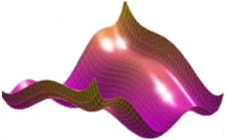
Polynomial Moment Optimization Database

Victor Magron & Michal Kocvara & Bernard Mourrain

General online Julia training, POEMA
16 April 2021



Part I - The Polynomial-Moment data

Victor Magron

Install the Julia package

In Julia, type:

```
] add https://github.com/PolynomialMomentOptimization/PMO.jl
```

Install the Julia package

In Julia, type:

```
] add https://github.com/PolynomialMomentOptimization/PMO.jl
```

To run the examples of this tutorial, you also need to install a package for multivariate polynomials:

```
] add DynamicPolynomials
```

Creating a github account and joining the project `data`

To create new data, you need a github account.

💡 Send a mail to `bernard.mourrain@inria.fr` with subject:
`Joining PMO/data <your.github.id>`

💡 We add you in the github project

`https://github.com/PolynomialMomentOptimization/data`

to push data in the database.

💡 You can configure your git access (in julia or in a terminal):

```
; git config --global user.name "Firstname Lastname"
; git config --global credential.helper store
```

What is in the database

The database contains the following types of data:

- Polynomial Optimization Programs (POP),
- Moment Optimization prograMs (MOM),
- Semi-Definite Programs (SDP)

These mathematical programs are described by

- an objective function (optional)
- sign constraints (optional)
- equality constraints (optional)

Polynomial data

$$\begin{aligned} \inf_{x\in\mathbb{R}^n} \quad & f(x) \\ s.t. \quad & g_1(x) \geq 0, \dots, g_s(x) \geq 0 \\ & h_1(x) = 0, \dots, h_t(x) = 0 \end{aligned}$$

where $f, g_j, h_i \in \mathbb{R}[x]$.

- f is the objective function,
- (g_j) are sign constraints,
- (h_i) are equality constraints.

Polynomial data

$$\begin{aligned} \inf_{x \in \mathbb{R}^n} \quad & f(x) \\ s.t. \quad & g_1(x) \geq 0, \ldots, g_s(x) \geq 0 \\ & h_1(x) = 0, \ldots, h_t(x) = 0 \end{aligned}$$

where $f, g_j, h_i \in \mathbb{R}[x]$.

- f is the objective function,
- (g_j) are sign constraints,
- (h_i) are equality constraints.

💡 Let's add this one:

$$\begin{aligned} \inf_{x,y} \quad & x^2y^2 + x^4 - y^3 \\ s.t. \quad & x^2 + \pi y^2 - 2 \leq 0 \\ & x \geq 0 \\ & 2y^2 - y = 0 \end{aligned}$$

Polynomial data: example

[1]:
```
using PMO, DynamicPolynomials

X = @polyvar x y
f  = x^2*y^2+x^4-y^3
g1 = x^2 + Float64(pi)*y^2 -2
g2 = x
h1 = 2*y^2-y

F  = PMO.polynomial((f,"inf"),
                    (g1,"<=0"),
                    (g2,">=0"),
                    (h1,"=0"))
F["doc"]=
"""
A first polynomial example.
"""
```

Polynomial data: example

Alternative for the same data F:

[2]:
```
F  = PMO.data((f,"inf"),
              (g1,"<=0"),
              (g2,">=0"),
              (h1,"=0") ; type = "polynomial")
```

Polynomial data: example

Alternative for the same data F:

[2]:
```
F  = PMO.data((f,"inf"),
              (g1,"<=0"),
              (g2,">=0"),
              (h1,"=0") ; type = "polynomial")
```

[2]:
```
Optimisation model:
  type => polynomial
  variables => ["x", "y"]
  nvar => 2
  constraints => [ x^2 + 3.141592653589793*y^2 - 2.0␣
 ↪<=0, x >=0, 2*y^2 - y =0 ]
  objective => inf x^4 + x^2*y^2 - y^3
  version => 0.0.1
  uuid => 89de08fc-9155-11eb-1bb2-3dc70c092fe5
```

Polynomial data: attributes

The data contains the following information:

- the type `P[:type]` specifying if it is a polynomial, moment or SDP program
- the variables `P[:variables]` as an array of strings
- the number of variables `P[:nvar]`
- the constraints `P[:constraints]` (optional)
- the objective function `P[:objective]` (optional)
- the version of the data `P[:version]`
- a universally unique identifier `P[:uuid]`

Attributes can be modified, e.g. `P[:version]= "0.0.2"`

New attributes can be added to the data, e.g. `P[:ref] = ["doi1", "doi2"]`.

Attributes can be removed, e.g. `P[:version]= nothing` (but this is not recommended)

Polynomial data: write & read

Print/save object of type `PMOData` in json format:

[3]:
```
PMO.write(F)
PMO.write("tmp.json",F)
```

Polynomial data: write & read

Print/save object of type `PMOData` in json format:

[3]:
```
PMO.write(F)
PMO.write("tmp.json",F)
```

Read data ⇒ `G` of type `PMOData`, read from the `json` file, where `F` was saved (it writes in `json` as `F`):

[4]:
```
G  = PMO.read("tmp.json")
PMO.write(G)
```

Moment data

$$\begin{array}{ll} \inf_{\mu_j} & \langle \mu_1 | f_1 \rangle + \cdots + \langle \mu_\nu | f_\nu \rangle \\ s.t. & \sum_{j=1}^{\nu} g_{1,j} \star \mu_j \succeq 0, \ldots, \\ & \sum_{j=1}^{\nu} h_{1,j} \star \mu_j = 0, \ldots, \\ & \sum_{j=1}^{\nu} \langle \mu_j | l_{1,j} \rangle + l_{1,\nu+1} \geq 0, \ldots, \\ & \sum_{j=1}^{\nu} \langle \mu_j | m_{1,j} \rangle + m_{1,\nu+1} = 0, \ldots, \end{array}$$

where μ_j are moment sequences, i.e. elements of the dual $\mathbb{R}[x]^*$, $f_j, g_{i,j}, h_{i,j}, l_{i,j}, m_{i,j} \in \mathbb{R}[x]$, $l_{i,\nu+1}, m_{i,\nu+1} \in \mathbb{R}$.

The constraints $\sum_{j=1}^{\nu} \langle \mu_i | l_{i,j} \rangle \geq 0$, $\sum_{j=1}^{\nu} \langle \mu_j | m_{i,j} \rangle = 0$ are respectively mass sign constraints and mass equality constraints.

The product $\star$ is defined as follows: for $p \in \mathbb{R}[x], \mu \in \mathbb{R}[x]^*$, $p \star \mu : q \mapsto \langle p \star \mu | q \rangle := \langle \mu | pq \rangle$.

Moment data: example

Let's add this one:

$$\begin{aligned} \inf_{\mu_1,\mu_2} \quad & \langle \mu_1 | x^2y^2 + x^4 - y^3 \rangle + \langle \mu_2 | xy \rangle \\ s.t. \quad & (2 - x^2 - \pi y^2) \star \mu_1 \succeq 0 \\ & \langle \mu_2 | x \rangle - 1 \geq 0 \\ & \langle \mu_1 | 2y^2 - y \rangle + \langle \mu_2 | x^2 + 2.1xy^2 \rangle - 2 = 0 \end{aligned}$$

Moment data: example

[5]:
```julia
using DynamicPolynomials, PMO
@polyvar x y
f1 = x^2*y^2+x^4-y^3
f2 = x*y

g1 = 2 -x^2 - Float64(pi)*y^2
g2 = x

h1 = 2*y^2-y
h2 = x^2+y*2.1*x*y

F  = PMO.moment(([f1,f2],"inf"),
                ([g1,0],">=0"),
                ([0,g2,-1], ">=0 *"),
                ([h1, h2], "=0 *")
                )
```

SDP data

$$
\begin{array}{rl}
\inf_{x\in\mathbb{R}^n} & f^T x \\
s.t. & \sum_{j=1}^n A_{1,j} x_j + A_{1,0} \succeq 0, \ldots, \\
 & \sum_{j=1}^n c_{1,1}^T x + d_{1,1} \geq 0, \ldots, \\
 & \sum_{j=1}^n c_{0,1}^T x + d_{0,1} = 0, \ldots,
\end{array}
$$

where $f, c_{i,j} \in \mathbb{R}^n$, $d_{i,j} \in \mathbb{R}$, $A_{i,j} \in S^{n_i}$ are symmetric matrices of size n_i.

💡 Let's add this one:

$$
\begin{array}{rl}
\inf_{x_1,x_2,x_3} & x_1 + 2x_2 + 3x_3 \\
s.t. & \begin{pmatrix} 2 & -1 & 0 \\ -1 & 2 & 0 \\ 0 & 0 & 2 \end{pmatrix} x_1 + \begin{pmatrix} 2 & 0 & -1 \\ 0 & 2 & 0 \\ -1 & 0 & 2 \end{pmatrix} x_3 \succeq 0, \\
 & \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix} x_1 + \begin{pmatrix} 0 & 3 \\ 3 & 0 \end{pmatrix} x_2 + \begin{pmatrix} 0 & -1 \\ -1 & 2 \end{pmatrix} \succeq 0, \\
 & 1.1x_1 + 2x_2 - 4 = 0, \quad -1.2x_2 + 3x_3 - 1 \leq 0
\end{array}
$$

SDP data: example

[6]:
```
using PMO, LinearAlgebra

LMI1 = [Symmetric([2 -1 0; 0 2 0; 0 0 2]),
        0,
        Symmetric([2 0 -1; 0 2 0; 0 0 2])
       ]
LMI2 = [Symmetric([1 0; 0 -1]),
        Symmetric([0 3; 3 0 ]),
        0,
        Symmetric([0 -1; -1 2])
       ]
F  = PMO.sdp(([1,2,3], "inf"),
             (LMI1,">=0"),
             (LMI2,">=0"),
             ([1.1,2,0,-4], "=0"),
             ([0,-1.2,3,-1],"<=0"))
```

SDP data: example with rank 1 matrix

Symmetric matrices of low rank can be used in the Linear Matrix Inequality constraints:

$$
\begin{aligned}
\inf_{x_1,x_2,x_3} \quad & x_3 \\
s.t. \quad & \begin{pmatrix} 2 & 1 & 0 \\ 1 & 1 & 0 \\ 0 & 0 & 0 \end{pmatrix} x_1 + \begin{pmatrix} 0 & 0 & 0 \\ 0 & 1 & 1 \\ 0 & 1 & 1 \end{pmatrix} x_2 \\
& + \begin{pmatrix} 0 & 0 & 1 \end{pmatrix} \otimes \begin{pmatrix} 0 & 0 & 1 \end{pmatrix} x_3 \\
& - \begin{pmatrix} 0 & 0 & 0 \\ 0 & 0 & 1 \\ 0 & 1 & 0 \end{pmatrix} \succeq 0
\end{aligned}
$$

[7]:
```
LMI1 = [Symmetric([2 1 0; 0 1 0; 0 0 0]),
        Symmetric([0 0 0; 0 1 1; 0 0 1]),
        [[0,0,1]],
        -Symmetric([0 0 0; 0 0 1; 0 0 0])]

F  = PMO.sdp(([0,0,1], "inf"), (LMI1, ">=0"))
```

Exercise

☞ Create a PMO data with your favorite example.