A Differentiable CVaR Projection Primitive for Risk-Constrained Optimization

Riyan Christy
Columbia University
rc3148@columbia.edu

June 12, 2026

Abstract

Conditional value-at-risk (CVaR), also known as expected shortfall, is a convex tail-risk measure used in finance, energy systems, logistics, control, and statistical learning. Recent work on CVaR-constrained quadratic programs (CVQPs) shows that the key computational primitive is not a generic conic or quadratic-programming reformulation, but a specialized projection onto a CVaR constraint. Efficient algorithms for this top-k-sum projection are now an active literature, including sorted-input linear-time and sort-free methods. This paper therefore treats the forward projection as a black box that returns the projected point and an active-face certificate, and focuses on the complementary differentiation problem.

We present a differentiable, batched CVaR projection primitive and describe how it can be used inside a CVQP layer. The backward pass exploits the active face of the projected point: on a fixed polyhedral face, the projection is affine, and its Jacobian is the orthogonal projector onto the tangent space of that face. This yields vector-Jacobian products in $O(m)$ time once an active-face certificate is available, including selected derivatives with respect to the projection input, the CVaR level κ, and, piecewise, the tail parameter β. No deterministic-equivalent CVaR quadratic program is formed. For full CVQPs, accurate solves are differentiated through a reduced active-face KKT system; early-stopped solves are differentiated by unrolling. We validate the primitive against an independent solver and finite differences: the projection-input vector-Jacobian product error is below 10^{-8}, including degenerate plateaus, and the d and κ adjoints are below 10^{-6}. On a single GPU the backward pass scales to 2×10^8 scenarios; a 200-million-scenario CVaR projection is differentiated in under 0.12 s. The primitive is also several orders of magnitude faster and more memory-frugal than a CPU CVXPYlayers/diffcp deterministic-equivalent epigraph baseline, which exhausts memory or times out near $m = 10^5$. A pre-specified five-seed screening experiment explains why this scale matters for decisions: hard CVaR constraints drive the realized-to-budget tail-risk ratio from 1.17 at $m = 10^3$ to 1.00 at $m = 10^5$ in distribution and from 1.55 to 1.01 under distribution shift, whereas an in-distribution calibrated fixed-multiplier penalty drifts to 1.76× budget under the same shift and an oracle-retuned penalty repairs the shifted mean but not the per-instance dispersion. An end-to-end demonstration trains through the layer at $m = 10^5$ (three seeds, 5.8 s per step in 1.3 GB). It is a capability and composition check, not evidence of a decision-focused outperformance claim. All code and a one-command reproduction are released.

Keywords: conditional value-at-risk (CVaR); top-k-sum projection; differentiable optimization; vector-Jacobian product; ADMM; risk-constrained quadratic programming.

MSC codes: 90C25, 90C20, 90C31, 65K05.

1 Introduction

Many deployed decision systems optimize a nominal objective while controlling rare but severe losses. A portfolio optimizer controls losses under market stress; an energy scheduler limits blackout or imbalance costs; a supply-chain planner limits worst-tail shortage costs; and a learning system may optimize a loss under distribution shift. CVaR is attractive in these settings because it is convex and directly controls the expected value of the tail of the loss distribution [22].

For samples $z_1, \ldots, z_m \in \mathbb{R}$ representing losses, the sample CVaR at level $\beta \in (0, 1)$ is

$$\phi_\beta(z) = \inf_{\alpha \in \mathbb{R}} \left\{ \alpha + \frac{1}{(1-\beta)m} \sum_{i=1}^{m} (z_i - \alpha)_+ \right\}. \tag{1}$$

The constraint $\phi_\beta(z) \leq \kappa$ has a standard linear-inequality epigraph representation with $O(m)$ auxiliary variables and constraints. This representation is convenient for modeling languages but expensive at large scenario counts. The CVQP solver of Luxenberg, Perez-Pineiro, Diamond, and Boyd avoids this deterministic-equivalent expansion by using operator splitting and a specialized projection onto the CVaR set [17]. Projection algorithms for the same top-k-sum sublevel set have also been studied directly, including sorted-input linear-time methods and sort-free algorithms [24, 20]. The forward scaling achieved by this solver and projection literature raises a natural next question:

> *Can a CVaR-constrained optimization problem be differentiated through at the same scenario scale at which it can be solved?*

This question matters for decision-focused learning, differentiable control, hyperparameter tuning, and risk-aware forecasting. In a predict-then-optimize pipeline, a model predicts scenario losses or returns, then an optimization layer chooses an action. Training end-to-end requires gradients through the risk-constrained decision rule. Existing differentiable optimization layers can differentiate through quadratic and conic programs [1, 3, 25]. Applied directly to CVaR constraints, however, they typically differentiate a sparse deterministic-equivalent epigraph with $O(m)$ auxiliary variables and constraints. The issue is not formal infeasibility in all cases, but the cost, fill-in, memory traffic, and lack of batched GPU structure that appear exactly where the specialized CVQP forward solver is useful. The experiments of Section 9 add a statistical reason to care about the large-m regime, beyond solver throughput: with a hard CVaR constraint, the realized out-of-sample tail risk of the decision converges to its budget as m grows, so the scenario count acts as a decision-quality variable. Differentiating the constraint at large m is what lets a learning pipeline operate at the sample sizes where its decisions actually meet their risk budgets.

Main idea. We make the CVaR projection itself differentiable. The projection is onto a polyhedron. Its active face can be recovered from the projected point $z^\star$, the constraint level, an active/inactive tolerance, and the violation (equivalently, multiplier) status of the forward solve. The last item matters exactly on the boundary: a feasible input with zero slack has zero multiplier, and a certificate built from $z^\star$ alone cannot distinguish this inactive boundary case from an active projection; the forward oracle can, because it knows whether it moved the point. Beyond this status bit, the original input v is needed only to recover solver diagnostics. Once that face is fixed, the Euclidean projection is an affine map, and the Jacobian is simply the orthogonal projection onto the tangent space of the active face. Therefore the projection backward pass needs no new sorting and no large conic KKT solve.

Contributions. This paper makes four contributions.

1. We formulate the CVaR projection as a differentiable primitive

$$v \mapsto \Pi_{\{z:\phi_\beta(z)\le\kappa\}}(v)$$

with a cached active-face representation.

2. We prove that, under a locally constant active face, the Jacobian of the projection is

$$D\Pi_{\mathcal{C}}(v) = I - G^\top (GG^\top)^\dagger G,$$

where $Gz = h$ is a compressed equality description of the active face. We also give the derivative with respect to $d = k\kappa$, κ, and piecewise in β, plus the no-tie formula for fractional tail weights. This gives an $O(m)$ vector-Jacobian product once the active-face certificate is available.

3. We embed this differentiable projection inside a differentiable CVQP layer. The forward pass may be the CVQP ADMM solver, but the recommended implicit backward pass for accurate solves differentiates the active-face KKT optimality conditions. For deliberately inexact or early-stopped solves, unrolling with checkpointing differentiates the computation actually carried out.

4. We give an open-source implementation and a full empirical evaluation: correctness against an independent solver and finite differences (including degenerate plateaus), $O(m)$ scaling to 2×10^8 scenarios on a single GPU, a head-to-head comparison against a general-purpose differentiable convex-optimization layer, a pre-specified hard-constraint-versus-penalty experiment at scale (frozen hypotheses, a dated amendment log, and screening intervals), an end-to-end training demonstration at $m = 10^5$, a robustness/failure analysis, and a decision-focused CVQP layer composition check. Code, exact environment, and a one-command reproduction are released.

2 Related work and positioning

CVaR and superquantiles are widely used in convex risk modeling [22]. A closely related machine-learning line uses superquantiles as objectives for robustness and fairness. Laguel, Pillutla, Malick, and Harchaoui review superquantile applications, explicit subgradient calculations, and smoothing by infimal convolution [15]; the SQwash package exposes PyTorch superquantile reducers, including a smooth reducer solved by exploiting special QP structure and sorting [21]. That work already supplies a batched, GPU-supported differentiable CVaR objective primitive. The distinction here is the constraint-projection setting: we differentiate the Euclidean projection onto $\{z : \phi_\beta(z) \le \kappa\}$ and use the exact selected active-face derivative rather than a smoothed objective derivative.

Differentiable convex optimization layers and code generation are also adjacent. OptNet differentiates QPs through KKT systems [3]; CVXPYlayers and diffcp differentiate cone programs through the cone-program optimality map and cone projections [1, 2]; and CVXPYgen-grad generates custom C code for solving and differentiating through parametrized convex programs [25]. The present contribution is not generic code generation. It is a CVaR-constraint-specialized projection VJP and reduced active-face KKT interface that avoids materializing the deterministic-equivalent CVaR epigraph. Recent differentiable-QP work such as BPQP also attacks the cost of backpropagating through optimization layers with many constraints [19]. The distinction is again specialization: BPQP is a general framework, while the primitive here uses the top-k/CVaR face structure to collapse the backward pass to grouping plus one weighted rank-one correction.

The forward CVaR projection is itself solved literature. Lapin, Hein, and Schiele use projection onto a related top-k simplex in top-k SVM training [16]; Davis studies projection onto the ordered weighted ℓ_1 ball [10]; Roth and Cui give sorted-input $O(n)$ algorithms for the top-k-sum sublevel projection [24]; and Pan and Yan propose a sort-free top-k-sum projection method based on intersecting monotone piecewise-linear functions [20]. These works reinforce the boundary of this paper: the forward projection is not the contribution. The contribution is the explicit selected backward pass, including structured tie/plateau handling, and its use in reduced CVQP differentiation.

The sorting and tie structure also connects to differentiable sorting, ranking, and top-k selection [6, 8, 14, 28]. Those works typically smooth or relax the selection, permutation, or indicator map. The object here is different: the nonsmoothed Euclidean projection onto the top-k-sum sublevel set. The closest structural relative is the Jacobian of the Euclidean projection onto the probability simplex used by sparsemax [18], which likewise centers the adjoint over an active support. The present setting differs in the polytope (a top-k-sum sublevel set rather than the simplex), in the first-class treatment of plateaus created by the projection itself, and in the constraint-level adjoints for κ and β. The grouped-tie VJP below is the projection analogue of the Jacobian of an isotonic-regression or pool-adjacent-violators block used in differentiable sorting. The new wrinkle is not block averaging alone, which is standard; it is that the top-tail normal must first be projected into the block-constant subspace, producing the weighted normal used in the rank-one correction. Smoothed alternatives include perturbed optimizers and Fenchel-Young losses, which give principled continuous surrogates for argmax-over-polytope maps [4, 5]. The motivation aligns with decision-focused and predict-then-optimize learning [12, 13, 27]; those literatures motivate gradients through downstream decisions, while this paper targets the specific tail-risk constraint primitive needed by large-scale CVQP layers.

One cannot always replace a CVaR constraint by a smoothed CVaR objective without changing the application. In deployment settings with a hard tail-risk budget, the constraint $\phi_\beta(Ax) \le \kappa$ is the object being certified. A penalized or Lagrangian objective requires choosing a multiplier, and the multiplier that matches the budget can change sharply under prediction shift or active-set changes. This is the setting where an exact selected derivative of the constraint projection is materially different from a smoothed objective primitive.

3 Scope

The intended contribution is one primitive: a backward rule for the CVaR projection. The CVQP layer is an application of that primitive, and its total backward cost can still be dominated by original-problem linear algebra when n is large or sparse factorization is expensive. The clean $O(m)$ statement therefore applies to the projection VJP, not unconditionally to every CVQP layer.

The empirical claims are validated in §9. We report head-to-head moderate-scale comparisons against deterministic-equivalent epigraph differentiation, and then show the specialized method extending to scenario counts where that baseline no longer runs. Throughout, "the contribution" refers to the backward pass; the forward projection is taken from existing literature and is included only so the experiments are self-contained.

4 CVaR and top-k representation

For the main exposition, assume that

$$k = (1 - \beta)m$$

is an integer. Then the sample CVaR constraint is equivalent to a sum-of-largest constraint

$$\phi_\beta(z) \le \kappa \quad \Longleftrightarrow \quad f_k(z) \le d, \qquad d = k\kappa, \tag{2}$$

where

$$f_k(z) = \sum_{i=1}^{k} z_{[i]}$$

and $z_{[1]} \ge z_{[2]} \ge \cdots \ge z_{[m]}$ are the sorted entries of z in nonincreasing order. The feasible set is the polyhedron

$$\mathcal{C}_{k,d} = \{z \in \mathbb{R}^m : f_k(z) \le d\}. \tag{3}$$

Equivalently,

$$f_k(z) = \max_{a \in \mathcal{A}_k} a^\top z, \qquad \mathcal{A}_k = \{a \in \{0,1\}^m : \mathbf{1}^\top a = k\}. \tag{4}$$

Thus

$$\mathcal{C}_{k,d} = \{z : a^\top z \le d \ \text{ for all } a \in \mathcal{A}_k\}. \tag{5}$$

When $(1-\beta)m$ is not an integer, the fractional case should be handled explicitly rather than treated as a minor corner case. Let

$$\tau = (1-\beta)m, \qquad s = \lfloor \tau \rfloor, \qquad \eta = \tau - s \in [0, 1).$$

For $\eta > 0$,

$$\phi_\beta(z) \le \kappa \quad \Longleftrightarrow \quad \sum_{i=1}^{s} z_{[i]} + \eta z_{[s+1]} \le \tau\kappa. \tag{6}$$

Equivalently the active normal is a weighted top-tail vector with s entries equal to one and one entry equal to η. The integer exposition below is therefore the special case $\eta = 0$ with $\tau = k$.

5 The projection layer

The projection layer maps

$$z^\star = \Pi_{\mathcal{C}}(v) = \operatorname*{arg\,min}_{z \in \mathcal{C}_{k,d}} \frac{1}{2} \|z - v\|_2^2 . \tag{7}$$

The forward projection may be computed by any exact top-k-sum projection routine. The key observation for differentiation is that the face of $\mathcal{C}_{k,d}$ containing $z^\star$ is recoverable from the projection output: one needs the violation or multiplier status of the forward solve, the tail plateau structure, and the tail weights. The status bit is not redundant: when $f_k(z^\star) = d$ to tolerance, $z^\star$ alone cannot distinguish a feasible input that sits exactly on the boundary (zero multiplier, identity derivative on the feasible side) from an active projection, while the forward oracle knows whether it moved the point. A solver may cache this metadata during the forward pass, but the backward rule does not depend on CVQP-specific internals or on a particular sorting-based projection algorithm.

5.1 Geometry

If $v \in \mathcal{C}_{k,d}$ with positive slack, then $\Pi_{\mathcal{C}}(v) = v$ and the Jacobian is the identity. If $v \notin \mathcal{C}_{k,d}$, then $z^\star$ lies on the boundary $f_k(z) = d$. A feasible v with zero slack is a normal-fan cell boundary point: the identity (from the feasible side) and the face projector (from the infeasible side) are both one-sided

derivatives, and the implementation's convention is to return the branch selected by the recorded violation status at the stated tolerance. The active face can be written in compressed form as

$$\mathcal{F} = \{z \in \mathbb{R}^m : Gz = h,\ Rz < r\}, \tag{8}$$

where $Gz = h$ contains the equality constraints active at $z^\star$, and $Rz < r$ contains inactive inequalities that remain inactive locally.

In a no-plateau region, the active top-k set

$$T = \operatorname*{Top}_k(z^\star)$$

is unique, and the face equality is simply

$$\sum_{i \in T} z_i = d. \tag{9}$$

Then $G = \mathbf{1}_T^\top$ and

$$D\Pi_{\mathcal{C}}(v)u = u - \mathbf{1}_T \frac{\sum_{i \in T} u_i}{k}. \tag{10}$$

This subtracts the mean perturbation over the tail-active set and leaves all non-tail coordinates unchanged.

Plateaus and ties are not merely floating-point pathologies. When the projection must reduce a large violation, the multiplier can pull top entries down until order statistics meet, creating a plateau even if the input entries are distinct. Thus grouped ties should be treated as a first-class regime, especially in early ADMM iterates or badly scaled problems. There may then be multiple active top-k indicators. A compressed G can be represented by:

1. the tail-sum equality $a^\top z = d$ for a representative active tail vector a, and
2. the rule that perturbations inside a *partially included* tied group must move together.

For VJPs, this representation should be applied implicitly. First average the adjoint over each tied group to project onto the within-group-constant subspace; then apply the tail-sum correction in group coordinates using the weighted group normal; finally scatter the group corrections back to coordinates. This keeps the grouped-tie VJP linear in m. Forming the $e - 1$ pairwise equality rows for a plateau of size e and solving the dense $GG^\top$ system would destroy that complexity.

Lemma 1 (Grouped-plateau VJP)**.** *Consider a face description with strict set S, $|S| = s$, and tied groups $P_1, \dots, P_t$ of sizes $g_1, \dots, g_t$, each* partially included*: the total tail weight q_j assigned to group P_j satisfies $0 < q_j < g_j$. For an integer top-k face the q_j are integer tail counts, while in a fractional-tail face the group containing the fractional boundary coordinate includes the fractional weight η in its q_j. Let A be the subspace of perturbations that are constant on each P_j, and let P_A be the orthogonal projector that averages within each group and leaves other coordinates unchanged. If a is any compatible representative tail vector, define*

$$b = P_A a = \mathbf{1}_S + \sum_{j=1}^{t} \frac{q_j}{g_j} \mathbf{1}_{P_j}, \qquad \|b\|_2^2 = s + \sum_{j=1}^{t} \frac{q_j^2}{g_j}.$$

Then the orthogonal projector onto $\mathcal{T} = A \cap \{u : b^\top u = 0\}$ is

$$P_{\mathcal{T}} \bar{z} = P_A \bar{z} - \frac{b^\top \bar{z}}{s + \sum_{j=1}^{t} q_j^2 / g_j}\, b. \tag{11}$$

When the description is the active face of $\mathcal{C}_{k,d}$ at $z^\star$, this operator is the exact selected VJP.

Proof. Because $0 < q_j < g_j$, for each group there exist two active selections that differ inside P_j; requiring the tail-sum equality for every compatible tail selection therefore forces perturbations to be constant within each tied group, giving the subspace A. In the quotient where each group is averaged, the tail-sum equality has normal $b = P_A a$, not the raw representative a. Since $b \in A$, $P_A b = b$, so the averaging projector P_A and the rank-one projector onto $b^\perp$ commute. Thus the orthogonal projection onto $A \cap b^\perp$ is the sequential operation in (11). □

Remark 1 (Endpoint weights, and merged certificates)**.** *Partial inclusion is what forces the averaging. A tied group with $q_j = g_j$ (all members in the tail, with a strict gap below it) or with $q_j = 0$ admits only one compatible selection pattern on that group, so the face imposes no within-group constancy: such coordinates enter the face description individually. A $q_j = g_j$ group is absorbed into the strict set S, and the no-tie formula (10) applies over $S \cup P_j$; averaging it anyway returns the projector onto a strict subspace of the tangent space — a conservative surrogate, not the classical derivative, which exists there. For the integer CVaR facet, at most one group can be partially included (the plateau at the boundary value, since any group at a strictly larger value must be fully included before a smaller value receives weight), and the exact projection of an input whose entries at the boundary value are distinct always yields $1 \le q < g$: total group weight $q = g$ forces every selection weight in the group to one, which forces exact input ties. Thus $t = 1$ in exact arithmetic, and $q = g$ arises only from exactly tied inputs. A thresholded certificate that merges several near-boundary groups yields the projector onto the merged group-constant subspace; by Proposition 3 this coincides with the exact selected derivative whenever the tolerance is below the cell margin, and is otherwise conservative.*

Using the raw representative a with normalization k is generally wrong in a plateau, because a is not invariant under group averaging. The formula above is also the implementation rule: average within each partially included group, apply one rank-one correction using b, and scatter back to coordinates.

5.2 Global piecewise-affine structure

Theorem 1 (Global structure)**.** *For fixed β and κ, the map $v \mapsto \Pi_{\mathcal{C}}(v)$ is globally Lipschitz, piecewise affine, and differentiable except on a finite union of polyhedral cell boundaries. On each full-dimensional cell, the projection lies in the relative interior of a fixed face of $\mathcal{C}$, strict complementarity holds, and the Jacobian is the tangent-space projector for that face.*

Proof. The CVaR set is a convex polyhedron. The Euclidean projection onto any closed convex set is nonexpansive, hence globally Lipschitz, and the projection onto a polyhedron is piecewise affine [26]. For a polyhedron, the domain decomposes into finitely many normal-fan cells: for each face $\mathcal{F}$, points v whose projection lies in $\mathcal{F}$ can be written as $v = z + n$ with $z \in \mathcal{F}$ and $n \in N_{\mathcal{C}}(\mathcal{F})$. On the relative interior of such a cell, the active face and normal cone are fixed, so the projection is the affine projection onto the affine hull of $\mathcal{F}$. The union of cell boundaries has Lebesgue measure zero, and the derivative on each full-dimensional cell is the projector computed in Theorem 2. □

5.3 Jacobian theorem

The following result is the CVaR-polyhedron specialization of standard projection sensitivity on a fixed polyhedral face [23, 2]. The useful specialization is the explicit active-face representation and the cheap VJP.

Theorem 2 (Projection derivative on a fixed face)**.** *Let $\mathcal{C} \subseteq \mathbb{R}^m$ be the CVaR set (3). Suppose v has projection $z^\star = \Pi_{\mathcal{C}}(v)$ on the face*

$$\mathcal{F} = \{z : Gz = h,\ Rz < r\},$$

where G has full row rank after removing redundant rows. Assume $z^\star$ lies in the relative interior of $\mathcal{F}$ and $v - z^\star$ lies in the relative interior of the normal cone of $\mathcal{C}$ along $\mathcal{F}$; for a single active CVaR facet, this is strict positivity of the projection multiplier. Then the active face is locally constant around v, and on that neighborhood $\Pi_{\mathcal{C}}$ is affine, differentiable, and

$$D\Pi_{\mathcal{C}}(v) = P_{\mathcal{T}} = I - G^\top (GG^\top)^{-1} G, \tag{12}$$

where

$$\mathcal{T} = \{u : Gu = 0\}$$

is the tangent space of the active face. If G is not full row rank, the same formula holds with $(GG^\top)^\dagger$ in place of $(GG^\top)^{-1}$. Moreover, if the face right-hand side h is perturbed while the same face description remains valid, then the selected local differential is

$$dz^\star = \left(I - G^\top (GG^\top)^\dagger G\right) dv + G^\top (GG^\top)^\dagger dh. \tag{13}$$

Proof. The relative-interior condition is the standard polyhedral strict complementarity condition ensuring that small perturbations of v keep the projection in the same face. On that neighborhood, projecting onto $\mathcal{C}$ is the same as projecting onto the affine hull of that face,

$$\{z : Gz = h\}.$$

The Euclidean projection of v onto this affine set is

$$\Pi_{\{Gz=h\}}(v) = v - G^\top (GG^\top)^\dagger (Gv - h).$$

This map is affine. Its derivative with respect to v is

$$I - G^\top (GG^\top)^\dagger G,$$

which is the orthogonal projector onto the nullspace of G, i.e., the tangent space of the face. Its derivative with respect to h is $G^\top (GG^\top)^\dagger$. □

For the integer no-tie CVaR boundary, $G = \mathbf{1}_T^\top$ and $h = d = k\kappa$. Equation (13) becomes

$$dz^\star = \left(I - \frac{1}{k}\mathbf{1}_T \mathbf{1}_T^\top\right) dv + \frac{1}{k}\mathbf{1}_T\, dd = \left(I - \frac{1}{k}\mathbf{1}_T \mathbf{1}_T^\top\right) dv + \mathbf{1}_T\, d\kappa. \tag{14}$$

If the constraint is inactive with positive slack, the local derivative with respect to κ is zero.

For the fractional no-tie case (6), let a denote the weighted tail vector with entries 1 on the top s coordinates, entry η on the next coordinate, and zeros elsewhere. With $d = \tau\kappa$ and $\|a\|_2^2 = s + \eta^2$, the selected derivative is

$$dz^\star = \left(I - \frac{aa^\top}{\|a\|_2^2}\right) dv + \frac{a}{\|a\|_2^2}\, dd = \left(I - \frac{aa^\top}{\|a\|_2^2}\right) dv + \frac{\tau a}{\|a\|_2^2}\, d\kappa. \tag{15}$$

The map is also piecewise differentiable with respect to β on fractional faces where $s = \lfloor \tau \rfloor$, the order, and the active face are fixed. Let r be the index of the fractional tail coordinate, let e_r be its basis vector, and write

$$n(\tau) = \|a(\tau)\|_2^2 = s + \eta^2, \qquad c(\tau) = a(\tau)^\top v - \tau\kappa, \qquad \lambda(\tau) = \frac{c(\tau)}{n(\tau)}.$$

On the active no-tie face,

$$z^\star = v - \lambda(\tau)a(\tau), \qquad \frac{da}{d\tau} = e_r, \qquad \frac{dn}{d\tau} = 2\eta,$$

and therefore

$$\frac{dz^\star}{d\tau} = -\left(\frac{(v_r - \kappa)n(\tau) - 2\eta c(\tau)}{n(\tau)^2}\right) a(\tau) - \lambda(\tau)e_r, \qquad \frac{dz^\star}{d\beta} = -m\frac{dz^\star}{d\tau}. \tag{16}$$

Thus β is not globally smooth, but it is differentiable on fixed fractional faces. Nonsmoothness occurs at integer tail-mass crossings, ties, and active-face changes.

At integer tail-mass crossings, $z^\star$ remains continuous in τ because the weighted top-tail constraint and its right-hand side vary continuously, but the derivative can jump. The one-sided derivatives are obtained from (16) by taking $\eta \to 0^+$ on the interval where a new fractional coordinate enters and $\eta \to 1^-$ on the interval where that coordinate becomes fully weighted.

Proposition 1 (Almost-everywhere correctness)**.** *Assume the forward projection is exact and the active-face certificate is exact. For Lebesgue-a.e. v, face mode returns the classical Jacobian of $v \mapsto \Pi_{\mathcal{C}}(v)$. At a cell boundary it returns the Jacobian of the selected adjacent affine region when that region is represented by the certificate.*

Proof. By Theorem 1, $\Pi_{\mathcal{C}}$ is affine on each full-dimensional normal-fan cell and nondifferentiability can occur only on the finite union of cell boundaries, which has measure zero. On a full-dimensional cell, the active-face certificate identifies the cell's affine formula, so Theorem 2 gives the classical Jacobian. On a boundary, selecting a neighboring face selects the limiting Jacobian from that neighboring affine region. □

Proposition 2 (Generalized-Jacobian membership)**.** *At any v, if the face-mode Jacobian is the limit of Jacobians from a sequence of differentiable points approaching v, then it is an element of the B-subdifferential $\partial_B \Pi_{\mathcal{C}}(v)$. Since the Clarke generalized Jacobian is the convex hull of $\partial_B \Pi_{\mathcal{C}}(v)$ for locally Lipschitz piecewise-affine maps, the same selected Jacobian is also an element of the Clarke generalized Jacobian. The damped mode with fixed $\epsilon > 0$ is not, in general, an element of either exact generalized Jacobian.*

Proof. For a locally Lipschitz piecewise-affine map, the B-subdifferential is precisely the set of limiting Jacobians along sequences of differentiability points. This is the stated hypothesis. The Clarke generalized Jacobian is the convex hull of those limiting Jacobians. The damped operator changes the rank-one or multi-row projection formula by Tikhonov regularization and need not coincide with any cell Jacobian. □

Proposition 3 (Certificate stability)**.** *Let v lie in the interior of a normal-fan cell of $\Pi_{\mathcal{C}}$ with Euclidean distance $\Delta > 0$ to the nearest cell boundary. (i) Every perturbed input $\tilde{v}$ with $\|\tilde{v} - v\| < \Delta$ lies in the same cell, so the certificate computed at $\tilde{v}$ identifies the same active face, and hence the same Jacobian, as at v. (ii) The certificate's face decision depends on the projected point only*

through finitely many threshold comparisons whose margins at v are the smallest inactive slack, the smallest tie gap, and the smallest active multiplier; a certificate built from an inexact projected point $\hat{z}$ with $\|\hat{z} - \Pi_{\mathcal{C}}(v)\|_\infty < \mu/2$, using a thresholding tolerance below $\mu/2$, flips none of these comparisons, where $\mu > 0$ denotes the smallest of those margins. Active-set errors therefore require forward error or tolerance comparable to the cell margin.

Proof. (i) The open ball of radius Δ around v is contained in the cell, so Theorem 1 gives the same affine formula and Jacobian throughout it. (ii) Each face decision is a binary comparison of a coordinate of the projected point (or a derived quantity) against a threshold, with slack at least μ at the exact projection; an entrywise perturbation below $\mu/2$ combined with a tolerance below $\mu/2$ changes no comparison outcome. The slack, gap, and multiplier margins are the coordinate-level forms this cell-boundary distance takes for the CVaR projection. □

Remark 2 (Degeneracy)**.** *At a point where the active face changes, the projection may fail to be classically differentiable. In autodiff systems we can return a selected generalized Jacobian determined by the face recovered in the forward pass. This is an exact derivative on regions where that selected face is locally constant, but it is discontinuous across face changes. The implementation must therefore make its active-set thresholding rule explicit and test gradient error as a function of forward solve tolerance.*

5.4 Vector-Jacobian product

For reverse-mode autodifferentiation, we need

$$\bar{v} = D\Pi_{\mathcal{C}}(v)^\top \bar{z}.$$

Since $D\Pi_{\mathcal{C}}(v)$ is an orthogonal projector, it is symmetric, and

$$\bar{v} = \bar{z} - G^\top (GG^\top)^\dagger G\bar{z}. \tag{17}$$

Thus, the backward pass is the same operation as the forward-mode Jacobian multiply.

If h is a differentiable parameter, the corresponding adjoint is

$$\bar{h} = (GG^\top)^\dagger G\bar{z}. \tag{18}$$

In the integer no-tie case this gives

$$\bar{d} = \frac{1}{k}\sum_{i\in T} \bar{z}_i, \qquad \bar{\kappa} = \sum_{i\in T} \bar{z}_i.$$

In the fractional no-tie case, with weighted tail vector a and $d = \tau\kappa$,

$$\bar{d} = \frac{a^\top \bar{z}}{\|a\|_2^2}, \qquad \bar{\kappa} = \tau \frac{a^\top \bar{z}}{\|a\|_2^2}.$$

In the no-tie case, this becomes

$$\bar{v}_i = \begin{cases} \bar{z}_i - \frac{1}{k}\sum_{j\in T} \bar{z}_j, & i \in T, \\ \bar{z}_i, & i \notin T. \end{cases} \tag{19}$$

This costs $O(m)$ time and $O(1)$ extra memory beyond the cached active set.

5.5 Algorithm

Forward pass. Input v, β, κ. Run any exact CVaR/top-k-sum projection routine and return $z^\star$. The minimum active-face certificate for the backward pass is:

1. whether the projection is inactive, recorded from the forward solve's violation or multiplier status (not inferred from $z^\star$ alone, which is ambiguous exactly on the boundary),
2. the strict-tail set size or weight, with fully included tied groups ($q_j = g_j$) absorbed into the strict set per Remark 1,
3. the tied groups that intersect the active tail boundary partially,
4. each such group's size g_j and assigned tail weight q_j,
5. the right-hand-side derivative information for d, κ, and optionally β,
6. the tolerance used to classify slack constraints, plateaus, and tail membership.

Sorting-based projection routines often produce this information naturally. A sort-free oracle must either return the same certificate or be followed by an $O(m)$ grouping pass over $z^\star$ at the stated tolerance; this grouping step is the main numerical fragility in the oracle-agnostic interface.

Backward pass. Input adjoint $\bar{z}$. If inactive, return $\bar{v} = \bar{z}$. Otherwise compute

$$\bar{v} = \bar{z} - G^\top (GG^\top)^\dagger G\bar{z}.$$

For no ties, use (19). For grouped ties, do not materialize the within-group equality rows. Apply group averaging, perform the weighted tail-sum correction from Lemma 1 in the compressed group coordinates, and scatter the result back to coordinates. Return $\bar{d}$ or $\bar{\kappa}$ from (18) when the constraint level is learnable.

Proposition 4 (Complexity)**.** *For a single vector $v \in \mathbb{R}^m$, the selected backward pass for the CVaR projection has*

$$\textit{backward cost } O(m).$$

The forward cost is inherited from the chosen projection oracle: comparison-sort implementations cost $O(m \log m)$, sorted-input or warm-started post-sort projection can be linear, and sort-free projection methods may remove sorting from the forward pass. The grouped-tie backward claim assumes the implicit averaging implementation above; an explicit dense solve in the plateau equality rows is not linear.

6 Differentiable CVQP layer

Consider the CVaR-constrained quadratic program

$$\begin{array}{ll} \text{minimize} & \frac{1}{2} x^\top P x + q^\top x \\ \text{subject to} & \phi_\beta(Ax) \le \kappa, \\ & l \le Bx \le u, \end{array} \tag{20}$$

with variable $x \in \mathbb{R}^n$. The CVQP solver introduces auxiliary variables $z = Ax$ and $\tilde{z} = Bx$ and applies ADMM [7]:

$$x^{t+1} = -M^{-1}\left(q - \rho A^\top (z^t - y^t) - \rho B^\top (\tilde{z}^t - \tilde{y}^t)\right), \tag{21}$$

$$z^{t+1/2} = \alpha A x^{t+1} + (1-\alpha) z^t, \tag{22}$$

$$\tilde{z}^{t+1/2} = \alpha B x^{t+1} + (1-\alpha) \tilde{z}^t, \tag{23}$$

$$z^{t+1} = \Pi_{\mathcal{C}}(z^{t+1/2} + y^t), \tag{24}$$

$$\tilde{z}^{t+1} = \Pi_{[l,u]}(\tilde{z}^{t+1/2} + \tilde{y}^t), \tag{25}$$

$$y^{t+1} = y^t + z^{t+1/2} - z^{t+1}, \tag{26}$$

$$\tilde{y}^{t+1} = \tilde{y}^t + \tilde{z}^{t+1/2} - \tilde{z}^{t+1}, \tag{27}$$

where

$$M = P + \rho(A^\top A + B^\top B).$$

The matrix M is factorized once unless ρ changes.

6.1 Unrolled differentiation

The simplest differentiable CVQP layer unrolls T ADMM iterations and applies autodiff to the iteration map. Our projection VJP supplies the missing component. This approach is easy to implement and often sufficient for learning, but its memory cost grows with T unless checkpointing is used. Unrolling is also the correct derivative when the layer intentionally returns an early-stopped ADMM iterate rather than an optimizer.

6.2 Implicit differentiation of the active-face KKT system

For accurate solves, the backward pass should differentiate the optimizer, not the ADMM iteration. At a solution $x^\star$, recover the active face of the CVaR constraint at $z^\star = Ax^\star$. Let

$$G_{\text{cvar}} z = h_{\text{cvar}}$$

be the corresponding equality description, with no rows if the CVaR constraint is inactive with positive slack. Let

$$C_{\text{box}} x = b_{\text{box}}$$

collect the active lower and upper box constraints on Bx, with signs chosen so each active row is an equality. On a fixed active face, the selected local problem is the equality-constrained QP

$$\begin{array}{ll} \text{minimize} & \frac{1}{2} x^\top P x + q^\top x \\ \text{subject to} & Cx = b, \end{array} \qquad C = \begin{bmatrix} G_{\text{cvar}} A \\ C_{\text{box}} \end{bmatrix}, \quad b = \begin{bmatrix} h_{\text{cvar}} \\ b_{\text{box}} \end{bmatrix}. \tag{28}$$

In the no-tie integer CVaR case, $G_{\text{cvar}} = \mathbf{1}_T^\top$ and $h_{\text{cvar}} = k\kappa$, so the full CVaR epigraph collapses to one active row. In tied cases, G_{cvar} is applied with the structured averaging operator from the projection VJP rather than by forming dense plateau equality matrices.

The KKT conditions for (28) are

$$Px^\star + q + C^\top \nu^\star = 0, \qquad Cx^\star = b. \tag{29}$$

Their linearization is

$$\begin{bmatrix} P & C^\top \\ C & 0 \end{bmatrix} \begin{bmatrix} dx \\ d\nu \end{bmatrix} = - \begin{bmatrix} dP\, x^\star + dq + dC^\top \nu^\star \\ dC\, x^\star - db \end{bmatrix}. \tag{30}$$

This is the same implicit-differentiation route used by QP layers: the novelty is that the CVaR contribution to C is the active face of the projection, not the deterministic-equivalent epigraph.

Theorem 3 (Local derivative of the converged CVQP layer)**.** *Suppose* (20) *has a primal-dual solution* $(x^\star, \nu^\star)$ *whose CVaR and box active faces remain locally constant. Suppose the active constraint matrix* C *has full row rank after redundant rows are removed, and the reduced KKT matrix in* (30) *is nonsingular. Then the selected optimizer map is locally differentiable with respect to smooth problem parameters, and its sensitivities are given by* (30)*. Reverse-mode VJPs are obtained by solving the transposed reduced KKT system and applying the adjoint of the residual map in* (29)*. No deterministic-equivalent CVaR epigraph variables are introduced.*

Proof. On a region where the active face is fixed, (20) reduces locally to the equality-constrained QP (28). The stated rank and nonsingularity assumptions are the standard regularity conditions for applying the implicit function theorem to the KKT equations (29). Differentiating those equations gives (30); transposing the same linear system gives the reverse-mode adjoint. The CVaR projection theorem supplies the selected active-face operator and the derivatives of h_{cvar} with respect to d and κ. □

Conditioning. The reduced KKT route can still be ill-conditioned or rank deficient, especially when active box rows are redundant or plateau constraints nearly duplicate other active rows. The exact derivative should therefore use a rank-revealing factorization and remove redundant rows of C. If the reduced KKT matrix remains poorly conditioned, a Tikhonov or Levenberg–Marquardt regularized adjoint solve can be exposed as a damped surrogate derivative, analogous to the damped projection mode below. This regularized solve is a numerical safeguard, not the exact selected KKT derivative; for intentionally inexact ADMM solves, unrolling remains the faithful derivative.

7 Batched and GPU implementation

The projection layer is naturally batched. For a batch

$$\{v^{(b)} \in \mathbb{R}^{m_b}\}_{b=1}^{B},$$

the forward pass applies the chosen CVaR projection oracle to each segment and records an active-face certificate. The backward pass applies the cached face projector independently to each segment.

CPU implementation. The CPU implementation uses:

1. parallel projection over batch elements,
2. contiguous storage of sorted values, inverse permutations, or sort-free certificates,
3. a compact active-face cache storing group starts, group lengths, and tail weights,
4. vectorized VJP kernels for no-tie and grouped-tie cases.

GPU implementation. If the forward oracle is sorting-based, the GPU implementation uses segmented radix sort or library segmented sort followed by a fused projection-sweep kernel. A sort-free oracle would instead provide the same active certificate without a sort. The backward pass is bandwidth bound: it computes group sums, averages tied groups, applies the weighted tail-sum correction in compressed group coordinates, and scatters corrections. In a no-tie region, the backward kernel reduces to one tail reduction and one scatter-subtract operation per batch element.

Warm-started sorting. Inside ADMM or model-predictive-control loops, consecutive projection inputs often change slowly. We therefore store the previous sorted order. A cheap verification pass checks whether the order remains valid or nearly valid. If few inversions occur, insertion-style repair can be faster than a cold sort. This is the same kind of sequential-projection setting targeted by Roth and Cui's approximate-sorting procedure [24]; it affects only sorting-based forward passes, while the backward pass remains linear.

8 Handling nondifferentiability

The layer should be mathematically explicit about nondifferentiability rather than hiding it. We propose three modes.

Face mode. Return the Jacobian associated with the active face discovered in the forward pass. This is the default and is analogous to returning the derivative of a chosen branch of a piecewise-linear operator. The implementation records the threshold used to identify plateaus and active constraints. When the selected face is the limit of differentiable neighboring regions, this Jacobian is an element of the B-subdifferential, and hence of the Clarke generalized Jacobian.

Damped mode. For highly degenerate ties, use

$$D_\epsilon = I - G^\top (GG^\top + \epsilon I)^{-1} G,$$

with small $\epsilon > 0$. This improves numerical stability while converging to the face projector as $\epsilon \downarrow 0$. It is a Tikhonov surrogate and is not, for fixed ϵ, an exact element of the generalized Jacobian.

Subgradient sampling mode. When many active top-k sets are tied, sample one compatible active set and return its Jacobian. This is a stochastic selected derivative, not an unbiased estimator of a canonical generalized gradient. If the sampled active set corresponds to a neighboring differentiable region, it returns an element of the B-subdifferential. It can still be useful as an exploratory training heuristic when ties are induced by quantization.

These modes differ from superquantile smoothing, perturbed optimizers, and Fenchel-Young regularization. Those approaches give continuous derivatives of perturbed or regularized maps; face mode gives the exact selected derivative of the nonsmoothed projection on fixed faces but is discontinuous across active-face changes.

9 Numerical experiments

Setup. All experiments were run on a single NVIDIA A100-SXM4 (40 GB) on Lambda Cloud (Ubuntu 22.04, Python 3.10.12, NVIDIA driver 580.105.08). The primitive is implemented in NumPy (CPU) and PyTorch 2.5.1 (GPU). Ground-truth projections use CVXPY 1.7.5 [11] with the

CLARABEL solver at high accuracy; the differentiable epigraph baseline uses CVXPYlayers 0.1.9 (diffcp 1.1.9). GPU timings use warm-up calls, CUDA synchronization or CUDA events, and medians over repeated runs. All code, the exact pinned environment, and a one-command reproduction are released. We report five parts: a constraint-targeting experiment at scale (the application headline), projection microbenchmarks, an end-to-end training demonstration at $m = 10^5$, a small decision-focused composition check, and a robustness and failure analysis. The constraint-targeting and training experiments were run from a frozen execution protocol with a dated amendment log (`docs/EXPERIMENT_EXECUTION_SPEC.md` in the released code); their full stage consumed 21.8 hours on the single A100 above, and all stages resume idempotently from their CSV logs.

9.1 Hard CVaR constraints versus penalties at scale

The microbenchmarks of Section 9.2 establish what the primitive costs; this subsection establishes why solving — and, for learning pipelines, differentiating — at large m matters for the resulting *decisions*. The deployment question is: if a tail-risk budget κ must hold out of sample, what is gained by enforcing $\phi_\beta(z(w)) \le \kappa$ as a hard constraint, rather than adding a calibrated penalty $\lambda\,\phi_\beta(z(w))$ to the objective? The protocol — data generator, calibration rules, tolerances, seed plan, directional hypotheses, and the four headline contrasts — was specified and frozen before the run; the protocol document and its dated amendment log ship with the released code (`docs/EXPERIMENT_EXECUTION_SPEC.md`).

Protocol. Each test instance is a CVaR-constrained Markowitz problem over $n = 50$ assets: maximize $\mu_j^\top w - \frac{\gamma}{2}\|w\|_2^2$ subject to $\phi_\beta(z_j(w)) \le \kappa$, $\mathbf{1}^\top w = 1$, and $0 \le w \le 0.2$, with $\gamma = 1$, $\beta = 0.99$, and per-instance scenario losses $z_j(w) = L_{\text{shock}} w - (\mu_j^\top w)\mathbf{1}$ built from a shared zero-mean shock bank (five-factor model with standardized Student-t shocks and heterogeneous instance means μ_j). Decisions are computed from a predicted bank of $m \in \{10^3, 10^4, 10^5\}$ scenarios and evaluated on a held-out bank of $M_{\text{eval}} = 10^6$ scenarios, in two regimes: in distribution (ID, t_8 shocks) and shifted (t_3 tails, volatility $\times 1.25$, risk premium $\times 1.75$). Four methods are compared. *Hard*: the CVQP (20), solved by batched ADMM through the projection primitive. *Fixed-λ*: the exact top-k-sum penalty (no smoothing), with λ calibrated per (seed, m, κ, repeat) on ID validation so the penalty's mean predicted CVaR matches the hard method's. *Oracle-λ*: recalibrated on shifted validation — an information advantage no deployed system has. *Floor*: unconstrained mean–variance. Two budgets $\kappa_1 = 0.722$ and $\kappa_2 = 1.022$ were frozen by a blind pilot allowed to examine only constraint activity, feasibility, and numerical stability (never method separation); a pre-authorized scope rule stops κ_2 at $m = 10^4$. The run uses 5 seeds (a screening run: all intervals below are *screening intervals* from a 10,000-resample seed bootstrap), 192 test instances per seed, predicted-bank repeats $\{4, 2, 1\}$ at $m = \{10^3, 10^4, 10^5\}$ averaged within seed, and solver tolerance 10^{-4} with per-instance feasibility certificates. The solver-quality metric pred_violation $= (\phi_\beta(\hat{z}) - \kappa)_+$ on the predicted bank separates solver failure from out-of-sample risk failure: it is below 10^{-5} on every hard solve, so everything that follows is a property of the decisions, not of solver slack.

Realized risk converges to budget only under the hard constraint. Figure 1 and Table 1 give the headline. In distribution, both calibrated methods sit near budget at large m (hard $1.174 \to 1.010 \to 1.000$; fixed-$\lambda$ $1.177 \to 1.028 \to 1.004$ at κ_1): this is expected, since λ is calibrated precisely there, and it is the honest baseline operating point. The separation is under shift. At $m = 10^5$ the fixed multiplier realizes $1.763\times$ budget [1.667, 1.879] with exceedance rate 1.00 and mean positive exceedance equal to 76% of the budget, while the hard constraint realizes $1.012\times$ [1.008, 1.017] with mean positive exceedance of 1.3% of budget — a $60\times$ reduction in overshoot

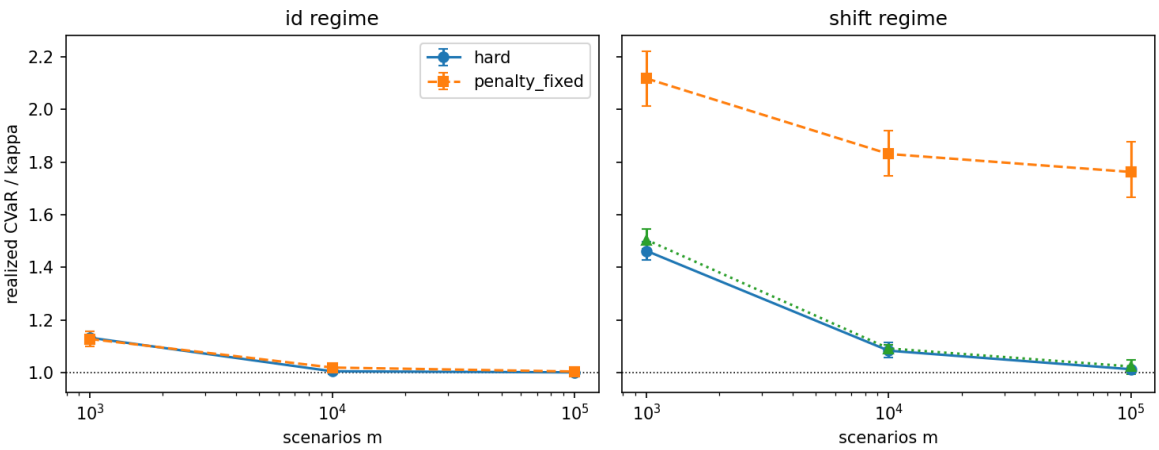

Figure 1: Realized out-of-sample CVaR over budget versus scenario count m, in distribution (left) and under shift (right); dotted line marks the budget. The hard constraint converges to its budget from above as the sample tail fills in; the ID-calibrated fixed-λ penalty does not recover under shift; the oracle-retuned penalty (green, shift panel) repairs the mean. Curves pool the two budget levels where both are run (κ_2 stops at $m = 10^4$ by the pre-authorized scope rule); Table 1 reports κ_1 separately. 5 seeds; error bars are screening intervals from a seed bootstrap.

magnitude at the same nominal target. The hard ratio falls from 1.546 ($m = 10^3$) through 1.100 (10^4) to 1.012 (10^5); convergence to budget, from above, was the frozen hypothesis, and the observed direction matches. The oracle-retuned penalty repairs the shifted mean (1.023 [0.995, 1.049]) but retains 3.3% mean overshoot and a 0.68 exceedance rate; the unconstrained floor realizes 2.1× (ID) to 3.8× (shift) budget throughout.

Mechanism: one multiplier cannot target heterogeneous instances. Figure 2 shows the predicted-CVaR side. The hard method pins every active instance at the budget — per-seed dispersion of ϕ_β/κ at most 2.4×10^{-5}, which is the constraint itself, enforced exactly by the projection — whereas the fixed-λ penalty disperses across instances (0.087 ID, 0.127 shift at $m = 10^5$) and drifts under shift; the oracle repairs the center but not the spread (0.041). Figure 3 shows why: the per-instance CVaR multipliers $\nu^\star$ recovered from the hard solves spread over an order of magnitude and move with the regime, so no single frozen λ — the penalty's operating assumption — can reproduce them. The optimism gap (realized minus predicted CVaR) isolates the finite-sample mechanism: for the hard method it decays from 0.125 to 0.0004 (ID) and from 0.394 to 0.0089 (shift) as m grows from 10^3 to 10^5, and because predicted CVaR sits at κ exactly, realized excess *equals* this gap (the decomposition realized excess = optimism gap + pred_violation holds with the second term ≈ 0). The hard method's residual 94% exceedance *rate* under shift at $m = 10^5$ is this same finite-m optimism — 1.3% of budget in magnitude — not a targeting failure: the rate contrast with the penalty becomes significant only at $m = 10^5$, while the magnitude and dispersion contrasts separate at every m (Table 2).

Frozen contrasts. Table 2 reports the four pre-specified seed-level paired contrasts. All four land in the frozen direction; with 5 seeds these are screening estimates, and a ten-seed confirmatory rerun is planned before any stronger claim.

method	m	active	feas.	exceed. rate	pos. exc. (% of κ)	realized/κ
ID test regime						
hard constraint	10^3	1.00	1.00	0.995	17.4	1.174
hard constraint	10^4	1.00	1.00	0.649	1.5	1.010
hard constraint	10^5	0.99	1.00	0.499	0.5	1.000
fixed-λ penalty	10^3	0.00	1.00	0.926	18.1	1.177
fixed-λ penalty	10^4	0.00	1.00	0.582	5.3	1.028
fixed-λ penalty	10^5	0.00	1.00	0.507	3.7	1.004
unconstrained floor	10^3	0.00	1.00	1.000	110.2	2.102
unconstrained floor	10^4	0.00	1.00	1.000	109.9	2.099
unconstrained floor	10^5	0.00	1.00	1.000	111.0	2.110
shifted test regime						
hard constraint	10^3	1.00	1.00	1.000	54.6	1.546
hard constraint	10^4	1.00	1.00	0.996	10.0	1.100
hard constraint	10^5	0.98	1.00	0.938	1.3	1.012
fixed-λ penalty	10^3	0.00	1.00	1.000	120.4	2.204
fixed-λ penalty	10^4	0.00	1.00	1.000	82.2	1.822
fixed-λ penalty	10^5	0.00	1.00	1.000	76.3	1.763
oracle-λ penalty	10^3	0.00	1.00	1.000	59.0	1.590
oracle-λ penalty	10^4	0.00	1.00	0.960	11.9	1.119
oracle-λ penalty	10^5	0.00	1.00	0.683	3.3	1.023
unconstrained floor	10^3	0.00	1.00	1.000	278.9	3.789
unconstrained floor	10^4	0.00	1.00	1.000	279.7	3.797
unconstrained floor	10^5	0.00	1.00	1.000	280.4	3.804

Table 1: Exceedance summary at $\kappa_1 = 0.722$ (the level run at every m; κ_2, run for $m \leq 10^4$, shows the same pattern — full CSVs ship with the tagged release). "Active"/"feas." are the hard constraint's activity and feasibility rates; positive exceedance is the mean of (realized CVaR $- \kappa)_+$ as a percentage of the budget. 5 seeds.

Tuning cost. The hard method requires one solve per target. The penalty requires a λ search per (seed, m, κ, repeat) — median 9 solver evaluations (range 4–11) on a 48-instance calibration subset, reaching median calibration error 0.3–0.6% of target (Table 3). Per-decision wall-clock favors the penalty (1.5 s versus 6.8 s at $m = 10^5$): exact targeting is what the hard solve's extra iterations buy, and the experiment's claim is risk placement, not solve speed.

Return at matched risk. Figure 4 traces mean realized return against realized risk under shift. At matched realized risk, the hard-constraint points lie on the same frontier as the oracle-retuned penalty: at this screening precision, exact budget targeting is not bought with a visible return sacrifice. We do not claim return dominance; the primary claim is controllable risk placement.

9.2 Projection microbenchmarks

We evaluate the differentiable projection primitive along four axes:

1. correctness against CLARABEL and finite differences, including projection-created plateaus,
2. CPU and GPU scaling of the forward projection and the selected backward pass,
3. a head-to-head comparison with the CVXPYlayers/diffcp epigraph baseline, and

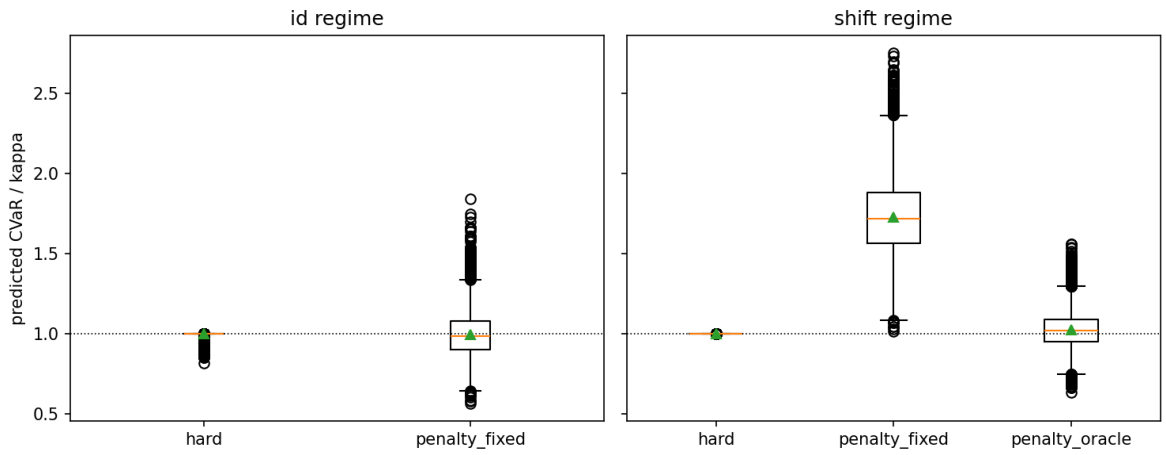

Figure 2: Predicted CVaR over budget across test instances at $m = 10^5$, κ_1 (boxes: IQR; triangles: means; 5 seeds pooled). The hard constraint places every active instance exactly at budget; the fixed-λ penalty disperses and, under shift, drifts; the oracle-retuned penalty re-centers but keeps the dispersion.

paired contrast (seed level, κ_1)	estimate	95% screening CI
hard − fixed-λ: exceedance rate (shift, $m = 10^5$)	-0.062	$[-0.118, -0.006]$
hard − fixed-λ: mean positive exceedance / κ (shift, $m = 10^5$)	-0.750	$[-0.868, -0.650]$
hard − fixed-λ: predicted-ratio dispersion (shift, $m = 10^5$)	-0.127	$[-0.161, -0.100]$
hard: realized ratio, $m = 10^3$ − $m = 10^5$ (ID)	$+0.173$	$[+0.148, +0.200]$
hard: realized ratio, $m = 10^3$ − $m = 10^5$ (shifted)	$+0.533$	$[+0.488, +0.574]$

Table 2: Pre-specified headline contrasts with seed-bootstrap percentile intervals (10,000 resamples; 5 seeds, screening run). Negative values favor the hard constraint on the first three rows; positive values on the last two confirm convergence of the hard method's realized ratio as m grows.

4. robustness to heavy-tailed data, active-set thresholding, and forward certificate perturbations.

   Metrics:

   forward time, backward time, peak memory, relative VJP error.

The head-to-head regime is $m = 10^3$ to 10^5, where the epigraph baseline can still be attempted. We report forward and backward timings separately because the forward speedup is inherited from specialized CVaR projection literature, whereas the backward column isolates the contribution of this paper. We then extend the specialized projection to $m = 2 \times 10^8$, where the generic epigraph baseline is unavailable, and validate with random-direction central finite differences and adjoint consistency checks.

Correctness. Against CLARABEL ground truth the forward projection agrees to 3.5×10^{-9} (20 random instances). The vector-Jacobian product matches central finite differences to 2.0×10^{-9} on generic no-tie instances (60 trials) and to 3.5×10^{-9} on projection-created plateaus (30 trials, plateau blocks up to size 24); the adjoints with respect to d and κ match to 5.3×10^{-7}. The GPU (PyTorch) implementation reproduces the NumPy reference to 7×10^{-16} (forward) and 6×10^{-17} (VJP), and a self-check enforces this agreement before every timing run.

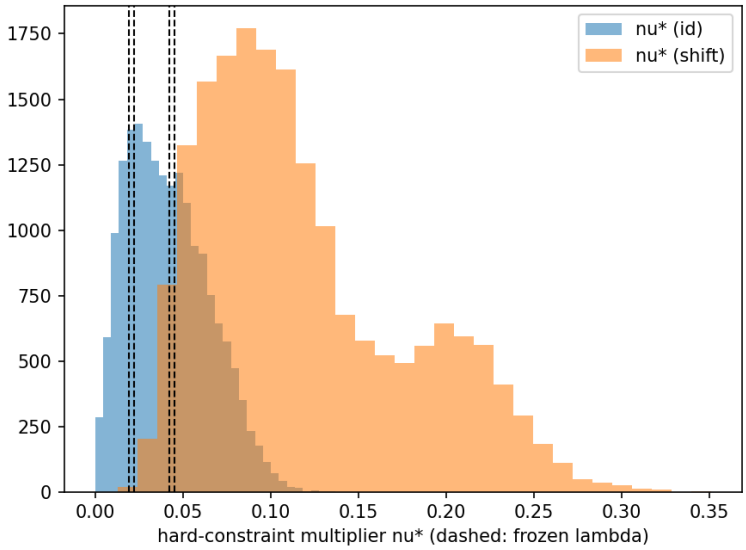

Figure 3: The mechanism: per-instance hard-constraint multipliers $\nu^\star$ (ID and shifted test), with the frozen calibrated λ values as dashed vertical lines. A single multiplier cannot match a regime-dependent distribution of per-instance multipliers. 5 seeds pooled.

	solves / evals per target			cal. error (% of κ)		median wall / decision (s)		
m	hard	fixed	oracle	fixed	oracle	hard	fixed	oracle
10^3	1	9	10	0.47	0.40	0.41	0.07	0.09
10^4	1	8	9	0.43	0.40	0.62	0.15	0.18
10^5	1	9	9	0.55	0.27	6.82	1.54	1.83

Table 3: Tuning and solve cost (medians over seeds, budgets, and repeats; 5 seeds). "Evals" are penalty solves consumed by the λ calibration search on its 48-instance subset; calibration error is the residual mismatch of mean predicted CVaR to its target.

Scaling. Table 4 reports single-instance GPU scaling. The backward pass is $O(m)$ with relative finite-difference error $\sim 10^{-10}$ across more than four orders of magnitude in m; a 200-million-scenario CVaR projection is differentiated in 95.9 ms forward + 18.6 ms backward using 19.8 GB of the 40 GB device. On CPU the backward pass scales identically (Figure 5, left), from 7.6 μs at $m = 10^3$ to 20.4 ms at $m = 10^7$. At $m = 10^7$ the GPU backward is roughly 18× faster than the NumPy CPU backward (1.12 ms versus 20.4 ms), and the end-to-end forward-plus-backward gap exceeds two orders of magnitude because the CPU forward is sort-dominated. Batched over independent problems, the GPU primitive reaches 1.9×10^5 projections/s at batch size 4096, $m = 10^4$.

Head-to-head versus epigraph differentiation. Table 5 and Figure 6 compare the proposed primitive to differentiating the deterministic-equivalent CVaR epigraph with CVXPYlayers/diffcp. This is an older CPU conic baseline, chosen because it is the standard deterministic-equivalent route and because it exposes the $O(m)$ epigraph blowup directly. For a like-for-like comparison, the proposed primitive is also run on the host CPU (the NumPy implementation) in this experiment, so the table isolates the formulation rather than the hardware; GPU timings for the proposed primitive appear in Table 4. Newer CVXPYlayers releases provide additional GPU backends [9], and a custom

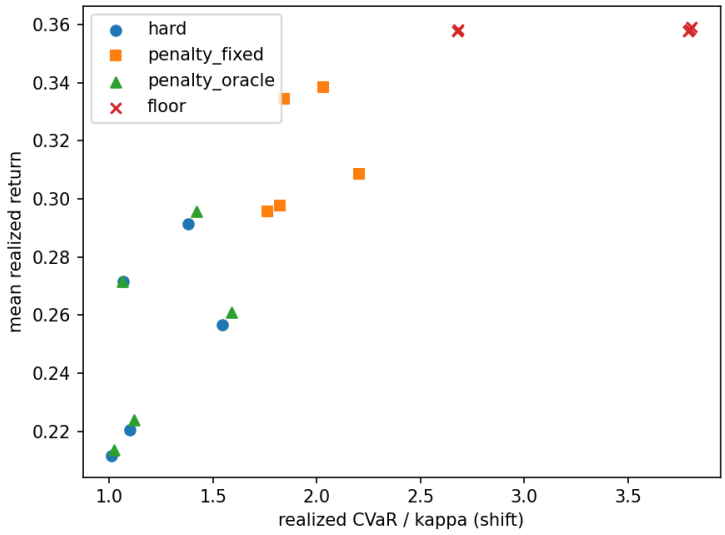

Figure 4: Mean realized return versus realized CVaR over budget (shifted test; points are (method, m, κ) cells; 5 seeds). Hard-constraint and oracle-penalty points share a frontier; the fixed-λ penalty and the floor buy return with uncontrolled tail risk. Secondary, descriptive evidence only.

m (scenarios)	forward (ms)	backward (ms)	peak GPU mem	rel. VJP error
10^4	0.46	0.33	1.0 MB	2.6×10^{-10}
10^5	0.46	0.33	9.9 MB	1.3×10^{-10}
10^6	0.73	0.35	101 MB	4.0×10^{-10}
10^7	5.49	1.12	991 MB	1.6×10^{-10}
10^8	47.9	9.37	9.9 GB	1.2×10^{-10}
2×10^8	95.9	18.6	19.8 GB	1.5×10^{-10}

Table 4: Single-instance CVaR projection and its VJP on one A100 (float64). The backward pass is $O(m)$; a 200M-scenario projection is differentiated in < 0.12 s. VJP error is relative to a random-direction central finite difference.

sparse-QP layer could reduce constants; those baselines are not evaluated here. The comparison should therefore be read as a test of the epigraph formulation's scaling, not as a claim against every possible generic implementation.

The gradients agree to 7×10^{-6} ($m = 200$) and 4×10^{-5} ($m = 1000$), confirming that the specialized VJP computes the same selected derivative. The epigraph layer is much slower in both phases. More importantly for this paper, the standalone backward pass is orders of magnitude slower than the active-face VJP; its peak memory grows to 42 GB at $m = 3 \times 10^4$; and it fails (300 s timeout or out-of-memory) at $m = 10^5$, exactly the regime where the specialized forward solver is useful. The proposed layer's resident memory is dominated by the framework runtime (≈ 0.5 GB) and is flat in m.

9.3 End-to-end training at $m = 10^5$

This demonstration validates that the primitive composes into a trainable layer at the scenario scale the primary experiment says matters. A scenario bank at $m = 10^5$ is parametrized by $\theta \in \mathbb{R}^8$ in

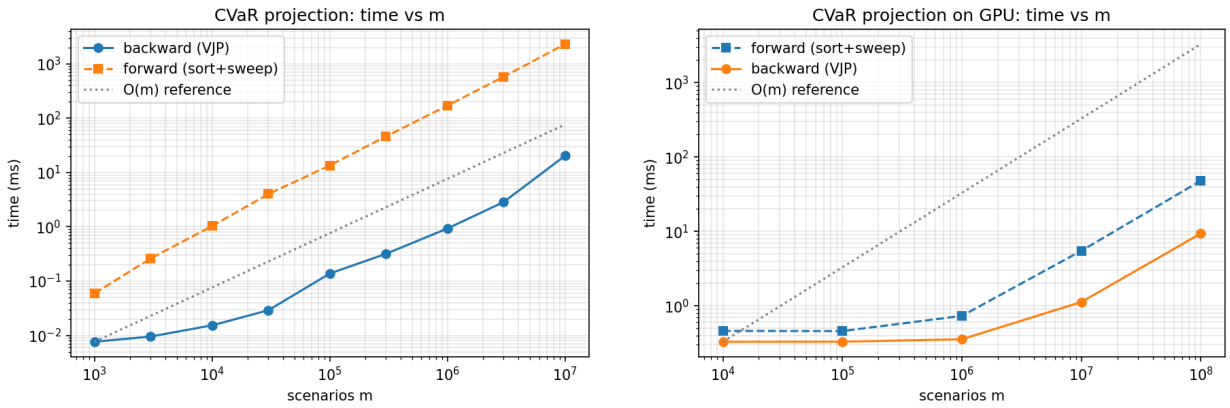

Figure 5: Forward and backward time versus scenario count m, on CPU (left, to $m = 10^7$) and on one A100 (right, to $m = 10^8$). The backward pass tracks the $O(m)$ reference in both cases.

m	prop. fwd	prop. bwd	epi. fwd	epi. bwd	epi. peak mem
10^3	0.107	0.007	123	356	0.5 GB
3×10^3	0.324	0.009	2,934	417	0.7 GB
10^4	1.176	0.015	43,040	2,089	5.2 GB
3×10^4	3.925	0.029	152,700	14,460	42 GB
10^5	14.52	0.157	timeout / out-of-memory		

Table 5: Proposed primitive versus differentiating the deterministic-equivalent epigraph (CVXPYlayers/diffcp) for the CVaR projection map; both methods run on the same host CPU. Times are in milliseconds and split into forward and backward phases. The forward speedup is not the paper's novelty; the backward column isolates the active-face VJP. The epigraph baseline still completes at $m = 3 \times 10^4$ but its peak memory reaches 42 GB there, and it times out or runs out of memory at $m = 10^5$; the proposed primitive continues (cf. Table 4).

rank-one operator form ($L_\theta w = Lw + \sum_p \theta_p u_p (v_p^\top w)$; the dense perturbation is never materialized), and θ is trained for 100 Adam steps to minimize a realized objective on a held-out validation bank, through a batched CVQP layer (32 instances per step, $n = 50$, $\beta = 0.99$, binding κ chosen per seed by a blind midpoint rule). The layer unrolls $T = 50$ ADMM iterations (Section 9 solver) started from a per-step, detached, non-differentiated warm solve at the current parameters (working tolerance 10^{-4}): the unrolled truncated adjoint is therefore evaluated at the solved point, the regime where it approximates the implicit derivative of Theorem 3, and every unrolled projection sits at the binding face where the certificate VJP — including its plateau branch, which is generic at the binding optimum — is exercised. The training loss exists to drive gradients through the layer; per the frozen protocol, no decision-quality claim is attached to it.

Table 6 summarizes. Instrumented use counters report a mean active-projection fraction of 1.00 on every seed (the demonstration exercises the contribution, rather than an identity branch), and $\|\nabla_\theta\| > 0$ at every logged step. A two-scale central finite-difference check of $\partial\,\text{loss}/\partial\theta$ through the full unrolled layer at a smaller binding instance ($m = 2000$; per-direction minimum over steps $\varepsilon \in \{10^{-6}, 10^{-5}\}$) gives maximum relative error 1.7×10^{-5} against an acceptance gate of 10^{-4} set by the finite-difference noise floor of the bisection-truncated forward; the primitive's own VJP is verified to 2–4 $\times 10^{-9}$ in Section 9.2. A learnable-budget variant ($\kappa = \kappa_{\min} + \text{softplus}(\eta)$, 20 steps per seed) exercises the d/κ adjoint end to end with $|\partial_\eta\,\text{loss}| \geq 0.24$ throughout. Median step time is

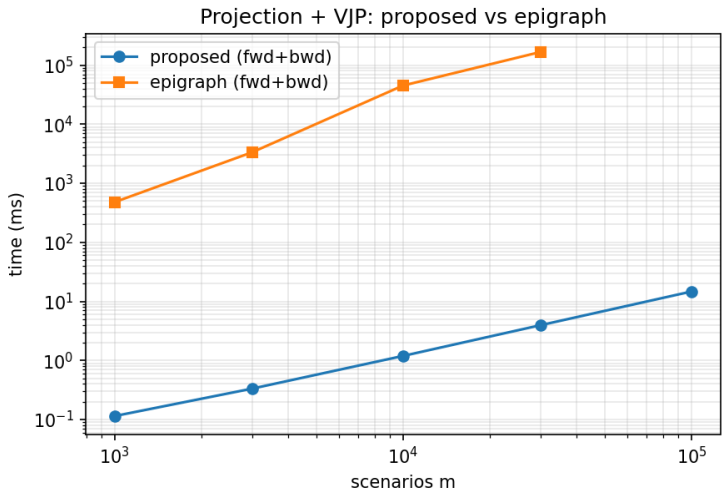

Figure 6: Forward+backward time for the CVaR projection map: proposed primitive versus deterministic-equivalent epigraph differentiation. The epigraph baseline is absent beyond $m = 3\times10^4$, where it runs out of time/memory.

seed	loss (step 0)	loss (step 99)	mean active frac.	med. step (s)	peak mem (GB)
0	0.9178	0.9177	1.00	5.81	1.24
1	0.9943	0.9943	1.00	5.80	1.29
2	0.9041	0.9037	1.00	5.81	1.29

Table 6: End-to-end training through the CVQP layer at $m = 10^5$ (100 Adam steps per seed, 32 instances per step, $T = 50$ unrolled iterations from a per-step detached warm solve). The loss is a gradient-flow target, not a decision-quality claim. 3 seeds.

5.8 s — dominated by the eager warm-start forward solve, not differentiation (the projection VJP itself costs ~ 0.3 ms at this m, Table 4) — within 1.3 GB of GPU memory. As a negative control, one backward pass through the CVXPYlayers/diffcp epigraph at the same $m = 10^5$ shape was attempted in a sandboxed process under the head-to-head protocol's 300 s cap: the process was terminated after 94 s due to memory exhaustion, consistent with Table 5.

9.4 Composition check: decision-focused portfolio learning

This experiment is a small-scale composition and correctness check of the CVQP layer, not a large-m deployment study; it asks whether the primitive trains end to end, remains feasible, and computes verifiable gradients when embedded in an unrolled solver. A predictor maps market features to return scenarios. The optimization layer chooses portfolio weights by solving a CVaR-constrained Markowitz problem. The training objective is realized out-of-sample utility or drawdown-adjusted return. We compare two ways of fitting the same predictor: supervised return loss followed by optimization (two-stage), and warm-started end-to-end training through the differentiable CVQP layer.

Results. We instantiate the pipeline with $n = 25$ assets, $m = 300$ loss scenarios, CVaR level $\beta = 0.95$ and budget $\kappa = 0.20$ (which binds), and a linear return predictor that is deliberately misspecified

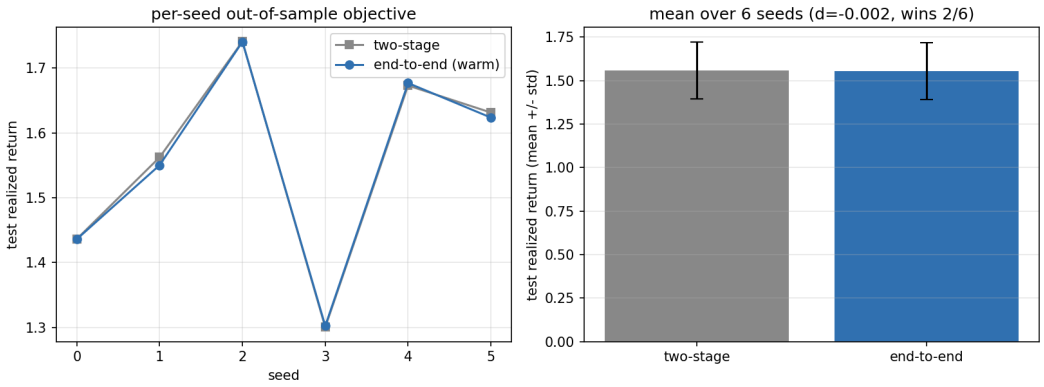

Figure 7: Decision-focused CVQP layer, six seeds. Left: per-seed out-of-sample realized return; the two-stage and warm-started end-to-end curves overlap. Right: means ± one standard deviation. End-to-end matches two-stage within noise, and both decisions satisfy the enforced CVaR budget.

against a nonlinear ground truth (a regime with $\sim 30\%$ oracle-versus-two-stage headroom). We fit the same predictor two ways: supervised mean-squared error (two-stage), and end-to-end through the differentiable CVQP layer, the latter *warm-started* at the two-stage solution and early-stopped on a held-out validation split. Over six seeds (Figure 7), end-to-end matches two-stage within noise: test realized return 1.557 ± 0.163 versus 1.555 ± 0.162, paired difference -0.0025 ± 0.0062 (end-to-end wins on two of six seeds; paired $t = -0.98$). Both satisfy the tail-risk budget (mean realized CVaR 0.201). A gradient check through the full unrolled layer passes, and the unrolled forward agrees with CVXPY to 1.6×10^{-2}.

We report this as evidence that the primitive composes into a *correct, feasible, end-to-end-trainable* CVQP layer, not as a claim that decision-focused training beats two-stage on this instance: under a fair protocol (equal budget, warm start, validation-based early stopping) the two are statistically indistinguishable here. This is consistent with the local nature of the gain — the oracle headroom is global, but the decision loss is locally flat around the two-stage optimum. Whether decision-focused training yields a robust improvement in some risk-constrained regime is left to future work; the contribution validated here is the differentiable primitive and its feasibility, not a predict-then-optimize benchmark win.

9.5 Robustness and failure analysis

A production primitive must survive non-ideal inputs. We therefore include:

1. heavy-tailed and badly scaled scenarios,
2. projection-created plateaus from large violations,
3. active-set threshold sweeps,
4. gradient error versus forward solve tolerance,
5. early-stopped ADMM solves, with unrolled gradients separated from reduced KKT gradients.

Results. A central finite difference is a valid reference only inside a single normal-fan cell; across a cell boundary the map is nondifferentiable, so we restrict all finite-difference checks to certificate-

stable cells (the full-measure differentiable regime of Theorem 1) and separately record how often a perturbation lands on a boundary.

- *Heavy-tailed and badly-scaled inputs* (Student-t, $\nu = 2$, with data scaled across seven orders of magnitude): the forward projection agrees with CLARABEL to 4.8×10^{-5}, and the VJP matches finite differences to 1.5×10^{-9} on stable cells.
- *Projection-created plateaus*: over 80 stable plateau cells (block sizes 2–25) the grouped-plateau VJP of Lemma 1 matches finite differences to 5.2×10^{-9}.
- *Active-set threshold sweep*: the certificate, and hence the VJP, is exact for every tolerance below the data resolution (VJP error 2.5×10^{-9} for tolerances 10^{-9}–10^{-3}) and degrades only once the tolerance exceeds it and merges genuinely distinct entries — the quantitative form of Proposition 3.
- *Gradient error versus forward-solve error*: injecting error of magnitude δ into the projected point, the recovered VJP error decreases monotonically in δ and reaches zero once δ falls below the cell margin, again as predicted by Proposition 3.

For deliberately inexact, early-stopped ADMM solves, the faithful derivative is the unrolled one rather than the reduced-KKT derivative, which applies only to accurate solves. The decision-focused layer of the previous subsection trains through exactly such an unrolled solve and is verified by a full-layer gradient check.

10 Released software interface

The released repository exposes a small Python API for the projection primitive. A minimal PyTorch-style call is:

```
z = cvar_project(v, beta=0.95, kappa=kappa)
```

The released implementation covers the integer-tail case $k = (1 - \beta)m$ used in all experiments. The fractional-tail and β derivatives around (6)–(16) are derived in this paper but not implemented in the released code; they should be read as analysis, not as a released feature. The library exposes the lower-level routines `project_topk_sum`, `extract_certificate`, `vjp_projection`, batched NumPy variants, and GPU-native PyTorch routines for the no-plateau large-scale path. The certificate follows the single-boundary-group structure of Remark 1, including the endpoint normalization: a fully included boundary group ($q = g$, strict gap below) is merged into the strict set, so the released VJP returns the classical no-tie derivative there rather than a conservative averaged surrogate. The certificate's active/inactive decision is taken from the forward solve's violation status, per the boundary convention above. The tests cover forward agreement with CLARABEL, finite-difference VJPs, plateau cells (including exactly tied inputs that fill the tail, checked against the no-tie formula and finite differences), the zero-slack boundary convention, d/κ adjoints, and torch-versus-NumPy self-validation.

The demonstration CVQP layer used in the portfolio experiment has the interface:

```
layer = CVQPLayer(P, A, B, beta, kappa, l, u)
x = layer(q)
loss = downstream_loss(x)
loss.backward()
```

This layer differentiates by unrolling the ADMM iteration. The reduced active-face KKT backward of Theorem 3 is specified and derived here but is not part of the released implementation; implementing and benchmarking it at scale is future work, as noted in the conclusion.

The repository includes a one-command reproduction script, pinned dependencies, captured hardware/driver metadata, CSV outputs for every table, and figure generation scripts. The released code is MIT licensed. Code-generation support for CVaR-specialized kernels and PyTorch/JAX custom VJP bindings is outside the current release.

11 Discussion

The central thesis is that differentiability should be added at the primitive that actually made the forward solver scalable. For CVQP, that primitive is the CVaR projection, not the deterministic-equivalent QP. The forward projection itself is well studied; the missing systems primitive is the exact selected backward pass and active-face certificate. Once that backward rule is available, the projection can be reused in many contexts: ADMM, projected gradient, model-predictive control, robust learning, and risk-aware resource allocation.

The proposed method is intentionally simple. The forward pass calls a projection oracle and records an active-face certificate. The backward pass projects the adjoint onto the active tangent space. This simplicity also clarifies the scope: the practical contribution is narrow but useful — an exact selected backward pass for CVaR constraints and projections, with structured tie handling and batched kernels, rather than differentiable CVaR objectives, forward projection algorithms, or generic differentiable optimization infrastructure.

12 Limitations

The method returns a selected derivative at nondifferentiable points. This is standard in autodiff but should be documented together with the active-set thresholds. The reduced KKT CVQP derivative assumes an accurate solve, locally constant active faces, and a nonsingular reduced KKT matrix; it is not the right derivative for an early-stopped ADMM iterate. In early-stopped or deliberately inexact regimes, unrolled differentiation with checkpointing is more faithful. The tail level κ and fractional-face tail parameter β are differentiable only on fixed active faces; β remains nonsmooth at integer tail-mass crossings and tie events. While the projection backward pass is linear, the overall CVQP backward pass can still be dominated by original-problem linear algebra, especially when n is large and sparse factorization is expensive.

The experiments establish correctness, scaling, hard-constraint targeting at scale, and end-to-end trainability, but they do not yet show a large-m *learning* problem where differentiating the CVaR constraint improves downstream decisions over a strong two-stage or penalized baseline; the $m = 10^5$ training demonstration validates gradient flow and cost, not decision quality. The portfolio experiment is deliberately reported as a null result: it verifies that the layer trains and respects the tail budget, not that decision-focused training wins on that instance. We also do not compare against smoothed or penalized CVaR objectives on the same task. Such a comparison would be the right way to demonstrate the practical value of enforcing a hard tail-risk budget rather than optimizing a smooth surrogate. The epigraph baseline is CVXPYlayers/diffcp through the deterministic-equivalent cone formulation; newer GPU backends and custom sparse-QP baselines may reduce constants, although they do not remove the epigraph variables and constraints. The constraint-targeting study is a five-seed screening run on synthetic factor-model data: its bootstrap intervals are labeled screening intervals, a ten-seed confirmatory rerun at full pre-specified sample

sizes is planned, and the hard method's residual exceedance rate at $m = 10^5$ under shift (94% of instances, at 1.3%-of-budget magnitude) is finite-sample tail optimism, not solver error (predicted violations stay below 10^{-5}).

13 Conclusion

We proposed a differentiable CVaR projection primitive for risk-constrained optimization, and a reduced-KKT path for using it inside converged CVQP layers. The key observation is that the CVaR projection is piecewise affine and that the active face needed for the backward pass is recoverable from the projection output together with its violation status. This turns the projection backward pass into an $O(m)$ tangent-space projection once an active-face certificate is available. The result is a specialized primitive for differentiating through tail-risk constraints at scenario counts where generic differentiated epigraph formulations incur large constant factors and memory costs.

Empirically, the projection-input VJP matches finite differences to better than 10^{-8} (including degenerate plateaus), differentiates a 200-million-scenario CVaR projection in under 0.12 s on a single GPU with an $O(m)$ backward pass, and is several orders of magnitude faster and more memory-frugal than differentiating the deterministic-equivalent epigraph with the CVXPYlayers/diffcp baseline, which exhausts memory or times out near $m = 10^5$. In the pre-specified five-seed screening experiment, hard CVaR constraints solved through the projection placed realized out-of-sample tail risk on budget as m grew ($1.55 \to 1.01$ times budget under distribution shift), where the fixed and oracle-retuned penalty multipliers left 76% and 3.3% mean budget overshoot respectively — evidence that the scenario count is a decision-quality variable and that hard tail-risk constraints are worth differentiating. It also composes into a feasible, end-to-end-trainable CVQP layer, although the portfolio experiment does not show a decision-focused improvement over two-stage training. The natural next steps are a vectorized on-GPU plateau forward (the backward already handles plateaus on-GPU), the reduced-KKT path at scale, a hard-constraint comparison against smoothed or penalized CVaR objectives, a ten-seed confirmatory rerun of the constraint-targeting experiment at the full pre-specified sample sizes, and risk-constrained settings beyond finance.

Reproducibility. All code, the exact pinned software environment, captured hardware/driver metadata, and a single-command reproduction of every table and figure are available at `https://github.com/riyan-christy/diff-cvar-projection`. Every number in this paper is generated from the CSV files under `results/`, which the reproduction script regenerates together with the figures, an environment snapshot (`env.json`, including hardware, driver, and package versions), and an auto-generated results summary; the CSVs, not the text, are the source of truth. The primary and secondary experiments run as resumable stages of the same one-command pipeline, and the frozen experiment protocol — data generator, calibration rules, tolerances, seed plan, and dated amendment log — is included in the repository under `docs/EXPERIMENT_EXECUTION_SPEC.md`; the reference `results/` snapshot ships with the tagged release. A tagged release will be pinned at submission.

References

[1] A. Agrawal, B. Amos, S. Barratt, S. Boyd, S. Diamond, and J. Z. Kolter. Differentiable convex optimization layers. In *Advances in Neural Information Processing Systems*, 2019.

[2] A. Agrawal, S. Barratt, S. Boyd, E. Busseti, and W. M. Moursi. Differentiating through a cone program. *Journal of Applied and Numerical Optimization*, 1(2):107–115, 2019.

[3] B. Amos and J. Z. Kolter. OptNet: Differentiable optimization as a layer in neural networks. In *International Conference on Machine Learning*, 2017.

[4] Q. Berthet, M. Blondel, O. Teboul, M. Cuturi, J.-P. Vert, and F. Bach. Learning with differentiable perturbed optimizers. In *Advances in Neural Information Processing Systems*, 2020.

[5] M. Blondel, A. F. T. Martins, and V. Niculae. Learning with Fenchel-Young losses. *Journal of Machine Learning Research*, 21(35):1–69, 2020.

[6] M. Blondel, O. Teboul, Q. Berthet, and J. Djolonga. Fast differentiable sorting and ranking. In *International Conference on Machine Learning*, 2020.

[7] S. Boyd, N. Parikh, E. Chu, B. Peleato, and J. Eckstein. Distributed optimization and statistical learning via the alternating direction method of multipliers. *Foundations and Trends in Machine Learning*, 3(1):1–122, 2011.

[8] M. Cuturi, O. Teboul, and J.-P. Vert. Differentiable ranking and sorting using optimal transport. In *Advances in Neural Information Processing Systems*, 2019.

[9] CVXPYlayers developers. CVXPYlayers release notes and package documentation. Software documentation, 2026. https://github.com/cvxpy/cvxpylayers/releases.

[10] D. Davis. An $O(n \log(n))$ algorithm for projecting onto the ordered weighted ℓ_1 norm ball. arXiv preprint arXiv:1505.00870, 2015.

[11] S. Diamond and S. Boyd. CVXPY: A Python-embedded modeling language for convex optimization. *Journal of Machine Learning Research*, 17(83):1–5, 2016.

[12] P. L. Donti, B. Amos, and J. Z. Kolter. Task-based end-to-end model learning in stochastic optimization. In *Advances in Neural Information Processing Systems*, 2017.

[13] A. N. Elmachtoub and P. Grigas. Smart "predict, then optimize". *Management Science*, 68(1):9–26, 2022.

[14] A. Grover, E. Wang, A. Zweig, and S. Ermon. Stochastic optimization of sorting networks via continuous relaxations. In *International Conference on Learning Representations*, 2019.

[15] Y. Laguel, K. Pillutla, J. Malick, and Z. Harchaoui. Superquantiles at work: Machine learning applications and efficient subgradient computation. *Set-Valued and Variational Analysis*, 29:967–996, 2021.

[16] M. Lapin, M. Hein, and B. Schiele. Top-k multiclass SVM. In *Advances in Neural Information Processing Systems*, 2015.

[17] E. Luxenberg, D. Pérez-Piñeiro, S. Diamond, and S. Boyd. An operator splitting method for large-scale CVaR-constrained quadratic programs. *Optimization and Engineering*, 2026. doi:10.1007/s11081-026-10091-8. Also available as arXiv:2504.10814.

[18] A. F. T. Martins and R. F. Astudillo. From softmax to sparsemax: A sparse model of attention and multi-label classification. In *International Conference on Machine Learning*, 2016.

[19] J. Pan, Z. Ye, X. Yang, X. Yang, W. Liu, L. Wang, and J. Bian. BPQP: A differentiable convex optimization framework for efficient end-to-end learning. In *Advances in Neural Information Processing Systems*, 2024.

[20] J. Pan and M. Yan. Fast projection onto the top-k-sum constraint. arXiv preprint arXiv:2512.10255, 2025.

[21] K. Pillutla. SQwash: Distributionally robust learning in PyTorch with one additional line of code. Python package, 2021. https://pypi.org/project/sqwash/.

[22] R. T. Rockafellar and S. Uryasev. Optimization of conditional value-at-risk. *Journal of Risk*, 2:21–42, 2000.

[23] R. T. Rockafellar and R. J.-B. Wets. *Variational Analysis*. Springer, 1998.

[24] J. Roth and Y. Cui. On $O(n)$ algorithms for projection onto the top-k-sum sublevel set. *Mathematical Programming Computation*, 17(2):307–348, 2025. doi:10.1007/s12532-024-00273-9.

[25] M. Schaller and S. Boyd. Code generation for solving and differentiating through convex optimization problems. *Optimization and Engineering*, 27:1425–1449, 2026. doi:10.1007/s11081-025-10057-2.

[26] S. Scholtes. *Introduction to Piecewise Differentiable Equations*. SpringerBriefs in Optimization. Springer, 2012.

[27] B. Wilder, B. Dilkina, and M. Tambe. Melding the data-decisions pipeline: Decision-focused learning for combinatorial optimization. In *AAAI Conference on Artificial Intelligence*, 2019.

[28] Y. Xie, H. Dai, M. Chen, B. Dai, T. Zhao, H. Zha, W. Wei, and T. Pfister. Differentiable top-k with optimal transport. In *Advances in Neural Information Processing Systems*, 2020.