MOSEK 10: affine conic constraints, new cones and more...

Bethlehem, PA 2022
ICCOPT
INTERNATIONAL CONFERENCE ON CONTINUOUS OPTIMIZATION

Utkarsh Detha
utkarsh.detha@mosek.com

www.mosek.com

MOSEK ApS: all-in on cones

- MOSEK is a solver for large-scale, continuous/mixed-integer linear and conic programs.
- Established in 1997 by the CEO, **Erling D. Andersen**.
- Based in Copenhagen, Denmark.
- Version 10 (beta) of MOSEK is now available on the website.

Free MOSEK cookbooks and trial licenses on the other side of this talk!

Outline

DIY on Google colab

`github.com/MOSEK/Tutorials/tree/master/max-volume-cuboid`

Max-volume axis-parallel cuboid inscribed in a Regular Icosahedron

This notebook presents an exercise in using affine conic constraints and the geometric mean cone (introduced as a standalone domain in v10). We implement the maximum volume cuboid example discussed in the MOSEK modelling cookbook, section 4.3.2.

Try on Google colab!

Section 1

Conic programming speedrun

Linear programs (LPs)

Primal:

$$\begin{aligned} \min_{x} \quad & c^T x \\ \text{s.t.} \quad & Ax \geq b \end{aligned}$$

Dual:

$$\begin{aligned} \max_{y} \quad & b^T y \\ \text{s.t.} \quad & A^T y = c \\ & y \geq 0 \end{aligned}$$

The winning traits of LPs

Theoretical and computational perspective:

- Farkas' lemma allows certifying infeasibilities.
- Duality theory can prove optimality by the means of *zero duality gap*.
- Simplex/interior-point solvers make it easy to solve even massive LPs.

Modeling perspective:

- Structurally simple and always convex.
- Modeling is "easy as ABC"; essentially amounts to specifying A, b and c.

Convex optimization programs

$$\begin{aligned} \min_{x} \quad & f_0(x) \\ \text{s.t.} \quad & f_i(x) \leq b \qquad\qquad \forall i = 1 \ldots m \end{aligned}$$

Theoretical/computational perspective:

- Allows nonlinearity insofar as all f_i are convex.
- Duality theory can be extended to convex programs.
- Interior-point solvers quite capable at handling these problems.

Modeling perspective:

- Verifying the convexity of a function is NP-hard.
- The structure is too vague.

So, *how does one bring over the structural qualities of LPs over to convex programs*?

Orderings of Euclidean spaces and cones [?]

Key idea: "Keep the $f_i(x)$'s linear and introduce nonlinearity in the inequality sign instead."

The ordering "$\geq$" between Ax and b has the following properties:

1. Reflexivity: $a \geq a$
2. Anti-symmetry: if $a \geq b$ and $b \geq a$, then $a = b$
3. Transitivity: if $a \geq b$ and $b \geq c$, then $a \geq c$
4. Linearity: if $a \geq b$ and $c \geq d$, then $\alpha a + \beta c \geq \alpha b + \beta d$ for $\alpha, \beta \geq 0$.

Orderings of Euclidean spaces and cones [?]

- Element-wise inequality is not the only way to satisfy the properties.
- $a \geq_{\mathcal{K}} b$ is an ordering and $\mathcal{K}$ is the subset of Euclidean space that satisfies this ordering.
- $a \geq_{\mathcal{K}} b \Leftrightarrow a - b \geq_{\mathcal{K}} 0 \Leftrightarrow a - b \in \mathcal{K}$.
- The ordering is *good* if $\mathcal{K}$ is a convex cone.

Conic programs (CPs)

Primal:

$$\begin{aligned} \min_x \quad & c^T x \\ \text{s.t.} \quad & Fx \geq_{\mathcal{K}} g \end{aligned}$$

Dual:

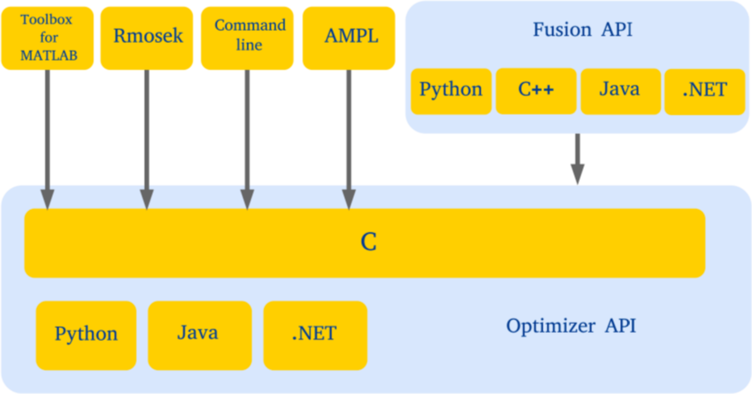
$$\begin{aligned} \max_x \quad & \langle g, \lambda \rangle \\ \text{s.t.} \quad & F^* \lambda = c \\ & \lambda \geq_{\mathcal{K}^*} 0 \end{aligned}$$

For a standard LP, $\mathcal{K}$ is simply the non-negative orthant, i.e. $\mathbb{R}^m_+$.

Section 2

Conic programming with MOSEK

Interfaces to MOSEK



MOSEK also interfaces with popular third-party modeling tools such as CVXPY, GAMS, AMPL, Pyomo, JuMP etc.

Optimizer API vs Fusion API

Optimizer API characteristics:

- Matrix oriented interface.
- Data is entered in sparse format allowing huge problems to be entered and solved easily.
- Lowest over-head out of all interfaces to MOSEK.

Fusion API characteristics:

- Expression oriented interface; code will closely resemble mathematical formulation.
- Very intuitive and allows fast-prototyping of problems.
- Despite being a layer on top of the Optimizer API, the performance overhead is minimal.

The performance is close enough that the choice is upto the use-case.

MOSEK 10 feature: Affine conic constraints

Optimizer API in MOSEK 10 allows restricting affine expressions to conic domains in CPs:

$$\begin{array}{ll} \min\limits_{x} & c^T x \\ \text{s.t.} & Fx \geq_{\mathcal{K}} g \;\Leftrightarrow\; Fx + g \in \mathcal{K} \end{array}$$

Primary advantages:

- Conic slacks are no longer essential.
- The same, simple problem struture as discussed before.
- Simplifies the process of expanding MOSEK's repertoire as new cones are included.

Linear cones in MOSEK 10

LPs can be modelled using affine conic constraints restricted to the following domains:

- $\mathbb{R}^n$
- $\mathbb{R}^n_0$ $(= 0)$
- $\mathbb{R}^n_+$ (≥ 0)
- $\mathbb{R}^n_-$ (≤ 0)

NOTE: The old approach for specifying linear data is required to use the simplex solver for LPs.

Symmetric nonlinear cones in MOSEK 10

Symmetric cones are self-dual and homogenous.

- **Quadratic cone**
  $$\mathcal{Q}^n = \left\{ x \in \mathbb{R}^n : x_0 \geq \sqrt{\sum_{j=1}^{n-1} x_j^2} \right\}$$
- **Rotated quadratic cone**
  $$\mathcal{Q}_r^n = \left\{ x \in \mathbb{R}^n : 2x_0 x_1 \geq \sum_{j=2}^{n-1} x_j^2, \quad x_0 \geq 0, \quad x_1 \geq 0 \right\}$$
- **Positive semidefinite cone** (for *variables*)
  $\mathcal{S}_+^r = \left\{ X \in \mathcal{S}^r : z^T X z \geq 0, \quad \forall z \in \mathbb{R}^r \right\}$
  (Vectorized positive semidefinite cone for *ACCs*. Think LMIs!)

Non-symmetric nonlinear cones in MOSEK 10

The following non-symmetric cones **and their dual cones** are supported in MOSEK 10:

- **Primal exponential cone**
  $\{x \in \mathbb{R}^3 : x_0 \geq x_1 \exp(x_2/x_1),\ x_0, x_1 \geq 0\}$
- **Primal power cone** ($n-$dimensional)
  $$\left\{ x \in \mathbb{R}^n \ : \ \prod_{i=0}^{n_\ell - 1} x_i^{\beta_i} \geq \sqrt{\sum_{j=n_\ell}^{n-1} x_j^2},\ x_0 \ldots, x_{n_\ell - 1} \geq 0 \right\}$$
- **Primal geometric mean cone**
  $$\left\{ x \in \mathbb{R}^n \ : \ \left(\prod_{i=0}^{n-2} x_i \right)^{1/(n-1)} \geq |x_{n-1}|,\ x_0 \ldots, x_{n-2} \geq 0 \right\}$$

Section 3

Exercise in affine conic constraints

Maximum volume cuboid

Q: How to find the max-volume axis-parallel cuboid inscribed in a conic representable set, $K \in \mathbb{R}^n$, such as a *regular icosahedron*?
A:

$$\begin{aligned} \max_{x,t,p} \quad & t \\ \text{s.t.} \quad & (x_1 \dots x_n)^{\frac{1}{n}} \geq t \\ & (p_1 + e^i_1 x_1, \dots, p_n + e^i_n x_n) \in K \qquad e^i_j \in \{0,1\} \\ & x \geq 0 \end{aligned}$$

where,

- $p \in \mathbb{R}^n$ is the left-most corner of the cuboid.
- x_j are edge lengths and t^n the volume of the cuboid
- Vectors e^j *enumerate* vertices of the cuboid (Eg.: $\{000\}, \{001\}, \dots, \{111\})$.

Convex hull of polyhedron vertices

If $K = \text{conv}(\text{vertices})$, then the second set of constraints becomes:

$$(p_1 + e_1^i x_1, \ldots, p_n + e_n^i x_n) = u_1^i v_1 + \cdots + u_m^i v_m \quad \forall i = 1, \ldots, 2^n$$

$$\sum_{j=1}^{m} u_j^i = 1 \quad \forall i = 1, \ldots, 2^n$$

$$u \geq 0$$

where:

- $v_k \ (k = 1, \ldots, m)$ are vertices of the polyhedron, each an n-vector.
- u_j^i are scalar variables used in the convex combination of the polyhedron vertices.
- The index i runs from $1, \ldots, 2^n$, corresponding to each *cuboid* vertex.

Affine conic constriant formulation

Matrix form of the model conceptually resembles the following:

$$
\begin{aligned}
\max_{x,t,p,u} \quad & t \\
\text{s.t.} \quad & \begin{bmatrix} I & \\ & 1 \end{bmatrix} \begin{bmatrix} x \\ t \end{bmatrix} \in \mathcal{K}_{geo}^{n+1} \\
& \begin{bmatrix} E^i & 0 & I & \cdots & -V \\ \cdots & 0 & & \cdots & 1 \end{bmatrix} \begin{bmatrix} x \\ t \\ p \\ u^i \end{bmatrix} + \begin{bmatrix} 0 \\ -1 \end{bmatrix} \in \mathbb{R}_0^{n+1} \quad \forall i = 1, . \\
& x, u \geq 0
\end{aligned}
$$

where, $E^i = \begin{bmatrix} e_1^i & & \\ & \ddots & \\ & & e_n^i \end{bmatrix}$; $V = \begin{bmatrix} v_1^1 & & v_1^m \\ \vdots & \ldots & \vdots \\ v_n^1 & & v_n^m \end{bmatrix}$

Optimizer API implementation

1. Create a MOSEK task object:

```
# MOSEK TASK
    task = Task()
    task.set_Stream(streamtype.log, streamprinter)
```

2. **Variables**: append variables to task and set bounds.

```
# VARIABLES: dim(x) = n; dim(t) = 1; dim(p) = n
    task.appendvars(2*n+1)
    task.putvarboundsliceconst(0, n, boundkey.lo, 0, inf)
    task.putvarboundsliceconst(n,
                               2*n+1,
                               boundkey.fr, -inf, inf)
```

Optimizer API implementation

❸ ACCs:

```python
# GEOMETRIC MEAN CONE:
    # 1. AFE
    task.appendafes(n+1)
    task.putafefentrylist(range(n+1),
                          range(n+1),
                          [1.0]*(n+1))
    # 2. Domain
    geo_cone = task.appendprimalgeomeanconedomain(n+1)
    # 3. AFE \in Domain --> ACC
    task.appendacc(geo_cone, range(n+1), None)

# CONVEX HULL CONSTRAINTS:
    # Re-use this domain instance for all ACCs hereafter.
    r_zero = task.appendrzerodomain(n+1)

    # One ACC for each vertex of the cuboid
    for i, c_v in enumerate(cuboid):
        convexHullConstraint(task, polyhedron, c_v, r_zero)
```

Optimizer API implementation

❸
```python
def convexHullConstraint(task, p_v, c_v, dom):
    m, n = len(p_v), len(p_v[0])
    nvar, nafe = task.getnumvar(), task.getnumafe()
    # VARIABLES: dim(u) = m
    task.appendvars(m)
    task.putvarboundsliceconst(nvar, nvar+m,
                               boundkey.lo, 0, inf)
    # Append n+1 affine expressions to the task.
    task.appendafes(n+1)
    for i in range(n):
        task.putafefrow(nafe + i,
                        [i,i+n+1]+list(range(nvar,nvar+m)),
                        [c_v[i], 1.0] + list(-p_v[:, i]))
    task.putafefrow(nafe + n,
                    range(nvar, nvar+m),
                    [1.0]*m)
    task.putafeg(nafe + n, -1)
    # Construct the ACC
    task.appendacc(dom, range(nafe, nafe+n+1), None)
```

task.optimize()

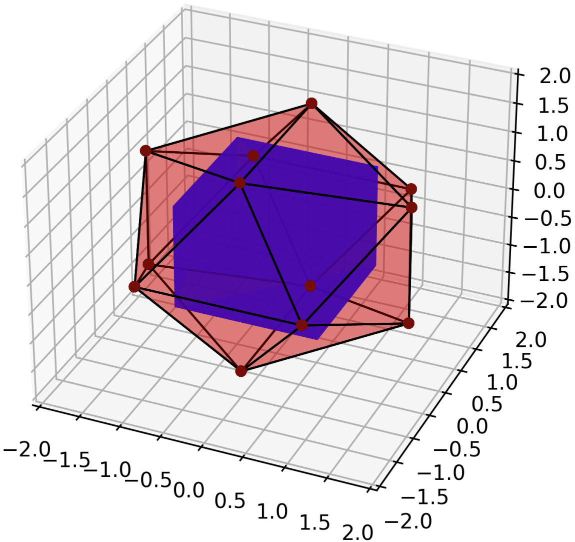
```
  Objective sense       : maximize
  Type                  : CONIC (conic optimization problem)
  Constraints           : 0
  Affine conic cons.    : 9
  Disjunctive cons.     : 0
  Cones                 : 0
  Scalar variables      : 103
  Matrix variables      : 0
  Integer variables     : 0

Optimizer  - threads                : 8
Optimizer  - solved problem         : the primal
Optimizer  - Constraints            : 33
Optimizer  - Cones                  : 3
Optimizer  - Scalar variables       : 106               conic                  : 10
Optimizer  - Semi-definite variables: 0                 scalarized             : 0
Factor     - setup time             : 0.00              dense det. time        : 0.00
Factor     - ML order time          : 0.00              GP order time          : 0.00
Factor     - nonzeros before factor : 345               after factor           : 345
Factor     - dense dim.             : 0                 flops                  : 7.46e+03
ITE PFEAS    DFEAS    GFEAS    PRSTATUS   POBJ              DOBJ              MU       TIME
0   1.3e+00  1.3e+00  1.0e+00  0.00e+00   0.000000000e+00   0.000000000e+00   1.0e+00  0.01
1   1.0e+00  1.0e+00  7.0e-01  6.38e+00   5.996194085e-01   8.628658378e-02   7.7e-01  0.03
2   6.1e-01  6.1e-01  3.9e-01  1.81e+00   1.197930307e+00   7.154515869e-01   4.7e-01  0.03
3   8.8e-02  8.8e-02  1.8e-02  1.21e+00   1.678293298e+00   1.633318388e+00   6.8e-02  0.03
4   4.5e-03  4.5e-03  2.2e-04  1.05e+00   1.741507264e+00   1.738847012e+00   3.5e-03  0.03
5   1.4e-04  1.4e-04  1.1e-06  1.00e+00   1.745179991e+00   1.745100951e+00   1.0e-04  0.03
6   3.1e-06  3.1e-06  3.9e-09  1.00e+00   1.745351325e+00   1.745349557e+00   2.4e-06  0.03
7   1.2e-08  1.2e-08  9.7e-13  1.00e+00   1.745355973e+00   1.745355966e+00   9.4e-09  0.03
Optimizer terminated. Time: 0.04


Interior-point solution summary
  Problem status  : PRIMAL_AND_DUAL_FEASIBLE
  Solution status : OPTIMAL
  Primal.  obj: 1.7453559726e+00    nrm: 2e+00    Viol.  var: 2e-09    acc: 4e-09
  Dual.    obj: 1.7453559656e+00    nrm: 1e+00    Viol.  var: 4e-09    acc: 0e+00

Volume of the inscribed cuboid = 5.3168211243744
```

Biggest axis-parallel cuboid



Section 4

Disjunctive constraints

Disjunction of conjunction of ACCs

MOSEK 10 introduces language for stating disjunctive constraints which are *logical-or* (optionally logical-*and*) based combinations of affine conditions.

$$T_{ij} = D_{ij}x + d_{ij} \in \mathcal{D}_{ij}$$
$$\downarrow$$
$$T_i = T_{i1} \text{ and } T_{i2} \text{ and } \cdots$$
$$\downarrow$$
$$\text{DJC} = T_1 \text{ or } T_2 \text{ or } \cdots$$

This language is incorporated into *both* the Optimizer API and the Fusion API.

Section 5

Performance improvements in MOSEK 10

Pedal to all the metals!

- Native support for Apple silicon.
- Multi-threading support on Linux ARM 64-bit.
- Significantly improved interior-point performance on AMD CPUs.
- Dramatically better multi-threaded performance on special SDPs.

Optimizing the optimizer

- Presolve: significant improvements for conic problems and mixed-integer problems.
- Interior-point solver: better performance on large-scale LPs
- Mixed-integer optimizer: introduced symmetry detection and reformulation methods for MIQCQPs. Improved cutting-plane separation.
- Faster file I/O and introduction of the PTF human-readable file format for CPs and SDPs.

Further information

- Mosek `https://mosek.com`
  - Trial and free academic license.
  - Solves linear and conic mixed problems.
  - Interfaces C/C++, Java, Julia, Matlab, R, Python, ...
- Documentation at `https://www.mosek.com/documentation/`
  - Modeling cookbook.
  - Portfolio optimization cookbook.
  - Modeling cheat sheet.
- Examples
  - Tutorials at Github: `https://github.com/MOSEK/Tutorials`
  - Example + **30 day license**: `https://github.com/MOSEK/Tutorials/tree/master/max-volume-cuboid`

References I

[1] A. Ben-Tal and A. Nemirovski.
Lectures on Modern Convex Optimization: Analysis, Algorithms, and Engineering Applications.
MPS/SIAM Series on Optimization. SIAM, 2001.