Sum of squares for control

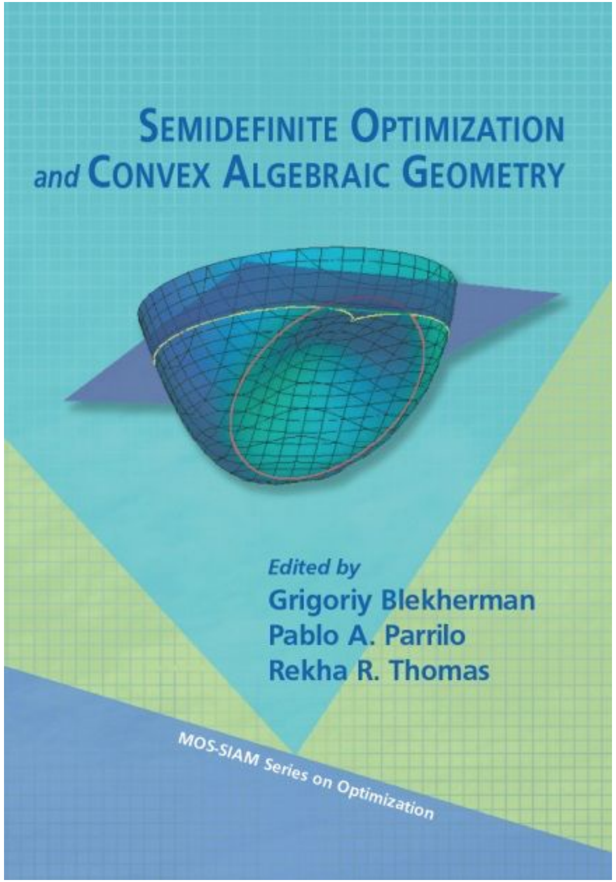
Thanks to Joey Huchette (MIT ORC) for this material

How to prove a polynomial *f(x)* is nonnegative for all *x*?

- One way: find a decomposition of *f(x)* into a sum of squared polynomials

$$f(x) = \sum_i g_i^2(x)$$

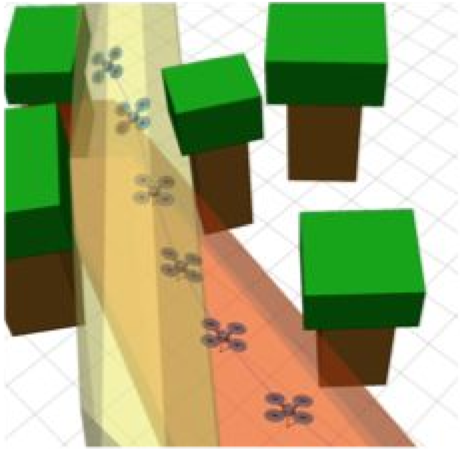
- Sufficient, not necessary (but often works!)
- Equivalent to the existence of a semidefinite matrix, subject to linear constraints on entries



What can we model with MISOS?

Collision avoidance

- Steer a quadcopter through obstacles [Deits & Tedrake, 2015]
- Want: polynomials $(p^x(t), p^y(t))$ to describe position at time $t \in [0, 1]$
  - Avoid obstacles
  - Initial/terminal conditions on trajectory
  - Possible objective: minimize “jerk” of path
- Idea:
  - Discretize time into subintervals
  - Split spatial domain into polyhedra (convex) that cover “safe region”
  - Copter must remain in one domain piece for each time interval
  - Discrete decisions: choose which region to be in at each time



Polynomial trajectory planning formulation

$$
\begin{aligned}
\min_{p} \quad & \sum_{i=1}^{N} ||p_i'''(t)||^2 \\
\text{s.t.} \quad & p_1(0) = X_0,\ p_1'(0) = X_0',\ p_1''(0) = X_0'' \\
& p_N(1) = X_f,\ p_N'(1) = X_f',\ p_N''(1) = X_f'' \\
& p_i(T_{i+1}) = p_{i+1}(T_{i+1}),\ p_i'(T_{i+1}) = p_{i+1}'(T_{i+1}),\ p_i''(T_{i+1}) = p_{i+1}''(T_{i+1}) \quad \forall i \\
& \bigvee_{r=1}^{R} [A^r p_i(t) \leq b^r] \ \text{ for } t \in [T_i, T_{i+1}] \quad \forall i
\end{aligned}
$$

- Minimize “total jerk”
- Initial/final/interstitial conditions are linear constraints on coefficients of *p*
- Need “mixed-integer programming” formulation for logical constraint

Mixed-integer sum of squares - small example

- If we can solve MISDP, we can solve MISOS for arbitrarily high degree trajectories (but computation time increases with chosen degree)
- Example: degree 4 polynomials, 9 time periods, 11 chambers

Mixed-integer sum of squares - small example

- 9 chambers, 8 time steps, degree 5 polynomials
- 58s to first feasible solution
  - 4s preprocessing
  - 54s in MIP solver
  - 8 conic subproblems

Mixed-integer sum of squares - small example

- 9 chambers, 8 time steps, degree 5 polynomials
- 651s to optimal solution
    - 60 conic subproblems

How can we model MISOS?

Modeling interfaces

SOSTOOLS and YALMIP are two MATLAB packages

- See SOSTOOLS manual and references cds.caltech.edu/sostools/

In **Julia**, we have a stack of new polynomial and SOS modeling packages (Collaborators: Benoît Legat, Robin Deits, Joey Huchette, Amelia Perry)

- MultivariatePolynomials.jl
- PolyJuMP.jl (a JuMP extension)
- SumOfSquares.jl

Can call Pajarito.jl's MISDP solver easily once problem is modeled

MultivariatePolynomials.jl

Support for multivariate polynomial construction, manipulation, etc.

```
using MultivariatePolynomials

@polyvar(x[1:2])
p = 2x[1] + 3x[1]x[2]^2 + x[2] + 3
differentiate(p, x[1]) # → 3x[2]^2 + 2
p([1,2], x) # → 19
```

PolyJuMP.jl

JuMP extension for polynomial optimization

```julia
using JuMP, PolyJuMP, MultivariatePolynomials

@polyvar(x, y)
model = Model()
Z = monomials([x,y], 0:3)
@polyvariable(model, p >= 0, Z)
@polyconstraint(model, p <= x^3, domain=(x >= 0))
```

SumOfSquares.jl

Automatically performs the SOS ➔ SDP transformation

See SOSTOOLS website to understand what happens “under the hood”

```
using JuMP, PolyJuMP, SumOfSquares

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

Polynomial trajectory planning formulation

$$
\begin{aligned}
\min_{p} \quad & \sum_{i=1}^{N} ||p_i'''(t)||^2 \\
\text{s.t.} \quad & p_1(0) = X_0,\ p_1'(0) = X_0',\ p_1''(0) = X_0'' \\
& p_N(1) = X_f,\ p_N'(1) = X_f',\ p_N''(1) = X_f'' \\
& p_i(T_{i+1}) = p_{i+1}(T_{i+1}),\ p_i'(T_{i+1}) = p_{i+1}'(T_{i+1}),\ p_i''(T_{i+1}) = p_{i+1}''(T_{i+1}) \quad \forall i \\
& \bigvee_{r=1}^{R} [A^r p_i(t) \leq b^r] \ \text{ for } t \in [T_i, T_{i+1}] \quad \forall i
\end{aligned}
$$

- Minimize “total jerk”
- Initial/final/interstitial conditions
- MIP formulation for logical constraint

Model the MISOS problem in 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
```

Model the MISOS problem in Julia

```
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))
```