Complementarity by Construction: A Lie-Group Approach to Solving Quadratic Programs with Linear Complementarity Constraints

Arun L. Bishop
Robotics Institute
Carnegie Mellon University
Pittsburgh, USA
arunleob@cmu.edu

Micah Reich
Robotics Institute
Carnegie Mellon University
Pittsburgh, USA
mreich@cmu.edu

Zachary Manchester
Department of Aeronautics and Astronautics
Massachusetts Institute of Technology
Cambridge, USA
zacm@mit.edu

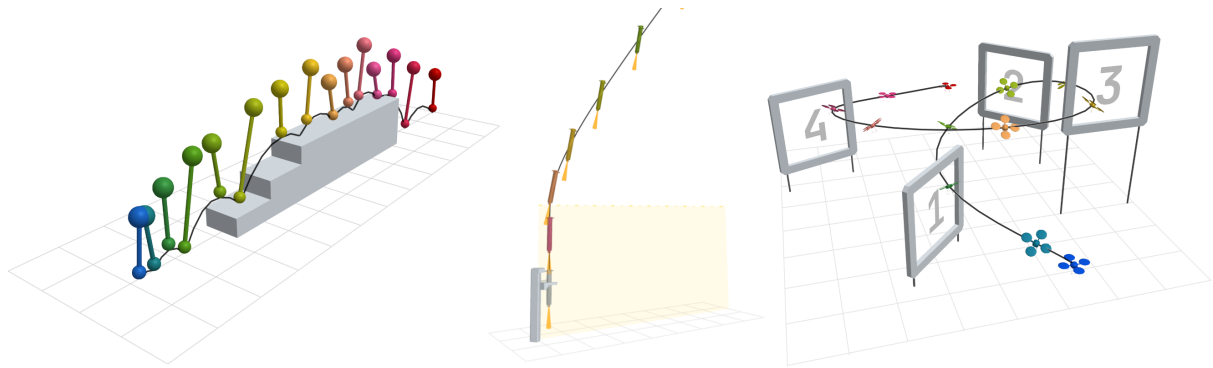

Fig. 3: LCQPs can model many robotics problems with non-smooth or switching constraints, including a hopper with contact dynamics and friction (left), a rocket catch with state-triggered safety constraints (middle), and quadrotor gate traversal with ordered completion constraints (right).

***Abstract*—Many problems in robotics require reasoning over a mix of continuous dynamics and discrete events, such as making and breaking contact in manipulation. These problems are locally well modeled by linear complementarity quadratic programs (LCQPs), which extend QPs to complementarity constraints. While expressive, LCQPs are non-convex, and few solvers exist for computing good local solutions. In this work, we observe that complementarity constraints form a Lie group under relaxation which can be used to perform on-manifold optimization. We introduce a retraction map that is numerically well behaved and use it to parameterize the constraints such that they are satisfied by construction. The resulting solver avoids many of the classical issues with complementarity constraints. We provide an open-source solver, Marble, that is implemented in C++, is competitive on a suite of benchmark problems, and solves robotics problems where existing approaches fail.**

***Index Terms*—robotics, optimization, complementarity, Lie groups**

I. INTRODUCTION

Many important planning and control problems in robotics involve optimizing over a mix of continuous and discrete elements. For example, manipulation and locomotion extend smooth motion planning to include decisions over when to make and break contact with the environment. Solving these problems globally is known to be NP-hard and combinatorially expensive, but in many cases we are satisfied with local feasible solutions, such as in model-predictive control.

It is well established that smooth nonlinear optimization problems are locally well modeled by quadratic programs (QPs) [1]. A natural choice for local approximations for non-smooth optimization is linear complementary quadratic programs (LCQPs), which extend standard QPs with linear complementarity constraints: $s \geq 0$, $t \geq 0$, and $s \circ t = 0$. LCQPs are very expressive, as shown by the example problems in Figure 3, but are non-convex and challenging to solve. Relatively little research has focused on developing fast local solvers [2], [3], [4].

In this work, we present a novel approach for solving LCQPs. Our approach solves a series of subproblems with relaxed complementarity constraints similar to [5] and is inspired by [6], which introduced a log-domain parameterization of relaxed complementarity constraints through an exponential retraction map. Our contributions are:

- A parameterization of relaxed complementarity using its Lie group structure
- A proposed retraction map that avoids the numerical ill-conditioning of the exponential map
- A solver with an open-source C++ implementation
- Comparisons to an existing solver on a benchmark suite and a variety of robotics problems showing improved performance and robustness

II. Background

A. Linear-Complementarity Quadratic Programs

Complementarity constraints take the following form over vectors $s \in \mathbb{R}^p$ and $t \in \mathbb{R}^p$ where $\circ$ denotes element-wise multiplication:

$$0 \leq s \perp t \geq 0 \iff \begin{cases} s \geq 0,\ t \geq 0 \\ s \circ t = 0 \end{cases} \tag{1}$$

The L-shaped feasible region for (1) is non-convex and non-smooth at the corner $(s,t) = (0,0)$, as shown in Figure 3.

LCQPs are optimization problems that minimize a quadratic cost subject to linear equality, inequality, and complementarity constraints:

$$\min_{z,s,t} \quad \frac{1}{2} z^T Q z + g^T z \tag{2a}$$

$$\text{subject to} \quad Az + b \geq 0 \tag{2b}$$

$$Lz + l = s \tag{2c}$$

$$Rz + r = t \tag{2d}$$

$$0 \leq s \perp t \geq 0 \tag{2e}$$

Here, $z \in \mathbb{R}^n$ is the solution vector, $A \in \mathbb{R}^{m \times n}$ and $L, R \in \mathbb{R}^{p \times n}$ are the constraint Jacobians, $b \in \mathbb{R}^m$ and $l, r \in \mathbb{R}^p$ are the constraint affine terms, and $s, t \in \mathbb{R}^p$ are complementarity slack variables.

B. Lie Groups and Lie Algebras

Lie groups are groups that also have a smooth manifold structure (i.e. they are continuous). Common examples in robotics are the 2D and 3D rotation groups. Lie groups must be closed under a multiplication operation, and must have an identity element and inverse. Importantly for us, there is a very well-developed theory of optimization on Lie groups that takes advantage of their differentiability and algebraic structure [7].

While Lie groups are not vector spaces, they can be locally linearized to enable vector-space calculations in optimization algorithms. Roughly speaking, calculations involving gradients, Jacobians, and Hessians can be performed on the Lie algebra, which is the linearization (i.e. tangent space) of a Lie group at the identity [8]. Vectors in this tangent space can then be mapped back onto the group via a retraction map [9] so that they are on the group manifold by construction.

III. Existing Works

Approaches to the general class of mathematical programs with complementarity constraints [10], [11] fall into three main categories: mixed-integer reformulations, smoothing or relaxation methods, and penalty methods.

1) Mixed-Integer Reformulation: Complementarity constraints (1) may be re-formulated using the big-M method with bounds $m_s \geq s, m_t \geq t$ and binary variable $\delta \in \{0,1\}$ [12]:

$$0 \leq s \perp t \geq 0 \iff \begin{cases} 0 \leq s \leq m_s \delta \\ 0 \leq t \leq m_t(1-\delta) \end{cases} \tag{3}$$

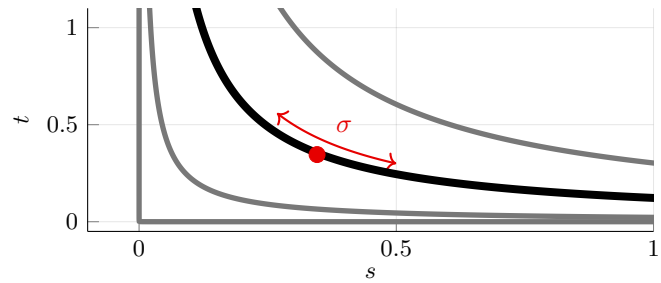

Fig. 3: Relaxed complementarity feasible sets for different relaxation parameters. σ smoothly and globally parametrizes a feasible set for a fixed relaxation (black, red dot).

These are often solved with commercial solvers like Gurobi[13] that use branch-and-bound methods, which scale combinatorially in the number of integer variables, leading to prohibitively long solution times.

2) Smoothing and Relaxation Methods: For a general review of smoothing and relaxation methods, including nonlinear program (NLP) reformulations, we refer to [14]. This work relaxes $s \circ t = 0$ to $s \circ t = \kappa$ for $\kappa \geq 0$ and performs *continuation*, solving subproblems while driving $\kappa \to 0$, similar to interior-point and barrier methods for inequality-constrained problems.

3) Penalty Methods: In penalty-based methods, constraints are incorporated through penalty terms in the cost function and continuation is used to solve subproblems for increasing penalty values. LCQPow, an open-source software package for LCQPs, includes $s \circ t = 0$ as a quadratic penalty $\rho \cdot s^T t$, leaving the inequality constraints explicit, which forms a QP subproblem. This method suffers from two weaknesses: QP subproblem failures due to the lack of a unique solution, and stalls at infeasible stationary points where the quadratic cost and penalty gradients point orthogonally into the infeasible region.

IV. Complementarity by Construction

Our approach avoids the challenges associated with complementarity constraints by performing on-manifold optimization. We relax the constraint $s \circ t = 0$ to $s \circ t = \kappa$ for $\kappa > 0$, resulting in a smooth feasible set which is a Lie group. As observed in [6], the relaxed complementarity manifold can be parameterized by a single variable, σ, in the Lie algebra as shown in Figure 3 through the exponential map as $s = \sqrt{\kappa} e^{\sigma}$, $t = \sqrt{\kappa} e^{-\sigma}$.

A. Softplus Retraction Map

In practice, we found that the exponential map suffers from numerical ill-conditioning due to unbounded gradients. To address these issues, we propose the following retraction map:

$$p_\kappa(\sigma) = \frac{\sqrt{\kappa}}{2} \left(\frac{\sigma}{\sqrt{\kappa}} + \sqrt{\left(\frac{\sigma}{\sqrt{\kappa}}\right)^2 + 4} \right) \tag{4}$$

This scaled *softplus* function is asymptotically linear outside the corner at $\sigma = 0$, quickly approaching $p_\kappa(\sigma) = \sigma$ for

positive σ and $p_\kappa(\sigma) = 0$ for negative σ. The scaling $\frac{\sigma}{\sqrt{\kappa}}$ makes the gradients bounded between 0 and 1.

V. Marble

A. Problem Formulation

Marble replaces the complementarity conditions (2c), (2d), and (2e), with the implicit softplus parameterization introduced in the previous section:

$$\min_{z,w,\sigma} \quad \frac{1}{2} z^T Q z + g^T z - \kappa \mathbf{1}^T \log(w) \tag{5a}$$

$$\text{subject to} \quad Az + b = w \tag{5b}$$

$$Lz + l = p_\kappa(\sigma) \tag{5c}$$

$$Rz + r = p_\kappa(-\sigma), \tag{5d}$$

where κ is the relaxation parameter, $p_\kappa(\sigma)$ is our retraction function (4), and inequalities are enforced through a log-barrier. For brevity in the rest of our derivation, we represent the implicit complementarity constraints as $Jz + c = h(\sigma)$:

$$J = \begin{bmatrix} L \\ R \end{bmatrix}, \quad c = \begin{bmatrix} l \\ r \end{bmatrix}, \quad h(\sigma) = \begin{bmatrix} p_\kappa(\sigma) \\ p_\kappa(-\sigma) \end{bmatrix} \tag{6}$$

We solve this problem using an augmented Lagrangian (AL) approach. The corresponding augmented Lagrangian for (5) is:

$$\begin{aligned} \mathcal{L}_A(z, w, \sigma; \kappa, \rho, \alpha, \beta) &= \frac{1}{2} z^T Q z + g^T z - \kappa \mathbf{1}^T \log(w) \\ &+ \begin{bmatrix} \alpha \\ \beta \end{bmatrix}^T \begin{bmatrix} Az + b - w \\ Jz + c - h(\sigma) \end{bmatrix} + \frac{\rho}{2} \left\| \begin{bmatrix} Az + b - w \\ Jz + c - h(\sigma) \end{bmatrix} \right\|_2^2, \end{aligned} \tag{7}$$

where α and β are Lagrange multiplier estimates. Our solver consists of an inner loop that minimizes $\mathcal{L}_A$ for fixed $\kappa, \rho, \alpha, \beta$ and an outer loop that updates $\kappa, \rho, \alpha, \beta$ as described in Section V-B.

We use Newton's method to minimize (7) in the inner loop. The KKT conditions are:

$$\frac{\partial \mathcal{L}_A}{\partial z} = Qz + g + \begin{bmatrix} A \\ J \end{bmatrix}^T \begin{bmatrix} \alpha + \rho(Az + b - w)) \\ \beta + \rho(Jz + c - h(\sigma))) \end{bmatrix} = 0 \tag{8a}$$

$$\frac{\partial \mathcal{L}_A}{\partial w} = -\left(\alpha + \rho(Az + b - w)\right) - \frac{\kappa}{w} = 0 \tag{8b}$$

$$\frac{\partial \mathcal{L}_A}{\partial \sigma} = -\left(\frac{\partial h}{\partial \sigma}\right)^T \left(\beta + \rho(Jz + c - h(\sigma))\right) = 0 \tag{8c}$$

The Jacobian $\partial h / \partial \sigma$ can be computed efficiently using properties of our retraction map (4):

$$\frac{\partial h}{\partial \sigma} = \begin{bmatrix} p'_\kappa(\sigma) \\ -p'_\kappa(-\sigma) \end{bmatrix} = \begin{bmatrix} p'_\kappa(\sigma) \\ p'_\kappa(\sigma) - I \end{bmatrix} \tag{9}$$

This system of equations becomes ill-conditioned as the penalty ρ increases, which we avoid using a primal-dual reformulation [1] by introducing dual variables $\lambda_w \in \mathbb{R}^m$ and $\lambda_\sigma \in \mathbb{R}^{2p}$:

$$\lambda_w = \alpha + \rho(Az + b - w) \tag{10}$$

$$\lambda_\sigma = \beta + \rho(Jz + c - h(\sigma)) \tag{11}$$

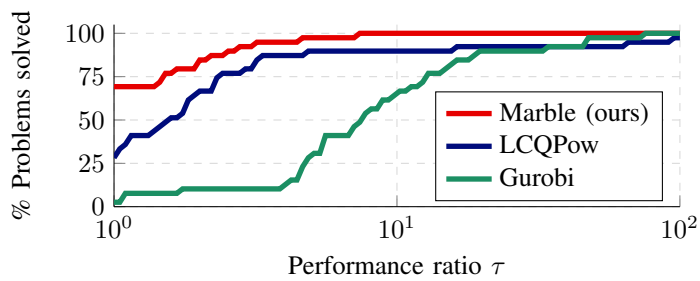

Fig. 4: Total problems solved versus performance ratio (solve time scaled by minimum time per problem) for each solver. Marble is the fastest for 69% of problems and is at most 8x slower.

Substituting λ_w into (8b) and multiplying both sides by the diagonal matrix $\text{diag}(w)$ results in the interior-point relaxed complementarity condition $\text{diag}(w)\lambda_w = -\kappa \mathbf{1}$. We satisfy this as in [6] using our retraction map p_κ with variable v.

$$w = p_\kappa(v), \;\; \lambda_w = -p_\kappa(-v) \tag{12}$$

We now arrive at the KKT residual r that is driven to zero in each inner loop:

$$\underbrace{\begin{bmatrix} Qz + g + A^T \lambda_w + J^T \lambda_\sigma \\ \lambda_w + p_\kappa(-v) \\ -\left(\frac{\partial h}{\partial \sigma}\right)^T \lambda_\sigma \\ \rho^{-1}(\alpha - \lambda_w) + Az + b - p_\kappa(v) \\ \rho^{-1}(\beta - \lambda_\sigma) + Jz + c - h(\sigma) \end{bmatrix}}_{r(z,\, v,\, \sigma,\, \lambda_w,\, \lambda_\sigma)} = 0 \tag{13}$$

B. Solver Strategy

Given the non-convexity of our problem, we adopt standard strategies from the nonlinear programming literature. The inner problem is solved using Newton's method with a filter line search for step evaluation and inertia correction as in IPOPT [1], [15].

The inner loop runs until the ∞-norm of the KKT residual defined by (13) is below a tolerance ϵ_r. In the outer loop, we first update the penalty parameter ρ. Once $\rho = \rho_{\max}$, we update κ and the Lagrange multiplier estimates. The full Marble algorithm can be found in Appendix A.

VI. Experiments

We evaluate Marble on the MacMPEC benchmark suite [16] and three robotics-specific problems, and compare the solutions to LCQPow [12], using qp-OASES as the underlying QP solver, and Gurobi [13] which provides the ground truth global solution.

A. MacMPEC Benchmark

The MacMPEC benchmarks contain problems from fields such as game theory, operations research, and structural dynamics [10]. Marble obtains feasible solutions for all problems and finds the global solution for 38 of 39 problems. It achieves a solution equal to or better than LCQPow, which only finds global solutions for 33 problems. Figure 4 shows the percent

TABLE I: Solver comparison across robotics tasks.

	Contact Dynamics $n_z = 1560,\ n_e = 780, n_i = 540,\ n_c = 540$			**Progress Constraints** $n_z = 1706,\ n_e = 760, n_i = 1942,\ n_c = 120$			**State-triggered Constraints** $n_z = 678,\ n_e = 444, n_i = 718,\ n_c = 40$		
Solver	Feasibility	Objective	Solve Time	Feasibility	Objective	Solve Time	Feasibility	Objective	Solve Time
Gurobi [13]	✓	29.20	344.56 s	✓	464.46*	898 ms	✓	380.71*	45 ms
LCQPow [12]	✓	29.21	72.38 s	✗	—	—	✗	—	—
Marble (ours)	✓	31.68	170 ms	✓	465.56	73 ms	✓	381.19	37 ms

* Indicates Gurobi achieves global optimum.

of problems solved with at most performance ratio τ for each solver, where τ is the solve time scaled by the minimum solve time across the solvers for each problem [17].

B. Robotics Benchmarks

We formulate and solve three robotics-specific problems. Code and detailed formulations of each problem are available at url. The feasibility, objective, and solve times for each solver are shown in Table I. Marble is faster for all problems and solves both problems that LCQPow fails on.

1) Contact: We solve a trajectory optimization problem for a hopper as shown in Figure 3. Contact is modeled as complementarity between the signed distance to the floor d and the normal force f_n. The hopper exerts a friction force against the ground, and complementarity constraints on the tangential velocity v_t prevent slipping, expressed as:

$$d,\ v_t^+,\ v_t^- \geq 0,\ \ v_t = v_t^+ - v_t^- \tag{14}$$

$$0 \leq v_t^+ + v_t^- + d \perp f_n \geq 0, \tag{15}$$

$$-\mu f_n \leq f_f \leq \mu f_n. \tag{16}$$

where the positive slack variables v_t^+, v_t^- upper-bound $|v_t|$ through $v_t^+ + v_t^-$. The last constraint on the friction force f_f represents the planar friction cone.

We compute the signed distance using $d = z_{\text{foot}} - h(x_{\text{foot}})$ where $h(x_{\text{foot}})$ is the height map of the stairs, represented as the sum of four shifted and scaled sign functions. Sign functions cannot be explicitly used in an LCQP, but can be implicitly represented through the KKT conditions of the linear program:

$$s = \text{sgn}(x) = \underset{-1 \leq s \leq 1}{\arg\min}(-sx). \tag{17}$$

A quadratic tracking cost on the x position is used to encourage moving across the platform at a constant speed, along with quadratic costs on the controls and control rates.

2) State-Triggered Constraints: State-triggered constraints (STCs) specify constraints $h(z) \geq 0$ that are only applied when a *trigger condition* $g(z) > 0$ is active. This can be expressed as:

$$g^+ - g^- = g(z), \quad g^+, g^- \geq 0 \tag{18}$$

$$h^+ - h^- = h_i(z),\ h^+, h^- \geq 0 \tag{19}$$

$$0 \leq g^+ \perp h^- \geq 0 \tag{20}$$

We use STCs in a trajectory optimization problem to guide a rocket into a catch tower. The rocket dynamics are planar and linearized about the upright state $\theta = 0$. When the rocket is within a specified range of the tower, both ends of the rocket must be in front of the tower with the engine pointed away from the tower to avoid plume impingement; these are written as:

$$g(y) = h_{\text{trigger}} - y \tag{21}$$

$$h_1(p, \theta) = \boldsymbol{e}_1^T \left(p + R(\theta) \begin{bmatrix} 0 \\ \ell \end{bmatrix} \right) \tag{22}$$

$$h_2(p, \theta) = \boldsymbol{e}_1^T \left(p + R(\theta) \begin{bmatrix} 0 \\ -\ell \end{bmatrix} \right) \tag{23}$$

$$h_3(\theta, \alpha) = \theta + \alpha, \tag{24}$$

where h_{trigger} is the altitude at which the constraints become active, ℓ is the half-length of the rocket, $p = [x, y]$ is the rocket position, and $R(\theta)$ is a linearized body-to-world rotation matrix. The trigger region and catch trajectory are shown in Figure 3.

3) Progress Constraints: We plan a trajectory for a quadrotor linearized about hover flying through a series of m rectangular gates. Gate completion is modeled with progress state $\gamma_k \in \mathbb{R}^m$ indicating whether a gate has been passed through, and progress control variables $\mu_k \in \mathbb{R}^m$ that update γ_k according to:

$$\gamma_k = \begin{cases} \mu_k & k = 1 \\ \gamma_{k-1} + \mu_k & 1 < k \leq N \end{cases} \tag{25}$$

To ensure that all gates are completed in order by the end of the trajectory, we enforce:

$$\gamma_N^{(j)} = 1 \qquad 1 \leq j \leq m \tag{26}$$

$$\gamma_k^{(j)} \geq \gamma_k^{(j+1)} \qquad 1 \leq j < m, \quad 1 \leq k \leq N \tag{27}$$

The progress control $\mu_k^{(j)}$ for gate j at timestep k may only be active when the drone passes through the bounds of the gate:

$$0 \leq \mu_k^{(j)} \perp \left\| R_W^j \left(r_k - o^{(j)} \right) + s_k^{(j)} \right\|_1 \geq 0 \tag{28}$$

$$-\ell^{(j)} \leq s_k^{(j)} \leq \ell^{(j)}, \tag{29}$$

where $\ell^{(j)}$ and $o^{(j)}$ are the half-extents and the position of gate j, respectively.

VII. Conclusions

In this work, we develop an approach to solving LCQPs that leverages the Lie group structure of relaxed complementarity constraints. We introduce a softplus retraction map that offers numerical advantages over the common exponential map, and demonstrate that the resulting solver is competitive across both standard benchmarks and robotics-specific problems. Our open-source solver, Marble, is implemented in C++ with Python and Julia bindings.

References

[1] J. Nocedal and S. J. Wright, *Numerical Optimization* (Springer Series in Operations Research and Financial Engineering), en. Springer New York, 2006.

[2] X. Chen and J. J. Ye, "A Class of Quadratic Programs with Linear Complementarity Constraints," en, *Set-Valued and Variational Analysis*, vol. 17, no. 2, pp. 113–133, Jun. 2009.

[3] L. Bai, J. E. Mitchell, and J.-S. Pang, "On convex quadratic programs with linear complementarity constraints," en, *Computational Optimization and Applications*, vol. 54, no. 3, pp. 517–554, Apr. 2013.

[4] D. Ralph and O. Stein, "The C-Index: A New Stability Concept for Quadratic Programs with Complementarity Constraints," *Mathematics of Operations Research*, vol. 36, no. 3, pp. 504–526, 2011.

[5] T. A. Howell, S. L. Cleac'h, K. Tracy, and Z. Manchester, *CALIPSO: A Differentiable Solver for Trajectory Optimization with Conic and Complementarity Constraints*, arXiv:2205.09255 [cs], Jan. 2023.

[6] F. Permenter, "Log-domain interior-point methods for convex quadratic programming," en, *Optimization Letters*, vol. 17, no. 7, pp. 1613–1631, Sep. 2023.

[7] P.-A. Absil, R. Mahony, and R. Sepulchre, *Optimization Algorithms on Matrix Manifolds*, en. Princeton University Press, Apr. 2009, Google-Books-ID: NSQGQeLN3NcC.

[8] J. Stillwell, *Naive Lie Theory*, en. Springer Science & Business Media, Dec. 2008, Google-Books-ID: SuR5OAgxyDIC.

[9] J. Solà, J. Deray, and D. Atchuthan, *A micro Lie theory for state estimation in robotics*, arXiv:1812.01537 [cs], Dec. 2021.

[10] Z.-Q. Luo, J.-S. Pang, and D. Ralph, *Mathematical Programs with Equilibrium Constraints*, en. Cambridge University Press, Nov. 1996, Google-Books-ID: UcjL-CgAAQBAJ.

[11] A. U. Raghunathan and L. T. Biegler, "An Interior Point Method for Mathematical Programs with Complementarity Constraints (MPCCs)," en, *SIAM Journal on Optimization*, vol. 15, no. 3, pp. 720–750, Jan. 2005.

[12] J. Hall, A. Nurkanovic, F. Messerer, and M. Diehl, "LCQPow – A Solver for Linear Complementarity Quadratic Programs," *Mathematical Programming Computation*, vol. 17, no. 1, pp. 81–109, Mar. 2025, arXiv:2211.16341 [math].

[13] Gurobi Optimization, LLC, *Gurobi Optimizer Reference Manual*, 2026.

[14] R. Fletcher* and S. Leyffer ‡, "Solving mathematical programs with complementarity constraints as nonlinear programs," en, *Optimization Methods and Software*, vol. 19, no. 1, pp. 15–40, Feb. 2004.

[15] A. Wächter and L. T. Biegler, "On the implementation of an interior-point filter line-search algorithm for large-scale nonlinear programming," en, *Mathematical Programming*, vol. 106, no. 1, pp. 25–57, Mar. 2006.

[16] S. Leyffer, *MacMPEC: AMPL collection of MPECs*.

[17] E. D. Dolan and J. J. Moré, "Benchmarking optimization software with performance profiles," en, *Mathematical Programming*, vol. 91, no. 2, pp. 201–213, Jan. 2002.

Appendix A
Marble Algorithm

Algorithm 1 Marble Algorithm

Require: Problem data, optional initial guess for z

Initialize $\rho \leftarrow \rho_0$, $\kappa \leftarrow \kappa_0, \mathcal{F} \leftarrow \varnothing$
Initialize $s \leftarrow [z, \sigma, v, \lambda_w, \lambda_\sigma]$, $[\alpha, \beta] \leftarrow [0, 0]$
for $i = 1, 2, \ldots, N_{\max}$ **do**
    Compute residual r, Jacobian ∇r from (13)
    Initialize regularizer $\delta \leftarrow 0$
    loop
        $[L, D] \leftarrow \mathrm{QDLDL}(\nabla r,\, \delta)$
        if D has incorrect inertia **then**
            $\delta \leftarrow \max(10^2,\, 10\delta)$; **continue**
        end if
        $\Delta s \leftarrow \textsc{SolveKKT}(L, D, r)$
        $s^+ \leftarrow \textsc{FilterLinesearch}(s, \Delta s)$
        if line-search succeeds **then**
            break
        else
            $\delta \leftarrow \max(10^2,\, 10\delta)$
        end if
    end loop
    $s \leftarrow s^+$
    if $\|$ (13) $\|_1 \leq \epsilon_{\mathrm{res}}$ **then**
        if constraint violations $\leq \epsilon_e,\, \epsilon_i,\, \epsilon_c$ **then**
            return s ▷ Solution found
        else if $\rho < \rho_{\max}$ **then**
            $\rho \leftarrow \min(\gamma_\rho\, \rho,\, \rho_{\max})$
        else
            $\kappa \leftarrow \max(\gamma_\kappa\, \kappa,\, \kappa_{\min})$
            $[\alpha, \beta] \leftarrow [\lambda_w, \lambda_\sigma]$
        end if
    end if
end for