Mathematical Programming Computation
https://doi.org/10.1007/s12532-024-00273-9

FULL LENGTH PAPER

On $O(n)$ algorithms for projection onto the top-k-sum sublevel set

Jake Roth[1] · **Ying Cui**[2]

Received: 11 October 2023 / Accepted: 13 November 2024

Abstract

The *top-k-sum* operator computes the sum of the largest k components of a given vector. The Euclidean projection onto the top-k-sum sublevel set serves as a crucial subroutine in iterative methods to solve composite superquantile optimization problems. In this paper, we introduce a solver that implements two finite-termination algorithms to compute this projection. Both algorithms have $O(n)$ complexity of floating point operations when applied to a sorted n-dimensional input vector, where the absorbed constant is *independent of* k. This stands in contrast to an existing grid-search-inspired method that has $O(k(n-k))$ complexity, a partition-based method with $O(n + D \log D)$ complexity, where $D \leq n$ is the number of distinct elements in the input vector, and a semismooth Newton method with a finite termination property but unspecified floating point complexity. The improvement of our methods over the first method is significant when k is linearly dependent on n, which is frequently encountered in practical superquantile optimization applications. In instances where the input vector is unsorted, an additional cost is incurred to (partially) sort the vector, whereas a full sort of the input vector seems unavoidable for the other two methods. To reduce this cost, we further derive a rigorous procedure that leverages approximate sorting to compute the projection, which is particularly useful when solving a sequence of similar projection problems. Numerical results show that our methods solve problems of scale $n = 10^7$ and $k = 10^4$ within 0.05 s, whereas the most competitive alternative, the semismooth Newton-based method, takes about 1 s. The existing grid-search method and Gurobi's QP solver can take from minutes to hours.

Keywords Projection · Top-k-sum · Superquantile · Z-matrix

✉ Ying Cui
yingcui@berkeley.edu

Jake Roth
roth0674@umn.edu

1 Department of Industrial and Systems Engineering, University of Minnesota, Minneapolis, MN 55414, USA

2 Department of Industrial Engineering and Operations Research, University of California, Berkeley, Berkeley, CA 94720, USA

Published online: 08 January 2025

Mathematics Subject Classification 90-04 · 90-08 · 90C06 · 90C20 · 90C25 · 90C33

1 Introduction

We consider the Euclidean projection onto the top-k-sum (also referred to as max-k-sum in various works such as [13, 17, 39]) sublevel set. Specifically, given a scalar budget $r \in \mathbb{R}$, an index $k \in \{1, 2, \ldots, n\}$, and an input vector $x^0 \in \mathbb{R}^n$, our aim is to develop a fast and finite-termination algorithm to obtain the exact solution of the strongly convex problem

$$\begin{aligned} &\underset{x \in \mathbb{R}^n}{\text{minimize}} \quad \frac{1}{2}\|x - x^0\|_2^2 \\ &\text{subject to} \quad x \in \mathcal{B}^r_{(k)} := \left\{ x \in \mathbb{R}^n : \mathsf{T}_{(k)}(x) := \sum_{i=1}^{k} \overleftarrow{x}_i \le r \right\}, \end{aligned} \tag{1}$$

where for any $x \in \mathbb{R}^n$, we write $\overleftarrow{x} \in \mathbb{R}^n$ as its sorted counterpart satisfying $\overleftarrow{x}_1 \ge \overleftarrow{x}_2 \ge \cdots \ge \overleftarrow{x}_n$, and $\mathsf{T}_{(k)}(x)$ represents the sum of the largest k elements of x.

The top-k-sum operator $\mathsf{T}_{(k)}(\bullet)$ is closely related to the *superquantile* of a random variable, which is also known as the *conditional value-at-risk* (CVaR) [34], *average-top-k* [22], *expected shortfall*, among other names. Specifically, consider X as a random variable. Its superquantile at confidence level $\tau \in (0, 1)$ is defined as $\mathsf{S}_\tau(X) := \min\{t + (1-\tau)^{-1}\mathbb{E}[\max(X-t, 0)]\}$. When X is supported on n atoms $x := (x_1, \ldots, x_n)^\top$, each with equal probability, then $\mathsf{T}_{(k)}(x)/k = \mathsf{S}_{\tau(k)}(X)$ averages the largest k realizations of X, where $\tau(k) := 1 - k/n$. In the context where X follows a continuous distribution, one may select n samples $x = \{x_i\}_{i=1}^n$ and construct an empirical sample average approximation of its superquantile at confidence $\tau = 1 - k/n$ using $\mathsf{T}_{(k)}(x)$.

Owing to the close relationship between the top-k-sum and the superquantile, the projection problem in (1) has applicability as a subroutine in solving composite optimization problems of the form

$$\underset{z}{\text{minimize}} \quad f(z) + \mathsf{S}_{\tau_0}\big(G^0(z; \omega_0)\big) \quad \text{subject to} \quad \mathsf{S}_\tau\big(G(z; \omega)\big) \le r \tag{2}$$

where f is a deterministic function, G^0 and G are random mappings that depend on both decision z and random vectors ω_0 and ω with finite supports, respectively; r is the sublevel-set parameter; and τ_0, τ are superquantile confidence parameters. Problem (2) addresses the empirical or sample average approximation of risk-averse CVaR problems, commonly used in safety-critical applications to manage adverse outcomes, such as in the robust design of complex systems [8, 14, 16, 24, 38]. Additionally, such problems arise from the convex approximation of chance constrained stochastic programs [9, 29], and are relevant to matrix optimization problems involving a matrix's Ky-Fan norm [30, 42], *i.e.,* the vector-k-norm of its (already sorted) singular values. Recently, optimization problems involving superquantiles have attracted significant attention in the machine learning community, proving instrumental in modeling problems which: (i) seek robustness against uncertainty, such as mitigating distributional shifts between training and test datasets [25], or measuring robustness through probabilistic guarantees on solution quality [33]; (ii) handle imbalanced data [32, 43]; or (iii) pursue notions of fairness [19, 26, 41]. Interested readers are encouraged to consult a recent survey [37] for a comprehensive review of superquantiles. A fast and reliable solver for computing the projection onto the top-k-sum sublevel set, especially when dealing with a large number of samples, is crucial for first- or second-order methods to solve the large-scale, complex composite superquantile problem (2). Interested readers are refereed to [36] for a recent study addressing this problem, which requires the projection oracle (1) in each iteration of the augmented Lagrangian method.

Given that problem (1) is a strongly convex quadratic program, it accommodates the straightforward use of off-the-shelf solvers, such as Gurobi, to compute its solution. However, numerical experiments indicate that Gurobi needs about 1-2 minutes at best to solve problems of size $n = 10^7$, thus preventing its use as an effective subroutine within an iterative approach to solve the composite problem (2); see Table 1 in Sect. 4 for details of the numerical results. In addition, generic quadratic programming solvers yield inexact solutions. This can lead to a challenge in precisely determining the (generalized) Jacobian associated with the projection operator (see [36, 42]), which may be needed in a second-order method to solve composite problems like (2). To overcome these issues, a finite-termination, grid-search-inspired method is introduced in [42], which has a complexity of $O(k(n-k))$ for a sorted n-dimensional input vector. In the context of composite problems such as (2), the value k is usually set as a fixed proportion of n, *i.e.,* $k = \lfloor (1-\tau)n \rfloor$ for an exogenous risk-tolerance $\tau \in (0, 1)$, resulting in $O(n^2)$ complexity in many practical instances. Consequently, adopting such a method to evaluate the projection repeatedly in an iterative algorithm to solve composite problem (2) is still prohibitively costly when n is large (say in the millions), even if τ is close to 1.

On the other hand, problem (1) is a special case of the vector-k-norm problem studied in [42], which is a special case of the OWL-norm projection problem studied in [15, 18, 27]. The paper [18] outlines a scalar rootfinding routine that was not shown to terminate finitely. An $O(n + D \log D)$ implementation of an algorithm is provided in [15], where D denotes the number of unique elements in the input vector x^0. A minor extension of the analysis in [15, Theorem 3.2] indicates that the procedure can be modified to yield an $O(n)$ method for solving the fully sorted vector-k-norm problem, but the implementation is nontrivial to modify. Finally, [27] introduced a finite termination semismooth Newton method that exhibited superior performance relative. However, a potential limitation shared by each of these proposed methods is that a full sort of the input vector x^0 seems unavoidable.

Lastly, the top-k-sum projection problem (1) is related to the isotonic projection problem $\min_{x\in\mathbb{R}^n}\{\frac{1}{2}\|x - x^0\|_2 : x_i \geq x_j,\ \forall (i, j) \in E\}$, where $E \subseteq \{1, \ldots, n\} \times \{1, \ldots, n\}$ is comprised of the edges of a directed acyclic graph over n nodes. When $E = \{(i, i+1) : 1 \leq i \leq n-1\}$ forms a chain [2], then the constraint set can be represented by a polyhedral cone known as a (monotone) *isotonic projection cone* [23, 28] for fully sorted $x^0 \in \mathbb{R}^n$. This problem can be solved in $O(n)$ complexity by the well-known *pool adjacent violators algorithm* [3] or its primal variant [4]. The constraint in our problem (1) can be viewed as the intersection of the isotonic constraints and a half space (see the formulation (7) in the next section). Unfortunately, the projection onto this intersection cannot be done sequentially onto the latter two sets, thereby necessitating a specialized approach.

Given this context, the primary contribution of this paper is to provide an efficient oracle for obtaining an exact solution to the top-k-sum sublevel set projection problem (1). We propose two finite-termination algorithms, one is based on specializing a pivoting method to solve the parametric linear complementarity problem for Z-matrices [12] (see also [31] for an application to a more general problem of the portfolio selection) , and the other is a variant of a grid-search based method introduced in [42] that we call *early-stopping grid-search*. Both methods have complexities of $O(n)$ for a sorted input vector $x^0 \in \mathbb{R}^n$, where the absorbed constant is independent of k. When the input vector is unsorted, an additional $O(\bar{k}_1 \log n)$ floating point operations are needed, where $\bar{k}_1 \leq n$ is not known *a priori* but can be determined dynamically. To reduce this additional cost for practical applications in which the procedure to solve (1) is called repeatedly in an iterative method, we further derive a property of the projection that can make use of the approximate permutation of x^0 (*e.g.,* from the previous iteration). Extensive numerical results show that our solvers are often (multiple) orders magnitude faster than (and never slower than) the grid-search method introduced in [42], the semismooth Newton method in [27], and Gurobi's inexact QP solver. The `Julia` implementation of our methods is available at https://github.com/jacob-roth/top-k-sum [35].

The remainder of the paper is organized as follows. In Sect. 2, we summarize several equivalent formulations for solving (1). In Sect. 3, we present a parametric-LCP (PLCP) algorithm and a new early-stopping grid-search (ESGS) algorithm. We also present modifications of these algorithms to handle the vector-k-norm projection problem. Proofs in the preceding two sections are deferred to "Appendices A and B"; additional detail on the proposed methods is collected in "Appendix C". We compare the numerical performance of PLCP and ESGS with existing projection oracles on a range of problems in Sect. 4. The paper ends with a concluding section.

Notation and preliminaries

For a matrix A, the submatrix formed by the rows in an index set $\mathcal{I}$ and the columns in an index set $\mathcal{J}$ is denoted $A_{\mathcal{I},\mathcal{J}}$, where ":" denotes MATLAB notation for index sets, *e.g.,* $\mathcal{I} = 1:n$. The vector of all ones in dimension n is denoted by $\mathbb{1}_n$; for an index set $\mathcal{I} \subseteq \{1, \ldots, n\}$, $\mathbb{1}_{\mathcal{I}}$ denotes the vector with ones in the indices corresponding to $\mathcal{I}$ and zeros otherwise; when clear from context, for example $k \leq n$, we abuse notation and use $\mathbb{R}^n \ni \mathbb{1}_k := (\mathbb{1}_{1:k}, 0_{k+1:n})$; and e^i denotes the i^{th} standard basis vector. For a vector $x \in \mathbb{R}^n$, x_i denotes the i^{th} element of x, and for a vector ν, ${}_{v}\nu$ denotes the position of v in ν (so that, *e.g.,* ${}_{x_i}x = i$). For a vector $x \in \mathbb{R}^n$, $\overleftarrow{x}$ denotes a nonincreasing rearrangement of x with the convention that $\overleftarrow{x}_0 := +\infty$ and $\overleftarrow{x}_{n+1} := -\infty$. For any sorted $x^0 \in \mathbb{R}^n$ and any positive integer k, we may assume without loss of generality that there exist integers k_0, k_1 satisfying $0 \leq k_0 \leq k-1$ and $k \leq k_1 \leq n$ such that

$$\overleftarrow{x}^0_1 \geq \cdots \geq \overleftarrow{x}^0_{k_0} > \overleftarrow{x}^0_{k_0+1} = \cdots = \overleftarrow{x}^0_k = \cdots = \overleftarrow{x}^0_{k_1} > \overleftarrow{x}^0_{k_1+1} \geq \cdots \geq \overleftarrow{x}^0_n, \tag{3}$$

with the convention that $k_0 = 0$ if $\vec{x}_1^0 = \vec{x}_k^0$ and $k_1 = n$ if $\vec{x}_n^0 = \vec{x}_k^0$. Note that the presence of strict inequalities is not limiting: for example, if $x = \mathbb{1}$, then we may take $k_0 = 0$ and $k_1 = n$. The indices (k_0, k_1) denote the *index-pair of* x^0 *associated with* k and define the related sets $\alpha{:=}\{1, \ldots, k_0\}$, $\beta{:=}\{k_0{+}1, \ldots, k_1\}$ and $\gamma{:=}\{k_1{+}1 \ldots, n\}$. For a vector $x^0 \in \mathbb{R}^n$, the inequality $x^0 \geq 0$ is understood componentwise, and $x^0 \odot y^0$ denotes the Hadamard product of x^0 and y^0. The binary operators "$\wedge$" and "$\vee$" represent "logical and" and "logical or," respectively. Algorithmic "complexity" refers to floating point operations.

We also recall the following concepts from convex analysis. The indicator function $\delta_S(x)$ of a set $S \subseteq \mathbb{R}^n$ takes the value 0 if $x \in S$ and $+\infty$ otherwise; the support function is denoted by $\sigma_S(x^0){:=}\sup_y\{\langle x^0, y\rangle : y \in S\}$; and $\mathsf{proj}_S(x^0) = \mathsf{prox}_{\delta_S}(x^0)$ denotes the metric projection and proximal operators, respectively.

Next we recall standard notation from the literature of the linear complementarity problem (LCP) [12]. An LCP(q, M), defined by vector q and matrix M, is the collection of all vectors z such that $0 \leq Mz + q \perp z \geq 0$ where "$\perp$" denotes orthogonality. Given a scalar parameter λ, the parametric LCP (PLCP) is a collection of LCPs represented by PLCP$(\lambda; q, d, M) = \{\text{LCP}(q + \lambda d, M) : \lambda \in \mathbb{R}\}$ with d being a direction vector. Finally, we say $M \in \mathbb{R}^{n\times n}$ is a Z-matrix if all off-diagonal elements are nonpositive [12].

2 Equivalent formulations and existing techniques

In this section, we review some equivalent formulations of the projection problem (1) and motivate the form that our algorithms use. In particular, the alternate formulations either (i) introduce additional variables that destroy desirable structure in the original problem; (ii) have structure that we do not presently know how to leverage in designing a finite termination algorithm with complexity independent of parameter k; or (iii) are no easier than the formulation we use. Before studying the projection problem in greater detail, we note that there are at least two cases where the solution of this projection problem is immediate (assuming that x^0 does not belong to the top-k-sum sublevel set): (i) $k = 1$: $\tilde{x}_i = \min\{r, x_i^0\}$; and (ii) $k = n$: $\tilde{x}_i = x_i^0 - (\mathbb{1}_n^\top x^0 - r)/n$.

2.1 Unsorted formulations

The Rockafellar-Uryasev formulation

Using the Rockafellar-Uryasev [34] variational form of the superquantile, the projection problem (1) for $x^0 \in \mathbb{R}^n$ can be cast as a convex quadratic program (QP) subject to linear constraints with $n + 1$ auxiliary variables:

$$(\tilde{x}, \tilde{t}, \tilde{z}) = \underset{x,t,z}{\arg\min}\left\{\tfrac{1}{2}\|x - x^0\|_2^2 : t + \frac{1}{k}\sum_{i=1}^{n} z_i \leq r/k,\ z \geq x - t\mathbb{1},\ z \geq 0\right\}. \quad (4)$$

The introduction of t and z destroys strong convexity of the original problem. Nonetheless, the interior-point method can be used to obtain a solution in polynomial time.

As an alternative, problem (4) can be solved via the solution method of the linear complementarity problem (LCP) [12, 20].

The unsorted top-k formulation

Given $x^0 \in \mathbb{R}^n$, let κ denote the (possibly unsorted) indices of the k largest elements of x^0 and $[k]{:=}{}_{(\overleftarrow{x}^{0}_{k})} x^0$ denote the position of the k^{th} largest element of x^0. Then $x^0_{[k]}$ and index $[k]$ can be identified $O(n)$ [1, 6] time,[1] and a second scan through x^0 can identify κ. Given the indices κ and $[k]$, consider the following problem

$$\tilde{x} = \arg\min_x \left\{ \tfrac{1}{2}\|x - x^0\|_2^2 : Bx \le b \right\}, \quad \text{where } b{:=}(r, 0_{n-1}^\top)^\top \text{ and} \tag{5a}$$

$$B{:=}\begin{bmatrix} \mathbb{1}_\kappa^\top \\ B'_{\mathcal{I}_1,:} \\ B'_{\mathcal{I}_2,:} \end{bmatrix}, \ (\mathbb{1}_\kappa)_j = \mathbb{1}_{\{t \in \kappa\}}(j), \tag{5b}$$

$$B'_{ij}{:=}\begin{cases} +1, & \text{if } (i \in \kappa) \wedge (j = [k]) \text{ or if } (i \notin \kappa) \wedge (j = i) \\ -1, & \text{if } (i \in \kappa) \wedge (j = i) \text{ or if } (i \notin \kappa) \wedge (j = [k]) \end{cases}, \tag{5c}$$

with $B' \in \mathbb{R}^{(n-1)\times n}$, $\mathcal{I}_1 = \{i \in \{1, \ldots, n\} : i \in \kappa \setminus [k]\}$, and $\mathcal{I}_2 = \{i \in \{1, \ldots, n\} : i \notin \kappa\}$. In general, the feasible region $\left\{x \in \mathbb{R}^n : \sum_{i\in\kappa} x_i \le r,\ x_i \ge x_{[k]},\ \forall i \in \kappa,\ x_j \le x_{[k]},\ \forall j \notin \kappa\right\}$ is a strict subset of $\mathcal{B}^r_{(k)}$, but the optimal solutions must coincide, as summarized in the following simple result.

Lemma 1 *The optimal solution of problem (1) is the same as that of the unsorted top-k problem (5).*

The KKT conditions expressed in (monotone) LCP form are given by $0 \le (b - Bx^0) + BB^\top z \perp z \ge 0$, where $z \ge 0$ is the dual variable. Aside from the positive definiteness of $BB^\top$, it is not clear if there is additional structure that can be used to design an efficient LCP solution approach.

The unsorted top-k formulation via the Moreau decomposition

On the other hand, the Moreau decomposition $x^0 = \mathsf{prox}_{\delta_{\mathcal{B}^r_{(k)}}}(x^0) + \mathsf{prox}_{\delta^*_{\mathcal{B}^r_{(k)}}}(x^0)$ provides an alternative formulation to compute a solution from $\tilde{x} = x^0 - \mathsf{prox}_{\delta^*_{\mathcal{B}^r_{(k)}}}(x^0)$. The conjugate function $\delta^*_{\mathcal{B}^r_{(k)}}$ can be computed easily by using properties of linear programs and is summarized in Theorem 2. Note that the following result is also useful for computing the dual objective value of a problem involving the top-k-sum sublevel set.

[1] See also `quickselect` [21] for $O(n + k \log n)$ expected time and `heapsort` and the max-heap data structure [40], as well as Algorithm 2 in [10] for $O(k + (n - k)\log k)$ time.

Lemma 2 *Let B be the unsorted-top-k matrix defined in (5) and $c \in \mathbb{R}^n$ be arbitrary. Then*

$$\delta^*_{\mathcal{B}^r_{(k)}}(c) = \begin{cases} \frac{r}{k}\mathbb{1}^\top c, & \text{if } B^{-\top}c \geq 0; \\ +\infty, & \text{otherwise.} \end{cases}$$

In addition, the condition $B^{-\top}c \geq 0$ can be checked in $O(n+k+(n-k)\log k)$ time for the worst case and $O(n+k\log n)$ time in expectation. Furthermore, a sufficient condition for $B^{-\top}c \geq 0$ can be checked in $O(n)$ time.

By Theorem 2, we can compute $\tilde{y} := \operatorname{prox}_{\delta^*_{\mathcal{B}^r_{(k)}}}(x^0)$ using the following formulation

$$\tilde{y} = \arg\min_{y \in \mathbb{R}^n} \left\{ \tfrac{1}{2}\|y - (x^0 - \tfrac{r}{k}\mathbb{1})\|_2^2 : B^{-\top}y \geq 0 \right\}, \tag{6}$$

with KKT conditions $0 \leq B^{-\top}(x^0 - \frac{r}{k}\mathbb{1}) + B^{-\top}B^{-1}z \perp z \geq 0$ for dual variable $z \geq 0$. The matrix $B^{-\top}B^{-1}$ shares properties similar to $BB^\top$, but the structure cannot be leveraged in an obvious manner.

2.2 Sorted formulations

The isotonic formulation

An alternative approach adopted in [42] is based on the observation employed in Theorem 1: if the initial input to the projection problem (1) is sorted in a nonincreasing order, *i.e.,* $x^0 = \overleftarrow{x}^0$, then the unique solution will also be sorted in a nonincreasing order, *i.e.,* $\tilde{x}_i \geq \tilde{x}_{i+1}$ for $i = 1, \ldots, n-1$. Thus (1) is equivalent to

$$\begin{array}{ll} \underset{x \in \mathbb{R}^n}{\text{minimize}} & \dfrac{1}{2}\|x - \overleftarrow{x}^0\|_2^2 \\ \text{subject to} & \left\{ \sum_{i=1}^{k} x_i \leq r, \quad x_i \geq x_{i+1}, \ \forall i \in \{1, \ldots, n-1\} \right\} =: \{x \in \mathbb{R}^n : Cx \leq b\}, \end{array} \tag{7}$$

where

$$b := (r, 0_{n-1}^\top)^\top, \quad C := \begin{bmatrix} (\mathbb{1}_k^\top, \ 0_{n-k}^\top) \\ -D \end{bmatrix}, \quad D := \begin{bmatrix} +1 & -1 & & \\ & \ddots & \ddots & \end{bmatrix} \in \mathbb{R}^{(n-1)\times n}, \tag{8}$$

which consists of nonseparable isotonic constraints ($Dx \geq 0$) and a single budget constraint ($\mathbb{1}_k^\top x \leq r$). Problems (1) and (7) are equivalent in the sense that $\bar{x}$ is the solution to (7) if and only if there exists a permutation π of $\{1, \ldots, n\}$ with inverse π^{-1} such that $\bar{x}_{\pi^{-1}} = \tilde{x}$ is the solution to (1). The solution to (7) is obtained by translating three contiguous subsets of $\overleftarrow{x}^0$, as depicted in Fig. 1, and it obeys the same ordering as $\overleftarrow{x}^0$. The difficulty is in identifying the breakpoints k_0 and k_1 at the solution that satisfies (3).

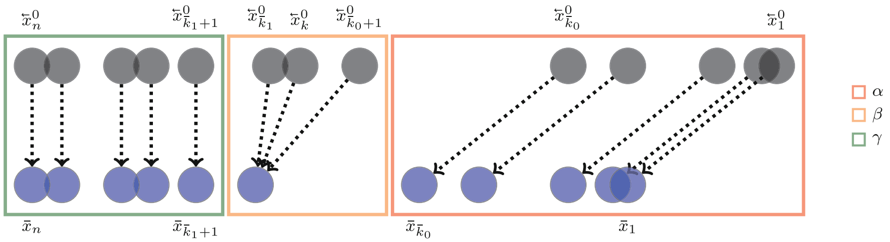


Fig. 1 Schematic of sorted input $\bar{x}^0 \in \mathbb{R}^n$ (grey, top) and sorted projection $\bar{x} \in \mathcal{B}^r_{(k)}$ (blue, bottom)

The constraint matrix C associated with the sorted problem (8) is readily seen to be invertible, and inspection of the KKT conditions yields the LCP(q, M) with data

$$q := (r, 0^\top_{n-1})^\top - C\,\bar{x}^0 \in \mathbb{R}^n, \quad M := CC^\top \in \mathbb{R}^{n\times n}. \tag{9}$$

By direct computation, $CC^\top$ is a symmetric positive definite Z-matrix (*i.e.,* a K-matrix *cf.* [12, Definition 3.11.1]), so Chandrasekaran's complementary pivoting method [7] can process the LCP in at most n steps (also see the *n-step scheme* summarized in [12, Algorithm 4.8.2]). The matrix M is also seen to be tridiagonal except for possibly the first row and first column, due to contributions from the budget constraint. As in the unsorted case, using the Moreau decomposition does not further simplify the problem.

The KKT grid-search

An alternative method for solving the sorted problem (7) is based on a careful study of the sorted problem's KKT conditions introduced in [42]. It is shown (*cf.* Step 2 in Algorithm 4 in [42]) that each $(k_0, k_1) \in \{0, \ldots, k-1\} \times \{k, \ldots, n\}$ defines a candidate primal solution. The true solution can be recovered by performing a grid-search over the sorting-indices k_0 and k_1 and terminating once the KKT conditions have been satisfied. For each (k_0, k_1), the KKT conditions can be checked in constant time that is independent of n and k, so the overall complexity is $O(k(n-k))$.

3 Proposed algorithms

In this section, we describe two efficient procedures for solving the projection problem (1) by viewing the top-k-sum sublevel set as the intersection of a summation constraint and an ordering constraint. The first method is a (dual) parametric pivoting procedure based on the Z-matrix structure of the sorted problem's KKT conditions, which ensures that iterates are sorted and terminates once the summation constraint is satisfied. The second method uses a detailed analysis of the KKT conditions to refine the (primal) grid-search procedure introduced in [42], which ensures that the summation constraint is satisfied (with equality) and terminates once the ordering constraint is satisfied.

3.1 A parametric-LCP approach

Penalizing the summation constraint in (7) with $\lambda \geq 0$ in the objective yields a sequence of subproblems parametrized by λ that can be handled by he parametric-LCP (PLCP) approach described in [12, Section 4.5]. Given a candidate solution that fails to satisfy both constraints simultaneously, the PLCP prescribes two forms of corrective actions: increasing the penalty parameter $\lambda \geq 0$ and/or expanding the set of active isotonic constraints. The former action is moderated by the need to seek a sorted ordering, which limits how much λ can be increased and results in a piecewise affine solution mapping with respect to λ. Due to the structure of the isotonic constraints, both actions can be implemented in constant time, and at most n adjustments to λ will be needed since the basis cannot contain more than n variables.

For fixed λ, the subproblem is given by

$$\begin{array}{ll} \underset{x\in\mathbb{R}^n}{\text{minimize}} & \frac{1}{2}\|x - \overleftarrow{x}^0\|_2^2 + \lambda \cdot \left(\mathbb{1}_k^\top x - r\right) \\ \text{subject to} & Dx \geq 0, \end{array} \tag{10}$$

where D is defined in (8) as the isotonic operator associated with the ordered, consecutive differences in x. Collecting these problems for $\lambda \geq 0$ yields the PLCP(λ; q, d, M) where $M{:=}DD^\top$ is a tridiagonal positive definite Z-matrix. To solve (7), it is sufficient to identify a $\lambda \geq 0$ so that (i) the budget constraint is satisfied; and (ii) the LCP optimality conditions associated with (10) hold. Let $z \geq 0_{n-1}$ be the dual variable associated with $Dx \geq 0$. The KKT conditions of (10) take the form of

$$0 \leq w := D(\overleftarrow{x}^0 - \lambda \mathbb{1}_k + D^\top z) \perp z \geq 0, \tag{11}$$

and yield the following PLCP(λ; q, d, M) formulation of the full projection problem (7), with PLCP data:

$$M{:=}DD^\top, \quad q{:=}D\,\overleftarrow{x}^0 \geq 0, \quad d{:=} -\, D\mathbb{1}_k = -e^k. \tag{12}$$

One can compute that

$$M = \begin{bmatrix} 2 & -1 & & & \\ -1 & 2 & -1 & & \\ & \ddots & \ddots & \ddots & \\ & & -1 & 2 & -1 \\ & & & -1 & 2 \end{bmatrix} \in \mathbb{R}^{(n-1)\times(n-1)}, \tag{13}$$

which is a positive definite Z-matrix. By increasing the parameter λ from 0 to $+\infty$, one can solve the problem by identifying the optimal basis $\xi \subseteq \{1, \ldots, n-1\}$ such that the budget constraint is satisfied, $z_{\xi^{\mathsf{c}}} = 0$ (where $\xi^{\mathsf{c}} = \{1, \ldots, n-1\} \setminus \xi$), and $w_\xi = 0$. Due to a special property of the Z-matrix that ensures monotonicity of the solution z as a function of λ [12, Discussion of Proposition 4.7.2], the unique solution $z(\bar{\lambda})$ can be obtained by solving at most n subproblems. Finally, since M is tridiagonal, each subproblem for a fixed λ can be solved in $O(1)$, and a primal solution can be recovered from the optimal dual vector by $\bar{x} = \overleftarrow{x}^0 - \lambda \mathbb{1}_k + D^\top z(\bar{\lambda})$, leading to an $O(n)$ method where the absorbed constant is independent of k.

Algorithm 1 PLCP projection onto $\mathcal{B}^r_{(k)}$

Initialize:

(i) If $\mathsf{T}_{(k)}(\overleftarrow{x}^0) \leq r$, return $\bar{x} = \overleftarrow{x}^0$.
(ii) Otherwise, if $k = 1$, set $\bar{x} = \min\{x^0, r\}$.
(iii) Otherwise, if $k = n$, set $\bar{x} = \overleftarrow{x}^0 - \mathbb{1}_n \cdot (\mathbb{1}_n^\top x^0 - r)/n$.
(iv) Otherwise, handle iteration $t = 0$.
 (a) Set $s^0 = \sum_{i=1}^k x_i^0$, $t = 0$, $\xi = \emptyset$, $q_k = x_k^0 - x_{k+1}^0$, $\lambda = q_k$, and $m = s^0 - k\lambda$.
 (b) If $m \leq r$, set $\bar{\lambda} = (s^0 - r)/k$ and $\bar{x} = x^0 - \bar{\lambda}\mathbb{1}_k$.
 (c) Otherwise, set $t = 1$, $\xi = \{k\}$, $a = b = s = k$, ${}_a\xi = {}_k\xi = {}_b\xi = 1$, $\lambda^a = \lambda^b = \lambda$, $z_a^0 = z_k^0 = z_b^0 = -q_k \cdot m_{ij}^{-1}({}_k\xi, {}_k\xi) = -q_k/2$, $\sigma = 2 - m_{ij}^{-1}({}_k\xi, {}_k\xi) = 3/2$, define the function $m_{ij}^{-1}({}_c\xi, {}_d\xi) := (M_{\xi\xi}^{-1})_{{}_c\xi, {}_d\xi} = \big(|\xi| + 1 - \max\{{}_c\xi, {}_d\xi\}\big) \cdot \min\{{}_c\xi, {}_d\xi\} \cdot \big(|\xi| + 1\big)^{-1}$, and proceed.

while true **do**
  Compute the $(t+1)^{\text{th}}$ breakpoint:
    $\lambda^a = (q_{a-1} - z_a^0)/m_{ij}^{-1}(\xi_k, \xi_a)$
    $\lambda^b = (q_{b+1} - z_b^0)/m_{ij}^{-1}(\xi_k, \xi_b)$
    $\lambda = \min\{\lambda^a, \lambda^b\}$
  Compute $\mathsf{T}_{(k)}(x(\lambda))$ to determine if optimal solution lies within $(0, \lambda]$:
    $T = s^0 - k \cdot \lambda + z_k^0 + m_{ij}^{-1}({}_k\xi, {}_k\xi) \cdot \lambda$ ▷ compute $\mathsf{T}_{(k)}(x(\lambda))$.
    if $T \leq r$ **then**
      $\bar{\lambda} = (s^0 - r + z_k)/\big(k - m_{ij}^{-1}({}_k\xi, {}_k\xi)\big)$ ▷ solve $\mathsf{T}_{(k)}(x(\lambda)) = r$ for λ.
      return $\bar{x} = \overleftarrow{x}^0 - \bar{\lambda}\mathbb{1}_k + D^\top z(\bar{\lambda})$ by calling Algorithm 2
    end if
  Update $z(0)$ via Schur complement, and update ξ by inspecting $\{\lambda^a, \lambda^b\}$:
    if $\lambda = \lambda^a$ **then**
      $z_a^0 = (z_a^0 - q_{a-1})/\sigma$
      $z_k^0 = z_k^0 + z_a^0 \cdot m_{ij}^{-1}({}_k\xi, {}_a\xi)$
      $z_b^0 = z_b^0 + z_a^0 \cdot m_{ij}^{-1}({}_b\xi, {}_a\xi)$
      $a = a - 1, \quad {}_k\xi = {}_k\xi + 1, \quad {}_b\xi = {}_b\xi + 1$ ▷ $\xi = (a - 1, \xi)$.
    else
      $z_b^0 = (z_b^0 - q_{b+1})/\sigma$
      $z_a^0 = z_a^0 + z_b^0 \cdot m_{ij}^{-1}({}_a\xi, {}_b\xi)$
      $z_k^0 = z_k^0 + z_b^0 \cdot m_{ij}^{-1}({}_k\xi, {}_b\xi)$
      $b = b + 1, \quad {}_b\xi = {}_b\xi + 1$ ▷ $\xi = (\xi, b + 1)$.
    end if
  Increment iteration:
    $t = t + 1$ and $\sigma = (t + 2)/(t + 1)$ ▷ Schur complement: $\sigma = 2 - m_{ij}^{-1}({}_a\xi, {}_a\xi)$.
end while

The parametric-LCP method specialized to the present problem is summarized in Algorithm 1. Additional detail on the mechanics of the pivots is provided in "Appendix C.1". In addition, it is worthwhile to point out that to avoid additional memory alloca-

Algorithm 2 $O(|\xi|)$ update of $y \leftarrow y + D^{\top}_{:,\xi} z_{\xi}$ from optimal basis elements $a, b, {}_{k}\xi$, dual λ, and $\overleftarrow{x}^0 \in \mathbb{R}^n$.

Initialize: Set $m = b - a + 1, c = 0, a_i = 0$.
Compute `cumsum`$((1, 2, \dots, m) \odot$ `reverse`$(-q^{\lambda}_{\xi}))$: ▷ `cumsum` is cumulative sum
▷ `reverse` reverses the order of a vector
  for $i \in \{1, \dots, m\}$ **do**
    $j = b - i + 1$
    $c = c + i \cdot \left(-(\overleftarrow{x}^0_j - \overleftarrow{x}^0_{j+1})\right)$
    if $m - i + 1 = {}_{k}\xi$ **then**; $c = c + \lambda$; **end if**
  end for
  $c = c/(m+1)$ and $y_a = y_a + c$
Compute `cumsum`$((1, 2, \dots, m) \odot -q^{\lambda}_{\xi})$:
  for $i \in \{2, \dots, m+1\}$ **do**
    $j = a + i - 2$
    $c = c - \left(-(\overleftarrow{x}^0_j - \overleftarrow{x}^0_{j+1})\right)$
    if $i - 1 = {}_{k}\xi$ **then**; $c = c + \lambda$; **end if**
    $y_{j+1} = y_{j+1} + c$
  end for

tion, the full dual solution z is not maintained explicitly throughout the algorithm (only three components, z_a, z_b, and z_k are maintained), so the step in line 10 of Algorithm 1 requires a simple but specialized procedure summarized in Algorithm 2 to implicitly reconstruct z and compute $D^{\top}z$. This can be done in $O(|\xi|)$ cost given an optimal basis ξ as outlined in Theorem 3. On the other hand, if the user is willing to store the full dual vector z, then by storing and filling in the appropriate entries of z after each pivot (line 23 in Algorithm 1[2]), it is clear that $D^{\top}z$ can be obtained in $O(|\xi|)$ cost due to the simple pairwise-difference structure of D and the complementarity structure which gives $z_{\xi^c} = 0$. Finally, we formally state the $O(n)$ complexity result, preceded by a lemma whose proof is deferred to "Appendix B.1".

Lemma 3 *Algorithm 2 computes $D^{\top}z$ in $O(|\xi|)$ from initial data $\overleftarrow{x}^0$ and optimal output of Algorithm 1: a, b, ${}_{k}\xi$, and λ.*

Using the above lemma, we arrive at the folowing conclusion.

Proposition 1 *The overall complexity to solve the sorted problem (7) by Algorithm 1 is $O(n)$.*

Proof We first cite a classical result regarding the number of pivots needed to identify the optimal basis. By [11, Theorem 2], for every $q \geq 0$ and for every d, since M is a symmetric, positive definite Z-matrix, the solution map $z(\lambda)$ is a point-to-point, nondecreasing and convex piecewise-linear function of λ such that $z(\lambda_1) \leq z(\lambda_2)$ for $0 \leq \lambda_1 \leq \lambda_2$. Therefore, once a variable z_i becomes basic, it remains in the basis ξ for every subsequent pivot. Since each z_i is monotone nondecreasing and piecewise-linear, the basis ξ^t in iteration t remains optimal over an interval $[\lambda^t, \lambda^{t+1}]$. As a result, the interval $[0, +\infty)$ can be partitioned into n pieces $S^0 \cup \cdots \cup S^{n-1}$

[2] The algorithm computes $z^0 := z(0)$ rather than $z(\lambda)$; the desired vector can be obtained from $z_{\xi}(\lambda) = z^0_{\xi} + \lambda M^{-1}_{\xi\xi} e^k_{\xi}$ given basis ξ.

with progressively larger bases ($|\xi^0| \leq \cdots \leq |\xi^{n-1}|$) such that the solution $z(\lambda^t)$ satisfies the LCP optimality conditions (12) for all $\lambda^t \in S^t$. Since the constraints in (7) are linearly independent, the optimal dual variables are unique and in particular the optimal $\bar{\lambda}$ is finite in (10).

Given solution $z(\lambda^t)$ in iteration t, each subsequent iteration performs three steps: (i) determining the next breakpoint λ^{t+1} (lines 2-5); (ii) checking whether or not the budget constraint is satisfied for some $\lambda \in [\lambda^t, \lambda^{t+1}]$ (lines 7-10); and (iii) updating the solution $z(\lambda^{t+1})$ for the new breakpoint λ^{t+1} if the budget constraint is not satisfied (lines 12-23). Each step can be performed in $O(1)$ cost due to the tridiagonal structure of M and the sparse structure of d. Detailed justification relies on the Sherman-Morrison and Schur complement identities and is provided in "Appendix C.1". Finally, the cost required to recover the primal solution from optimal dual $z(\bar{\lambda})$ with basis $\bar{\xi}$ via $\bar{x} = \overleftarrow{x}^0 - \mathbb{1}_k\lambda + D^\top z(\bar{\lambda})$ is readily seen to be $O(|\bar{\xi}|)$ by the structure of D, using Theorem 3. Thus, the overall complexity is $O(n)$. □

3.2 An early-stopping grid-search approach

Since our second algorithm depends on the framework described in [42], we reproduce some background. Recall that the constraint $\mathsf{T}_{(k)}(x) \leq r$ can be represented by finitely many linear inequalities, so the following KKT conditions are necessary and sufficient for characterizing the unique solution $\bar{x}$ and its multiplier $\bar{\lambda}$ to the sorted problem (7):

$$\bar{x} = \overleftarrow{x}^0 - \bar{\lambda}\mu \quad \text{for some } \mu \in \partial \mathsf{T}_{(k)}(\bar{x}), \tag{14.1}$$

$$0 \leq [\, r - \mathsf{T}_{(k)}(\bar{x})\,] \perp \bar{\lambda} \geq 0. \tag{14.2}$$

Recall the index-pair (k_0, k_1) of x^0 associated with k in (3) and its related sets

$$\alpha{:=}\{1, \ldots, k_0\}, \quad \beta{:=}\{k_0+1, \ldots, k_1\}, \quad \gamma{:=}\{k_1+1, \ldots, n\}. \tag{15}$$

For any $x^0 \in \mathbb{R}^n$ where $\overleftarrow{x}^0$ satisfies the order structure (3), Lemma 2.2 in [42] gives

$$\partial \mathsf{T}_{(k)}(\overleftarrow{x}^0) = \left\{\mu \in \mathbb{R}^n : \ \mu_\alpha = \mathbb{1}_{|\alpha|},\ \mu_\beta \in \phi_{k_1-k_0, k-k_0},\ \mu_\gamma = 0\right\}, \tag{16}$$

where for $m_1 \geq m_2$,

$$\phi_{m_1, m_2}{:=}\left\{w \in \mathbb{R}^{m_1} : 0 \leq w \leq \mathbb{1},\ \mathbb{1}^\top w = m_2\right\}. \tag{17}$$

The procedure in [42] utilizes (16) to design a finite-termination algorithm that finds a subgradient $\bar{\mu}$ with indices $\bar{k}_0$ and $\bar{k}_1$ so that $\bar{\mu} \in \partial \mathsf{T}_{(k)}(\bar{x})$.

3.2.1 KKT conditions

To solve the KKT conditions (14), there are two cases. If $\mathsf{T}_{(k)}(x^0) \leq r$, then $\bar{x} = \overleftarrow{x}^0$ and $\bar{\lambda} = 0$ is the unique solution. On the other hand, if $\mathsf{T}_{(k)}(x^0) > r$, then $\mathsf{T}_{(k)}(\bar{x}) = r$ and $\bar{\lambda} > 0$. This holds because if $\bar{\lambda} = 0$, then $\bar{x} = \overleftarrow{x}^0$ by stationarity, resulting in contradiction: by assumption, $\mathsf{T}_{(k)}(x^0) > r$, but by primal feasibility $\mathsf{T}_{(k)}(\overleftarrow{x}^0) = \mathsf{T}_{(k)}(\bar{x}) \le r$.

Let us now focus on the solution method for the second case where $\mathsf{T}_{(k)}(x^0) > r$. The KKT conditions (14) can be expressed as

$$\begin{array}{llll} \bar{x}_{\bar{\alpha}} & = \overleftarrow{x}^0_{\bar{\alpha}} - \bar{\lambda}\mathbb{1}_{\bar{\alpha}} & \bar{\mu}_{\bar{\alpha}} & = \mathbb{1}_{\bar{\alpha}} \\ \bar{x}_{\bar{k}_0} & > \bar{\theta} > \bar{x}_{\bar{k}_1+1} & \bar{\mu}_{\bar{\beta}} & \in \phi_{\bar{k}_1-\bar{k}_0, k-\bar{k}_0} \\ \bar{x}_{\bar{\beta}} & = \bar{\theta}\mathbb{1}_{\bar{\beta}} & \bar{\mu}_{\bar{\gamma}} & = 0 \\ r & = \mathbb{1}_{\bar{\alpha}}^\top \overleftarrow{x}^0_{\bar{\alpha}} - \bar{k}_0\bar{\lambda} + (k-\bar{k}_0)\bar{\theta} & \bar{\lambda} & > 0 \end{array} \tag{18}$$

for appropriate indices $\bar{k}_0$ and $\bar{k}_1$. Based on (16), any candidate index-pair (k_0', k_1') gives rise to a candidate primal solution, which we denote by $x'(k_0', k_1')$, by solving a 2-dimensional linear system in variables (θ', λ') derived from summing the α' and β' components of the stationarity conditions; see "Appendix C.2" for more detail. Where it is clear from context, we simplify notation by dropping the dependence on (k_0', k_1') and let x' denote $x'(k_0', k_1')$. On the other hand, we overload notation to let $\lambda(k_0', k_1')$ and $\theta(k_0', k_1')$ denote the solution to the linear system associated with index pair (k_0', k_1'). Using the form of the KKT conditions, we can recover the candidate primal solution x' as follows:

$$x'_{\alpha'} = \overleftarrow{x}^0_{\alpha'} - \lambda'\mathbb{1}_{\alpha'}, \quad x'_{\beta'} = \theta'\mathbb{1}_{\beta'}, \quad x'_{\gamma'} = \overleftarrow{x}^0_{\gamma'} \tag{19.1}$$

$$\theta' = \Big(k_0'\mathbb{1}_{\beta'}^\top \overleftarrow{x}^0_{\beta'} - (k-k_0')\big(\mathbb{1}_{\alpha'}^\top \overleftarrow{x}^0_{\alpha'} - r\big)\Big)/\rho' \tag{19.2}$$

$$\lambda' = \Big((k-k_0')\mathbb{1}_{\beta'}^\top \overleftarrow{x}^0_{\beta'} + (k_1'-k_0')\big(\mathbb{1}_{\alpha'}^\top \overleftarrow{x}^0_{\alpha'} - r\big)\Big)/\rho' \tag{19.3}$$

$$\rho' = k_0'(k_1'-k_0') + (k-k_0')^2. \tag{19.4}$$

Based on (18), the candidate solution (x', λ', θ') is optimal if and only if the following five conditions hold:

$$\lambda' > 0, \tag{20.1}$$

$$\overleftarrow{x}^0_{k_0'} > \theta' + \lambda', \tag{20.2}$$

$$\theta' + \lambda' \ge \overleftarrow{x}^0_{k_0'+1}, \tag{20.3}$$

$$\overleftarrow{x}^0_{k_1'} \ge \theta', \tag{20.4}$$

$$\theta' > \overleftarrow{x}^0_{k_1'+1}. \tag{20.5}$$

See "Appendix C.2" for detail on the above step, which uses the form of the subdifferential from (16). To obtain a solution $(\bar{x}, \bar{\lambda})$ to the KKT conditions (14), it is sufficient to perform a grid search over $k_0 \in \{0, \dots, k-1\}$ and $k_1 \in \{k, \dots, n\}$ and check (20), which is the approach adopted in [42, Algorithm 4].

On the other hand, detailed inspection of monotonicity properties of the reduced KKT conditions (20) yield an $O(n)$ primal-based procedure for solving the sorted problem (7) as outlined in Algorithm 3. Instead of executing a grid-search over

$\{0, \dots, k-1\} \times \{k, \dots, n\}$, our procedure exploits the hidden properties of the KKT conditions. We construct a path from $(k_0, k_1) = (k-1, k)$ to $(\bar{k}_0, \bar{k}_1)$, composed solely of decrements to k_0 and increments to k_1. This path maintains satisfaction of KKT conditions 1, 3, and 4, and seeks a pair (k_0, k_1) that yields (x, λ) satisfying complementarity, *i.e.,* KKT conditions 2 and 5. In general, this procedure generates a different sequence of "pivots" from Algorithm 1 and to the best of our knowledge does not exist in the current literature.

3.2.2 Implementation

The procedure, as summarized in Algorithm 3, is very simple. The algorithm's behavior is depicted in Fig. 2, in which a path from $(k-1, k)$ to $(\bar{k}_0, \bar{k}_1)$ is generated by following the orange and blue arrows. Due to the ordering properties of the problem, a sequence of index-pairs can be constructed in which (20.1)∧(20.2)∧(20.3) always hold. Since the optimal LCP basis is contiguous (as shown in "Appendix C.1") and again due to certain monotonicity properties, updates to the index-pair (k_0, k_1) either decrement k_0 or increment k_1. The former occurs when (20.2) fails to hold, the latter occurs when (20.5) fails to hold, and the algorithm terminates when both (20.2∧(20.5)) hold.

Algorithm 3 ESGS projection onto $\mathcal{B}^r_{(k)}$.

Initialize:

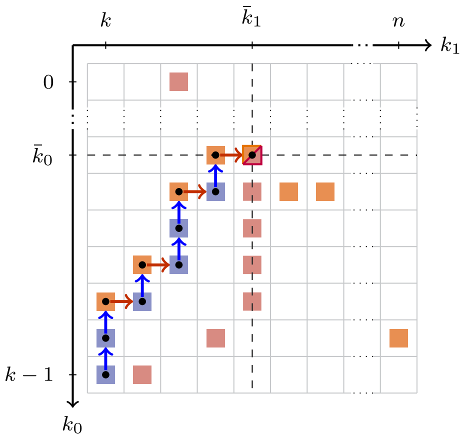
(i) If $\mathsf{T}_{(k)}(x^0) \leq r$, return $\bar{x} = x^0$.
(ii) Else, if $k = 1$, set $\bar{x} = \min\{x^0, r\}$.
(iii) Else, if $k = n$, set $\bar{x} = x^0 - \mathbb{1}_n \cdot (\mathbb{1}_n^\top x^0 - r)/n$.
(iv) Else, set $k_0 = k - 1$, $k_1 = k$, and $\texttt{solved} = 0$.

while true **do**
  Compute candidate θ and λ and KKT indicators:
    $\rho = k_0(k_1 - k_0) + (k - k_0)^2$
    $\theta = \left(k_0 \mathbb{1}_\beta^\top x_\beta^0 - (k - k_0)\left(\mathbb{1}_\alpha^\top x_\alpha^0 - r\right)\right)/\rho$
  $\theta + \lambda = \left(k \mathbb{1}_\beta^\top x_\beta^0 + (k_1 - k)\left(\mathbb{1}_\alpha^\top x_\alpha^0 - r\right)\right)/\rho$
  $\texttt{kkt}_2 = x_{k_0}^0 > \theta + \lambda, \quad \texttt{kkt}_5 = \theta > x_{k_1+1}^0$
  Check KKT conditions:
    if $\texttt{kkt}_2 \wedge \texttt{kkt}_5$ **then** $\texttt{solved} = 1$
    else if $\texttt{kkt}_2$ **then** $k_1 \leftarrow k_1 + 1$
    else if $\neg\texttt{kkt}_2$ **then** $k_0 \leftarrow k_0 - 1$
    end if
    if $\texttt{solved} = 1$ **then**
      return $\bar{x}$ via $\bar{x}_\alpha = x_\alpha^0 - \lambda \mathbb{1}_\alpha$, $\bar{x}_\beta = \theta \mathbb{1}_\beta$, and $\bar{x}_\gamma = x_\gamma^0$.
    end if
end while

3.2.3 Analysis

The analysis of Algorithm 3 builds on the framework introduced in [42] for performing a grid search over candidate index-pairs (k_0, k_1). The grid search proceeds with outer loop over $k_1 \in \{k, \dots, n\}$ and inner loop over $k_0 \in \{0, \dots, k-1\}$, though similar

Fig. 2 Schematic for Algorithm 3. Orange shading indicates that $\mathtt{kkt}_2 \wedge \mathtt{kkt}_3$ holds; blue shading indicates that $\neg\mathtt{kkt}_2 \wedge \mathtt{kkt}_3$ holds; red shading indicates that $\mathtt{kkt}_4 \wedge \mathtt{kkt}_5$ holds; black circles trace the trajectory taken by ESGS and indicate that $\mathtt{kkt}_1 \wedge \mathtt{kkt}_3 \wedge \mathtt{kkt}_4$ holds



analysis holds when reversing the search order (but is not discussed further). The new observation leveraged in Proposition 2 is that the order structure of the KKT conditions induces various forms of monotonicity in the KKT residuals across k_0 and k_1. This monotonicity holds globally (*i.e.,* across all k_0 for given k_1 or all k_1 for given k_0) and implies that the local KKT information at a candidate index-pair (k_0, k_1) provides additional information about which indices can be "skipped" in the full grid search procedure.

For example, in the sample trajectory traced in Fig. 2, the algorithm starts at $(k_0, k_1) = (k-1, k)$ and searches over all k_0 in the first column for an index-pair that satisfies KKT conditions 2 and 3. Suppose that such an index is found in some row, which we denote by $k_0^*(k)$ to emphasize the dependence of such a row on the column $k_1 = k$. Further suppose that the pair $(k_0^*(k), k)$ does not satisfy *all* of the KKT conditions. Then monotonicity implies that no candidate index-pair (k_0', k) for $k_0' \leq k_0^*(k)$ can satisfy *all* of the KKT conditions, providing an early termination to the inner search over k_0 associated with column $k_1 = k$. This justifies terminating the current search over k_0 in column $k_1 = k$ and increasing $k_1 \leftarrow k_1 + 1$. At this point, the full grid search would begin again at $(k-1, k+1)$. However, another monotonicity property of the KKT residuals implies that all $k_0' > k_0^*(k)$ cannot be optimal, which justifies starting the grid search "late" at $(k_0^*(k), k+1)$ rather than $(k-1, k+1)$. Based on these ideas, the main effort in establishing the correctness of Algorithm 3 is in showing how to use monotonicity properties of the KKT residuals to justify transitions "up" or "right". The first idea is summarized in Lemma 5 and the second is summarized in Lemma 6. Proposition 2 is obtained by combining these earlier results, immediately giving the desired complexity analysis.

We begin the analysis of Algorithm 3 by making the following assumption.

Assumption 1 (*Strict projection*) For the initial vector $x^0 \in \mathbb{R}^n$, it holds that $\mathsf{T}_{(k)}(x^0) > r$.

For simplicity of notation, we also assume that the initial point is sorted $x^0 = \overleftarrow{x}^0$ and drop the sorting notation in the remainder of this section (and its proofs).

At a candidate index-pair (k_0, k_1), define the KKT satisfaction indicators

$$\texttt{kkt}_1(k_0, k_1) := \mathbb{1}_{\{t>0\}}\big(\lambda(k_0, k_1)\big), \tag{21.1}$$

$$\texttt{kkt}_2(k_0, k_1) := \mathbb{1}_{\{t>0\}}\big(x^0_{k_0} - (\theta + \lambda)(k_0, k_1)\big), \tag{21.2}$$

$$\texttt{kkt}_3(k_0, k_1) := \mathbb{1}_{\{t\geq 0\}}\big((\theta + \lambda)(k_0, k_1) - x^0_{k_0+1}\big), \tag{21.3}$$

$$\texttt{kkt}_4(k_0, k_1) := \mathbb{1}_{\{t\geq 0\}}\big(x^0_{k_1} - \theta(k_0, k_1)\big), \tag{21.4}$$

$$\texttt{kkt}_5(k_0, k_1) := \mathbb{1}_{\{t>0\}}\big(\theta(k_0, k_1) - x^0_{k_1+1}\big), \tag{21.5}$$

where $\mathbb{1}_{\{t\in S\}}(x) = \texttt{true}$ is $x \in S$ and $\texttt{false}$ otherwise; and where $\lambda(\bullet, \bullet)$ and $\theta(\bullet, \bullet)$ denote the values of λ and θ corresponding to a particular index-pair. By (20), an index-pair (k_0, k_1) is optimal if and only if

$$\texttt{kkt}_1(k_0, k_1) \wedge \texttt{kkt}_2(k_0, k_1) \wedge \texttt{kkt}_3(k_0, k_1) \wedge \texttt{kkt}_4(k_0, k_1) \wedge \texttt{kkt}_5(k_0, k_1). \tag{22}$$

Note that the presence of strict inequalities precludes arguments that appeal to linear programming. Instead we conduct a detailed study of the KKT conditions directly.

We begin by arguing that starting from $(k_0, k_1) = (k-1, k)$, we may "forget" about checking conditions 1, 3, and 4 and instead only seek satisfaction of conditions 2 and 5. We refer to Fig. 2 when referencing "rows" and "columns" of the search space. We also delay verification of claims involving "direct computation" to "Appendix B.2" but provide references in the text.

Lemma 4 *Beginning from $(k_0, k_1) = (k-1, k)$, the trajectory taken by Algorithm 3 always satisfies KKT conditions 1, 3, and 4.*

Proof We proceed inductively. Let $k_0 = k - 1$ and $k_1 = k$ be the initial point and note that KKT conditions 1, 3, and 4 hold at (k_0, k_1) by direct computation ("Claim B.1"). This establishes the base case.

Next, let (k_0, k_1) be any candidate index-pair where $\texttt{kkt}_1(k_0, k_1)$, $\texttt{kkt}_3(k_0, k_1)$, and $\texttt{kkt}_4(k_0, k_1)$ hold and (k_0', k_1') be the next iterate generated by the procedure. To show that KKT conditions 1, 3, and 4 at (k_0', k_1'), consider the four cases based on which of the remaining KKT conditions (2 and 5) hold at (k_0, k_1). For shorthand, let $\texttt{kkt}_i = \texttt{kkt}_i(k_0, k_1)$ and $\texttt{kkt}_i' = \texttt{kkt}_i(k_0', k_1')$ for $i \in \{1, \ldots, 5\}$.

(i) $\texttt{kkt}_2 \wedge \texttt{kkt}_5$: The algorithm terminates at the current iterate, which is the unique solution.

(ii) $\texttt{kkt}_2 \wedge \neg\texttt{kkt}_5$: $k_1 \leftarrow \min\{k_1 + 1, n\}$. The index-pair $(k_0', k_1') = (k_0, k_1 + 1)$ is the next point generated by the algorithm. At (k_0', k_1'), the conditions $\texttt{kkt}_1'$, $\texttt{kkt}_3'$, and $\texttt{kkt}_4'$ hold because of the following argument.

- $\texttt{kkt}_1'$: Direct computation (Claim B.4(4)) shows that $\neg\texttt{kkt}_5 \iff \Delta_{k_1}\lambda(k_0, k_1) \geq 0 \iff \lambda(k_0, k_1 + 1) \geq \lambda(k_0, k_1)$. Since $\lambda(k_0, k_1) > 0$ by assumption, it holds that $\lambda(k_0', k_1') = \lambda(k_0, k_1 + 1) > 0$.
- $\texttt{kkt}_3'$: By definition, $\texttt{kkt}_3 \iff (\theta + \lambda)(k_0, k_1) \geq x^0_{k_0+1}$, so it suffices to show that $(\theta + \lambda)(k_0', k_1') \geq (\theta + \lambda)(k_0, k_1)$. The desired condition is

equivalent to $\Delta_{k_1}(\theta+\lambda)(k_0,k_1) \geq 0$. Direct computation (Claim B.4(5)) shows that $\Delta_{k_1}(\theta+\lambda)(k_0,k_1) \geq 0 \iff \neg\texttt{kkt}_5$. Since $\neg\texttt{kkt}_5$ holds by assumption, the desired condition $\texttt{kkt}_3'$ holds.

– $\texttt{kkt}_4'$: Direct computation (Claim B.2(4)) shows that $\neg\texttt{kkt}_5 \iff \texttt{kkt}_4'$.

(iii) $\neg\texttt{kkt}_2 \wedge \neg\texttt{kkt}_5$: $k_0 \leftarrow \max\{k_0-1, 0\}$. The index-pair $(k_0', k_1') = (k_0-1, k_1)$ is the next point generated by the algorithm. At (k_0', k_1'), the conditions $\texttt{kkt}_1'$, $\texttt{kkt}_3'$, and $\texttt{kkt}_4'$ hold because of the following argument.

- $\texttt{kkt}_1'$: It suffices to show that $\lambda(k_0-1,k_1) \geq \lambda(k_0,k_1)$, *i.e.,* $\Delta_{k_0}\lambda(k_0-1,k_1) \leq 0$. Direct computation (Claim B.4(2)) shows that $\neg\texttt{kkt}_2 \iff \Delta_{k_0}\lambda(k_0-1,k_1) \leq 0$.
- $\texttt{kkt}_3'$: Direct computation (Claim B.2(2)) shows $\neg\texttt{kkt}_2 \iff \texttt{kkt}_3'$.
- $\texttt{kkt}_4'$: Since $\texttt{kkt}_4 \iff x^0_{k_1} \geq \theta(k_0,k_1)$ and $\texttt{kkt}_4' \iff x^0_{k_1} \geq \theta(k_0-1,k_1)$, it suffices to show that $\theta(k_0,k_1) \geq \theta(k_0-1,k_1)$, *i.e.,* $\Delta_{k_0}\theta(k_0-1,k_1) \geq 0$. Direct computation (Claim B.4(1)) shows that $\neg\texttt{kkt}_2 \iff \Delta_{k_0}\theta(k_0-1,k_1) \geq 0$.

Further, observe that transitions from case (iii) to case (iv) cannot occur, *i.e.,* that $\neg\texttt{kkt}_2 \wedge \neg\texttt{kkt}_5$ cannot transition to $\neg\texttt{kkt}_2' \wedge \texttt{kkt}_5'$. To show this, it suffices to show that $\neg\texttt{kkt}_5'$ must hold. This is true because of the following argument:

$$\neg\texttt{kkt}_2 \overset{(a)}{\iff} \Delta_{k_0}\theta(k_0-1,k_1) \geq 0 \iff \theta(k_0-1,k_1) \leq \theta(k_0,k_1) \overset{(b)}{\leq} x^0_{k_1+1}$$

where (a) follows from direct computation (Claim B.4(1)) and (b) follows from the definition of $\neg\texttt{kkt}_5$. Thus $\neg\texttt{kkt}_5' \equiv \neg\texttt{kkt}_5(k_0-1,k_1)$ must hold.

(iv) $\neg\texttt{kkt}_2 \wedge \texttt{kkt}_5$: $k_0 \leftarrow \max\{k_0-1, 0\}$. By the argument in case (iii), transitions to case (iv) must come from case (ii). The next point generated by the algorithm is $(k_0', k_1') = (k_0-1, k_1)$. At (k_0', k_1'), the conditions $\texttt{kkt}_1'$, $\texttt{kkt}_3'$, and $\texttt{kkt}_4'$ hold because of the following argument.

- $\texttt{kkt}_1'$: Direct computation (Claim B.4(2)) shows that $\neg\texttt{kkt}_2 \iff \Delta_{k_0}\lambda(k_0-1,k_1) \leq 0$, which implies that $\lambda(k_0-1,k_1) \geq \lambda(k_0,k_1) > 0$.
- $\texttt{kkt}_3'$: Direct computation (Claim B.2(2)) shows that $\neg\texttt{kkt}_2 \iff \texttt{kkt}_3$.
- $\texttt{kkt}_4'$: Direct computation (Claim B.4(1)) shows that $\neg\texttt{kkt}_2 \iff \Delta_{k_0}\theta(k_0-1,k_1) \geq 0$, which implies that $\theta(k_0-1,k_1) \leq \theta(k_0,k_1) \leq x^0_{k_1}$.

Finally, setting $x^0_0 := +\infty$ and $x^0_{n+1} := -\infty$, it is clear that the algorithm is guaranteed to terminate since the last index-pair possibly scanned is $(k_0,k_1) = (0,n)$, and if $k_0 = 0$, then $\texttt{kkt}_2$; otherwise, if $k_1 = n$, then $\texttt{kkt}_5$. Therefore, the procedure maintains satisfaction of KKT conditions 1, 3, and 4 throughout its trajectory. ☐

It remains only to identify an index-pair that satisfies both conditions 2 and 5. Toward this end, we justify stopping the inner k_0 loop early (within a column) if an index-pair is found that satisfies condition 2 by showing that no index-pair "above" the current iterate (in the same column) can satisfy condition 2.

Lemma 5 (Early stop) *Suppose that KKT condition 2 holds at a candidate index-pair* (k_0,k_1) *along the trajectory of Algorithm* 3 (e.g., *in case (ii) of the proof of Lemma*

4*). Then there does not exist an index-pair* (k_0', k_1) *that satisfies KKT condition 2 for* $k_0' < k_0$.

Proof The claim holds because for any k_1, there is at most one element $k_0' \in \{0, \ldots, k-1\}$ such that $\texttt{kkt}_2(k_0', k_1)$ and $\texttt{kkt}_3(k_0', k_1)$ both hold. The proof is by contradiction. Fix any $k_1 \in \{k, \ldots, n\}$, and suppose that there exist two indices k_0' and k_0'' for which $\texttt{kkt}_2(k_0', k_1) \wedge \texttt{kkt}_3(k_0', k_1)$ and $\texttt{kkt}_2(k_0'', k_1) \wedge \texttt{kkt}_3(k_0'', k_1)$ hold. By direct computation (Claim B.3), the sets $K^2(k_1)$ $(K^{\neg 2}(k_1))$ and $K^3(k_1)$ $(K^{\neg 3}(k_1))$ where KKT conditions 2 and 3 are satisfied (not satisfied) are contiguous, so it suffices to consider $k_0'' = k_0' - 1$. By direct computation (Claim B.2(2)), $\texttt{kkt}_2(k_0', k_1) \iff \neg\texttt{kkt}_3(k_0' - 1, k_1) = \neg\texttt{kkt}_3(k_0'', k_1)$, a contradiction. Therefore there can be at most one element that satisfies KKT conditions 2 and 3, and stopping early is justified. □

Finally, we justify the "late start" of the inner k_0 loop after moving from column k_1 to $k_1 + 1$.

Lemma 6 (Late start) *For a given* k_0*, consider a transition from* k_1 *to* $k_1 + 1$ *along the trajectory of Algorithm 3. The optimal solution* $(\bar{k}_0, \bar{k}_1)$ *must satisfy* $\bar{k}_0 \leq k_0$ *and* $\bar{k}_1 \geq k_1$.

Proof Evaluate KKT condition 2 at the new iterate (k_0, k_1+1). If $\texttt{kkt}_2(k_0, k_1+1)$, then by the preceding two lemmas, k_0 is the unique element from the column associated with $k_1 + 1$ that satisfies both KKT conditions 2 and 3. Otherwise, if $\neg\texttt{kkt}_2(k_0, k_1 + 1)$, then by direct computation (Claim B.3), any index $k_0' > k_0$ will not satisfy KKT condition 2 and therefore not be optimal. □

Combining our prior results, we obtain the desired conclusion.

Proposition 2 *The procedure given in Algorithm 3 terminates at the unique solution* $(\bar{k}_0, \bar{k}_1)$ *in at most* n *steps using* $O(n)$ *elementary operations.*

Proof The conclusion is an immediate consequence of Lemmas 4, 5, and 6 and the form of the updates in Algorithm 3. Since there can be at most $n - k$ transitions of the form $k_1 \leftarrow k_1 + 1$ and at most k transitions of the form $k_0 \leftarrow k_0 - 1$, there are at most n total transitions, and hence the proof is complete. □

3.3 Partial sorting

Until now, the sorted problem (7) has assumed that a full sorting permutation is given and has been applied to the input data $x^0 \in \mathbb{R}^n$. However, it is of significant practical interest to note that since the elements of $\bar{\gamma}$ are unperturbed, it is only necessary that the initial permutation that sort the largest $L \geq \bar{k}_1 \equiv n - |\bar{\gamma}|$ indices of $x^0 \in \mathbb{R}^n$. Let us call a permutation with $L \geq \bar{k}_1$ an "optimal" permutation, which can be obtained offline in $O(L \log n)$ by `heapsort`. When $k \ll n$, then $|\gamma|$ may be close to n, and the initial cost of obtaining an optimal permutation may be greatly reduced. Since $\bar{k}_1$ is not known at runtime, though, determining whether a candidate permutation is optimal *a priori* is not possible. However, in the following proposition, we provide an implementable condition for checking whether the result of a projection based on a candidate permutation is indeed optimal.

On the other hand, we may instead seek to construct an optimal permutation in an "as-needed" fashion by embedding the sorting within the solution procedure. That is, since Algorithms 1 and 3 only inspect elements of x^0 in a contiguous and increasing subset of $\{1, \ldots, n\}$, the sorting may be performed online. This leads to an $O(\bar{k}_1 \log n)$ overall procedure, despite the fact that $\bar{k}_1$ is not known at runtime, as summarized in the following proposition.

Proposition 3 *The following two claims hold for any given $x^0 \in \mathbb{R}^n$:*

1. *Let $\hat{\pi}$ be a given permutation of x^0 (not necessarily in the nonincreasing order). Define $\hat{x} := \mathrm{proj}_{\mathcal{B}^r_{(k)}}(x^0_{\hat{\pi}})$ and $\tilde{x} := \mathrm{proj}_{\mathcal{B}^r_{(k)}}(x^0)$ as the solutions of the candidate and unsorted problems, respectively, and let $\hat{k}_1$ be the index corresponding to $\hat{x}$ in (3). If (i) $\hat{x}_{\hat{k}_1} > (x^0_{\hat{\pi}})_i$ for all $i \in \{\hat{k}_1 + 1, \ldots, n\}$, and (ii) the elements in $\hat{x}_{1:\hat{k}_1}$ are sorted in the nonincreasing order, then one can obtain $\tilde{x}$ via $(\tilde{x}_{\hat{\pi}})_{1:\hat{k}_1} = \hat{x}_{1:\hat{k}_1}$ and $(\tilde{x}_{\hat{\pi}})_{\hat{k}_1+1:n} = (x^0_{\hat{\pi}})_{\hat{k}_1+1:n}$,* i.e., *$\hat{\pi}$ is an optimal permutation.*
2. *A solution $\tilde{x}$ to the unsorted problem can be obtained in $O(\bar{k}_1 \log n)$ operations, where $\bar{k}_1 = n - |\bar{\gamma}|$ is the second component of the unique optimal index-pair satisfying KKT conditions (21), which is unknown-at-runtime but can be identified dynamically.*

Proof Let $\bar{\pi}$ be a full (and hence optimal) sorting permutation of x^0 so that $x^0_{\bar{\pi}} = \vec{x}^0$. The KKT conditions require a solution to satisfy $\bar{x}_{\bar{k}_1} \equiv \bar{\theta} > (x^0_{\bar{\pi}})_{\bar{k}_1+1}$. By the ordering on $x^0_{\bar{\pi}}$, it holds that $(x^0_{\bar{\pi}})_i \leq (x^0_{\bar{\pi}})_{\bar{k}_1+1}$ for all $i \geq \bar{k}_1 + 1$. If $\hat{x}$ satisfies all the KKT conditions associated with the $\hat{\pi}$-permuted problem and $(\hat{x}_{\hat{\pi}})_{\hat{k}_1} > (x^0_{\hat{\pi}})_i$ for all $i \in \{\hat{k}_1 + 1, \ldots, n\}$, then it satisfies the KKT conditions of the fully sorted problem.

To justify the $O(\bar{k}_1 \log(n))$ complexity, we argue that the `heapsort` algorithm can be embedded within the iterative approaches of Algorithms 1 and 3. We provide an argument for Algorithm 1 and note that Algorithm 3 can be handled similarly. Explicitly, construct a (binary) max-heap based on the input vector x^0 in $O(n)$ cost. The largest element of any heap can be extracted in $O(\log n)$ time. Therefore, sorted elements $\vec{x}^0_1, \ldots, \vec{x}^0_\ell$ can be obtained in $O(\ell \log n)$ time by by extracting the maximum element of the (successively modified) heap ℓ times. Next, given a suboptimal candidate index-pair (k_0, k_1), the next pivot requires inspecting either $\vec{x}^0_{k_0-1}$ or $\vec{x}^0_{k_1+1}$; the former can be stored from the initial extraction process, and the latter can be obtained in $O(\log n)$ time from the modified heap. Since the optimal index-pair $(\bar{k}_0, \bar{k}_1)$ is unique, the procedure performs exactly $\bar{k}_1$ extractions for a cost of $O(\bar{k}_1 \log n)$. □

Proposition 3 is useful in applications in which the solution to the projection problem is not expected to change significantly from iteration to iteration. An important example of this is in solving a sequence of related projection problems (such as in linesearch or projected gradient descent) when the input vector does not change significantly. For the first problem in the sequence, the online sorting property may be used to identify an initial permutation $\bar{\pi}^{(1)} \equiv \hat{\pi}^{(1)}$ in $O(\bar{k}_1 \log n)$ via claim 2. For subsequent problems (indexed by ν), the previous problem's sorting permutation $\hat{\pi}^{(\nu)}$ may be used to warm-start the construction of an approximate sorting permutation $\hat{\pi}^{(\nu+1)}$.

3.4 Relation to the vector-*k*-norm ball

We now briefly state how the previous two approaches can be utilized when solving the related but distinct (and slightly more complicated) problem of projection onto the (Ky-Fan) vector-k-norm ball studied in [42]. The vector-k-norm ball of radius $r \geq 0$ is defined by $\mathcal{V}^r_{(k)} := \{z \in \mathbb{R}^n : \sum_{i=1}^k |\overleftarrow{z}|_i \leq r\}$, and the sorted vector-$k$-norm problem only differs from the sorted top-k-sum problem (7) by the additional constraint $z_n \geq 0$. That is, the sorted formulation

$$\bar{z} := \underset{z \in \mathbb{R}^n}{\arg\min} \left\{ \tfrac{1}{2} \| z - \overleftarrow{z}^0 \|_2^2 : \mathbb{1}_k^\top z \leq r, \; z_i \geq z_{i+1}, \; \forall i \in \{1, \ldots, n-1\}, \; z_n \geq 0 \right\} \tag{23}$$

has polyhedral region $\{z \in \mathbb{R}^n : Vz \leq v\}$ with data $V = \begin{bmatrix} (\mathbb{1}_k^\top, \, 0_{n-k}^\top) \\ -E \end{bmatrix}$, $E := \begin{bmatrix} D \\ (0_{n-1}^\top, \, 1) \end{bmatrix}$, $v := (r, 0_n^\top)^\top$, where D is the isotonic difference operator defined in (8). The penalized problem with PLCP data $M := EE^\top, q := E\overleftarrow{z}^0$ and direction vector $d := -E\mathbb{1}_k = -e^k$ shares nearly identical structure with (12), so pivots can be performed in a similar manner to the approach outlined in Algorithm 1.

On the other hand, [42] provide a two-step routine (Algorithm 4) for solving the vector-k-norm projection problem based on the observation that the k^{th} largest value of the solution must satisfy (i) $\bar{z}_{[k]} = 0$; or (ii) $\bar{z}_{[k]} > 0$. The first step identifies a solution satisfying condition (i), if one exists, in $O(k)$ complexity; otherwise if it does not exist, the second step identifies a solution satisfying condition (ii) by performing a grid search over all index-pairs (k_0, k_1). The second step is the algorithm that we refer to as the "KKT grid-search" method in Sect. 2. The KKT conditions of the second case coincide with the KKT conditions of the top-k-sum problem, so Algorithm 3 can be substituted for the second step in [42, Algorithm 4], yielding a procedure with overall complexity of $O(n)$ on sorted input vector $\overleftarrow{z}^0$. Each step needs to run sequentially, though, so the ESGS-based approach incurs an additional $O(k)$ cost in instances that are not solved in the first step.

4 Numerical experiments

To evaluate the performance of our proposed algorithms, we conduct a series of numerical experiments on synthetic datasets. We implement the algorithms in `Julia` [5] and execute the tests on a 125GB RAM machine with Intel(R) Xeon(R) W-2145 CPU @ 3.70GHz processors running `Julia v1.9.1`.

The experimental problems were formulated based on the following protocol: the index k and right-hand side r are set as $k = \tau_k^{\mathsf{c}} \cdot n$ and $r = \tau_r \cdot \mathsf{T}_{(k)}(x^0)$, where $\tau_k^{\mathsf{c}} = 1 - \tau_k$ and τ_r take on values from the sets:

- $\tau_r \in \{-8, \, -4, \, -2, \, -1, \, -\frac{1}{2}, \, -\frac{1}{10}, \, 0, \, \frac{1}{10}, \, \frac{1}{2}, \, \frac{9}{10}, \, \frac{99}{100}, \, \frac{999}{1000}\}$;
- $\tau_k^{\mathsf{c}} \in \{\frac{1}{10000}, \, \frac{1}{1000}, \, \frac{1}{100}, \, \frac{5}{100}, \, \frac{1}{10}, \, \frac{1}{2}, \, \frac{9}{10}, \, \frac{99}{100}, \, \frac{999}{1000}, \, \frac{9999}{10000}\}$.

For context, in many practical scenarios (*e.g.,* the risk-averse superquantile constrained problems), the values of τ_k^{c} typically fall between 1% and 10%. To facilitate more intuitive interpretation of our computational findings, initial vectors are generated uniformly from $[0, 1]^n$ in double precision. As τ_r approaches 1, the projection problems tend to become easier because $\mathsf{T}_{(k)}(x^0) \approx r$. Conversely, as τ_r trends towards $-\infty$, the projection problems become more challenging as the solution will have a substantial deviation from the original point. The problem dimension n is set from $n \in \{10^1, 10^2, \ldots, 10^7\}$, and 100 instances are generated for each scenario unless stated otherwise.

The ESGS, PLCP, GRID (our implementation of the grid-search method from [42]), and SSN (our implementation of the semismooth Newton method from [27]) are written in `Julia` and use double precision for the experiments, though they can also handle arbitrary precision floats and rational data types. The partition-based method [15] was not studied since it was found to be significantly slower than the SSN method in [27]. The finite-termination methods use a single core but can make use of `simd` operations. The QP solver utilizes the barrier method provided by Gurobi `v10.0` to solve both the sorted (7) and unsorted (5) formulations, called GRBS and GRBU, respectively. The feasibility and optimality tolerances are set to 10^{-9}, the presolve option to the default, and the method is configured to use up to 8 cores. Model initialization time is not counted towards Gurobi's solve time. Both Gurobi methods and the GRID method have time-limits of 10,000 seconds for each instance.

Results and discussion

The numerical results are summarized in Table 1, and Figs. 3 and 4.

Across all values of n, our two proposed methods consistently outperform the existing grid-search method, the Gurobi QP solver, and the SSN method, often achieving improvements by several orders of magnitude. The scaling profile in Fig. 3 reveals the linear behavior of our proposed methods, the sparsity-exploiting inexact method GRBU based on the formulation (5), and the SSN method. In contrast, the grid-search method exhibits quadratic scaling, and the sorted inexact method GRBS based on (7) exhibits performance that degrades in harder large-n cases. For problem sizes where $n \in \{10^6, 10^7\}$, the solution time of the grid-search and Gurobi methods is on the order of minutes or hours; on the other hand, our methods obtain solutions in fractions of a second. In addition, our procedures require significantly less computational time than the partial-sort time threshold, which the SSN method exceeds in each of the six experiments.

In Table 1, we highlight the computational results for a few experiments based on parameters which we expect to be of practical interest. In Experiments 1 through 3, we fix k to be a small proportion of n and vary the budget r from large to small; Experiments 4 through 6 follow a similar pattern, but with a larger value of k. A clear takeaway from the results is that the (full) sorting procedure requires more time than solution procedure of our proposed algorithms for sorted input. This highlights the importance of Proposition 3. Sorting is also more costly than the grid-search method for small n, but for moderate-to-large n, the $O(k(n-k))$ computational cost of the grid-

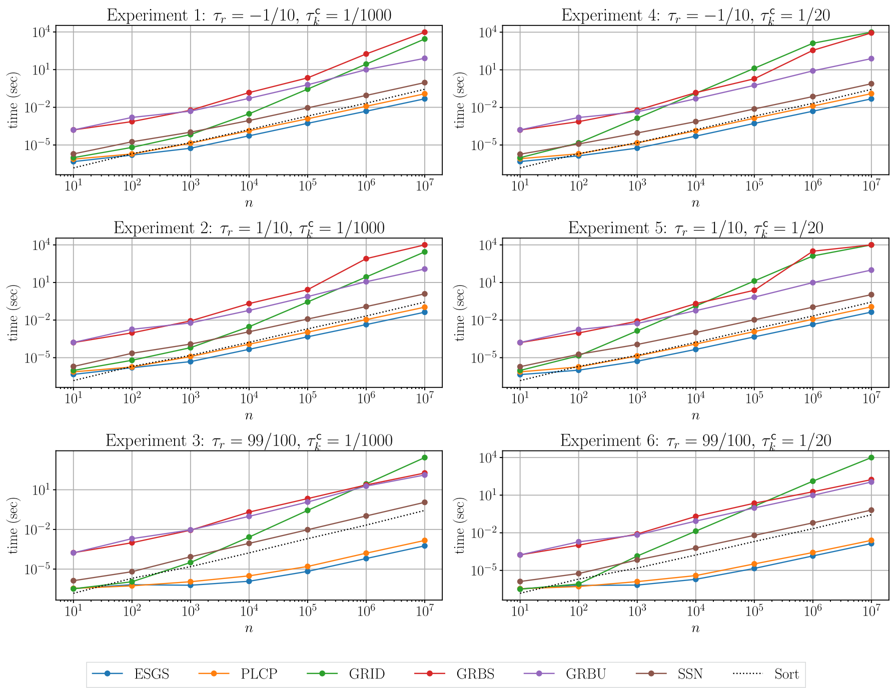
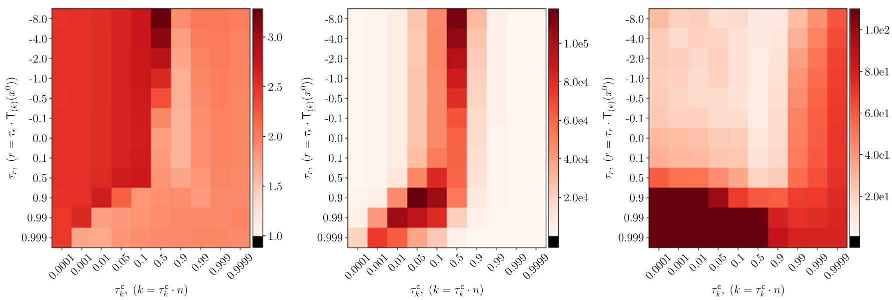


Fig. 3 Average total computation time excluding sort time and full sort time vs n. All results are averaged over 100 instances, except for methods GRID, GRBS, and GRBU with $n \in \{10^6, 10^7\}$ in which a time-limit of 10^4 seconds is imposed across 2 instances



(a) PLCP vs ESGS at $n = 10^7$.

(b) GRID vs ESGS at $n = 10^5$.

(c) SSN vs ESGS at $n = 10^7$.

Fig. 4 Computation time relative to ESGS averaged over 100 instances. A value of $c > 0$ indicates that ESGS was c times faster than the other method; a value of $c < 0$ indicates that the other method was c times faster than ESGS. Across all scenarios at $n = 10^5$, the better of GRBS and GRBU was at best ≈ 350 times slower than ESGS (and never faster)

Table 1 Mean computation time and (standard deviation) in seconds for projecting an initial sorted vector x^0 drawn uniformly from $[0,1]^n$ onto $\mathcal{B}^r_{(k)}$ with $r = \tau_r \cdot \mathbb{1}_k^\top x^0$ and $k = \tau_k^{\mathsf{c}} \cdot n$

	Experiment 1: $\tau_r = -1/10,\ \tau_k^{\mathsf{c}} = 1/1000$				
	$n = 10^3$	$n = 10^4$	$n = 10^5$	$n = 10^6$	$n = 10^7$
ESGS	**5.5e-6** (1e-6)	**5.4e-5** (1e-5)	**5.3e-4** (10e-5)	**4.9e-3** (6e-5)	**4.7e-2** (10e-4)
PLCP	1.4e-5 (2e-6)	1.4e-4 (3e-5)	1.2e-3 (2e-4)	1.2e-2 (3e-4)	1.2e-1 (3e-3)
GRID	6.8e-5 (8e-6)	3.0e-3 (2e-4)	2.8e-1 (1e-2)	2.7e+1* (5e-2*)	2.8e+3* (3e0*)
GRBS	5.8e-3 (4e-4)	1.5e-1 (10e-3)	2.2e+0 (2e-1)	1.8e+2* (1e-1*)	9.6e+3* (2e1*)
GRBU	4.9e-3 (3e-4)	5.1e-2 (9e-3)	6.2e-1 (10e-2)	9.9e+0* (8e-2*)	7.9e+1* (1e0*)
SSN	1.0e-4 (8e-6)	8.8e-4 (3e-5)	9.0e-3 (2e-4)	8.8e-2 (4e-4)	9.3e-1 (3e-2)
	Experiment 2: $\tau_r = 1/10,\ \tau_k^{\mathsf{c}} = 1/1000$				
	$n = 10^3$	$n = 10^4$	$n = 10^5$	$n = 10^6$	$n = 10^7$
ESGS	**4.9e-6** (1e-7)	**4.7e-5** (8e-6)	**4.6e-4** (7e-5)	**4.4e-3** (5e-5)	**4.3e-2** (6e-4)
PLCP	1.2e-5 (1e-6)	1.2e-4 (2e-5)	1.1e-3 (2e-4)	1.1e-2 (2e-4)	1.1e-1 (2e-3)
GRID	6.3e-5 (5e-6)	2.9e-3 (2e-4)	2.8e-1 (1e-2)	2.7e+1* (6e-2*)	2.8e+3* (4e0*)
GRBS	8.6e-3 (1e-3)	2.1e-1 (1e-2)	2.7e+0 (3e-1)	8.0e+2* (2e0*)	1.0e+4* (−*)
GRBU	5.9e-3 (5e-4)	5.9e-2 (7e-3)	7.5e-1 (1e-1)	1.1e+1* (2e-1*)	1.2e+2* (4e0*)
SSN	1.2e-4 (1e-5)	1.2e-3 (4e-5)	1.2e-2 (1e-3)	1.2e-1 (6e-3)	1.2e+0 (4e-2)
	Experiment 3: $\tau_r = 99/100,\ \tau_k^{\mathsf{c}} = 1/1000$				
	$n = 10^3$	$n = 10^4$	$n = 10^5$	$n = 10^6$	$n = 10^7$
ESGS	**6.0e-7** (1e-7)	**1.2e-6** (3e-7)	**6.7e-6** (1e-6)	**6.3e-5** (7e-6)	**5.7e-4** (1e-5)
PLCP	1.1e-6 (2e-7)	3.1e-6 (10e-7)	1.6e-5 (3e-6)	1.6e-4 (7e-6)	1.5e-3 (3e-5)
GRID	3.2e-5 (2e-6)	2.6e-3 (5e-4)	2.7e-1 (2e-2)	2.7e+1* (7e-2*)	2.7e+3* (4e0*)
GRBS	8.7e-3 (6e-4)	2.1e-1 (1e-2)	2.1e+0 (2e-1)	2.3e+1* (4e-1*)	1.8e+2* (3e-1*)
GRBU	9.2e-3 (4e-4)	9.8e-2 (1e-2)	1.2e+0 (1e-1)	1.9e+1* (7e-1*)	1.3e+2* (2e-1*)
SSN	8.6e-5 (9e-6)	8.8e-4 (6e-5)	9.5e-3 (6e-4)	1.0e-1 (6e-3)	1.1e+0 (6e-3)

Table 1 continued

	Experiment 1: $\tau_r = -1/10,\ \tau_k^c = 1/1000$				
	$n = 10^3$	$n = 10^4$	$n = 10^5$	$n = 10^6$	$n = 10^7$
	Experiment 4: $\tau_r = -1/10,\ \tau_k^c = 1/20$				
	$n = 10^3$	$n = 10^4$	$n = 10^5$	$n = 10^6$	$n = 10^7$
ESGS	**5.6e-6** (1e-6)	**5.1e-5** (3e-6)	**5.3e-4** (9e-5)	**5.0e-3** (5e-5)	**4.8e-2** (4e-4)
PLCP	1.5e-5 (2e-6)	1.4e-4 (10e-6)	1.4e-3 (2e-4)	1.3e-2 (2e-4)	1.3e-1 (2e-3)
GRID	1.4e-3 (9e-5)	1.3e-1 (2e-3)	1.3e+1 (1e-1)	1.3e+3* (2e0*)	1.0e+4* (−*)
GRBS	6.0e-3 (3e-4)	1.5e-1 (9e-3)	1.9e+0 (2e-1)	3.6e+2* (3e0*)	8.8e+3* (2e0*)
GRBU	4.4e-3 (2e-4)	4.9e-2 (8e-3)	5.7e-1 (7e-2)	8.2e+0* (1e-1*)	7.8e+1* (6e0*)
SSN	9.1e-5 (5e-6)	7.3e-4 (3e-5)	7.5e-3 (2e-4)	7.3e-2 (2e-3)	7.7e-1 (3e-2)
	Experiment 5: $\tau_r = 1/10,\ \tau_k^c = 1/20$				
	$n = 10^3$	$n = 10^4$	$n = 10^5$	$n = 10^6$	$n = 10^7$
ESGS	**5.2e-6** (7e-7)	**4.6e-5** (2e-6)	**4.6e-4** (7e-6)	**4.5e-3** (5e-5)	**4.4e-2** (4e-4)
PLCP	1.4e-5 (3e-6)	1.2e-4 (4e-6)	1.2e-3 (2e-5)	1.2e-2 (2e-4)	1.1e-1 (2e-3)
GRID	1.4e-3 (8e-5)	1.3e-1 (2e-3)	1.3e+1 (1e-1)	1.3e+3* (2e0*)	1.0e+4* (−*)
GRBS	8.2e-3 (7e-4)	2.0e-1 (1e-2)	2.4e+0 (2e-1)	3.1e+3* (2e1*)	1.0e+4* (−*)
GRBU	5.3e-3 (3e-4)	5.7e-2 (5e-3)	6.7e-1 (5e-2)	9.8e+0* (2e0*)	9.8e+1* (4e0*)
SSN	1.1e-4 (9e-6)	1.0e-3 (4e-5)	1.0e-2 (6e-5)	1.1e-1 (2e-3)	1.1e+0 (6e-3)
	Experiment 6: $\tau_r = 99/100,\ \tau_k^c = 1/20$				
	$n = 10^3$	$n = 10^4$	$n = 10^5$	$n = 10^6$	$n = 10^7$
ESGS	**6.7e-7** (2e-7)	**1.9e-6** (2e-7)	**1.4e-5** (3e-6)	**1.4e-4** (3e-6)	**1.4e-3** (2e-5)
PLCP	1.3e-6 (2e-7)	3.8e-6 (1e-6)	3.3e-5 (4e-6)	2.7e-4 (6e-6)	2.5e-3 (5e-5)
GRID	1.4e-4 (5e-5)	1.3e-2 (2e-3)	1.3e+0 (6e-2)	1.3e+2* (2e0*)	1.0e+4* (−*)
GRBS	8.0e-3 (5e-4)	2.0e-1 (2e-2)	2.3e+0 (3e-1)	1.9e+1* (2e-1*)	1.7e+2* (4e0*)

Table 1 continued

	Experiment 1: $\tau_r = -1/10,\ \tau_k^c = 1/1000$				
	$n = 10^3$	$n = 10^4$	$n = 10^5$	$n = 10^6$	$n = 10^7$
GRBU	6.8e-3 (3e-4)	8.5e-2 (2e-2)	9.5e-1 (7e-2)	9.7e+0* (2e0*)	1.1e+2* (4e0*)
SSN	6.8e-5 (9e-6)	6.0e-4 (4e-5)	6.2e-3 (2e-4)	6.2e-2 (5e-4)	6.4e-1 (4e-3)
Full	1.5e-5	1.7e-4	2.0e-3	2.1e-2	2.7e-1
Top-1%	8.0e-6	7.2e-5	7.4e-4	7.0e-3	7.3e-2

Statistics are computed over 100 (2*) instances, and the fastest method is listed in bold. A dash "–" for standard deviation indicates that the experiment timed out in each instance. The Full and Top-1% fields are sort times that are listed for reference but are not counted towards the solve time for any method search method dominates the sorting time, even for very small τ_k as in Experiments 1 through 3.

Another observation from Table 1 is that the performance of the finite-termination algorithms is problem-dependent, with the grid-search method being the most variable. On the other hand, the performance of the Gurobi QP solver based on the unsorted formulation (5) is relatively stable across different instances, in addition to being significantly more efficient than the sorted formulation (7). A plausible reason for this phenomenon is that the number of active constraints at the solutions are different for the two formulations: for instances in Experiments 3 and 6 with $n = 10^6$, the sorted formulation yields averages of 10^4 and 9.6×10^3 number of active constraints, compared to 9.8×10^3 and 3.8×10^3 for the unsorted formulation. Theoretically, only $|\beta|$ number of constraints (see Fig. 1) should be binding for the unsorted formulation, whereas many more constraints could be binding for the sorted formulation.

Figure 4 compares the relative performance of the proposed methods across the entire spectrum of the parameters r and k. Figure 4a shows that ESGS performs about 1.5-2 times better (and never worse) than PLCP across the spectrum, though its advantage degrades in the easier instances in which $\tau_r \approx 1$. Figure 4b shows that ESGS performs significantly better than GRID in many cases of practical interest (see the bottom of the "backwards L" in the lower lefthand corner) where we observe excesses of 10^4-fold run-time improvement. Figure 4c shows that ESGS performs no less than about 5 times faster than SSN and can be orders of magnitude faster in instances where the input vector nearly satisfies the constraint.

5 Conclusions

We have provided two efficient, finite-termination algorithms, PLCP and ESGS, that are capable of exactly solving the top-k-sum sublevel set projection problem (1). When the input vector is unsorted, the solution requires a floating point complexity of $O(n \log n)$, and when sorted, it reduces to $O(n)$. These implementations improve upon existing methods by orders of magnitude in many cases of interest; notably, they can be over 100 times faster than Gurobi, the grid-search method, and the SSN method. Our numerical experiments also show that ESGS is faster than PLCP by a factor of ≈ 2 in harder instances where many pivots are required while maintaining a slight advantage in easier instances that require fewer pivots. Such instances may arise when solving a sequence of similar problems, as is the case when employing an iterative method to solve superquantile constrained composite optimization problems in the form of (2), which necessitate repeated calls to a projection oracle. Moreover, our proposed techniques, with minimal modifications, can be applied to compute the projection onto the vector-k-norm ball. In this case, PLCP can avoid incurring an additional $O(k)$ cost that an ESGS-based approach unavoidably pays (see Step 1 in [42, Algorithm 4]). Finally, it is anticipated that projection onto the top-k-sum sublevel set can find use in projection onto more complex composite superquantile regions, which can be leveraged within iterative solvers for addressing general composite superquantile problems such as (2).

A Proofs for Section 2 (Equivalent Formulations and Existing Techniques)

Lemma 1 *The optimal solution of problem* (1) *is the same as that of the unsorted top-k problem* (5).

Proof Equivalence follows from a direct application of the observation that for any $y, z \in \mathbb{R}^n$,

$$\langle y, z\rangle \le \langle \overleftarrow{y}, \overleftarrow{z}\rangle, \qquad (*)$$

and equality holds if and only if there exists a permutation π that simultaneously sorts y and z, *i.e.,* $y_\pi = \overleftarrow{y}$ and $z_\pi = \overleftarrow{z}$.

Because the objectives are strongly convex, both problems have unique solutions $\tilde{x}^{(1)}$ and $\tilde{x}^{(5)}$. Let $\mathcal{F}_{(1)}$ and $\mathcal{F}_{(5)}$ denote the feasible regions of the projection problems, and $\nu_{(1)}$ and $\nu_{(5)}$ denote the optimal values. It is clear that $\mathcal{F}_{(5)} \subseteq \mathcal{F}_{(1)}$ (in general, strict subset), so $\nu_{(1)} \le \nu_{(5)}$. On the other hand, by $(*)$ both $\tilde{x}^{(1)}$ and $\tilde{x}^{(5)}$ must have the same ordering as x^0, and thus $\tilde{x}^{(1)} \in \mathcal{F}_{(5)}$, so $\nu_{(5)} \le \nu_{(1)}$ by the optimality of $\tilde{x}^{(5)}$ for (5) and the fact that both problems share the same objective function. Therefore the problems are equivalent. □

Lemma 2 *Let B be the unsorted-top-k matrix defined in* (5) *and $c \in \mathbb{R}^n$ be arbitrary. Then*

$$\delta^*_{\mathcal{B}^r_{(k)}}(c) = \begin{cases} \frac{r}{k}\mathbb{1}^\top c, & \text{if } B^{-\top}c \ge 0;\\ +\infty, & \text{otherwise.}\end{cases}$$

In addition, the condition $B^{-\top}c \ge 0$ can be checked in $O(n + k + (n-k)\log k)$ time for the worst case and $O(n + k\log n)$ time in expectation. Furthermore, a sufficient condition for $B^{-\top}c \ge 0$ can be checked in $O(n)$ time.

Proof For $c \in \mathbb{R}^n$, we have for B and b defined in (5)

$$\begin{aligned}\delta^*_{\mathcal{B}^r_{(k)}}(c) &= \max_x\{c^\top x - \delta_{\mathcal{B}^r_{(k)}}(x)\} = \max_x\{c^\top x : Bx \le b\} && \text{(definition)}\\ &= \max_y\{c^\top B^{-1}y : y \le b\} = \max_y\{(B^{-\top}c)^\top y : y \le b\}. && (B \text{ is invertible})\end{aligned}$$

Suppose that there is an index i^* such that $(B^{-\top}c)_{i^*} < 0$. Then by taking $y_{i^*} \downarrow -\infty$, the objective tends to $+\infty$. When $B^{-\top}c \ge 0$, then the problem has an optimal (finite) solution that occurs at an extreme point, *i.e.,* $y^* = b$, with objective value $c^\top B^{-1}b = \frac{r}{k}\mathbb{1}^\top c$ by direct verification.

Next consider verifying the condition $B^{-\top}c \ge 0$. First identify the index $[k] = {}_{c_{[k]}}c$ (the index of the k^{th} largest element of c) in $O(k+(n-k)\log k)$ by using max-heaps or $O(n)$ expected time using `quickselect`. Next, scan c to identify the elements and

$\kappa = \{i \in \{1, \ldots, n\} : c_i \geq c_{[k]}\}$, ensuring that $|\kappa| = k$ (ties can be split arbitrarily) in $O(n)$ time. Then, the form of B^{-1} can be verified to take the form

$$B^{-1} = \begin{matrix} i=[k] & i \in \kappa \setminus [k] & i \notin \kappa \\ \big[\frac{1}{k}\mathbb{1} & V & W\big] \end{matrix}$$

where V is a matrix of columns of the form $\frac{1}{k}\mathbb{1} - e^i$ for $i \in \kappa \setminus [k]$ for standard basis vector e^i and W is a matrix of columns of the form e^i for $i \notin \kappa$. Thus $B^{-\top}c \geq 0$ can be checked in $O(n + k + (n-k)\log k)$ time or $O(n)$ expected time by checking (i) $c_i \geq 0$ for $i \notin \kappa$; and (ii) $\frac{1}{k}\mathbb{1}^\top c - c_i \geq 0$ for $i \in \kappa \setminus [k]$. As a consequence of the form of B^{-1}, a sufficient condition for $B^{-\top}c \geq 0$ is: $c \geq 0$ and $c_i \leq \frac{1}{k}\mathbb{1}^\top c$ for all $i \in \{1, \ldots, n\}$, which can be checked in $O(n)$ time by disregarding the identification of κ. □

B Proofs for Section 3 (Proposed Algorithms)

B.1 PLCP

Lemma 3 *Algorithm 2 computes $D^\top z$ in $O(|\xi|)$ from initial data $\overleftarrow{x}^0$ and optimal output of Algorithm 1: a, b, ${}_k\xi$, and λ.*

Using the above lemma, we arrive at the folowing conclusion.

Proof Let ξ be a given contiguous subset with $|\xi| = m$ and suppress the dependence on ξ. The goal is to compute $D_{\xi,:}^\top z_\xi$ where $z_\xi = -M_{\xi\xi}^{-1}(q_\xi + \lambda d_\xi)$. For convenience, we drop the dependence on ξ. The matrix $M^{-1} = (DD^\top)^{-1}$ is completely dense and symmetric with (lower triangular) entries given by

$$(DD^\top)^{-1}_{\triangleleft} = \frac{1}{m+1}\begin{bmatrix} 1\cdot m & & & \\ 1\cdot(m-1) & 2\cdot(m-1) & & \\ 1\cdot(m-2) & 2\cdot(m-2) & \ddots & \\ \vdots & \vdots & & \ddots \\ 1\cdot 2 & 2\cdot 2 & & \\ 1\cdot 1 & 2\cdot 1 & \cdots & m\cdot 1 \end{bmatrix} \in \mathbb{R}^{m\times m}.$$

By direct computation,

$$D^\top(DD^\top)^{-1} = \frac{1}{m+1}\left[\begin{array}{cccccc} m & m-1 & m-2 & \cdots & 2 & 1 \\ -1 & m-1 & m-2 & \cdots & 2 & 1 \\ -1 & -2 & m-2 & \cdots & 2 & 1 \\ \vdots & \vdots & \vdots & \ddots & \vdots & 1 \\ -1 & -2 & -3 & \cdots & 2 & 1 \\ -1 & -2 & -3 & \cdots & -(m-1) & 1 \\ \hline -1 & -2 & -3 & \cdots & -(m-1) & -m \end{array}\right] \in \mathbb{R}^{(m+1)\times m}.$$

Then it is clear that $D^\top(DD^\top)^{-1}v$ computes the difference of two cumulative sum vectors of v, $c^1 = \texttt{cumsum}((1,2,\ldots,m) \odot \texttt{reverse}(v))$ and $c^2 = \texttt{cumsum}((1,2,\ldots,m) \odot v)$, *i.e.*, $\left(D^\top(DD^\top)^{-1}v\right)^\top = (c^1, 0) - (0, c^2)$, where cumsum denotes the cumulative sum operation and reverse reverses the order of a vector. □

B.2 ESGS

In this section, we provide additional observations and verifications of the computations claimed in the analysis of Algorithm 3 in Sect. 3.2.3.

B.2.1 Simple observations

We begin by observing that for any candidate (k_0, k_1) with $k_0 \in \{0,\ldots,k-1\}$ and $k_1 \in \{k,\ldots,n\}$, it holds that $\rho(k_0,k_1){:=}k_0 \cdot (k_1 - k_0) + (k - k_0)^2 > 0$. Next we verify that conditions 1, 3, and 4 hold at the initial iterate.

Claim B.1 (Initial candidate index-pair $(k-1,k)$) *For any $k \geq 1$, it holds that (i)* $\texttt{kkt}_1(k-1,k)$*; (ii)* $\texttt{kkt}_3(k-1,k)$*; and* $\texttt{kkt}_4(k-1,k)$.

Proof Let $k \geq 1$. Using the fact that $\rho(k-1,k) = k > 0$, the proofs follow by direct computation.

1. Using $0 < \sum_{i=1}^k x_i^0 - r$ from Assumption 1, it holds that $\rho(k-1,k)\cdot\lambda(k-1,k) = k\cdot\lambda(k-1,k) = k\cdot\left(x_k^0 + \sum_{i=1}^{k-1} x_i^0 - r\right) > 0$.
2. Since $k \geq 1$ and since x^0 is sorted, it holds that $\rho(k-1,k)\cdot(\theta+\lambda)(k-1,k) = k\sum_{i=k}^k x_i^0 + (k-k)\cdot\left(\sum_{i=1}^{k-1} x_i^0 - r\right) = kx_k^0 \geq kx_{k_0-1+1}^0 = \rho(k-1,k)\cdot x_k^0$.
3. Using $0 < \sum_{i=1}^k x_i^0 - r$ from Assumption 1, it holds that $\rho(k-1,k)\cdot\theta(k-1,k) = (k-1)x_k^0 - (k-(k-1))\left(\sum_{i=1}^{k-1} x_i^0 - r\right) = kx_k^0 - \left(\sum_{i=1}^k x_i^0 - r\right) < kx_k^0 = \rho(k-1,k)\cdot x_k^0$.

Thus the claims are proved. □

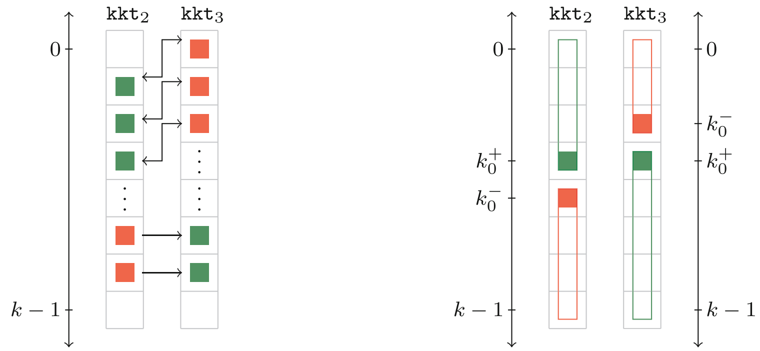
B.2.2 Linking conditions

Next we summarize characterize some relationships between various KKT conditions as depicted in Fig. 5.

Claim B.2 (Linking 2 & 3 and 4 & 5) *It holds that*

1. $\neg\texttt{kkt}_2(k_0,k_1) \implies \texttt{kkt}_3(k_0,k_1)$*;*
2. $\texttt{kkt}_2(k_0,k_1) \iff \neg\texttt{kkt}_3(k_0-1,k_1)$.
3. $\neg\texttt{kkt}_4(k_0,k_1) \implies \texttt{kkt}_5(k_0,k_1)$*;*
4. $\neg\texttt{kkt}_5(k_0,k_1) \iff \texttt{kkt}_4(k_0,k_1+1)$.

Proof The proofs follow from direct computation.



(a) Depiction of relationship between kkt$_2$ and kkt$_3$ in Claim B.2 for given k_1. Green: indicates a condition holds. Red: indicates a condition does not hold. Arrows represent implication.

(b) Depiction of Claim B.3 for given k_1: local information (shaded) becomes global (outlined). Green outlines denote $K_0^2(k_1)$ and $K_0^3(k_1)$; red outlines denote $K_0^{\neg 2}(k_1)$ and $K_0^{\neg 3}(k_1)$.

Fig. 5 Linking conditions

1. Consider the contrapositive: $\neg\texttt{kkt}_3(k_0, k_1) \implies \texttt{kkt}_2(k_0, k_1)$. Since $\neg\texttt{kkt}_3(k_0, k_1) \iff (\theta+\lambda)(k_0, k_1) < x^0_{k_0+1}$, and since x^0 is sorted, because $x^0_{k_0+1} \le x^0_{k_0}$, it follows that $(\theta+\lambda)(k_0, k_1) < x^0_{k_0}$.
2. Since $\rho(k_0, k_1) > 0$ for all valid k_0 and k_1, it holds that

$$\begin{aligned}
&\neg\texttt{kkt}_3(k_0-1, k_1)\\
&\iff (\theta+\lambda)(k_0-1, k_1) < x^0_{(k_0-1)+1} = x^0_{k_0}\\
&\iff k\sum_{i=(k_0-1)+1}^{k_1} x^0_i + (k_1-k)\Big(\sum_{i=1}^{k_0-1} x^0_i - r\Big) < (\rho+2k-k_1)\cdot x^0_{k_0}\\
&\iff \rho(k_0, k_1)\cdot(\theta+\lambda)(k_0, k_1) + (2k-k_1)\cdot x^0_{k_0} < (\rho+2k-k_1)\cdot x^0_{k_0}\\
&\iff (\theta+\lambda)(k_0, k_1) < x^0_{k_0} \iff \texttt{kkt}_2(k_0, k_1).
\end{aligned}$$

3. Consider the contrapositive: $\neg\texttt{kkt}_5(k_0, k_1) \implies \texttt{kkt}_4(k_0, k_1)$. Since $\neg\texttt{kkt}_5(k_0, k_1) \iff \theta(k_0, k_1) \le x^0_{k_1+1}$, and because $x^0_{k_1+1} \le x^0_{k_1}$, it follows that $\theta(k_0, k_1) \le x^0_{k_1}$.
4. Since $\rho(k_0, k_1) > 0$ for all valid k_0 and k_1, it holds that

$$\begin{aligned}
&\texttt{kkt}_4(k_0, k_1+1)\\
&\iff x^0_{k_1+1} - \theta(k_0, k_1+1) \ge 0\\
&\iff (\rho(k_0, k_1)+k_0)x^0_{k_1+1} \ge k_0\sum_{i=k_0+1}^{k_1} x^0_i + k_0 x^0_{k_1+1} - (k-k_0)\Big(\sum_{i=1}^{k_0} x^0_i - r\Big)\\
&\iff x^0_{k_1+1} \ge \theta(k_0, k_1) \iff \neg\texttt{kkt}_5(k_0, k_1).
\end{aligned}$$

Thus the claims are proved. □

Claim B.3 (Linking $\texttt{kkt}_2(k_0)$ & $\texttt{kkt}_2(k_0 + j)$ and $\texttt{kkt}_3(k_0)$ & $\texttt{kkt}_3(k_0 + j)$) *Fix $k_1 \in \{k, \ldots, n\}$. The following are true.*

1. *Let $k_0^- \in \{0, \ldots, k-1\}$ be such that $\neg\texttt{kkt}_2(k_0^-, k_1)$. Then $\neg\texttt{kkt}_2(k_0', k_1)$ for any $k_0^- \leq k_0'$.*
2. *Let $k_0^+ \in \{0, \ldots, k-1\}$ be such that $\texttt{kkt}_2(k_0^+, k_1)$. Then $\texttt{kkt}_2(k_0', k_1)$ for any $k_0' \leq k_0^+$.*
3. *Let $k_0^- \in \{0, \ldots, k-1\}$ be such that $\neg\texttt{kkt}_3(k_0^-, k_1)$. Then $\neg\texttt{kkt}_3(k_0', k_1)$ for any $k_0' \leq k_0^-$.*
4. *Let $k_0^+ \in \{0, \ldots, k-1\}$ be such that $\texttt{kkt}_3(k_0^+, k_1)$. Then $\texttt{kkt}_3(k_0', k_1)$ for any $k_0^+ \leq k_0'$.*

This means that for every k_1, the following sets are contiguous

$$
\begin{aligned}
K_0^2(k_1) &:= \{k_0 \in \{0, \ldots, k-1\} : \texttt{kkt}_2(k_0, k_1)\}\\
K_0^{\neg 2}(k_1) &:= \{k_0 \in \{0, \ldots, k-1\} : \neg\texttt{kkt}_2(k_0, k_1)\},\\
K_0^3(k_1) &:= \{k_0 \in \{0, \ldots, k-1\} : \texttt{kkt}_3(k_0, k_1)\}\\
K_0^{\neg 3}(k_1) &:= \{k_0 \in \{0, \ldots, k-1\} : \neg\texttt{kkt}_3(k_0, k_1)\}.
\end{aligned}
$$

Proof The proof is by direct computation. For fixed k_1, suppress the dependence on k_1 where clear, and define $\eta(k_0) := \rho(k_0) \cdot (\theta + \lambda)(k_0)$.

1. Let $k_0 = k_0^-$. By induction, it suffices to show the claim for $j = 1$, *i.e.,* that

$$(*) : x_{k_0} \leq (\theta + \lambda)(k_0) \implies (**) : x^0_{k_0+1} \leq (\theta + \lambda)(k_0 + 1).$$

But $(\theta + \lambda)(k_0 + 1) = \big(\eta(k_0) + (k_1 - 2k)x^0_{k_0+1}\big)/(\rho(k_0) + k_1 - 2k)$ so

$$
\begin{aligned}
(**) &\iff (\rho + k_1 - 2k) \cdot x^0_{k_0+1} \leq \eta(k_0) + (k_1 - 2k)x^0_{k_0+1}\\
&\iff x^0_{k_0+1} \leq (\theta + \lambda)(k_0).
\end{aligned}
$$

Therefore, since x^0 is sorted, it is clear that $(*) \implies (**)$.

2. Let $k_0 = k_0^+$. By induction, it suffices to show the claim for $j = 1$, *i.e.,* that

$$(*) : x^0_{k_0} > (\theta + \lambda)(k_0) \implies (**) : x^0_{k_0-1} > (\theta + \lambda)(k_0 - 1).$$

But $(\theta + \lambda)(k_0 - 1) = \big(\eta(k_0) + (2k - k_1)x^0_{k_0+1}\big)/(\rho(k_0) + 2k - k_1)$ so

$$
\begin{aligned}
(**) &\iff \big(\rho(k_0) + (2k - k_1)\big) \cdot x^0_{k_0-1} > \eta(k_0) + (2k - k_1)x^0_{k_0}\\
&\iff \rho(k_0) \cdot x^0_{k_0-1} > \eta(k_0) + (2k - k_1) \cdot \big(x^0_{k_0} - x^0_{k_0-1}\big).
\end{aligned}
$$

For contradiction, suppose that (∗∗) does not hold. Then dividing by $\rho(k_0) > 0$,

$$\begin{aligned}\neg(**) \iff& x^0_{k_0-1} \le (\theta+\lambda)(k_0)\frac{2k-k_1}{\rho(k_0)} \cdot (x^0_{k_0} - x^0_{k_0-1})\\ \overset{(a)}{\implies}& x^0_{k_0-1} < x^0_{k_0-1}\frac{2k-k_1}{\rho(k_0)} \cdot (x^0_{k_0} - x^0_{k_0-1})\\ \iff& 0 < \big(\rho(k_0)+2k-k_1\big)\cdot(x^0_{k_0} - x^0_{k_0-1}) \implies 0<0\end{aligned}$$

where (a) follows from (∗), and $x^0_{k_0} - x^0_{k_0-1} \le 0$ follows by since x^0 is sorted.

3. Let $k_0 = k_0^-$. By induction, it suffices to show the claim for $j = 1$, *i.e.,* that

$$(*) : (\theta+\lambda)(k_0) < x_{k_0+1} \implies (**) : (\theta+\lambda)(k_0-1) < x^0_{k_0}.$$

But $(\theta+\lambda)(k_0-1) = \big(\eta(k_0) + (2k-k_1)x^0_{k_0}\big)/(\rho(k_0)+2k-k_1)$ so

$$\begin{aligned}(**) \iff& \eta(k_0) + (2k-k_1)\cdot x^0_{k_0} < (\rho+2k-k_1)\cdot x^0_{k_0}\\ \iff& (\theta+\lambda)(k_0) < x^0_{k_0}.\end{aligned}$$

Therefore, since x^0 is sorted, it is clear that $(*) \implies (**)$.

4. Let $k_0 = k_0^+$. By induction, it suffices to show the claim for $j = 1$, *i.e.,* that

$$(*) : (\theta+\lambda)(k_0) \ge x_{k_0+1} \implies (**) : (\theta+\lambda)(k_0+1) \ge x^0_{k_0+2}.$$

But $(\theta+\lambda)(k_0+1) = \big(\eta(k_0) + (k_1-2k)x^0_{k_0+1}\big)/(\rho(k_0)+k_1-2k)$ so

$$\begin{aligned}(**) \iff& \eta(k_0) + (k_1-2k)x^0_{k_0+1} \ge (\rho+(k_1-2k))x^0_{k_0+2}\\ \overset{(a)}{\impliedby}& \rho(k_0)\cdot x^0_{k_0+1} \ge \rho(k_0)\cdot x^0_{k_0+2} + (k_1-2k)\cdot(x^0_{k_0+2} - x^0_{k_0+1})\\ \iff& (\rho(k_0)+k_1-2k)\cdot(x^0_{k_0+1} - x^0_{k_0+2}) \ge 0\end{aligned}$$

where (a) follows from (∗), and $x^0_{k_0+1} - x^0_{k_0+2} \ge 0$ follows since x^0 is sorted.

Thus the claims are proved. □

B.2.3 Difference conditions

Next we summarize some conditions based on the successive differences of θ, λ, and $\theta+\lambda$. To do so, it is useful to introduce the discrete difference operator Δ of a function $f(x, y)$ of two arguments, defined as $\Delta_x f(x, y){:=}f(x+1, y) - f(x)$, $\Delta_y f(x, y){:=}f(x, y+1) - f(x, y)$. We will be concerned with differences in both arguments of the index-pair (k_0, k_1).

Claim B.4 (Δ_{k_0} & $\neg\texttt{kkt}_2$ and Δ_{k_1} & $\neg\texttt{kkt}_5$) *For fixed $k_1 \in \{k, \ldots, n\}$, it holds that*

1. $\Delta_{k_0}\theta(k_0-1, k_1) \ge 0 \iff \neg\texttt{kkt}_2(k_0, k_1)$*;*

2. $\Delta_{k_0}\lambda(k_0-1,k_1)\le 0 \iff \neg\texttt{kkt}_2(k_0,k_1)$;
3. $\Delta_{k_1}\theta(k_0,k_1)\ge 0 \iff \neg\texttt{kkt}_5(k_0,k_1)$;
4. $\Delta_{k_1}\lambda(k_0,k_1)\ge 0 \iff \neg\texttt{kkt}_5(k_0,k_1)$;
5. $\Delta_{k_1}(\theta+\lambda)(k_0,k_1)\ge 0 \iff \neg\texttt{kkt}_5(k_0,k_1)$.

Proof For shorthand, let $\tau_0 := \sum_{i=1}^{k_0} x_i^0 - r$, $\tau_1 := \sum_{i=1}^{k_1} x_i^0 - r$, $\eta_\theta(k_0,k_1) := \rho(k_0,k_1)\cdot\theta(k_0,k_1)$, $\eta_\lambda(k_0,k_1) := \rho(k_0,k_1)\cdot\lambda(k_0,k_1)$, and $\eta_{(\theta+\lambda)}(k_0,k_1) := \rho(k_0,k_1)\cdot(\theta+\lambda)(k_0,k_1)$.

1. Compute $\Delta_{k_0}\theta(k_0-1,k_1) = \frac{(2k-k_1)\cdot\eta_\theta(k_0,k_1)-\rho(k_0,k_1)\cdot(kx_{k_0}^0-\tau_1)}{\rho(k_0,k_1)\cdot\left(\rho(k_0,k_1)+2k-k_1\right)}$. Then

$$
\begin{aligned}
&\Delta_{k_0}\theta(k_0-1,k_1)\ge 0\\
&\overset{(a)}{\iff} (2k-k_1)\cdot\eta_\theta(k_0,k_1)-\rho(k_0,k_1)\cdot(kx_{k_0}^0-\tau_1)\ge 0\\
&\iff \frac{(2k-k_1)\cdot\eta_\theta(k_0,k_1)+\rho(k_0,k_1)\cdot\tau_1}{\rho(k_0,k_1)\cdot k}\ge x_{k_0}^0\\
&\overset{(b)}{\iff} \frac{k\cdot\eta_\theta(k_0,k_1)}{k_0\cdot\rho(k_0,k_1)}-\frac{\eta_\theta(k_0,k_1)}{k\cdot k_0}+\frac{\tau_1}{k}\ge x_{k_0}^0\\
&\iff \frac{k\sum_{i=k_0+1}^{k_1}x_i^0-\frac{k\cdot(k-k_0)}{k_0}\tau_0}{\rho(k_0,k_1)}-\left(\frac{1}{k}\sum_{i=k_0+1}^{k_1}x_i^0-\frac{k-k_0}{k\cdot k_0}\tau_0\right)+\frac{\tau_1}{k}\ge x_{k_0}^0\\
&\overset{(c)}{\iff} \frac{k\sum_{i=k_0+1}^{k_1}x_i^0-\frac{k\cdot(k-k_0)}{k_0}\tau_0}{\rho(k_0,k_1)}+\frac{\tau_0}{k}+\frac{k-k_0}{k\cdot k_0}\tau_0\ge x_{k_0}^0\\
&\iff \frac{k\sum_{i=k_0+1}^{k_1}x_i^0}{\rho(k_0,k_1)}-\left(\frac{k\cdot(k-k_0)}{k_0\cdot\rho(k_0,k_1)}+\frac{1}{k}+\frac{k-k_0}{k\cdot k_0}\right)\cdot\tau_0\ge x_{k_0}^0\\
&\overset{(d)}{\iff} \frac{k\sum_{i=k_0+1}^{k_1}x_i^0}{\rho(k_0,k_1)}-\frac{k_1-k}{\rho(k_0,k_1)}\cdot\tau_0\ge x_{k_0}^0 \iff (\theta+\lambda)(k_0,k_1)\ge x_{k_0}^0\\
&\iff \neg\texttt{kkt}_2(k_0,k_1)
\end{aligned}
$$

   where (a) follows since $\rho(k_0,k_1)>0$ for all valid arguments, (b) follows from the partial fractions identity $(2k-k_1)/\rho(k_0,k_1) = \frac{k}{k_0\cdot\rho(k_0,k_1)}-\frac{1}{k\cdot k_0}$, (c) follows from $\frac{1}{k}\tau_1-\frac{1}{k}\sum_{i=k_0+1}^{k_1}x_i^0=\frac{1}{k}\tau_0$, and (d) follows from algebraic manipulation.
2. Compute

$$
\begin{aligned}
&\lambda(k_0-1,k_1)\\
&=\left((k-(k_0-1))\sum_{i=k_0}^{k_1}x_i^0+(k_1-(k_0-1))\left(\sum_{i=1}^{k_0-1}x_i^0-r\right)\right)/\rho(k_0-1,k_1)\\
&=\left(\eta_\lambda(k_0,k_1)+(k-k_1)\cdot x_{k_0}^0+\tau_1\right)/\left(\rho(k_0,k_1)+2k-k_1\right)
\end{aligned}
$$

and $\Delta_{k_0}\lambda(k_0-1,k_1) = \frac{(2k-k_1)\cdot\eta_\lambda(k_0,k_1)-\rho(k_0,k_1)\cdot\left((k-k_1)\cdot x^0_{k_0}+\tau_1\right)}{\rho(k_0,k_1)\cdot\left(\rho(k_0,k_1)+2k-k_1\right)}$. Then

$$\begin{aligned}
&\Delta_{k_0}\lambda(k_0-1,k_1)\le 0\\
&\overset{(a)}{\iff} (2k-k_1)\cdot\eta_\lambda(k_0,k_1)-\rho(k_0,k_1)\cdot\left((k-k_1)\cdot x^0_{k_0}+\tau_1\right)\le 0\\
&\iff \frac{(2k-k_1)\cdot\eta_\lambda(k_0,k_1)}{\rho(k_0,k_1)}-\tau_1\le (k-k_1)\cdot x^0_{k_0}\\
&\overset{(b)}{\iff} \frac{(2k-k_1)\cdot\eta_\lambda(k_0,k_1)}{\rho(k_0,k_1)\cdot(k-k_1)}-\frac{\rho(k_0,k_1)\cdot\tau_1}{\rho(k_0,k_1)\cdot(k-k_1)}\ge x^0_{k_0}\\
&\iff c_1\sum_{i=k_0+1}^{k_1}x^0_i+c_2\cdot\tau_0\ge x^0_{k_0}
\end{aligned}$$

where (a) holds because $\rho(k_0,k_1)>0$ for all valid arguments, (b) holds since $(k-k_1)<0$ for $k_1>k$ (and at $k_1=k$, it holds since $\rho(k_0,k)>k\cdot(k-k_0)$), and $c_1{:=}\frac{(2k-k_1)\cdot(k-k_0)-\rho(k_0,k_1)}{\rho(k_0,k_1)\cdot(k-k_1)}$ and $c_2{:=}\frac{(2k-k_1)\cdot(k_1-k_0)-\rho(k_0,k_1)}{\rho(k_0,k_1)\cdot(k-k_1)}$. Then after simplification, $c_1=k/\rho(k_0,k_1)$ and $c_2=(k_1-k)/\rho(k_0,k_1)$ so we identify $\Delta_{k_0}\lambda(k_0-1,k_1)\le 0 \iff (\theta+\lambda)(k_0,k_1)\le x^0_{k_0} \iff \neg\texttt{kkt}_2(k_0,k_1)$.

3. Compute $\Delta_{k_1}\theta(k_0,k_1)=\frac{\rho(k_0,k_1)\cdot k_0\cdot x^0_{k_1+1}-k_0\eta_\theta(k_0,k_1)}{\rho(k_0,k_1)\cdot(\rho(k_0,k_1)+k_0)}$. Then

$$\begin{aligned}
&\Delta_{k_1}\theta(k_0,k_1)\ge 0\\
&\overset{(a)}{\iff} \rho(k_0,k_1)\cdot k_0\cdot x^0_{k_1+1}\ge k_0\eta_\theta(k_0,k_1) \iff x^0_{k_1+1}\ge\theta(k_0,k_1)\\
&\iff \neg\texttt{kkt}_5(k_0,k_1).
\end{aligned}$$

4. Compute $\Delta_{k_1}\lambda(k_0,k_1)=\frac{\rho(k_0,k_1)\cdot\left((k-k_0)x^0_{k_1+1}+\tau_0\right)-k_0\eta_\lambda(k_0,k_1)}{\rho(k_0,k_1)\cdot\left(\rho(k_0,k_1)+k_0\right)}$. Then

$$\begin{aligned}
&\Delta_{k_1}\lambda(k_0,k_1)\ge 0\\
&\overset{(a)}{\iff} \rho(k_0,k_1)\cdot\left((k-k_0)x^0_{k_1+1}+\tau_0\right)-k_0\eta_\lambda(k_0,k_1)\ge 0\\
&\iff x^0_{k_1+1}\ge\frac{k_0}{k-k_0}\frac{\eta_\lambda(k_0,k_1)}{\rho(k_0,k_1)}-\frac{\tau_0}{k-k_0} \overset{(b)}{\iff} x^0_{k_1+1}\ge\theta(k_0,k_1)\\
&\iff \neg\texttt{kkt}_5(k_0,k_1)
\end{aligned}$$

where (a) holds because $\rho(k_0,k_1)>0$ for all valid arguments and (b) holds because $\frac{k_0\cdot(k_1-k_0)}{\rho(k_0,k_1)\cdot(k-k_0)}-\frac{1}{k-k_0}=\frac{k_0-k}{\rho(k_0,k_1)}$ by algebraic manipulation.

5. Consider the equivalent form of the original statement given by $\Delta_{k_1}(\theta+\lambda)(k_0,k_1)<0 \iff \texttt{kkt}_5(k_0,k_1)$. Compute $\Delta_{k_1}(\theta+\lambda)(k_0,k_1){:=}(\theta+\lambda)(k_0,k_1+1)-(\theta+\lambda)(k_0,k_1)$ giving

$$\Delta_{k_1}(\theta+\lambda)(k_0,k_1)=\frac{\rho(k_0,k_1)\left(kx^0_{k_1+1}+\tau_0\right)-k_0\left(k\sum_{i=k_0+1}^{k_1}x^0_i+(k_1-k)\tau_0\right)}{\rho(k_0,k_1)\cdot\left(\rho(k_0,k_1)+k_0\right)}.$$

Then

$$\begin{aligned}
&\Delta_{k_1}(\theta+\lambda)(k_0,k_1)<0\\
&\overset{(a)}{\iff} \rho(k_0,k_1)\cdot\left(k\cdot x^0_{k_1+1}+\tau_0\right)-k_0\Big(k\sum_{i=k_0+1}^{k_1}x^0_i+(k_1-k)\tau_0\Big)<0\\
&\iff \rho(k_0,k_1)\cdot kx^0_{k_1+1}<k_0\Big(k\sum_{i=k_0+1}^{k_1}x^0_i+(k_1-k)\cdot\tau_0\Big)-\rho(k_0,k_1)\cdot\tau_0\\
&\iff \rho(k_0,k_1)\cdot kx^0_{k_1+1}<k_0k\sum_{i=k_0+1}^{k_1}x^0_i+\left(k_0(k_1-k)-\rho(k_0,k_1)\right)\cdot\tau_0\\
&\overset{(b)}{\iff} \rho(k_0,k_1)\cdot kx^0_{k_1+1}<k_0k\sum_{i=k_0+1}^{k_1}x^0_i+k\cdot(k_0-k)\cdot\tau_0\\
&\iff \rho(k_0,k_1)\cdot x^0_{k_1+1}<k_0\sum_{i=k_0+1}^{k_1}x^0_i+\cdot(k_0-k)\cdot\tau_0\\
&\iff x^0_{k_1+1}<\frac{k_0\sum_{i=k_0+1}^{k_1}x^0_i+\cdot(k_0-k)\cdot\tau_0}{\rho(k_0,k_1)}=\theta(k_0,k_1)\\
&\iff \mathrm{kkt}_5(k_0,k_1)>0.
\end{aligned}$$

where (a) follows because $\rho>0$ for all valid arguments and (b) follows from the identity $k_0(k_1-k)-\rho(k_0,k_1)=k(k_0-k)$.

Thus the claims are proved. □

C Algorithmic detail

C.1 PLCP

We recall some background for processing a PLCP(λ; q, d, M) where $q\geq 0_{n-1}$ and M is a symmetric, positive definite Z-matrix before specializing to the sorted projection problem (7). The approach largely follows the *symmetric parametric principal pivoting method* outlined in [12, Algorithm 4.5.2], but instead of computing the upper bound for λ ahead of time, we check the whether or not the existing solution satisfies the top-k-sum budget constraint at each iteration. We begin by noting that any basis $\xi\subseteq\{1,\dots,n-1\}$ partitions the affine relationship into the following system

$$\begin{bmatrix}w_\xi(\lambda)\\ w_{\xi^c}(\lambda)\end{bmatrix}=\begin{bmatrix}M_{\xi\xi} & M_{\xi\xi^c}\\ M_{\xi^c\xi} & M_{\xi^c\xi^c}\end{bmatrix}\begin{bmatrix}z_\xi(\lambda)\\ z_{\xi^c}(\lambda)\end{bmatrix}+\begin{bmatrix}q_\xi+\lambda d_\xi\\ q_{\xi^c}+\lambda d_{\xi^c}\end{bmatrix}, \tag{24}$$

where the notation $w(\lambda)$ and $z(\lambda)$ is used to emphasize the dependence on parameter λ, and where the linear system always has a unique solution for every ξ since M has positive principal minors. The PLCP solves the projection problem (7) by identifying an optimal basis $\bar{\xi}$ and parameter $\bar{\lambda} \geq 0$ such that the solution $w(\bar{\lambda})$ and $z(\bar{\lambda})$ satisfy:

- subproblem optimality for LCP($q + \bar{\lambda}d, M$): this consists of (i) complementarity: $w_{\bar{\xi}}(\bar{\lambda}) = 0$ and $z_{\bar{\xi}^c}(\bar{\lambda}) = 0$; and (ii) feasibility: $z_{\bar{\xi}}(\bar{\lambda}) \geq 0$, $w_{\bar{\xi}^c}(\bar{\lambda}) \geq 0$, and $w(\bar{\lambda}) = Mz(\bar{\lambda}) + q + \bar{\lambda}d$;
- outer problem optimality: the primal solution $\bar{x}(\bar{\lambda}) = \bar{x}^0 - \bar{\lambda}\mathbb{1}_k + D^\top \bar{z}$, which is an implicit function of λ, satisfies $\mathbb{1}_k^\top x(\bar{\lambda}) = r$, assuming that $\mathbb{1}_k^\top \bar{x}^0 > r$.

Under the present setting, the parametric LCP procedure begins by solving the trivial LCP(q, M) associated with $\lambda = 0$ by taking $\xi = \emptyset$, $z \equiv 0$, and $w = q \geq 0$. This solution, denoted $\big(z(0), w(0)\big)$ may not satisfy the budget constraint, in which case λ needs to be increased. The remaining steps utilize the fact that the solution map $z(\lambda)$ associated with the LCP subproblem at value λ is a piecewise-linear and monotone nondecreasing function in λ. Therefore, as λ increases, the procedure only "adds" nonnegative components to $z(\lambda)$. It stops once a large enough $\lambda \geq 0$ has been identified so that the budget constraint is satisfied.

The mechanics of the PLCP specialized to our problem are as follows. To solve any LCP subproblem associated with λ, we seek identify a complementary, feasible basis ξ (depending on λ) of dimension m that gives rise to the solution map

$$z_\xi(\lambda) = M_{\xi\xi}^{-1}(-q_\xi - \lambda d_\xi) = z_\xi(0) - \lambda M_{\xi\xi}^{-1} d_\xi \tag{25.1}$$

$$w_{\xi^c}(\lambda) = M_{\xi^c\xi} z_\xi(\lambda) + q_{\xi^c} + \lambda d_{\xi^c} \tag{25.2}$$

via the linear system (24) where $M_{\xi\xi}^{-1} := (M_{\xi\xi})^{-1}$.

We will show three things: (i) beginning from $\xi^1 = \{k\}$ in iteration 1, ξ^t remains contiguous for all subsequent iterations $t \geq 1$, which leads to a simple form of the minimum ratio test for identifying the breakpoint λ^{t+1} and indicates that a basis ξ^t is subproblem-optimal for $\lambda \in [\lambda^t, \lambda^{t+1}]$; (ii) checking whether or not λ^{t+1} is "large enough" simplifies, *i.e.,* that there exists $\bar{\lambda} \leq \lambda^{t+1}$ such that the primal solution $x(\bar{\lambda})$ satisfies the budget constraint; and (iii) updating the solution map $z(\lambda^{t+1})$ associated with the new breakpoint λ^{t+1} from the previous solution $z(\lambda^t)$ simplifies. In the below subsections, we drop the dependence on t and use "+" to denote the next value when clear. The simplified expressions for each step involve the observation that $M_{\xi\xi}^{-1}$ has an explicit form given by

$$(M_{\xi\xi})_{ij}^{-1} = \frac{(|\xi| + 1 - \max(i, j)) \cdot \min(i, j)}{|\xi| + 1}. \tag{26}$$

Identifying the next breakpoint

Suppose that $w(\lambda) \geq 0$ and $z(\lambda) \geq 0$ are optimal for the subproblem with parameter λ and contiguous basis ξ (*i.e.,* $\xi = \{a, a+1, \ldots, b-1, b\}$ for $n - 1 \geq b \geq a \geq 1$) that contains k with $|\xi| = m$. Inspecting (25.1), notice that $z_\xi(\lambda) = (\leq 0)_\xi + \lambda M_{\xi\xi}^{-1} e_\xi^k$ because $q \geq 0$, $d = -e^k$, and because of the fact that M is a Z-matrix implies that $M_{ij}^{-1} \geq 0$. Since $M_{\xi\xi}$ also is a Z-matrix, it holds that $M_{\xi\xi}^{-1} \geq 0$ and thus that

$z_\xi(\lambda') \geq z_\xi(\lambda)$ for $\lambda' \geq \lambda$. On the other hand, suppose that the current solution does not satisfy the budget constraint, *i.e.,* λ is not large enough. Because of the form of $M = DD^\top$, $M_{\xi^c\xi}$ is the matrix of zeros except for at most two negative elements in different columns. Explicitly, the two elements of $M_{\xi^c\xi} z_\xi(\lambda)$ are: $-z_{\xi_1}(\lambda)$ in index $\xi_1 - 1$ and $-z_{\xi_m}(\lambda)$ in index $\xi_m + 1$. Since $d = -e^k$, we may neglect the term $d_{\xi^c} = 0$, so there are only two possible indices where $w_{\xi^c}(\lambda)$ may fail to be nonnegative: $a - 1$ where $a{:=}\xi_1$, and $b + 1$ where $b{:=}\xi_m$.[3] As a result, the "minimum ratio test" only requires two comparisons per pivot where the smallest parameter such that $w_{i^*}(\lambda) = 0$ for some $i^* \in \xi^c$ is given by

$$\begin{aligned}
\lambda^+ &{:=} \min_{\lambda' \geq \lambda} \left\{ \lambda' : M_{\xi^c\xi} z_\xi(\lambda') + q_{\xi^c} + \lambda d_{\xi^c} = 0 \right\} \\
&\overset{(*)}{=} \min_{\lambda' \geq \lambda} \left\{ \lambda' : 0 = -z_a(\lambda) + q_{a-1},\ 0 = -z_b(\lambda) + q_{b+1} \right\} \\
&= \min_{\lambda' \geq \lambda} \left\{ \lambda' : 0 = -z_a(0) - \lambda' (M_{\xi\xi}^{-1} e_\xi^k)_{a^\xi} + q_{a-1} \right., \\
&\qquad\qquad \left. 0 = -z_b(0) - \lambda' (M_{\xi\xi}^{-1} e_\xi^k)_{b^\xi} + q_{b+1} \right\} \\
&= \min\left\{ \big(q_{a-1} - z_a(0)\big) / (M_{\xi\xi}^{-1} e_\xi^k)_1 \ ,\ \big(q_{b+1} - z_b(0)\big) / (M_{\xi\xi}^{-1} e_\xi^k)_m \right\} \\
&= \min\left\{ \lambda^a, \lambda^b \right\}
\end{aligned} \tag{27}$$

where $(*)$ follows (after the first iteration) because of the form of d. The constant cost of determining λ^+ is clear from the explicit expression for $(M_{\xi\xi}^{-1})_{ij}$ via (26). If $\lambda^+ = \lambda^a$, then we define $s{:=}a - 1$ and update the basis $\xi^+ = (s, \xi)$; otherwise we define $s{:=}b + 1$ and set $\xi^+ = (\xi, s)$. Therefore, the next basis ξ^+ is contiguous and contains k. Next we must check whether $\lambda^+ \geq \bar{\lambda}$, *i.e.,* whether the current basis contains a solution that satisfies the budget constraint for some parameter in the range $[\lambda, \lambda^+]$.

Checking optimality

The procedure terminates based on the observation that for the next breakpoint λ^+ (as determined above with current basis ξ), if $\mathsf{T}_{(k)}(x(\lambda^+)) < r$ with current basis ξ, then there must exist a $\bar{\lambda} < \lambda^+$ that solves $\mathsf{T}_{(k)}(x(\bar{\lambda})) = r$ for basis ξ. From the primal solution map $x(\lambda) = \overleftarrow{x}^0 - \lambda \mathbb{1}_k + D^\top z(\lambda)$, which is derived from stationarity of the Lagrangian, evaluation of the top-k-sum simplifies to $\mathsf{T}_{(k)}(x(\lambda)) = \mathbb{1}_k^\top x(\lambda) = \sum_{i=1}^k \overleftarrow{x}_i^0 - k\lambda + z_k(\lambda)$ where

$$z_k(\lambda) = -(M_{\xi\xi}^{-1} q_\xi)_{k^\xi} + \lambda \cdot (M_{\xi\xi}^{-1} e_\xi^k)_{k^\xi} = z_k(0) + \lambda \cdot (M_{\xi\xi}^{-1} e_\xi^k)_{k^\xi}.$$

If $\mathsf{T}_{(k)}(x(\lambda^+)) < r$, then $\bar{\lambda}$ satisfies

$$\bar{\lambda} = \Big(\sum_{i=1}^k \overleftarrow{x}_i^0 - r + z_k(0) \Big) / \big(k - (M_{\xi\xi}^{-1})_{k^\xi k^\xi} \big), \tag{28}$$

[3] Note that a parametric pivoting method with similar direction vector d was studied in [31].

which can be done in constant time due to (26). Finally, we can reconstruct $\bar{x}(\bar{\lambda}) = \overleftarrow{x}^0 - \bar{\lambda}\mathbb{1}_k + D^\top z(\bar{\lambda})$. Since D has only two elements per row, and by observing that $z_{\xi^c}(\bar{\lambda}) = 0$, the matrix-vector multiplication can be performed in $O(|\xi|)$ time. Otherwise, it remains to update the solution maps $z_{\xi^+}(\lambda^+)$ and $w_{\xi^+}(\lambda^+)$ and then return to the breakpoint identification step.

Updating the subproblem solution

Thus far, excluding the recovery of a primal optimal solution, our procedure has required computations involving only a very particular subset of $z_\xi(0)$, namely $z_a(0)$, $z_k(0)$, and $z_b(0)$. This observation allows for performing a constant number of updates per iteration. Since $\xi^+ \setminus \xi = \{s\}$ changes by one element per iteration and ξ^+ remains contiguous, the Schur complement rule can be used to update the three elements of $z_{\xi^+}(0)$ in constant time, which in turn provides the new solution via $z_{\xi^+}(\lambda^+) = z_{\xi^+}(0) + \lambda M^{-1}_{\xi^+\xi^+} e^k$, where the latter term can be computed in constant time from the form of M^{-1} given by (26).

Accordingly, the goal of this section is to compute $z_{\xi^+_{a^+}}(0)$, $z_{\xi^+_{k^+}}(0)$, and $z_{\xi^+_{b^+}}(0)$ for basis ξ^+ at (new) locations a^+, k^+, and b^+ in ξ^+ from an existing solution $z_{\xi a}(0)$, $z_{\xi_k}(0)$, and $z_{\xi_b}(0)$ with basis ξ. There are two cases, corresponding to $\xi^+ = \xi \cup \{s\}$ with

1. $s = a - 1$. Then $\xi^+ = (a-1, \xi)$ so that $a^+ = a - 1$, and

$$z_{\xi^+}(0) = \begin{bmatrix} z_s(0) \\ z_\xi(0) \end{bmatrix} = -M^{-1}_{\xi^+\xi^+} q_{\xi^+} = \begin{bmatrix} 2 & (-1, 0, \dots, 0) \\ (-1, 0, \dots, 0)^\top & M_{\xi\xi} \end{bmatrix}^{-1} \begin{bmatrix} -q_s \\ -q_\xi \end{bmatrix}$$
$$= \begin{bmatrix} \frac{1}{\sigma}(z_a(0) - q_{a-1} \\ z_\xi(0) + \frac{1}{\sigma}(M^{-1}_{\xi\xi})_{\xi,1} \cdot (z_a(0) - q_{a-1}) \end{bmatrix},$$

where $t_a = (-1, 0, \dots, 0)^\top$ and $\sigma = 2 - t_a^\top M^{-1}_{\xi\xi} t_a = 2 - (M^{-1}_{\xi\xi})_{1,1} = (m + 2)/(m+1)$. Thus

$$a^+ = a - 1, \quad {}_{a^+}\xi^+ = 1, \quad {}_{k^+}\xi^+ = {}_k\xi + 1, \quad {}_{b^+}\xi^+ = {}_b\xi + 1$$
$$z_{a^+}(0) = \tfrac{1}{\sigma}(z_a(0) - q_{a-1}(0))$$
$$z_{k^+}(0) = z_k(0) + (M^{-1}_{\xi\xi})_{{}_k\xi,1} \cdot \tfrac{1}{\sigma}(z_a(0) - q_{a-1}(0)) = z_k(0) + (M^{-1}_{\xi\xi})_{{}_k\xi,1} \cdot z_{a^+}(0)$$
$$z_{b^+}(0) = z_b(0) + (M^{-1}_{\xi\xi})_{m,1} \cdot \tfrac{1}{\sigma}(z_a(0) - q_{a-1}(0)) = z_b(0) + (M^{-1}_{\xi\xi})_{\xi_b,\xi_a} \cdot z_{a^+}(0).$$

2. $s = b + 1$. Then $\xi^+ = (\xi, b+1)$ so that $b^+ = b + 1$, and

$$z_{\xi^+}(0) = \begin{bmatrix} z_\xi(0) \\ z_s(0) \end{bmatrix} = -M^{-1}_{\xi^+\xi^+} q_{\xi^+} = \begin{bmatrix} M_{\xi\xi} & (0, \dots, 0, -1)^\top \\ (0, \dots, 0, -1) & 2 \end{bmatrix}^{-1} \begin{bmatrix} -q_\xi \\ -q_s \end{bmatrix}$$
$$= \begin{bmatrix} z_\xi(0) + \frac{1}{\sigma}(M^{-1}_{\xi\xi})_{\xi,m} \cdot (z_b(0) - q_{b+1}) \\ \frac{1}{\sigma}(z_b(0) - q_{b+1}) \end{bmatrix},$$

where $t_b = (0, \ldots, 0, -1)^\top$ and $\sigma = 2 - t_b^\top (M_{\xi\xi}^{-1}) t_b = 2 - (M_{\xi\xi}^{-1})_{m,m} = (m+2)/(m+1)$. Thus

$$\begin{aligned}
&b^+ = b + 1, \quad {}_{a^+}\xi^+ = 1, \quad {}_{k^+}\xi^+ = {}_k\xi, \quad {}_{b^+}\xi^+ = {}_b\xi + 1 \\
&z_{a^+}(0) = z_a(0) + (M_{\xi\xi}^{-1})_{1,m} \cdot \tfrac{1}{\sigma}(z_b(0) - q_{b+1}) = z_a(0) + (M_{\xi\xi}^{-1})_{1,m} \cdot z_{b^+}(0) \\
&z_{k^+}(0) = z_k(0) + (M_{\xi\xi}^{-1})_{{}_k\xi,m} \cdot \tfrac{1}{\sigma}(z_b(0) - q_{b+1}) = z_k(0) + (M_{\xi\xi}^{-1})_{{}_k\xi,m} \cdot z_{b^+}(0) \\
&z_{b^+}(0) = \tfrac{1}{\sigma} \cdot (z_b(0) - q_{b+1}).
\end{aligned}$$

The constant cost of the solution update procedure is clear due to the explicit formula for $(M_{\xi\xi}^{-1})_{ij}$.

Recovering k_0 and k_1

Given optimal indices $\bar{a}$ and $\bar{b}$ and a solution $\bar{x}$ produced by PLCP, the sorting-indices k_0 and k_1 can be recovered without inspecting $\bar{x}$ by setting $(\bar{k}_0, \bar{k}_1) = (k-1, k)$ if the problem is solved immediately; otherwise $(\bar{k}_0, \bar{k}_1) = \left(\max\{\bar{a}-1, 0\}, \min\{\bar{b}+1, n\}\right)$.

C.2 ESGS

We now provide additional details justifying the form of (i) the candidate solution for a given candidate index-pair (k_0', k_1') in (19); and (ii) the form of the KKT conditions (20) that are used in the analysis of the ESGS algorithm.

Candidate solution

We now provide an argument justifying the construction of the linear system in θ', λ' and candidate solution $x'(k_0', k_1')$ from (19). The linear equations in λ' and θ' are recovered by summing various components of the KKT conditions

$$k_0'\lambda' = \sum_{i=1}^{k_0'} x_i^0 - r + \theta'(k - k_0'), \quad \text{and} \quad (k_1' - k_0')\theta' = \sum_{i=k_0'+1}^{k_1'} x_i^0 - \lambda'(k - k_0'). \tag{29}$$

The equation for λ' is recovered by summing the stationarity conditions corresponding to indices in α' and eliminating $\mathbb{1}_{\alpha'}^\top x_{\alpha'}'$ using the constraint. That is, $\sum_{i=1}^{k_0'} x_i' = \sum_{i=1}^{k_0'} (\overleftarrow{x}_i^0 - \lambda')$ and $r = \sum_{i=1}^{k_0'} x_i' + (k - k_0')\theta'$ imply $\sum_{i=1}^{k_0'} \overleftarrow{x}_i^0 - k_0'\lambda' = r - (k - k_0')\theta'$. On the other hand, equation θ' is recovered by summing the stationarity conditions corresponding to indices in β' and using $\mathbb{1}_{\beta'}^\top \mu' = (k - k_0)$. That is, $\sum_{i=k_0'+1}^{k_1'} x_i' = \sum_{i=k_0'+1}^{k_1'} \overleftarrow{x}_i^0 - (k - k_0')\lambda'$ and $\sum_{i=k_0'+1}^{k_1'} x_i' = (k_1' - k_0')\theta'$. The linear system $A \cdot (\lambda, \theta)^\top = b$ has explicit solution

$$A := \begin{bmatrix} k_0 & -(k-k_0) \\ k-k_0 & k_1-k_0 \end{bmatrix}, \quad b := \begin{bmatrix} \sum_{i=1}^{k_0} \overleftarrow{x}_i^0 - r \\ \sum_{i=k_0+1}^{k_1} \overleftarrow{x}_i^0 \end{bmatrix}, \quad A^{-1} = \frac{1}{\rho} \begin{bmatrix} k_1-k_0 & k-k_0 \\ -(k-k_0) & k_0 \end{bmatrix},$$

where $\rho := \det[A] = k_0(k_1-k_0) + (k-k_0)^2$.

KKT conditions

Next we justify the reduction of the KKT conditions from (18) to the five conditions listed in (20). The KKT conditions (18) are equivalent to (20) because of the following argument. Inspecting condition $\bar{x}_{\bar{k}_0} > \bar{\theta}$ at index k_0' and using $\bar{x}_{\bar{\beta}} = \bar{\theta}_{\bar{\beta}}$ to obtain $x'_{k_0'+1} = \theta'$, it holds that

$$\begin{cases} x'_{k_0'} > \theta' \iff \overleftarrow{x}^0_{k_0'} - \lambda' > \theta' \iff \overleftarrow{x}^0_{k_0'} > \theta' + \lambda' \\ \theta' + \lambda' = x'_{k_0'+1} + \lambda' = \overleftarrow{x}^0_{k_0'+1} - \lambda' \mu'_{k_0'+1} + \lambda' \overset{(*)}{\geq} \overleftarrow{x}^0_{k_0'+1} \end{cases}$$
$$\iff \overleftarrow{x}^0_{k_0'} > \theta' + \lambda' \geq \overleftarrow{x}^0_{k_0'+1},$$

where $(*)$ holds since $\mu'_{k_0'+1} \in [0,1]$ and $\lambda' > 0$. Similarly, inspecting condition $\bar{\theta} > \bar{x}_{\bar{k}_1+1}$ at index k_1', and using the form of $\bar{\mu}$ and $\bar{x}_{\bar{\beta}} = \bar{\theta} \mathbb{1}_{\bar{\beta}}$ to obtain $x'_{k_1'} = \theta'$, it holds that

$$\begin{cases} \theta' > x'_{k_1'+1} = \overleftarrow{x}^0_{k_1'+1} \\ \theta' = x'_{k_1'} = \overleftarrow{x}^0_{k_1'} - \mu'_{k_1'+1} \lambda' \overset{(**)}{\leq} x^0_{k_1'} \end{cases} \iff \overleftarrow{x}^0_{k_1'} \geq \theta' > \overleftarrow{x}^0_{k_1'+1},$$

where $(**)$ holds since $\mu'_{k_1'} \in [0,1]$ and $\lambda' > 0$.

Acknowledgements The authors wish to thank the associate editor and anonymous reviewers, whose insightful comments and careful reading helped improve the presentation and experiments, as well as Eric Sager Luxenberg for discussion related to Proposition 3. After posting the first version of this manuscript online in October 2023, the authors became aware, through private communication, of independent work by Eric Sager Luxenberg on an approach to solving problem (1) similar to PLCP.

Funding The second author is partially supported by the National Science Foundation under Grants CCF-2416172 and DMS-2416250, and the National Institutes of Health under grant R01CA287413-01.

Code and data availability The full code and data was made available for review. Reference [35] provides a link to the publicly available source code, experiments, and data.

Declarations

Competing interests The authors have no relevant financial or non-financial interests to disclose.

References

1. Alexandrescu, A.: Fast deterministic selection. In: Iliopoulos, C.S., Pissis, S.P., Puglisi, S.J., Raman, R. (eds.) 16th International Symposium on Experimental Algorithms (SEA 2017), volume 75 of Leibniz International Proceedings in Informatics (LIPIcs), pp. 24:1–24:19, Dagstuhl, Germany, 2017. Schloss Dagstuhl – Leibniz-Zentrum für Informatik
2. Bach, F.: Learning with submodular functions: a convex optimization perspective. Found. Trends Mach. Learn. **6**(2–3), 145–373 (2013)
3. Barlow, R.E., Brunk, H.D.: The isotonic regression problem and its dual. J. Am. Stat. Assoc. **67**(337), 140–147 (1972)
4. Best, M.J., Chakravarti, N.: Active set algorithms for isotonic regression; a unifying framework. Math. Prog. **47**(1), 425–439 (1990)
5. Bezanson, J., Edelman, A., Karpinski, S., Shah, V.B.: Julia: a fresh approach to numerical computing. SIAM Rev. **59**(1), 65–98 (2017)
6. Blum, M., Floyd, R.W., Pratt, V., Rivest, R.L., Tarjan, R.E.: Time bounds for selection. J. Comput. Syst. Sci. **7**, 448–461 (1973)
7. Chandrasekaran, R.: A special case of the complementary pivot problem. Opsearch **7**, 263–268 (1970)
8. Chaudhuri, A., Kramer, B., Norton, M., Royset, J.O., Willcox, K.: Certifiable risk-based engineering design optimization. Am. Inst. Aeronaut. Astronaut. J. **60**(2), 551–565 (2022)
9. Chen, W., Sim, M., Sun, J., Teo, C.-P.: From CVaR to uncertainty set: Implications in joint chance-constrained optimization. Oper. Res. **58**(2), 470–485 (2010)
10. Condat, L.: Fast projection onto the simplex and ℓ_1 ball. Math. Program. Ser. A **158**, 575–585 (2016)
11. Cottle, R.W.: Monotone solutions of the parametric linear complementarity problem. Math. Prog. **3**(1), 210–224 (1972)
12. Cottle, R.W., Pang, J.-S., Stone, R.E.: The Linear Complementarity Problem. SIAM, Philadelphia (2009)
13. Cucala, D.T., Grau, B.C., Motik, B., Kostylev, E.V.: On the correspondence between monotonic max-sum GNNs and datalog (2023). arXiv:2305.18015
14. Dahlgren, R., Liu, C.-C., Lawarrée, J.: Risk assessment in energy trading. IEEE Trans. Power Syst. **18**(2), 503–511 (2003)
15. Davis., D.: An $O(n \log(n))$ algorithm for projecting onto the ordered weighted ℓ_1 norm ball (2015). arXiv:1505.00870
16. Dolatabadi, A., Mohammadi-Ivatloo, B.: Stochastic risk-constrained scheduling of smart energy hub in the presence of wind power and demand response. Appl. Therm. Eng. **123**, 40–49 (2017)
17. Eigen, H., Sadovnik, A.: TopKConv: Increased adversarial robustness through deeper interpretability. In: 20th IEEE International Conference on Machine Learning and Applications (ICMLA), pp. 15–22 (2021)
18. Figueiredo, M.A.T., Zeng, X.: The ordered weighted ℓ_1 norm: atomic formulation, projections, and algorithms (2014). arXiv:1409.4271
19. Fröhlich, C., Williamson, R.C.: Risk measures and upper probabilities: coherence and stratification (2022). arXiv:2206.03183
20. Güler, O.: Generalized linear complementarity problems. Math. Oper. Res. **20**(2), 441–448 (1995)
21. Hoare, C.A.R.: Algorithm 65: find. Commun. ACM **4**, 321–322 (1961)
22. Hu, S., Wang, X., Lyu, S.: Rank-based decomposable losses in machine learning: a survey. IEEE Trans. Pattern Anal. Mach. Intell. PP (2023)
23. Isac, G., Németh, A.B.: Monotonicity of metric projections onto positive cones of ordered Euclidean spaces. Arch. Math. **46**(6), 568–576 (1986)
24. Jabr, R.A.: Distributionally robust CVaR constraints for power flow optimization. IEEE Trans. Power Syst. **35**(5), 3764–3773 (2020)
25. Laguel, Y., Pillutla, K., Malick, J., Harchaoui, Z.: Superquantiles at work: machine learning applications and efficient subgradient computation. Set-Valued Var. Anal. **29**, 967–996 (2021)
26. Leqi, L., Prasad, A., Ravikumar, P.K.: On human-aligned risk minimization. Adv. Neural Inf. Process. Syst. **32**, 15029–15038 (2019)
27. Li, Q., Li, X.: Fast projection onto the ordered weighted ℓ_1 norm ball. Sci. China Math. **65**, 869–886 (2021)
28. Németh, A.B., Németh, S.Z.: Isotonicity of the projection onto the monotone cone (2012). arXiv:1201.4677

29. Nemirovski, A., Shapiro, A.: Convex approximations of chance constrained programs. SIAM J. Optim. **17**(4), 969–996 (2006)
30. Overton, M.L., Womersley, R.S.: Optimality conditions and duality theory for minimizing sums of the largest eigenvalues of symmetric matrices. Math. Prog. **62**(1), 321–357 (1993)
31. Pang, J.-S.: A parametric linear complementarity technique for optimal portfolio selection with a risk-free asset. Oper. Res. **28**(4), 927–941 (1980)
32. Peng, L., Travadi, Y., Zhang, R., Cui, Y., Sun, J.: Imbalanced classification in medical imaging via regrouping (2022). arXiv:2210.12234
33. Robey, A., Chamon, L., Pappas, G.J., Hassani, H.: Probabilistically robust learning: Balancing average and worst-case performance. In: Proceedings of the 39th International Conference on Machine Learning, vol. 162, pp. 18667–18686. PMLR (2022)
34. Rockafellar, R.T., Uryasev, S.: Optimization of conditional value-at-risk. J. Risk **2**(3), 21–41 (1999)
35. Roth, J., Cui, Y.: Top-k-sum.jl. https://doi.org/10.5281/zenodo.14189753
36. Roth, J., Cui, Y.: Fast computation of superquantile-constrained optimization through implicit scenario reduction (2023). arXiv:2405.07965
37. Royset, J.O.: Risk-adaptive approaches to learning and decision making: a survey (2022). arXiv:2212.00856
38. Tavakoli, M., Shokridehaki, F., Funsho Akorede, M., Marzband, M., Vechiu, I., Pouresmaeil, E.: CVaR-based energy management scheme for optimal resilience and operational cost in commercial building microgrids. Int. J. Electr. Power Energy Syst. **100**, 1–9 (2018)
39. Todd, M.J.: On max-k-sums. Math. Prog. **171**(1–2), 489–517 (2018)
40. Williams, J.W.J.: Algorithm 232: heapsort. Commun. ACM **7**, 347–348 (1964)
41. Williamson, R.C., Menon, A.: Fairness risk measures. In: Proceedings of the 36th International Conference on Machine Learning, vol. 97, pp. 6786–6797. PMLR (2019)
42. Wu, B., Ding, C., Sun, D., Toh, K.-C.: On the Moreau-Yosida regularization of the vector k-norm related functions. SIAM J. Optim. **24**(2), 766–794 (2014)
43. Yuan, P., You, X., Chen, H., Peng, Q., Zhao, Y., Xu, Z., Jing, X.-Y., He, Z.: Group sparse additive machine with average top-k loss. Neurocomputing **395**, 1–14 (2020)