Introducing GloptiPoly for linear programming on the cone of nonnegative measures

Didier HENRION

LAAS-CNRS, Univ. Toulouse, France
Czech Tech. Univ. Prague, Czech Rep.

MAP Konstanz

September 2012

LP for measures

Linear programming (LP) problem

$$\begin{array}{ll} \text{min} & \langle c, \mu \rangle \\ \text{s.t.} & A(\mu) = b \\ & \mu \geq 0 \end{array}$$

where $\mu = (\mu_i)$ is a vector of nonnegative Borel measures and

$$\langle c, \mu \rangle = \sum_i \langle c_i, \mu_i \rangle = \sum_i \int c_i d\mu_i$$

and

$$A(\mu) = b \Longleftrightarrow \langle a_j, \mu \rangle = \sum_i \int a_{ij} d\mu_i = b_j$$

From J. B. Lasserre's talk

Nonconvex polynomial optimization problem

$$\min_{x \in X} g_0(x)$$

on basic semialgebraic set

$$X = \{x \in \mathbb{R}^n \,:\, g_k(x) \geq 0,\ k = 1, 2, \ldots\}$$

formulated as a convex linear measure problem

$$\begin{array}{rl} \min & \langle g_0, \mu \rangle \\ \text{s.t.} & \langle 1, \mu \rangle = 1 \end{array}$$

where the unknown is a nonnegative measure μ on X

Generalized problem of moments

Several measures μ_i supported on semialgebraic sets X_i

All the data are polynomials, so we can replace measures by their moments (e.g. $\int_{X_i} c_i(x)d\mu_i = \int_{X_i} \sum_\alpha c_{i\alpha}x^\alpha d\mu_i = \sum_\alpha c_{i\alpha} \int_{X_i} x^\alpha d\mu_i$)

$$\begin{array}{ll} \min_\mu & \sum_i \int_{X_i} c_i d\mu_i \\ \text{s.t.} & \sum_i \int_{X_i} a_{ij} d\mu_i = b_j \\ & \text{measures } \mu_i \end{array} \qquad \begin{array}{ll} \min_y & \sum_i \sum_\alpha c_{i\alpha} y_{i\alpha} \\ \text{s.t.} & \sum_i \sum_\alpha a_{ij_\alpha} y_{i\alpha} = b_j \\ & \text{moments } y_i \end{array}$$

provided we can handle the representation condition

$$y_{i\alpha} = \int_{X_i} x^\alpha d\mu_i(x)$$

Moment LP as LMI

Using Putinar's representation conditions we obtain

$$\begin{array}{ll} \text{min}_y & c^T y \\ \text{s.t.} & Ay = b \\ & y_\alpha = \int_X x^\alpha d\mu \\ & X = \{x \,:\, g_k(x) \geq 0, \forall k\} \end{array}$$

infinite-dimensional

LP problem

$$\begin{array}{ll} \text{min}_y & c^T y \\ \text{s.t.} & Ay = b \\ & M_d(y) \succeq 0 \\ & M_d(g_k y) \succeq 0, \forall k \end{array}$$

finite-dim. LMI

relaxation of order d

producing (under some assumption) a converging
hierarchy of finite-dimensional LMI relaxations

Discretization, analogy with Fourier analysis

What is GloptiPoly ?

Matlab parser for generalized problems of moments:

1. Generates SDP relaxation of given order
in SeDuMi format (A, b, c, K)

$$\begin{array}{lll} \min_x & c^T x \\ \text{s.t.} & Ax = b \\ & x \in K \end{array} \qquad \begin{array}{lll} \max_y & b^T y \\ \text{s.t.} & z = c - A^T y \\ & z \in K \end{array}$$

2. Call an SDP solver:
- either SeDuMi (default)
- or any solver interfaced with YALMIP

3. (Try to) extract solutions from moment matrices
(numerical linear algebra over quotient ideal)

Matlab classes

Multivariate polynomials `mpol`

Linear combinations of monomials depending on variables declared in the Matlab working space

```
>> 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)
```

Matlab classes

All standard Matlab operators overloaded for class `mpol`

```
>> 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
```

Matlab classes

Measures `meas`

- variables associated with real-valued measures
- one variable associated with only one measure
- measures handled internally as labels

```
>> 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 $d\mu_1(x)$ on $\mathbb{R}$
and a measure $d\mu_2(y)$ on $\mathbb{R}^2$

Matlab classes

Moments `mom`

Linear combinations of moments of a measure

```
>> 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 $d\mu_k$, and k is the measure label

Matlab classes

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
```

Matlab classes

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]
```

```
>> 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]
```

Matlab classes

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
```

Matlab classes

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
```

Matlab classes

An objective function to be minimized or maximized is also of class `moncon`:

```
>> 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

Discrete measures

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 $d\mu_1$ is set to the Dirac at the point 2

Convertors

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
```

Convertors

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) =       ans(:,:,2) =       ans(:,:,3) =
   -1.0000               2.0000                0
    0.3333               0.2500               -2
```

Moment SDP problem

Moment SDP problem `msdp`

Declared by calling constructor `msdp`
with arguments of classes `supcon` and `momcon`
built from `mpol` and `mom` objects

The moment SDP problem can then be solved with function `msol`

Here are some typical examples..

Unconstrained minimization

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 moment LP problem

$$\begin{array}{ll} \min_\mu & \int g_0(x) d\mu(x) \\ \text{s.t.} & \int d\mu(x) = 1 \end{array}$$

Unconstrained minimization

Minimizing the two-dimensional six-hump camel back function (six local minima including two global minima)

```
>> 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));
GloptiPoly 3.6
Define moment SDP problem
...
(GloptiPoly output suppressed)
...
Generate moment SDP problem
```

Unconstrained minimization

```
>> P = msdp(min(g0))
Moment SDP problem
  Measure label               = 1
  Relaxation order            = 3
  Decision variables          = 27
  Semidefinite inequalities = 10x10
>> 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

Unconstrained minimization

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
```

Constrained minimization

Polynomial optimization problem

$$\min_{x \in X} g_0(x)$$

with

$$X = \{x \in \mathbb{R}^n \,:\, g_k(x) \geq 0,\ k = 1, 2, \ldots\}$$

basic semialgebraic

This (nonconvex polynomial) problem can be formulated as the (convex linear) moment problem

$$\begin{array}{ll} \min_\mu & \int_X g_0(x) d\mu(x) \\ \text{s.t.} & \int_X d\mu(x) = 1 \end{array}$$

where the indeterminate is a probability measure μ supported on set X

Constrained minimization

GloptiPoly input sequence

```
>> mpol x 3
>> g0 = -2*x(1)+x(2)-x(3);
>> X = [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), X)
...
Moment SDP problem
  Measure label              = 1
  Relaxation order           = 1
  Decision variables         = 9
  Linear inequalities        = 8
  Semidefinite inequalities = 4x4
```

Constrained minimization

```
>> [status,obj] = msol(P)
GloptiPoly 3.6
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

Constrained minimization

Higher order SDP relaxations with increasing number of variables and constraints

```
>> P = msdp(min(g0), X, 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
```

Constrained minimization

Third relaxation..

```
>> P = msdp(min(g0), X, 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
```

Constrained minimization

Mononotically increasing sequence of lower bounds
Global optimum reached numerically at relaxation 4:

```
>> P = msdp(min(g0), X, 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
```

homepages.laas.fr/henrion/software/gloptipoly