A Fast MM Algorithm for Group Lasso

Abstract

The group lasso regression is a popular method for performing model selection on groups of predictors with a natural structure. When each group is orthogonalized, the optimization yields the widely-used block coordinate descent (BCD) algorithm. Despite its simplicity, the BCD algorithm converges slowly when the matrix of features is poorly conditioned or when the solution is not highly sparse. This article presents a novel iterative algorithm for the group lasso based on the majorize-minimize (MM) principle. Unlike previous algorithms, the proposed MM algorithm optimizes a reparameterized objective for the group lasso that maintains the same Karush-Kuhn-Tucker (KKT) conditions as the original problem. This change of variables enables an adaptive MM algorithm that, at each iteration, prioritizes the groups that are furthest from meeting the KKT conditions. As a result the convergence rate of the MM algorithm is accelerated with minimal additional cost. We establish the global convergence of the MM algorithm to the group-lasso KKT point. Using real data and simulations, we demonstrate numerically that the MM algorithm converges up to an order of magnitude faster than the BCD algorithm when computed over a regularization path.

Keywords: lasso penalty, global convergence, group selection, majorization-minimization (MM)

1 Introduction

The objective of group selection methods is to identify which groups of predictors are important for predicting the outcome variable and to estimate their corresponding regression coefficients. These methods are particularly useful in scenarios where the predictor variables naturally form meaningful groups or when prior knowledge indicates that certain groups of variables might be linked to the outcome variable. For example, in genomics, genes within the same pathway often exhibit similar functions, and incorporating group structures, as seen in gene set enrichment analysis (Subramanian et al., 2005), has proven effective for biomarker identification (Meier et al., 2008).

To formulate this problem, suppose we are given a dataset $(\boldsymbol{y}, \mathbf{X})$ consisting of a response vector $\boldsymbol{y} \in \mathbb{R}^n$ and a feature matrix $\mathbf{X} \in \mathbb{R}^{n \times p}$, with n and p denoting the number of observations and the number of predictors, respectively. In *group selection*, we decompose the feature matrix $\mathbf{X}$ into J submatrices as

$$\mathbf{X} = [\mathbf{X}_1, \ldots, \mathbf{X}_J],$$

where $\mathbf{X}_j \in \mathbb{R}^{n \times p_j}$ for all j and $p_1 + \cdots + p_J = p$. Suppose that we are interested in determining which of the J feature submatrices are important in the linear regression model given by

$$\mathbf{Y} = \beta_0 \mathbf{1} + \mathbf{X}\boldsymbol{\beta} = \beta_0 \mathbf{1} + \mathbf{X}_1 \boldsymbol{\beta}_1 + \cdots + \mathbf{X}_J \boldsymbol{\beta}_J + \boldsymbol{\epsilon}, \tag{1}$$

where $\boldsymbol{\beta}_j \in \mathbb{R}^{p_j}$ are the coefficients for group j and $\boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, \sigma^2 \mathbf{I}_n)$. Henceforth, we assume without loss of generality that all predictors and the response have been centered to have zero mean; in this case, the intercept term β_0 may be omitted (Tibshirani, 1996).

One method to select entire groups of features is the group lasso technique (Yuan and Lin, 2006), an extension of the well-known lasso method (Tibshirani, 1996). The original group lasso algorithm Yuan and Lin (2006) required that each group matrix $\mathbf{X}_j$ be orthogonal. To address more general scenarios, Simon and Tibshirani (2012) introduced

the standardized group lasso. Similar to the vanilla lasso, the group lasso proposed in Simon and Tibshirani (2012) requires that the features be standardized as a preliminary step. However, it advocates a specific standardization technique called *sphering*, which has been shown to offer both theoretical and empirical advantages compared to not using sphering. Assuming that all $\mathbf{X}_j$'s are full rank, sphering requires that each block $\mathbf{X}_j$ of features is replaced by $\mathbf{Z}_j$, where $\mathbf{X}_j = \mathbf{Z}_j\mathbf{R}_j$ is the QR decomposition of $\mathbf{X}_j$ and

$$\mathbf{Z}_j^\top \mathbf{Z}_j = \mathbf{I}_{p_j}, \quad j = 1, \ldots, J, \tag{2}$$

where $\mathbf{I}_{p_j} \in \mathbb{R}^{p_j \times p_j}$ is the identity matrix. Define $\boldsymbol{\theta}_j := \mathbf{R}_j\boldsymbol{\beta}_j$ for all j. We then minimize, in terms of $\boldsymbol{\theta} = [\boldsymbol{\theta}_1^\top, \ldots, \boldsymbol{\theta}_J^\top]^\top$, the group lasso objective:

$$f(\boldsymbol{\theta}) := \frac{1}{2n}\left\|\boldsymbol{y} - \sum_{j=1}^{J}\mathbf{Z}_j\boldsymbol{\theta}_j\right\|_2^2 + \lambda\sum_{j=1}^{J}\sqrt{p_j}\,\|\boldsymbol{\theta}_j\|_2. \tag{3}$$

Denote $\widehat{\boldsymbol{\theta}}_j$, $k = 1, \ldots, J$ to be the minimizer of (3). We can then obtain estimates for the variables $\boldsymbol{\beta}_j$ in the original model by applying the transformation $\widehat{\boldsymbol{\beta}}_j = \mathbf{R}_j^{-1}\widehat{\boldsymbol{\theta}}_j$ for all j. This transformation assumes that each matrix $\mathbf{X}_j$ is of full rank and that $\mathbf{R}_j$ is an upper-triangular matrix. For completeness, in the supplementary Section A we describe how sphering is modified in the rank-deficient case.

The minimizer of (3) achieves group sparsity due to the non-smoothness of $\|\cdot\|_2$ in the penalty $\sum_j \sqrt{p_j}\,\|\boldsymbol{\theta}_j\|_2 = \sum_j \sqrt{p_j}\|\mathbf{X}_j\boldsymbol{\beta}_j\|_2$. The penalty encourages entire groups of mean responses $\{\mathbf{X}_j\boldsymbol{\beta}_j\}$ to be set to zero, thereby selecting only the most relevant groups for prediction. In particular, for a given value of $\lambda > 0$, either the entire vector $\widehat{\boldsymbol{\theta}}_j$ associated with a group of predictors will be zero or all its elements will be non-zero. The larger the value of λ, the more stringent the penalty, leading to more groups being shrunk to zero. Consequently, a higher λ value promotes greater sparsity.

Unlike in the sphered case, when the sphered matrices $\mathbf{Z}_j$ are replaced with the original non-sphered group matrices $\mathbf{X}_j$ in (3), a closed-form block coordinate update does not exist. Numerous algorithms have been proposed for this setting, including proximal gradient

methods such as the so-called ISTA and FISTA algorithms in Qin et al. (2013), as well as Newton-type algorithms (Yang and Hastie, 2024). These algorithms employ sub-iterative methods to solve each block coordinate update.

A simple block coordinate descent algorithm for the sphered group lasso is detailed in Breheny and Huang (2015), with an accompanying R package, `grpreg`. While block coordinate descent (BCD) is straightforward, we observe in our simulations that it is numerically slow when computing the entire regularization path with many activated variables. This slowdown is particularly noticeable when both the number of activated groups in the solution and the condition number of the feature matrix $\mathbf{X}$ are large.

In this paper, we present a fast majorization-minimization (MM) algorithm (Hunter and Lange, 2004) for solving the sphered grouped lasso problem. Previous works, such as Wu and Lange (2008); Yang and Zou (2015); Helwig (2025), provide a group-wise MM algorithm on the original parameters for the non-sphered problem using a quadratic bound. Our method, however, minimizes a novel objective with auxiliary variables that satisfy the group-lasso Karush-Kuhn-Tucker (KKT) conditions. By leveraging the orthogonality of the $\mathbf{Z}_j$ matrices, we derive an adaptive majorizing bound, where at each iteration, we optimally select a weight for each group to provide the best possible majorizing bound. This method is implemented efficiently using conjugate gradient techniques and methods that store and work only with activated variables.

In Section 2, we review the block coordinate descent algorithm for group lasso. In Section 3, we formulate our MM algorithm for minimizing (3) using auxiliary variables. We also describe how the algorithm is parametrized by weights and how these weights can be optimally chosen. Additionally, we prove the global convergence of our MM algorithm in this section. In Section 4, we provide extensive numerical experiments comparing the MM algorithm with the BCD algorithm. Concluding remarks and potential future research directions are discussed in Section 5.

2 Block Coordinate Descent

In this section, we restate the method from Yuan and Lin (2006); Simon and Tibshirani (2012); Breheny and Huang (2015) for minimizing (3) using a *block coordinate descent* (BCD) algorithm. In BCD, variables are partitioned into non-overlapping blocks, and at each iteration, we minimize over the variables within a block, while keeping all other variables fixed. More precisely, we select a group k in each iteration and update the corresponding group parameters $\boldsymbol{\theta}_j$ using

$$\boldsymbol{\theta}_j \leftarrow \underset{\boldsymbol{\theta}_j}{\operatorname{argmin}} f\left(\boldsymbol{\theta}_1, \boldsymbol{\theta}_2, \ldots, \boldsymbol{\theta}_{j-1}, \boldsymbol{\theta}_j, \boldsymbol{\theta}_{j+1}, \ldots, \boldsymbol{\theta}_J\right), \quad j = 1, \ldots, J. \tag{4}$$

Conventionally, the J blocks are updated in the fixed order $\boldsymbol{\theta}_1 \to \boldsymbol{\theta}_2 \to \cdots \to \boldsymbol{\theta}_J$.

The function $f(\boldsymbol{\theta})$ is a sum of a smooth sum-of-squares error term and a penalty that is separable with respect to the block variables $\boldsymbol{\theta}_1, \ldots, \boldsymbol{\theta}_J$, that is, it can be written as $\sum_j \rho_j(\boldsymbol{\theta}_j)$ for some functions $\{\rho_j\}$. For functions composed in this way, the updating rule (4) produces a sequence $\{\boldsymbol{\theta}^{(t)}\}_{t\geq 0}$ that converges to the global minimizer; see Tseng (2001) for further details.

To derive an exact formula for (4), we need to solve the first-order necessary condition. To see this, we define the partial residual that omits the k-th group as follows:

$$\boldsymbol{r}_{\neg k} := \boldsymbol{y} - \sum_{j \neq k} \mathbf{Z}_j \boldsymbol{\theta}_j.$$

Further, if we define $\alpha_j := n\lambda\sqrt{p_j}$ for all $j = 1, \ldots, J$, then the first-order necessary condition for the group minimization (4) can be expressed as

$$\mathbf{Z}_j^\top (\boldsymbol{r}_{\neg j} - \mathbf{Z}_j \boldsymbol{\theta}_j) = \alpha_j \boldsymbol{g}(\boldsymbol{\theta}_j), \quad j = 1, \ldots, J, \tag{5}$$

where

$$\boldsymbol{g}(\boldsymbol{\eta}) = \begin{cases} \frac{\boldsymbol{\eta}}{\|\boldsymbol{\eta}\|_2}, & \boldsymbol{\eta} \neq \mathbf{0} \\ \in \{\boldsymbol{s} : \|\boldsymbol{s}\|_2 \leq 1\}, & \boldsymbol{\eta} = \mathbf{0} \end{cases},$$

is the subgradient of the function $\|\boldsymbol{\eta}\|_2$. Using $\mathbf{Z}_j^\top \mathbf{Z}_j = \mathbf{I}_{p_j}$, the solution to (5) for a single coordinate j is

$$\boldsymbol{\theta}_j \leftarrow S_{\alpha_j}(\mathbf{Z}_j^\top \boldsymbol{r}_{\neg j}), \tag{6}$$

where $[a]_+ := \max(0, a)$ and

$$S_\lambda(\boldsymbol{x}) := \boldsymbol{x}[1 - \lambda/\|\boldsymbol{x}\|_2]_+ \tag{7}$$

is the lasso shrinkage operator with $S_\lambda(\mathbf{0}) = \mathbf{0}$.

If $\boldsymbol{r} := \boldsymbol{y} - \sum_j \mathbf{Z}_j \boldsymbol{\theta}_j$ represents the ordinary residual, then the block coordinate descent (BCD) algorithm described in Breheny and Huang (2015) simplifies the update in (6) to,

$$\boldsymbol{\theta}_j \leftarrow S_{\alpha_j}(\mathbf{Z}_j^\top \boldsymbol{r} + \boldsymbol{\theta}_j).$$

This provides a straightforward method to compute the update in (4). For completeness, we restate the BCD algorithm from Breheny and Huang (2015) as Algorithm B.1 in the supplementary Section B.

Overall, one block coordinate cycle requires $\mathcal{O}(np)$ operations. While the cost of $\mathcal{O}(np)$ per cycle is not prohibitive, the overall runtime of the BCD algorithm depends on the number t of cycle iterations needed to achieve convergence. Define $\mathbf{Z} := [\mathbf{Z}_1, \ldots, \mathbf{Z}_J]$ as the concatenation of the sphered group matrices. The magnitude of t is influenced by the training set $\{\boldsymbol{y}, \mathbf{Z}\}$, the values of the penalty parameter λ, and the error tolerance. Empirically, we observe that if λ is small and the condition number of the matrix $\mathbf{Z}$ of all features is large, then the number of iterations t necessary for convergence can be prohibitively large. In the next section, we present an MM algorithm that is more efficient in terms of the number of iterations needed to achieve fast convergence under these conditions.

3 Auxiliary MM Algorithm

We now introduce an alternative and more efficient algorithm for minimizing (3) based on the MM principle. Denote $\mathbb{R}_+$ as the set of non-negative real numbers. The proposed MM

algorithm operates over auxiliary variables $\boldsymbol{\gamma} = [\gamma_1, \ldots, \gamma_J]^\top \in \mathbb{R}_+^J$, instead of the original variables $\boldsymbol{\theta} = [\boldsymbol{\theta}_1^\top, \ldots, \boldsymbol{\theta}_J^\top]^\top$.

In this section, we establish a direct relationship between $\boldsymbol{\gamma}$ and $\boldsymbol{\theta}$ and formulate a new objective function $g(\boldsymbol{\gamma})$ solely in terms of $\boldsymbol{\gamma}$. We thus reparameterize the optimization from a space in $\mathbb{R}^p$ to a space in $\mathbb{R}^J$. We demonstrate that the first-order conditions for minimizing $g(\boldsymbol{\gamma})$ are equivalent to the Karush-Kuhn-Tucker (KKT) conditions (5). We then derive an efficient MM algorithm to minimize the new auxiliary objective in $\boldsymbol{\gamma}$-space, ultimately obtaining the minimizer for (3). Finally, we establish the global convergence of the proposed MM algorithm.

Before defining the auxiliary variables $\boldsymbol{\gamma}$, it is useful to restate the Karush-Kuhn-Tucker (KKT) conditions (5) that the minimizer $\widehat{\boldsymbol{\theta}}$ satisfies in a different form. Define the set of so-called *activated* groups as

$$\mathcal{A} := \{j : \widehat{\boldsymbol{\theta}}_j \neq \mathbf{0}\},$$

which represents the set of non-zero groups in $\widehat{\boldsymbol{\theta}}$. Recall that $\mathbf{Z} = [\mathbf{Z}_1, \ldots, \mathbf{Z}_J]$ is the concatenation of the sphered group matrices. Then, equation (5) can be written as:

$$\begin{aligned} \mathbf{Z}_j^\top (\boldsymbol{y} - \mathbf{Z}\widehat{\boldsymbol{\theta}}) &= \frac{\widehat{\boldsymbol{\theta}}_j}{\|\widehat{\boldsymbol{\theta}}_j\|_2} \alpha_j, \quad && \forall j \in \mathcal{A}, \\ \left\| \mathbf{Z}_j^\top (\boldsymbol{y} - \mathbf{Z}\widehat{\boldsymbol{\theta}}) \right\|_2 &\leq \alpha_j, && \forall j \notin \mathcal{A}. \end{aligned} \tag{8}$$

3.1 Auxiliary Variable Group Lasso

Consider the auxiliary variables $\boldsymbol{\gamma} = [\gamma_1, \ldots, \gamma_J]^\top \in \mathbb{R}_+^J$. Define

$$\boldsymbol{\Gamma} := \mathbf{blcdiag}(\sqrt{\gamma_1}\, \mathbf{I}_{p_1}, \ldots, \sqrt{\gamma_J}\, \mathbf{I}_{p_J}), \tag{9}$$

where the operator $\mathbf{blkdiag}(\mathbf{A}_1, \ldots, \mathbf{A}_J)$ refers to the block diagonal matrix formed by placing the matrices $\mathbf{A}_1, \mathbf{A}_2, \ldots, \mathbf{A}_J$ along the diagonal and filling the off-diagonal blocks with zeros. Further, define

$$\mathbf{G}(\boldsymbol{\gamma}) := \left(\mathbf{I} + \mathbf{Z}\boldsymbol{\Gamma}^2\mathbf{Z}^\top\right)^{-1} = \left(\mathbf{I} + \sum_{j=1}^{J} \gamma_j \mathbf{Z}_j \mathbf{Z}_j^\top\right)^{-1}.$$

When we use $\widehat{\boldsymbol{\gamma}}$ in (9), instead of $\boldsymbol{\gamma}$, then we add a hat symbol to the matrix $\boldsymbol{\Gamma}$. Recall that $\alpha_j = n\lambda\sqrt{p_j}$. With these definitions, we consider the following theorem, whose proof is provided in supplementary Section D.

Theorem 3.1 (Auxiliary Variable KKT Conditions). *Let* $\boldsymbol{\theta} = \boldsymbol{\Gamma}^2\mathbf{Z}^\top\mathbf{G}(\boldsymbol{\gamma})\boldsymbol{y}$, *where each block within* $\boldsymbol{\theta} = [\boldsymbol{\theta}_1^\top, \ldots, \boldsymbol{\theta}_J^\top]^\top$ *is* $\boldsymbol{\theta}_j = \gamma_j\mathbf{Z}_j^\top\mathbf{G}(\boldsymbol{\gamma})\boldsymbol{y}$ *for* $j = 1, \ldots, J$. *Then,* $\boldsymbol{\theta}$ *satisfies the group-lasso KKT conditions* (8) *if and only if* $\boldsymbol{\gamma}$ *satisfies*

$$\begin{aligned} \|\mathbf{Z}_j^\top\mathbf{G}(\boldsymbol{\gamma})\boldsymbol{y}\|_2 &= \alpha_j, \quad \forall j \in \mathcal{A}, \\ \left\|\mathbf{Z}_j^\top\mathbf{G}(\boldsymbol{\gamma})\boldsymbol{y}\right\|_2 &\leq \alpha_j, \quad \forall j \notin \mathcal{A}. \end{aligned} \tag{10}$$

Moreover, $\gamma_j = \|\boldsymbol{\theta}_j\|/\alpha_j$ *for* $j \in \mathcal{A}$ *and* $\gamma_j\|\mathbf{Z}_j^\top\mathbf{G}(\boldsymbol{\gamma})\boldsymbol{y}\|_2 = 0$ *for* $j \notin \mathcal{A}$.

An advantage of the alternative set of equations (10) is that they correspond to the KKT conditions of a new objective function expressed solely in terms of the auxiliary variables $\boldsymbol{\gamma}$. Define $\boldsymbol{\alpha} := [\alpha_1, \ldots, \alpha_J]^\top$ and consider the function:

$$g(\boldsymbol{\gamma}) := \boldsymbol{y}^\top\mathbf{G}(\boldsymbol{\gamma})\boldsymbol{y} + (\boldsymbol{\alpha} \odot \boldsymbol{\alpha})^\top\boldsymbol{\gamma}, \tag{11}$$

where $\odot$ denotes the Hadamard product (element-wise product) of two vectors. The following lemma asserts that the minimizer of (11) satisfies the conditions (10).

Lemma 3.1 (Auxiliary Problem). *Let* $\widehat{\boldsymbol{\gamma}}$ *be the global minimizer of the convex constrained optimization problem:*

$$\widehat{\boldsymbol{\gamma}} \in \underset{\boldsymbol{\gamma}\in\mathbb{R}_+^J}{\operatorname{argmin}}\, g(\boldsymbol{\gamma}). \tag{12}$$

The solution $\widehat{\boldsymbol{\gamma}}$ *satisfies the auxiliary KKT conditions given by*

$$\left\|\mathbf{Z}_j^\top\mathbf{G}(\widehat{\boldsymbol{\gamma}})\boldsymbol{y}\right\|_2 = \alpha_j \quad \forall\, \widehat{\gamma}_j > 0, \tag{13}$$

$$\left\|\mathbf{Z}_j^\top\mathbf{G}(\widehat{\boldsymbol{\gamma}})\boldsymbol{y}\right\|_2 \leq \alpha_j \quad \forall\, \widehat{\gamma}_j = 0. \tag{14}$$

and therefore satisfies (10) *in Theorem 3.1.*

Corollary 3.1. *The vector* $\widehat{\boldsymbol{\theta}} = \widehat{\boldsymbol{\Gamma}}^2 \mathbf{Z}^\top \mathbf{G}(\widehat{\boldsymbol{\gamma}})\boldsymbol{y}$ *is a minimizer of the original group lasso problem* (3). *Furthermore, the activated set* $\mathcal{A}(\widehat{\boldsymbol{\theta}}) \equiv \{j : \widehat{\gamma}_j > 0\}$.

Once $\widehat{\boldsymbol{\gamma}}$ is obtained, we can interpret $\mathcal{A}^c := \{j : j \notin \mathcal{A}\} = \{j : \widehat{\gamma}_j = 0\}$ as the set of groups that are unactivated and, conversely, $\mathcal{A} = \{j : \widehat{\gamma}_j > 0\}$ as the set of activated groups. The activated group coefficients for $j \in \mathcal{A}$ can be recovered using the formula $\widehat{\boldsymbol{\theta}}_j = \widehat{\gamma}_j \mathbf{Z}_j^\top \mathbf{G}(\widehat{\boldsymbol{\gamma}})\boldsymbol{y}$ and the group coefficients that are unactivated are set to $\widehat{\boldsymbol{\theta}}_j = \mathbf{0}$.

At first glance, solving the problem (12) may seem more computationally challenging than minimizing the original group lasso objective (3). For example, one approach to solving (12) would be to implement a coordinate descent algorithm over the variables $\boldsymbol{\gamma}$. If we define $\mathbf{G}^{-1}(\boldsymbol{\gamma}) := \mathbf{I} + \sum_j \gamma_j \mathbf{Z}_j \mathbf{Z}_j^\top$ and $\boldsymbol{\gamma}_{\neg j}$ as the vector that omits the j-th element of $\boldsymbol{\gamma}$. Then, each coordinate update is

$$\gamma_j \leftarrow \underset{x \in \mathbb{R}_+}{\operatorname{argmin}} \; \boldsymbol{y}^\top \left(\mathbf{G}^{-1}(\boldsymbol{\gamma}_{\neg k}) + x \mathbf{Z}_j \mathbf{Z}_j^\top \right)^{-1} \boldsymbol{y} + \alpha_j^2 x.$$

This approach requires an iterative algorithm, such as Newton's or the Secant method, to generate a sequence $\{x^{(l)}\}_{l \geq 0}$ that converges to the correct coordinate update. At each step l of the iterative algorithm, we need to solve the linear system

$$\left(\mathbf{G}^{-1}(\boldsymbol{\gamma}_{\neg j}) + x^{(l)} \mathbf{Z}_j \mathbf{Z}_j^\top \right)^{-1} \boldsymbol{y}.$$

In the group lasso context, however, this coordinate descent approach to solve (12) was not competitive against the BCD algorithm (see supplementary Algorithm B.1) due to the time it takes to run Newton's method.

Instead of using coordinate descent, we leverage the orthogonality of the matrices $\mathbf{Z}_j$ to derive an MM algorithm that updates all the variables $\boldsymbol{\gamma}^{(t)} = [\gamma_1^{(t)}, \ldots, \gamma_J^{(t)}]^\top$ concurrently using a closed-form updating formula. In a single MM update, we solve the linear system $\mathbf{G}(\boldsymbol{\gamma}^{(t)})\boldsymbol{y}$ only once to update all the variables $\gamma_1, \ldots, \gamma_J$.

3.2 Constructing an MM Algorithm

The MM algorithm (Hunter and Lange, 2004) is an iterative method for minimization that can be used on the objective function $g(\boldsymbol{\gamma})$ given in (11). It does this by successively majorizing $g(\boldsymbol{\gamma})$ with a surrogate function $h(\boldsymbol{\gamma}, \widetilde{\boldsymbol{\gamma}})$, where $\widetilde{\boldsymbol{\gamma}} \in \mathbb{R}_+^J$ are latent variables. The conditions that define the majorization are

$$\begin{aligned} g(\widetilde{\boldsymbol{\gamma}}) &= h(\widetilde{\boldsymbol{\gamma}}, \widetilde{\boldsymbol{\gamma}}), \\ g(\boldsymbol{\gamma}) &\leq h(\boldsymbol{\gamma}, \widetilde{\boldsymbol{\gamma}}), \quad \boldsymbol{\gamma} \in \mathbb{R}_+^J. \end{aligned} \tag{15}$$

In other words, the function $h(\boldsymbol{\gamma}, \widetilde{\boldsymbol{\gamma}})$ lies above $g(\boldsymbol{\gamma})$ and is tangent to it at $\boldsymbol{\gamma} = \widetilde{\boldsymbol{\gamma}}$. Suppose at iteration t of the MM algorithm, we set $\widetilde{\boldsymbol{\gamma}} \leftarrow \boldsymbol{\gamma}^{(t)}$, that is, we set the latent variables as the current iterate. To obtain the next iterate, we minimize the surrogate $h(\boldsymbol{\gamma}, \boldsymbol{\gamma}^{(t)})$ in $\boldsymbol{\gamma}$. Thus, the MM map, $M(\cdot) := \operatorname{argmin}_{\boldsymbol{\gamma} \in \mathbb{R}_+^J} h(\boldsymbol{\gamma}, \cdot)$ allows us to write the update as

$$\boldsymbol{\gamma}^{(t+1)} \leftarrow M(\boldsymbol{\gamma}^{(t)}). \tag{16}$$

The latter update ensures the descent property $g(\boldsymbol{\gamma}^{(t+1)}) = g(M(\boldsymbol{\gamma}^{(t)})) \leq g(\boldsymbol{\gamma}^{(t)})$, because

$$g(\boldsymbol{\gamma}^{(t+1)}) \leq h(\boldsymbol{\gamma}^{(t+1)}, \boldsymbol{\gamma}^{(t)}) \leq h(\boldsymbol{\gamma}^{(t)}, \boldsymbol{\gamma}^{(t)}) = g(\boldsymbol{\gamma}^{(t)}),$$

where the first inequality and the last equality follow from conditions (15) and the second inequality follows from (16). We now construct a surrogate function $h(\boldsymbol{\gamma}, \widetilde{\boldsymbol{\gamma}})$ for $g(\boldsymbol{\gamma})$ that can be easily minimized. To achieve this, we first obtain an upper bound for the term $\boldsymbol{y}^\top \mathbf{G}(\boldsymbol{\gamma}) \boldsymbol{y}$.

Let $\boldsymbol{w} = [w_1, \ldots, w_J]^\top$ be a probability vector (i.e., $\sum_{j=1}^J w_j = 1$ and $w_j \geq 0$ for all j). Then,

$$\mathbf{G}^{-1}(\boldsymbol{\gamma}) = \mathbf{I} + \sum_{j=1}^J \gamma_j \mathbf{Z}_j \mathbf{Z}_j^\top = \sum_{j=1}^J (\gamma_j + w_j) \mathbf{Z}_j \mathbf{Z}_j^\top + \sum_{j=1}^J w_j (\mathbf{I} - \mathbf{Z}_j \mathbf{Z}_j^\top).$$

Consider the following *matrix inequality* (Zhou et al., 2019)[1]:

$$\left(\sum_{j=1}^{2J} \sigma_j \mathbf{V}_j \right)^{-1} \preceq \left(\sum_{j=1}^{2J} \widetilde{\sigma}_j \mathbf{V}_j \right)^{-1} \left(\sum_{j=1}^{2J} \frac{\widetilde{\sigma}_j^2}{\sigma_j} \mathbf{V}_j \right) \left(\sum_{j=1}^{2J} \widetilde{\sigma}_j \mathbf{V}_j \right)^{-1}, \tag{17}$$

[1]Here $\mathbf{A} \succeq \mathbf{0} \Rightarrow \mathbf{A}$ is positive semi-definite.

where $\mathbf{V}_j \in \mathbb{R}^{n\times n}$ are symmetric positive semi-definite matrices for all j, $\boldsymbol{\sigma} \in \mathbb{R}^{2J}_{++}$ and $\widetilde{\boldsymbol{\sigma}} \in \mathbb{R}^{2J}_{+}\backslash\{\mathbf{0}\}$. Equality is achieved when $\widetilde{\sigma}_j = \sigma_j$ for $j = 1, \ldots, 2J$. The matrix inequality (17) is a generalization of Sedrakyan's inequality (a special case of the Cauchy-Schwarz inequality) for the real numbers $\widetilde{\sigma}_1, \ldots, \widetilde{\sigma}_{2J}$ and positive real numbers $\sigma_1, \ldots, \sigma_{2J}$:

$$\frac{\left(\sum_{j=1}^{2J} \widetilde{\sigma}_j\right)^2}{\sum_{j=1}^{2J} \sigma_j} \leq \sum_{j=1}^{2J} \frac{\widetilde{\sigma}_j^2}{\sigma_j}.$$

We apply inequality (17) to obtain an upper bound for $\mathbf{G}(\boldsymbol{\gamma})$. Since $\mathbf{Z}_j^\top \mathbf{Z}_j = \mathbf{I}_{p_j}$ implies that $\mathbf{I} - \mathbf{Z}_j\mathbf{Z}_j^\top$ is a projection matrix and positive semi-definite, we apply (17) with

$$\sigma_j = \gamma_j + w_j, \quad \widetilde{\sigma}_j = \widetilde{\gamma}_j + w_j, \quad \mathbf{V}_j = \mathbf{Z}_j\mathbf{Z}_j^\top, \quad j = 1, \ldots, J,$$

and

$$\sigma_{J+j} = w_j, \quad \widetilde{\sigma}_{J+j} = w_j, \quad \mathbf{V}_{J+j} = \mathbf{I} - \mathbf{Z}_j\mathbf{Z}_j^\top, \quad j = 1, \ldots, J.$$

Hence, we obtain the matrix inequality:

$$\mathbf{G}(\boldsymbol{\gamma}) \preceq \sum_j \frac{(\widetilde{\gamma}_j + w_j)^2}{\gamma_j + w_j} \mathbf{G}(\widetilde{\boldsymbol{\gamma}})\mathbf{Z}_j\mathbf{Z}_j^\top\mathbf{G}(\widetilde{\boldsymbol{\gamma}}) + \sum_j w_j \mathbf{G}(\widetilde{\boldsymbol{\gamma}})(\mathbf{I} - \mathbf{Z}_j\mathbf{Z}_j^\top)\mathbf{G}(\widetilde{\boldsymbol{\gamma}}). \tag{18}$$

We can obtain a bound for $\boldsymbol{y}^\top\mathbf{G}(\boldsymbol{\gamma})\boldsymbol{y}$ by multiplying this inequality on the left by $\boldsymbol{y}^\top$ and on the right by $\boldsymbol{y}$. Adding the linear term $(\boldsymbol{\alpha}\odot\boldsymbol{\alpha})^\top\boldsymbol{\gamma}$ to the bound for $\boldsymbol{y}^\top\mathbf{G}(\boldsymbol{\gamma})\boldsymbol{y}$, we obtain the surrogate function

$$h_{\boldsymbol{w}}(\boldsymbol{\gamma}, \widetilde{\boldsymbol{\gamma}}) = \sum_j \left[\frac{(\widetilde{\gamma}_j + w_j)^2 \|\mathbf{Z}_j^\top\mathbf{G}(\widetilde{\boldsymbol{\gamma}})\boldsymbol{y}\|^2}{\gamma_j + w_j} + w_j \left(\|\mathbf{G}(\widetilde{\boldsymbol{\gamma}})\boldsymbol{y}\|^2 - \|\mathbf{Z}_j^\top\mathbf{G}(\widetilde{\boldsymbol{\gamma}})\boldsymbol{y}\|^2 \right) + \alpha_j^2\gamma_j \right],$$

parametrized by weights $\boldsymbol{w}$. Equality with $g(\boldsymbol{\gamma})$ is achieved when $\widetilde{\boldsymbol{\gamma}} = \boldsymbol{\gamma}$ and $g(\boldsymbol{\gamma}) \leq h_{\boldsymbol{w}}(\boldsymbol{\gamma}, \widetilde{\boldsymbol{\gamma}})$ thereby satisfying conditions (15). The key advantage is that the surrogate $h_{\boldsymbol{w}}(\boldsymbol{\gamma}, \widetilde{\boldsymbol{\gamma}})$ is simpler to minimize than $g(\boldsymbol{\gamma})$ as it separates the variables $\gamma_1, \ldots, \gamma_J$. Minimizing $h_{\boldsymbol{w}}(\boldsymbol{\gamma}, \boldsymbol{\gamma}^{(t)})$ with respect to $\gamma_j \in [0, \infty)$ yields the simple update

$$\gamma_j^{(t+1)} \leftarrow \frac{(w_j + \gamma_j^{(t)}) \left\|\mathbf{Z}_j^\top\mathbf{G}(\boldsymbol{\gamma}^{(t)})\boldsymbol{y}\right\|}{\alpha_j} \left[1 - \frac{w_j\alpha_j}{(w_j + \gamma_j^{(t)}) \left\|\mathbf{Z}_j^\top\mathbf{G}(\boldsymbol{\gamma}^{(t)})\boldsymbol{y}\right\|} \right]_+. \tag{19}$$

Using $S_\lambda(\boldsymbol{x})$, which is the lasso shrinkage operator (7), we simplify the update for $\gamma_1, \ldots, \gamma_j$ as

$$\gamma_j^{(t+1)} \leftarrow S_{\alpha_j w_j}\left((w_j + \gamma_j^{(t)})\tau_j^{(t)}\right) \Big/ \alpha_j, \quad j = 1, \ldots, J,$$

where $\tau_j^{(t)} := \left\|\mathbf{Z}_j^\top \mathbf{G}(\boldsymbol{\gamma}^{(t)})\boldsymbol{y}\right\|$. The MM steps guarantee a non-increasing sequence $\{g(\boldsymbol{\gamma}^{(t)})\}_{t\geq 0}$ as long as valid weights parametrize the surrogate function $h_{\boldsymbol{w}}$. At each MM step, we select weights that yield the "best" majorizing function. The optimal weight allocation is described in more detail in Section 3.3.

We present a summary of the MM algorithm for minimizing $f(\boldsymbol{\theta})$ in Algorithm C.1, included in the supplementary Section C. In this algorithm, we initialize the activated set to be the empty set, $\mathcal{A}^{(0)} = \emptyset$, and keep track of $\mathcal{A}^{(t)}$ and $\mathbf{Z}_{\mathcal{A}^{(t)}}$ as we add activated groups. This approach allows us to efficiently compute the vector $\mathbf{G}(\boldsymbol{\gamma}^{(t)})\boldsymbol{y}$ at each iteration t. We compute $\mathbf{G}(\boldsymbol{\gamma}^{(t)})\boldsymbol{y}$ via the conjugate gradient (CG) method (Golub and Van Loan, 2013), an iterative algorithm that only requires matrix-vector multiplications with $\mathbf{G}^{-1}(\boldsymbol{\gamma}^{(t)})$. At iteration t, we initialize the CG with the initial guess $\mathbf{G}(\boldsymbol{\gamma}^{(t-1)})\boldsymbol{y}$ and express the matrix $\mathbf{G}^{-1}(\boldsymbol{\gamma}^{(t)})$ as

$$\mathbf{G}^{-1}(\boldsymbol{\gamma}^{(t)}) = \mathbf{I} + \sum_{j\in\mathcal{A}^{(t)}} \gamma_j^{(t)} \mathbf{Z}_j \mathbf{Z}_j^\top = \mathbf{I} + \mathbf{Z}_{\mathcal{A}^{(t)}} \boldsymbol{\Gamma}^{(t)}_{\mathcal{A}^{(t)}} \mathbf{Z}_{\mathcal{A}^{(t)}}^\top.$$

To approximate $\mathbf{G}(\boldsymbol{\gamma}^{(t)})\boldsymbol{y}$, we take $c < n$ steps of the CG method with the matrix-vector multiplication operator $\boldsymbol{x} \mapsto \boldsymbol{x} + \mathbf{Z}_{\mathcal{A}^{(t)}}\left(\boldsymbol{\gamma}_{\mathcal{A}^{(t)}} \odot (\mathbf{Z}_{\mathcal{A}^{(t)}}^\top \boldsymbol{x})\right)$, at a cost of $\mathcal{O}(c \times n \times \sum_{j\in\mathcal{A}^{(t)}} p_j)$.

We find that the optimal value of c lies within a "Goldilocks" range. If c is too small, there is an excessive error in approximating $\mathbf{G}(\boldsymbol{\gamma}^{(t)})\boldsymbol{y}$, leading to a large number t of iterations required to converge, as each MM step is not optimal. On the other hand, if c is too large, too much time is spent approximating $\mathbf{G}(\boldsymbol{\gamma}^{(t)})\boldsymbol{y}$ to an unnecessarily high precision, increasing the cost per iteration. In our experiments, c is set to a relatively small number (less than 10).

In the worst case, one iteration of the MM algorithm requires $\mathcal{O}(cnp)$ operations, however, when $\boldsymbol{\gamma}^{(t)}$ is sparse and $|\mathcal{A}^{(t)}|$ is small, the cost per iteration is significantly reduced. If

$|\mathcal{A}^{(t)}|$ is small enough, the most expensive operation in the MM iteration becomes computing the vector $\boldsymbol{\tau}^{(t)} = \left[\left\|\mathbf{Z}_1^\top \mathbf{G}(\boldsymbol{\gamma}^{(t)})\boldsymbol{y}\right\|, \ldots, \left\|\mathbf{Z}_J^\top \mathbf{G}(\boldsymbol{\gamma}^{(t)})\boldsymbol{y}\right\|\right]$ which requires $\mathcal{O}(np)$ operations.

3.3 Optimal Weights

Recall that we only need to ensure $w_j \geq 0$ for all j and $\sum_j w_j = 1$ to guarantee a valid majorization and thus a non-increasing sequence of objective evaluations. When $w_j = 0$, the MM update (19) simplifies to

$$\gamma_j^{(t+1)} \leftarrow \gamma_j^{(t)} \frac{\left\|\mathbf{Z}_j^\top \mathbf{G}(\boldsymbol{\gamma}^{(t)})\boldsymbol{y}\right\|}{\alpha_j}. \tag{20}$$

This multiplicative update implies that we can only switch an activated group to unactivated and vice versa when $w_j > 0$. In other words, the lasso shrinkage operator (7) appears only when $w_j > 0$. It might seem appropriate to set $\boldsymbol{w} \propto \mathbf{1}$ and assign a strictly positive weight to each group. However, assigning positive weight to only a few carefully selected groups is more effective. We now describe how to allocate these positive weights.

The rate of convergence of the MM algorithm depends on how well the surrogate function $h_{\boldsymbol{w}}(\boldsymbol{\gamma}, \widetilde{\boldsymbol{\gamma}})$ approximates the function $g(\boldsymbol{\gamma})$ (Lange, 2016). In fact, the objective evaluated at $\boldsymbol{\gamma}^{(t+1)}$ is bounded from above as

$$g(\boldsymbol{\gamma}^{(t+1)}) \leq h_{\boldsymbol{w}}\big(M(\boldsymbol{\gamma}^{(t)}), \boldsymbol{\gamma}^{(t)}\big).$$

At each iteration of the MM algorithm, we select the least upper bound $h_{\boldsymbol{w}}\big(M(\boldsymbol{\gamma}^{(t)}), \boldsymbol{\gamma}^{(t)}\big)$ to "drive down" the value of $g(\boldsymbol{\gamma}^{(t+1)})$ as much as possible. This equates to solving

$$\boldsymbol{w}^{(t+1)} = \operatorname*{argmin}_{\boldsymbol{w} \in \mathbb{R}_+^J, \mathbf{1}^\top \boldsymbol{w} = 1} h_{\boldsymbol{w}}\big(M(\boldsymbol{\gamma}^{(t)}), \boldsymbol{\gamma}^{(t)}\big) = \operatorname*{argmin}_{\boldsymbol{w} \in \mathbb{R}_+^J, \mathbf{1}^\top \boldsymbol{w} = 1} \min_{\boldsymbol{\gamma} \in \mathbb{R}_+^J} h_{\boldsymbol{w}}(\boldsymbol{\gamma}, \boldsymbol{\gamma}^{(t)}). \tag{21}$$

This problem is a constrained convex optimization that can be solved exactly in the fast $\mathcal{O}(J \ln J)$ time. In supplementary Section G, we derive the solution and describe its computation in the function `getWeights` which is implemented in Algorithm G.1. In short, the function `getWeights` strategically assigns positive weights to the groups for which the

values of $\xi_j := (\alpha_j - \tau_j^{(t)})^2 = (\alpha_j - \left\|\mathbf{Z}_j^\top \mathbf{G}(\boldsymbol{\gamma}^{(t)})\boldsymbol{y}\right\|)^2$ are the largest. We interpret these assignments via the auxiliary KKT conditions (13) and (14).

If ξ_j is large and group j is activated ($\gamma_j^{(t)} > 0$), then, either condition (13) or (14) is far from being met. In this case, by assigning $w_j > 0$, we prioritize group j in the MM update, pushing it closer to satisfying one of the KKT conditions. If ξ_j is large, group j is unactivated ($\gamma_j^{(t)} = 0$), and $(\alpha_j - \left\|\mathbf{Z}_j^\top \mathbf{G}(\boldsymbol{\gamma}^{(t)})\boldsymbol{y}\right\|) < 0$, then group j should belong to the activated set. In this case, `getWeights` assigns a positive weight $w_j > 0$, effectively moving j into the activated set ($\gamma_j^{(t+1)} > 0$). Conversely, if $\gamma_j^{(t)} = 0$ and $(\alpha_j - \left\|\mathbf{Z}_j^\top \mathbf{G}(\boldsymbol{\gamma}^{(t)})\boldsymbol{y}\right\|) \geq 0$, then (14) is met, and `getWeights` assigns $w_j = 0$, meaning group j remains unactivated ($\gamma_j^{(t+1)} = 0$). In summary, `getWeights` selectively assigns the weights $\{w_j\}$ at each iteration to focus on the groups that are furthest from satisfying the KKT conditions.

Recall that in a single MM iteration, only a mass of 1 can be assigned to the weights $\{w_j\}$. In practice, only a few groups are prioritized with positive weight. For the rest of the groups for which $w_j = 0$, the multiplicative update (20) is used. In other words, if $\gamma_j^{(t)} = 0$ and $w_j = 0$, then $\gamma_j^{(t+1)}$ remains zero.

Assigning weights $\{w_j\}$ provides functionality similar to the greedy coordinate descent method for lasso (Wu and Lange, 2008), where the variables selected for updating are those with the steepest negative directional derivative. Our approach updates all variables in a single MM iteration but emphasizes the groups that are furthest from satisfying the KKT conditions. This accelerates the MM algorithm's convergence rate with minimal additional cost, as the function `getWeights` only requires $\mathcal{O}(J \ln J)$ time, because the most resource-intensive step in `getWeights` is sorting the array $[\xi_1, \ldots, \xi_J]$.

3.4 Convergence and Uniqueness

We now establish the global convergence of the MM algorithm, with the main convergence result given in Theorem 3.2. Before presenting the theorem, we must first address the issue

of uniqueness. Although the objective $g(\boldsymbol{\gamma})$ is convex for $\boldsymbol{\gamma} \in \mathbb{R}^{\mathbb{J}}{}_{+}$, it is not necessarily strictly convex. As a result, the optimal solution set $\{\widehat{\boldsymbol{\gamma}} \in \mathbb{R}_{+}^{J} : \widehat{\boldsymbol{\gamma}} \in \operatorname{argmin}_{\boldsymbol{\gamma} \in \mathbb{R}_{+}^{J}} g(\boldsymbol{\gamma})\}$ may either contain infinitely many points or consist of a single point. As stated in Theorem 3.2, we can only guarantee the global convergence of the MM algorithm when $\widehat{\boldsymbol{\gamma}}$ is unique.

The original group lasso problem (3) yields a unique solution if the columns of $\mathbf{Z}$ are linearly independent, as shown by Mishkin and Pilanci (2022). A weaker condition, which can hold when $p > n$, referred to as *group general position* is also introduced in Mishkin and Pilanci (2022) and shown to be sufficient for uniqueness. We now state this assumption and demonstrate that it is also sufficient for the auxiliary problem (12) to yield a unique solution $\widehat{\boldsymbol{\gamma}}$.

Assumption 3.1 (Group General Position, Mishkin and Pilanci (2022))**.** *For every* $\mathcal{A} \subseteq \{1, \ldots, J\}, |\mathcal{A}| \leq n+1$, *there do not exist unit vectors* $\boldsymbol{v}_i \in \mathbb{R}^{p_i}$ *such that for every* $j \in \mathcal{A}$,

$$p_j^{-1/2} \mathbf{Z}_j \boldsymbol{v}_j \in \textit{affine}\left(\{p_i^{-1/2} \mathbf{Z}_i \boldsymbol{v}_i : i \in \mathcal{A} \setminus j\}\right),$$

where $\textit{affine}(\{\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_k\}) = \left\{\sum_{i=1}^{k} s_i \boldsymbol{x}_i : \sum_{i=1}^{k} s_i = 1,\ s_i \in \mathbb{R}\right\}$.

This assumption is referred to as *group general position* because it naturally extends the concept of *general position* (Tibshirani, 2013) to groups of column vectors. Although this condition may appear specific, the group general position assumption is naturally satisfied when the entries of $\mathbf{X}$ are drawn from a continuous probability distribution and $p_j < n$ for $j = 1, \ldots, J$. Assumption 3.1 is sufficient to ensure the uniqueness of (12), with a detailed proof provided in supplementary Section E.

Lemma 3.2 (Uniqueness of Auxiliary Objective)**.** *Suppose Assumption 3.1 holds. Then, the solution* $\widehat{\boldsymbol{\gamma}}$ *to* (12) *is unique.*

To prove convergence, we assume that the update $\mathbf{G}(\widetilde{\boldsymbol{\gamma}})\boldsymbol{y} \leftarrow \mathbf{G}(\boldsymbol{\gamma}^{(t)})\boldsymbol{y}$ is computed exactly at each iteration t, ignoring any numerical error from the CG approximation. Define the MM map with the optimal choice of weights as

$$\widehat{M}(\widetilde{\boldsymbol{\gamma}}) := \underset{\boldsymbol{\gamma} \in \mathbb{R}_+^J}{\operatorname{argmin}}\, h_{\widehat{\boldsymbol{w}}}(\boldsymbol{\gamma}, \widetilde{\boldsymbol{\gamma}})$$

where $\widehat{\boldsymbol{w}} := \operatorname{argmin}_{\boldsymbol{w} \in \mathbb{R}_+^J, \mathbf{1}^\top \boldsymbol{w} = 1} \min_{\boldsymbol{\gamma} \in \mathbb{R}_+^J} h_{\boldsymbol{w}}(\boldsymbol{\gamma}, \widetilde{\boldsymbol{\gamma}})$. Our main convergence result, whose proof is given in supplementary Section F, is the following.

Theorem 3.2 (MM Convergence)**.** *Let $\{\boldsymbol{\gamma}^{(t)}\}_{t \geq 0}$ be the sequence generated by $\widehat{M}$. If the solution $\widehat{\boldsymbol{\gamma}}$ to problem (12) is unique, then the MM sequence $\{\boldsymbol{\gamma}^{(t)}\}_{t \geq 0}$ converges to $\widehat{\boldsymbol{\gamma}}$.*

4 Numerical Experiments

In this section, we assess the efficiency of the MM algorithm (supplementary Algorithm C.1) and compare it against the current best alternative for the sphered group lasso, that is, the BCD algorithm (supplementary Algorithm B.1). All numerical experiments were conducted using Julia (version 1.10.4) on a Google Cloud Platform c2-standard-16 instance, which features 16 vCPUs and 64 GB of memory. In supplementary Section H, we briefly describe how to obtain $\widehat{\boldsymbol{\theta}}$ on the grid $\lambda_1, \ldots, \lambda_m$ of λ values.

We conduct our experiments in the following settings. In Section 4.1, we simulate data for the matrices $\mathbf{X}_j$ and the response $\boldsymbol{y}$ based on model (1). In Section 4.2, $\mathbf{X}$ and $\boldsymbol{y}$ are derived from a real dataset with anonymized data on foot traffic and sales volumes for a major Chinese supermarket (Wang, 2009; Thompson and Vahid, 2024). In supplementary Section I, we provide additional numerical studies. These include simulations with pseudo-real data, where $\mathbf{X}_j$ is constructed from a real dataset while $\boldsymbol{y}$ is simulated. We also investigate the empirical convergence behavior near the optimal solution and simulate $\mathbf{X}$ with groups of size one to mimic the lasso problem.

The metrics that we use in our experiments emphasize the computational cost of running the group lasso algorithm rather than the statistical accuracy of the solution. We run the MM algorithm and compare its runtime and number of iterations t until convergence with

the BCD algorithm. Unless specified otherwise, we set the error tolerance parameter for both algorithms to $\epsilon = 10^{-5}$. This means we terminate once $\max_j p_j^{-1} \|\boldsymbol{\theta}_j^{(t)} - \boldsymbol{\theta}_j^{(t-1)}\|_2 < 10^{-5}$. The number of conjugate gradient iterations in the MM algorithm is set to $c = 4$ for all experiments. While `R` packages such as `grpreg` implement the BCD algorithm, we chose to implement both algorithms in the programming language `Julia`. This choice ensures a proper comparison of running times and consistency in setting algorithm parameters. The `Julia` source code for our implementation and experiments is available in the supplementary Section.

4.1 Simulations

4.1.1 Synthetic Data Generation

To compare the computational performance of the MM algorithm and BCD algorithm, we use datasets simulated from the group linear model (1). Each group size is set to a fixed constant, $p_1 = p_2 = \cdots = p_J = d$. Each row of the feature matrix $\mathbf{X} = [\mathbf{X}_1, \ldots, \mathbf{X}_J]$ is generated from a multivariate normal distribution with zero mean and covariance matrix $\boldsymbol{\Sigma}$. The covariance matrix is constructed to have between-group positive correlation $\psi \in (0, 1)$ and within-group positive correlation $\rho \in (0, 1)$ as in Simon and Tibshirani (2012). Specifically,

$$\boldsymbol{\Sigma}_{i,j} = \begin{cases} 1, & \text{if } i = j, \\ \rho, & \text{if } \lceil i/d \rceil = \lceil j/d \rceil, \\ \psi, & \text{otherwise.} \end{cases}$$

The noise variance, σ^2, is chosen to achieve a signal-to-noise ratio (SNR) of 1, where $\text{SNR} := \mathbb{V}\text{ar}(\mathbf{X}\boldsymbol{\beta})/\sigma^2$.

The coefficient vector $\boldsymbol{\beta}$ contains k non-zero groups uniformly distributed among the J groups. We consider group sizes of $d = 3$ and $d = 5$ in the simulations. The following two settings are used for the values of the non-zero groups:

- **Setting 1**: For all activated groups j, $\boldsymbol{\beta}_j = [-1, 0, 1]^\top$ when $d = 3$, and $\boldsymbol{\beta}_j = [-2, -1, 0, 1, 2]^\top$ when $d = 5$.

- **Setting 2**: For all activated groups j, the elements of the vector $\boldsymbol{\beta}_j \in \mathbb{R}^d$ are drawn from a standard normal distribution.

We generate $\boldsymbol{y}$ using (1) and sphere the matrices $\mathbf{X}_1, \ldots, \mathbf{X}_J$ to obtain $\mathbf{Z}_1, \ldots, \mathbf{Z}_J$.

4.1.2 Computational Performance of Regularization Paths

We now present results where $\widehat{\boldsymbol{\theta}}$ is computed over a grid $\lambda_1 < \ldots < \lambda_m$, as described in supplementary Section H. We set $\lambda_1 = \omega\lambda_m$, where $\omega = 10^{-1}$. We run the MM algorithm and the BCD algorithm on datasets simulated under Setting 1 and Setting 2 with group sizes $d = 3$ and $d = 5$, with varying correlation values. In these simulations, the number of observations is $n = 1,000$ and the total number of predictors is $p = Jd = 1,500$. The number of activated groups is equal to $k = J/10$. Mean runtimes over 50 simulations are reported in Table 1. A table of mean itearations is provided in supplementary Section I.1.

The results indicate that our MM algorithm significantly outperforms the BCD algorithm, achieving up to a 50-fold reduction in runtime under Setting 2 with a smaller group size of $d = 3$. The most pronounced improvement is observed when the correlation among the predictors is moderate ($\rho = 0.6, \psi = 0.6$).

To better understand this behavior, consider the condition number of a positive definite matrix $\mathbf{A}$, denoted as $\kappa(\mathbf{A})$, which is the ratio of the largest eigenvalue to the smallest eigenvalue of $\mathbf{A}$. Under our simulation settings, where $\rho = \psi$, the covariance matrix can be written as

$$\boldsymbol{\Sigma} = (1 - \rho)\mathbf{I} + \rho\mathbf{1}\mathbf{1}^\top,$$

and we can show that the condition number of the expected matrix $\mathbf{X}^\top\,\mathbf{X}$ is

$$\kappa\left(\mathbb{E}[\mathbf{X}^\top\mathbf{X}]\right) = \kappa\left(\boldsymbol{\Sigma}\right) = \frac{1 + (n-1)\rho}{1 - \rho}. \tag{22}$$

This condition number increases as $\rho \to 1$, indicating that higher correlation among predictors results in a more ill-conditioned matrix. Our numerical results suggest that the condition number of the matrix $\mathbf{X}^\top\mathbf{X}$ plays a critical role in the performance of the algorithm.

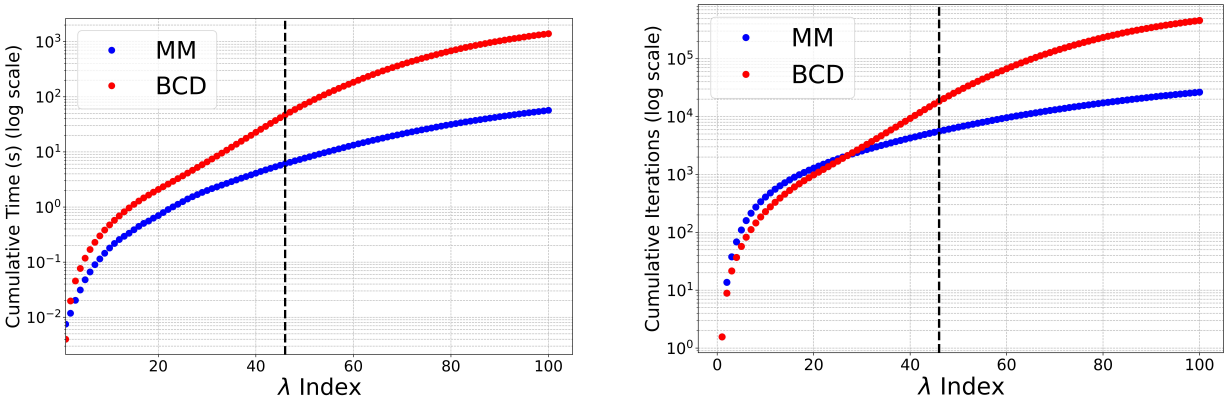
Table 1: Running Time (s) for Setting 1 and Setting 2. Means of 50 replications are reported.

			Correlation (ψ, ρ)		
Setting	Group Size	Method	(0.3, 0.3)	(0.6, 0.6)	(0.9, 0.9)
1	3	BCD	8.39×10^2	1.59×10^3	7.62×10^2
		MM	3.91×10^1	3.47×10^1	2.98×10^1
	5	BCD	9.96×10^2	1.07×10^3	4.95×10^2
		MM	5.54×10^1	5.09×10^1	4.68×10^1
2	3	BCD	1.45×10^3	2.37×10^3	1.32×10^3
		MM	6.00×10^1	4.73×10^1	3.71×10^1
	5	BCD	8.01×10^2	9.12×10^2	5.59×10^2
		MM	4.41×10^1	3.92×10^1	3.70×10^1

The dominant factor influencing the number of iterations t and runtime for both the MM algorithm and BCD algorithm is the number of non-zero groups at each value of λ_i, $i = 1, \ldots, m$. When fitting a regularization path, a disproportionate amount of time is spent at the least sparse end of the path, where λ is small. The MM algorithm increasingly outperforms the BCD algorithm in this region of the path, as the value of λ decreases. In Figure 1, we present the cumulative runtime and cumulative iterations as both algorithms progress from λ_m to λ_1 using warm starts. These values represent the total runtime and

iterations required to compute the lasso path up to a certain index. In these simulations, we maintain the settings used for Table 1 but only consider Setting 2, with group size $d = 3$ and correlation $\rho = 0.3, \psi = 0.3$. For ease of readability, the indexes $1, \ldots, m$ are reversed in the figures. This means the first λ index corresponds to the largest λ value and the starting value for both algorithms. In Figure 1, we indicate the average λ index that minimizes the prediction error $\|\mathbf{X}\widehat{\boldsymbol{\beta}}(\lambda) - \mathbf{X}\boldsymbol{\beta}\|$, representing the optimal λ.

In certain applications, computing the entire regularization path is necessary. For example, a common approach to control the False Discovery Rate (FDR) during group selection is to compute knockoff groups (Dai and Barber, 2016). This procedure requires obtaining the scores $V_j := \sup\{\lambda : \widehat{\boldsymbol{\theta}}_j(\lambda) \neq \mathbf{0}\}$, for $j = 1, \ldots, J$. These scores indicate the first time each group $\mathbf{Z}_j$ enters the group lasso regularization path. In such scenarios, where computing the entire regularization path is essential, simulations show that the MM algorithm significantly outperforms the BCD algorithm.

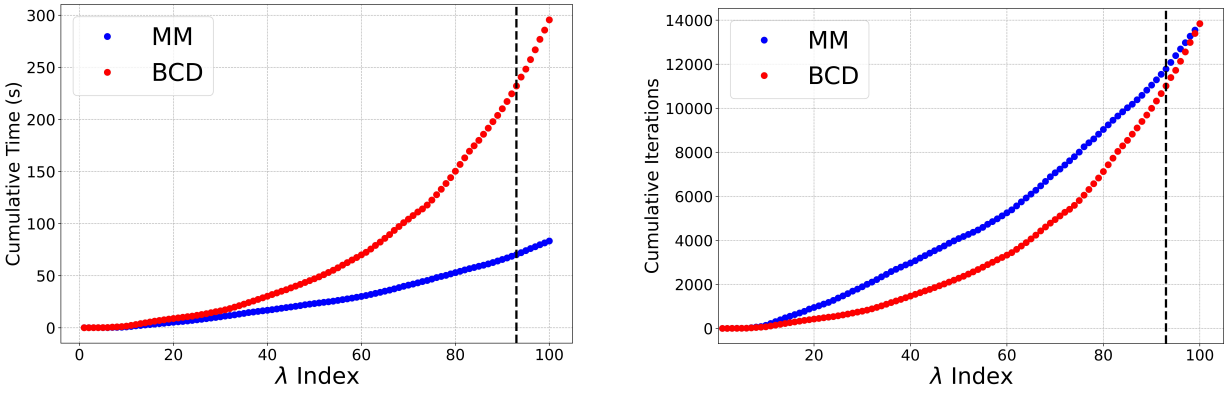


(a) Cumulative Time(s)

(b) Cumulative Iterations

Figure 1: Cumulative runtime and iterations t until convergence for the BCD and MM algorithms, as λ decreases in a regularization path. The mean values from 50 simulations are reported, with the vertical dotted black line indicating the mean index at which prediction error is minimized.

4.2 Real Data



(a) Cumulative Time(s)

(b) Cumulative Iterations

Figure 2: Cumulative runtime and number of iterations t until convergence for the BCD and MM algorithms, as λ decreases along the regularization path for the supermarket foot traffic dataset.

We now consider a real dataset containing anonymized information on foot traffic and sales volumes for a major Chinese supermarket (Wang, 2009; Thompson and Vahid, 2024).

The task is to model foot traffic using the sales volumes of different products. We fit a sparse model using group lasso to identify a subset of products that effectively predict foot traffic. Sales volumes are available for 6,398 products. In this dataset, two groups are created for each product: a linear group containing the linear term and a nonlinear group containing a four-term cubic polynomial. This results in $p = 31,990$ total predictors and $J = 12,796$ groups (6,398 single predictor groups ($d = 1$) plus 6,398 groups with $d = 4$).

The sample consists of $n = 464$ days. We randomly hold out 10% of the data as a testing set to validate the model. The MM algorithm and the BCD algorithm are run on the remaining 90% of the dataset for a grid of λ values, where $\lambda_1 = \omega\lambda_m$ with $\omega = 0.1$. For further details on the construction of this grid, refer to supplementary Section H. In Figure 2, we show the cumulative runtime and cumulative number of iterations as both algorithms progress from λ_m to λ_1. In the figure, the vertical dashed line indicates the

λ index that minimizes the test error $\|\boldsymbol{y}_{\text{test}} - \mathbf{X}_{\text{test}}\widehat{\boldsymbol{\beta}}(\lambda)\|$. The results in Figure 2 show a marginal difference in the number of iterations required to compute the entire regularization path, while exhibiting a significant improvement in the cumulative runtime of the MM algorithm compared to the BCD algorithm. This indicates that the MM algorithm has a cost-per-iteration advantage over the BCD algorithm for this particular dataset, despite no significant differences in the number of iterations.

5 Conclusions

In this paper, we investigated an efficient method for obtaining the group lasso solution, which involves minimizing a convex objective defined over block parameters $\boldsymbol{\theta} = [\boldsymbol{\theta}_1^\top, \ldots, \boldsymbol{\theta}_J^\top]^\top \in \mathbb{R}^p$. While the popular block coordinate descent (BCD) algorithm offers a straightforward updating rule under the assumption of orthogonalized groups, it can suffer from slow convergence, particularly when the regularization parameter λ is small or when the data matrix $\mathbf{X}$ has a high condition number.

We proposed an alternative algorithm based on the majorization-minimization (MM) principle to address these limitations. This approach minimizes an objective defined over (low-dimensional) auxiliary variables $\boldsymbol{\gamma} = [\gamma_1, \ldots, \gamma_J]^\top \in \mathbb{R}_+^J$, rather than the (high-dimensional) original block coefficients. We established a clear relationship between the auxiliary variables $\boldsymbol{\gamma}$ and the original variables $\boldsymbol{\theta}$. Through extensive numerical studies, we demonstrated that the MM algorithm consistently converges faster than the BCD algorithm, both on simulated datasets and a real-world dataset.

Although the experiments in Section 4 show that the MM algorithm offers significant advantages over the BCD algorithm, the MM algorithm may less suitable if many groups become active early in the solution path, because the conjugate gradient runtime depends on the number of active features $\sum_{j \in \mathcal{A}} p_j$.

SUPPLEMENTARY MATERIAL

mmgl_jcgs_suppl.pdf: A PDF file containing notations, proofs of theoretical results, additional numerical studies, and pseudocode. (.pdf file)

code: Repository containing Julia code for implementing the MM and CD algorithms described in the article. The repository also includes the datasets used as examples in the article and the code to generate simulation results. (GNU zipped tar file)

References

Breheny, P. and Huang, J. (2015). Group descent algorithms for nonconvex penalized linear and logistic regression models with grouped predictors. *Statistics and Computing*, 25:173–187.

Dai, R. and Barber, R. (2016). The knockoff filter for fdr control in group-sparse and multitask regression. In *International Conference on Machine Learning*, pages 1851–1859. PMLR.

Golub, G. H. and Van Loan, C. F. (2013). *Matrix computations*. JHU press.

Helwig, N. E. (2025). Versatile descent algorithms for group regularization and variable selection in generalized linear models. *Journal of Computational and Graphical Statistics*, 34(1):239–252.

Hunter, D. R. and Lange, K. (2004). A tutorial on mm algorithms. *The American Statistician*, 58(1):30–37.

Lange, K. (2016). *MM optimization algorithms*. SIAM.

Meier, L., Van De Geer, S., and Bühlmann, P. (2008). The group lasso for logistic regression. *Journal of the Royal Statistical Society Series B: Statistical Methodology*, 70(1):53–71.

Mishkin, A. and Pilanci, M. (2022). The solution path of the group lasso. In *OPT 2022: Optimization for Machine Learning (NeurIPS 2022 Workshop)*.

Qin, Z., Scheinberg, K., and Goldfarb, D. (2013). Efficient block-coordinate descent algorithms for the group lasso. *Mathematical Programming Computation*, 5(2):143–169.

Simon, N. and Tibshirani, R. (2012). Standardization and the group lasso penalty. *Statistica Sinica*, 22(3):983.

Subramanian, A., Tamayo, P., Mootha, V. K., Mukherjee, S., Ebert, B. L., Gillette, M. A., Paulovich, A., Pomeroy, S. L., Golub, T. R., Lander, E. S., et al. (2005). Gene set enrichment analysis: a knowledge-based approach for interpreting genome-wide expression profiles. *Proceedings of the National Academy of Sciences*, 102(43):15545–15550.

Thompson, R. and Vahid, F. (2024). Group selection and shrinkage: Structured sparsity for semiparametric additive models. *Journal of Computational and Graphical Statistics*, pages 1–12.

Tibshirani, R. (1996). Regression shrinkage and selection via the lasso. *Journal of the Royal Statistical Society: Series B (Methodological)*, 58(1):267–288.

Tibshirani, R. J. (2013). The lasso problem and uniqueness. *Electronic Journal of Statistics*, 7(none):1456 – 1490.

Tseng, P. (2001). Convergence of a block coordinate descent method for nondifferentiable minimization. *Journal of Optimization Theory and Applications*, 109:475–494.

Wang, H. (2009). Forward regression for ultra-high dimensional variable screening. *Journal of the American Statistical Association*, 104(488):1512–1524.

Wu, T. T. and Lange, K. (2008). Coordinate descent algorithms for lasso penalized regression. *The Annals of Applied Statistics*.

Yang, J. and Hastie, T. (2024). A fast and scalable pathwise-solver for group lasso and elastic net penalized regression via block-coordinate descent. *arXiv preprint arXiv:2405.08631*.

Yang, Y. and Zou, H. (2015). A fast unified algorithm for solving group-lasso penalize learning problems. *Statistics and Computing*, 25(6):1129–1141.

Yuan, M. and Lin, Y. (2006). Model selection and estimation in regression with grouped variables. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 68(1):49–67.

Zhou, H., Hu, L., Zhou, J., and Lange, K. (2019). Mm algorithms for variance components models. *Journal of Computational and Graphical Statistics*, 28(2):350–361.

Supplementary Materials for "A Fast MM Algorithm for Group Lasso"

A Sphering in Rank-Deficient Case

If $\mathbf{X}_j$ is not full rank, we can use the thin QR factorization, $\mathbf{X}_j = \mathbf{Z}'_j\mathbf{R}'_j$, where $\mathbf{Z}'_j \in \mathbb{R}^{n\times r_j}$ is a full-rank matrix with r_j orthogonal columns and $\mathbf{R}'_j \in \mathbb{R}^{r_j\times p_j}$, with $r_j = \text{rank}(\mathbf{Z}'_j) < p_j$. We note that in this case there are infinitely many $\widehat{\boldsymbol{\beta}}_j$ that satisfy $\widehat{\boldsymbol{\theta}}_j = \mathbf{R}_j\widehat{\boldsymbol{\beta}}_j$. This implies that the solution to the original problem is not unique, which is not necessarily due to the sphering process alone. Even without sphering, the group lasso solution (minimizer of (3) where $\mathbf{Z}_j$ is replaced with $\mathbf{X}_j$) may not be unique when $\mathbf{X}_j$ is not full rank. The so-called *group general position* is a sufficient condition for uniqueness. In Section 3.4 we discuss this assumption in more detail. While the transformation $\widehat{\boldsymbol{\theta}}_j = \mathbf{R}_j\widehat{\boldsymbol{\beta}}_j$ may be underdetermined, we interpret $\widehat{\boldsymbol{\theta}}_j = \mathbf{0}$ to indicate that the corresponding group of features $\boldsymbol{\beta}_j$ is not relevant for prediction. This interpretation is motivated by the observation that the penalty in problem (3), when applying the transformation $\boldsymbol{\theta}_j = \mathbf{R}_j\boldsymbol{\beta}_j$ and utilizing (2), is equivalent to $\lambda\sqrt{p_j}\,\|\boldsymbol{\theta}_j\|_2 = \lambda\sqrt{p_j}\|\mathbf{X}_j\boldsymbol{\beta}_j\|_2$. In other words, rather than penalizing the coefficient $\boldsymbol{\beta}_j$, we penalize the mean response $\mathbf{X}_j\boldsymbol{\beta}_j$. Using this penalty is advantageous because: i) it results in a elegant closed-form block-coordinate updating formula, which we detail in Section 2, ii) it improves model selection, and iii) it aligns with the standard practice of normalizing each predictor to have a unit norm when groups consist of a single predictor. These advantages are empirically and theoretically examined in Simon and Tibshirani (2012).

B BCD Algorithm

The following code provides a simple coordinate descent algorithm for minimizing (3).

Algorithm B.1: Minimizing $f(\boldsymbol{\theta})$ via BCD (Breheny and Huang, 2015)

input: $\{\mathbf{Z}_1, \ldots, \mathbf{Z}_d\}, \boldsymbol{y}, \lambda, \epsilon$, initial $\{\boldsymbol{\theta}_j\}$ and residual $\boldsymbol{r} := \boldsymbol{y} - \sum_j \mathbf{Z}_j \boldsymbol{\theta}_j$

output: (global) minimizer $\{\widehat{\boldsymbol{\theta}}_j\}$ and updated $\boldsymbol{r} := \boldsymbol{y} - \sum_j \mathbf{Z}_j \widehat{\boldsymbol{\theta}}_j$

$\boldsymbol{\alpha} \leftarrow n\lambda \left[\sqrt{p_1}, \ldots, \sqrt{p_J}\right]$, $t \leftarrow 0$

repeat // iterate over CD cycles

  $\boldsymbol{\theta}^{\text{old}} \leftarrow \boldsymbol{\theta}$, $t \leftarrow t + 1$

  for $j = 1, \ldots, J$ **do** // cycle

    $\boldsymbol{\theta}_j^{\text{new}} \leftarrow S_{\alpha_j}\left(\mathbf{Z}_j^\top \boldsymbol{r} + \boldsymbol{\theta}_j\right)$

    if $\boldsymbol{\theta}_j \neq \boldsymbol{\theta}_j^{\text{new}}$ **then** // change in $\boldsymbol{\theta}_j$ $\Rightarrow$ update $\boldsymbol{r}$

      $\boldsymbol{r} \leftarrow \boldsymbol{r} + \mathbf{Z}_j(\boldsymbol{\theta}_j - \boldsymbol{\theta}_j^{\text{new}})$ // update $\boldsymbol{r}$

      $\boldsymbol{\theta}_j \leftarrow \boldsymbol{\theta}_j^{\text{new}}$

until $\max_j p_j^{-1} \|\boldsymbol{\theta}_j - \boldsymbol{\theta}_j^{\text{old}}\|_2 < \epsilon$

return $\{\widehat{\boldsymbol{\theta}}_j\}, \boldsymbol{r}$

The main computational cost in Algorithm B.1 (Breheny and Huang, 2015) arises from calculating the term $\mathbf{Z}_j^\top \boldsymbol{r}$ for each coordinate and the term $\mathbf{Z}_j(\boldsymbol{\theta}_j - \boldsymbol{\theta}_j^{\text{new}})$ whenever an update is made.

C MM Algorithm

The following code provides a MM algorithm for minimizing (3).

Algorithm C.1: Minimizing $f(\boldsymbol{\theta})$ via MM

input: $\{\mathbf{Z}_1, \ldots, \mathbf{Z}_J\}, \boldsymbol{y}, \lambda, \epsilon, c$

output: (global) minimizer $\{\widehat{\boldsymbol{\theta}}_j\}$, $\widehat{\boldsymbol{\gamma}}$, $\mathcal{A}$, $\mathbf{G}\boldsymbol{y}$

$\boldsymbol{\gamma} \leftarrow \mathbf{0}$, $\boldsymbol{\theta} \leftarrow \mathbf{0}$, $\mathcal{A} \leftarrow \emptyset$, $\boldsymbol{\alpha} \leftarrow n\lambda \left[\sqrt{p_1}, \ldots, \sqrt{p_J}\right]$, $\mathbf{G}\boldsymbol{y} \leftarrow \boldsymbol{y}$, $\mathbf{Z}_{\mathcal{A}} \leftarrow \mathbf{0}$, $t \leftarrow 0$

repeat // iterate over MM cycles

  $t \leftarrow t + 1$

  $\boldsymbol{\gamma}^{\text{old}} \leftarrow \boldsymbol{\gamma}$

  $\boldsymbol{\theta}^{\text{old}} \leftarrow \boldsymbol{\theta}$

  $\boldsymbol{\tau} \leftarrow \left[\|\mathbf{Z}_1^\top \mathbf{G}\boldsymbol{y}\|, \quad \ldots \quad , \|\mathbf{Z}_J^\top \mathbf{G}\boldsymbol{y}\|\right]$

  $\boldsymbol{w} \leftarrow \texttt{getWeights}(\boldsymbol{\alpha}, \boldsymbol{\tau}, \boldsymbol{\gamma})$

  for $j = 1, \ldots, J$ **do** // cycle

    $\gamma_{\text{new}} \leftarrow S_{\alpha_j w_j}\big((w_j + \gamma_j)\tau_j\big)/\alpha_j$

    if $\gamma_j \neq \gamma_{\text{new}}$ **then** // change in γ_j $\Rightarrow$ update $\mathcal{A}$

      if $\gamma_{\text{new}} = 0$ **then** // downsize $\mathcal{A}, \mathbf{Z}_{\mathcal{A}}$

        remove j and $\mathbf{Z}_j$ from $\mathcal{A}$ and $\mathbf{Z}_{\mathcal{A}}$, respectively

      else if $j \notin \mathcal{A}$ **then** // upsize $\mathcal{A}, \mathbf{Z}_{\mathcal{A}}$

        $\mathcal{A} \leftarrow \mathcal{A} \cup j$, $\mathbf{Z}_{\mathcal{A}} \leftarrow [\mathbf{Z}_{\mathcal{A}}, \mathbf{Z}_j]$

      $\gamma_j \leftarrow \gamma_{\text{new}}$

      $\boldsymbol{\theta}_j \leftarrow \gamma_j \mathbf{Z}_j^\top \mathbf{G}\boldsymbol{y}$

  Using the linear map $\mathbf{A} : \boldsymbol{x} \mapsto \boldsymbol{x} + \mathbf{Z}_{\mathcal{A}}(\boldsymbol{\gamma}_{\mathcal{A}} \odot (\mathbf{Z}_{\mathcal{A}}^\top \boldsymbol{x}))$ and initializing from $\texttt{Gy}$,

  take c steps of the conjugate gradient algorithm to update $\mathbf{G}\boldsymbol{y} \approx \mathbf{A}^{-1}\boldsymbol{y}$.

until $\max_j p_j^{-1} \|\boldsymbol{\theta}_j - \boldsymbol{\theta}_j^{\text{old}}\|_2 < \epsilon$

$\widehat{\boldsymbol{\gamma}} \leftarrow \boldsymbol{\gamma}$

$\widehat{\boldsymbol{\theta}} \leftarrow \boldsymbol{\theta}$

return $\{\widehat{\boldsymbol{\theta}}_j\}, \widehat{\boldsymbol{\gamma}}, \mathcal{A}, \mathbf{G}\boldsymbol{y}$

Algorithm C.1 takes as inputs the sphered group matrices $\mathbf{Z}_1, \mathbf{Z}_2, \ldots, \mathbf{Z}_J$, the n-dimensional

response vector $\boldsymbol{y}$, the regularization parameter λ, the tolerance parameter ϵ, and the number of conjugate gradient steps c. In Step 7, the algorithm calls the function `getWeights` to optimally assign weights, as detailed in supplementary Section G. Once the termination condition in Step 19 is satisfied, the algorithm outputs the minimizer $\widehat{\boldsymbol{\theta}}$ for (3), the minimizer $\widehat{\boldsymbol{\gamma}}$ for (11), the activated set $\mathcal{A}$, and the vector $\mathbf{G}(\widehat{\boldsymbol{\gamma}})\boldsymbol{y}$, which is equivalent to the residual vector $\boldsymbol{r}$. This equivalence is proven in supplementary Section D.

D Proof of KKT Optimality Conditions

Proof of Theorem 3.1. First, note that $\mathbf{A}^\top(\mathbf{A}\mathbf{A}^\top + \mathbf{I}) = (\mathbf{A}^\top\mathbf{A} + \mathbf{I})\mathbf{A}^\top$ implies the push-through identity given by

$$(\mathbf{A}^\top\mathbf{A} + \mathbf{I})^{-1}\mathbf{A}^\top = \mathbf{A}^\top(\mathbf{A}\mathbf{A}^\top + \mathbf{I})^{-1}.$$

Since $\boldsymbol{\theta} = \boldsymbol{\Gamma}^2\mathbf{Z}^\top\mathbf{G}(\boldsymbol{\gamma})\boldsymbol{y}$ and

$$\mathbf{G}(\boldsymbol{\gamma}) = \left(\mathbf{I} + \mathbf{Z}\boldsymbol{\Gamma}^2\mathbf{Z}^\top\right)^{-1},$$

the push-through and Woodbury identities (Henderson and Searle, 1981) yields the following:

$$\begin{aligned}
\mathbf{G}(\boldsymbol{\gamma})\boldsymbol{y} &= (\mathbf{I} - \mathbf{Z}\boldsymbol{\Gamma}(\mathbf{I} + \boldsymbol{\Gamma}\mathbf{Z}^\top\mathbf{Z}\boldsymbol{\Gamma})^{-1}\boldsymbol{\Gamma}\mathbf{Z}^\top)\boldsymbol{y} \\
&= \boldsymbol{y} - \mathbf{Z}\boldsymbol{\Gamma}(\mathbf{I} + \boldsymbol{\Gamma}\mathbf{Z}^\top\mathbf{Z}\boldsymbol{\Gamma})^{-1}\boldsymbol{\Gamma}\mathbf{Z}^\top\boldsymbol{y} \\
&= \boldsymbol{y} - \mathbf{Z}\boldsymbol{\Gamma}^2\mathbf{Z}^\top(\mathbf{I} + \mathbf{Z}\boldsymbol{\Gamma}^2\mathbf{Z}^\top)^{-1}\boldsymbol{y} = \boldsymbol{y} - \mathbf{Z}\boldsymbol{\theta}. \qquad \text{(D.1)}
\end{aligned}$$

Assume that (8) holds. Then, $\mathbf{Z}_j^\top(\boldsymbol{y} - \mathbf{Z}\boldsymbol{\theta}) = \mathbf{Z}_j^\top\mathbf{G}(\boldsymbol{\gamma})\boldsymbol{y} = \alpha_j\boldsymbol{\theta}_j/\|\boldsymbol{\theta}_j\|_2$ for $j \in \mathcal{A}$. Taking the norm on both sides and combining (D.1) proves that (8)⇒(10).

Next, assume (10) holds. Then, $\|\boldsymbol{\theta}_j\|_2 = \gamma_j\|\mathbf{Z}_j^\top\mathbf{G}(\boldsymbol{\gamma})\boldsymbol{y}\|_2 = \gamma_j\alpha_j$ when $j \in \mathcal{A}$ which implies $\gamma_j = \|\boldsymbol{\theta}_j\|/\alpha_j$. Substituting $\gamma_j = \|\boldsymbol{\theta}_j\|/\alpha_j$ into $\boldsymbol{\theta}_j = \gamma_j\mathbf{Z}_j^\top\mathbf{G}(\boldsymbol{\gamma})\boldsymbol{y}$ and considering (D.1) we have (10)⇒(8).

With (8)$\Leftrightarrow$(10) established, we have $\gamma_j = \|\boldsymbol{\theta}_j\|/\alpha_j$ for $j \in \mathcal{A}$ and $\mathbf{0} = \gamma_j \mathbf{Z}_j^\top \mathbf{G}(\boldsymbol{\gamma})\boldsymbol{y} \Rightarrow 0 = \gamma_j \|\mathbf{Z}_j^\top \mathbf{G}(\boldsymbol{\gamma})\boldsymbol{y}\|$ for $j \notin \mathcal{A}$. □

Proof of Lemma 3.1. Consider the partial derivatives of g given by,

$$\frac{\partial g(\boldsymbol{\gamma})}{\partial \gamma_j} = -\left\|\mathbf{Z}_j^\top \mathbf{G}(\boldsymbol{\gamma})\boldsymbol{y}\right\|_2^2 + \alpha_j^2, \quad j = 1, \ldots, J,$$

where we use $\frac{\partial \mathbf{G}(\boldsymbol{\gamma})}{\partial \gamma_j} = -\mathbf{G}(\boldsymbol{\gamma})\mathbf{Z}_j\mathbf{Z}_j^\top \mathbf{G}(\boldsymbol{\gamma})$. The KKT necessary conditions for the global minimizer $\widehat{\boldsymbol{\gamma}}$ (since g is convex) require each component of the gradient vector to satisfy,

$$\left.\frac{\partial g(\boldsymbol{\gamma})}{\partial \gamma_j}\right|_{\gamma_j = \widehat{\gamma}_j} \in \begin{cases} \{0\}, & \widehat{\gamma}_j > 0 \\ [0, \infty), & \widehat{\gamma}_j = 0 \end{cases}, \quad j = 1, \ldots, J.$$

This implies equations (13) and (14). □

Proof of Corollary 3.1. Since $\widehat{\boldsymbol{\gamma}}$ satisfies equations (13) and (14), Theorem 3.1 implies that $\widehat{\boldsymbol{\theta}} = \widehat{\boldsymbol{\Gamma}}^2 \mathbf{Z}^\top \mathbf{G}(\widehat{\boldsymbol{\gamma}})\boldsymbol{y}$ is a minimizer of the original group lasso problem (3). Moreover, from Theorem 3.1 we have that $\widehat{\gamma}_j = \|\widehat{\boldsymbol{\theta}}_j\|/\alpha_j$ for $j \in \mathcal{A}$ and $\widehat{\gamma}_j \|\mathbf{Z}_j^\top \mathbf{G}(\widehat{\boldsymbol{\gamma}})\boldsymbol{y}\|_2 = 0$ for $j \notin \mathcal{A}$. This implies that either $\widehat{\gamma}_j = 0$ or $\|\mathbf{Z}_j^\top \mathbf{G}(\widehat{\boldsymbol{\gamma}})\boldsymbol{y}\|_2 = 0$ when $j \notin \mathcal{A}$. We now argue $\|\mathbf{Z}_j^\top \mathbf{G}(\widehat{\boldsymbol{\gamma}})\boldsymbol{y}\|_2 = 0 \Rightarrow \widehat{\gamma}_j = 0$ which further implies that $\widehat{\gamma}_j = 0$ for all $j \notin \mathcal{A}$. This is true because the inequality in the KKT condition $\|\mathbf{Z}_j^\top \mathbf{G}(\widehat{\boldsymbol{\gamma}})\boldsymbol{y}\| \leq \alpha_j$ is strict when $\|\mathbf{Z}_j^\top \mathbf{G}(\widehat{\boldsymbol{\gamma}})\boldsymbol{y}\|_2 = 0$ as $\alpha_j > 0$, and therefore the partial derivative $\frac{\partial g(\boldsymbol{\gamma})}{\partial \gamma_j}$ is strictly positive. The stationarity of a KKT point then implies that we must have $\widehat{\gamma}_j = 0$. Therefore, $\mathcal{A} \equiv \{j : \widehat{\gamma}_j > 0\}$.

Therefore, the conditions (10) are satisfied, and the vector $\widehat{\boldsymbol{\theta}} = \widehat{\boldsymbol{\Gamma}}^2 \mathbf{Z}^\top \mathbf{G}(\widehat{\boldsymbol{\gamma}})\boldsymbol{y}$ meets the group lasso KKT conditions. These equations also provide an alternative representation of the group lasso KKT conditions (8) in terms of the auxiliary variables $\boldsymbol{\gamma}$ alone. □

E Proof of Group Lasso Uniqueness

We now prove Lemma 3.2, which establishes the uniqueness of the problem (12) under the Assumption 3.1.

Before we delve into the proof, we introduce some useful notation. Suppose $\boldsymbol{v} = \left[\boldsymbol{v}_1^\top, \ldots, \boldsymbol{v}_J^\top\right]^\top \in \mathbb{R}^p$, where $\boldsymbol{v}_j \in \mathbb{R}^{p_j}$ and $\sum_{j=1}^J p_j = p$. If $\mathcal{D}$ is a subset of J groups, i.e., $\mathcal{D} \subseteq \{1, \ldots, J\}$, then $\boldsymbol{v}_\mathcal{D}$ is the vector of length $\sum_{j\in\mathcal{D}} p_j$ formed by restricting $\boldsymbol{v}$ to the group coordinates indexed by $\mathcal{D}$. Specifically, $\boldsymbol{v}_\mathcal{D}$ consists of the components of $\boldsymbol{v}_j \in \mathbb{R}^{p_j}$ for each $j \in \mathcal{D}$, ordered according to the indices in $\mathcal{D}$.

Similarly, if $\mathbf{V} \in \mathbb{R}^{n\times p}$ is a real matrix, then $\mathbf{V}_\mathcal{D} \in \mathbb{R}^{n\times\sum_{j\in\mathcal{D}} p_j}$ denotes the submatrix of $\mathbf{V}$ formed by selecting the columns corresponding to the group coordinates in $\mathcal{D}$. Additionally, $\mathbf{V}_{\mathcal{D},\mathcal{D}}$ represents the principal submatrix of $\mathbf{V}$, obtained by selecting both the rows and columns indexed by the group coordinates in $\mathcal{D}$.

Proof of Lemma 3.2. We reformulate the inequality-constrained problem in (12) as

$$\min\ g(\boldsymbol{\gamma}) \quad \text{subject to}\ \ c_j(\boldsymbol{\gamma}) \le 0, \quad j = 1, \ldots, J,$$

where $c_j(\boldsymbol{\gamma}) = -\gamma_j$ for $j = 1, \ldots, J$. The corresponding Lagrangian function is given by

$$\mathcal{L}(\boldsymbol{\gamma}, \boldsymbol{\mu}) = g(\boldsymbol{\gamma}) - \sum_{j=1}^J \mu_j \gamma_j.$$

Assume that $\widehat{\boldsymbol{\gamma}}$ satisfies the first-order KKT conditions as stated in Lemma (3.1). Furthermore, suppose there exists a corresponding vector of Lagrange multipliers $\widehat{\boldsymbol{\mu}}$ such that, for all $\boldsymbol{z} \neq \mathbf{0}$ satisfying $\nabla c_j(\widehat{\boldsymbol{\gamma}})^\top \boldsymbol{z} = 0$ for every $j \in \mathcal{A}(\boldsymbol{\gamma}^*)^c = \{j : c_j(\boldsymbol{\gamma}^*) = 0\}$, we have

$$\boldsymbol{z}^\top \nabla^2_{\boldsymbol{\gamma}\boldsymbol{\gamma}} \mathcal{L}(\widehat{\boldsymbol{\gamma}}, \widehat{\boldsymbol{\mu}}) \boldsymbol{z} > 0. \tag{E.1}$$

Under these conditions, $\boldsymbol{\gamma}^*$ is a strict local minimizer (see Chapter 3.1 in Bertsekas (2014)).

Since the constraints are linear, $\nabla^2_{\boldsymbol{\gamma}\boldsymbol{\gamma}} \mathcal{L}(\boldsymbol{\gamma}, \boldsymbol{\mu})$ simplifies to the Hessian matrix $\mathbf{H}(\boldsymbol{\gamma})$, eliminating its dependence on $\widehat{\boldsymbol{\mu}}$. The entries of $\mathbf{H}(\boldsymbol{\gamma})$ are given by $[\mathbf{H}(\boldsymbol{\gamma})]_{jk} := \frac{\partial^2 g(\boldsymbol{\gamma})}{\partial\gamma_j \partial\gamma_k}$. For each $j, k \in 1, \ldots, J$, we have

$$\frac{\partial^2 g(\boldsymbol{\gamma})}{\partial\gamma_j \partial\gamma_k} = \boldsymbol{y}^\top \frac{\partial^2 \mathbf{G}(\boldsymbol{\gamma})}{\partial\gamma_j \partial\gamma_k} \boldsymbol{y}.$$

To find $\frac{\partial^2 \mathbf{G}(\boldsymbol{\gamma})}{\partial\gamma_j \partial\gamma_k}$, we begin with the first-order derivatives:

$$\frac{\partial \mathbf{G}(\boldsymbol{\gamma})}{\partial\gamma_j} = -\mathbf{G}(\boldsymbol{\gamma}) \frac{\partial \mathbf{G}(\boldsymbol{\gamma})^{-1}}{\partial\gamma_j} \mathbf{G}(\boldsymbol{\gamma}) = -\mathbf{G}(\boldsymbol{\gamma}) \mathbf{Z}_j \mathbf{Z}_j^\top \mathbf{G}(\boldsymbol{\gamma}). \tag{E.2}$$

Thus, the second-order derivatives are given by

$$\frac{\partial^2 \mathbf{G}(\boldsymbol{\gamma})}{\partial \gamma_j \partial \gamma_k} = -\mathbf{G}(\boldsymbol{\gamma})\mathbf{Z}_j\mathbf{Z}_j^\top \frac{\partial \mathbf{G}(\boldsymbol{\gamma})}{\partial \gamma_k} - \frac{\partial \mathbf{G}(\boldsymbol{\gamma})}{\partial \gamma_k}\mathbf{Z}_j\mathbf{Z}_j^\top \mathbf{G}(\boldsymbol{\gamma}).$$

Using (E.2), we define $\mathbf{A}_{jk}(\boldsymbol{\gamma}) := \mathbf{G}(\boldsymbol{\gamma})\mathbf{Z}_j\mathbf{Z}_j^\top \mathbf{G}(\boldsymbol{\gamma})\mathbf{Z}_k\mathbf{Z}_k^\top \mathbf{G}(\boldsymbol{\gamma})$, so that

$$\frac{\partial^2 \mathbf{G}(\boldsymbol{\gamma})}{\partial \gamma_j \partial \gamma_k} = \mathbf{A}_{jk}(\boldsymbol{\gamma}) + [\mathbf{A}_{jk}(\boldsymbol{\gamma})]^\top,$$

and therefore,

$$\boldsymbol{y}^\top \frac{\partial^2 \mathbf{G}(\boldsymbol{\gamma})}{\partial \gamma_j \partial \gamma_k}\boldsymbol{y} = \boldsymbol{b}_j(\boldsymbol{\gamma})^\top \mathbf{G}(\boldsymbol{\gamma})\boldsymbol{b}_k(\boldsymbol{\gamma}) + \boldsymbol{b}_k(\boldsymbol{\gamma})^\top \mathbf{G}(\boldsymbol{\gamma})\boldsymbol{b}_j(\boldsymbol{\gamma}),$$

where $\boldsymbol{b}_j(\boldsymbol{\gamma}) := \mathbf{Z}_j\mathbf{Z}_j^\top \mathbf{G}(\boldsymbol{\gamma})\boldsymbol{y}$ for each j. Consequently, we can write

$$\frac{\partial^2 g(\boldsymbol{\gamma})}{\partial \gamma_j \partial \gamma_k} = 2\boldsymbol{b}_j(\boldsymbol{\gamma})^\top \mathbf{G}(\boldsymbol{\gamma})\boldsymbol{b}_k(\boldsymbol{\gamma}).$$

Letting $\mathbf{B}(\boldsymbol{\gamma})$ be the matrix whose j-th column is $\boldsymbol{b}_j(\boldsymbol{\gamma})$, the Hessian of $g(\boldsymbol{\gamma})$ is

$$\mathbf{H}(\boldsymbol{\gamma}) = 2\mathbf{B}(\boldsymbol{\gamma})^\top \mathbf{G}(\boldsymbol{\gamma})\mathbf{B}(\boldsymbol{\gamma}).$$

This expression shows that $\mathbf{H}(\boldsymbol{\gamma})$ is positive semi-definite for all $\boldsymbol{\gamma} \in \mathbb{R}_+^J$, making $g(\boldsymbol{\gamma})$ convex on $\mathbb{R}_+^J$. Therefore, the second-order condition (E.1) provides a criterion for the uniqueness of $\widehat{\boldsymbol{\gamma}}$. Noting that $\nabla c_j(\widehat{\boldsymbol{\gamma}}) = -\boldsymbol{e}_j$, where $\boldsymbol{e}_j$ is the unit vector with a 1 in the j-th position and 0's elsewhere, we have that

$$\{\boldsymbol{z} \in \mathbb{R}^J : \boldsymbol{z}^\top \nabla c_j(\widehat{\boldsymbol{\gamma}}) = 0, \forall j \in \mathcal{A}(\widehat{\boldsymbol{\gamma}})^c\} = \{\boldsymbol{z} \in \mathbb{R}^J : \boldsymbol{z}_{\mathcal{A}(\widehat{\boldsymbol{\gamma}})^c} = \mathbf{0}\}.$$

Consequently, (E.1) reduces to verifying that the reduced Hessian $\mathbf{H}_{\mathcal{A},\mathcal{A}}(\widehat{\boldsymbol{\gamma}})$ is positive definite. In summary, if $\mathbf{H}_{\mathcal{A},\mathcal{A}}(\widehat{\boldsymbol{\gamma}})$ is positive definite, then $\widehat{\boldsymbol{\gamma}}$ must be an isolated local minimizer and hence unique.

We now use the contradiction argument from Proposition 12 in Mishkin and Pilanci (2022) to show that positive definiteness holds under Assumption 3.1 (group general position). Suppose $\mathbf{H}_{\mathcal{A},\mathcal{A}}(\widehat{\boldsymbol{\gamma}})$ is not positive definite. Then, as $\mathbf{G}(\widehat{\boldsymbol{\gamma}})$ is positive definite, the

columns of the matrix $\mathbf{B}(\widehat{\boldsymbol{\gamma}})_{\mathcal{A}}$ must be linearly dependent and there exist $s_i \in \{+1, -1\}$ and $\delta_i \geq 0$ such that

$$s_j \mathbf{b}_j(\widehat{\boldsymbol{\gamma}}) = \sum_{i \in \mathcal{A} \setminus j} \delta_i s_i \mathbf{b}_j(\widehat{\boldsymbol{\gamma}})$$

$$\implies s_j \mathbf{Z}_j \mathbf{Z}_j^\top \mathbf{G}(\widehat{\boldsymbol{\gamma}}) \boldsymbol{y} = \sum_{i \in \mathcal{A} \setminus j} \delta_i s_i \mathbf{Z}_i \mathbf{Z}_i^\top \mathbf{G}(\widehat{\boldsymbol{\gamma}}) \boldsymbol{y}$$

$$\implies \mathbf{Z}_j \mathbf{Z}_j^\top \mathbf{G}(\widehat{\boldsymbol{\gamma}}) \boldsymbol{y} = \sum_{i \in \mathcal{A} \setminus j} \delta_i s_j s_i \mathbf{Z}_i \mathbf{Z}_i^\top \mathbf{G}(\widehat{\boldsymbol{\gamma}}) \boldsymbol{y}.$$

Taking the inner product of both sides with the residual $\mathbf{G}(\widehat{\boldsymbol{\gamma}})\boldsymbol{y}$, and using (13) along with the definition $\alpha_j = n\lambda\sqrt{p_j}$,

$$\implies p_j n^2 \lambda^2 = \sum_{i \in \mathcal{A} \setminus j} \delta_i s_j s_i p_i n^2 \lambda^2$$

$$\implies 1 = \sum_{i \in \mathcal{A} \setminus j} \delta_i s_j s_i \frac{p_i}{p_j}.$$

Thus, we deduce that

$$\mathbf{Z}_j \left(\mathbf{Z}_j^\top \mathbf{G}(\widehat{\boldsymbol{\gamma}}) \boldsymbol{y} \right) = \sum_{i \in \mathcal{A} \setminus j} \delta_i s_j s_i \mathbf{Z}_i \left(\mathbf{Z}_i^\top \mathbf{G}(\widehat{\boldsymbol{\gamma}}) \boldsymbol{y} \right)$$

$$\implies p_j^{-1/2} \mathbf{Z}_j \left(\frac{\mathbf{Z}_j^\top \mathbf{G}(\widehat{\boldsymbol{\gamma}}) \boldsymbol{y}}{\alpha_j} \right) = \sum_{i \in \mathcal{A} \setminus j} \beta_i \, p_i^{-1/2} \mathbf{Z}_i \left(\frac{\mathbf{Z}_i^\top \mathbf{G}(\widehat{\boldsymbol{\gamma}}) \boldsymbol{y}}{\alpha_i} \right)$$

where $\beta_i := \delta_i s_j s_i \frac{p_i}{p_j}$ for $i \in \mathcal{A} \setminus j$ and $\sum_{i \in \mathcal{A} \setminus j} \beta_i = 1$. Now, suppose that $|\mathcal{A}| > n + 1$. Then, $\{\mathbf{Z}_i \left(\mathbf{Z}_i^\top \mathbf{G}(\widehat{\boldsymbol{\gamma}}) \boldsymbol{y} \right) : i \in \mathcal{A} \setminus j\}$ are linearly dependent and, by eliminating dependent vectors $\mathbf{Z}_i \left(\mathbf{Z}_i^\top \mathbf{G}(\widehat{\boldsymbol{\gamma}}) \boldsymbol{y} \right)$, we can repeat the above proof with a subset $\mathcal{A}'$ of at most $n + 1$ blocks. The last equation implies the existence of unit vectors $\boldsymbol{v}_i$ which contradicts the group general position assumption. This completes the proof. □

F Proof of Convergence

We now present three technical lemmas required to prove our main convergence result. The following results and corresponding proofs are adapted from Zhou et al. (2019) with some

modifications.

Lemma F.1 (Coercive Property)**.** *The function $g(\boldsymbol{\gamma})$ is coercive in the sense that the sub-level set $R_c := \{\boldsymbol{\gamma} \in \mathbb{R}_+^J : g(\boldsymbol{\gamma}) \leq c\}$ is compact for every c.*

Proof. To prove the coercive property, it is sufficient to show that $g(\mathbf{0}) = m$ for some $m \in \mathbb{R}$ and $g(\boldsymbol{\gamma}) \uparrow \infty$ as $\|\boldsymbol{\gamma}\|_\infty \uparrow \infty$. First, we show that $g(\boldsymbol{\gamma})$ is bounded from below for $\boldsymbol{\gamma} \in \mathbb{R}_+^J$. Assume $\|\boldsymbol{y}\|_2 > 0$. Then,

$$\boldsymbol{y}^\top \mathbf{G}(\boldsymbol{\gamma})\boldsymbol{y} + (\boldsymbol{\alpha} \odot \boldsymbol{\alpha})^\top \boldsymbol{\gamma} > 0$$

since $\mathbf{0} \prec \mathbf{G}(\boldsymbol{\gamma})$ and $\boldsymbol{y}^\top \mathbf{G}(\mathbf{0})\boldsymbol{y} = \|\boldsymbol{y}\| > 0$. Given that $\alpha_j > 0$ for $j = 1, \ldots, J$, we have

$$g(\boldsymbol{\gamma}) \geq (\boldsymbol{\alpha} \odot \boldsymbol{\alpha})^\top \boldsymbol{\gamma} \geq \|\boldsymbol{\gamma}\|_\infty \max_j \alpha_j^2.$$

Thus, $g(\boldsymbol{\gamma}) \uparrow \infty$, as $\|\boldsymbol{\gamma}\|_\infty \uparrow \infty$. □

Lemma F.2 (Lemma S.2. in Zhou et al. (2019))**.** *If $g(\widehat{M}(\boldsymbol{\gamma}^*)) = g(\boldsymbol{\gamma}^*)$, then $\boldsymbol{\gamma}^*$ is a fixed point of $\widehat{M}$ in the sense that $\widehat{M}(\boldsymbol{\gamma}^*) = \boldsymbol{\gamma}^*$. Furthermore, for each component $\gamma_j^* > 0$, we have $\frac{\partial g(\boldsymbol{\gamma})}{\partial \gamma_j}\Big|_{\gamma_j = \gamma_j^*} = 0$.*

Proof. Using conditions (15) we have,

$$g(\widehat{M}(\boldsymbol{\gamma}^*)) \overset{\text{majorizing property}}{\leq} h_{\widehat{\boldsymbol{w}}}(\widehat{M}(\boldsymbol{\gamma}^*), \boldsymbol{\gamma}^*) \overset{\text{minimization of MM map}}{\leq} h_{\widehat{\boldsymbol{w}}}(\boldsymbol{\gamma}^*, \boldsymbol{\gamma}^*) = g(\boldsymbol{\gamma}^*).$$

Since $g(\boldsymbol{\gamma}^*) = g(\widehat{M}(\boldsymbol{\gamma}^*))$, equality must hold above. Therefore,

$$h_{\widehat{\boldsymbol{w}}}(\widehat{M}(\boldsymbol{\gamma}^*), \boldsymbol{\gamma}^*) = h_{w^*}(\boldsymbol{\gamma}^*, \boldsymbol{\gamma}^*).$$

As $h_{\widehat{\boldsymbol{w}}}(\boldsymbol{\gamma}, \boldsymbol{\gamma}^*)$ has a unique minimum we have $\widehat{M}(\boldsymbol{\gamma}^*) = \boldsymbol{\gamma}^*$. Moreover, if $\gamma_j^* > 0$ we have

$$\frac{\partial}{\partial \gamma_j} h_{\widehat{\boldsymbol{w}}}(\boldsymbol{\gamma}, \boldsymbol{\gamma}^*)\Big|_{\gamma_j = \gamma_j^*} = 0.$$

Note that $h_{\widehat{\boldsymbol{w}}}(\boldsymbol{\gamma}, \boldsymbol{\gamma}^*) - g(\boldsymbol{\gamma}) \geq 0$ has a global minimum at $\boldsymbol{\gamma} = \boldsymbol{\gamma}^*$, so that if $\gamma_j > 0$, then

$$\frac{\partial}{\partial \gamma_j}(h_{\widehat{\boldsymbol{w}}}(\boldsymbol{\gamma}, \boldsymbol{\gamma}^*) - g(\boldsymbol{\gamma}))\Big|_{\gamma_j = \gamma_j^*} = 0$$

implies that $\frac{\partial g}{\partial \gamma_j}(\boldsymbol{\gamma})\Big|_{\gamma_j = \gamma_j^*} = 0$. □

Lemma F.3 (Lemma S.3. in Zhou et al. (2019)). *The distance between successive iterates* $\|\boldsymbol{\gamma}^{(t+1)} - \boldsymbol{\gamma}^{(t)}\|_2$ *converges to 0.*

Proof. Suppose on the contrary that $\|\boldsymbol{\gamma}^{(t+1)} - \boldsymbol{\gamma}^{(t)}\|_2$ does not converge to 0. Then one can extract a subsequence $\{t_k\}_{k\geq 1}$ such that

$$\|\boldsymbol{\gamma}^{(t_{k+1})} - \boldsymbol{\gamma}^{(t_k)}\|_2 \geq \epsilon > 0$$

for all k. Let R_0 be the compact sub-level set $\{\boldsymbol{\gamma} \in \mathbb{R}_+^J : g(\boldsymbol{\gamma}) \leq g(\boldsymbol{\gamma}^{(0)})\}$. Since the sequence $\{\boldsymbol{\gamma}^{(t_k)}\}_{k\geq 1}$ is confined to R_0, one can pass to a subsequence if necessary and assume that $\boldsymbol{\gamma}^{(t_k)}$ converges to a limit $\boldsymbol{\gamma}^*$ and that $\boldsymbol{\gamma}^{(t_k+1)}$ converges to a limit $\boldsymbol{\gamma}^{**}$. Taking limits in the relation $\boldsymbol{\gamma}^{(t_k+1)} = \widehat{M}(\boldsymbol{\gamma}^{(t_k)})$ and invoking the continuity $\widehat{M}(\boldsymbol{\gamma})$ imply that $\boldsymbol{\gamma}^{**} = \widehat{M}(\boldsymbol{\gamma}^*)$.

Because the sequence $g(\boldsymbol{\gamma}^{(t_k)})$ is monotonically non-increasing in k and bounded below on R_0, it converges to a limit g_*. Hence, the continuity of $g(\boldsymbol{\gamma})$ implies

$$g(\boldsymbol{\gamma}^*) = \lim_{k\to\infty} g(\boldsymbol{\gamma}^{(t_k)}) = g_* = \lim_{k\to\infty} g(\boldsymbol{\gamma}^{(t_k+1)}) = g(\boldsymbol{\gamma}^{**}) = g(\widehat{M}(\boldsymbol{\gamma}^*)). \tag{F.1}$$

Lemma F.2 therefore gives $\boldsymbol{\gamma}^{**} = \boldsymbol{\gamma}^* = \widehat{M}(\boldsymbol{\gamma}^*) = \boldsymbol{\gamma}^*$, contradicting the bound $\|\boldsymbol{\gamma}^* - \boldsymbol{\gamma}^{**}\|_2 \geq \epsilon$ entailed by inequality (F.1). □

With these lemmas, we are ready to prove the main convergence result.

Proof of Theorem 3.2. First, the sequence $\{\boldsymbol{\gamma}^{(t)}\}_{t\geq 0}$ is contained within the set $R_0 = \{\boldsymbol{\gamma} \in \mathbb{R}_+^J : g(\boldsymbol{\gamma}) \leq g(\boldsymbol{\gamma}^{(0)})\}$, which ensures the existence of a convergent subsequence. Consequently, the set of limit points T of $\{\boldsymbol{\gamma}^{(t)}\}_{t\geq 0}$ is non-empty. Since $\{\boldsymbol{\gamma}^{(t)}\}$ lies in the bounded and closed set R_0, and by Lemma F.3 we have $\|\boldsymbol{\gamma}^{(t+1)} - \boldsymbol{\gamma}^{(t)}\| \to 0$, it follows that T is connected (by Ostrowski's theorem, Proposition 8.2.1, Lange et al. (2010)). Furthermore, Lemma F.3 implies that for any $\boldsymbol{\gamma}^* \in T$, we have $\boldsymbol{\gamma}^* = \widehat{M}(\boldsymbol{\gamma}^*)$.

We now show that $\boldsymbol{\gamma}^* \in T$ satisfies the KKT conditions for problem (12). Specifically, $\boldsymbol{\gamma}_j^*$ must satisfy either:

1. $\boldsymbol{\gamma}_j^* > 0$ and $\frac{\partial}{\partial \boldsymbol{\gamma}_j} g(\boldsymbol{\gamma})\Big|_{\boldsymbol{\gamma}_j = \boldsymbol{\gamma}_j^*} = 0$, or

2. $\boldsymbol{\gamma}_j^* = 0$ and $\frac{\partial}{\partial \boldsymbol{\gamma}_j} g(\boldsymbol{\gamma})\Big|_{\boldsymbol{\gamma}_j = \boldsymbol{\gamma}_j^*} \geq 0$,

for $j = 1, \ldots, J$.

If $\boldsymbol{\gamma}_j^* > 0$, Lemma F.2 gives $\frac{\partial g(\boldsymbol{\gamma})}{\partial \boldsymbol{\gamma}_j}\Big|_{\boldsymbol{\gamma}_j = \boldsymbol{\gamma}_j^*} = 0$, which proves the first part. For the second part, suppose by way of contradiction that the set

$$\mathcal{B} = \left\{ j : \boldsymbol{\gamma}_j^* = 0, \frac{\partial}{\partial \boldsymbol{\gamma}_j} g(\boldsymbol{\gamma})\Big|_{\boldsymbol{\gamma}_j = \boldsymbol{\gamma}_j^*} < 0 \right\}$$

is non-empty. Note that $\frac{\partial}{\partial \boldsymbol{\gamma}_j} g(\boldsymbol{\gamma})\Big|_{\boldsymbol{\gamma}_j = \boldsymbol{\gamma}_j^*} = \alpha_j^2 - (\tau_j^*)^2$, where $\tau_j^* = \left\| \mathbf{Z}_j^\top \mathbf{G}(\boldsymbol{\gamma}^*) \right\|$. Then, for $j \in \mathcal{B}$, we have $(\alpha_j - \tau_j^*) < 0$, as $\alpha_j > 0$ and $\tau_j^* \geq 0$. If we define $j' := \arg\max_{j \in \{1,\ldots,J\}} \xi_j = (\alpha_j - \tau_j^*)^2$, then by the optimal weight allocation we have $\widehat{w}_{j'} > 0$. Now, since $\alpha_j - \tau_j^* = 0$ for $j \in \mathcal{A}(\boldsymbol{\gamma}^*)$ and $(\alpha_k - \tau_j^*) \geq 0$ for $j \in \mathcal{B}^c \cap \mathcal{A}(\boldsymbol{\gamma}^*)^c$, we conclude that $j' \in \mathcal{B}$ (see supplementary Section G for further details on weight allocation).

Therefore, there exists $j \in \mathcal{B}$ such that

$$\widehat{M}(\gamma_j^*) = \gamma_j^* \frac{\tau_j^*}{\alpha_j} + \frac{(\tau_j^* - \alpha_j)}{\alpha_j} \widehat{w}_j > \gamma_j^*,$$

where we use the MM updating formula (19) and know that $\widehat{w}_j > 0$ and $\tau_j^* - \alpha_j > 0$. This contradicts $\boldsymbol{\gamma}^* = \widehat{M}(\boldsymbol{\gamma}^*)$ and proves the second part of the KKT conditions.

Thus, the points $\boldsymbol{\gamma}^* \in T$ satisfy the KKT conditions for problem (12). If the KKT point for problem (12) is unique, the set of limit points T reduces to a single point, and $\lim_{t \to \infty} \boldsymbol{\gamma}^{(t)}$ exists and is equal to $\boldsymbol{\gamma}^*$ (Proposition 8.2.1, Lange et al. (2010)). □

G Appendix: Derivation of Optimal Weights

Recall, that to obtain the lowest possible upper bound $h_{\boldsymbol{w}}\big(M(\widetilde{\boldsymbol{\gamma}}), \widetilde{\boldsymbol{\gamma}}\big) \geq g(M(\widetilde{\boldsymbol{\gamma}}))$, we solve,

$$\begin{aligned} \widehat{\boldsymbol{w}} &= \operatorname*{argmin}_{\boldsymbol{w} \in \mathbb{R}_+^J, \mathbf{1}^\top \boldsymbol{w} = 1} \min_{\boldsymbol{\gamma} \in \mathbb{R}_+^J} h_{\boldsymbol{w}}(\boldsymbol{\gamma}, \widetilde{\boldsymbol{\gamma}}) \\ &= \operatorname*{argmin}_{\boldsymbol{w} \in \mathbb{R}_+^J, \mathbf{1}^\top \boldsymbol{w} = 1} h_{\boldsymbol{w}}\big(M(\widetilde{\boldsymbol{\gamma}}), \widetilde{\boldsymbol{\gamma}}\big). \end{aligned} \tag{G.1}$$

Here, the upper bound $h_{\boldsymbol{w}}(\boldsymbol{\gamma}, \widetilde{\boldsymbol{\gamma}})$ is given as

$$h_{\boldsymbol{w}}(\boldsymbol{\gamma}, \widetilde{\boldsymbol{\gamma}}) = \sum_j \left[\frac{\|(w_j + \widetilde{\gamma}_j)\mathbf{Z}_j^\top \mathbf{G}(\widetilde{\boldsymbol{\gamma}})\boldsymbol{y}\|^2}{w_j + \gamma_j} + w_j \left(\|\mathbf{G}(\widetilde{\boldsymbol{\gamma}})\boldsymbol{y}\|^2 - \|\mathbf{Z}_j^\top \mathbf{G}(\widetilde{\boldsymbol{\gamma}})\boldsymbol{y}\|^2 \right) + \alpha_j^2 \gamma_j \right].$$

Solving $\widehat{\boldsymbol{w}}$ is a convex problem and can be solved exactly. To proceed, we consider the unique solution $\breve{\boldsymbol{\gamma}} := \operatorname{argmin}_{\boldsymbol{\gamma} \in \mathbb{R}_+^J} h_{\boldsymbol{w}}(\boldsymbol{\gamma}, \widetilde{\boldsymbol{\gamma}})$ which, based on the update rule in (19), can be reformulated as,

$$\breve{\gamma}_j \leftarrow \frac{(w_j + \widetilde{\gamma}_j)\|\mathbf{Z}_j^\top \mathbf{G}(\widetilde{\boldsymbol{\gamma}})\boldsymbol{y}\|}{\alpha_j} \left[1 - \frac{w_j \alpha_j}{(w_j + \widetilde{\gamma}_j)\|\mathbf{Z}_j^\top \mathbf{G}(\widetilde{\boldsymbol{\gamma}})\boldsymbol{y}\|} \right]_+, \quad j = 1, \ldots, J. \tag{G.2}$$

Recalling the definitions $\tau_j := \|\mathbf{Z}_j^\top \mathbf{G}(\widetilde{\boldsymbol{\gamma}})\boldsymbol{y}\| \geq 0$ and $\alpha_j := n\lambda\sqrt{p_j} > 0$, we can simply $\breve{\boldsymbol{\gamma}}$ in (G.2) as,

$$\breve{\gamma}_j = \begin{cases} 0, & \text{if} \quad w_j(\alpha_j - \tau_j) \geq \widetilde{\gamma}_j \tau_j \\ \frac{(\widetilde{\gamma}_j + w_j)\tau_j}{\alpha_j} - w_j, & \text{if} \quad w_j(\alpha_j - \tau_j) < \widetilde{\gamma}_j \tau_j \end{cases}.$$

Substituting back into the upper bound $h_{\boldsymbol{w}}$, we have

$$\begin{aligned} h_{\boldsymbol{w}}(\breve{\boldsymbol{\gamma}}, \widetilde{\boldsymbol{\gamma}}) = & \sum_{j: w_j(\alpha_j - \tau_j) \geq \widetilde{\gamma}_j \tau_j} \frac{(w_j + \widetilde{\gamma}_j)^2}{w_j} \tau_j^2 - w_j \tau_j^2 \\ & + \sum_{j: w_j(\alpha_j - \tau_j) < \widetilde{\gamma}_j \tau_j} 2(w_j + \widetilde{\gamma}_j)\alpha_j \tau_j - w_j(\alpha_j^2 + \tau_j^2) \\ & + \sum_j w_j (\|\mathbf{G}(\widetilde{\boldsymbol{\gamma}})\boldsymbol{y}\|^2). \end{aligned}$$

When considering the feasible set $\{\boldsymbol{w} \in \mathbb{R}_+^J : \mathbf{1}^\top \boldsymbol{w} = 1\}$, the last term, $\sum_j w_j (\|\mathbf{G}(\widetilde{\boldsymbol{\gamma}})\boldsymbol{y}\|^2)$, can be ignored. The first sum represents the contribution of the bound from the variables set to zero, while the second sum represents the contribution from the activated variables. For variables that remain unactivated ($\widetilde{\gamma}_j = 0$), there is no contribution to the bound. This is true since,

$$\frac{(w_j + 0)^2}{w_j} \tau_j^2 - w_j \tau_j^2 = 0.$$

By further simplifying $h_{\boldsymbol{w}}(\breve{\boldsymbol{\gamma}}, \widetilde{\boldsymbol{\gamma}})$, the problem (G.1) can be stated as,

$$\widehat{\boldsymbol{w}} = \underset{\boldsymbol{w}\in\mathbb{R}_+^J, \mathbf{1}^\top \boldsymbol{w}=1}{\operatorname{argmin}} \sum_{j:w_j(\alpha_j-\tau_j)\geq \widetilde{\gamma}_j\tau_j} \left(\frac{(\widetilde{\gamma}_j\tau_j)^2}{w_j} + 2\widetilde{\gamma}_j\tau_j^2 \right) + \sum_{j:w_j(\alpha_j-\tau_j)<\widetilde{\gamma}_j\tau_j} \left(2\widetilde{\gamma}_j\alpha_j\tau_j - w_j(\alpha_j-\tau_j)^2\right).$$

Note that when $\alpha_j - \tau_j < \widetilde{\gamma}_j\tau_j$, the update γ_j cannot be made unactivated for any $w_j \in [0,1]$. Considering this distinction, we can simplify the problem by focusing on the contributions to the bound $h_{\boldsymbol{w}}(\breve{\boldsymbol{\gamma}}, \widetilde{\boldsymbol{\gamma}})$ where, regardless of the choice of w_j, the variables either remain activated or are set to zero. Thus, problem (G.1) simplifies to:

$$\begin{aligned}\widehat{\boldsymbol{w}} = \underset{\boldsymbol{w}\in\mathbb{R}_+^J, \mathbf{1}^\top \boldsymbol{w}=1}{\operatorname{argmin}} & \sum_{j:\alpha_j-\tau_j<\widetilde{\gamma}_j\tau_j} \left(2\widetilde{\gamma}_j\alpha_j\tau_j - w_j(\alpha_j-\tau_j)^2\right) \\ & + \sum_{j:\alpha_j-\tau_j\geq\widetilde{\gamma}_j\tau_j} \Bigg[\mathbf{1}_{\{w_j<\frac{\widetilde{\gamma}_j\tau_j}{\alpha_j-\tau_j}\}} \left(2\widetilde{\gamma}_j\alpha_j\tau_j - w_j(\alpha_j-\tau_j)^2\right) \\ & \quad + \mathbf{1}_{\{w_j\geq\frac{\widetilde{\gamma}_j\tau_j}{\alpha_j-\tau_j}\}} \left(\frac{(\widetilde{\gamma}_j\tau_j)^2}{w_j} + 2\widetilde{\gamma}_j\tau_j^2 \right) \Bigg].\end{aligned}$$

This expression shows that problem (G.1) can be rewritten as a sum of convex functions $p_j : [0,1] \to \mathbb{R}$. Specifically,

$$\widehat{\boldsymbol{w}} = \underset{\boldsymbol{w}\in\mathbb{R}_+^J, \mathbf{1}^\top \boldsymbol{w}=1}{\operatorname{argmin}} \sum_j p_j(w_j). \tag{G.3}$$

Define $c_j := \frac{\widetilde{\gamma}_j\tau_j}{\alpha_j-\tau_j}$. Then,

$$p_j(w_j) := \begin{cases} \frac{(\widetilde{\gamma}_j\tau_j)^2}{w_j} + 2\widetilde{\gamma}_j\tau_j^2, & \text{if} \quad w_j \geq c_j \,\text{and}\, c_j \in (0,1] \\ 2\widetilde{\gamma}_j\alpha_j\tau_j - w_j(\alpha_j-\tau_j)^2, & \text{else} \end{cases}.$$

If $c_j \in (0,1]$, then $\lim_{w_j\to c_j^-} p_j(w_j) = \lim_{w_j\to c_j^+} p_j(w_j) = \widetilde{\gamma}_j\tau_j(\alpha_j+\tau_j)$. Moreover, $\lim_{w_j\to c_j^-} p_j'(w_j) = \lim_{w_j\to c_j^+} p_j'(w_j) = -(\alpha_j-\tau_j)^2$. This means each $p_j(w_j)$ is a continuously differentiable function for $w_j \in [0,1]$. See Figure 1, where $w_j = c_j$ is a natural

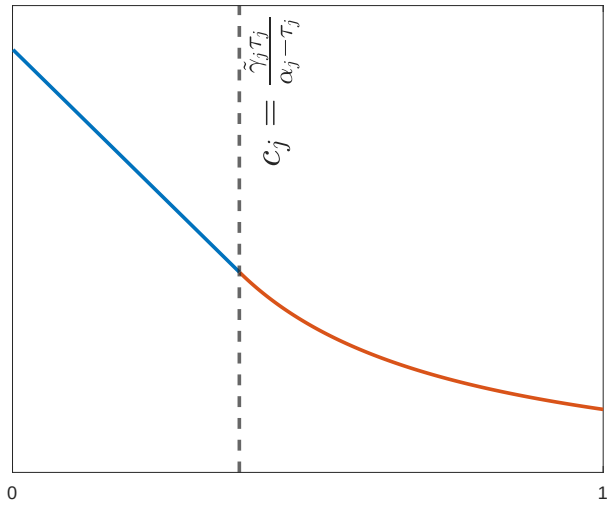


Figure 1: The function $p_j(w_j)$.

knot-point where the variable γ_j is made unactivated. For $w_j < c_j$, the contribution to the bound decreases linearly as a function of w_j. For $w_j > c_j$, the contribution to the bound decreases at a decreasing rate as a function of w_j.

To solve (G.3), we examine the KKT conditions with Lagrange multipliers $\boldsymbol{\eta}^* \in \mathbb{R}^J$ for the inequality constraints $\boldsymbol{w} \in \mathbb{R}^J_+$ and $\nu^* \in \mathbb{R}$ for the equality constraint $\mathbf{1}^\top \boldsymbol{w} = 1$. The conditions are given as,

$$p_j'(\widehat{w}_j) - \eta_j^* + \nu^* = 0, \quad \eta_j^* \widehat{w}_j = 0, \quad \eta_j^* \geq 0, \quad j = 1, \ldots, J. \tag{G.4}$$

Given $\mathbf{1}^\top \widehat{\boldsymbol{w}} = 1$ and $\widehat{\boldsymbol{w}} \in \mathbb{R}^J_+$, we note from the first condition that each η_j^* acts as a slack variable and can be eliminated, leaving,

$$\widehat{\boldsymbol{w}} \in \mathbb{R}^J_+, \qquad \mathbf{1}^\top \widehat{\boldsymbol{w}} = 1, \qquad \widehat{w}_j(\nu^* + p_j'(\widehat{w}_j)) = 0, \quad j = 1, \ldots, J,$$
$$\nu^* \geq -p_j'(\widehat{w}_j), \quad j = 1, \ldots, J.$$

These equations imply that when $\widehat{w}_j > 0$, the negative gradient $-p_j'(\widehat{w}_j) = \nu^*$. Note that,

$$-p_j'(w_j) := \begin{cases} \frac{(\tilde{\gamma}_j \tau_j)^2}{w_j^2}, & \text{if} \quad w_j \geq c_j \,\text{and}\, c_j \in (0, 1] \\ (\alpha_j - \tau_j)^2, & \text{else} \end{cases}.$$

Since $-p_j(w_j)$ are non-increasing functions for $w_j \in [0,1]$ and $-p_j(0) = (\alpha_j - \tau_j)^2$, we must have $\widehat{w}_j = 0$ when $(\alpha_j - \tau_j)^2 < \nu^*$. Without loss of generality, we assume the indexes are sorted such that $(\alpha_1 - \tau_1)^2 \geq (\alpha_2 - \tau_2)^2 \geq \cdots \geq (\alpha_J - \tau_J)^2$ (we can reverse the sort to obtain the original index after weight allocation). Our algorithm checks two possible cases for ν^*.

Case 1.

$$\nu^* = (\alpha_k - \tau_k)^2, \quad \text{for} \quad k \in \{1, \ldots, J\}.$$

In this case,

$$\widehat{w}_j := \begin{cases} (\widetilde{\gamma}_j \tau_j)/\sqrt{\nu^*}, & j < k \\ 1 - \sum_{i<k} \widehat{w}_i, & j = k \\ 0, & j > k \end{cases}.$$

This solution is feasible if $-p_k(\widehat{w}_k) = \nu^*$, where $\widehat{w}_k = 1 - \sum_{j \neq k} \widehat{w}_j$. This holds if $0 < \widehat{w}_k$ and $\widehat{w}_k < c_k$ or $c_k \notin [0,1]$.

Case 2.

$$(\alpha_k - \tau_k)^2 > \nu^* > (\alpha_{k+1} - \tau_{k+1})^2, \quad \text{for} \quad k \in \{1, \ldots, J\}.$$

In this case,

$$\nu^* = -p_j(\widehat{w}_j) = \frac{(\widetilde{\gamma}_j \tau_j)^2}{w_j^*}, \quad \text{for} \quad j \leq k.$$

Solving simultaneously with the constraint $\sum_j \widehat{w}_j = 1$, we obtain:

$$\nu^* = \left(\sum_{j \leq k} \widetilde{\gamma}_j \tau_j \right)^2.$$

The optimal weights are then,

$$\widehat{w}_j := \begin{cases} (\widetilde{\gamma}_j \tau_j)/\sqrt{\nu^*}, & j \leq k \\ 0, & j > k \end{cases}.$$

Algorithm G.1 iterates over $k = 1, \ldots, J$, evaluating the two cases described above. It optimally updates the weight vector, terminating once the correct case is identified and returning the feasible solution. Note that we remove j from sorted indexes if $\widetilde{\gamma}_j = 0$ and $\alpha_j - \tau_j \geq 0$ (in this case, $\widehat{w}_j \leftarrow 0$ and γ_j remains unactivated).

Algorithm G.1: `getWeights`; Optimal weights for $h_{\boldsymbol{w}}(\boldsymbol{\gamma}, \widetilde{\boldsymbol{\gamma}})$

input: $\boldsymbol{\alpha} = [n\lambda\sqrt{p_1}, \ldots, n\lambda\sqrt{p_J}], \boldsymbol{\tau} = [\|\mathbf{Z}_1^\top \mathbf{G}(\widetilde{\boldsymbol{\gamma}})\boldsymbol{y}\|, \ldots, \|\mathbf{Z}_J^\top \mathbf{G}(\widetilde{\boldsymbol{\gamma}})\boldsymbol{y}\|], \widetilde{\boldsymbol{\gamma}}$

output: optimal weights $\boldsymbol{w}$

$\boldsymbol{w} \leftarrow \mathbf{0}$

$U \leftarrow [u_1, \ldots, u_J]$ so that $(\alpha_{u_1} - \tau_{u_1})^2 \geq (\alpha_{u_2} - \tau_{u_2})^2, \ldots, \geq (\alpha_{u_J} - \tau_{u_J})^2$

$U \leftarrow U \setminus \{j : \alpha_j - \tau_j \geq 0, \widetilde{\gamma}_j = 0\}$ // remove groups that remain unactivated

$d \leftarrow |U|$

for $k = 1, \ldots, d$ **do**

  $\nu \leftarrow (\alpha_{u_k} - \tau_{u_k})^2$

  $w_{u_j} \leftarrow (\widetilde{\gamma}_{u_j} \tau_{u_j})/\sqrt{\nu}$ for $j < k$

  $w_{u_k} \leftarrow 1 - \sum_{j<k} w_{u_j}$

  $c_k \leftarrow \widetilde{\gamma}_{u_k} \tau_{u_k} / (\alpha_{u_k} - \tau_{u_k})$

  if $0 \leq w_{u_k}$ *and* $(w_{u_k} < c_k$ *or* $c_k \notin (0, 1])$ **then** // check Case 1

    break

  $\nu \leftarrow (\sum_{j \leq k} \widetilde{\gamma}_{u_j} \tau_{u_j})^2$

  if $(\alpha_{u_k} - \tau_{u_k})^2 > \nu > (\alpha_{u_{k+1}} - \tau_{u_{k+1}})^2$ **then** // check Case 2

    $w_{u_j} \leftarrow (\widetilde{\gamma}_{u_j} \tau_{u_j})/\sqrt{\nu}$ for $j \leq k$

    break

return $\boldsymbol{w}$

H Fitting Over a Grid

The BCD Algorithm B.1 and the MM Algorithm C.1 solve for $\widehat{\boldsymbol{\theta}}$ for a single value of λ. Usually, the best λ is unknown before fitting the group lasso, and the optimal λ is determined through cross-validation over a grid. Consider a grid of m values:

$$\lambda_1 < \lambda_2 < \ldots < \lambda_m.$$

Typically, m is 100 or 1000. Larger values of λ correspond to sparser solutions (more zero-groups in $\widehat{\boldsymbol{\theta}}_\lambda$), while smaller values correspond to fewer zero-groups in the solution vector.

The largest value, λ_m, is chosen so that it is equal to the smallest λ for which $\widehat{\boldsymbol{\theta}}_\lambda = \mathbf{0}$. By inspecting the KKT conditions (8), it can be shown that $\lambda^* := \max_j \frac{\|\mathbf{Z}_j^\top \boldsymbol{y}\|}{n\sqrt{p_j}}$ satisfies this condition. We start with the zero solution $\widehat{\boldsymbol{\theta}}_{\lambda_m}$ corresponding to $\lambda_m := \lambda^*$ and then compute $\widehat{\boldsymbol{\theta}}_{\lambda_{m-1}}$ for a smaller λ_{m-1} using $\widehat{\boldsymbol{\theta}}_{\lambda_m}$ as an initial guess. Once $\widehat{\boldsymbol{\theta}}_{\lambda_{m-1}}$ is computed, we proceed to $\widehat{\boldsymbol{\theta}}_{\lambda_{m-2}}$ using $\widehat{\boldsymbol{\theta}}_{\lambda_{m-1}}$ as the initial guess, and so on, until the entire grid of size m is covered. This approach, often called "warm starts", ensures that the initial values are close to the solution.

Typically, the smallest value is set to $\lambda_1 := \omega\lambda_m$, where ω is a small number, say 10^{-1}. The values between λ_1 and λ_m are determined by creating a log-spaced grid:

$$\lambda_j = \exp\left(\ln(\lambda_1) + \frac{j-1}{m-1}(\ln(\lambda_m) - \ln(\lambda_1))\right), \quad j = 1, \ldots, m.$$

In other words, the logarithms of the λ values are equally spaced on a grid with m points.

I Additional Numerical Studies

I.1 Iterations for Regularization Paths

Table I.1 displays the mean iterations over 50 simulation for the experiment conducted in Section 4.1.2.

Table 1: Iterations for Setting 1 and Setting 2. Means of 50 replications are reported.

			Correlation (ψ, ρ)		
Setting	Group Size	Method	(0.3, 0.3)	(0.6, 0.6)	(0.9, 0.9)
1	3	BCD	4.63×10^5	8.71×10^5	4.18×10^5
		MM	2.97×10^4	2.65×10^4	2.18×10^4
	5	BCD	3.71×10^5	4.11×10^5	1.88×10^5
		MM	2.28×10^4	2.02×10^4	1.79×10^4
2	3	BCD	4.58×10^5	7.08×10^5	4.59×10^5
		MM	2.64×10^4	2.46×10^4	2.75×10^4
	5	BCD	2.97×10^5	3.37×10^5	2.18×10^5
		MM	2.00×10^4	2.03×10^4	2.69×10^4

I.2 Example Convergence Paths

In the following experiment, we compare the convergence of the MM Algorithm C.1 and the BCD Algorithm B.1 for fixed values of λ, n, d, and J. We generate 100 datasets under setting 2, with 100 activated groups ($k = 100$), a group size of $d = 3$, and a total of 1000 groups ($J = 1000$). The correlation parameters are set as $\rho = 0.3$ and $\psi = 0.3$. We fix $\lambda = 15/n$ where $n = 1000$ observations. We run MM Algorithm C.1 and the BCD Algorithm B.1 on these datasets to obtain $\widehat{\boldsymbol{\theta}}$. The average number of non-zero groups in $\widehat{\boldsymbol{\theta}}$ is 256.17, where the average is computed over all the datasets generated.

Figure 2 shows that, under these simulation settings, the MM Algorithm C.1 outperforms the BCD Algorithm B.1 in terms of convergence with respect to the number of iterations t, whether measuring the maximum change in $\boldsymbol{\theta}$ or the minimum obtained objective

function f. Initially, the MM Algorithm may appear slower because it starts with $\boldsymbol{\theta}^{(0)} = \mathbf{0}$, and only a few groups become activated at each iteration due to the restricted number of positive weights. However, after approximately 200 iterations, the MM Algorithm displays more rapid convergence. Therefore, the MM Algorithm is significantly quicker in achieving a high-accuracy solution in this setting.

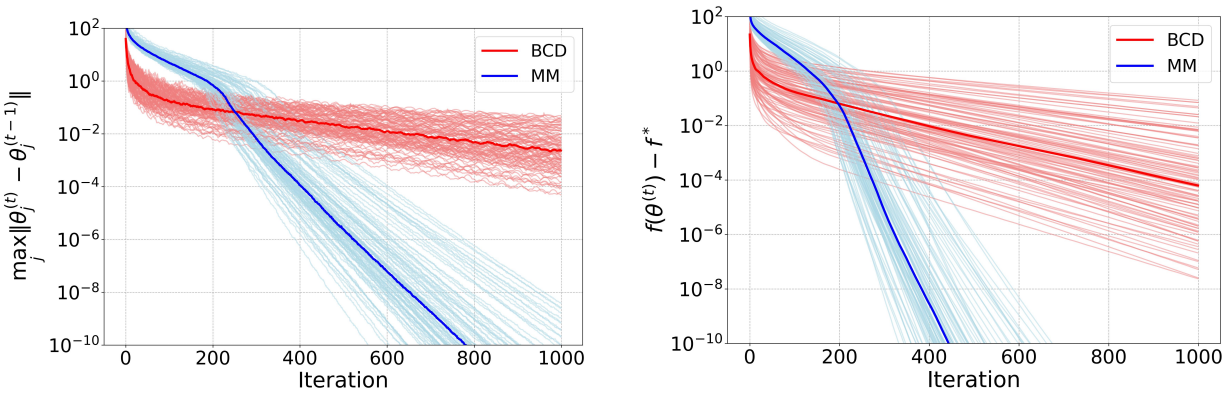
Another way to demonstrate the superiority of the MM algorithm C.1 in this simulation setting is by showing the iteration t at which each algorithm identifies the activated set at the group lasso solution, that is, for what t does $\mathcal{A}^{(t)} = \mathcal{A}$. Figure 3 shows that the MM Algorithm, while making fewer changes to $\mathcal{A}^{(t)}$ at each iteration, reaches $\mathcal{A}$ in fewer iterations (on average 400) compared to the BCD Algorithm B.1. The BCD Algorithm approximates the correct activated set in relatively few iterations but takes many more iterations to finally stabilize on the correct activated set, typically identifying the correct activated set by iteration 1000.

I.3 Computational Performance for Varying Number of Observations and Groups

We now present results studying the effect of the number of groups J and the number of observations n on the performance of the MM Algorithm C.1 and the BCD Algorithm B.1. We generate datasets with a fixed group size of $d = 3$. The correlation parameters are set to $\rho = 0.3$ and $\phi = 0.3$.

For each scenario, where we vary either J or n, we set λ by first constructing a grid as described in supplementary Section H with $\lambda_1 = \omega\lambda_m$, where $\omega = 10^{-1}$. Then, in each simulation, we run the MM Algorithm C.1 and the BCD Algorithm B.1 for a single $\lambda = \lambda_{50}$ value, i.e., the mid-value on the grid.

Figure 4 shows the runtime and mean iterations until convergence for varying numbers of groups J, averaging over 50 simulations. Each dataset is simulated under Setting 2,



(a) (b)

Figure 2: Comparison of the BCD and MM algorithms over two convergence criteria across iterations. Figure 2a shows the convergence for the maximum change in $\boldsymbol{\theta}$ over the number of iterations t. Figure 2b shows the convergence for the objective function f over the number of iterations t. The value of f^* is obtained by running the MM Algorithm C.1 for $t^* = 10^3$ iterations and evaluating the objective $f^* := f(\widehat{\boldsymbol{\theta}}^{(t^*)})$. Results from all the 100 simulations are shown, with the median path highlighted.

with $k = J/10$ activated groups and $n = 1000$ observations. Figure 4 also shows the mean runtime and mean iterations t for varying numbers of observations n, averaged over 50 simulations. Each dataset is simulated under Setting 2, with $k = 100$ activated groups and $J = 1000$ total groups.

We observe an order of magnitude improvement in the MM Algorithm over the BCD Algorithm in both iterations and runtime for all values of J and n. As J increases, the number of iterations for both algorithms shows minimal growth, but the runtime increases, as each cycle becomes more computationally expensive. As n increases, the number of iterations required for convergence decreases for both algorithms. These simulations indicate that the MM Algorithm is well-suited for problems with large n and J.

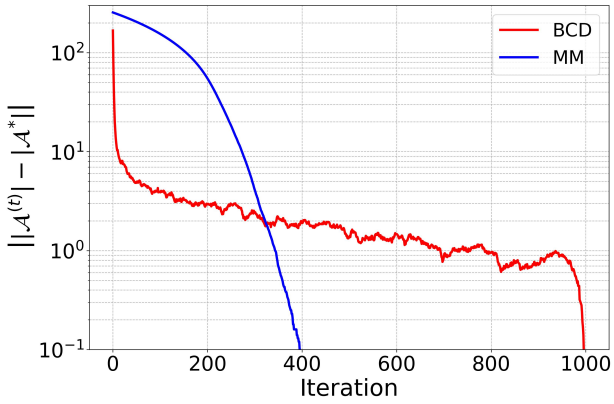
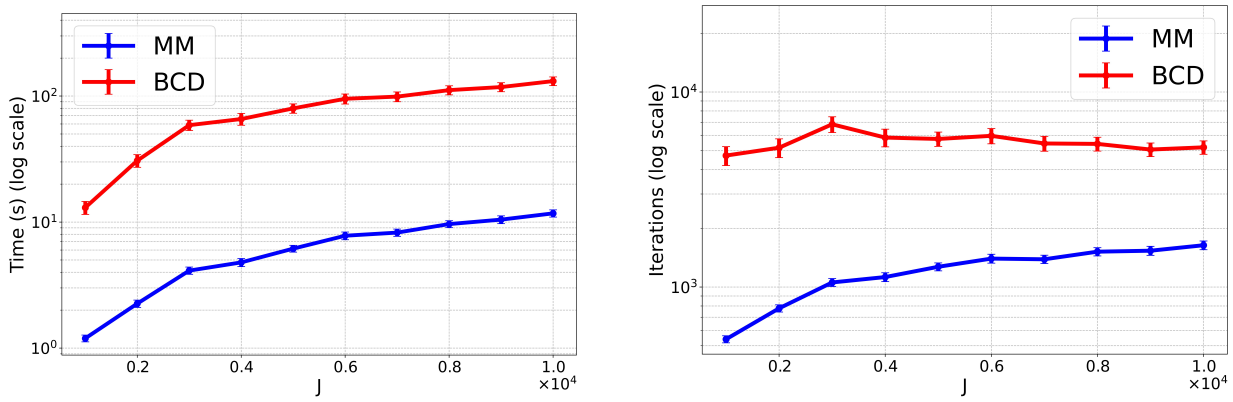


Figure 3: Comparison of the BCD and MM algorithms based on the sizes of the activated set. The value of $\mathcal{A}^*$ is obtained by running the MM Algorithm C.1 for $t^* = 10^3$ iterations and defining $\mathcal{A}^* := \mathcal{A}^{(t^*)}$. The mean path from 100 simulations is shown.

I.4 Pseudo Real Data

In the following experiemtns we simulate $\boldsymbol{y}$ from model (1), using matrices $\mathbf{X}_j$ constructed from a real dataset containing genetic variants of mice. Each $\mathbf{X}_j$ is constructed from the mouse single nucleotide polymorphism (SNP) array data set available from the Open Mendel project (Zhou et al., 2020). The dataset consists of $\mathbf{X}$, an $n \times p$ matrix representing p genetic variants for n individual mice. Each entry of the matrix represents an allele count and can take on the discrete values $\mathbf{X}_{i,j} = \{0, 1, 2\}$. For this experiment, $p = 10{,}150$ and $n = 1{,}940$. We artificially generate J different genetic regions by partitioning the columns of $\mathbf{X}$ into $\mathbf{X}_{j=1,\ldots,J}$ gene matrices, where $\mathbf{X}_j \in \mathbb{R}^{n \times d}$, and d is the group size. We simulate $\boldsymbol{\beta}$ and $\boldsymbol{y}$ as described in Section 4.1.1. In this case, $\boldsymbol{y}$ mimics a vector of quantitative trait measurements for the n mice.

Each simulation scenario is replicated 25 times. The mean running time and mean number of iterations are reported. The results of the genetic study simulation are provided in Table 2. We observe that the MM algorithm outperforms the BCD algorithm for all values of d for this dataset in both the number of iterations and running time until convergence. These simulations provide a good example of the suitability of the MM algorithm



(a) Running Time (s)

(b) Iterations

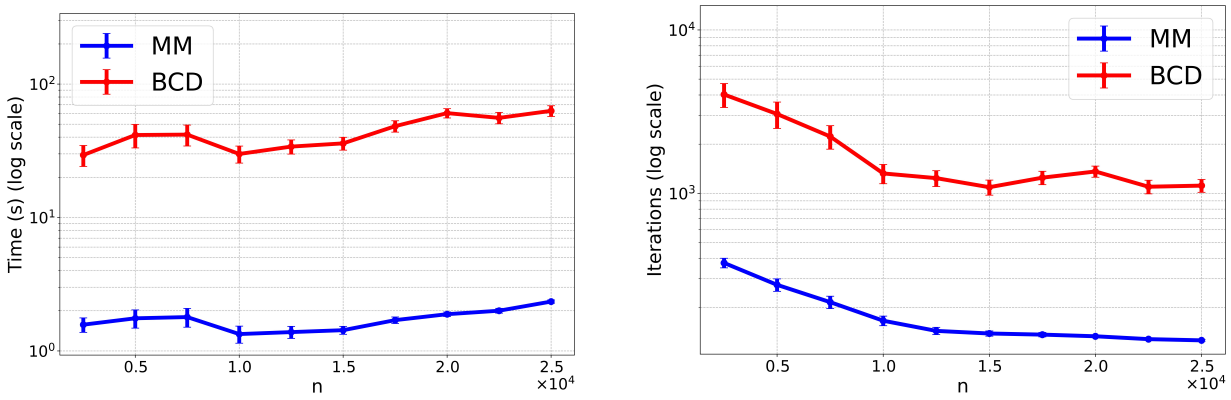
Figure 4: Runtime and iterations t until convergence for the BCD and MM algorithms for increasing number of groups J. The mean values from 50 simulations are reported, with standard errors represented by error bars.

for non-Gaussian data.

I.5 Local Convergence Experiment

In the following study we investigate the empirical convergence behavior under the assumption that both algorithms have already identified the activated set, providing insights into their performance locally near the optimal solution. In Figure 3, we demonstrated that under mild correlation between the predictors, the MM algorithm can identify the activated set in fewer iterations compared to the BCD algorithm. We now investigate the convergence behavior under the assumption that both algorithms have already identified the activated set. This analysis should provide insight into how the MM and BCD algorithms perform locally near the optimal solution.

To this end, we simulate datasets where the true activated set is large and easily identifiable by both algorithms. Specifically, we set the group size to $d = 3$ with a total of $J = 100$ groups, of which $k = 90$ are activated. The number of observations is $n = 1000$, and the sparsity parameter is set to $\lambda = 5/n$ to encourage non-sparse solutions. We consider two



(a) Running Time (s)

(b) Iterations vs n

Figure 5: Runtime and iterations t until convergence for the BCD and MM algorithms for increasing number of observations n. The mean values from 50 simulations are reported, with standard errors represented by error bars.

correlation settings: one where the correlation parameters are $\rho = 0.3$ and $\psi = 0.3$, and another where $\rho = 0$ and $\psi = 0$. In the latter case, the condition number of the data matrix $\mathbf{X}$ will be close to one, as $\mathbb{E}\left[\mathbf{X}^\top \mathbf{X}\right] = \mathbf{I}_p$. We define the iteration counter t' as the number of iterations starting from the point at which the activated set $\mathcal{A}$ is identified by the algorithm. At this stage, no new groups become activated or unactivated, and the focus shifts solely to satisfying the KKT conditions for the activated groups.

The results in Figure 6a indicate that when the data is very well-conditioned, both the MM Algorithm C.1 and the BCD Algorithm B.1 converge to the solution in a few iterations. In Figure 6b, under mild correlation from (22) the expected condition number of $\mathbf{X}^\top \mathbf{X}$, as described by (22), is higher than in Figure 6a, the MM algorithm is minimally impacted and, on average, converges to a tolerance of 10^{-11} in 50 iterations. In contrast, the conditioning affects the BCD algorithm more and the algorithm shows minimal progress toward the solution within the same number of iterations. These experiments suggest that the auxiliary objective $g(\boldsymbol{\gamma})$ is better-behaved near the solution when the data $\mathbf{X}$ is poorly conditioned. This observation motivates future theoretical investigations into the local

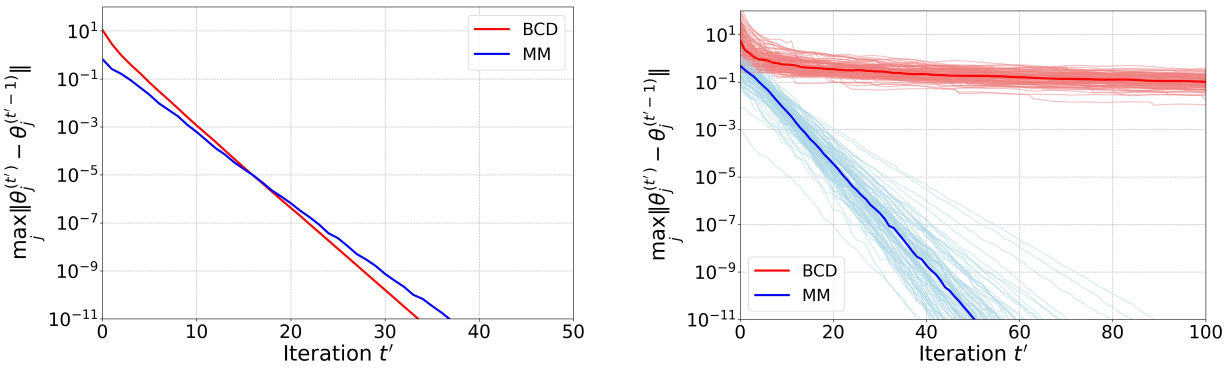
Table 2: Running Time (s) and No. of Iterations for Pseudo Data. Means of 25 replications are reported.

Setting	Group Size	Method	Running Time (s)	No. of Iterations
1	10	BCD	1.00×10^3	6.28×10^4
		MM	5.12×10^2	3.84×10^4
	25	BCD	6.20×10^2	4.18×10^4
		MM	3.12×10^2	2.02×10^4
2	10	BCD	7.99×10^2	5.11×10^4
		MM	4.46×10^2	3.65×10^4
	25	BCD	5.59×10^2	3.76×10^4
		MM	2.94×10^2	1.98×10^4

convergence rates of the MM Algorithm C.1 and the BCD Algorithm B.1.

I.6 Lasso Simulations

In this section, we demonstrate that the MM algorithm is competitive when the group size is one ($p_j = 1$) for $j = 1, \ldots, J$. In this case, the BCD algorithm reduces to the standard coordinate descent algorithm for lasso. As in Section 4.1.2, we present results where $\widehat{\boldsymbol{\theta}}$ is computed over a grid of λ values, as described in supplementary Section H. We set $\lambda_1 = \omega\lambda_m$, where $\omega = 10^{-1}$. We run the MM Algorithm C.1 and the BCD Algorithm B.1 on datasets simulated according to Setting 1 as described in Section 4.1.1, with group sizes $d = 1$. In these simulations, the number of observations is $n = 1000$, and the total number of predictors is $p = 1000$. The number of activated groups is $k = 100$ with SNR $= 1$.



(a) $\rho = 0, \psi = 0$ (b) $\rho = 0.3, \psi = 0.3$

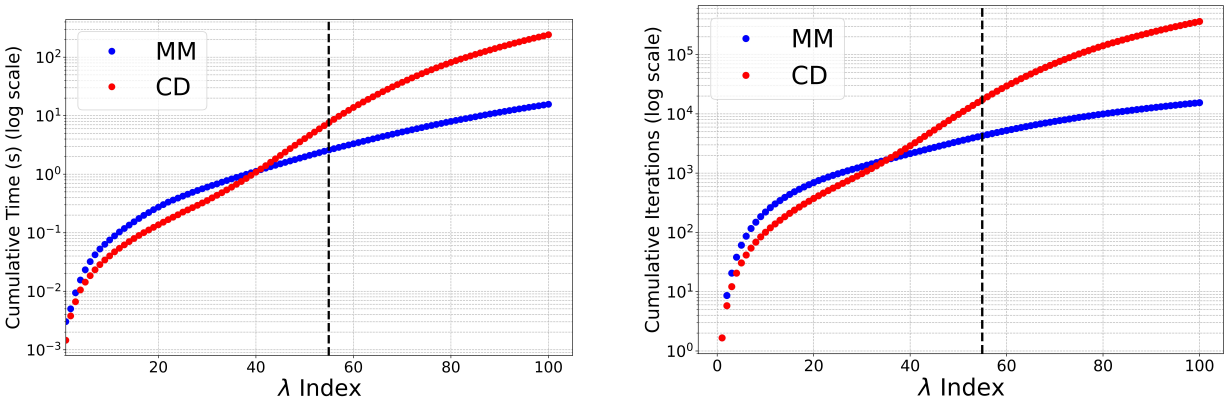
Figure 6: Comparison of the BCD and MM algorithms once $\mathcal{A}$ identified. Figure 6a shows the convergence when $\rho = 0, \psi = 0$. Figure 6b shows the convergence when $\rho = 0.3, \psi = 0.3$. Iteration counter t' is the number of iterations after $\mathcal{A}$ is identified. Results from 100 simulations are shown, with the mean path highlighted.

In Figure 7, we show the cumulative runtime and cumulative iterations as both algorithms progress from λ_m to λ_1. Let $\widehat{\boldsymbol{\beta}}(\lambda)$ denote the solution to the group lasso problem computed with the regularization parameter λ. We also indicate the average λ index that minimizes the prediction error $\|\mathbf{X}\widehat{\boldsymbol{\beta}}(\lambda) - \mathbf{X}\boldsymbol{\beta}\|$, marking the optimal λ. Figure 7 clearly shows that the MM algorithm increasingly outperforms the BCD algorithm in the less sparse regions of the path, as λ decreases. The MM algorithm achieves up to two orders of magnitude improvement in runtime.

References

Bertsekas, D. P. (2014). *Constrained optimization and Lagrange multiplier methods*. Academic press.

Breheny, P. and Huang, J. (2015). Group descent algorithms for nonconvex penalized linear and logistic regression models with grouped predictors. *Statistics and Computing*, 25:173–187.



(a) Cumulative Time(s)

(b) Cumulative Iterations

Figure 7: Cumulative runtime and iterations t until convergence for the CD and MM algorithms, as λ decreases in a regularization path. The mean values from 50 simulations are reported, with the vertical dotted black line indicating the mean index at which prediction error is minimized.

Henderson, H. V. and Searle, S. R. (1981). On deriving the inverse of a sum of matrices. *SIAM review*, 23(1):53–60.

Lange, K., Chambers, J., and Eddy, W. (2010). *Numerical analysis for statisticians*, volume 1. Springer.

Mishkin, A. and Pilanci, M. (2022). The solution path of the group lasso. In *OPT 2022: Optimization for Machine Learning (NeurIPS 2022 Workshop)*.

Simon, N. and Tibshirani, R. (2012). Standardization and the group lasso penalty. *Statistica Sinica*, 22(3):983.

Zhou, H., Hu, L., Zhou, J., and Lange, K. (2019). Mm algorithms for variance components models. *Journal of Computational and Graphical Statistics*, 28(2):350–361.

Zhou, H., Sinsheimer, J. S., Bates, D. M., Chu, B. B., German, C. A., Ji, S. S., Keys, K. L., Kim, J., Ko, S., Mosher, G. D., et al. (2020). Openmendel: a cooperative programming project for statistical genetics. *Human genetics*, 139:61–71.