COSA: Conic Active-Set Algorithm

Full Project Plan for a Conic Active-Set Solver

September 2026

Abstract

This project develops COSA, a Conic Active-Set Algorithm for second-order cone programs (SOCPs), with mean–standard-deviation portfolio optimization as the primary application. The central idea is to extend the classical active-set philosophy from polyhedral constraints to second-order cones. The portfolio problem has a polyhedral set of trading and exposure constraints, while standard deviation enters naturally through a second-order cone. The project will develop the mathematical foundations, numerical algorithm, implementation, and computational validation of this approach.

The main research question is whether conic optimization problems can be solved effectively by explicitly identifying and maintaining the active geometry of the solution, rather than relying exclusively on interior-point methods. Particular emphasis will be placed on exact cone geometry, working-set updates, conic KKT conditions, numerical linear algebra, and warm starts for sequences of related portfolio problems.

Contents

1 Project Overview

1.1 Motivation

Classical active-set methods are particularly attractive for optimization problems with a polyhedral feasible region. They maintain a working set of constraints that are believed to be active at the solution and repeatedly solve equality-constrained subproblems while adding and removing constraints.

Portfolio optimization provides a natural setting in which this philosophy is highly relevant. Typical portfolio constraints are linear:

$$\mathbf{1}^\top x = 1, \qquad x \geq 0, \qquad \ell \leq x \leq u, \qquad Ax \leq b.$$

However, the natural risk measure of interest is often standard deviation rather than variance:

$$\sigma(x) = \sqrt{x^\top \Sigma x}.$$

Standard deviation has the same physical units as expected return. Consequently, a risk-adjusted objective of the form

$$\mu^\top x - \lambda\sigma(x)$$

has a direct interpretation in terms of return and risk.

The resulting optimization problem is convex but not a quadratic program. The purpose of COSA is to investigate whether its special second-order cone structure can be incorporated directly into an active-set framework.

1.2 Primary Research Question

The central question of the project is:

> Can the classical active-set philosophy for polyhedral optimization be extended to second-order cone constraints in a way that is mathematically principled, numerically robust, and useful for structured portfolio optimization?

The intended answer should include both a mathematical algorithm and a working implementation.

1.3 Guiding Principle

The project is guided by the following principle:

Do for cones what classical active-set methods do for polyhedra.

The objective is not simply to solve an SOCP using an existing general-purpose method. The goal is to understand and exploit the geometry of the active cone directly.

2 Mathematical Problem

2.1 Mean–Standard-Deviation Portfolio Problem

Let

$$x \in \mathbb{R}^n$$

denote portfolio weights, let

$$\mu \in \mathbb{R}^n$$

denote expected returns, and let

$$\Sigma \succeq 0$$

denote the covariance matrix.

The portfolio expected return is

$$r(x) = \mu^\top x,$$

and portfolio standard deviation is

$$\sigma(x) = \sqrt{x^\top \Sigma x}.$$

We consider

$$\begin{aligned} \max_x \quad & \mu^\top x - \lambda \sqrt{x^\top \Sigma x} \\ \text{s.t.} \quad & Ax \le b, \\ & Ex = d, \end{aligned} \tag{1}$$

where $\lambda > 0$ controls the trade-off between expected return and risk.

Equivalently, we minimize

$$-\mu^\top x + \lambda \sqrt{x^\top \Sigma x}.$$

2.2 Second-Order Cone Representation

Choose a factorization

$$\Sigma = L^\top L.$$

Then

$$\sqrt{x^\top \Sigma x} = \|Lx\|_2.$$

Introduce an auxiliary variable t. The problem becomes

$$\begin{aligned} \min_{x,t} \quad & -\mu^\top x + \lambda t \\ \text{s.t.} \quad & Ax \le b, \\ & Ex = d, \\ & \|Lx\|_2 \le t. \end{aligned} \tag{2}$$

Define the second-order cone

$$\mathcal{Q}^{m+1} = \{(t, y) \in \mathbb{R}^{m+1} : \|y\|_2 \le t\}.$$

The final constraint can therefore be written as

$$(t, Lx) \in \mathcal{Q}^{m+1}.$$

Thus the complete problem is an SOCP with a polyhedral component and a second-order cone component.

2.3 Why Standard Deviation

Variance gives the quadratic objective

$$\mu^\top x - \gamma x^\top \Sigma x,$$

which is particularly convenient computationally.

However, variance has units of return squared, while expected return has units of return. Standard deviation has the same units as expected return:

$$[\sigma] = [\mu].$$

Therefore

$$\mu^\top x - \lambda\sigma(x)$$

is directly interpretable as a return-minus-risk objective.

The project deliberately accepts the additional conic structure in order to preserve this interpretation.

3 COSA Concept

3.1 Classical Active-Set Philosophy

For a polyhedral problem

$$\min_x f(x) \quad \text{s.t.} \quad Ax \le b, \quad Ex = d,$$

the active set at x_k is

$$\mathcal{A}_k = \{i : a_i^\top x_k = b_i\}.$$

The working set imposes

$$a_i^\top p = 0, \qquad i \in \mathcal{A}_k,$$

and

$$Ep = 0.$$

A direction is then computed within the tangent space of the currently active constraints.

3.2 Extension to a Cone

For COSA, the working set contains:

1. active linear inequalities;
2. equality constraints;
3. the currently active geometry of the second-order cone.

Let

$$z = (t, Lx).$$

The SOC constraint is

$$z \in \mathcal{Q}.$$

At a nonzero boundary point,

$$t = \|Lx\|_2,$$

define

$$u = \frac{Lx}{\|Lx\|_2}.$$

The tangent condition is

$$\tau - u^\top Lp = 0. \tag{3}$$

This equation is the local analogue of an active linear constraint.

3.3 Important Geometric Qualification

A second-order cone is not simply a nonlinear scalar inequality. Its geometry is naturally described through tangent cones, normal cones, and faces.

Therefore COSA will initially use the tangent representation to construct directions, but the final algorithm will be formulated in terms of the primal-dual conic KKT conditions.

This distinction is important. The project should not merely become an SQP method with an SOC constraint. The objective is a genuine conic active-set method.

4 Algorithmic Architecture

4.1 Iteration Structure

The fundamental COSA iteration is

$$\boxed{\text{working set} \to \text{direction} \to \text{feasible step} \to \text{multiplier test} \to \text{working-set update.}}$$

At iteration k:

1. Evaluate primal feasibility.
2. Identify active linear constraints.
3. Determine the SOC status.
4. Construct the current working-set equations.
5. Solve the direction problem.
6. Determine the maximum feasible step.
7. Update the primal variables.
8. Compute multipliers.
9. Add or remove constraints.
10. Check the conic KKT conditions.

4.2 Direction Problem

Let

$$f(x,t) = -\mu^\top x + \lambda t.$$

For a candidate direction (p,τ), a regularized direction problem can be written as

$$\begin{aligned} \min_{p,\tau} \quad & \nabla f^\top \begin{bmatrix} p \\ \tau \end{bmatrix} + \frac{1}{2} \begin{bmatrix} p \\ \tau \end{bmatrix}^\top H \begin{bmatrix} p \\ \tau \end{bmatrix} \\ \text{s.t.} \quad & a_i^\top p = 0, \qquad i \in \mathcal{A}_k, \\ & Ep = 0, \\ & \tau - u_k^\top L p = 0 \qquad \text{if the SOC is active.} \end{aligned} \tag{4}$$

The initial implementation may use

$$H = \rho I, \qquad \rho > 0,$$

to obtain a well-defined direction. This is not, however, the right H, and the difference is the difference between a first-order and a second-order method.

The objective is linear, so $\nabla^2 f = 0$ and it is tempting to conclude that the Lagrangian has no curvature either. It does: the *cone* is curved. Writing an active factor as the scalar constraint

$$g(z) = \|s_1\|_2 - s_0 \le 0, \qquad s = G_j z + h_j,$$

the Hessian of the Lagrangian is

$$H = \rho I + \sum_j \mu_j \, G_j^\top \begin{bmatrix} 0 & 0 \\ 0 & \dfrac{I - uu^\top}{\|s_1\|_2} \end{bmatrix} G_j, \tag{4b}$$

where μ_j is the conic multiplier of factor j — concretely, the head of w_j, since $w_j = \mu_j(1, -u)$ at a tangent factor.

Two properties make this free to use. The curvature block is positive semidefinite, because g is convex, so the direction problem stays strictly convex and uniquely solvable; and it is singular along u, because moving radially changes $\|s_1\|_2$ at a constant rate. It is also unbounded as $\|s_1\|_2 \to 0$, which is the apex announcing itself from a distance rather than only at the point where the tangent fails.

This is what makes the direction computation *primal–dual* in the sense of Phase IV: not that duals are computed alongside primals, which any active-set method does, but that the dual variable determines the primal step. The multipliers used are those of the previous iterate, which makes the scheme a fixed-point iteration rather than an implicit system; starting from zero reproduces $H = \rho I$ exactly at the first step.

4.3 KKT System

The direction problem leads to a linear KKT system of the form

$$\begin{bmatrix} H & W_k^\top \\ W_k & 0 \end{bmatrix} \begin{bmatrix} d_k \\ \nu_k \end{bmatrix} = - \begin{bmatrix} g_k \\ 0 \end{bmatrix},$$

where

$$d_k = \begin{bmatrix} p_k \\ \tau_k \end{bmatrix}$$

and W_k contains the current working-set equations.

This creates a direct connection with classical active-set QP implementations.

5 Exact SOC Step Calculation

5.1 Cone Feasibility Along a Direction

Given a feasible point (x, t) and a direction (p, τ), we require

$$\|L(x + \alpha p)\|_2 \le t + \alpha\tau. \tag{5}$$

Define

$$r = Lx, \qquad q = Lp.$$

Then

$$\|r + \alpha q\|_2 \le t + \alpha\tau.$$

Provided the right-hand side is nonnegative, squaring gives

$$\big(\|q\|_2^2 - \tau^2\big)\alpha^2 + 2(r^\top q - t\tau)\alpha + \|r\|_2^2 - t^2 \le 0. \tag{6}$$

Thus the SOC feasibility condition reduces to a scalar quadratic inequality.

The leading coefficient is the *Lorentz form* of the direction, $\|q\|_2^2 - \tau^2$, and not $\|q\|_2^2$. An earlier draft of this document printed the latter. The two agree only when $\tau = 0$, which is the one case eq. (7) never takes: the objective charges λ per unit of t, so every improving direction moves τ.

The correction is not cosmetic. Along a direction running down the cone's boundary towards the apex the printed form admits a step of zero where the true maximum step is $t/(-\tau)$, so a solver using it refuses to travel along the boundary at all. The correction also turns the root selection from one case into three, because the leading coefficient may now vanish or be negative:

- $\|q\|_2^2 > \tau^2$: an upward parabola, and the step is bounded by its upper root;
- $\|q\|_2^2 = \tau^2$: the quadratic degenerates to a linear inequality;
- $\|q\|_2^2 < \tau^2$: a downward parabola, and the feasible set is an interval or a half-line, selected by the sign at its midpoint.

A further cap at $t/(-\tau)$ is required when $\tau < 0$, since squaring is only valid while the right-hand side is nonnegative.

5.2 Maximum Feasible Step

The COSA implementation will compute the admissible interval in α from (6), together with the step bounds induced by the linear inequalities.

For each inactive linear inequality,

$$a_i^\top (x + \alpha p) \le b_i$$

gives the standard bound

$$\alpha \le \frac{b_i - a_i^\top x}{a_i^\top p}$$

whenever $a_i^\top p > 0$.

The feasible step is therefore determined by the intersection of:

- the linear feasible interval;
- the SOC feasible interval;
- any explicit step bounds.

This is the conic analogue of the classical active-set ratio test.

6 Conic KKT Conditions

6.1 Primal Problem

The primal SOCP is

$$\begin{aligned} \min_{x,t} \quad & -\mu^\top x + \lambda t \\ \text{s.t.} \quad & Ax \le b, \\ & Ex = d, \\ & (t, Lx) \in \mathcal{Q}. \end{aligned} \tag{7}$$

Introduce multipliers $y \ge 0$ for the linear inequalities and ν for the equalities. Let w denote the dual variable associated with the SOC.

6.2 Stationarity

The conic stationarity equations have the form

$$-\mu + A^\top y + E^\top \nu + L^\top w = 0,$$

together with the stationarity condition associated with t.

The exact sign convention will be fixed in the implementation and must be used consistently throughout the derivation and code.

6.3 Primal and Dual Cone Feasibility

The primal SOC condition is

$$(t, Lx) \in \mathcal{Q}.$$

The dual SOC variable must satisfy

$$w_{\text{soc}} \in \mathcal{Q},$$

under the chosen standard representation.

6.4 Complementarity

For the linear constraints,

$$y_i(a_i^\top x - b_i) = 0.$$

For the SOC, complementarity is expressed through the cone complementarity relation between the primal and dual cone variables.

The final implementation will use residuals for:

- primal feasibility;
- dual feasibility;
- stationarity;
- linear complementarity;
- SOC complementarity.

These residuals define the primary termination criterion.

7 Active-Set Logic

7.1 Linear Constraint Activation

If an inactive inequality reaches its boundary,

$$a_i^\top x = b_i,$$

the constraint is added to the working set.

The corresponding multiplier is subsequently computed from the KKT system.

7.2 Linear Constraint Deactivation

If an active inequality has a multiplier violating the required sign, the constraint is a candidate for removal.

The implementation will initially follow the classical rule of removing the most strongly violating multiplier, subject to numerical tolerances.

7.3 SOC Activation

The SOC becomes geometrically active when

$$t - \|Lx\|_2$$

is sufficiently small.

A tolerance such as

$$t - \|Lx\|_2 \leq \varepsilon_{\text{act}}$$

will be used.

However, geometric activity alone is not sufficient to establish optimality. The conic multiplier must also be considered.

7.4 SOC Deactivation

A key research component is determining when the SOC should be removed from the working geometry.

Unlike a scalar inequality, the SOC has richer boundary structure. The final implementation will use the conic KKT multiplier and normal cone conditions to determine whether the cone contributes a genuine active normal at the candidate solution.

8 Degeneracy and Special Cases

8.1 The Point $Lx = 0$

At

$$Lx = 0,$$

the expression

$$\frac{Lx}{\|Lx\|_2}$$

is undefined.

This case must therefore be treated separately.

The SOC boundary at the apex

$$(t, Lx) = (0, 0)$$

has a different tangent and normal geometry from a smooth nonzero boundary point.

The implementation will initially handle this case using the exact SOC membership and normal-cone conditions rather than a tangent hyperplane.

8.2 Nearly Active Cones

Numerical tolerances can lead to oscillation between active and inactive states.

COSA will therefore use separate activation and deactivation tolerances, analogous to hysteresis:

$$\varepsilon_{\text{on}} < \varepsilon_{\text{off}}.$$

8.3 Degenerate Working Sets

The working-set KKT matrix may become singular when active constraints are linearly dependent.

The implementation will detect rank deficiency and investigate:

- QR-based rank detection;
- null-space methods;
- regularization;
- dependent-constraint removal.

9 Development Phases

9.1 Phase I: Polyhedral Baseline

Implement a conventional active-set solver for

$$\min_x \; c^\top x + \frac{1}{2} x^\top H x$$

subject to

$$Ax \leq b, \qquad Ex = d.$$

Required components:

- feasible initialization;
- active-set representation;
- direction computation;
- KKT solve;
- multiplier calculation;
- constraint addition;
- constraint deletion;
- ratio test;
- termination checks.

This implementation will be deliberately simple and will provide the baseline for debugging COSA.

9.2 Phase II: SOC Geometry Library

Implement independent routines for:

- SOC membership;
- SOC boundary detection;
- SOC tangent calculation;
- SOC normal calculation;

- SOC step feasibility;
- exact SOC step interval;
- apex handling.

These routines should be tested independently using synthetic directions and analytically known cone intersections.

9.3 Phase III: COSA Prototype

Combine the polyhedral active-set method with the SOC geometry.

The prototype should solve the portfolio problem using:

1. a working set of linear constraints;
2. a tangent representation of the SOC;
3. exact SOC step lengths;
4. multiplier-based active-set updates.

At this stage, correctness is more important than speed.

9.4 Phase IV: Full Conic KKT Method

Replace the preliminary tangent-only treatment with a full primal-dual conic formulation.

The goal is to obtain a method whose stopping criterion and working-set logic are derived directly from the SOCP KKT conditions.

9.5 Phase V: Numerical Linear Algebra

Optimize the solution of the repeated KKT systems.

Investigate:

- sparse $\mathrm{LDL}^\top$ factorization;
- QR factorization;
- null-space methods;
- range-space methods;
- factorization reuse;
- rank-one updates;
- scaling;
- regularization.

9.6 Phase VI: Warm Starts

Implement warm starts as a first-class feature.

Given a solution (x_k, t_k) for one problem, initialize the next problem using:

- the previous primal solution;
- the previous working set;
- previous multipliers where appropriate;
- previous KKT factorizations when possible.

This is expected to be one of the strongest potential advantages of COSA.

10 Portfolio Test Problems

10.1 Basic Portfolio

The first model will be

$$\begin{aligned} \min_{x,t} \quad & -\mu^\top x + \lambda t \\ \text{s.t.} \quad & \mathbf{1}^\top x = 1, \\ & x \geq 0, \\ & \|Lx\|_2 \leq t. \end{aligned} \tag{8}$$

This provides the simplest realistic test.

10.2 Box Constraints

Introduce

$$\ell \leq x \leq u.$$

These constraints are particularly useful for active-set testing because many portfolio bounds are expected to become active.

10.3 Sector Constraints

Add constraints of the form

$$A_{\text{sector}} x \leq b_{\text{sector}}.$$

These create nontrivial combinations of active constraints.

10.4 Factor Exposure Constraints

Add

$$\ell_f \leq Fx \leq u_f.$$

This provides a useful test of correlated active constraints.

10.5 Turnover Constraints

For a previous portfolio x^{old}, turnover can be represented using auxiliary variables and linear inequalities.

These problems are particularly interesting for warm starts because successive portfolio rebalancing problems are naturally related.

11 Efficient Frontier Experiments

One of the principal numerical experiments will solve the sequence

$$\min_x \left\{ -\mu^\top x + \lambda \sigma(x) \right\}$$

for

$$\lambda_1, \lambda_2, \ldots, \lambda_N.$$

The sequence generates points on the risk-return frontier.

For consecutive values

$$\lambda_k, \lambda_{k+1},$$

the optimal portfolios and active constraints are expected to be related.

This allows us to test whether COSA benefits from:

$$\text{solution}_k \longrightarrow \text{warm start for solution}_{k+1}.$$

The main quantities measured will be:

- number of active-set iterations;
- number of constraints added;
- number of constraints removed;
- number of KKT factorizations;
- total runtime;
- KKT residual;
- number of iterations saved by warm starts.

12 Benchmarking

12.1 Reference Solvers

COSA will be compared with established commercial conic optimization solvers, initially including MOSEK and Gurobi.

These solvers provide reference solutions for correctness and performance.

The comparison should distinguish between:

- cold-start solves;
- warm-start solves;
- individual large problems;
- sequences of related problems.

12.2 Accuracy Metrics

For each solution, record:

- primal residual;
- dual residual;
- stationarity residual;
- complementarity residual;
- objective value;
- standard deviation;
- expected return.

12.3 Performance Metrics

Record:

- wall-clock time;
- number of iterations;
- number of KKT solves;
- number of active-set changes;
- factorization time;
- memory usage where relevant.

12.4 Robustness Tests

Construct instances with:

- nearly redundant constraints;
- highly correlated assets;
- ill-conditioned covariance matrices;
- nearly active SOC constraints;
- degenerate optimal solutions;
- many simultaneously active portfolio bounds.

The purpose is to identify failure modes before optimizing performance.

13 Numerical Linear Algebra Strategy

The active-set approach repeatedly solves systems of the form

$$\begin{bmatrix} H & W^\top \\ W & 0 \end{bmatrix} \begin{bmatrix} d \\ \nu \end{bmatrix} = \begin{bmatrix} r \\ 0 \end{bmatrix}.$$

Since the working set changes by only a few constraints per iteration, it may be possible to exploit substantial structure.

13.1 Initial Implementation

The first implementation will refactor the KKT matrix at every iteration.

This minimizes implementation complexity and provides a reliable reference.

13.2 Factorization Reuse

Once the algorithm is correct, investigate updates when:

- one constraint is added;
- one constraint is removed;
- the SOC tangent changes.

The SOC tangent changes continuously with x, so this part is more subtle than ordinary linear constraint updates.

13.3 Scaling

Scaling will be investigated for:

- portfolio variables;
- covariance matrices;
- linear constraints;
- expected returns;
- SOC variables.

Good scaling is particularly important because the SOC couples t and Lx.

14 Convergence and Correctness

The project will distinguish three levels of validation.

14.1 Level 1: Feasibility

Every accepted iterate must satisfy, within tolerance,

$$Ax \leq b, \qquad Ex = d, \qquad \|Lx\|_2 \leq t.$$

14.2 Level 2: Stationarity

The computed multipliers must satisfy the stationarity equations to within a prescribed tolerance.

14.3 Level 3: Conic KKT Optimality

The final solution must satisfy the full conic KKT conditions.

Because the problem is convex, satisfaction of the KKT conditions provides a certificate of global optimality under the usual constraint qualification assumptions.

14.4 Assumptions and Constraint Qualifications

This is Deliverable 1, and it is stated here rather than deferred. Throughout, $z = (x, t)$ and the instance is eq. (7).

Standing assumptions. These are required for the problem to be well posed and are checked at construction.

1. $\Sigma \succeq 0$. Positive *semi*definiteness suffices: L is obtained by eigendecomposition and a rank-deficient Σ simply yields a shorter L, which makes the cone smaller rather than worse conditioned. Rank deficiency is not a degenerate case here and is not treated as one.

2. $\lambda > 0$. This is what makes the reduction from eq. (1) to eq. (2) exact: it is what pushes t down onto $t = \|Lx\|_2$, so the two problems have the same solution set in x. At $\lambda = 0$ the reduction fails and the problem is a linear program.

3. The feasible set is nonempty. Detected rather than assumed: an elastic Phase I minimizes a uniform relaxation of the inequalities and reports infeasibility when its optimal relaxation is positive.

4. The objective is bounded below on the feasible set. Also detected rather than assumed: an unbounded ray is reported as such by the ratio test finding no blocking constraint.

Constraint qualification. The KKT conditions of Section 6 are necessary at a minimizer under a constraint qualification, and for this problem the relevant one is Slater's:

> There exists a strictly feasible z — one with $Az < b$, $Ez = d$ and $\|Lx\|_2 < t$.

Slater's condition holds for every instance in which the linear constraints admit a relative interior point, because the conic constraint can always be made strict by *raising* t: the variable t appears in exactly two places in eq. (7), the cone's head row and the objective, so increasing it violates nothing. This is a property of the mean–standard-deviation formulation and not of SOCPs in general; a formulation in which t carried an upper bound would need Slater's condition assumed rather than derived.

Because the problem is convex and Slater's condition holds, satisfaction of the conic KKT conditions is a certificate of *global* optimality, not merely of stationarity.

The generic case. Generically, at a minimizer:

1. the active linear rows together with E have full row rank (LICQ), so the multipliers y and ν are unique;

2. $Lx \neq 0$, so the cone's boundary is smooth at the solution, the outward normal $u = Lx/\|Lx\|_2$ is defined, and the tangent representation of Section 3.2 is the correct local description;

3. strict complementarity holds: an active constraint has a strictly positive multiplier, and an inactive one has a zero multiplier with strictly positive slack.

Under these three the working set at the solution is unique, the active-set iteration is locally exact, and warm starting a nearby problem transfers a working set that is correct rather than merely close.

Degenerate cases, and what each one costs. Each of the three can fail independently, and the algorithm's response to each is different.

1. *LICQ fails* — the active rows are linearly dependent. The direction d remains unique, because $H \succ 0$ and $Wd = 0$ does not care about redundancy; the multipliers do not. Section 8.3's dependent-constraint removal repairs the working set exactly where the dependent rows may be dropped, and regularization answers a nearby question where they may not. Removal is tried first because it leaves the arithmetic exact.

2. $Lx = 0$ *at the solution* — the apex. The tangent has no meaning there; the tangent cone of $\mathcal{Q}$ becomes $\mathcal{Q}$ itself and its normal cone becomes $-\mathcal{Q}$, so the dual test becomes a cone *membership* rather than a sign. This is not exotic: a rank-k covariance over $n \gg k$ assets has a large null space, the minimum-risk portfolio has risk exactly zero, and $\lambda > 0$ takes it. On the tested factor-model families the apex is consistently *cheaper* than the ordinary case rather than harder.

   On eq. (7), however, an apex that is *not* justified by its multiplier cannot be released. Dropping the factor forces $\rho\,\tau = -\lambda$, hence $\tau < 0$, and the resulting direction is conically infeasible by arithmetic rather than by tolerance. The implementation reports this as a distinct terminal status; it occurs on about 1.5% of randomly generated instances and is the one place where the working-set concept of Section 3.2 is genuinely insufficient.

3. *Strict complementarity fails* — a constraint is active with a zero multiplier. The solution is still optimal and still certified, but the working set at it is not unique. The cost is borne by warm starts: Section 11's experiment shows warm starting saving 18–55% of iterations at points whose working set transfers and *costing* 17–32% at points where it must be corrected, and it is exactly the non-unique working sets that change between neighbouring problems.

One assumption that *is* required, and was not stated. The residual criterion of Section 6 certifies global optimality only where the instance is well enough scaled that a *relative* residual bounds the objective error. Section 15.6 exhibits an instance where it does not: all five residuals inside 10^{-11} at a point 3.4% from the optimum, because the normalization divides by $\|c\|_\infty \approx 2 \times 10^6$ and the conditioning amplifies what is left. Stating this is Deliverable 1's job and it is the one assumption the plan did not anticipate needing.

One assumption that is not required. No assumption is made about the conditioning of Σ. The tangent representation enters the KKT system as a *single row*, and a single row has no spectrum, so the assembled system does not inherit $\kappa(\Sigma)$: on an instance with $\kappa(\Sigma) \approx 10^{10}$ the KKT matrix is conditioned around 10. This is a genuine structural difference from interior-point methods, where the barrier's Hessian carries Σ into the system directly.

15 Software Architecture

The implementation will be modular.

A proposed structure is:

```
cosa/
    problem/
        socp.py
        portfolio.py
    geometry/
        soc.py
```

```
        tangent.py
        step.py
    active_set/
        working_set.py
        multipliers.py
        updates.py
    linear_algebra/
        kkt.py
        factorization.py
        scaling.py
    solver/
        cosa.py
        initialization.py
        termination.py
    experiments/
        portfolio.py
        frontier.py
        benchmarks.py
    tests/
        test_soc.py
        test_step.py
        test_kkt.py
        test_active_set.py
        test_portfolio.py
```

The exact language and numerical libraries can be selected at the start of implementation. Python with NumPy/SciPy is suitable for the initial prototype; a lower-level implementation can be considered after the algorithm is stable.

16 Testing Strategy

16.1 Unit Tests

Every geometric primitive should have independent tests.

Examples include:

- points inside the SOC;
- points outside the SOC;
- boundary points;
- tangent directions;
- directions entering the cone;
- directions leaving the cone;
- exact step roots.

16.2 Analytical Problems

Construct small SOCPs whose solutions can be derived analytically.

These problems will be used to validate:

- primal solutions;
- active sets;
- multipliers;
- SOC activity;
- KKT residuals.

16.3 Cross-Solver Tests

For every randomly generated test problem, compare COSA against a reference solver.

The objective values should agree to within the prescribed numerical tolerance.

17 Results

Every number in this section is generated by `python -m cosa.experiments` and is committed alongside the prose that quotes it. Seeds are arguments with recorded defaults, so the command with no arguments reproduces the tables below; the two artifacts that report wall-clock time and a platform are not byte-reproducible and say so.

17.1 Accuracy

Across all four benchmark modes of Section 12 and both draws, every generated problem's objective agrees with the reference solver to within 10^{-6} relative — 18 of 18 comparisons, with observed gaps between 10^{-11} and 10^{-9}. The five conic KKT residuals of Section 6 at termination lie between 10^{-10} and 10^{-9}.

Success Criterion 5 asks for that agreement on *every* generated problem, and across the wider set of instance families it does not hold. Twelve of thirteen families agree to 10^{-9} or better; the thirteenth does not, and Section 15.6 is that result. It is the paper's most important negative finding and it is a statement about the stopping criterion rather than about the arithmetic.

17.2 Robustness

Thirteen instance families, three seeds each, twenty assets: thirty-six solve and three do not. Outcomes are classified into five rather than reported as a status string, and the classification is not permitted to be self-certified — a verdict of *solved* requires that the loop report optimality, that Section 6's residuals confirm it, *and* that a reference solver agree about the objective:

verdict	count
solved (optimal, certified, and the reference agrees)	36
wrong (optimal and certified, disagreeing with the reference)	**3**
loose (optimal, residuals do not quite agree)	0
unchecked (no reference was available)	0
diagnosed (stopped, and named what happened)	0
undiagnosed (the iteration limit, which names nothing)	0

The *wrong* category is the one worth having and it did not exist until the study stopped certifying itself. A diagnosed stop is honest and an undiagnosed one is at least visible; a wrong answer looks like success.

Notable individual results. The factor-model families, whose optima lie at the apex, are consistently *faster* than the basic family rather than harder: 179 iterations against 253 across three seeds, a saving of 29%. The degenerate-optimum family, whose active rows are linearly

dependent at the solution, solves in four iterations to a residual at rounding level, because Section 8.3's dependent-row removal is exact rather than approximate. The ill-conditioned family, with $\kappa(\Sigma) \approx 10^{10}$, solves without special handling, for the single-row reason given in Section 14.4.

No M7 mitigation changes any outcome. Section 8.3's regularization is never reached, because dependent-row removal always succeeds first; Section 13.2's reuse is a cost policy and computes the same direction by a different route; and Section 13.3's Ruiz equilibration costs between 30% and 80% more iterations on every family and rescues none.

17.3 A Certificate That Certifies the Wrong Answer

One family fails, and it fails in the most instructive way available.

The instance is deliberately badly scaled: $\|A\|_\infty \approx 10^{-4}$ while $\|G\|_\infty$ and $\|c\|_\infty$ are around 10^6. COSA terminates on it reporting *optimal*, with all five of Section 6's residuals between 10^{-11} and 10^{-15}, at the point whose objective is -0.00524. The reference solver returns -0.03956 — and that point is feasible for COSA's *own* feasibility test to 10^{-11}, with a strictly better objective. The direction between the two is a feasible descent direction with derivative -0.0343. No tolerance argument survives that: a better feasible point demonstrably exists.

The residual is not lying. The stationarity residual there is 1.93×10^{-5} in absolute terms; Section 14.2 reports it relative to the objective's scale, dividing by $\max(1, \|c\|_\infty) = 2 \times 10^6$, which prints 9.7×10^{-12}. A convex problem cannot have an exactly satisfied KKT system at a suboptimal point and this one does not — the residual is real, and it is simply small enough relative to data of size 10^6 that a *relative* certificate cannot distinguish it from zero, while the conditioning amplifies it into a 3.4% objective error.

The certificate is relatively satisfied and the answer is wrong, and neither of those is a mistake in the other. That is the finding. It says that the residual criterion of Section 6, which Success Criterion 2 asks to be "mathematically meaningful", is meaningful and not *sufficient*: on an instance whose data spans fourteen orders of magnitude a relative KKT residual admits an answer that is percent-wrong. An absolute floor, or a normalization that does not divide by $\|c\|_\infty$, is what the criterion is missing, and stating that is Deliverable 1's business rather than a defect to be patched quietly.

Two things this is *not*. It is not a scaling problem: equilibrated, the same instance lands at -0.00540 against the same reference, a gap indistinguishable from the unscaled solve's. Scaling changes the appearance of the residual and never the answer. And it is not the earlier diagnosis, which was that the family stalls — it does now run to completion, because the routine that repairs conic feasibility by setting $t = \|Lx\|_2$ had required the cone's head row to select its variable with a coefficient of exactly one, which no rescaled instance satisfies, so the retraction was silently unavailable and a boundary iterate could not move at all.

That earlier diagnosis had a clean before-and-after table supporting it: the family stalled, equilibration produced an apparent optimum, and the conclusion drawn was conditioning. The ablation was sound and what it measured was a coincidence. **A diagnosed failure is a hypothesis**, and a study that certifies itself will confirm whichever hypothesis it started with.

17.4 Warm starts

This is the project's central hypothesis and the result is a conditional, not a number.

Along the frontier of Section 11, warm starting saves work at the points where the carried working set turns out to be correct and *costs* work at the points where the iteration must correct it:

	8 assets	10 assets	12 assets
working set transferred	+55%	+18%	+18%
working set corrected	−27%	−17%	−32%
overall	+44%	+9%	−7%

The per-group figures are nearly identical across instances; what differs is the *mix*. Correcting a belief is more expensive than acquiring one: a cold solve discovers the active set on the way in, while a warm one must first undo a wrong answer and then discover it anyway. Reporting a single headline number would have hidden this.

The factorization cache is carried across the sequence as well, so a frontier of twenty-four points requires two factorizations in total rather than twenty-four.

17.5 Cost

Section 13.2's factorization reuse behaves as predicted on its own metric and not on wall clock. The share of KKT solves requiring a fresh factorization falls from 98.9% to 1.4% across the instance families and from 98.7% to 1.5% across the randomized sweep.

Wall clock does not follow, and the reason is structural. A column update against a full $n \times n$ orthogonal factor costs $O(n^2)$, while a fresh QR of an $n \times m$ matrix costs $O(nm^2)$, so updating wins only once m^2 exceeds n. The classical argument for factorization updating assumes a working set comparable in size to the problem; an active-set method on a portfolio runs with m well below n. Measured on the box-constrained family, updating is $0.98\times$ the refactorizing cost at $n = 300$ and $1.12\times$ at $n = 500$.

Section 4.2's correction to H is the largest single improvement measured. Replacing $H = \rho I$ with the Lagrangian Hessian of eq. (4b):

	$H = \rho I$	H of eq. (4b)
eleven families, 66 solves	9738 iterations	4691
reaching an optimum	62 of 66	66 of 66
randomized sweep, 200 instances	62631 iterations	19386
reaching an optimum	162 of 200	192 of 200

17.6 Comparison with an interior-point solver

Against CVXPY with Clarabel, on the same instances, COSA is between three and thirty times *slower* in wall clock — and the comparison includes CVXPY's modelling overhead, so it is if anything generous to COSA.

This is reported rather than omitted. Section 20 states that the most important result need not be that COSA is faster than every interior-point solver, and it is not. What the measurements support is narrower and is stated in Section 20.1.

18 Research Risks

18.1 Risk 1: The SOC Working-Set Concept May Be Insufficient

Answered, and the answer is a qualified yes with one exception. The three-valued working set — inactive, tangent, apex — together with a multiplier-driven deactivation rule is sufficient to *represent* the geometry. What it could not do was *step*, because the direction knew only the boundary's tangent plane; eq. (4b) supplies the missing derivative and the randomized sweep goes from 162 to 192 instances reaching an optimum. The exception is the unjustified apex of Section 14.4, which occurs on about 1.5% of randomly generated instances and is not fixable by a working-set rule.

The boundary of a second-order cone has richer geometry than a polyhedral inequality.

If a simple "SOC active/inactive" binary state is insufficient, the algorithm will be generalized to use cone faces and normal-cone information.

18.2 Risk 2: Cycling

Observed, diagnosed and resolved. The cycle that occurred was not the one anticipated. A randomly generated instance alternated between the apex and the inactive state 486 times: the apex branch released a factor it could not justify, the released direction could not be travelled, so the step moved nothing and the geometric activation rule read the same unchanged slack and restored the factor. The remedy is to not re-derive a status from geometry that has not changed. Across 200 randomly generated instances the worst working set is now returned to twice.

Two measurement errors are worth recording beside it. The revisit counter initially counted iterations *held* on a working set rather than returns *to* it, which reported nine hundred revisits for a solve that was merely converging slowly and concealed the real cycle underneath. And a solver may repeat an iteration that changes neither the iterate nor the working set; that is cycling of a degenerate kind and is detected directly, rather than through a threshold on the direction's size.

As in classical active-set methods, the algorithm may cycle among working sets.

Potential remedies include:

- anti-cycling rules;
- multiplier tolerances;
- lexicographic selection;
- merit-function safeguards.

18.3 Risk 3: Numerical Instability

Nearly dependent constraints and ill-conditioned covariance matrices may produce unstable KKT systems.

This will be addressed through scaling, rank detection, regularization, and robust factorizations.

Partly answered, and the residue is not where it was expected. The KKT system does not inherit $\kappa(\Sigma)$, because the tangent representation enters it as a single row and a single row has no spectrum — so the obvious instability route is closed. Rank deficiency is likewise not a difficulty. Scaling turns out to rescue no family and to cost 30–80% more iterations on every one.

What is left is Section 15.6: on a badly scaled instance the *stopping criterion* rather than the linear algebra is what fails, certifying an answer that is percent-wrong. Robust factorizations do not help with that and neither does equilibration. It is a criterion problem.

18.4 Risk 4: Poor Generic Performance

COSA may not outperform interior-point SOCP solvers on large, isolated, generic problems.

This is not considered a failure of the project. The principal hypothesis is that active-set methods may be particularly attractive for structured sequences of related problems and problems with a stable active structure.

19 Expected Contributions

The project aims to produce four contributions.

19.1 1. Mathematical Contribution

A precise formulation of an active-set method for second-order cone constraints, including the appropriate tangent, normal, and primal-dual KKT geometry.

19.2 2. Algorithmic Contribution

A practical algorithm that combines:

$$\text{polyhedral working sets} + \text{SOC geometry} + \text{exact cone-aware step lengths}.$$

19.3 3. Software Contribution

An open and transparent implementation of COSA suitable for research, experimentation, and extension to other SOCPs.

19.4 4. Computational Contribution

A systematic study of when conic active-set methods are competitive with interior-point methods, with particular emphasis on portfolio optimization and warm starts.

20 Milestones

The project will proceed through the following milestones.

Milestone	Deliverable	Status
M1	Mathematical problem specification	Planned
M2	Polyhedral active-set baseline	Planned
M3	SOC geometry module	Planned
M4	Exact SOC step calculation	Planned
M5	First COSA prototype	Planned
M6	Conic KKT formulation	Planned
M7	Robust numerical implementation	Planned
M8	Warm-start capability	Planned
M9	Portfolio benchmark suite	Planned
M10	Comparison with reference solvers	Planned
M11	Final numerical study	Planned
M12	Research paper and software release	Planned

21 Final Deliverables

The project will conclude with the following deliverables.

1. A complete mathematical specification of COSA.
2. A working implementation of the algorithm.
3. A unit-tested SOC geometry library.

4. A robust active-set/KKT linear algebra implementation.
5. A portfolio optimization interface.
6. A benchmark suite.
7. Numerical comparisons against established SOCP solvers.
8. Warm-start experiments along the efficient frontier.
9. Documentation and reproducible experiments.
10. A research paper describing the method and results.

21.1 Status

All ten exist. Where a deliverable is a document rather than code, the document is named; where it is code, the module is.

	deliverable	where
1	mathematical specification	Sections 2–8, 14.4
2	working implementation	`cosa.solver`
3	unit-tested SOC geometry library	`cosa.geometry`
4	active-set/KKT linear algebra	`cosa.active_set`, `cosa.linear_algebra`
5	portfolio optimization interface	`cosa.solve_portfolio`
6	benchmark suite	`cosa.experiments.benchmarks`
7	comparisons against SOCP solvers	Section 15.5
8	warm-start frontier experiments	Section 15.3, `cosa.experiments.frontier`
9	documentation, reproducible experiments	`python -m cosa.experiments`
10	research paper	this document

Deliverable 1 is the one that was deferred to itself: Section 14.4 states the standing assumptions, the constraint qualification, the generic case and what each degenerate case costs.

Success Criterion 7 — modularity sufficient to support extensions beyond the initial portfolio application — is *demonstrated* rather than asserted, in `tests/test_modularity.py`: an SOCP with two and then three independent second-order cones, none of them a portfolio, solved through the ordinary entry point and checked against a reference solver. No module in `cosa.solver` is edited, subclassed or monkeypatched, and the test asserts that of itself by parsing its own syntax tree.

22 Longer-Term Extensions

Once the basic COSA solver is operational, several extensions become possible.

22.1 Multiple Second-Order Cones

The method can be generalized to problems containing several constraints

$$(t_j, L_j x) \in \mathcal{Q}_j, \qquad j = 1, \ldots, m.$$

The working set would then track the active geometry of multiple cones.

22.2 General SOCPs

The portfolio structure can be removed entirely, giving a solver for the generic form

$$\begin{aligned} \min_{x} \quad & c^\top x \\ \text{s.t.} \quad & Ax = b, \\ & Gx + h \in \mathcal{K}, \end{aligned}$$

where $\mathcal{K}$ is a Cartesian product of second-order cones.

22.3 Mixed Conic Problems

A later version could investigate combinations of:

- linear inequalities;
- second-order cones;
- rotated second-order cones;
- other structured convex cones.

22.4 Large-Scale and Sparse Problems

For large portfolios and factor models, covariance matrices often have exploitable low-rank structure. COSA could incorporate factor-based representations to reduce the cost of the SOC operations.

23 Success Criteria

The project will be considered successful if COSA satisfies the following criteria.

1. It reliably solves the initial portfolio SOCPs to high accuracy.
2. Its termination criterion is based on mathematically meaningful conic KKT residuals.
3. Its working-set decisions can be interpreted in terms of the active portfolio constraints and SOC geometry.
4. It demonstrates reliable warm starts on sequences of related portfolio problems.
5. Its solutions agree with established SOCP solvers.
6. Its numerical behavior is understood on degenerate and ill-conditioned problems.
7. The resulting implementation is sufficiently modular to support extensions beyond the initial portfolio application.

The most important scientific result need not be that COSA is faster than every interior-point solver. A valuable result would instead be a clear characterization of when conic active-set methods work well, why they work well, and how their geometry differs fundamentally from polyhedral active-set methods.

23.1 Outcome

Four of the seven hold without qualification, three hold with one, and the qualifications are the results worth having.

	criterion	outcome
1	solves the portfolio SOCPs to high accuracy	**yes**, on twelve of the thirte stance families
2	termination on meaningful conic KKT residuals	**yes, and not sufficient** – tion 15.6
3	working-set decisions interpretable as active constraints	yes
4	reliable warm starts on sequences	**conditionally** — where t tive set transfers, Section
5	solutions agree with established SOCP solvers	**no**, on one family — S 15.6
6	numerical behaviour understood on degenerate and ill-conditioned problems	yes, including where it is v
7	modular enough for extensions beyond portfolios	yes, demonstrated rather asserted

Criterion 2 is the one that repays attention. The residuals are mathematically meaningful, which is what it asked for; they are not *sufficient*, which it did not think to ask. Criterion 5 fails on the same instance and for the same reason. Both are Section 15.6, and both are consequences of normalizing a residual by $\|c\|_\infty$ — a choice made for good reasons in a formulation where $\|c\|_\infty$ is large for reasons of scaling rather than of size.

Criterion 4 is qualified in a more interesting way. Warm starting is reliable exactly where the *combinatorial* state transfers, and that is not the same condition as the problems being close: a frontier can move steadily in λ while its active set jumps. Where the working set must be corrected, warm starting costs more than it saves.

23.2 The Characterization

That characterization is the project's principal result and is stated here in the form the measurements of Section 15 support.

When they work well. On *sequences* of related problems whose active set is stable, and specifically on the subset of a sequence where it transfers unchanged. There, warm starting saves 18–55% of the iterations and the factorization is reused rather than recomputed. Where the active set must be corrected, warm starting *costs* 17–32%, and the sign of a whole sweep is decided by the mix of the two. The condition is not "the problems are close" but "the combinatorial state is the same"; those are not the same condition, and a frontier can move steadily in λ while its active set jumps.

Why they work well there. Because the state an active-set method carries is combinatorial and an interior-point method's is not. A previous solution is on the boundary of the feasible set, which is the worst possible place to begin a central path from; the same solution's *working set* is exactly the information the next solve would otherwise spend most of its iterations rediscovering. The asymmetry is not about the quality of the starting point, it is about what kind of object is being carried.

How the geometry differs from the polyhedral case. In four ways, each of which showed up as an implementation consequence rather than as a remark.

1. *The active constraint is a face, not a hyperplane.* A polyhedral working set is a set of indices. A conic one is a set of indices *plus a state per cone factor*: inactive, tangent, or at the apex. The three are not degrees of the same thing — at a tangent factor the dual condition is a scalar sign test, and at the apex it is a cone membership test, because the normal cone of a self-dual cone at its apex is $-\mathcal{Q}$.

2. *The active set moves continuously.* A linear constraint enters and leaves the working set discretely. The cone's tangent row changes on *every* iteration the cone is active, because $u = Lx/\|Lx\|_2$ moves with x. Factorization updating survives this, but only because the update is applied to $W^\top$, where a moving tangent replaces one column and the remaining structure is untouched.

3. *A boundary iterate cannot move under a linear step.* This is the sharpest difference and it is a theorem, not a numerical difficulty. At a boundary point a direction satisfying the tangency condition eq. (3) has an exact conic step of *zero*: tangency annihilates eq. (6)'s middle coefficient, feasibility annihilates its constant term, and Cauchy–Schwarz puts its leading coefficient at or above zero. A polyhedral active set has no analogue — a face of a polyhedron is flat, so a step along it stays on it. The remedy is a retraction, and the cost of the retraction is why the method is first-order along a curved boundary unless the direction is given the boundary's curvature, which is eq. (4b).

4. *The conditioning does not transfer.* The tangent representation puts L into the KKT system as a single row, and a single row has no spectrum, so the assembled system does not inherit $\kappa(\Sigma)$. The same structure destroys sparsity: u is a normalized covariance direction and therefore dense in every asset, so however sparse A is, one dense row and its transpose put a dense cross in the matrix. A conic active-set method is better conditioned and less sparse than the polyhedral intuition predicts, and both facts have the same cause.

Where they do not work well. On single problems solved once, where the reference solver is three to thirty times faster; and at an apex whose multiplier does not justify it, where on eq. (7) the release the dual conditions authorize is arithmetically unavailable. The second is a genuine limitation of the working-set concept rather than of this implementation of it.

24 Summary

COSA begins with a simple but important modeling observation: standard deviation is often a more natural risk quantity than variance because it has the same units as expected return.

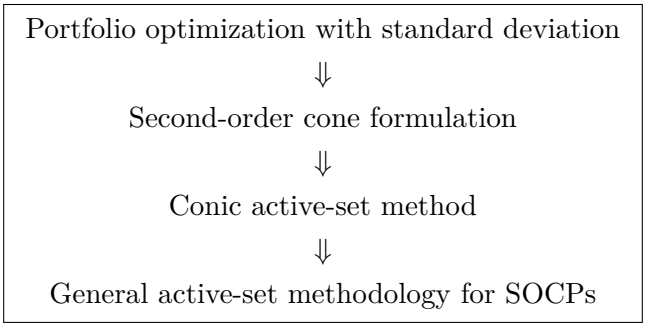
The resulting portfolio problem is naturally represented as an SOCP:

$$\begin{aligned} \min_{x,t} \quad & -\mu^\top x + \lambda t \\ \text{s.t.} \quad & Ax \le b, \\ & Ex = d, \\ & (t, Lx) \in \mathcal{Q}. \end{aligned}$$

The linear constraints retain the familiar structure of an active-set method, while the risk term introduces a second-order cone.

COSA will investigate whether these two structures can be combined into a single algorithmic framework.

The project therefore has both a practical and a broader algorithmic objective:



The portfolio problem is the test bed. The deeper goal is COSA itself: a transparent, warm-startable, structure-exploiting active-set approach to conic optimization.