

Exploiting the Structure of a Polynomial Optimization Problem

Benoît Legat

May 14th, 2023

SIAM Dynamical Systems 2023

Table of contents

Challenges

Geometric or standard form

Newton polytope

Conclusion

Challenges

Sum-of-Squares Programming

It may look simple at first...

$p(x)$ is SOS $\rightarrow p(x) = b(x)^\top Q b(x)$ where Q is PSD.

- PSD $\rightarrow$ (scaled) diagonally dominant ? *DSOS/SDSOS*
- reformulation as geometric or standard conic form ?
- what polynomial space for $b(x)$? *Newton polytope*
- which basis for $b(x)$?
- which basis for $p - b^\top Q b$? *Ill-conditioned* change of basis ?
- any group symmetry ? Can we reduce *symbolically* ?
- *Chordal* sparsity ? *Term* sparsity ? *Sign* symmetry ?
- extract roots of $p(x)$ from dual *moment matrix* ?
- Different formulation ? *Hypatia/Alphone*, Burer-Monteiro ?

$$p(x) = s_0(x) + \sum \lambda_i(x) h_i(x) + \sum s_i(x) g_i(x), \quad s_i(x) = b_i(x)^\top Q_i b_i(x)$$

- Explicit λ_i or remainder with Gröbner basis ?
- Putinar or Schüdgen certificate ?
- **what polynomial space for $b_i(x)$? *Newton polytope***
- which basis for $b_i(x)$?
- ...

Geometric or standard form

Geometric and standard form

Standard conic form SDP:

$$\begin{aligned} \min_{Q \in \mathcal{S}^n} \quad & \langle C, Q \rangle \\ \text{subject to:} \quad & \langle A_i, Q \rangle = b_i, \quad i = 1, 2, \dots, m \\ & Q \succeq 0, \end{aligned}$$

Geometric conic form SDP:

$$\begin{aligned} \max_{y \in \mathbb{R}^m} \quad & \langle b, y \rangle \\ \text{subject to:} \quad & C \succeq \sum_{i=1}^m A_i y_i \\ & y \text{ free,} \end{aligned}$$

Standard conic form

Notation

$$p(x) = \sum_{\alpha} p_{\alpha} x^{\alpha} \quad \mathcal{A}_{\alpha} = \{(\beta, \gamma) \in b^2 \mid x^{\beta} x^{\gamma} = x^{\alpha}\}$$

Standard conic form

$$\left\langle \sum_{(\beta, \gamma) \in \mathcal{A}_{\alpha}} e_{\beta} e_{\gamma}^{\top}, Q \right\rangle = p_{\alpha}, \quad \forall \alpha$$

$$Q \succeq 0$$

Geometric conic form

Notation

$$p(x) = \sum_{\alpha} p_{\alpha} x^{\alpha} \quad \mathcal{A}_{\alpha} = \{(\beta, \gamma) \in b^2 \mid x^{\beta} x^{\gamma} = x^{\alpha}\}$$

Let $(\beta_{\alpha}, \gamma_{\alpha}) \in \mathcal{A}_{\alpha}$.

Geometric conic form

$$\sum_{\alpha} p_{\alpha} e_{\beta_{\alpha}} e_{\gamma_{\alpha}}^{\top} + \sum_{(\beta, \gamma) \in \mathcal{A}_{\alpha} \setminus (\beta_{\alpha}, \gamma_{\alpha})} y_{\beta, \gamma} (e_{\beta} e_{\gamma} - e_{\beta_{\alpha}} e_{\gamma_{\alpha}}^{\top})^{\top} \geq 0$$

$y_{\beta, \gamma}$ free

Which one do I choose ?

- **Standard** conic form is good when **low** number of **variables** and **high degree**. Univariate $2d$: standard gives **linear** $m = 2d + 1$ and geometric gives **quadratic** $m = d(d + 1)/2 - (2d + 1)$.
- **Geometric** conic form is good when **high** number of **variables** and **low degree**. Quadratic: standard gives **quadratic** m and geometric gives $m = 0$.

What's the **threshold** used in practice ? None, **user** chooses !

Which one do I choose ?

- **Standard** conic form is good when **low** number of **variables** and **high degree**. Univariate 2d: standard gives **linear** $m = 2d + 1$ and geometric gives **quadratic** $m = d(d + 1)/2 - (2d + 1)$.
- **Geometric** conic form is good when **high** number of **variables** and **low degree**. Quadratic: standard gives **quadratic** m and geometric gives $m = 0$.

What's the **threshold** used in practice ? None, **user** chooses !

In YALMIP, `sosmodel` uses geometric and `solvesos` uses standard.

In `SumOfSquares.jl`, formulation matches **solver's** conic form !

Hence the importance of playing with **Dualization.jl** !

Newton polytope

Newton polytope

Notation

$$p(x) = \sum_{\alpha} p_{\alpha} x^{\alpha} \qquad \mathcal{A}_{\alpha} = \{(\beta, \gamma) \mid x^{\beta} x^{\gamma} = x^{\alpha}\}$$

Key observation 1

If $\mathcal{A}_{\alpha} = \{(\beta, \beta)\}$ and $p_{\alpha} = 0$ then $Q_{\beta, \beta} = 0$.

Key observation 2

If $Q_{\beta, \beta} = 0$ then $Q_{\beta, \gamma} = Q_{\gamma, \beta} = 0, \forall \gamma$.

Newton chip method

```
while  $\exists \alpha, \beta$  s.t.  $\mathcal{A}_\alpha = \{(\beta, \beta)\}, p_\alpha = 0$  do  
    remove  $\beta$  from  $b$   
end while
```

- Easy to implement
- Easy to generalize, e.g., [BKP16, Section 2.3]¹
- Costly : **quadratic** in length of $b \rightarrow$ best trim down b before
- **Signed** variant: change $p_\alpha = 0$ into $p_\alpha \leq 0$

[BKP16] Burgdorf, Klep, and Povh. *Optimization of polynomials in non-commuting variables*. 2016.

Polytopic reformulation of key observation 2

$\mathcal{N}(b) + \mathcal{N}(b) = 2\mathcal{N}(b)$ where $+$ is Minkowski sum.

So the result of the commutative unsigned newton chip method is:

$$\mathcal{N}(b) = \lfloor \mathcal{N}(p)/2 \rfloor$$

- Less costly because not quadratic in length of b
- How to enumerate the elements given V-rep of $\lfloor \mathcal{N}(p)/2 \rfloor$?
 - solve LP's [Lof09]²
 - Compute H-rep with CDD/LRS/PPL [Pra+04]³

[Lof09] Lofberg. "Pre-and post-processing sum-of-squares programs in practice". 2009.

[Pra+04] Prajna et al. "New developments in sum of squares optimization and SOSTOOLS". 2004.

Wait, isn't polyhedral computation not cheap ?

If polyhedral computation is costly, we might as well just do Newton chip method!

We can at least compute a cheap outer approximation of

$$\lfloor \mathcal{N}(p)/2 \rfloor$$

Outer-approximation of [Pra+04]⁴

- Min and max total degree
- Min and max degree of each variable
- Min and max degree groups (*multipartite*)

[Pra+04] Prajna et al. "New developments in sum of squares optimization and SOSTOOLS". 2004.

Support function interpretation

$2\mathcal{N}(b) \subseteq \mathcal{N}(p)$ is equivalent to

$$\forall y \in \mathbb{R}^n, 2\delta^*(y|\mathcal{N}(b)) \leq \delta^*(y|\mathcal{N}(p)).$$

Generalizes previous cases:

- Max total degree: $y = \mathbf{1}$
- Min total degree: $y = -\mathbf{1}$
- Max degree of x_i : $y = e_i$
- Min degree of x_i : $y = -e_i$
- Max degree group $\{x_{i_1}, \dots, x_{i_j}\}$, $y = e_{i_1} + \dots + e_{i_j}$
- Min degree group $\{x_{i_1}, \dots, x_{i_j}\}$, $y = -e_{i_1} - \dots - e_{i_j}$

Signed Newton chip for Putinar

Notation

$$g_0 = 1 \qquad \mathcal{A}_\alpha^i = \{(\beta, \gamma) \in b_i^2 \mid x^\beta x^\gamma = x^\alpha\}$$

Generalized key observation 1

If

$$\nexists i, \beta \neq \gamma \in b_i, \delta \in \mathcal{N}(g_i) \text{ s.t. } x^\delta x^\gamma x^\beta = x^\alpha$$

and

$$\{g_{i,\delta} \mid \exists \beta_i \in b_i \text{ s.t. } x^\delta x^{2\beta_i} = x^\alpha\}$$

has constant sign and p_α is zero or has opposite sign, then

$$Q_{i,\beta_i,\beta_i} = 0$$

Support function for Putinar

$$\forall y \in \mathbb{R}^n, \delta^*(y|\mathcal{N}(p)) = \delta^*(y|\mathcal{N}(\sum g_i s_i)).$$

Could be **cancellations** if **negative** coefficients in g_i

$$\forall y \in \mathbb{R}^n, \delta^*(y|\mathcal{N}(p)) \subseteq \delta^*(y|\sum \mathcal{N}(g_i) + \mathcal{N}(s_i)).$$

Support function on **Minkowski** sum:

$$\delta^*(y|\mathcal{S} + \mathcal{T}) = \delta^*(y|\mathcal{S}) + \delta^*(y|\mathcal{T})$$

$$\begin{aligned} \forall y \in \mathbb{R}^n, \delta^*(y|\mathcal{N}(p)) &\subseteq \sum \delta^*(y|\mathcal{N}(g_i)) + \delta^*(y|\mathcal{N}(s_i)) \\ &\subseteq \sum \delta^*(y|\mathcal{N}(g_i)) + 2\delta^*(y|\mathcal{N}(b_i)) \end{aligned}$$

If support function uniquely maximized by **diagonal** elements $Q_{i_1}, \dots, Q_{i_j}$ with same signs $g_{i_1}, \dots, g_{i_j}$. If $\delta^*(y|\mathcal{N}(p))$ **strictly** smaller or different sign, **reduce** by removing $b_{i_1}, \dots, b_{i_j}$.

Conclusion

Newton polytope vs sparsity/symmetry

- Newton polytope **reduces** the basis b
- Sparsity/symmetry try to block diagonalize b , but b is **not reduced**,

Complementary reductions, a good Newton polytope reduction is as important as sparsity/symmetry reduction.

- Always try to solve your problem with and without `Dualization.jl`
- Specify constraints with `domain` keyword to get Newton polytope-reduced bases.
- Implemented in `SumOfSquares.jl`, feedback is welcome!

Algebraic geometry

- Interface `MultivariatePolynomials.jl` with implementations
 - `DynamicPolynomials.jl`
 - `TypedPolynomials.jl`
 - `SIMDPolynomials.jl`
 - `CondensedMatterSOS.jl`
- Polynomials bases in `MultivariateBases.jl`
- Gröbner bases and algebraic system solving in `SemialgebraicSets.jl`. Interface with Buchberger and multiplication matrices by default with other implementations:
 - `HomotopyContinuation.jl`
 - `Groebner.jl`
- Extract roots from moment matrix with `MultivariateMoments.jl`

Optimization with JuMP

- Solver interface `MathOptInterface.jl` implemented by (SDP-only):
 - Nonsymmetric cone interior point: `Hypatia.jl`
 - `MosekTools.jl`
 - ADMM: `SCS.jl`, `COSMO.jl`
 - MATLAB: `SeDuMi.jl`, `SDPNAL.jl`, `SDPT3.jl`
 - C/C++: `CSDP.jl`, `SDPA.jl`, `DSDP.jl`
 - Burer-Monteiro: `SDPLR.jl`
 - BMI, NLSDP: `Penopt.jl`
- `JuMP.jl` extension for optimization with polynomials `PolyJuMP.jl`
 - Solve KKT system with `SemialgebraicSets.jl`
 - Sums of AM/GM Exponential (SAGE) :
<https://github.com/jump-dev/SumOfSquares.jl/pull/240/>
 - Sum-of-Squares with `SumOfSquares.jl`

