraints:

$$\min_{x\in\mathbb{R}^n, Y_1\in\mathbb{S}^{p_1},\ldots,Y_k\in\mathbb{S}^{p_k}} f(x,Y)$$

$$\begin{aligned} \text{subject to} \quad & g_i(x,Y) \le 0, && i=1,\ldots,m_g \\ & h_i(x,Y) = 0, && i=1,\ldots,m_h \\ & \underline{\lambda}_i I \preceq Y_i \preceq \overline{\lambda}_i I, && i=1,\ldots,k \end{aligned} \qquad \text{(NLP-SDP)}$$

PENNON interfaces

- Matlab
- AMPL
- C/C++ and Fortran

Linear SDP: standard input (SeDuMi, SDPA) or call from C/C++

For nonlinear functions it needs first and second derivatives

PENLAB

PENLAB — free, open source, fully functional version of PENNON coded in Matlab

Designed and coded by Jan Fiala (NAG)

- Open source, all in MATLAB (one MEX function)
- The basic algorithm is identical
- Some data handling routines not (yet?) implemented
- PENLAB runs just as PENNON but is slower

Pre-programmed procedures for

- linear SDP (reading SDPA input files)
- bilinear SDP (=BMI)
- SDP with polynomial MI (PMI)
- static output feedback (SOF)
- and more (NCM, QP, robust QP, TTO...)

PENLAB

Solving a problem:

- prepare a structure `penm` containing basic problem data
- `>> prob = penlab(penm);` MATLAB class containing all data
- `>> solve(prob);`
- results in class `prob`

The user has to provide MATLAB functions for

- function values
- gradients
- Hessians (for nonlinear functions)

of all $f, g, \mathcal{A}$.

Linear SDP with SDPA input

Pre-programmed. All you need to do:

```
>> sdpdata=readsdpa('datafiles/arch0.dat-s');
>> penm=sdp_define(sdpdata);
>> prob=penlab(penm);
>> prob.solve();
```

Bilinear matrix inequalities (BMI)

Pre-programmed. All you need to do:

```
>> bmidata=define_my_problem; %matrices A, K, ...
>> penm=bmi_define(bmidata);
>> prob=penlab(penm);
>> prob.solve();
```

$$\min_{x \in \mathbb{R}^n} c^T x$$

s.t.

$$A_0^i + \sum_{k=1}^{n} x_k A_k^i + \sum_{k=1}^{n} \sum_{\ell=1}^{n} x_k x_\ell K_{k\ell}^i \succcurlyeq 0, \quad i = 1, \ldots, m$$

Polynomial matrix inequalities (PMI)

Pre-programmed. All you need to do:

```
>> load datafiles/example_pmidata;
>> penm = pmi_define(pmidata);
>> problem = penlab(penm);
>> problem.solve();
```

$$\min_{x\in\mathbb{R}^n} \frac{1}{2}x^T Hx + c^T x$$

$$\text{subject to} \quad b_{\text{low}} \le Bx \le b_{\text{up}}$$

$$\mathcal{A}_i(x) \succcurlyeq 0, \quad i = 1,\ldots,m$$

with

$$\mathcal{A}(x) = \sum_i x_{\kappa(i)} Q_i$$

where $\kappa(i)$ is a multi-index with possibly repeated entries and $x_{\kappa(i)}$ is a product of elements with indices in $\kappa(i)$.

Other pre-programmed modules

- Nearest correlation matrix
- Truss topology optimization (stability constraints)
- Static output feedback (input from COMPlib, formulated as PMI)
- Robust QP

Experience with other SDP algorithms

- ADMM
- decomposition of LMI constraints
- using special (sparsity) structure of the problem
- ...

More advance problems

- Bilevel optimization with SDP/SOCP on the low level

Static output feedback

Given a linear system with $A \in \mathbb{R}^{n\times n}$, $B \in \mathbb{R}^{n\times m}$, $C \in \mathbb{R}^{p\times n}$

$$\dot{x} = Ax + Bu$$
$$y = Cx$$

we want to stabilize it by static output feedback

$$u = Ky\,.$$

Find $K \in \mathbb{R}^{m\times p}$ s.t. the eigenvalues of the closed-loop system $A + BKC$ belong to the left half-plane (continuous-time stability).

Lyapunov stability theory: $A + BKC$ has all its eigenvalues in the open left half-plane if and only if there exists a matrix $P \in \mathbb{R}^{n\times n}$ such that

$$(A + BKC)^T P + P(A + BKC) \prec 0, \quad P = P^T \succ 0$$

Bilinear matrix inequality.

SOF H_2 problem

Let $h_i(t) = Ce^{At}B_i$ be the i-th column of the impulse response of the system G.

SOF H_2 problem Find a SOF gain K such that $A + BKC$ is Hurwitz and $\|G\|_2^2 = \sum \|h_i\|_2^2$ is minimal.

The problem (in K, Q):

$$\begin{aligned} & \min \operatorname{Trace}(CQC^T) \quad (= \|G\|_2^2) \\ \text{s.t.} \quad & (A + BKC)Q + Q(A + BKC)^T + BB^T \preceq 0 \\ & Q \succ 0 \end{aligned}$$

SOF H_2 problem

Equivalent to

$$\begin{aligned} & \min \operatorname{Trace}(X) \\ \text{s.t.} \quad & (A+BKC)Q + Q(A+BKC)^T + BB^T \preceq 0 \\ & Q \succ 0 \\ & \begin{bmatrix} X & C \\ C^T & Q \end{bmatrix} \succeq 0, \end{aligned}$$

in variables $K \in \mathbb{R}^{m\times p}, Q \in \mathbb{R}^{n\times n}, X \in \mathbb{R}^{p\times p}$.

SOF H_∞ problem

Find a SOF matrix F such that $A + BKC$ is Hurwitz and the $\|G\|_\infty$ is minimal.

BMI (in variables $K \in \mathbb{R}^{m\times p}, P \in \mathbb{R}^{n\times n}, \gamma \in \mathbb{R}$):

$$\begin{aligned} & \min \gamma \\ \text{s.t.} \quad & \begin{bmatrix} (A+BKC)^T P + P(A+BKC) & PB & C^T \\ B^T P & -\gamma I & D^T \\ C & D & -\gamma I \end{bmatrix} \prec 0 \\ & P \succ 0 \\ & \gamma > 0 \end{aligned}$$

Solving SOF problems by PENBMI

Ex.	CPU (sec)	N	M	n	p	m	maximal real EV of $A(F)$	$\mathcal{H}_2$-perf
AC1	0.8	27	17	5	3	3	-2.6507e-01	1.0070e-03
AC6	3.3	64	28	7	4	2	-8.7223e-01	3.7982e+00
HE1	0.2	15	14	4	1	2	-1.2101e-01	9.5462e-02
HE5	8.7	54	28	8	2	4	-9.8307e-03	5.4382e+00
REA1	1.0	26	16	4	3	2	-1.6809e+00	1.8204e+00
DIS1	6.4	88	32	8	4	4	-4.3379e-01	2.6601e+00
MFP	0.6	26	16	4	2	3	-3.2088e-02	9.1620e+00
TF2	37.7	44	25	7	3	2	-1.0000e-05	1.4491e-01
PSM	2.6	49	26	7	3	2	-7.8437e-01	1.5043e+00
NN4	0.4	26	16	4	3	2	-5.8731e-01	1.8752e+00
NN8	0.3	16	12	3	2	2	-4.6487e-01	2.2797e+00
NN15	0.4	20	13	3	2	2	-1.1779e-01	4.9028e-02
NN16	46.6	62	28	8	4	4	-8.0625e-06	3.0732e-01

SOF—How to avoid Lyapunov?

Recall SOF H_2 variables: $K \in \mathbb{R}^{m\times p}$, $Q \in \mathbb{R}^{n\times n}$, $X \in \mathbb{R}^{p\times p}$.

Typically: $n > p, m$, often $n \gg p, m$. Lyapunov variable dominates. But it is just auxiliary. Can we get rid of it?
Yes, using polynomial formulation.

Let $k = \mathrm{vec}K$, $k \in \mathbb{R}^{mp}$. Define the characteristic polynomial of $A + BKC$:

$$q(s,k) = \det(sI - A - BKC) = \sum_{i=0}^{n} q_i(k)s^i$$

Here $q_i(k) = \sum_\alpha q_{i\alpha}k^\alpha$ and $\alpha \in \mathbb{N}^{mp}$ are all monomial powers.
Stability: all roots of $q(s,k)$ belong to the stability region $\mathcal{D}$.

Hermite stability criterion

The roots of $q(s,k)$ belong to the stability region $\mathcal{D}$ iff

$$H(q) = \sum_{i=0}^{n} \sum_{j=0}^{n} q_i(k) q_j(k) H_{ij} \succ 0\,.$$

Coefficients H_{ij} depend only on the stability region $\mathcal{D}$.

E.g., for $n = 3$, $\mathcal{D}$ left half-plane:

$$q(s,k) = q_0(k) + q_1(k)s + q_2(k)s^2 + q_3(k)s^3$$

is stable when

$$H(q) = \begin{bmatrix} 2q_0q_1 & 0 & 2q_0q_3 \\ 0 & 2q_1q_2 - 2q_0q_3 & 0 \\ 2q_0q_3 & 0 & 2q_2q_3 \end{bmatrix} \succ 0$$

i.e., $q_0, q_2 > 0$ and $q_iq_2 > q_0$.

Hermite stability criterion

The roots of $q(s,k)$ belong to the stability region $\mathcal{D}$ iff

$$H(q) = \sum_{i=0}^{n} \sum_{j=0}^{n} q_i(k) q_j(k) H_{ij} \succ 0\,.$$

Coefficients H_{ij} depend only on the stability region $\mathcal{D}$.

In our case, $H(q) = H(k)$ depends polynomially on k:

$$H(k) = \sum_{\alpha} H_\alpha k^\alpha \succ 0$$

Polynomial matrix inequality

Lemma

Matrix K solves problem SOF iff $k := \mathrm{vec} K$ solves PMI

$$H(k) = \sum_{\alpha} H_\alpha k^\alpha \succ 0$$

where $H_\alpha = H_\alpha^T \in \mathbb{R}^{n \times n}$ and $\alpha \in \mathbb{N}^{mp}$ describes all monomial powers.

Compare

Lyapunov variables: $K \in \mathbb{R}^{m \times p}, Q \in \mathbb{R}^{n \times n}$.

PMI variables: $k \in \mathbb{R}^{mp}$ only.

Solving PMI–SOF problems by PENLAB

To solve COMPlib problems (F. Leibfritz):

```
function [K]=sof(fname)
%
pmidata=complib2pmi(fname);
penm=pmi_define(pmidata);
%
prob=penlab(penm); solve(prob); x = prob.x;
%
[A,B1,B,C1,C]=COMPleib(fname);
K = reshape(x(1:end-1),size(B,2),size(C,1));
eig(A+B*K*C)
```

Solving PMI–SOF problems by PENLAB

Problem	degree	n	mp	λ_{opt}	CPU (sec)	iter
AC1	5	5	9	$-0.871 \cdot 10^{0}$	2.1	24/55
AC6	4	7	8	$-0.114 \cdot 10^{4}$	0.8	17/17
AC12	6	4	12	$0.479 \cdot 10^{0}$	10.8	62/118
HE1	2	4	2	$-0.686 \cdot 10^{2}$	0.6	22/62
HE5	4	8	8	$0.131 \cdot 10^{2}$	1.5	32/66
REA1	4	4	6	$-0.726 \cdot 10^{2}$	0.9	33/92
DIS1	8	8	16	$-0.117 \cdot 10^{2}$	25.1	26/54
MFP	3	4	6	$-0.370 \cdot 10^{-1}$	0.8	20/20
TF2	4	7	6	$-0.949 \cdot 10^{-7}$	0.8	19/19
PSM	4	7	6	$-0.731 \cdot 10^{2}$	0.7	20/52
NN4	4	4	6	$-0.187 \cdot 10^{2}$	0.8	32/68
NN8	3	3	4	$-0.103 \cdot 10^{1}$	0.7	19/40
NN15	3	3	4	$-0.226 \cdot 10^{0}$	0.6	15/15
NN16	7	8	16	$-0.623 \cdot 10^{3}$	129.3	111/269