Algebraic Reduction and Numerical Optimization Methods for Solving Quantum Control Sum of Squares Problems

Alexander Leong

IEEE QCE 2026 - QTEM NIER

Toronto, Canada

Solving Quantum Control Problems via Mathematical Optimization

- Many state of the art methods e.g. GRAPE, Krotov are **local** optimization methods
- Global optimization approaches are being proposed
  - Bondar et al proposed QCPOP method for Global Optimal Control in 2025 using SOS programming
  - QCSOS method (Leong, 2026) builds on Bondar et al to shrink the SOS problem down using algebraic reduction methods

Fixed Time Quantum Control as Polynomial Optimization Problem

Quantum control: find $E(t) = \sum_k x_k \, t^k$ that drives a system to a target unitary U^* given control Hamiltonian, V and drift Hamiltonian, H_0

$$\begin{aligned} \text{minimize} \quad & \mathrm{Tr}\left(M^\dagger M\right) \\ \text{subject to} \quad & \partial_t U(t) = A(t)U(t) \\ & U(0) = \mathbb{1} \\ & A(t) = -i(H_0 + E(t)V) \end{aligned}$$

$$M \approx e^{\Lambda_n/2} - e^{-\Lambda_n/2} U^*$$

Result: the objective becomes a polynomial in x_k — solvable globally via Sum of Squares programming.

Magnus Expansion

Represent a time-dependent matrix IVP ODE as an infinite series expansion given by

$$\partial_t U(t) = A(t)U(t)$$

using the Magnus expansion below

$$U(T) = \lim_{n\to\infty} \exp(\Lambda_n), \qquad \Lambda_n = \sum_{k=1}^{n} \Omega_k$$

$$\Omega_1(t) = \int_0^t A(t_1)dt_1$$

$$\Omega_2(t) = \frac{1}{2!}\int_0^t d\,t_1 \int_0^{t_1} d\,t_2 [A(t_1), A(t_2)]$$

$$\Omega_3(t) = \frac{1}{3!}\int_0^t d\,t_1 \int_0^{t_1} d\,t_2 \int_0^{t_2} d\,t_3 \left(\left[A(t_1), [A(t_2), A(t_3)]\right] + \left[A(t_3), [A(t_2), A(t_1)]\right]\right)$$

Chebyshev Polynomial Expansion

$$e^{\alpha\Lambda_n} \approx J_0(\alpha)\mathbf{1} + 2\sum_{k=1}^{p} J_k(\alpha)T_k$$

where $J_k(x)$ is the Bessel function

and $\mathrm{T_k}$ is the matrix valued Chebyshev polynomial defined by

$$T_{k+1} = 2\Lambda_n T_k + T_{k-1}$$

with $\mathrm{T_0} = 1$ and $\mathrm{T_1} = \Lambda_\mathrm{n}$

SOS-SDP Problem

- This can be written as a sum of squares program solved using semidefinite programming!

$$\begin{aligned} \text{minimize} \quad & [x]_d^T Q [x]_d \\ \text{subject to} \quad & Q \in S_+^d \end{aligned}$$

- We solve for the polynomial control coefficients, Q and choose an appropriate basis, $[x]_d$
- Even for small quantum systems (e.g. 3x3), SDPs become intractable quickly!

Symmetry Reduction:
`SymbolicWedderburn.jl`

Block-diagonalize the SDP via Wedderburn decomposition over the Symmetric Group:

$$V = \bigoplus_{i=1}^{r} V_i \quad \Rightarrow \quad \text{4 independent, smaller SDPs}$$

Affine constraints become per-block:

$$A^i = \sum_j V_i^j, \qquad b^i = \alpha \cdot V_i^j$$

Reconstruct the full solution as: $\sum_i V_i \, X V_i^T$

27× variable reduction before the solver even starts — and better numerical conditioning as a bonus.

Face Reduction

ConicSolveFR.jl

Symmetry reduction alone leaves **structural degeneracy** — the solution lives on a lower-dimensional face of the cone. Interior point solvers stall without this.

Algorithm (Permenter 2017): iteratively expose the minimal face via TSVD:

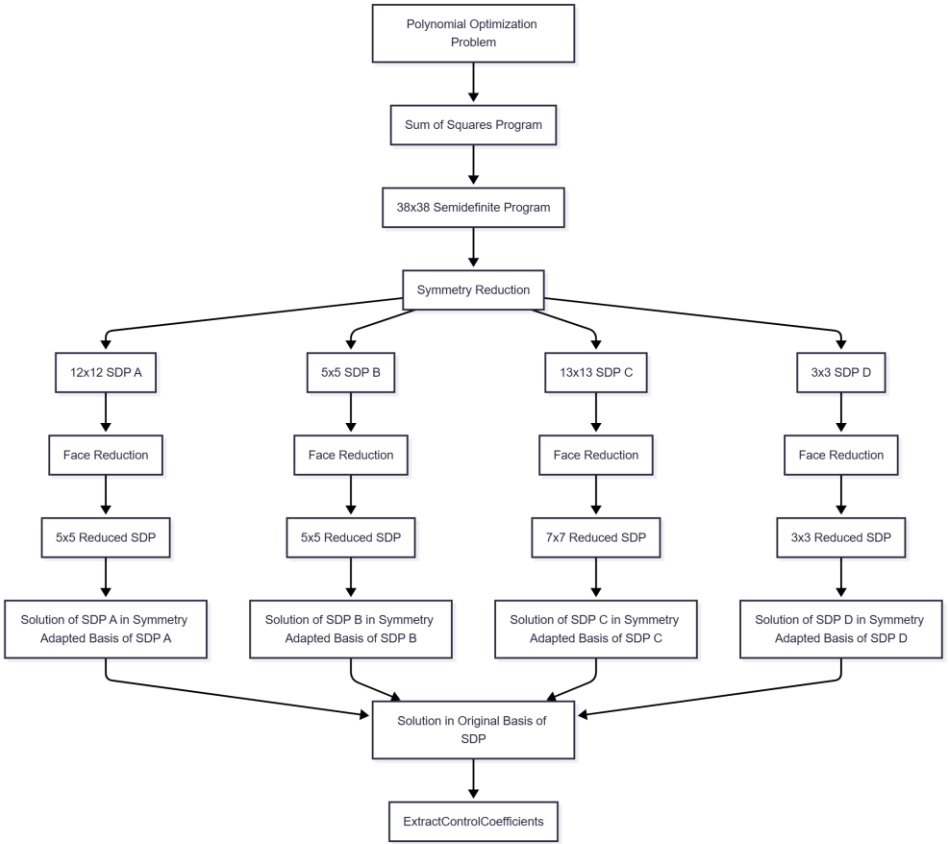
```
while SDP(*) is feasible:
    solve SDP(*) → get X̃
    update face: X̃ = U_r Λ_r U_r^T   (truncated SVD)
return minimal face F
```

Solve the reduced problem, then invert: $X_{n-1} = UX_kU^T$

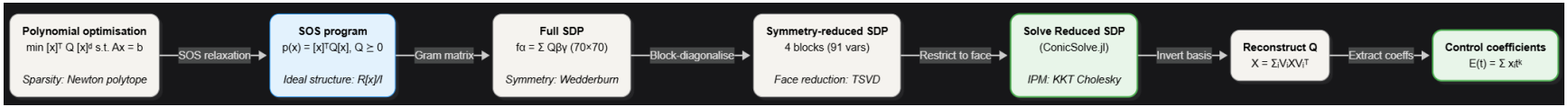
TSVD threshold η_λ is the key tuning parameter — too small: stalls; too large: diverges from original problem.

QCSOS Method Computational Graph Example

- Symmetry Reduction reduced 38x38 SDP to 4 separate SDPs, the largest SDP being 13x13
- Face reduction reduced the largest 13x13 SDP to a 7x7 SDP
- Polynomial functions of higher order can be defined over larger symmetry group to simultaneously increase number of SDPs and decrease size of each SDP
- Caveats
  - Numerical brittleness with larger decompositions
  - Basis choice and scaling matters!



The QCSOS method computational pipeline



Achieve global convergence via
Structural exploitation -> better numerical stability and efficiency -> faster convergence and tighter numerical guarantees

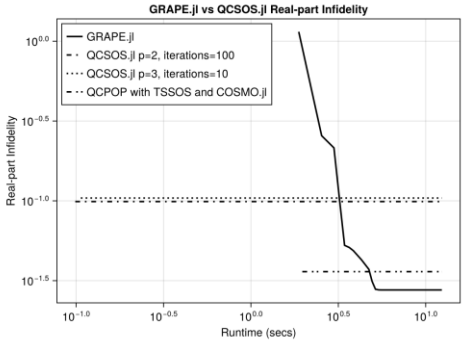
QCSOS Example

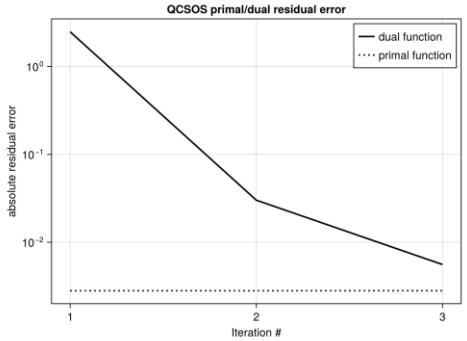
	Recommended Values
`ϵ`: absolute tolerance	1e-3 – 1e-4
`η_eps`: absolute tolerance to determine exposed face (using duality gap)	1e-2 – 1e-3
`η_lambda`: absolute tolerance to remove near redundant constraints	1e-3 – 1e-4
`p`: order of Chebyshev expansion for approximating matrix exponential	2 or 3

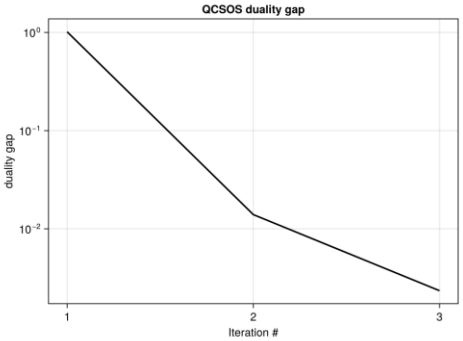
Results: QCSOS vs GRAPE.jl and QCPOP

For a Three-level system

Performance Measure	GRAPE.jl	QCPOP with TSSOS + COSMO.jl	QCSOS.jl
Infidelity	0.0277	0.036	**0.099**
Runtime (seconds)	53.2	1.97	**0.1**







QCSOS.jl is ~500× faster at ~1e-1 infidelity compared to GRAPE.jl

GRAPE plateaus at iteration ~10. QCSOS exhibits polynomial-time convergence.

Discussion & Future Work

What's working: - Layered algebraic reduction without relaxing the original problem - Composable Julia packages — drop-in preprocessing for existing solvers

Current limitations: - Numerically brittle — careful parameter tuning required - Residuals plateau at physics model ceiling (Magnus/Chebyshev truncation)

Next steps: - Alternate polynomial bases (e.g. Chebyshev) for better numerical stability - Partial face reduction to reduce overhead of multiple face-exposing SDPs - Scaling to larger quantum systems for safety-critical applications

📦 github.com/alexander-leong/QCSOS.jl

📧 toshibaalexander@gmail.com

Thanks to the Julia Community for enabling this research.