

# Introducing PENLAB a MATLAB code for NLP-SDP

Michal Kočvara

School of Mathematics, The University of Birmingham

jointly with

Jan Fiala

Numerical Algorithms Group

Michael Stingl

University of Erlangen-Nürnberg

Toulouse, July, 2013

# PENNON collection

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

*– one algorithm to rule them all –*

## READY

- PENNLP    AMPL, MATLAB, C/Fortran
- PENSDP    MATLAB/YALMIP, SDPA, C/Fortran
- PENBMI    MATLAB/YALMIP, C/Fortran

## (relatively) NEW

- PENNON (NLP + SDP)    extended AMPL, MATLAB, C/Fortran

# The problem

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

$$\begin{aligned} & \min_{x \in \mathbb{R}^n, Y_1 \in \mathbb{S}^{p_1}, \dots, Y_k \in \mathbb{S}^{p_k}} f(x, Y) \\ & \text{subject to} \quad g_i(x, Y) \leq 0, & i = 1, \dots, m_g \\ & \quad \quad \quad h_i(x, Y) = 0, & i = 1, \dots, m_h \quad (\text{NLP-SDP}) \\ & \quad \quad \quad \underline{\lambda}_i I \preceq Y_i \preceq \bar{\lambda}_i I, & i = 1, \dots, k. \end{aligned}$$

# The algorithm

Based on penalty/barrier functions  $\varphi_g : \mathbb{R} \rightarrow \mathbb{R}$  and  $\Phi_P : \mathbb{S}^p \rightarrow \mathbb{S}^p$ :

$$g_i(x) \leq 0 \iff p_i \varphi_g(g_i(x)/p_i) \leq 0, \quad i = 1, \dots, m$$
$$Z \preceq 0 \iff \Phi_P(Z) \preceq 0, \quad Z \in \mathbb{S}^p.$$

Augmented Lagrangian of (NLP-SDP):

$$F(x, Y, u, \underline{U}, \overline{U}, p) = f(x, Y) + \sum_{i=1}^{m_g} u_i p_i \varphi_g(g_i(x, Y)/p_i) \\ + \sum_{i=1}^k \langle \underline{U}_i, \Phi_P(\Delta_i I - Y_i) \rangle + \sum_{i=1}^k \langle \overline{U}_i, \Phi_P(Y_i - \bar{\lambda}_i I) \rangle;$$

here  $u \in \mathbb{R}^{m_g}$  and  $\underline{U}_i, \overline{U}_i$  are Lagrange multipliers.

# The algorithm

A generalized Augmented Lagrangian algorithm (based on R. Polyak '92, Ben-Tal–Zibulevsky '94, Stingl '05):

Given  $x^1, Y^1, u^1, \underline{U}^1, \overline{U}^1; p_i^1 > 0, i = 1, \dots, m_g$  and  $P > 0$ .  
For  $k = 1, 2, \dots$  repeat till a stopping criterium is reached:

- (i) Find  $x^{k+1}$  and  $Y^{k+1}$  s.t.  $\|\nabla_x F(x^{k+1}, Y^{k+1}, u^k, \underline{U}^k, \overline{U}^k, p^k)\| \leq K$
- (ii)  $u_i^{k+1} = u_i^k \varphi'_g(g_i(x^{k+1})/p_i^k), \quad i = 1, \dots, m_g$   
 $\underline{U}_i^{k+1} = D_{\mathcal{A}} \Phi_P((\underline{\lambda}_i l - Y_i); \underline{U}_i^k), \quad i = 1, \dots, k$   
 $\overline{U}_i^{k+1} = D_{\mathcal{A}} \Phi_P((Y_i - \overline{\lambda}_i l); \overline{U}_i^k), \quad i = 1, \dots, k$
- (iii)  $p_i^{k+1} < p_i^k, \quad i = 1, \dots, m_g$   
 $P^{k+1} < P^k.$

# Interfaces

How to enter the data – the functions and their derivatives?

- Matlab interface
- AMPL interface
- c/Fortran interface

Key point: Matrix variables are treated as vectors

# What's new

PENNON being implemented in NAG (The Numerical Algorithms Group) library

The first routines should appear in the NAG Fortran Library, Mark 24 (Autumn 2013)

By-product:

**PENLAB** — free, open, fully functional version of PENNON coded in MATLAB

# PENLAB

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

- 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

- standard NLP (with AMPL input!)
- linear SDP (reading SDPA input files)
- bilinear SDP (=BMI)
- SDP with polynomial MI (PMI)
- easy to add more (QP, robust QP, SOF, TTO...)



# PENLAB

The problem

$$\begin{aligned} & \min_{x \in \mathbb{R}^n, Y_1 \in \mathbb{S}^{p_1}, \dots, Y_k \in \mathbb{S}^{p_k}} f(x, Y) \\ & \text{subject to} \quad g_i(x, Y) \leq 0, & i = 1, \dots, m_g \\ & \quad \quad \quad h_i(x, Y) = 0, & i = 1, \dots, m_h \\ & \quad \quad \quad \mathcal{A}_i(x, Y) \succeq 0, & i = 1, \dots, m_A \\ & \quad \quad \quad \underline{\lambda}_i I \preceq Y_i \preceq \bar{\lambda}_i I, & i = 1, \dots, k \end{aligned} \quad (\text{NLP-SDP})$$

$\mathcal{A}_i(x, Y)$ ... nonlinear matrix operators

# 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  $f, g, \mathcal{A}$ .

## Structure `penm` and `f/g/h` functions

Example:  $\min x_1 + x_2 \quad \text{s.t.} \quad x_1^2 + x_2^2 \leq 1, \quad x_1 \geq -0.5$

```
penm = [];  
penm.Nx = 2;  
penm.lbx = [-0.5 ; -Inf];  
penm.NgNLN = 1;  
penm.ubg = [1];  
penm.objfun = @(x,Y) deal(x(1) + x(2));  
penm.objgrad = @(x,Y) deal([1 ; 1]);  
penm.confun = @(x,Y) deal([x(1)^2 + x(2)^2]);  
penm.congrad = @(x,Y) deal([2*x(1) ; 2*x(2)]);  
penm.conhess = @(x,Y) deal([2 0 ; 0 2]);  
% set starting point  
penm.xinit = [2,1];
```

# Toy NLP-SDP example 1

$$\min_{x \in \mathbb{R}^2} \frac{1}{2}(x_1^2 + x_2^2)$$

$$\text{subject to } B + A_1 x_1 + A_2 x_2 := \begin{pmatrix} 1 & x_1 - 1 & 0 \\ x_1 - 1 & 1 & x_2 \\ 0 & x_2 & 1 \end{pmatrix} \succeq 0$$

D. Noll, 2007

## Structure `penm` and `f/g/h` functions

```
B = [1 -1 0; -1 1 0; 0 0 1];
```

```
A{1} = [0 1 0; 1 0 0; 0 0 0];
```

```
A{2} = [0 0 0; 0 0 1; 0 1 0];
```

```
penm = [];
```

```
penm.Nx=2;
```

```
penm.NALIN=1;
```

```
penm.lbA=zeros(1,1);
```

```
penm.objfun = @(x,Y) deal(-.5*(x(1)^2+x(2)^2));
```

```
penm.objgrad = @(x,Y) deal(-[x(1);x(2)]);
```

```
penm.objhess = @(x,Y) deal(-eye(2,2));
```

```
penm.mconfun=@(x,Y,k)deal(B+A{1}*x(1)+A{2}*x(2));
```

```
penm.mcongrad=@(x,Y,k,i)deal(A{i});
```

## Example: nearest correlation matrix

Find a nearest correlation matrix:

$$\min_X \sum_{i,j=1}^n (X_{ij} - H_{ij})^2 \quad (1)$$

subject to

$$X_{ii} = 1, \quad i = 1, \dots, n$$

$$X \succeq 0$$

## Example: nearest correlation matrix

The condition number of the nearest correlation matrix must be bounded by  $\kappa$ .

Using the transformation of the variable  $X$ :

$$z\tilde{X} = X$$

The new problem:

$$\min_{z, \tilde{X}} \sum_{i,j=1}^n (z\tilde{X}_{ij} - H_{ij})^2 \quad (2)$$

subject to

$$z\tilde{X}_{ii} = 1, \quad i = 1, \dots, n$$

$$I \preceq \tilde{X} \preceq \kappa I$$

## Structure `penm` and `f/g/h` functions

```
function [f,userdata] = objfun(x,Y,userdata)
    YH = svec2(x(1).*Y{1}-userdata.H);
    f = YH(:)'*YH(:);
```

```
function [df, userdata]=objgrad(x,Y,userdata)
    YH=svec2(x(1).*Y{1}-userdata.H);
    df(1) = sum(2*svec2(Y{1})*YH);
    df(2:length(YH)+1) = 2*x(1).*YH;
```

```
function [ddf, userdata] = objhess(x,Y,userdata)
    YH=svec2(x(1).*Y{1}-userdata.H);
    yy = svec2(Y{1});
    n = length(yy); ddf = zeros(n+1,n+1);
    ddf(1,1) = 2*sum(yy.^2);
    ddf(1,2:n+1) = 2.*(x(1).*yy+YH);
    ddf(2:n+1,1) = 2.*(x(1).*yy'+YH');
    for i=1:n, ddf(i+1,i+1) = 2*x(1)^2; end
```



# NLP with AMPL input

Pre-programmed. All you need to do:

```
>> penm=nlp_define('datafiles/chain100.nl');  
>> prob=penlab(penm);  
>> prob.solve();
```

## NLP with AMPL input

problem	vars	constr.	constr. type	PENNON		PENLAB	
				sec	iter.	sec	iter.
chain800	3199	2400	=	1	14/23	6	24/56
pinene400	8000	7995	=	1	7/7	11	17/17
channel800	6398	6398	=	3	3/3	1	3/3
torsion100	5000	10000	$\leq$	1	17/17	17	26/26
lane_emd10	4811	21	$\leq$	217	30/86	64	25/49
dirichlet10	4491	21	$\leq$	151	33/71	73	32/68
henon10	2701	21	$\leq$	57	49/128	63	76/158
minsurf100	5000	5000	box	1	20/20	97	203/203
gasoil400	4001	3998	= & b	3	34/34	13	59/71
duct15	2895	8601	= & $\leq$	6	19/19	9	11/11
marine400	6415	6392	$\leq$ & b	2	39/39	22	35/35
steering800	3999	3200	$\leq$ & b	1	9/9	7	19/40
methanol400	4802	4797	$\leq$ & b	2	24/24	16	47/67

# Linear SDP with SDPA input

Pre-programmed. All you need to do:

```
>> sdpdata=readsdpdata('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.

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

## Polynomial matrix inequalities (PMI)

Pre-programmed. All you need to do:

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

$$\begin{aligned} \min_{x \in \mathbb{R}^n} & \frac{1}{2} x^T H x + c^T x \\ \text{subject to} & \quad b_{\text{low}} \leq Bx \leq b_{\text{up}} \\ & \quad \mathcal{A}_i(x) \succcurlyeq 0, \quad i = 1, \dots, m \end{aligned}$$

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

# Availability

PENNON: Free time-limited academic version of the code available

PENLAB: Free open MATLAB version available from NAG

# What's missing?

SOCP (Second-Order Conic Programming) - nonlinear,  
integrated in PENLAB (and PENNON)

Postdoctoral research position in Birmingham  
(sponsored by NAG)

- development of NL-SOCP algorithm (compatible with PENNON algorithm)
- implementation in PENLAB and PENNON
- Alain Zemkoho, started April 2013



# Sensor network localization

(Euclidean distance matrix completion, Graph realization)

We have (in  $\mathbb{R}^2$  (or  $\mathbb{R}^d$ ))

$n$  points  $a_i$ , **anchors** with **known** location

$m$  points  $x_i$ , **sensors** with **unknown** location

$d_{ij}$  **known** Euclidean distance between “close” points

$$d_{ij} = \|x_i - x_j\|, \quad (i, j) \in \mathcal{I}_x$$

$$\bar{d}_{kj} = \|a_k - x_j\|, \quad (k, j) \in \mathcal{I}_a$$

**Goal:** Find the positions of the sensors!

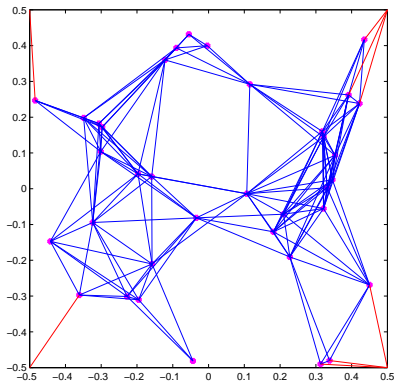
Find  $x \in \mathbb{R}^{2 \times m}$  such that

$$\|x_i - x_j\|^2 = d_{ij}^2, \quad (i, j) \in \mathcal{I}_x$$

$$\|a_k - x_j\|^2 = \bar{d}_{kj}^2, \quad (k, j) \in \mathcal{I}_a$$

# Sensor network localization

Example, 4 anchors, 36 sensors



# Sensor network localization

## Applications

- Wireless sensor network localization
  - habitat monitoring system in the Great Duck Island
  - detecting volcano eruptions
  - industrial control in semiconductor manufacturing plants
  - structural health monitoring
  - military and civilian surveillance
  - moving object tracking
  - asset location
- Molecule conformation
- ...

# Sensor network localization

Solve the least-square problem

$$\min_{x_1, \dots, x_m} \sum_{(i,j) \in \mathcal{I}_x} \left( \|x_i - x_j\|^2 - d_{ij}^2 \right)^2 + \sum_{(i,j) \in \mathcal{I}_a} \left( \|a_k - x_j\|^2 - \bar{d}_{kj}^2 \right)^2$$

to global minimum. This is an NP-hard problem.

# SDP relaxation

(P. Biswas and Y. Ye, '04)

Let  $X = [x_1 \ x_2 \ \dots \ x_n]$  be a  $d \times n$  unknown matrix. Then

$$\|x_i - x_j\|^2 = (e_i - e_j)^T X^T X (e_i - e_j)$$

$$\|a_k - x_j\|^2 = (a_k; -e_j)^T \begin{bmatrix} I_d \\ X^T \end{bmatrix} [I_d \ X] (a_k; -e_j)$$

and the problem becomes

$$(e_i - e_j)^T X^T X (e_i - e_j) = d_{ij}^2$$

$$(a_k; -e_j)^T Z (a_k; -e_j) = \bar{d}_{kj}^2$$

$$Z = \begin{pmatrix} I_d & X \\ X^T & X^T X \end{pmatrix}$$

$Z_{1:d,1:d} = I_d$ ,  $Z \succeq 0$ ,  $Z$  has rank  $d$

# SDP relaxation

Now relax

$$Z_{1:d,1:d} = I_d, \quad Z \succeq 0, \quad Z \text{ has rank } d$$

to

$$Z_{1:d,1:d} = I_d, \quad Z \succeq 0$$

Relaxed problem:

min 0

subject to

$$(0; e_i - e_j)^T Z (0; e_i - e_j) = d_{ij}^2 \quad \forall (i, j) \in \mathcal{I}_x$$

$$(a_k; -e_j)^T Z (a_k; -e_j) = \bar{d}_{kj}^2 \quad \forall (k, j) \in \mathcal{I}_a$$

$$Z_{1:d,1:d} = I_d$$

$$Z \succeq 0$$

Full SDP relaxation, FSDP (linear SDP)

# SDP relaxation

Equivalent formulation:

$$\begin{aligned} \min \quad & \sum_{(i,j) \in \mathcal{I}_x} \left( (0; \mathbf{e}_i - \mathbf{e}_j)^T Z (0; \mathbf{e}_i - \mathbf{e}_j) - d_{ij}^2 \right)^2 \\ & + \sum_{(k,j) \in \mathcal{I}_a} \left( (\mathbf{a}_k; -\mathbf{e}_j)^T Z (\mathbf{a}_k; -\mathbf{e}_j) - \bar{d}_{kj}^2 \right)^2 \end{aligned}$$

subject to

$$Z_{1:d,1:d} = I_d$$

$$Z \succeq 0$$

Full SDP relaxation, FSDP (nonlinear SDP)

# SDP relaxation

For larger problems, FSDP is not solvable numerically:

- many variables (number of sensors)
- large and full matrix constraint (although low-rank)

Can we exploit sparsity of  $\mathcal{I}_x$  and  $\mathcal{I}_a$  at the relaxation modelling level?

Recently several approaches:

- Wolkowicz
- Toh
- Kojima
- Su

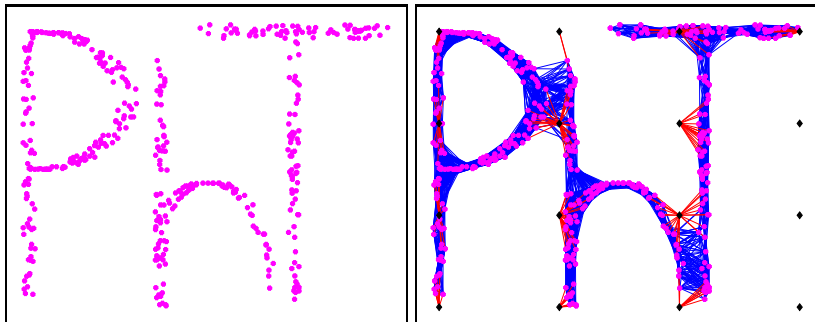
Simple, yet powerful way: **Edge-based relaxation (ESDP,** Wang-Zheng-Ye-Boyd, '08).



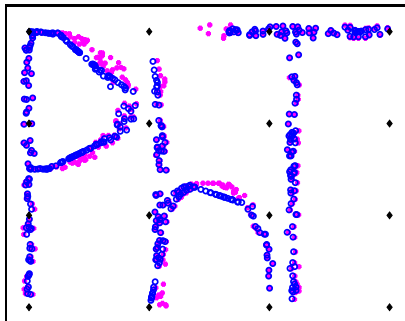
## Example, 16 anchors, 455 sensors



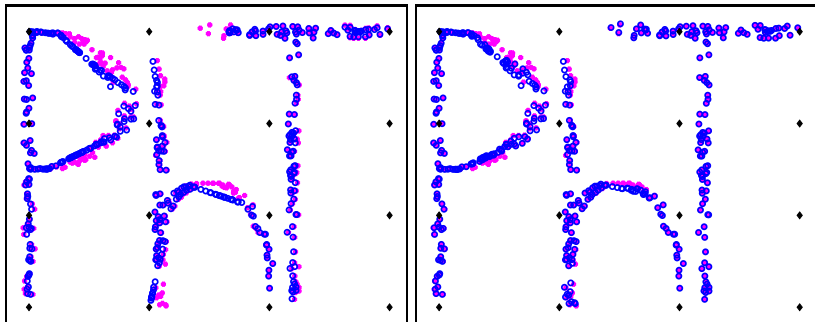
## Example, 16 anchors, 455 sensors



## Example, 16 anchors, 455 sensors



## Example, 16 anchors, 455 sensors



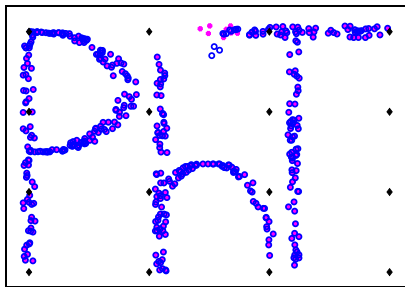
problem	rmsd	out-3	out-2
E-linear	0.0191	307	147
E-quadratic	0.0105	156	85

SDP: 6714 variables, 5349 ( $4 \times 4$ ) LMIs

# Solution refinement

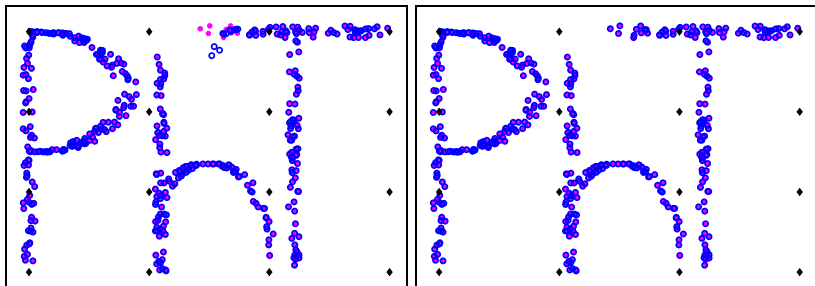
Take the SDP solution as initial approximation for the original unconstrained nonconvex problem. Solve both by PENNON.

## Example, 16 anchors, 455 sensors



problem	rmsd	out-3	out-2
E-linear	0.0191	307	147
orig from lin	0.0083	10	7

## Example, 16 anchors, 455 sensors



problem	rmsd	out-3	out-2
E-linear	0.0191	307	147
E-quadratic	0.0105	156	85
orig from lin	0.0083	10	7
orig from qua	0	0	0

Happy Birthday,

PNT