MATLAB tools in POEMA DB

Michal Kočvara, University of Birmingham

General online Julia training

POEMA 16 April 2021

Poema
Polynomial Optimization, Efficiency through Moments and Algebra

European Commission

Horizon 2020
European Union funding
for Research & Innovation

SDP with p linear matrix inequalities and explicit linear equality constraints:

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

subject to

$$\sum_{i=1}^{n} A_i^{(j)} x_i - A_0^{(j)} \succcurlyeq 0, \qquad j = 1, \ldots, p$$

$$c_k \; x - d_k \geq \; 0, \qquad k \in \mathcal{I}$$

$$c_k \; x - d_k = \; 0, \qquad k \in \mathcal{E}$$

with data $b \in \mathbb{R}^n$, $A_i^{(j)} \in \mathbb{S}^{m_j}$, $j = 1, \ldots, p$, $C \in \mathbb{R}^{q \times n}$, $d \in \mathbb{R}^q$. Denote by $Y^{(j)}$ the dual variable associated with the j-th LMI, $Y^{(j)} \in \mathbb{S}^{m_j}$.

Standard SDP software (Mosek, SDPT3, SeDuMi, CSDP, PENSDP, …) reads data in two popular formats:

- SDPA (ASCII file)
- SeDuMi (Matlab binary, includes also data for other conic inequalities)

In addition to the data, the SDP input format may also contain the following information, specific to POP relaxations:

- $Y^{(j)}$ has (unknown) low rank for some specified $j = 1, \ldots, p$
- when known, Rank($Y^{(j)}$) for specified $j = 1, \ldots, p$
- $A_i^{(j)}$ is a low rank matrix defined by r vectors as $A_i^{(j)} = \sum_{k=1}^{r} (a_i^j)_k \, (a_i^j)_k^\top$

SDPA format is not very MATLAB friendly, a nontrivial reader is needed.

POEMA-MATLAB format:

Data stored in a MATLAB structure with fields inherited from SDPA format and identical to `json` attributes.

Recall `json` attributes. Red attributes represent extension to data in SDPA format.

"type": "sdp"

"nvar" : n

"objective" : [$b_1, \ldots, b_p$]

"constraints" : {

"nlmi": number of linear matrix inequalities [p] (int)

"msizes": dimensions of the LMIs [$m_1, \ldots, m_p$]

"lmiduallr": (optional) binary vector of length nlmi, set to 1 if $Y^{(j)}$ assumed low rank

"lmidualrank": (optional) vector of length nlmi with ranks of $Y^{(j)}$ (when known a-priori)

"lmi_symat": matrices $A_i^{(j)}$ in upper triangular sparse format

"lmi_lrmat": matrices $A_i^{(j)}$ defined in low rank sparse format by r vectors $(a_i^j)_k$

"nlsi": (optional) number of linear inequalities [q]

"lsi_mat": (optional) matrix C in sparse format

"lsi_vec": (optional) vector d, [$d_1, \ldots, d_q$]

"lsi_op": (optional) binary vector of length nlsi, set to 0 for equality constraints ($k \in \mathcal{E}$) and 1 for inequality ($k \in \mathcal{I}$) constraints

}

json

```
{ "type" : ["sdp"],
"nvar" : n,
"objective" : [b_1,...,b_p] ,
"constraints" : {
   "nlmi" : p  ,
   "msizes" : [m_1,...,m_p] ,
   "lmiduallr" : [...],
   "lmidualrank" : [...] ,
   "lmi_symat" : [
        [ 1.0, 0, 2, 1, 2]
        [ 2.0, 0, 2, 2, 2]
        [ 2.0, 1, 1, 1, 1]
      ]
          ⋮
   },
}
```

POEMA-MATLAB

```
struct ("type", "sdp", ...
"nvar" , [n], ...
"objective" , [b_1,...,b_p] , ...
"constraints" , struct(...
   "nlmi" , [ p ] , ...
   "msizes" , [m_1,...,m_p] , ...
   "lmiduallr", [...], ...
   "lmidualrank", [...] , ...
   "lmi_symat", [...
        [ 1.0 0 2 1 2] ;...
        [ 2.0 0 2 2 2] ;...
        [ 2.0 1 1 1 1] ;...
      ], ...
          ⋮
   ), ...
);
```

Example:

$$\min_{x\in\mathbb{R}^3} x_1 + 2\,x_2 + 3\,x_3$$
subject to

$$\begin{bmatrix} 2 & -1 & 0 \\ -1 & 2 & 0 \\ 0 & 0 & 2 \end{bmatrix} x_1 + \begin{bmatrix} 2 & 0 & -1 \\ 0 & 2 & 0 \\ -1 & 0 & 2 \end{bmatrix} x_3 \succcurlyeq 0$$

$$\begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix} x_1 + \begin{bmatrix} 0 & 3 \\ 3 & 0 \end{bmatrix} x_2 - \begin{bmatrix} 0 & 1 \\ 1 & 2 \end{bmatrix} \succcurlyeq 0$$

MATLAB output:

```
sdp1 =

  struct with fields:

          type: "sdp"
          nvar: 3
     objective: [1 2 3]
   constraints: [1×1 struct]
          name: "My first example"
       comment: "Two LMIs"
```

```
sdp1 = struct(...
    "type","sdp",...
    "nvar", [3],...
    "objective", [ 1.0 2.0 3.0 ],...
    "constraints",struct(...
        "nlmi", [2],...
        "msizes", [ 3 2 ],...
        "lmi_symat", [...
        [ 1.0 0 2 1 2] ;...
        [ 2.0 0 2 2 2] ;...
        [ 2.0 1 1 1 1] ;...
        [ 2.0 1 1 2 2] ;...
        [ 2.0 1 1 3 3] ;...
        [-1.0 1 1 1 2] ;...
        [ 1.0 1 2 1 1] ;...
        [-1.0 1 2 2 2] ;...
        [ 3.0 2 2 1 2] ;...
        [ 2.0 3 1 1 1] ;...
        [ 2.0 3 1 2 2] ;...
        [ 2.0 3 1 3 3] ;...
        [-1.0 3 1 1 3] ;...
    ]),...
    "name","My first example",...
    "comment","Two LMIs"...
    )
```

POEMA-MATLAB format:

The good

- direct comparison with json
- editing/adding attributes

The not-so-good

- Files get very large for large problems (ASCII file)

POEMA-MATLAB SPARSE format

All data are again stored in a MATLAB structure but while most of the attributes remain the same, sparse matrices and vectors are converted to sparse MATLAB arrays.

POEMA-MATLAB

```
sdp1 =
struct with fields:
          type: "sdp"
          nvar: 3
          name: "My first example"
       comment: "Two LMIs"
     objective: [1 2 3]
   constraints: [1×1 struct]
-------------
>> sdp1.constraints.lmi_symat
ans =
     1     0     2     1     2
     2     0     2     2     2
     2     1     1     1     1
     2     1     1     2     2
     2     3     1     3     3
        ...
```

POEMA-MATLAB SPARSE

```
sdp1sp =
struct with fields:
       type: "sdp"
       nvar: 3
       name: "My first example"
    comment: "Two LMIs"
          c: [1 2 3]
       nlmi: 2
     msizes: [3 2]
          A: {2×4 cell}
-------------
sdp1sp.A{2,1}
ans =
   (2,1)        1
   (1,2)        1
   (2,2)        2
```

$$\begin{bmatrix} 0 & 1 \\ 1 & 2 \end{bmatrix}$$

MATLAB utilities in POEMA database

https://github.com/PolynomialMomentOptimization/Matlab-suite

`poema2json.m`	converts POEMA-MATLAB structure to POEMA json format
`poema2sdpa.m`	converts POEMA-MATLAB structure to SDPA format
`poema2sparse.m`	converts POEMA-MATLAB structure to POEMA-MATLAB SPARSE
`sdpa2poema.m`	reads SDPA sparse format and returns POEMA-MATLAB structure

For instance, in our IP code, we read SDPA files as follows:

```
>> d = sdpa2poema('database/problems/SDPA/tru7.dat-s');
>> d = poema2sparse(d);
>> ...
>> A{i,j} = d.A{i,j+1};
>> C{i}   = d.A{i,1};
```

Exercise:

```
% Call

>> ex1

% to generate a structure including data for Example 1. To convert it into
% json format and write it into an ASCII file, call

>> sdp_json = savejson([],sdp1);  %the output is a character array
>> fileID = fopen('myfile.json','w');
>> fprintf(fileID,'%s\n',sdp_json)

% To read the json format from myfile.txt and convert it back into a Matlab
% structure, call

>> sdp_json = fileread('myfile.json');
>> my_sdp_structure = loadjson(sdp_json);
```