# Polynomial optimization using MOSEK and Julia

ISMP, Pittsburgh, July 12-17, 2015

Joachim Dahl, MOSEK ApS

collaborators: Martin S. Andersen (DTU), Frank Permenter (MIT)

www.mosek.com

# Polyopt.jl

## A brief overview.

- Julia package for polynomial optimization (requires Julia 0.4).
- Implements the Lasserre hierarchy of moment relaxations.
- Uses the MOSEK conic optimizer to solve the relaxations.

### Installation

```
Pkg.clone("https://github.com/MOSEK/Polyopt.jl.git")
```

# Polynomial optimization

## What problems do we solve?

We consider polynomials optimization problems

$$\begin{array}{ll} \text{minimize} & f(x) \\ \text{subject to} & g_i(x) \geq 0, \quad i = 1, \ldots, m \\ & h_i(x) = 0, \quad i = 1, \ldots, l \\ & x \in \mathbb{R}^n \end{array}$$

for real polynomials $f, g_i, h_j : \mathbb{R}^n \mapsto \mathbb{R}$.

- Solved by a sequence of relaxations.
- An important recent application of semidefinite optimization.
- The relaxations can be difficult to solve numerically.

We consider polynomials optimization problems

$$
\begin{array}{ll}
\text{minimize} & f(x) \\
\text{subject to} & g_i(x) \geq 0, \quad i = 1, \dots, m \\
& h_i(x) = 0, \quad i = 1, \dots, l \\
& x \in \mathbb{R}^n
\end{array}
$$

for real polynomials $f, g_i, h_j : \mathbb{R}^n \mapsto \mathbb{R}$.

- Solved by a sequence of relaxations.
- An important recent application of semidefinite optimization.
- The relaxations can be difficult to solve numerically.

We consider polynomials optimization problems

$$\begin{array}{ll} \text{minimize} & f(x) \\ \text{subject to} & g_i(x) \geq 0, \quad i = 1, \ldots, m \\ & h_i(x) = 0, \quad i = 1, \ldots, l \\ & x \in \mathbb{R}^n \end{array}$$

for real polynomials $f, g_i, h_j : \mathbb{R}^n \mapsto \mathbb{R}$.

- Solved by a sequence of relaxations.
- An important recent application of semidefinite optimization.
- The relaxations can be difficult to solve numerically.

# Software packages for polynomial optimization

## Why develop a new package?

Other Matlab packages with same functionality exists:

- **GloptiPoly**, standard moment relaxations.
- **SparsePoP**, sparse moment relaxations.
- **SOSTools**, general sum-of-squares problems.
- **Yalmip**, general sums-of-squares and polynomial optimization.

Motivations for developing a new package:

- Test and improve the MOSEK semidefinite solver.
- Have full control of the generated semidefinite problems.
- Investigate other approaches for exploiting sparsity.
- Implement it in Julia to remove dependency on Matlab.

# Software packages for polynomial optimization

## Why develop a new package?

Other Matlab packages with same functionality exists:

- **GloptiPoly**, standard moment relaxations.
- **SparsePoP**, sparse moment relaxations.
- **SOSTools**, general sum-of-squares problems.
- **Yalmip**, general sums-of-squares and polynomial optimization.

Motivations for developing a new package:

- Test and improve the MOSEK semidefinite solver.
- Have full control of the generated semidefinite problems.
- Investigate other approaches for exploiting sparsity.
- Implement it in Julia to remove dependency on Matlab.

Other Matlab packages with same functionality exists:

- **GloptiPoly**, standard moment relaxations.
- **SparsePoP**, sparse moment relaxations.
- **SOSTools**, general sum-of-squares problems.
- **Yalmip**, general sums-of-squares and polynomial optimization.

Motivations for developing a new package:

- Test and improve the MOSEK semidefinite solver.
- Have full control of the generated semidefinite problems.
- Investigate other approaches for exploiting sparsity.
- Implement it in Julia to remove dependency on Matlab.

Other Matlab packages with same functionality exists:

- **GloptiPoly**, standard moment relaxations.
- **SparsePoP**, sparse moment relaxations.
- **SOSTools**, general sum-of-squares problems.
- **Yalmip**, general sums-of-squares and polynomial optimization.

Motivations for developing a new package:

- Test and improve the MOSEK semidefinite solver.
- Have full control of the generated semidefinite problems.
- Investigate other approaches for exploiting sparsity.
- Implement it in Julia to remove dependency on Matlab.

- Standard moment relaxation:

$$\begin{array}{ll} \text{minimize} & p^T y \\ \text{subject to} & y_0 = 1 \\ & M_k(y) \succeq 0 \\ & M_{k-d_{g_j}}(g_j y) \succeq 0,\ j = 1, \dots, m \\ & M_{k-d_{h_i}}(h_i y) = 0,\ i = 1, \dots, l. \end{array}$$

- Dual problem (which we feed into MOSEK):

$$\begin{array}{ll} \text{maximize} & t \\ \text{subject to} & \displaystyle\sum_{j=1}^{m} A_0^j \bullet X^j + \sum_{k=1}^{l} B_0^k \bullet Z^k = p_0 - t \\ & \displaystyle\sum_{j=1}^{m} A_i^j \bullet X^j + \sum_{k=1}^{l} B_i^k \bullet Z^k = p_i, \quad i = 1, \dots, r \\ & X^j \succeq 0,\ Z^k \text{ are free symmetric matrices.} \end{array}$$

# Lasserre hierarchy of moment relaxations

## Primal and dual formulations

- Standard moment relaxation:

$$
\begin{array}{ll}
\text{minimize} & p^T y \\
\text{subject to} & y_0 = 1 \\
& M_k(y) \succeq 0 \\
& M_{k-d_{g_j}}(g_j y) \succeq 0,\ j = 1, \ldots, m \\
& M_{k-d_{h_i}}(h_i y) = 0,\ \ i = 1, \ldots, l.
\end{array}
$$

- Dual problem (which we feed into MOSEK):

$$
\begin{array}{ll}
\text{maximize} & t \\
\text{subject to} & \displaystyle\sum_{j=1}^{m} A_0^j \bullet X^j + \sum_{k=1}^{l} B_0^k \bullet Z^k = p_0 - t \\
& \displaystyle\sum_{j=1}^{m} A_i^j \bullet X^j + \sum_{k=1}^{l} B_i^k \bullet Z^k = p_i, \quad i = 1, \ldots, r \\
& X^j \succeq 0,\ Z^k \text{ are free symmetric matrices.}
\end{array}
$$

# How to solve a simple example in Julia

$$\begin{array}{ll} \text{minimize} & -x_1 - x_2 \\ \text{subject to} & 2x_1^4 - 8x_1^3 + 8x_1^2 - x_2 + 2 \geq 0 \\ & 4x_1^4 - 32x_1^3 + 88x_1^2 - 96x_1 - x_2 + 36 \geq 0 \\ & 0 \leq x_1 \leq 3, \quad 0 \leq x_2 \leq 4. \end{array}$$

```julia
using Polyopt
x1, x2 = variables(["x1", "x2"])
f = -x1-x2
g = [ 2*x1^4 - 8*x1^3 + 8*x1^2 - x2 + 2,
      4*x1^4 - 32*x1^3 + 88*x1^2 - 96*x1 - x2 + 36,
      x1, 3-x1,
      x2, 4-x2 ]
prob = momentprob(4, f, g)
X, Z, t, y, solsta = solve_mosek(prob)
```

More examples on Github...

# Concluding remarks

Package overview:

- Lasserre's hierarchy of moment relaxations in Julia.
- Correlative sparsity and chordal relaxations by Waki et al.
- No solution extracting method by Henrion and Lasserre; perturb problem to extract a single global optimizer.

Modern features of Julia facilitate lean implementation:

- `polynomial.jl` 258 lines of code.
- `cliques.jl` 66 lines of code.
- `Polyopt.jl` 268 lines of code.
- `solver_mosek.jl` 134 lines of code.

Important for improving conic solver in upcoming MOSEK 8.0.

# Concluding remarks

Package overview:

- Lasserre's hierarchy of moment relaxations in Julia.
- Correlative sparsity and chordal relaxations by Waki et al.
- No solution extracting method by Henrion and Lasserre; perturb problem to extract a single global optimizer.

Modern features of Julia facilitate lean implementation:

- `polynomial.jl` 258 lines of code.
- `cliques.jl` 66 lines of code.
- `Polyopt.jl` 268 lines of code.
- `solver_mosek.jl` 134 lines of code.

Important for improving conic solver in upcoming MOSEK 8.0.

# Concluding remarks

Package overview:

- Lasserre's hierarchy of moment relaxations in Julia.
- Correlative sparsity and chordal relaxations by Waki et al.
- No solution extracting method by Henrion and Lasserre; perturb problem to extract a single global optimizer.

Modern features of Julia facilitate lean implementation:

- `polynomial.jl` 258 lines of code.
- `cliques.jl` 66 lines of code.
- `Polyopt.jl` 268 lines of code.
- `solver_mosek.jl` 134 lines of code.

Important for improving conic solver in upcoming MOSEK 8.0.

**Thank you!**

Joachim Dahl, MOSEK ApS

collaborators: Martin S. Andersen (DTU), Frank Permenter (MIT)

`www.mosek.com`