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

1st Alexander Leong
Brisbane, Australia
toshibaalexander@gmail.com

Abstract—**Solving constrained quantum control problems is difficult. Many optimization methods are based on first-order methods that get stuck in local minima and require significant computation time and resources to reach an acceptable level of accuracy. Global optimization methods such as Quantum Control via Polynomial Optimization (QCPOP) utilizing sum of squares programming have been shown to reduce computation solve time by orders of magnitude. However, such approaches can be challenging to realize due to scale and compute requirements. We propose a computational pipeline called Quantum Control Sum of Squares (QCSOS) that exploits symmetry to reduce problem size and uses face reduction to tackle numerical ill conditioning and degeneracy that stalls constrained optimization solvers. For a three-level system, results show our QCSOS method to achieve several order-of-magnitude runtime performance speedup and several fold reduction in infidelity over the GRAPE method.**

Index Terms—**Optimal control, Quantum computing, Semidefinite programming, Polynomials, Convex optimization, Numerical stability**

I. Introduction

Existing quantum control algorithms including Gradient Ascent Pulse Engineering (GRAPE), Chopped Random-Basis (CRAB) and Krotov are based on gradient search methods such as the Broyden-Fletcher-Goldfarb-Shanno (BFGS) algorithm which requires discretization before solving. Gradient search methods are prone to getting trapped in local-extrema. Other stochastic methods exist, such as those based on machine learning but are nevertheless expensive to run. Fortunately, recent developments in sum of squares programming have made the application of algebraic methods more accessible to fields such as quantum control.

Sum of Squares programs are effectively solved using constrained optimization solvers based on interior point methods. Symmetric self-dual solvers are a class of interior point methods well known for their properties such as polynomial-time convergence and achieving solutions to minimal error. These solvers are based on Euclidean-Jordan algebras that enable the efficient computation of gradients and hessians without relying on automatic differentiation (AD) based approaches.

Despite this Sum of Squares programs solved using Semidefinite programming [2] remains challenging. First, the computational blowup in memory and runtime cost as the problem size increases, second is the ill conditioning of solving polynomial optimization problems using the monomial basis, third is the numerical struggle that solvers must deal with.

Sum of squares programming is prominent in control (Lyapunov stability), robotics (trajectory optimization), as they are critical tools in the validation and verification of safety critical systems. Because of this global convergence and solution quality are becoming increasingly relevant in quantum control and quantum information sciences. Scaling quantum systems without compromising solution quality remains an ongoing challenge. This motivates the question; how can we adapt existing well established algebraic and computational tools in the quantum realm.

In this article, we will focus on the fixed-time control problem of finding the control that achieves as close as possible a given target unitary at the end of a given evolution time. The QCPOP method [5] popularized by Bondar et al., 2025 uses TSSOS.jl to solve the sum of squares (SOS) programming problem. We take a different approach to solving the SOS problem by proposing a computational pipeline based on the ideas discussed in [3] to apply face reduction on top of symmetry reduction as a preprocessing step before solving the SOS problem. This is attractive for several theoretical/practical reasons, it exploits problem structure without relaxing the original problem, and furthermore these mathematical methods can act as additional layers to extend existing solvers, i.e. packages such as SymbolicWedderburn.jl exist for exploiting algebraic structure.

An ongoing challenge in adopting these tools is a robust solver architecture. Many constrained optimization solvers must contend with solving generic problems, have mechanisms for handling degeneracy and numerical ill-conditioning. The use of matrix decompositions, equilibration, and regularization are the main methods to improve solver robustness. Our new Julia package QCSOS.jl builds on the Julia solver, ConicSolve.jl, an efficient constrained optimization solver optimized to take advantage of algebraic reduction methods, symmetry reduction, and face reduction methods.

II. Background: Fixed time quantum control

The fixed time control problem involves finding the control coefficients in $E(t)$ that result in a unitary $U(T)$ that is as close as possible to a given target unitary, U^* at time T.

Given the drift-Hamiltonian, H_0, control-Hamiltonian, V, and target unitary, U^*, we solve the optimization problem

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

where the objective function is approximated with

$$M = U(T) - U^* \approx e^{\Lambda_n/2} - e^{-\Lambda_n/2}U^* \tag{2}$$

and the control function takes the form

$$E(t) = \Sigma_{k=0}^{m-1} x_k t^k \tag{3}$$

where x_k are the coefficients to solve for.

We use a second-order Chebyshev polynomial expansion to estimate the unitary by approximating the matrix exponent

$$e^{\alpha\Lambda_n} \approx J_0(\alpha) * 1 + 2\Sigma_{k=1}^{p} J_k(\alpha)T_k \tag{4}$$

where
- $J_k(x)$ is the Bessel function
- T_k is the matrix valued Chebyshev polynomial defined via the recurrence relation $T_{k+1} = 2\Lambda_n T_k + T_{k-1}$ with $T_0 = 1$ and $T_1 = \Lambda_n$

We can approximate the solution to the initial value problem of the ODE

$$\partial_t U(t) = A(t)U(t), U(0) = 1$$

using the Magnus expansion; we use the first three terms $n = 3$, i.e.

$$\begin{aligned} \Omega_1(t) &= \int_0^t A(t_1)dt_1 \\ \Omega_2(t) &= \frac{1}{2!}\int_0^t dt_1 \int_0^{t_1} dt_2 [A(t_1), A(t_2)] \\ \Omega_3(t) &= \frac{1}{3!}\int_0^t dt_1 \int_0^{t_1} dt_2 \int_0^{t_2} dt_3 ([A(t_1), [A(t_2), A(t_3)]] \\ &\quad + [A(t_3), [A(t_2), A(t_1)]]) \end{aligned}$$

$$U(T) = \lim_{n\to\infty} \exp\left(\sum_{k=1}^{n} \Omega_k\right) \tag{5}$$

III. Background: Sum of Squares Programming

Sum of squares programs can be solved with conic solvers by lifting up the degree d polynomial to degree $2d$ polynomial expressed in quadratic form. The expression $p(x) = [x]_d^T Q[x]_d$ represents a sum-of-squares polynomial where Q are the coefficients to solve for.

The polynomial optimization problem

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

is then solved using semidefinite programming.

IV. Background: Reduction methods

A. Symmetry Reduction

Symmetry reduction plays a crucial role in reducing the size of the problem and improving numerical conditioning by finding a special projection map that takes the problem and maps it onto itself (the fixed point subspace).

Applying symmetry reduction before face reduction is what enables massive computational efficiency gains in the QCSOS method by simultaneously performing dimensional reduction and improving numerical conditioning.

The size of an SDP problem (measured by the number of variables) dominates the overall solve time. For a fourth-order polynomial (as in our sample problem) without symmetry reduction, we would have had 70 monomial terms or $70 * 71/2$ variables to solve for. "SymbolicWedderburn.jl" [4] is a package used by "ConicSolve.jl" to obtain a direct sum decomposition of the system to solve for. This results in a block diagonalization of the problem into 4 smaller SDP problems, the largest SDP $13 * 14/2$ variables to solve for.

The Wedderburn decomposition is used to give a direct-sum representation of the algebra defined by the group action over Symmetric Group d. This is expressed as

$$V = \oplus_{i=1}^{r} V_i \tag{6}$$

characterized by the set of projections $\pi_i : V \mapsto V_i$
If we denote the *j*th diagonal block of V_i as V_{ij} we have the affine constraints for the *i*th SDP given by

$$A^i = \Sigma_j V_i^j \quad ; \quad b^i = C \cdot V_i^j \tag{7}$$

where C are the coefficients of the degree $2d$ polynomial $p(x)$. The set of projections π_i is chosen so that a block-diagonalized decomposition given by (6) gives separate SDP programs where the affine constraints given by A and b is just the summation over the respective diagonalized blocks in (7).

Reconstructing the solution from the fixed point subspace to obtain the coefficients of the 70 monomial terms is just the sum over each of the direct summand elements and the solution x (in matrix form, X).

$$\Sigma_{i=1}^{n} V_i X V_i^T \tag{8}$$

B. Face Reduction

Symmetry reduction does not remove the structural degeneracy inherent in quantum control problems, which causes stalling and numerical failure of interior point method-based solvers. We apply face reduction as a numerical method to remove structural degeneracy where the solution to the problem lies on a lower dimensional subspace. The following algorithm implements Algorithm 1.1 as described by Permenter 2017.

```
Algorithm 1 Face Reduction as SDP($A$)
  Inputs: affine set $A$
  Output: a face $F$
  begin
    while (*) is feasible do
      Expose face by solving the following SDP($*$)
        minimize    $0$
        subject to  $Ax = b$
                    $x \in F$
      Update face
        $\tilde{X} = U_r \Lambda_r U_r^T$
    end while
    Return $F$
  end
```

$U_r \Lambda_r U_r^T$ is found by taking the truncated singular value decomposition of X where r is determined by a numerical threshold η_λ.

The performance of face reduction depends on:

- the solver's ability to get a near zero duality gap
- a numerical threshold on S that is as small as practical

The following optimization problem now solves for the solution in terms of the minimal face

$$\begin{array}{ll} \text{minimize} & \text{vec}(U^T \text{mat}(c) U) x \\ \text{subject to} & A = \{x \in V : Ax = b\} \end{array} \tag{9}$$

The solution is then mapped back to the original subspace by reversing the projection operations $X_n = U^T X_{n-1} U$ which is

$$X_{n-1} = U X_k U^T \tag{10}$$

V. The QCSOS Method using ConicSolve.jl

ConicSolve.jl is written to solve cone quadratic problems of the form

$$\begin{array}{ll} \text{minimize} & (1/2)x^T P x + c^T x \\ \text{subject to} & Gx + s = h \\ & Ax = b \\ & s \succeq 0 \end{array}$$

This formulation also solves cone linear programs by setting $P = 0$, $c = \text{vec}(C)$, $G = -I$ and $h = 0$ gives

$$\begin{array}{ll} \text{minimize} & c^T x \\ \text{subject to} & Ax = b \\ & x \succeq 0 \end{array}$$

$[x]_d^T Q [x]_d$ is the objective function; In conic form, we set $c = \mathbb{1}$ where x is vec(Q). The affine constraint parameters A and b are chosen such that it represents the problem in terms of the symmetry reduced fixed-point subspace and encodes the quantum control polynomial optimization problem as an SOS problem in terms of x.

We now describe the QCSOS method

```
function QCSOS Method
  1. Formulate $M^\dagger M$ using Equation 2
  2. Perform symmetry reduction using Equation 6 and 7
  3. Perform face reduction using Algorithm 1.
  4. Solve reduced problem using Equation 9
  5. Compute $X_1$ from $X_n$ using Equation 10
  Return solution in terms of Equation 8
End Function
```

A. *QCSOS Method parameters*

We list the tuning parameters here:
ϵ is the absolute tolerance used to determine the convergence of the solver based on floating-point error. Lower values such as less than 1e-6 are better; however, this can result in solver instability and premature termination.
η_ϵ is the absolute tolerance used by face reduction to determine a hyperplane that exposes a face, i.e. the maximum value of the duality gap in the SDP problem that exposes a face.
η_λ is the absolute tolerance used by the solver to eliminate redundant constraints due to structural degeneracy and floating-point error.
p the number of terms to take in the Chebyshev approximation of the matrix-exponential.

It is recommended that the solver use 64-bit floating-point with equilibration and rank-revealing qr regularization enabled. The solver uses QR and Cholesky factorization for the KKT linear system solve, which has demonstrated rapid and stable polynomial-time convergence for most problems.

B. *Dual certificates and optimality conditions*

Infeasibility is determined only to within certain numerical tolerances via Farkas-like lemma certificates. The solver determines a problem is infeasible when the following conditions are met: (i) primal and dual residuals are above a given tolerance, ϵ, i.e.

Dual residual error

$$Px + A^T y + G^T z + c > \epsilon$$

Primal residual error

$$Ax - b > \epsilon$$

(ii) cone membership of x in cone K fails, i.e. $x \notin K$, and (iii) the norm of $\|y\|$ is large, e.g., >1e-6.

K is the positive semidefinite cone, i.e., the set of all positive semidefinite matrices.

VI. Results

To measure the relative performance of the QCSOS method, we compare the optimization convergence rates and infidelities for a given three-level system with Julia's GRAPE.jl [6] implementation. To compare GRAPE.jl with QCSOS.jl we optimize against the Hilbert Schmidt inner product, as in (1) to minimize the infidelity real part functional. We report significant out-performance in the QCSOS method compared to GRAPE.jl in terms of infidelities and runtime performance. We use BenchmarkTools.jl and sufficient sample runs to account for Julia's Just-In-Time (JIT) runtime overhead for measuring runtime cost.

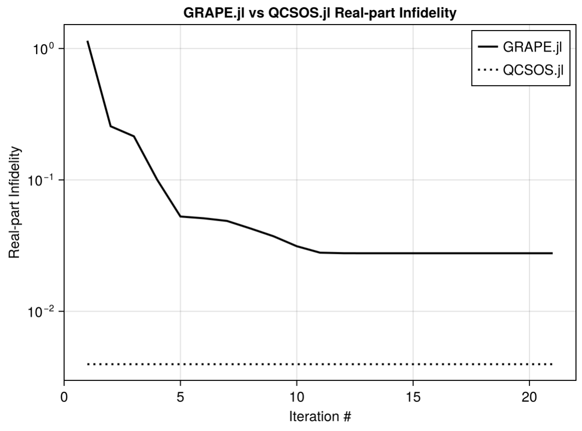

Fig. 1. Convergence rate of the real-part infidelity using GRAPE.jl

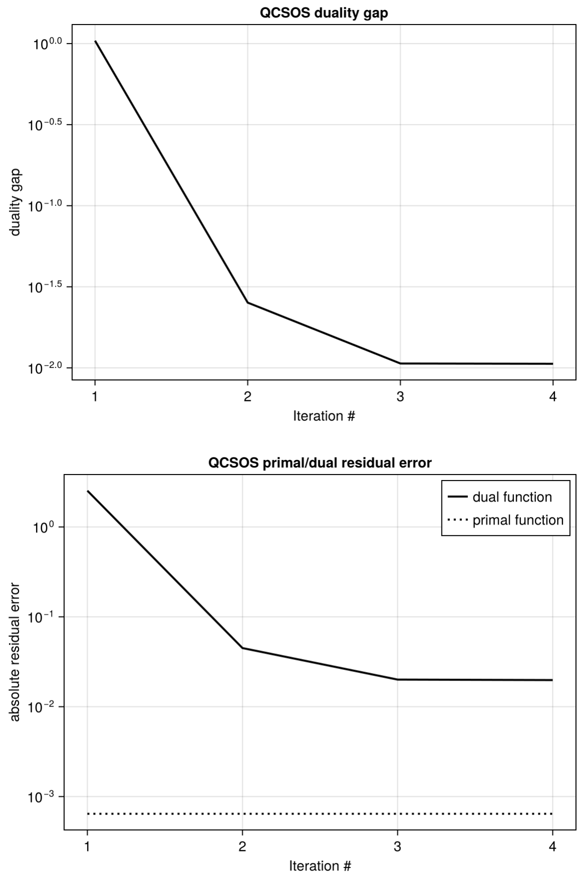

Fig. 2. Primal/dual residuals and duality gap of the reduced SDP optimization problem as determined by the face reduction step in the QCSOS method

Performance measure	GRAPE.jl	QCSOS method
Infidelity	0.0277	0.00396
Runtime cost (seconds)	53.2	0.247

TABLE I
BENCHMARK RESULTS BETWEEN GRAPE AND QCSOS METHOD

VII. Discussion

The results show that infidelities obtained by GRAPE.jl begin to plateau after 10 iterations to 0.0277. We see that after the 1st iteration of QCSOS we obtain an infidelity of 0.00396. Infidelity does not necessarily improve with the QCSOS method despite a closing duality gap since primal residual plateaued after a single iteration. This is likely attributed to the QCSOS solver hitting the numerical limits of the underlying physics model, which uses the truncated Magnus expansion and Chebyshev expansion for approximating the matrix exponential. We recommend fine-tuning the numerical parameters of the QCSOS method to obtain optimal performance. The QCSOS method is numerically brittle, and robustness requires that solvers such as ConicSolve.jl incorporate numerical safeguards for handling degeneracy. ConicSolve.jl is an symmetric self-dual Interior point methods cone solver based on CVXOPT [1] that uses a combination of equilibration and regularization methods to ensure that the optimization problem is numerically well-conditioned to find an optimal solution without hitting numerical limits.

VIII. Conclusion

The use of algebraic reduction methods such as symmetry reduction and face reduction is important for overcoming the degeneracy inherent in many quantum control problems. Interior point method solvers such as ConicSolve.jl assume that problems are numerically well conditioned in order for the solver to make progress and obtain a decent solution. Face reduction is numerically challenging and delicate, it requires solving multiple SDP problems and a threshold parameter to identify the minimal face. If the threshold parameter is set too small, the solver would likely stall and become numerically unstable, and if set too large, information may be lost, and the problem being solved diverges from the original problem.

Solving quantum control problems via global optimization methods like Sum of Squares programming remains elusive as quantum systems increasingly make their way into safety critical applications. Future work will likely include exploring alternate basis beyond the monomial basis to further improve numerical stability and robustness and more work is needed to exploit structure so partial face reduction methods can be used to further improve performance.

The code used to reproduce the results in this paper can be found here https://github.com/alexander-leong/QCSOS.jl

IX. Acknowledgement

The author wishes to thank the hosts of the Numerical Methods in Quantum Information Science QNumerics 2025 summer school for motivating the idea of using constrained optimization in solving quantum control problems.

References

[1] "The cvxopt linear and quadratic cone program solvers. "[Online]. Available: https://www.seas.ucla.edu/~vandenbe/publications/coneprog.pdf.

[2] G. Blekherman, P. A. Parrilo, and R. R. Thomas, *Semidefinite Optimization and Convex Algebraic Geometry*, G. Blekherman, P. A. Parrilo, and R. R. Thomas, Eds. Philadelphia, PA: Society for Industrial and Applied Mathematics, 2012. DOI: 10.1137/1.9781611972290. eprint: https://epubs.siam.org/doi/pdf/10.1137/1.9781611972290. [Online]. Available: https://epubs.siam.org/doi/abs/10.1137/1.9781611972290.

[3] F. Permenter, "Reduction methods in semidefinite and conic optimization," PhD Thesis, Massachusetts Institute of Technology, Cambridge, MA, 2017. [Online]. Available: https://dspace.mit.edu/bitstream/handle/1721.1/114005/1023862004-MIT.pdf.

[4] M. Kaluba, P. W. Nowak, and N. Ozawa, "Aut$(\mathbb{F}_5)$ has property (t)," *Mathematische Annalen*, vol. 375, no. 3–4, pp. 1169–1191, Aug. 2019, ISSN: 1432-1807. DOI: 10.1007/s00208-019-01874-9. [Online]. Available: http://dx.doi.org/10.1007/s00208-019-01874-9.

[5] D. I. Bondar, L. B. Gaggioli, G. Korpas, J. Marecek, J. Vala, and K. Jacobs, "Globally optimal control of quantum dynamics," *Physical Review Research*, vol. 7, no. 4, Nov. 2025, ISSN: 2643-1564. DOI: 10.1103/g4fb-xm13. [Online]. Available: http://dx.doi.org/10.1103/g4fb-xm13.

[6] M. H. Goerz, S. C. Carrasco, A. Marshall, and V. S. Malinovsky, "Grape.jl: Gradient ascent pulse engineering in julia," *Journal of Open Source Software*, vol. 10, no. 115, p. 8813, Nov. 2025, ISSN: 2475-9066. DOI: 10.21105/joss.08813. [Online]. Available: http://dx.doi.org/10.21105/joss.08813.