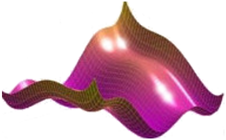
Polynomial Moment Optimization Database

Victor Magron & Michal Kocvara & Bernard Mourrain

General online Julia training, POEMA
16 April 2021



Part III - Using the database

Bernard Mourrain

Using the database

The PMO data base is loaded as follows:

```
[1]: using PMO
     t = PMO.table()
```

It is a table of triplets (`uuid` , `name`, `url`)

- `uuid`: a universally unique identifier,
- `name`: a string containing tags to recover easily the data,
- `url`: uniform resource location of the data file.

This table only contains references to the PMO data. These data files are available in the local folder `$HOME/.julia/PMO/data`.

Using the database

Data can be selected from their `:name` attribute, by regular expressions or matching strings.

[2]:
```
t2 = select(t,r"[Mm]otz")
t3 = select(t,"Motzkin")
```

[2]:
```
PMO.DataBase(Table with 3 rows, 3 columns:
Columns:
      #  colname  type
      1  uuid     String
      2  name     String
      3  url      String)
```

To select one column of a table:

[3]:
```
select(t2, :name)
```

Using the database

How to recover data in a table:

The ith entry:

[4]:
```
t2[1]
```

Matching strings or regular expressions:

[5]:
```
t2[r"Motz."]
m = t3["Motz"]
```

Exercise

1) Find the data which name matches `"Motz.*bounded"`
2) Get the `:objective` functions
3) Get the vector of `:constraints` polynomial(s)

Exercise

1) Find the data which name matches `"Motz.*bounded"`
2) Get the `:objective` functions
3) Get the vector of `:constraints` polynomial(s)

Solution:

[6]:
```
using PMO
t = PMO.table()
P = t[r"Motz.*bounded"][1]
f = P[:objective][1]
g = [ p[1] for p in P[:constraints] ]
```

Creating a new data and pushing it

We define the data and set its the `:name` and `:author` attributes:

[7]:
```
using PMO, DynamicPolynomials
X = @polyvar x y
motz = x^4*y^2 + x^2*y^4 + 1 - 3*x^2*y^2

Motz = PMO.data((motz,"inf"), (2-x^2-y^2, ">=0"))
Motz[:name] = "Motzkin bounded"
Motz[:author] = "Joe Test"
```

To push the data to the database, in the file `motzkin_bounded` :

[8]:
```
push(t, Motz, file="motzkin_bounded")
```

To remove it from the database:

[9]:
```
PMO.rm(t, Motz)
```

Modifying a data and pushing it

[10]:
```
using PMO
t = PMO.table()
F = t[2]
F[:version] = "0.0.2"
push(t,F)
```

Notice that the new version of the data will be committed in the database.

Modifying an existing data should be done with care.

Adding your own (family of) data

- Construct the polynomial constraints and objective functions
- Construct the PMO data
- Specify the name so that it can be recovered easily afterwards
- push the data in the database

Exercise

1) Write a function which tests if a data has a polynomial type and has more than 3 two variables (using `getdata(x[:url])`);
2) Select all data which satisfy this test (using `select`);
3) Write the table in a file `mytable.csv`;
4) Define a new table from this file.

Exercise

1) Write a function which tests if a data has a polynomial type and has more than 3 two variables (using `getdata(x[:url])`);
2) Select all data which satisfy this test (using `select`);
3) Write the table in a file `mytable.csv`;
4) Define a new table from this file.

[11]:
```
using PMO; t = PMO.table()
function hasprop(x)
    P = getdata(x[:url])
    return (P[:type]=="polynomial" && P[:nvar] >= 3)
end
tp = select(t, hasprop)
write("mytable.csv", tp)
mt = PMO.table("mytable.csv")
```

Solving optimization problems

The PMO data can be transformed with `vec` into an array of pairs of polynomials and constraints that can be easily used in other functions:

[12]:
```
P = t[r"Motz.*bound"][1];
vec(P)
```

[12]:
```
2-element Array{Any,1}:
 (xy² + x²y - 3x²y² + 1, "inf")
 (-x² - y² + 2, ">=0")
```

Solving optimization problems: example with MomentTools & CSDP

```
] add CSDP
] add https://gitlab.inria.fr/
↪AlgebraicGeometricModeling/MomentTools.jl
```

[13]:
```
using MomentTools, CSDP

optimizer = CSDP.Optimizer
v, M = optimize(vec(P),variables(P), 3, optimizer) ␣
↪    #Moment relaxation at order 3
get_minimizers(M)
```

Solving optimization problems: example with MomentTools & CSDP

```
] add CSDP
] add https://gitlab.inria.fr/
↪AlgebraicGeometricModeling/MomentTools.jl
```

[13]:
```
using MomentTools, CSDP

optimizer = CSDP.Optimizer
v, M = optimize(vec(P),variables(P), 3, optimizer) ␣
↪   #Moment relaxation at order 3
get_minimizers(M)
```

...

[13]:
```
2×4 Array{Float64,2}:
 -0.999949  0.999949  -0.999949   0.999949
  0.999949  0.999949  -0.999949  -0.999949
```

Solving optimization problems: example with `TSSOS` & `MosekTools`

```
] add MosekTools
] add https://github.com/wangjie212/TSSOS
```

[14]:
```
using TSSOS

tssos_first(first.(vec(P)), variables(P), 3)
```

Solving optimization problems: example with TSSOS & MosekTools

```
] add MosekTools
] add https://github.com/wangjie212/TSSOS
```

[14]:
```julia
using TSSOS

tssos_first(first.(vec(P)), variables(P), 3)
```

[14]:
```
...
optimum = 5.302965058909364e-8
...
```