Conic Optimization in Julia and JuMP

Juan Pablo Vielma

JuliaCon 2020

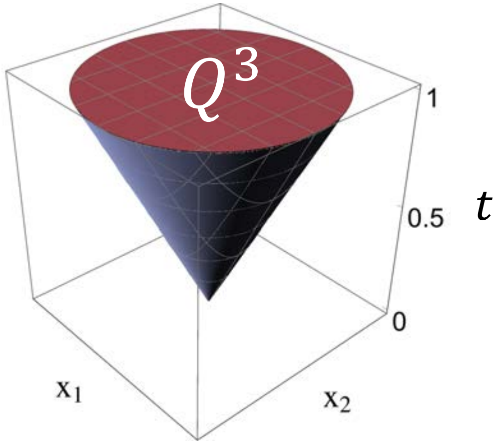
Funded by NSF OAC-1835443

Conic Optimization in JuMP

- From https://github.com/jump-dev/JuMPTutorials.jl - conic_programming.ipynb

$$
\begin{aligned}
\min \quad & ||u - u_0|| \\
\text{s.t.} \quad & p' \cdot u = q
\end{aligned}
\quad \longrightarrow \quad
\begin{aligned}
\min \quad & t \\
\text{s.t.} \quad & p' \cdot u = q \\
& (t, u - u_0) \in Q^{n+1}
\end{aligned}
$$

$$Q^n = \{(t, x) \in \mathbb{R}^n : t \geq ||x||_2\}$$




```
model = Model(optimizer_with_attributes(ECOS.Optimizer, "printlevel" => 0))
@variable(model, u[1:10])
@variable(model, t)
@objective(model, Min, t)
@constraint(model, [t, (u - u0)...] in SecondOrderCone())
@constraint(model, u' * p == q)
optimize!(model)
```

Outline / Goal ?

- Why Conic (or Function-in-set) v/s expression/“(Julia) function”-based :

$$
\begin{array}{ll}
\min & ||u - u_0|| \\
\text{s.t.} & p' \cdot u = q
\end{array}
$$

$$
\boxed{
\begin{array}{lr}
\min & t \\
\text{s.t.} & p' \cdot u = q \\
 & (t, u - u_0) \in Q^{n+1}
\end{array}
}
$$

$$
\boxed{
\begin{array}{llll}
\min & f(x) & & \\
\text{subject to} & g(x) & \leq & 0,
\end{array}
}
$$

- Why does Pajarito.jl only work on JuMP ≤ v0.18?
- Many links for skipped mathematical details (Technical but, “standard”)

Conic Optimization in JuMP

- From https://github.com/jump-dev/JuMPTutorials.jl - conic_programming.ipynb
- In general, a **conic optimization problem** is

$$\begin{aligned} \min_{x \in \mathbb{R}^n} \quad & a_0^T x + b_0 \\ \text{s.t.} \quad & A_i x + b_i \in C_i \qquad i = 1 \ldots m \end{aligned}$$

C_i is a closed convex cone

- e.g. cone of positive semi-definite (SDP) matrices:

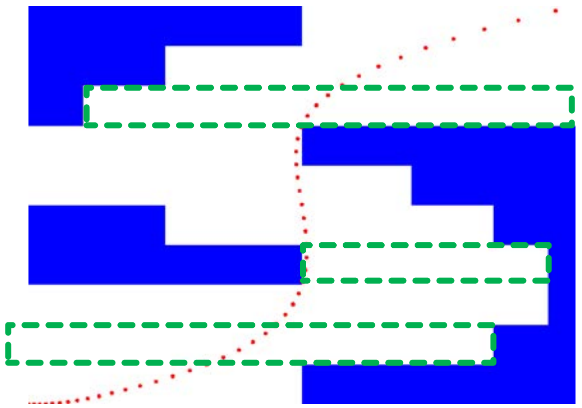
$$\mathcal{S}_+^n = \{X \in \mathcal{S}^n \mid z^T X z \geq 0, \ \forall z \in \mathbb{R}^n\}$$

– Largest Eigenvalue of a Symmetric Matrix:

```
model = Model(optimizer_with_attributes(SCS.Optimizer, "verbose" => 0))
@variable(model, t)
@objective(model, Min, t)
@constraint(model, t .* Matrix{Float64}(I, 3, 3) - A in PSDCone())
```

A more interesting (mixed-integer) conic example

- **Obstacle** avoiding **polynomial trajectory**:



$(x(t), y(t))_{t \in [0,1]}$

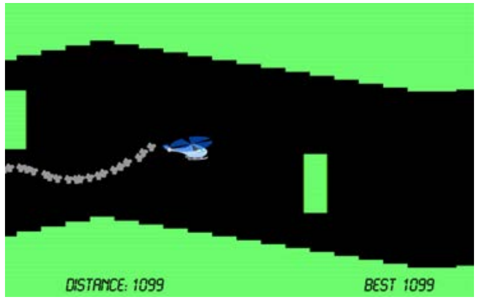
- Step 1: discretize to piecewise polynomial
  $0 = T_1 < T_2 < \cdots < T_N$ s.t.
  $(x(t), y(t)) = p_i(t) \quad t \in [T_i, T_{i+1}]$

- Step 2: "safe polyhedrons"
  $P^r = \{x \in \mathbb{R}^2 : A^r x \le b^r\}$ s.t.
  $\forall\, i\ \exists\, r\ \ s.t.\ \ p_i(t) \in P^r \quad t \in [T_i, T_{i+1}]$

- $p_i(t) \in P^r \rightarrow q_{i,r}(t) \ge 0\ \forall t$
- Sum-of-Squares (SOS):
  - $q_{i,r}(t) = \sum_j r_j^2(t)$
  - Polynomials $r_j(t)$
  - Bound degree of polynomials:
    - ✓ SDP
- More details:
  - Lecture at LANL Grid Science Winter School (https://github.com/juan-pablo-vielma/grid-science-2019)
  - PolyJuMP.jl docs (including JuliaCon 2019 slides)
  - Book by Blekherman, Parrilo and Thomas (http://www.mit.edu/~parrilo/sdocag/)

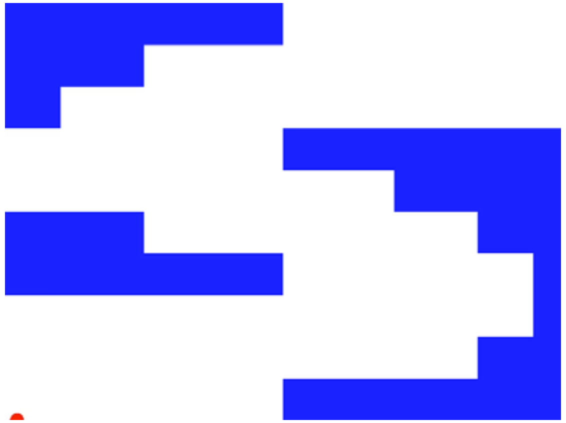
Computational Results (SIAM Opt 2017, Joey Huchette)

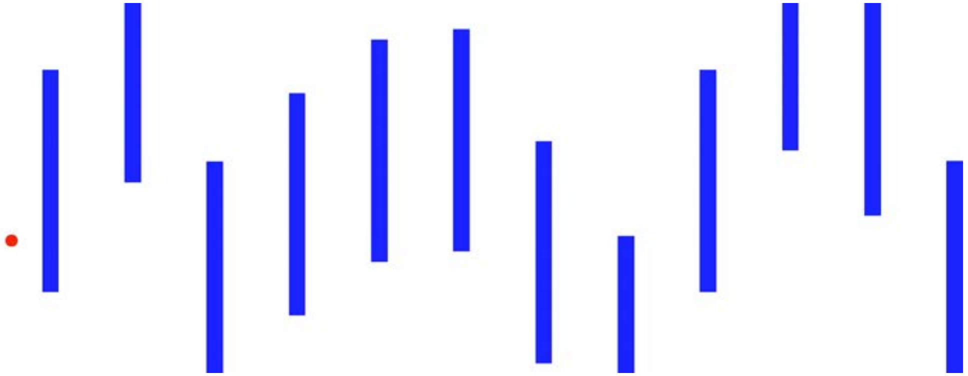
- PolyJuMP.jl
- SumOfSquares.jl
- MI-SDP Solver:
  - **Pajarito.jl**



9 Regions & 8 time steps:
Optimal “Smoothness” in 651 seconds

60 horizontal segments & obstacle every 5:
Optimal “clicks” in 80 seconds





Do I really need to learn conic optimization?

- For polynomial optimization:
  - PolyJuMP.jl
- For something close to function/expression-based modeling:
  - Disciplined Convex Programming (DCP) with Convex.jl

```julia
model = SOSModel(solver=PajaritoSolver())

@polyvar(t)
Z = monomials([t], 0:r)

@variable(model, H[1:N,boxes], Bin)

p = Dict()
for j in 1:N
   @constraint(model, sum(H[j,box] for box in boxes) == 1)
   p[(:x,j)] = @polyvariable(model, _, Z)
   p[(:y,j)] = @polyvariable(model, _, Z)
   for box in boxes
       xl, xu, yl, yu = box.xl, box.xu, box.yl, box.yu
       @polyconstraint(model, p[(:x,j)] >= Mxl + (xl-Mxl)*H[j,box], domain = (t >= T[j] && t <= T[j+1]))
       @polyconstraint(model, p[(:x,j)] <= Mxu + (xu-Mxu)*H[j,box], domain = (t >= T[j] && t <= T[j+1]))
       @polyconstraint(model, p[(:y,j)] >= Myl + (yl-Myl)*H[j,box], domain = (t >= T[j] && t <= T[j+1]))
       @polyconstraint(model, p[(:y,j)] <= Myu + (yu-Myu)*H[j,box], domain = (t >= T[j] && t <= T[j+1]))
   end
end

for ax in (:x,:y)
   @constraint(model,                p[(ax,1)        ]([0], [t]) == X₀[ax])
   @constraint(model, differentiate(p[(ax,1)], t    )([0], [t]) == X₀′[ax])
   @constraint(model, differentiate(p[(ax,1)], t, 2)([0], [t]) == X₀′′[ax])
   for j in 1:N-1
       @constraint(model,                p[(ax,j)      ]([T[j+1]],[t]) ==                p[(ax,j+1)      ]([T[j+1]],[t]))
       @constraint(model, differentiate(p[(ax,j)],t   )([T[j+1]],[t]) == differentiate(p[(ax,j+1)],t   )([T[j+1]],[t]))
       @constraint(model, differentiate(p[(ax,j)],t,2)([T[j+1]],[t]) == differentiate(p[(ax,j+1)],t,2)([T[j+1]],[t]))
   end
   @constraint(model,                p[(ax,N)        ]([1], [t]) == X₁[ax])
   @constraint(model, differentiate(p[(ax,N)], t    )([1], [t]) == X₁′[ax])
   @constraint(model, differentiate(p[(ax,N)], t, 2)([1], [t]) == X₁′′[ax])
end

@variable(model, γ[keys(p)] ≥ 0)
for (key,val) in p
   @constraint(model, γ[key] ≥ norm(differentiate(val, t, 3)))
end
@objective(model, Min, sum(γ))
```

Why Conic 1: Linear-programming-like duality

- From https://github.com/jump-dev/JuMPTutorials.jl - conic_programming.ipynb

(Primal)

$$\min_{x \in \mathbb{R}^n} \; a_0^T x + b_0$$
$$\text{s.t.} \quad A_i x + b_i \in \mathcal{C}_i \quad i = 1 \ldots m$$

(Dual)

$$\max_{y_1, \ldots, y_m} \; - \sum_{i=1}^{m} b_i^T y_i + b_0$$
$$\text{s.t.} \quad a_0 - \sum_{i=1}^{m} A_i^T y_i = 0$$
$$y_i \in \mathcal{C}_i^* \quad i = 1 \ldots m$$

- “Explanation” of optimality : Think Max-flow/Min-cut or Menger’s theorem.
  - More: *Set Programming : Theory and Computation*, June 2020.
    Ph.D. Thesis by Dr. Benoît Legat 🎉 (https://blegat.github.io/publications/#phd_thesis)
- and…

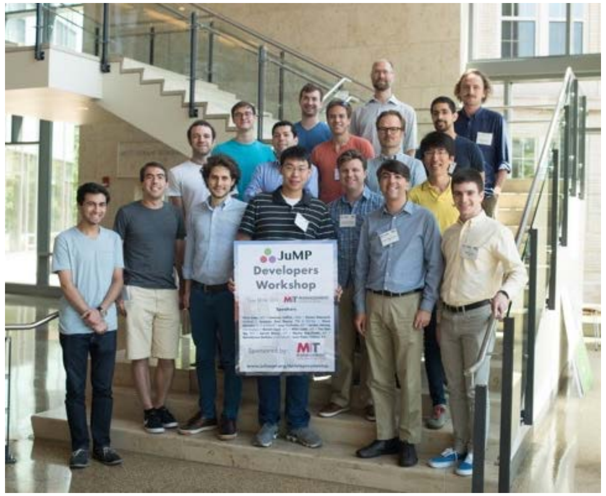
Why Conic 2: Faster and more stable algorithms

- Avoid non-differentiability issues, exploit primal-dual form, strong theory on barriers for interior point algorithms.
- Industry change in 2018:
  - KNITRO® version 11.0 adds support for SOCP constraints
  - mosek version 9.0 deprecates expression/function-based formulations and focuses on pure conic (linear, SOCP, rotated SOCP, SDP, exp & power)

Conic Optimization @ JuMP-dev I

- **Monday June 12, 2017**
  - 09:30 *The design of* ***JuMP*** *and* ***MathProgBase*** (**Miles Lubin**, MIT)
  - 10:30 *The design and architecture of* ***Pajarito*** (**Chris Coey**, MIT)
- **Tuesday June 13, 2017,**
  - 09:15 ***Sum-of-squares*** *optimization in Julia* [**remote presentation**] (**Benoît Legat**, Université Catholique de Louvain)

1st JuMP-dev Workshop, Cambridge, MA



Slides and videos:
https://jump.dev/meetings/mit2017/

Conic Optimization @ JuMP-dev II

- ***Artelys Knitro*** *11.0, a* ***new conic solver*** *and other novelties* (**Jean-hubert Hours**, Artelys)
- ***Power*** *and* ***exponential Cones*** *with* ***Mosek*** (**Ulf Worsøe**, MOSEK)
- *ProxSDP.jl: A* ***semidefinite programming solver*** *written in Julia* (**Joaquim Dias Garcia** & **Mario Souto**, PUC-Rio)
- *A Julia JuMP-based module for* ***polynomial optimization*** *with complex variables applied to Optimal Power Flow* (**Julie Sliwak**, RTE)
- ***MathOptInterface*** *and* ***JuMP 0.19*** (**Miles Lubin**, Google)
- ***Automatic reformulation using constraint bridges*** (**Benoît Legat**, UCLouvain)

2nd JuMP-dev Workshop, Bordeaux, France, June, 2018

Slides and videos:
https://jump.dev/meetings/bordeaux2018/

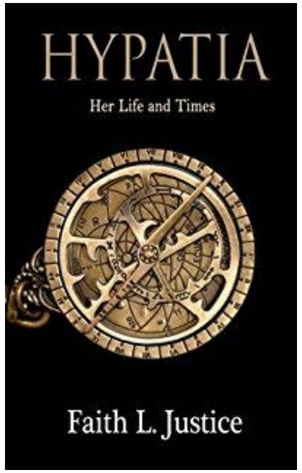
A “Few” Cones in MathOptInterface’s Function-in-Set Interface

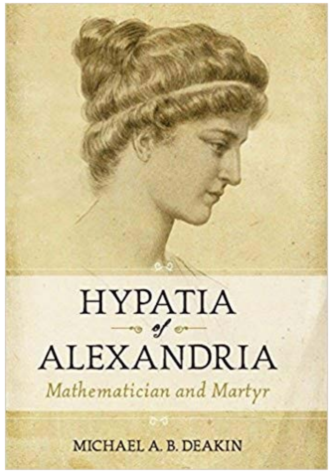
https://jump.dev/MathOptInterface.jl/dev/apimanual/index.html#Standard-form-problem-1

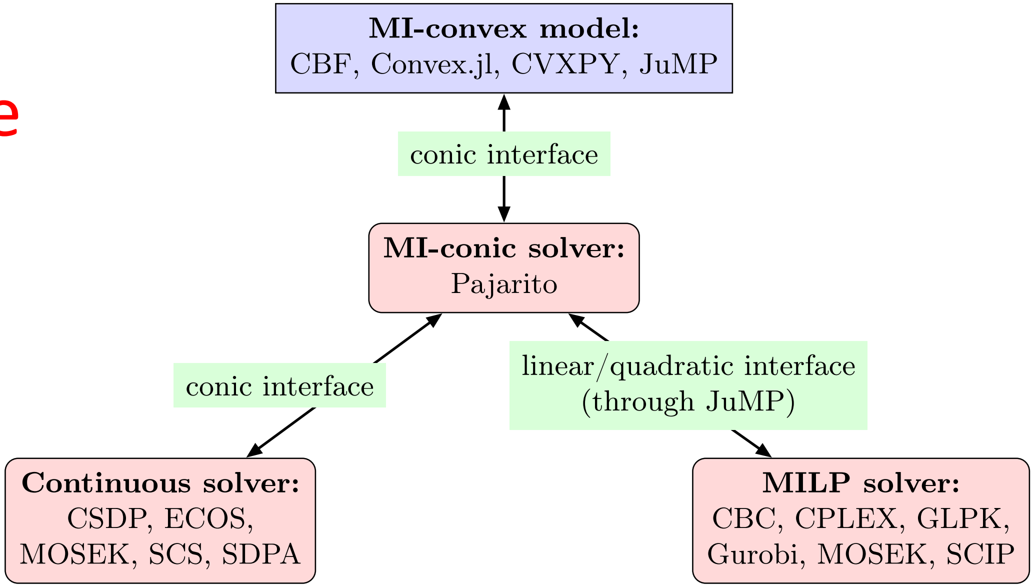
- `Reals(dimension)`: $\mathbb{R}^{\text{dimension}}$
- `Zeros(dimension)`: $0^{\text{dimension}}$
- `Nonnegatives(dimension)`: $\{x \in \mathbb{R}^{\text{dimension}} : x \geq 0\}$
- `Nonpositives(dimension)`: $\{x \in \mathbb{R}^{\text{dimension}} : x \leq 0\}$
- `NormInfinityCone(dimension)`: $\{(t, x) \in \mathbb{R}^{\text{dimension}} : t \geq \|x\|_\infty = \max_i |x_i|\}$
- `NormOneCone(dimension)`: $\{(t, x) \in \mathbb{R}^{\text{dimension}} : t \geq \|x\|_1 = \sum_i |x_i|\}$
- `SecondOrderCone(dimension)`: $\{(t, x) \in \mathbb{R}^{\text{dimension}} : t \geq ||x||_2\}$
- `RotatedSecondOrderCone(dimension)`: $\{(t, u, x) \in \mathbb{R}^{\text{dimension}} : 2tu \geq ||x||_2^2, t, u \geq 0\}$
- `GeometricMeanCone(dimension)`: $\{(t, x) \in \mathbb{R}^{n+1} : x \geq 0, t \leq \sqrt[n]{x_1 x_2 \cdots x_n}\}$ where n is $dimension - 1$
- `ExponentialCone()`: $\{(x, y, z) \in \mathbb{R}^3 : y \exp(x/y) \leq z, y > 0\}$
- `DualExponentialCone()`: $\{(u, v, w) \in \mathbb{R}^3 : -u \exp(v/u) \leq exp(1)w, u < 0\}$
- `PowerCone(exponent)`: $\{(x, y, z) \in \mathbb{R}^3 : x^{\text{exponent}} y^{1-\text{exponent}} \geq |z|, x, y \geq 0\}$
- `DualPowerCone(exponent)`: $\{(u, v, w) \in \mathbb{R}^3 : \frac{u}{\text{exponent}}^{\text{exponent}} \frac{v}{1-\text{exponent}}^{1-\text{exponent}} \geq |w|, u, v \geq 0\}$
- `RelativeEntropyCone(dimension)`: $\{(u, v, w) \in \mathbb{R}^{\text{dimension}} : u \geq \sum_i w_i \log(\frac{w_i}{v_i}), v_i \geq 0, w_i \geq 0\}$
- `NormSpectralCone(row_dim, column_dim)`: $\{(t, X) \in \mathbb{R}^{1+\text{row\_dim}\times\text{column\_dim}} : t \geq \sigma_1(X), X \text{is a matrix with row\_dim rows and column\_dim columns}\}$
- `NormNuclearCone(row_dim, column_dim)`: $\{(t, X) \in \mathbb{R}^{1+\text{row\_dim}\times\text{column\_dim}} : t \geq \sum_i \sigma_i(X), X \text{is a matrix with row\_dim rows and column\_dim columns}\}$
- `PositiveSemidefiniteConeTriangle(dimension)`: $\{X \in \mathbb{R}^{\text{dimension}(\text{dimension}+1)/2} : X \text{is the upper triangle of a PSD matrix}\}$
- `PositiveSemidefiniteConeSquare(dimension)`: $\{X \in \mathbb{R}^{\text{dimension}^2} : X \text{is a PSD matrix}\}$
- `LogDetConeTriangle(dimension)`: $\{(t, u, X) \in \mathbb{R}^{2+\text{dimension}(1+\text{dimension})/2} : t \leq u \log(\det(X/u)), X \text{is the upper triangle of a PSD matrix}, u > 0\}$
- `LogDetConeSquare(dimension)`: $\{(t, u, X) \in \mathbb{R}^{2+\text{dimension}^2} : t \leq u \log(\det(X/u)), X \text{is a PSD matrix}, u > 0\}$
- `RootDetConeTriangle(dimension)`: $\{(t, X) \in \mathbb{R}^{1+\text{dimension}(1+\text{dimension})/2} : t \leq det(X)^{1/\text{dimension}}, X \text{is the upper triangle of a PSD matrix}\}$
- `RootDetConeSquare(dimension)`: $\{(t, X) \in \mathbb{R}^{1+\text{dimension}^2} : t \leq \det(X)^{1/\text{dimension}}, X \text{is a PSD matrix}\}$

Just need to port Pajarito to MathOptInterface (MOI) …

- Pajarito: A Julia?-based MICP Solver
- Chris Coey ~ July, 2018: How about a pure Julia-based **continuous solver** with direct support for **more cones** (no-bridge)?
- Not experts on **continuous solvers**, but Julia makes you bold!
- Hypatia.jl







mosek

"Only" standard cones: linear, SOCP, rotated SOCP, SDP, exp & power cones (still enough to *model* all MOI cones through bridges).

Selection of Conic Optimization @ JuMP-dev III

- ***The Hypatia.jl solver: conic interior point algorithms and interfaces*** (Chris Coey, MIT)
- ***Modeling with new and nonsymmetric cones*** (Lea Kapelevich, MIT)
- ***Tulip.jl: An interior-point [LP] solver with abstract linear algebra*** (Mathieu Tanneau, Polytechnique Montréal)
  - Only pure Julia solver included in classical benchmarks by H. Mittelmann (http://plato.asu.edu/ftp/lpbar.html)
- ***Set Programming with JuMP*** (Benoît Legat, UC Louvain)
  - https://blegat.github.io/publications/#phd_thesis
- ***JuliaMoments*** (Tillmann Weisser, Los Alamos National Laboratory)
  - Dual of Sum-of-Squares

3rd JuMP-dev Workshop, Santiago, Chile, March, 2019

Slides and videos:
https://jump.dev/meetings/santiago2019/

Hypatia: Pure Julia-based IPM Beyond “Standard” Cones

- A homogeneous interior-point solver for non-symmetric cones (Coey, Kapelevich & V. https://arxiv.org/abs/2005.01136)
- Versatility & performance = More Cones! :
  - Two dozen predefined **standard** and **exotic cones**
    - e.g. **SDP**, **Sum-of-Squares** and **“Matrix” Sum-of-Squares for convexity/shape constraints**
  - Customizable: “Bring your own **barrier**” = “Bring your own cone”
  - Take advantage of **Natural formulations**
  - Take advantage of Julia: multi-precision arithmetic, abstract linear operators, etc.

Interior Point Algorithms, Central Path and **Barriers**

- After some math (e.g see [paper]): Optimize = "Follow" μ: 1 → 0 in

$$
\begin{array}{lll}
Ew = \mu E w^0 & \kappa\tau = \mu, & (z, \tau, s, \kappa) \in \operatorname{int}\left(\mathcal{K}^* \times \mathbb{R}_{\geq} \times \mathcal{K} \times \mathbb{R}_{\geq}\right). \\
\boxed{z_k = -\mu g_k(s_k)} & \forall k \in K_{\mathrm{pr}}, & \\
s_k = -\mu g_k(z_k) & \forall k \in K_{\mathrm{du}}, & \mathcal{K} = \mathcal{K}_1 \times \cdots \times \mathcal{K}_K
\end{array}
$$

- $g_k(s_k)$: gradient of **Logarithmically Homogeneous Self-Concordant Barrier** $f_k(s_k)$ for "primitive" cone $\mathcal{K}_k$
- **Barrier** examples:
  - `Nonnegatives(dimension)`: $\{x \in \mathbb{R}^{\text{dimension}} : x \geq 0\}$, dimension $= 1$:

$$f(x) = -\log(x)$$

  - `LogDetConeSquare(dimension)`: $\{(t, u, X) \in \mathbb{R}^{2+\text{dimension}^2} : t \leq u \log(\det(X/u)), X \text{is a PSD matrix}, u > 0\}$:

$$f(t, u, X) = 16^2(-\log(u \log(\det(X/u) - t) - \log(\det(X)) - (d+1)\log(u))$$

- Bring-Your-Own-**Barrier** for new cones: **barrier** could be Julia code!

More cones = Smaller Models: e.g. D-Optimal Experimental Design

$$\max_{x \in \mathbb{R}^m, \lambda \in \mathbb{R}} \quad \lambda :$$
$$e \cdot x = n_{tot},$$
$$\lambda \le \operatorname{logdet}(V \operatorname{diag}(x) V^T),$$
$$x \ge 0$$

Natural Conic Formulation

$$\max_{x \in \mathbb{R}^m, \lambda \in \mathbb{R}} \quad \lambda :$$
$$e \cdot x = n_{tot},$$
$$(\lambda, 1, V \operatorname{diag}(x) V^T) \in \mathcal{K}_{\text{logdet}},$$
$$x \ge 0$$

$\mathcal{K}_{\text{logdet}} =$ `LogDetConeSquare`

Extended Conic (SDP+Exp) Formulation

$$\max_{x \in \mathbb{R}^m, \lambda \in \mathbb{R}, \Delta \in \mathbb{S}^k, t \in \mathbb{R}^k} \quad \lambda :$$
$$e \cdot x = n_{tot},$$
$$\begin{pmatrix} V \operatorname{diag}(x) V^T & \Delta \\ \Delta^T & \operatorname{diag}(\Delta) \end{pmatrix} \in \mathcal{K}_{SDP}$$
$$\Delta_{i,j} = 0 \quad \forall i < j,$$
$$\lambda \le \sum_{i=1}^k t_i,$$
$$(\Delta_{i,i}, 1, t_i) \in \mathcal{K}_{exp} \quad \forall i$$
$$x \ge 0$$

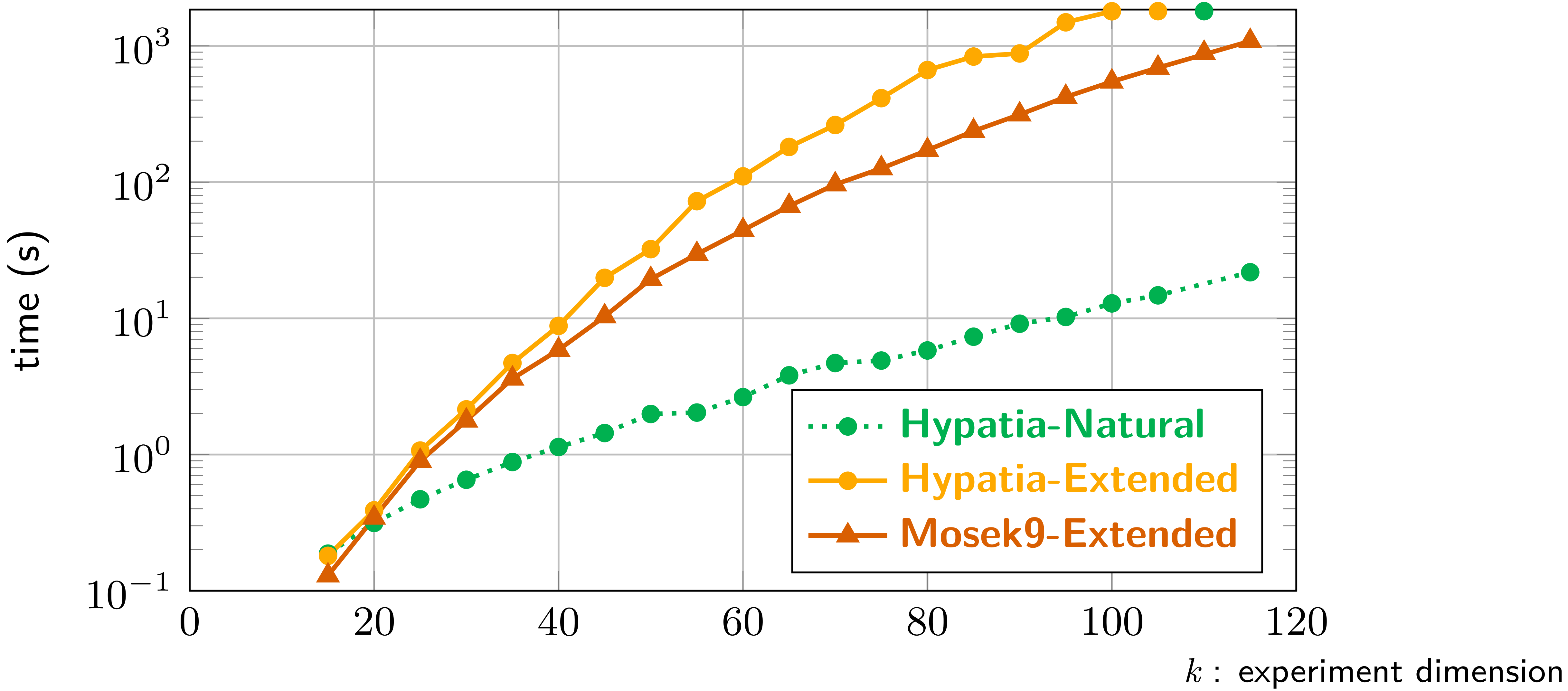
Compatible with MOSEK

$\mathcal{K}_{SDP}$ = `PositiveSemidefiniteConeSquare`

$\mathcal{K}_{exp}$ = `ExponentialCone`

More cones = Lower Memory Use and Speed Improvements

*** Older version of experiments.**



Final Remarks

- More on Hypatia (repo and more soon):
  - Coey, Kapelevich & V. https://arxiv.org/abs/2005.01136
- Conic Optimization @ JuMP-dev IV?
  - 4th JuMP-dev Workshop, Louvain-la-Neuve, Belgium, June, 2020. Canceled due to COVID-19
  - Yurii Nesterov was scheduled to give a plenary
- Interesting JuMP applications?
  - Please email oscar.dowson at northwestern.edu
- Parting thought: Julia makes you bold. Use it to try new math before learning it, not to avoid learning it