GloptiPoly 3: moments, optimization and semidefinite programming

Didier Henrion[1,2], Jean-Bernard Lasserre[1,3], Johan Löfberg[4]

Version 3.3 of September 23, 2008

1 What is GloptiPoly ?

Gloptipoly 3 is intended to solve, or at least approximate, the Generalized Problem of Moments (GPM), an infinite-dimensional optimization problem which can be viewed as an extension of the classical problem of moments [10]. The GPM can be formulated as follows:

$$
\begin{aligned}
\min_{\mu} \text{ (or max)} \quad & \sum_k \int_{\mathbb{K}_k} g_{0k}(x) d\mu_k(x) \\
\text{s.t.} \quad & \sum_k \int_{\mathbb{K}_k} h_{jk}(x) d\mu_k(x) \geq \text{ (or =) } b_j, \quad j = 0, 1, \ldots
\end{aligned}
\tag{1}
$$

where measures μ_k are supported on basic semialgebraic sets

$$
\mathbb{K}_k = \{x \in \mathbb{R}^{n_k} : g_{ik}(x) \geq 0, \quad i = 1, 2 \ldots\}.
$$

In the above notations, $g_{ik}(x)$, $h_{jk}(x)$ are given real polynomials and b_j are given real constants. From a theoretical viewpoint, the GPM has developments and impact in various areas of mathematics such as algebra, Fourier analysis, functional analysis, operator theory, probability and statistics, to cite a few. In addition, and despite a rather simple and short formulation, the GPM has a large number of important applications in various fields such as optimization, probability, finance, control, signal processing, chemistry, cristallography, tomography, etc. For an account of various methodologies as well as some of potential applications, the interested reader is referred to [1, 2] and the nice collection of papers [6].

[1]LAAS-CNRS, University of Toulouse, France

[2]Faculty of Electrical Engineering, Czech Technical University in Prague, Czech Republic

[3]Institute of Mathematics, University of Toulouse, France

[4]Department of Electrical Engineering, Linköping University, Sweden

The approach is similar to that used in the former version 2 of GloptiPoly [3]. The software allows to build up a hierarchy of semidefinite programming (SDP), or linear matrix inequality (LMI) relaxations of the GPM, whose associated monotone sequence of optimal values converges to the global optimum. For more details on the approach, the interested reader is referred to [10].

2   Installation

GloptiPoly 3 is a freeware subject to the General Public Licence (GPL) policy. It is available for Matlab 7.2 (Release 2006) and later versions. It can be downloaded at

$$\texttt{www.laas.fr/}{\sim}\texttt{henrion/software/gloptipoly3}$$

The package, available as a compressed archive, consists of several m-files and subdirectories, and it contains no binaries. Extracted files are placed in a `gloptipoly3` directory that should be declared in the Matlab working path, using e.g. Matlab's command

```
>> addpath gloptipoly3
```

GloptiPoly 3 uses by default the semidefinite programming solver SeDuMi [13], so this package should be properly installed. Other semidefinite solvers can also be used provided they are installed and interfaced through YALMIP [12].

3   Getting started

Please type the command

```
>> gloptipolydemo
```

to run interactively the basic example that follows.

Consider the classical problem of minimizing globally the two-dimensional six-hump camel back function [3]

$$\min_{x \in \mathbb{R}^2} g_0(x) = 4x_1^2 + x_1x_2 - 4x_2^2 - 2.1x_1^4 + 4x_2^4 + \frac{1}{3}x_1^6.$$

The function has six local minima, two of them being global minima.

Using GloptiPoly 3, this optimization problem can be modeled as a moment problem as follows:

```
>> mpol x1 x2
>> g0 = 4*x1^2+x1*x2-4*x2^2-2.1*x1^4+4*x2^4+x1^6/3
Scalar polynomial
4x1^2+x1x2-4x2^2-2.1x1^4+4x2^4+0.33333x1^6
>> P = msdp(min(g0));
GloptiPoly 3.0
Define moment SDP problem
...
(GloptiPoly output suppressed)
...
Generate moment SDP problem
>> P
Moment SDP problem
  Measure label            = 1
  Relaxation order         = 3
  Decision variables       = 27
  Semidefinite inequalities = 10x10
```

Once the moment problem is modeled, a semidefinite solver can be used to solve it numerically. Here we use SeDuMi [13] which is assumed to be installed and accessible from the Matlab working path:

```
>> [status,obj] = msol(P)
GloptiPoly 3.0
```

3

```
Solve moment SDP problem

****************************************************

Calling SeDuMi

SeDuMi 1.1R3 by AdvOL, 2006 and Jos F. Sturm, 1998-2003.

...

(SeDuMi output suppressed)

...

2 globally optimal solutions extracted

>> status

status =

     1

>> obj

obj =

    -1.0316

>> x = double([x1 x2]);

x(:,:,1) =

    0.0898   -0.7127

x(:,:,2) =

   -0.0898    0.7127
```

The flag `status = 1` means that the moment problem is solved successfully and that Glop-
tiPoly can extract two globally optimal solutions reaching the objective function `obj = -1.0316`.

4 From version 2 to version 3

The major changes incorporated into GloptiPoly when passing from version 2 to 3 can be sum-
marized as follows:

- Use of native polynomial objects and object-oriented programming with specific classes
  for multivariate polynomials, measures, moments, and corresponding overloaded opera-
  tors. In contrast with version 2, the Symbolic Toolbox for Matlab (gateway to the Maple

4

kernel) is not required anymore to process polynomial data.

- Generalized problems of moments featuring several measures with semialgebraic support constraints and linear moment constraints can be processed and solved. Version 2 was limited to moment problems on a unique measure without moment constraints.

- Explicit moment substitutions are carried out to reduce the number of variables and constraints.

- The moment problems can be solved numerically with any semidefinite solver, provided it is interfaced through YALMIP. In contrast, version 2 used only the solver SeDuMi.

5 Solving generalized problems of moments

GloptiPoly 3 uses advanced Matlab features for object-oriented programming and overloaded operators. The user should be familiar with the following basic objects.

5.1 Multivariate polynomials (`mpol`)

A multivariate polynomial is an affine combination of monomials, each monomial depending on a set of variables. Variables can be declared in the Matlab working space as follows:

```
>> clear
>> mpol x
>> x
Scalar polynomial
x
>> mpol y 2
>> y
2-by-1 polynomial vector
(1,1):y(1)
(2,1):y(2)
```

```
>> mpol z 3 2
>> z
3-by-2 polynomial matrix
(1,1):z(1,1)
(2,1):z(2,1)
(3,1):z(3,1)
(1,2):z(1,2)
(2,2):z(2,2)
(3,2):z(3,2)
```

Variables, monomials and polynomials are defined as objects of class `mpol`.

All standard Matlab operators have been overloaded for `mpol` objects:

```
>> y*y'-z'*z+x^3
2-by-2 polynomial matrix
(1,1):y(1)^2-z(1,1)^2-z(2,1)^2-z(3,1)^2+x^3
(2,1):y(1)y(2)-z(1,1)z(1,2)-z(2,1)z(2,2)-z(3,1)z(3,2)+x^3
(1,2):y(1)y(2)-z(1,1)z(1,2)-z(2,1)z(2,2)-z(3,1)z(3,2)+x^3
(2,2):y(2)^2-z(1,2)^2-z(2,2)^2-z(3,2)^2+x^3
```

Use the instruction

```
>> mset clear
```

to delete all existing GloptiPoly variables from the Matlab working space.

5.2 Measures (`meas`)

Variables can be associated with real-valued measures, and one variable is associated with only one measure. For GloptiPoly, measures are identified with a label, a positive integer. When starting a GloptiPoly session, the default measure has label 1. By default, all created variables

are associated with the current measure. Measures can be handled with the class `meas` as follows:

```
>> mset clear
>> mpol x
>> mpol y 2
>> meas
Measure 1 on 3 variables: x,y(1),y(2)
>> meas(y) % create new measure
Measure 2 on 2 variables: y(1),y(2)
>> m = meas
1-by-2 vector of measures
1:Measure 1 on 1 variable: x
2:Measure 2 on 2 variables: y(1),y(2)
>> m(1)
Measure number 1 on 1 variable: x
```

The above script creates a measure $\mu_1(dx)$ on $\mathbb{R}$ and a measure $\mu_2(dy)$ on $\mathbb{R}^2$.

Use the instruction

```
>> mset clearmeas
```

to delete all existing GloptiPoly measures from the working space. Note that this does not delete existing GloptiPoly variables.

5.3 Moments (mom)

Linear combinations of moments of a given measure can be manipulated with the `mom` class as follows:

```
>> mom(1+2*x+3*x^2)
```

```
Scalar moment

I[1+2x+3x^2]d[1]

>> mom(y*y')

2-by-2 moment matrix

(1,1):I[y(1)^2]d[2]

(2,1):I[y(1)y(2)]d[2]

(1,2):I[y(1)y(2)]d[2]

(2,2):I[y(2)^2]d[2]
```

The notation I[p]d[k] stands for $\int p\,d\mu_k$ where p is a polynomial of the variables associated with measure μ_k, and k is the measure label.

Note that it makes no sense to define moments over several measures, or nonlinear moment expressions:

```
>> mom(x*y(1))

??? Error using ==> mom.mom

Invalid partitioning of measures in moments

>> mom(x)*mom(y(1))

??? Error using ==> mom.times

Invalid moment product
```

Note also the distinction between a constant term and the mass of a measure:

```
>> 1+mom(x)

Scalar moment

1+I[x]d[1]

>> mom(1+x)

Scalar moment

I[1+x]d[1]

>> mass(x)

Scalar moment

I[1]d[1]
```

8

Finally, let us mention three equivalent notations to refer to the mass of a measure:

```
>> mass(meas(y))
Scalar moment
I[1]d[2]
>> mass(y)
Scalar moment
I[1]d[2]
>> mass(2)
Scalar moment
I[1]d[2]
```

The first command refers explicitly to the measure, the second command is a handy short-cut to refer to a measure via its variables, and the third command refers to GloptiPoly's labeling of measures.

5.4   Support constraints (`supcon`)

By default, a measure on n variables is defined on the whole $\mathbb{R}^n$. We can restrict the support of a mesure to a given semialgebraic set as follows:

```
>> 2*x^2+x^3 == 2+x
Scalar measure support equality
2x^2+x^3 == 2+x
>> disk = (y'*y <= 1)
Scalar measure support inequality
y(1)^2+y(2)^2 <= 1
```

Support constraints are modeled by objects of class `supcon`. The first command means that variable x must satisfy $x^3 + 2x^2 - x - 2 = (x-1)(x+1)(x+2) = 0$, i.e. measure μ_1 must be discrete, a linear combination of three Dirac at 1, -1 and -2. The second command restricts measure μ_2 within the unit disk.

Note that it makes no sense to define a support constraint on several measures:

```
>> x+y(1) <= 1
??? Error using ==> supcon.supcon
Invalid reference to several measures
```

5.5 Moment constraints (momcon)

We can constrain linearly the moments of several measures:

```
>> mom(x^2+2) == 1+mom(y(1)^3*y(2))
Scalar moment equality constraint
I[2+x^2]d[1] == 1+I[y(1)^3y(2)]d[2]
>> mass(x)+mass(y) <= 2
Scalar moment inequality constraint
I[1]d[1]+I[1]d[2] <= 2
```

Moment constraints are modeled by objects of class momcon.

For GloptiPoly an objective function to be minimized or maximized is considered as a particular moment constraint:

```
>> min(mom(x^2+2))
Scalar moment objective function
min I[2+x^2]d[1]
>> max(x^2+2)
Scalar moment objective function
max I[2+x^2]d[1]
```

The latter syntax is a handy short-cut which directly converts an mpol object into an momcon object.

5.6 Floating point numbers (`double`)

Variables in a measure can be assigned numerical values:

```
>> m1 = assign(x,2)
Measure 1 on 1 variable: x
supported on 1 point
```

which is equivalent to enforcing a discrete support for the measure. Here μ_1 is set to the Dirac at the point 2.

The `double` operator converts a measure or its variables into a floating point number:

```
>> double(x)
ans =
     2
>> double(m1)
ans =
     2
```

Polynomials can be evaluated similarly:

```
>>double(1-2*x+3*x^2)
ans =
     9
```

Discrete measure supports consisting of several points can be specified in an array:

```
>> m2 = assign(y,[-1 2 0;1/3 1/4 -2])
Measure 2 on 2 variables: y(1),y(2)
supported on 3 points
>> double(m2)
ans(:,:,1) =
```

```
    -1.0000

     0.3333

ans(:,:,2) =

     2.0000

     0.2500

ans(:,:,3) =

     0

    -2
```

5.7   Moment SDP problems (`msdp`)

GloptiPoly 3 can manipulate and solve Generalized Problems of Moments (GPM) as defined in (1). The decision variables in the GPM are measures $\mu_k(x)$, and GloptiPoly 3 allows to optimize over them through their moments

$$y_{\alpha_k} = \int_{\mathbb{K}_k} x^{\alpha_k} d\mu_k(x), \quad \alpha_k \in \mathbb{N}^{n_k}$$

where the α_k are multi-indices.

5.8   Solving moment problems `msol`

Once a moment problem is defined, it can be solved numerically with the instruction `msol`. In the sequel we give several examples of GPMs handled with GloptiPoly 3.

5.8.1   Unconstrained minimization

Following [7], given a multivariate polynomial $g_0(x)$, the unconstrained optimization problem

$$\min_{x \in \mathbb{R}^n} g_0(x)$$

can be formulated as a linear moment optimization problem

$$\min_\mu \quad \int g_0(x) d\mu(x)$$
$$\text{s.t.} \quad \int d\mu(x) = 1$$

12

where measure μ lives in the space $\mathbb{B}^n$ of finite Borel signed measures on $\mathbb{R}^n$. The equality constraint indicates that the mass of μ is equal to one, or equivalently, that μ is a probability measure.

In general, this linear (hence convex) reformulation of a (typically nonconvex) polynomial problem is not helpful because there is no computationally efficient way to represent measures and their underlying Borel spaces. The approach proposed in [7] consists in using convex semidefinite representations of the space $\mathbb{B}^n$ truncated to finite degree moments. GloptiPoly 3 allows to input such moment optimization problems in an user-friendly way, and to solve them using existing software for semidefinite programming (SDP).

In Section 3 we already encountered an example of an unconstrained polynomial optimization solved with GloptiPoly 3. Let us revisit this example:

```
>> mset clear
>> mpol x1 x2
>> g0 = 4*x1^2+x1*x2-4*x2^2-2.1*x1^4+4*x2^4+x1^6/3
Scalar polynomial
4x1^2+x1x2-4x2^2-2.1x1^4+4x2^4+0.33333x1^6
>> P = msdp(min(g0));
...
>> msol(P)
...
2 globally optimal solutions extracted
Global optimality certified numerically
```

This indicates that the global minimum is attained with a discrete measure supported on two points. The measure can be constructed from the knowledge of its first moments of degree up to 6:

```
>> meas
Measure 1 on 2 variables: x1,x2
  with moments of degree up to 6, supported on 2 points
```

```
>> double(meas)
ans(:,:,1) =

     0.0898

    -0.7127
ans(:,:,2) =

    -0.0898

     0.7127
>> double(g0)
ans(:,:,1) =

    -1.0316
ans(:,:,2) =

    -1.0316
```

When converting to floating point numbers with the operator double, it is essential to make the distinction between mpol and mom objects:

```
>> v = mmon([x1 x2],2)'
1-by-6 polynomial vector
(1,1):1
(1,2):x1
(1,3):x2
(1,4):x1^2
(1,5):x1x2
(1,6):x2^2
>> double(v)
ans(:,:,1) =
    1.0000    0.0898   -0.7127    0.0081   -0.0640    0.5079
ans(:,:,2) =
    1.0000   -0.0898    0.7127    0.0081   -0.0640    0.5079
>> double(mom(v))
ans =
    1.0000    0.0000   -0.0000    0.0081   -0.0640    0.5079
```

14

The first instruction `mmon` generates a vector of monomials `v` of class `mpol`, so the command `double(v)` calls the convertor `@mpol/double` which evaluates a polynomial expression on the discrete support of a measure (here two points). The last command `double(mom(v))` calls the convertor `@mom/double` which returns the value of the moments obtained after solving the moment problem.

Note that when inputing moment problems on a unique measure whose mass is not constrained, GloptiPoly assumes by default that the measure has mass one, i.e. that we are seeking a probability measure. Therefore, if `g0` is the polynomial defined previously, the two instructions

```
>> P = msdp(min(g0));
```

and

```
>> P = msdp(min(g0), mass(meas(g0))==1);
```

are equivalent. See also Section 5.3 for handling masses of measures and Section 5.8.2 for more information on mass constraints.

5.8.2 Constrained minimization

Consider now the constrained polynomial optimization problem

$$\min_{x \in \mathbb{K}} g_0(x)$$

where

$$\mathbb{K} = \{x \in \mathbb{R}^n : g_i(x) \geq 0, \, i = 1, 2, \ldots\}$$

is a basic semialgebraic set described by given polynomials $g_i(x)$. Following [7], this (nonconvex polynomial) problem can be formulated as the (convex linear) moment problem

$$\min_{\mu} \quad \int_{\mathbb{K}} g_0(x) d\mu(x)$$
$$\text{s.t.} \quad \int_{\mathbb{K}} d\mu(x) = 1$$

where the indeterminate is a probability measure μ of $\mathbb{B}^n$ which is now supported on set $\mathbb{K}$. In other words

$$\int_{\mathbb{R}^n / \mathbb{K}} d\mu(x) = 0.$$

As an example, consider the non-convex quadratic problem of Section 4.4 in [3]:

$$\min \quad -2x_1 + x_2 - x_3$$
$$\text{s.t.} \quad 24 - 20x_1 + 9x_2 - 13x_3 + 4x_1^2 - 4x_1x_2 + 4x_1x_3 + 2x_2^2 - 2x_2x_3 + 2x_3^2 \geq 0$$
$$x_1 + x_2 + x_3 \leq 4, \quad 3x_2 + x_3 \leq 6$$
$$0 \leq x_1 \leq 2, \quad 0 \leq x_2, \quad 0 \leq x_3 \leq 3$$

Each constraint in this problem is interpreted by GloptiPoly 3 as a support constraint on the measure associated with variable x, see Section 5.4:

```
>> mpol x 3
>> x(1)+x(2)+x(3) <= 4
Scalar measure support inequality
x(1)+x(2)+x(3) <= 4
```

The whole problem can be entered as follows:

```
>> mpol x 3
>> g0 = -2*x(1)+x(2)-x(3);
>> K = [24-20*x(1)+9*x(2)-13*x(3)+4*x(1)^2-4*x(1)*x(2) ...
 +4*x(1)*x(3)+2*x(2)^2-2*x(2)*x(3)+2*x(3)^2 >= 0, ...
 x(1)+x(2)+x(3) <= 4, 3*x(2)+x(3) <= 6, ...
 0 <= x(1), x(1) <= 2, 0 <= x(2), 0 <= x(3), x(3) <= 3];
>> P = msdp(min(g0), K)
...
Moment SDP problem
  Measure label            = 1
  Relaxation order         = 1
  Decision variables       = 9
  Linear inequalities      = 8
  Semidefinite inequalities = 4x4
```

The moment problem can then be solved:

```
>> [status,obj] = msol(P)

GloptiPoly 3.0

Solve moment SDP problem

...

Global optimality cannot be ensured

status =

     0

obj =

   -6.0000
```

Since `status=0` the moment SDP problem can be solved but it is impossible to detect global optimality. The value `obj=-6.0000` is then a lower bound on the global minimum of the quadratic problem.

The measure associated with the problem variables can be retrieved as follows:

```
>> mu = meas

Measure 1 on 3 variables: x(1),x(2),x(3)
  with moments of degree up to 2
```

Its vector of moments can be built as follows:

```
>> mv = mvec(mu)

10-by-1 moment vector

(1,1):I[1]d[1]

(2,1):I[x(1)]d[1]

(3,1):I[x(2)]d[1]

(4,1):I[x(3)]d[1]

(5,1):I[x(1)^2]d[1]

(6,1):I[x(1)x(2)]d[1]

(7,1):I[x(1)x(3)]d[1]

(8,1):I[x(2)^2]d[1]
```

```
(9,1):I[x(2)x(3)]d[1]
(10,1):I[x(3)^2]d[1]
```

These moments are the decision variables of the SDP problem solved with the above `msol` command. Their numerical values can be retrieved as follows:

```
>> double(mv)
ans =
     1.0000
     2.0000
    -0.0000
     2.0000
     7.6106
     1.4671
     2.3363
     4.8335
     0.5008
     8.7247
```

The numerical moment matrix can be obtained using the following commands:

```
>> double(mmat(mu))
ans =
     1.0000     2.0000    -0.0000     2.0000
     2.0000     7.6106     1.4671     2.3363
    -0.0000     1.4671     4.8335     0.5008
     2.0000     2.3363     0.5008     8.7247
```

As explained in [7], we can build a hierarchy of nested moment SDP problems, or relaxations, whose solutions converge monotically and asymptotically to the global optimum, under mild technical assumptions. By default the command `msdp` builds the relaxation of lowest order, equal to half the degree of the highest degree monomial in the polynomial data. An additional input argument can be specified to build higher order relaxations:

```
>> P = msdp(min(g0), K, 2)

...

Moment SDP problem
  Measure label            = 1
  Relaxation order         = 2
  Decision variables       = 34
  Semidefinite inequalities = 10x10+8x(4x4)
>> [status,obj] = msol(P)

...

Global optimality cannot be ensured
status =

     0
obj =

   -5.6922
>> P = msdp(min(g0), K, 3)

...

Moment SDP problem
  Measure label            = 1
  Relaxation order         = 3
  Decision variables       = 83
  Semidefinite inequalities = 20x20+8x(10x10)
>> [status,obj] = msol(P)

...

Global optimality cannot be ensured
status =

     0
obj =

   -4.0684
```

We observe that the moment SDP problems feature an increasing number of variables and con-
straints. They generate a mononotically increasing sequence of lower bounds on the global

optimum, which is eventually reached numerically at the fourth relaxation:

```
>> P = msdp(min(g0), K, 4)
...
Moment SDP problem
  Measure label            = 1
  Relaxation order         = 4
  Decision variables       = 164
  Semidefinite inequalities = 35x35+8x(20x20)
>> [status,obj] = msol(P)
...
2 globally optimal solutions extracted
Global optimality certified numerically
status =
     1
obj =
   -4.0000
>> double(x)
ans(:,:,1) =
     2.0000
     0.0000
     0.0000
ans(:,:,2) =
     0.5000
     0.0000
     3.0000
>> double(g0)
ans(:,:,1) =
   -4.0000
ans(:,:,2) =
   -4.0000
```

5.8.3 Rational minimization

Minimization of a rational function can also be formulated as a linear moment problem. Given two polynomials $g_0(x)$ and $h_0(x)$, consider the rational optimization problem

$$\min_{x\in\mathbb{K}}\frac{g_0(x)}{h_0(x)}$$

where

$$\mathbb{K}=\{x\in\mathbb{R}^n : g_i(x)\geq 0, i=1,2,\ldots\}$$

is a basic semialgebraic set described by given polynomials $g_i(x)$. Following [5], the corresponding moment problem is given by

$$\min_{\mu\in\mathbb{B}^n}\quad \int_{\mathbb{K}}g_0(x)d\mu(x)$$
$$\text{s.t.}\quad \int_{\mathbb{K}}h_0(x)d\mu(x)=1.$$

In contrast with the polynomial optimization problem of Section 5.8.2, the optimal measure μ supported on $\mathbb{K}$ is not necessarily a probability measure. Denoting $h_0(x)=\sum_\alpha h_{0\alpha}x^\alpha$, the moments y_α of μ must satisfy a linear constraint

$$\int_{\mathbb{K}}h_0(x)d\mu(x)=\sum_\alpha h_{0\alpha}y_\alpha=1.$$

As an example, consider the one-variable rational minimization problem [5, Ex. 2]:

$$\min\frac{x^2-x}{x^2+2x+1}.$$

We can solve this problem with GloptiPoly 3 as follows:

```
>> mpol x
>> g0 = x^2-2*x; h0 = x^2+2*x+1;
>> P = msdp(min(g0), mom(h0) == 1);
>> [status,obj] = msol(P)

...

Global optimality certified numerically
status =

     1
```

```
obj =

    -0.3333

>> double(x)

ans =

    0.4999
```

5.8.4   Several measures

GloptiPoly 3 can handle several measures whose moments are linearly related.

For example, consider the GPM arising when solving polynomial optimal control problems as detailed in [9]. We are seeking two occupation measures $\mu_1(dx, du)$ and $\mu_2(dx)$ of a state vector $x(t)$ and input vector $u(t)$ whose time variation are governed by the differential equation

$$\frac{dx(t)}{dt} = f(x, u), \, x(0) = x_0, \, u(0) = u_0$$

with $f(x, u)$ a given polynomial mapping and x_0, u_0 given initial conditions. Measure μ_1 is supported on a given semialgebraic set $\mathbb{K}_1$ corresponding to constraints on x and u. Measure μ_2 is supported on a given semialgebraic set $\mathbb{K}_2$ corresponding to performance requirements. For example $\mathbb{K}_2 = 0$ indicates that state x must reach the origin.

Given a polynomial test function $g(x)$ we can relax the dynamics constraint with the moment constraint

$$\int_{\mathbb{K}_2} g(x) d\mu_2(x) - g(x_0) = \int_{\mathbb{K}_1} \frac{dg(x)}{dx} f(x, u) d\mu_1(x, u)$$

linking linearly moments of μ_1 and μ_2. As explained in [9], a lower bound on the minimum time achievable by any feedback control law $u(x)$ is then obtained by minimizing the mass of μ_1 over all possible measures μ_1, μ_2 satisfying the support and moment constraints. The gap between the lower bound and the exact minimum time is narrowed by enlarging the class of test functions g.

In the following script we solve this moment problem in the case of a double integrator with state and input constraints:

```
% bounds on minimal achievable time for optimal control of
```

```matlab
% double integrator with state and input constraints


x0 = [1; 1]; u0 = 0; % initial conditions
d = 6; % maximum degree of test function


% analytic minimum time
if x0(1) >= -(x0(2)^2-2)/2
 tmin = 1+x0(1)+x0(2)+x0(2)^2/2;
elseif x0(1) >= -x0(2)^2/2*sign(x0(2))
 tmin = 2*sqrt(x0(1)+x0(2)^2/2)+x0(2);
else
 tmin = 2*sqrt(-x0(1)+x0(2)^2/2)-x0(2);
end


% occupation measure for constraints
mpol x1 2
mpol u1
m1 = meas([x1;u1]);


% occupation measure for performance
mpol x2 2
m2 = meas(x2);


% dynamics
scaling = tmin; % time scaling
f = scaling*[x1(2);u1];


% test function
g1 = mmon(x1,d);
g2 = mmon(x2,d);
```

```
% initial condition
assign([x1;u1],[x0;u0]);
g0 = double(g1);


% moment problem
P = msdp(min(mass(m1)),...
 u1^2 <= 1,... % input constraint
 x1(2) >= -1,... % state constraint
 x2'*x2 <= 0,... % performance = reach the origin
 mom(g2) - g0 == mom(diff(g1,x1)*f)); % linear moment constraints


% solve
[status,obj] = msol(P);
obj = scaling*obj;


disp(['Minimum time = ' num2str(tmin)]);
disp(['LMI ' int2str(d) ' lower bound = ' num2str(obj)])
```

For the initial condition $x_0 = [1\ 1]$ the exact minimum time is equal to 3.5. In Table 1 we report the monotically increasing sequence of lower bounds obtained by solving moment problems with test functions of increasing degrees. We used the above script and the semidefinite solver SeDuMi 1.1R3.

degree	2	4	6	8	10	12	14	16
bound	1.0019	2.3700	2.5640	2.9941	3.3635	3.4813	3.4964	3.4991

Table 1: Minimum time optimal control for double integrator with state and input constraints: lower bounds on exact minimal time 3.5 achieved by solving moment problems with test functions of increasing degrees.

5.9   Using YALMIP

By default GloptiPoly 3 uses the semidefinite solver SeDuMi [13] for solving numerically SDP moment problems. It is however possible to use any solver interfaced through YALMIP [12] by setting a configuration flag with the `mset` command:

```
>> mset('yalmip',true)
```

Parameters for YALMIP, handled with the YALMIP command `sdpsettings`, can be forwarded to GloptiPoly 3 with the `mset` command. For example, the following command tells YALMIP to use the SDPT3 solver (instead of SeDuMi) when solving moment problems with GloptiPoly:

```
>> mset(sdpsettings('solver','sdpt3'));
```

5.10   SeDuMi parameters settings

The default parameters settings of SeDuMi [13] can be altered as follows:

```
>> pars.eps = 1e-10;
>> mset(pars)
```

where `pars` is a structure of parameters consistent with SeDuMi's format.

5.11   Exporting moment SDP problems

A moment problem `P` of class `msdp` can be converted into SeDuMi's input format:

```
>> [A,b,c,K] = msedumi(P);
```

The SDP problem can then be solved with SeDuMi as follows:

```
>> [x,y,info] = sedumi(A,b,c,K);
```

See [13] for more information on SeDuMi's input data format.

Similarly, a moment SDP problem can be converted into YALMIP's input format:

```
>> [F,h,y] = myalmip(P);
```

where variable F contains the LMI constraints (YALMIP class lmi), h is the objective function (YALMIP class sdpvar) and y is the vector of moments (YALMIP class sdpvar). The SDP problem can then be solved with any semidefinite solver interfaced through YALMIP as follows:

```
>> solvesdp(F,h);
>> ysol = double(y);
```

5.12   Moment substitutions

By performing explicit moment substitutions it is often possible to reduce significantly the number of variables and constraints in moment SDP problems. Version 2 of GloptiPoly implemented these substitutions for mixed-integer 0-1 problems only [3]. With version 3, these substitutions can be carried out in full generality.

GloptiPoly 3 carries out moment substitutions as soon as the left hand-side of a support or moment equality constraint consists of an isolated monic monomial. Otherwise, no substitution is achieved and the equality constraint is preserved.

For example, consider the AW_2^9 Max-Cut problem studied in [3, §4.7], with variables x_i taking values -1 or $+1$ for $i = 1, \ldots, 9$. These integer constraints can be expressed algebraically as $x_i^2 = 1$. The following piece of code builds up the third relaxation of this problem:

```
>> W = diag(ones(8,1),1)+diag(ones(7,1),2)+diag([1 1],7)+diag(1,8);
>> W = W+W'; n = size(W,1); e = ones(1,n); Q = (diag(e*W)-W)/4;
>> mset clear
>> mpol('x', n)
>> P = msdp(max(x'*Q*x), x.^2 == 1, 3)
```

```
GloptiPoly 3.0

Define moment SDP problem

  Valid objective function

  Number of support constraints = 9 including 9 substitutions

  Number of moment constraints = 0

Measure #1

  Maximum degree = 2

  Number of variables = 9

  Number of moments = 5005

Order of SDP relaxation = 3

Mass of measure 1 set to one

Total number of monomials = 5005

Perform moment substitutions

Perform support substitutions

Number of monomials after substitution = 465

Generate moment and support constraints

Generate moment SDP problem


Moment SDP problem

  Measure label              = 1

  Relaxation order           = 3

  Decision variables         = 465

  Semidefinite inequalities = 130x130
```

We see that out of the 5005 moments (corresponding to all the monomials of 9 variables of degree up to 6), only 465 linearly independent moments appear in a reduced moment matrix of dimension 130.

With the following syntax, moment substitutions are not carried out:

```
>> P = msdp(max(x'*Q*x), x.^2-1 == 0, 3)

...
```

```
Mass of measure 1 set to one

Total number of monomials = 5005

Perform moment substitutions

Number of monomials after substitution = 5004

Generate moment and support constraints

Generate moment SDP problem


Moment SDP problem
  Measure label            = 1
  Relaxation order         = 3
  Decision variables       = 5004
  Linear equalities        = 6435
  Semidefinite inequalities = 220x220
```

Only the mass is substituted, and the remaining 5004 moments linked by 6435 linear equalities (many of which are redundant) now appear explicitly in a full-size moment matrix of dimension 220.

6 Performance

The computational burden in GloptiPoly 3 is twofold:

- problem modeling, operations on multivariate polynomials;

- solving the resulting semidefinite programming problem.

For simple problems, it is expected that the overhead cost of manipulating multivariate polynomials and generating the semidefinite programming problem is negligible when compared to the cost of solving the semidefinite programming problem. However, the user should know that GloptiPoly 3 routines for multivariate polynomials are not optimized for performance and numerical reliability. They are essentially aimed at providing a convenient user-friendly interface. If the user wants to manipulate efficiently large polynomials or polynomial matrices,

GloptiPoly 3 is not the right tool to use. For this purpose, computer algebra software such as Maple or Mathematica are certainly preferable. Similarly, GloptiPoly 3 does not exploit problem sparsity on its own, and other packages such as SparsePOP [14] may be more appropriate for optimizing over sparse polynomials. However, if structured sparsity is already available, then one may use it with GloptiPoly 3 by implementing the sparse relaxations defined in [8].

Once the generalized problem of moments and its semidefinite relaxation are modeled, the performance of GloptiPoly 3 strongly depends on the performance of the underlying semidefinite solver. See [3] for computational times using SeDuMi on a standard computer (with GloptiPoly 2, in 2003) on a comprehensive collection of benchmark examples of non-convex polynomial optimization problems. See also [4] for benchmark polynomial systems of equations.

Finally, the numerical behavior of solvers applied to semidefinite problems arising from generalized problems of moments is not well understood. Developing a sound numerical analysis in this context (allowing for example to evaluate or approximate the problem conditioning, and to rescale ill-conditioned problems by an appropriate choice of polynomial basis) remains a very challenging research direction.

Acknowledgment

References

[1] N. I. Akhiezer. *The classical moment problem*. Hafner, New York, 1965.

[2] N. I. Akhiezer, M. G. Krein. *Some questions in the theory of moments*. American Mathematical Society Translations Vol. 2, 1962.

[3] D. Henrion, J. B. Lasserre. GloptiPoly: global optimization over polynomials with Matlab and SeDuMi. *ACM Transactions on Mathematical Software*, 29(2):165–194, 2003.

[4] D. Henrion, J. B. Lasserre. Solving global optimization problems over polynomials with GloptiPoly 2.1. In C. Bliek, C. Jermann, A. Neumaier (Editors). *Global optimization and constraint satisfaction*. Lecture Notes on Computer Science, 2861:45-58, Springer, 2003.

[5] D. Jibetean, E. de Klerk. Global optimization of rational functions: a semidefinite programming approach. *Mathematical Programming*, 106:93–109, 2006.

[6] H. J. Landau. *Moments in mathematics*. In H. J. Landau (Editor), Proceedings of Symposium on Applied Mathematics, 37, American Mathematical Society, 1980.

[7] J. B. Lasserre. Global optimization with polynomials and the problem of moments. *SIAM Journal on Optimization*, 11(3):796–817, 2001.

[8] J. B. Lasserre. Convergent SDP relaxations in polynomial optimization with sparsity. *SIAM Journal on Optimization*, 17(3):822-843, 2006.

[9] J. B. Lasserre, D. Henrion, C. Prieur, E. Trélat. Nonlinear optimal control via occupation measures and LMI relaxations. *SIAM Journal on Control and Optimization*, 47(4):1643-1666, 2008.

[10] J. B. Lasserre. A semidefinite programming approach to the generalized problem of moments. *Mathematical Programming*, 112:65–92, 2008.

[11] M. Laurent. Sums of squares, moment matrices and optimization over polynomials. In M. Putinar and S. Sullivant (Editors), IMA volume on *Emerging Applications of Algebraic Geometry*, Vol. 149, 2008.

[12] J. Löfberg. YALMIP : a toolbox for modeling and optimization in Matlab. *Proceedings of the IEEE Symposium on Computer-Aided Control System Design (CACSD)*, Taipei, Taiwan, 2004.

[13] J. F. Sturm and the Advanced Optimization Laboratory at McMaster University, Canada. *SeDuMi version 1.1R3*, 2006.

[14] H. Waki, S. Kim, M. Kojima, M. Muramatsu. SparsePOP: a sparse semidefinite programming relaxation of polynomial optimization problems. *ACM Transactions on Mathematical Software*, 35(2), 2008.