A Homotopic Method to Solve the Lasso Problems with an Improved Upper Bound of Convergence Rate

Yujie Zhao  
*Georgia Institute of Technology, Atlanta, U.S.*  
E-mail: yzhao471@gatech.edu  
Xiaoming Huo  
*Georgia Institute of Technology, Atlanta, U.S.*  
E-mail: huo@gatech.edu

**Summary.** In optimization, it is known that when the objective functions are strictly convex and well-conditioned, gradient based approaches can be extremely effective, e.g., achieving the exponential rate in convergence. On the other hand, the existing Lasso-type of estimator in general cannot achieve the optimal rate due to the undesirable behavior of the absolute function at the origin. A homotopic method is to use a sequence of surrogate functions to approximate the $\ell_1$ penalty that is used in the Lasso-type of estimators. The surrogate functions will converge to the $\ell_1$ penalty in the Lasso estimator. At the same time, each surrogate function is strictly convex, which enables provable faster numerical rate of convergence. In this paper, we demonstrate that by meticulously defining the surrogate functions, one can prove faster numerical convergence rate than any existing methods in computing for the Lasso-type of estimators. Namely, the state-of-the-art algorithms can only guarantee $O(1/\epsilon)$ or $O(1/\sqrt{\epsilon})$ convergence rates, while we can prove an $O((\log(1/\epsilon))^2)$ for the newly proposed algorithm. Our numerical simulations show that the new algorithm also performs better empirically.

**Keywords:** Lasso, homotopic method, convergence rate, $\ell_1$ regularization

1. Introduction

Lasso (Tibshirani, 1996) has demonstrated to be an effective method in model estimation and selection in the past decades. Lasso aims to enable sparsity during the estimation process when the dimension becomes increasingly large. We review the formulation of Lasso-type estimator in the following. Let $y \in \mathbb{R}^n$ denote a response vector and $X \in \mathbb{R}^{n \times p}$ be a model matrix (of predictors), and vector $\beta^*$ is the true regression coefficients. We want to estimate $\beta$. Vector $w$ contains white-noise entries, which are independently and identically distributed following, e.g., the Normal distribution $N(0, \sigma^2)$. Accordingly, the data generation mechanism in the classical linear regression model can be written as

$$y = X\beta^* + w.$$  

The Lasso estimator $\hat{\beta}$ is commonly written as

$$\hat{\beta} = \arg\min_{\beta} \left\{ \frac{1}{2n} ||y - X\beta||^2_2 + \lambda||\beta||_1 \right\}, \quad (1)$$
where parameter $\lambda$ controls the trade-off between the sparsity and model’s goodness of fit. Essentially, solving equation (1) is an optimization problem, where many research in operations research and optimization have devoted to. For simplicity, we denote the objective function as follows:

$$F(\beta) = \frac{1}{2n} \|y - X\beta\|_2^2 + \lambda \|\beta\|_1.$$  \hfill (2)

In this paper, we don’t consider the selection of parameter $\lambda$, which by itself has a large literature; consequently, we don’t include $\lambda$ in the notation $F(\beta)$.

Lasso-algorithms are those algorithms that aim to solve Lasso problems with an objective function as in (2). This paper considers a Lasso-algorithm that has a better provable upper bounds in its convergence rate. The convergence rate will be derived in terms of order of computational complexity. Because there is no close form solution of the minimizer $\hat{\beta}$ (unless subgradient is utilized), most Lasso-algorithms are iterative with an iteration index $k$.

For an iterative estimator/minimizer, their distance to the optimal estimator/minimizer can be measured by $\epsilon$-precision, whose definition is listed below.

**Definition 1.1.** Suppose $\beta^{(k)}$ is the $k$th iterative estimator in a certain Lasso-algorithm and $\hat{\beta}$ is the global minimizer that is defined as $\hat{\beta} = \arg\min_\beta F(\beta)$. Recall that $F(\beta)$ is defined in (2). For any pre-fixed $\epsilon > 0$, if we have

$$F(\beta^{(k)}) - F(\hat{\beta}) \leq \epsilon,$$  \hfill (3)

then we declare that $\beta^{(k)}$ achieves the $\epsilon$-precision.

The order of complexity will be utilized to measure how fast an iterative algorithm converges to the global minimum. Recall that our optimization problem is to minimize the function $F(\beta)$. We obtain the iterative estimator $\beta^{(k)}$ at the $k$th iteration. Recall that $\hat{\beta}$ denotes the global minimum as in Definition 1.1. The order of complexity measures the number of operations needed to achieve the $\epsilon$-precision defined in Definition 1.1. More specifically, we adopt the $O$ notation as follows. The order of complexity of a Lasso-algorithm is $O(np^{1\epsilon})$, if in order to achieve the $\epsilon$-precision, the number of all needed numeric operations can be upper bounded by a constant multiplies $np^{1\epsilon}$. Notice that the order of complexity gives an upper bound of the number of numerical operations in order to achieve certain precision. It does not say anything on the average performance of the algorithm. It is possible that an algorithm with larger upper bounds performs better in some cases than an algorithm with lower upper bounds. In this paper, we consider a numerical strategy that can lead to an iterative algorithm that can achieve the aforementioned $\epsilon$-precision with a lower order of complexity. Recall that $\epsilon > 0$ is typically small. In theory, an $O\left(np^{\frac{1}{\sqrt{\epsilon}}}\right)$ Lasso-algorithm has a lower order of complexity than an $O\left(np^{\frac{1}{\epsilon}}\right)$ Lasso-algorithm. Moreover, an $O\left(np\log(\frac{1}{\epsilon})\right)$ Lasso-algorithm has an even lower order of complexity. We will then use numerical simulations to compare with some representative algorithms in some well-studied cases.

Due to the nature of the objective function in the Lasso problem, which is $F(\beta) = \frac{1}{2n} \|y - X\beta\|_2^2 + \lambda \|\beta\|_1$, the first-order method is often used. The first term in $F(\beta)$ is a nice quadratic function, which is numerically amenable. The challenge is rooted in the
second term of $F(\beta)$, the $\ell_1$ regularization term $\|\beta\|_1$, which is not differentiable at the origin. We review some state-of-the-art Lasso-algorithms, which will serve as the benchmarks of our algorithm. Daubechies et al. (2004) proposes a Lasso-algorithm using the first-gradient and the Hessian matrix of the first term in $F(\beta)$. Daubechies et al. (2004) approximate $\frac{1}{n}\|y - X\beta\|^2_2$ by its second-order Taylor expansion. It is computationally expensive to directly calculate the Hessian matrix of $\frac{1}{n}\|y - X\beta\|^2_2$, i.e., $X'X/n$, especially when $p$ is large. To avoid the time-consuming calculation of the Hessian matrix, a key idea in Daubechies et al. (2004) is to approximate the Hessian matrix by a diagonal matrix, whose diagonal entries are the maximal eigenvalue of $X'X/n$. After the quadratic approximate of the objective function, a proximal mapping is formed, where soft-thresholding operator can be easily applied.

Proximal gradient descent is adopted in Daubechies et al. (2004). See additional mathematical review in Appendix A. Early foundational work on proximal gradient descent can be found in Brègman (1966), Hestenes (1969), and Rockafellar (1976). As the techniques matures, they became widely used in different fields. As a result, they have been referred to by a diverse set of names, including proximal algorithm, proximal point, and so on. In the survey of Polson et al. (2015), it can be seen that, many other widely-known statistical methods – including, the Majorization-Minimization (MM) (Lange et al. (2000), Hunter and Li (2005)), and the Alternating Direction Method of Multipliers (ADMM) (Boyd et al., 2011) – fall into the proximal framework. In the review of the present paper, many Lasso-algorithms follow the proximal gradient descent as well.

The classical first-order method use the gradient at the immediate previous solution. To learn from the “history,” Beck and Teboulle (2009) propose a Lasso-algorithm, which takes advantage of the gradients at previous two solutions. Since it uses the historic information, it is also referred as the momentum algorithm. Essentially, Beck and Teboulle (2009) falls into the framework of the Accelerate Gradient Descent (AGD), which is proposed by Nesterov (1983), and later widely applied into many optimization problems to speed up the convergence rate, seeing examples in Nesterov (2003), Beck and Teboulle (2009), Nesterov (2013), Li and Lin (2013), and many more. The mathematical details of the aforementioned two Lasso-algorithms (as well as the coming ones) are provided in Appendix A.

The above two Lasso-algorithms update their estimates globally. On the contrary, the third Lasso-algorithm utilizes coordinate descent to update the estimate. This method is widely used and an corresponding R package named glmnet (Friedman et al., 2010) has fueled its adoption. The coordinate descent method has been proposed for the Lasso problem for a number of times, but only after Friedman et al. (2010), was its power fully appreciated. Early research work on the coordinate descent include the discovery by Hildreth (1957) and Warga (1963), and the convergence analysis by Tseng (2001). There are research work done on the applications of coordinate descent on Lasso problems, such as Fu (1998), Shevade and Keerthi (2003), Friedman et al. (2007), Wu et al. (2008), and so on. We choose Friedman et al. (2010) as a method to compare, since its implementation in the R package, glmnet, is very well-known by statisticians.

The aforementioned three Lasso-algorithms do not use a surrogate for the $\ell_1$ regularization term. Different from them, the fourth Lasso-algorithm aims to find a sur-
Table 1. The available orders of complexity of four existing Lasso-algorithms and ours (in the last column) for achieving the $\epsilon$-precision. The common factor that involves $n$ (the sample size) and $p$ (the dimensionality of the parameter) is omitted for simplicity.

| method      | ISTA | FISTA | CD   | SL   | Ours                  |
|-------------|------|-------|------|------|-----------------------|
| Order of complexity | $O(1/\epsilon)$ | $O(1/\sqrt{\epsilon})$ | $O(1/\epsilon)$ | $O(1/\epsilon)$ | $O\left([\log(1/\epsilon)]^2\right)$ |

rogate of the $\ell_1$ penalty term. Recall that the non-differentiability of the $\ell_1$ penalty at the origin makes it hard to enable fast convergence rate when applying the gradient descent method. [Schmidt et al. (2007)] proposes a surrogate function of the $\ell_1$ penalty by taking advantage of the non-negative projection operator (seeing equation (2) and (3) in [Schmidt et al. (2007)] for more details), where the surrogate function is twice differentiable. Consequently, the EM algorithms [Figueiredo, 2003] are used for the optimization. Among this type of Lasso-algorithms, we select Smooth Lasso (SL) [Mukherjee and Seelamantula, 2016] as a benchmark, because it is developed recently and is an improved version of [Schmidt et al. (2007)]. The difference from our proposed algorithm is that, we design a homotopic path (i.e., a sequence of surrogate functions) to make the surrogate functions closer and closer to the $\ell_1$ penalty. We will design a two-layer iteration algorithm to achieve a provable faster convergence rate. To the best of our knowledge, such a faster rate has not appeared in the literature.

1.1. Contribution
We propose a new Lasso-algorithm that has a better provable upper bounds in its convergence rate, in terms of the order of complexity. The state-of-the-art Lasso-algorithms we compare include

(a) the Iterative Shrinkage-Thresholding Algorithm (ISTA) [Daubechies et al., 2004],
(b) the Fast Iterative Shrinkage-Thresholding Algorithm (FISTA) [Beck and Teboulle, 2009],
(c) a Coordinate Descent (CD) [Friedman et al., 2010] algorithm, and
(d) the Smooth Lasso [Mukherjee and Seelamantula, 2016].

These algorithms are representative in the literature. See Appendix A for technical details on the above four benchmark algorithms. To show the advantage of our proposed method, in Table 1 we list the provable upper bounds in convergence rate of four benchmark algorithms and our algorithm.

We see that our algorithm achieves an order of complexity of log-polynomial of $1/\epsilon$, while other benchmarks have order of complexity of polynomial of $1/\epsilon$. The order of complexity of our proposed algorithm is established in Theorem 3.5.

1.2. Organization of this Paper
The organization of the rest of the paper is as follows. We develop our Lasso-algorithm in Section 2. The related main theory is established in Section 3. Numerical examples are shown in Section 4. Some discussion are presented in Section 5. In Appendix A, we summarize some necessary technical details of these benchmark algorithms. A useful
2. The proposed Algorithm

To circumvent the undesirable behavior of the $\ell_1$ penalty function ($\|\beta\|_1$) at the origin, we design an algorithm that solves a sequence of optimization problems: in each subproblem, the $\ell_1$ penalty function is replaced by a surrogate function. The surrogate functions ultimately converge to the $\ell_1$ penalty function. Our approach falls into the general framework of homotopic methods, therefore, we name our algorithm a Homotopy-Shrinkage (HS) algorithm. Its connection to shrinkage becomes evident when we describe the algorithm in details.

This section is organized as follows. A general description of the proposed HS algorithm is shown in Section 2.1. In Section 2.2, we describe how to choose the initial value of the hyper-parameter $t$, as well as its updating scheme. In Section 2.3, we present our design of early stopping in the inner loop, which is critical to achieve the lower order of complexity. The design of the surrogate functions (to approximate the $\ell_1$ penalty) is provided in Section 2.4.

2.1. Overview of the Proposed Algorithm

We design our Homotopy-Shrinkage algorithm that has two layer of loops: an outer-loop and an inner-loop. In an outer-iteration (in the outer-loop), the following objective function is minimized

$$F_t(\beta) = \frac{1}{2n} \|y - X\beta\|_2^2 + \lambda f_t(\beta)$$

where $f_t(\beta)$ is a surrogate function of the target function $\|\beta\|_1$. Here $t > 0$ is a parameter in the function, which controls the closeness between $\|\beta\|_1$ and $f_t(\beta)$. More specifically, if $t$ decreases to zero, the function $f_t(\beta)$ converges to $\|\beta\|_1$. As mentioned earlier, the design of function $f_t(\beta)$ is postponed to Section 2.4. Each outer-iteration takes the previous stopping position (from the previous outer-iteration) as the starting point of this iteration. In the outer-loop, we start with a large $t$ initially, and then gradually decrease the value of $t$ until the desired accuracy $\epsilon$ is reached. The way to decrease the value of $t$ is a non-trivial task to make the designed algorithm has a provable lower order of complexity. The details on how to decrease the value of $t$ are given later.

In an inner-loop, for a fixed $t$, we employ the accelerated gradient descent (AGD) to minimize the current surrogate objective function $F_t(\beta)$. Note that by design, the surrogate function $F_t(\beta)$ will be strongly convex and well conditioned, consequently a lower order of complexity becomes achievable in the inner-loop. In particular, one can prove an log-polynomial computational complexity for this algorithm. To summarize the above key idea of our proposed algorithm, we present the pseudo code in Algorithm 1. Details of the proposed algorithm are discussed in the remainder of this section.

2.2. Value of the Hyper-parameter in the Proposed Algorithm

The first detail we would like to discuss is the choice of the initial value of the hyperparameter $t$, which is denoted as $t_0$. A well-designed initial point $t_0$ is important, because
Algorithm 1: Pseudo code of the proposed Homotopy-Shrinkage algorithm

**Input:** A response vector $y$, a model matrix $X$, a parameter $\lambda$ that relates to the Lasso.

**Output:** an estimator of $\beta$, which satisfies the $\epsilon$-precision.

1. **Hyper-parameter initialization (See Section 2.2)**
   - **Outer-Iteration:**
     - While the precision $\epsilon$ is not achieved do
       1. shrink $t$; /* For detailed shrinkage procedure, please refer to line 13 in Algorithm 2. */
       2. **Inner-Iteration:**
          - Use AGD to minimize $F_t(\beta)$ until the precision of the inner-iteration is achieved /* $F_t(\beta) = \frac{1}{2n} \|y - X\beta\|^2 + \lambda f_t(\beta)$ */
          - /* For detailed procedure of AGD, please refer to the inner-iteration in Algorithm 2. */

starting with an unnecessarily large $t_0$ would end up with more shrinkage steps (i.e., the outer-iterations), which in turns costs more numerical operations. We derive a minimal value of $t_0$ in equation (4) in Lemma 2.1.

**Lemma 2.1.** Suppose in a Lasso problem, we have the response vector $y \in \mathbb{R}^n$ and a model matrix $X \in \mathbb{R}^{n \times p}$. For our proposed algorithm, there exist a value $t_0$ that satisfies the following:

$$t_0 \in \left\{ t : \sum_{j=1}^{p} M(t)_{ij} (X'y/n)_j \leq t \right\}, \quad \forall i = 1, \ldots, p,$$

where $M(t) = \left( \frac{XX'}{n} + \frac{\lambda}{\delta^2} [\log(1+t)]^2 I \right)^{-1}$. Here $X'$ represents the transpose of matrix $X$, and we use this notation in the remaining of the paper. When one chooses the aforementioned $t_0$ as the initial point in the proposed algorithm, we have $|\beta_i^{(0)}| \leq t_0$ for any $1 \leq i \leq p$, where $\beta_i^{(0)}$ denotes the $i$th entry in the vector $\beta^{(0)} = M(t_0)X'y/n$.

**Proof.** See Appendix C.1.

The motivation of the above lemma is to ensure that when $t = t_0$ in our proposed algorithm, the initial estimator $\beta^{(0)}$ is going to be bounded by $t_0$ entrywise.

The second detail we would like to discuss is the design of the shrinkage path of $t$ in line 3 in Algorithm 1. This is designed as follows. First, we start with a relative large $t_0$, which has already been discussed above. Then, in the $k$th outer-iteration ($k \geq 0$), we shrink the $t_k$ to $t_{k+1} = t_k(1 - h)$, where $h$ is set to be a predetermined values. In our proofs, we will show that this can lead to a provable lower bound in the order of complexity.
2.3. Early Stopping in the Inner-Loop and the Complete Algorithm

To achieve a better order of complexity, it is critical to stop the inner iteration early. Specifically speaking, in the \(k\)th outer-iteration, the inner-iteration is stopped when

\[
F_{t_k}(\beta(k)[s]) - F_{\min,k} < \tilde{\epsilon}_k,
\]

where \(\beta(k)[s]\) denotes the iterative estimator in the \(s\)th inner-iteration of the \(k\)th outer iteration, and we have

\[
F_{\min,k} = \min_{\beta} F_{t_k}(\beta).
\]

Here, we set \(\tilde{\epsilon}_k = \frac{\lambda B}{32} \left(\log(1 + t_k)\right)^2\), where \(B\) is the upper bound of \(\left|\beta_i^{(k)}\right|\) for all \(i = 1, 2, \ldots, p\) and \(k = 1, 2, \ldots\). The justification of our choice of \(\tilde{\epsilon}_k\) is elaborated in our proof; its detailed derivation can be found in Appendix C.4. It is worth noting that, theoretically, our algorithm can achieve the order of complexity of \(O\left((\log(1/\epsilon))^2\right)\), yet, in practice, it may not be implementable. We may have to use some alternative, such as stopping the inner-iteration after a fixed number of steps. The matter of fact is that, the stopping rule in the inner-iteration of our algorithm requires knowing the value of \(F_{\min,k}\), which is not possible. In simulations, it seems that we can get around it by setting a fixed number of inner iterations.

We redescribe Algorithm 1 in Algorithm 2, including some of the additional details that are discussed above. In particular, we elaborate the detailed steps of the inner-iteration, which used to be the line 4 of Algorithm 1; recall that this is an implementation of an accelerated gradient descent algorithm.

2.4. Design of the replacement function \(f_t(\beta)\)

This section discusses the design of the surrogate function \(f_t(\beta)\), which is to replace the \(\ell_1\) penalty in the original objective function that is in (2). It is widely acknowledged that, if the objective function is strongly convex and well conditioned, then the gradient descent method can achieve very fast convergence rate. It is also known that the \(\ell_1\) norm (\(\|\beta\|_1\) in our paper) is not strongly convex, while the quadratic function (such as \(\|\beta\|_2^2\)) can be easily proved to be strongly convex. Motivated by these facts, we try to replace the \(\ell_1\) penalty (\(\|\beta\|_1\) in \(F(\beta) = \frac{1}{2n} \|y - X\beta\|_2^2 + \lambda \|\beta\|_1\)) by \(f_t(\beta)\), which is quadratic near 0 and almost linear outside. By making this replacement, the surrogate objective function \(F_t(\beta) = \frac{1}{2n} \|y - X\beta\|_2^2 + \lambda f_t(\beta)\) can achieve strongly convex. It is nontrivial to find a good surrogate function \(f_t(\beta)\). We list the requirements of \(f_t(\beta)\) in Condition 2.2.

**Condition 2.2.** Assume function \(f_t(x)\) satisfies the following conditions.

(a) When \(t \to 0\), we have \(f_0(x) = |x|\), where \(|x|\) is the absolute value function.
(b) For fixed \(t > 0\), function \(x \mapsto f_t(x)\) is quadratic on \([-t, t]\), here \(\mapsto\) indicates that the left hand side (i.e., \(x\)) is the variable in the function in the right hand side (i.e., \(f_t(x)\)). We following this convention in the rest of this paper.
(c) Function \(x \mapsto f_t(x)\) is \(C^1\). Here \(C^1\) is the set of all continuously differentiable functions.
(d) Function \(f_t(x)\) has the second derivative with respect to \(x\).
Algorithm 2: A detailed version of our proposed algorithm

Input: $y_{n \times 1}, X_{n \times p}, \lambda, t_0, h, \epsilon, B$
Output: an estimator of $\beta$, noted as $\beta^{(k)}$, which achieves the $\epsilon$-precision.

1. initialization $t_0, h, k = 1$, $\beta^{(0)} = \left[ X’X + \frac{2n\lambda(\log(1 + t_0))^2}{3\beta} \right]^{-1} X’y$
2. $\triangleright$ Outer-Iteration: $\triangleright$ while $F(\beta^{(k-1)}) - F_{\min} > \epsilon$ do
3. \hspace{1em} $s = 1$
4. \hspace{1em} $\beta^{(k)}[0] = \beta^{(k-1)}$
5. \hspace{1em} $\tilde{\beta}^{(k)}[0] = \beta^{(k-1)}$
6. \hspace{1em} $\tilde{c}_k = \frac{\lambda^2}{\lambda + \beta(t_k)^2}$
7. $\triangleright$ Inner-Iteration: $\triangleright$ while $F_{\tilde{\beta}^{(k)}[s-1]} - F_{\min,k} > \tilde{c}_k$ do
8. \hspace{1em} $\beta^{(k)}[s] = (1 - q_s)\tilde{\beta}^{(k)}[s-1] + q_s\beta^{(k-1)}[s-1]$
9. \hspace{1em} $\beta^{(k)}[s] = \arg\min_{\beta} \{ \gamma_s \left[ \beta' \nabla F_{\tilde{\beta}^{(k)}[s]} + \mu_k V(\beta^{(k-1)}, \beta) \right] + V(\beta^{(k-1)}, \beta) \}$
10. \hspace{1em} $s = s + 1$
11. \hspace{1em} $\beta^{(k)}[s] = \beta^{(k-1)}[s]$
12. \hspace{1em} $t_k = t_{k-1}(1 - h)$
13. \hspace{1em} $k = k + 1$

In line 2 $F_{\min} = \min_{\beta} F(\beta)$.
In line 4 and the rest of this paper, we use parenthesis $(k)$ to denote the $k$th outer-iteration, and we use bracket $[s]$ to denote the $s$th inner-iteration.
In line 7 $F_{\min,k} = \min_{\beta} F_{\tilde{\beta}^{(k)}[s]}$.
In line 8 in this paper, we choose $q_s$ as $q_s = q = \frac{\alpha_k - \mu_k/L_k}{1 - \mu_k/L_k}$ for $s = 1, 2, \ldots$, where $\alpha_k = \sqrt{\frac{\beta}{L_k}}$. And $L_k, \mu_k$ is defined as $\|\nabla F_{\tilde{\beta}^{(k)}}(x) - \nabla F_{\tilde{\beta}^{(k)}}(y)\|_2 \leq L_k \|x - y\|_2$, $F_{\tilde{\beta}^{(k)}}(y) \geq F_{\tilde{\beta}^{(k)}}(x) + \nabla F_{\tilde{\beta}^{(k)}}(y - x) + \frac{\mu_k}{2} \|y - x\|_2^2$.
In line 9 we choose $\gamma_s$ as $\gamma_s = \gamma = \frac{\alpha_k(1 - \alpha_k)}{\alpha_k}$ for $s = 1, 2, \ldots$. Here $V(x,z)$ is defined as $V(x,z) = v(z) - [v(x) + \nabla v(x)'(z - x)]$, with $v(x) = \|x\|_2^2/2$.

Following the requirements in Condition 2, 2, we design $f_t(x)$ in the following equation, where the input variable $x$ is a scalar.

$$f_t(x) = \begin{cases} \frac{1}{3t} \left[ \log(1 + t) \right]^2 x^2, & \text{if } |x| \leq t, \\ \frac{1}{3t} \left[ \log(1 + t) \right]^2 x^2 + \frac{1}{3t} \left[ \log(1 + t) \right]^2 - \frac{1}{t} \left[ \log(1 + t) \right]^2, & \text{otherwise}. \end{cases}$$

Fig. displays this surrogate function $f_t(x)$ and its derivatives when the parameter $t$ takes different values. The first row in Fig. shows the closeness between $f_t(x)$ and $|x|$ when $t$ changes, and the second and third rows present their first and second derivatives, respectively. It can be seen that, when $t \rightarrow 0$, both $f_t(x)$, its first and second derivative become closer to the counterparts of the function $|x|$.

It is also worth noting that when the input variable is a vector instead of a scaler,
Fig. 1. The red solid line in the first, second, third row represents the function $f_t(x)$, its first derivative, and its second derivatives, respectively, under the scenario when $t = 1$, $t = 0.1$ and $t = 0.01$. The blue dashed line in the first, the second, and the third row represents $|x|$, its first derivative, and its second derivatives, respectively, under the same scenarios. For function $|x|$, the first and second derivatives are not defined at the origin. This figure shows the closeness between $f_t(x)$ and $|x|$ when $t$ converges to zero.
for example if we have $\beta = (\beta_1 \cdots \beta_p)'$, then $f_t(\beta)$ can be defined accordingly: $f_t(\beta) = \sum_{i=1}^p f_i(\beta_i)$.

**Remark 2.3.** The design of $f_t(x)$ in equation (6) is not unique but needs to satisfy some special requirements. Generally speaking, we can assume that $f_t(x)$ has the following format:

$$f_t(x) = \begin{cases} d(t)x^2, & \text{if } |x| \leq t, \\ a(t)|x| + b(t)g(x) + c(t), & \text{otherwise.} \end{cases}$$

(6)

The requirement in Condition 2.2 is equivalently transformed into:

(a) both $x \mapsto f_t(x)$ and $t \mapsto f_t(x)$ are $C^1$.

(b) $a(0) = 1$, $b(0) = 0$, $c(0) = 0$, so that $f_0(x) = |x|$.

Besides, we wish the second derivative of $f_t(x)$ has the format of $f''_t(x) = h(t) \max\{t, |x|\}^v$, where $h(t)$ is a function of $t$ and $v$ is a constant. Accordingly, it is reasonable to suppose that $g(x) = \frac{1}{(1-v)(2-v)}x^{2-v}$. Combining all the requests above, one has

$$a(t) = \frac{v}{1+v} t^{1+v} b(t).$$

Since $a(0) = 1$, we choose $b(t) = \frac{1+v}{v} [\log(1+t)]^{1+v}$. Other choice of $b(t)$ can be $\sin(\cdot)$ function or other functions, which makes $t^{1-v} b(t)$ as a constant when $t = 0$.

The purpose of designing $f_t(\beta)$ is to replace the $\ell_1$ penalty ($\|\beta\|_1$) and then shrink $t$ in each iteration $k$, i.e., $t_k = t_{k-1}(1 - h)$, where $h$ is a parameter controlling the shrinking degree of $t$. Because of the first statement in Condition 2.2, the replacement $f_t(\beta)$ gets more and more close to $\|\beta\|_1$ as $t \to 0$.

**Remark 2.4.** Our idea is similar to Bubeck et al. (2018) in appearance, however, the differences are as follows.

(a) Bubeck et al. (2018) aimed at $\ell_p$ penalty, where $p \in \{1, 2, +\infty\}$, while we focus on the $p = 1$, which is not discussed in Bubeck et al. (2018) and the theory in Bubeck et al. (2018) is not easily-extendable to the situation when $p = 1$.

(b) Bubeck et al. (2018) minimizes a linear function instead of the quadratic residual $\frac{1}{2n} \|y - X\beta\|_2^2$, where the Hessian matrix of the objective function needs different treatment.

Suppose that in the $k$th iteration, by replacing $\|\beta\|_1$ with $f_t_k(\beta)$, the objective function of the Lasso problem — $F(\beta) = \frac{1}{2n} \|y - X\beta\|_2^2 + \lambda \|\beta\|_1$ — is transformed into

$$F_{t_k}(\beta) = \frac{1}{2n} \|y - X\beta\|_2^2 + \lambda f_{t_k}(\beta).$$

(7)

The surrogate function $f_{t_k}(x)$ in equation (6) has a property in the following lemma.
LEMMA 2.5. Suppose that from the beginning of our algorithm to the end of our algorithm, we have that $\beta_i^{(k)} \leq B$ for any $i \in \{1, 2, \ldots, p\}$ and $k \in \{1, 2, \ldots\}$. Then for any $k \in \{1, 2, \ldots\}$, the surrogate function defined in equation (5), i.e., $f_t(x)$, has the following property:

$$ f_t(B) - B \leq f_t(x) - |x| \leq 0. \quad (8) $$

PROOF. See Appendix C.2.

3. Order of complexity of the HS Algorithm

This section deals with the order of complexity of the HS algorithm, i.e., how many number of operations needed to achieve the $\epsilon$-precision that is defined in Definition 1.1. Since our HS algorithm involves two layers of iterations — one is for the shrinkage of $t$ (we call it an outer-loop), and the other is the AGD optimization (we call it the inner-loop) — the order of complexity is mainly affected by: (i) the number of inner-iterations, (ii) the number of outer-iterations, (iii) the number of operations in each inner-iteration. To solve these components respectively, we discuss (i) in Section 3.1, and (ii) in Section 3.2. And because (iii) is very similar to that in Algorithm 3, Algorithm 4, Algorithm 5, and Algorithm 6, we will not discuss it separately in a section and will only discuss it briefly in Section 3.3.

3.1. Number of Inner-Iteration

Recall the Line 7 - 11 in Algorithm 2, for a fixed $t_k$, the inner-loop aims at finding an estimator $\beta^{(k)}$, whose precision is shown in the following equation

$$ F_{t_k}(\beta^{(k)}) - F_{\min,k} \leq \tilde{\epsilon}_k, \quad (9) $$

where $\tilde{\epsilon}_k$ is the precision we set for the AGD algorithm on optimizing function $F_{t_k}(\beta) = \frac{1}{2n} \|y - X\beta\|_2^2 + \lambda f_{t_k}(\beta)$ and the value of $\tilde{\epsilon}_k$ will be specified later. We denote $\hat{\beta}^{(k)} = \arg \min_{\beta} \{ \frac{1}{2n} \|y - X\beta\|_2^2 + \lambda f_{t_k}(\beta) \}$, and we have $F_{\min,k} = F(\hat{\beta}^{(k)})$. In the $k$th outer-iteration, an upper bound on the number of inner-iterations that are needed to achieve the $\tilde{\epsilon}_k$ is shown in the following theorem.

THEOREM 3.1 (INNER-LOOP). Recall that a Lasso problem has a response vector $y \in \mathbb{R}^n$ and a model matrix $X \in \mathbb{R}^{n \times p}$. To minimize the Lasso objective function $F(\beta) = \frac{1}{2n} \|y - X\beta\|_2^2 + \lambda \|\beta\|_1$, we design a homotopic approach, i.e., in the $k$th outer-iteration of our proposed algorithm, we use AGD algorithm to minimize a surrogate function $F_{t_k}(\beta) = \frac{1}{2n} \|y - X\beta\|_2^2 + \lambda f_{t_k}(\beta)$. Instead of converging to the minimizer of $F_{t_k}(\beta)$, we do an early stopping to control the total number of numerical operations. And we denote the early stopping estimation as $\beta^{(k)[s]}$, where $s$ is the number of AGD-iterations (inner-iterations). We assume that for any $k = 1, 2, \ldots, s = 1, 2, \ldots$, we have $|\beta_i^{(k)[s]}| \leq B$, where $\beta_i^{(k)[s]}$ is the $i$th entry of vector $\beta^{(k)[s]}$ ($i = 1, 2, \ldots, p$), and $B$ is a constant. And we further assume that our proposed algorithm stops when $t_k < \tau$. Under the above assumption, we know that in the $k$th outer-iteration, the condition number of
function \( F_{t_k}(\cdot) \) can be bounded by 
\[
\frac{3B^3\lambda_{\max}(\frac{\lambda}{\lambda_{\max}})}{2\lambda_{\max}(1+\tau)^2} + \left(\frac{\lambda}{\lambda_{\max}}\right)^3.
\]
Accordingly, after \( C_1 \log(1/\tilde{\epsilon}_k) \) inner-iterations, one is guaranteed to achieve the following precision 
\[
F_{t_k}(\beta^{(k)}) - F_{\min,k} \leq \tilde{\epsilon}_k,
\]
where \( F_{\min,k} = \min_\beta F_{t_k}(\beta) \), \( \tilde{\epsilon}_k = \frac{\lambda p}{3B}[\log(1+t_k)]^2 \) and \( C_1 \) is a constant that does not depend on the value of \( t_k \).

**Proof.** See Appendix C.3.

**Remark 3.2.** Our assumption that the entries of \( \beta \) are bounded by a constant \( B \) is reasonable, as such a condition has appeared widely in the literature. A popular way to justify is that the values have to be manageable in a modern computer, which are restricted by the largest value that can be stored in the corresponding computer platform.

**Remark 3.3.** In the above theorem, we need a condition: \( t_k > \tau \), where \( \tau > 0 \) is a predetermined constant. This condition prevents function \( f_{t_k}(\beta) \) from converging to the function \( \|\beta\| \). In Section 5.1, we will argue that our result applies in the “warm-up” stage of a Lasso-algorithm. That is, when \( t_k \) is small enough, under some conditions, the stopping point of our algorithm provides an estimator that is close enough to the ultimate Lasso estimator; therefore from our estimator, we may reliably estimate the support of the ultimate Lasso estimator, and simply run an ordinary regression on this support set. In this sense, our result finds a “warm start” for solving the Lasso problems, at the same time, achieves a provable faster convergence rate.

### 3.2. Number of Outer-Iteration

This section discusses the minimal number of outer-iterations needed to achieve the \( \epsilon \)-precision defined in Definition 1.1, which explains the Line 2 in Algorithm 2.

**Theorem 3.4 (Number of outer-iteration).** With the conditions in Theorem 3.1 being satisfied, and suppose the following conditions are also satisfied:

(a) For \( k = 1, 2, \ldots \), we have \( t_k = t_0(1-h)^k \) where \( t_0, h \) are pre-specified.
(b) The precision of AGD in minimizing function \( F_{t_k}(\beta) \) is set as \( \tilde{\epsilon}_k = \frac{\lambda p}{3B}[\log(1+t_k)]^2 \), i.e., we run the AGD until the following inequality is achieved: \( F_{t_k}(\beta^{(k)[s]}) - F_{k,\min} < \tilde{\epsilon}_k \), where quantity \( \beta^{(k)[s]} \) is the iterative estimator in the \( s \)-th inner-iteration at the \( k \)-th outer-iteration, and recall that \( F_{k,\min} = \min_\beta F_{t_k}(\beta) \).

Then when \( k \geq \frac{-1}{\log(1-h) \log \left( \frac{\lambda p t_0(2B+1)}{\epsilon} \right)} \), our proposed algorithm finds a point \( \beta^{(k)} \) such that
\[
F(\beta^{(k)}) - F_{\min} \leq \epsilon,
\]
where \( F_{\min} = \min_\beta F(\beta) \) with \( F(\beta) = \frac{1}{2n} \|y - X\beta\|^2 + \lambda \|\beta\|_1 \), which is defined in (2).

**Proof.** The proof is shown in Appendix C.4.
3.3. Order of complexity for HS Algorithm

With all the above blocks, we develop the main theory, i.e., the order of complexity, in this section. Recall that, the definition of order of complexity is the total number of operations needed to achieve the $\epsilon$-precision. The reason for us to adopt the order of complexity instead of the running time is that the order of complexity is independent of (different) computer platforms, while running time possibly depends on different platforms. Consequently, the order of complexity provides a more reliable way for us to compare different algorithms.

**Theorem 3.5 (Main Theory).** Under the conditions that are listed in Theorem 3.4, we can find $\beta^{(k)}$ such that

$$F(\beta^{(k)}) - F_{\min} \leq \epsilon$$

with the number of numerical operations has the order of complexity

$$p^2O\left(\left[ \frac{-1}{\log(1 - h)} \log \left( \frac{\lambda pt_0(2B + 1)}{\epsilon} \right) \right]^2 \right).$$

**Proof.** See Appendix C.5.

4. Numerical Examples

In this section, we compare the performance of HS algorithm with other state-of-the-art algorithms through numerical experiments. As we mentioned in Section 1, there are many Lasso-algorithms. Proximal mapping appears to be a major tool in developing Lasso-algorithms. We take ISTA (Daubechies et al., 2004) as a representative. Starting from the proximal mapping, some Lasso-algorithms utilize the accelerated gradient descent, and develop a faster approach in proximal mapping. For this type of Lasso-algorithms, we select FISTA (Beck and Teboulle, 2009) as a representative. For the third type of Lasso-algorithms, we adopt coordinate descent. And the last type of Lasso-algorithms apply surrogate functions to approximate the $\ell_1$ penalty; here we select SL (Mukherjee and Seelamantula, 2016) as an representative. Based on the theoretically analysis in Section 1 and Appendix A, we see that ISTA (Daubechies et al., 2004), CD (Friedman et al., 2010) and SL (Mukherjee and Seelamantula, 2016) share the same order of complexity, so we will only pick ISTA as a representative of these three. FISTA (Beck and Teboulle, 2009) is also selected as a benchmark since it has the best order of complexity among the existing Lasso-algorithms.

In this section, two numerical examples are shown. The difference between the two simulations lies on the setting of the true parameter $\beta$. In first simulation, the $i$th entry of the true parameter $\beta \in \mathbb{R}^p$ is generated by $\beta_i = (-1)^i \exp(-2(i - 1)/20)$ for $i = 1, \ldots, p$. This style of parameter follows an traditional fashion, which is similar to Friedman et al. (2010). In our second simulation, we set $\beta_i = (-1)^i \exp(-2(i - 1)/20) 1 \{i \leq 10\}$. This parameter setting assumes that most of the entries in $\beta$ is zero, which renders a case with sparse truth.
4.1. Simulation 1

The objective in this numerical example is to explore whether HS has better performance than the benchmarks when estimating the true parameter under the sparse linear regression model. For a fair comparison, the simulation setting and data generation mechanism is similar to Friedman et al. (2010). The difference is that in Friedman et al. (2010), the running time is compared under different simulation setting, while we adopt the number of numeric operations here. (Recall that running time may depend on the platforms, while the number of numerical operations does not.)

The data generation mechanism is as follows. We generate Gaussian data with \( n \) observations and \( p \) covariates, with each predictor is associated with a random vector \( X_j \in \mathbb{R}^n \), and the model matrix is \( X = (X_1, \ldots, X_j, \ldots, X_p) \). Here we assume that the random vector \( X_j \) follows the multivariate normal distribution with zero mean, variances being equal to 1, and identical population correlation \( \rho \), that is, the covariance matrix of \( X_j \) is of the following form:

\[
\begin{pmatrix}
1 & \rho & \cdots & \rho \\
\rho & 1 & \cdots & \rho \\
\vdots & \vdots & \ddots & \cdots \\
\rho & \rho & \cdots & 1
\end{pmatrix}
\]

In this simulation, we set \( \rho = 0.1 \). The response values were generated by

\[ y = \sum_{j=1}^{p} X_j \beta_j + qz, \tag{10} \]

where the \( i \)-th \( (1 \leq i \leq p) \) entry in vector \( \beta = (\beta_1 \cdots \beta_p)' \) is generated by \( \beta_i = (-1)^i \exp(-2(i - 1)/20) \), which are constructed to have alternating signs and to be exponentially decreasing. Besides, \( z = (z_1 \cdots z_p)' \) is the white noise with \( z_i \) satisfying the standard normal distribution \( \mathcal{N}(0, 1) \). Quantity \( q \) is chosen so that the signal-to-noise ratio is 3.0. The turning parameter \( \lambda \) is set to be \( 10^{-3} \). And in our simulation, two scenarios are discussed, where the first scenarios is \( n = 50, p = 20 \) and the second scenario is \( n = 50, p = 80 \).

Table 2 summarizes the numerical results of the number of operations for ISTA, FISTA, and our algorithm to achieve the different \( \epsilon \)-precision. And Fig. 3 visualizes the numerical results in Table 2 where the blue line, red line, and yellow line represent the number of numerical operations of ISTA, FISTA, and our method, respectively. The x-axis is the \( \log(1/\epsilon) \) (Recall \( \epsilon \) in equation 3). And y-axis is the logarithms of the number of numerical operations to achieve the corresponding \( \epsilon \)-precision.

In both two scenarios, i.e., \( n = 50, p = 20 \) and \( n = 50, p = 80 \), there are some common properties of these three methods (ISTA, FISTA, and Ours). Generally speaking, as the precision \( \epsilon \) approaches to 0, it costs more number of numerical operations for the designed algorithms to get the optimizer that achieves the desired precision. Therefore, no matter ISTA, FISTA or our method, the common characteristic of Fig. 3 is that, both three methods have an increasing tendency.

In both two scenarios, i.e., \( n = 50, p = 20 \) and \( n = 50, p = 80 \), there are also some differences among these three methods (ISTA, FISTA, and Ours). Generally speaking,
both ISTA and FISTA requires larger number of numerical operations than that of our method. For example, in the second scenario, when $\epsilon$ is 0.005, our method only requires 36,457 operations, however, ISTA and FISTA need 190,131 and 55,133 operations, respectively. So, it is obvious that our method, compared with the state-of-the-art Lasso-algorithms (where FISTA is the most efficient one), requires less number of numerical operations to achieve the same $\epsilon$-precision. In the first scenario ($n = 50, p = 20$) with large $\epsilon$, the number of numerical operations of the three types of algorithms are very similar, because when $\epsilon$ and $p$ are very small, the hidden constant before the complexity ($O(p^2/\epsilon)$ for ISTA, $O(p^2/\sqrt{\epsilon})$ for FISTA, and $O(p^2/\sqrt{\epsilon})$ for HS) are dominated.

The pattern in Fig. 3 matches our theoretical results. As we have shown in Section 3, the number of numerical operations of ISTA and FISTA are $O(1/\epsilon)$ and $O(1/\sqrt{\epsilon})$, respectively. If we take the logarithm of these two number of numerical operations, then they ought to be $O(\log(1/\epsilon))$ and $O\left(\frac{1}{p}\log(1/\epsilon)\right)$. Therefore, in principle, the slop of ISTA and FISTA in Fig. 3 should be 1 and $\frac{1}{2}$ respectively. To verify this conjecture, we perform a linear regression and find that the slop of the ISTA curve in the right panel of Fig. 3 is 0.71411, and the slop of the FISTA curve in the left panel of Fig. 3 is 0.25211. It should be acknowledged that there is some deviation of the slopes from the theoretically predicted values, when comparing the numerical results with the theoretical analysis. The bias in both two scenarios is because $\epsilon$ is not small enough. If we decrease $\epsilon$ to 0, then the bias would be reduced, because the term related to $\epsilon$ in the complexity, i.e., $O(p^2/\epsilon)$ for ISTA, $O(p^2/\sqrt{\epsilon})$ for FISTA, and $O(p^2/\sqrt{\epsilon})$ for HS, will be dominated. For our proposed algorithm, whose computational complexity is $O\left(\left[\log(1/\epsilon)\right]^2\right)$, its shape in Fig. 3 should be similar to $\log(2\log(x))$ theoretically. Yet, in real practice, it is difficult to achieve this ideal computational complexity, because in each outer-iteration, it is difficult to know exactly when the inner-iteration should stop (see line 7 in Algorithm 2). However, through the optimal computational complexity is hard to achieve in real practice, the computational complexity of our propose algorithm is still lower than that of ISTA and FISTA.

### Table 2. Numerical complexity of ISTA, FISTA, HS in the first simulation

| method | Precision $\epsilon$ | 0.05 | 0.03 | 0.02 | 0.01 | 0.009 | 0.008 | 0.007 | 0.006 | 0.005 |
|--------|----------------------|------|------|------|------|-------|-------|-------|-------|-------|
| ISTA   | 0 = 50, $p = 20$     | 5,070| 6,016| 7,005| 9,585| 10,101| 10,703| 11,434| 12,294| 13,369|
| FISTA  |                     | 4,781| 5,117| 5,453| 6,461| 6,685 | 6,797 | 7,021 | 7,133 | 7,357 |
| Ours   |                     | 5,478| 5,478| 5,479| 5,479| 5,479 | 5,479 | 5,479 | 5,479 | 6,005 |
| ISTA   | 0 = 50, $p = 80$     | 37,400| 50,277| 65,273| 109,772| 119,226| 130,799| 145,306| 164,377| 190,131|
| FISTA  |                     | 31,237| 34,533| 37,417| 45,657| 47,305 | 48,541 | 50,189 | 52,249 | 55,133 |
| Ours   |                     | 30,919| 30,919| 32,765| 34,611| 34,611 | 34,611 | 34,611 | 36,457 | 36,457 |

1 There is the parameters settings of our HS algorithm: $t_0 = 3, h = 0.1, \lambda = 1e-3, \beta^{(0)} = 1_{p \times 1}$.
Fig. 2. Number of Operations of ISTA, FISTA, and our algorithm under different $\epsilon$ in the first simulation.

Fig. 3. Empirical cumulative distribution function (left) and histogram (right) of the 1000 simulations in the first numerical example.
In this section, we discuss another simulation setting different from that one in Section 4.2. Simulation 2

In our theoretical result, we required the presence of a constant $\tau > 0$, such that $t_k \geq \tau$ for all $k$. Such a condition prevents the hyper-parameter $t$ from converge to zero. In Section 5.1, we show that when the $\tau$ is chosen to be small enough, an early-stopped

Table 3. Numerical complexity of ISTA, FISTA, HS in the second simulation

| method | 0.05 | 0.03 | 0.02 | 0.01 | 0.009 | 0.008 | 0.007 | 0.006 | 0.005 |
|--------|------|------|------|------|-------|-------|-------|-------|-------|
| ISTA   | 5,242| 6,274| 7,263| 9,370| 9,757 | 10,187| 10,703| 11,305| 12,122|
| FISTA  | 4,781| 5,229| 5,565| 6,125| 6,349 | 6,461 | 6,573 | 6,797 | 7,021 |
| Ours   | 5,479| 5,479| 5,479| 6,005| 6,005 | 6,005 | 6,005 | 6,005 | 6,005 |

| n = 50, p = 20 |
|----------------|
| ISTA           | 39,519| 55,330| 72,119| 112,869| 120,693| 130,473| 142,698| 158,346| 179,373|
| FISTA          | 31,649| 35,769| 39,065| 45,657 | 46,893 | 48,129 | 49,365 | 51,013 | 53,485 |
| Ours           | 30,918| 32,763| 32,763| 32,763 | 32,763 | 32,763 | 32,763 | 32,763 | 32,763 |

| n = 50, p = 80 |
|----------------|
| ISTA           | 39,519| 55,330| 72,119| 112,869| 120,693| 130,473| 142,698| 158,346| 179,373|
| FISTA          | 31,649| 35,769| 39,065| 45,657 | 46,893 | 48,129 | 49,365 | 51,013 | 53,485 |
| Ours           | 30,918| 32,763| 32,763| 32,763 | 32,763 | 32,763 | 32,763 | 32,763 | 32,763 |

1 The parameters settings of our HS algorithm: $t_0 = 3, h = 0.1, \lambda = 1 - \epsilon - 3, \beta(0) = 0.1_{p \times 1}$
homotopic approach will find the support of the global solution, therefore, one can simply run the ordinary regression on this support set, without losing anything.

In Section 5.2, we discuss other seemingly similar homotopic ideas, and articulate the differences between theirs and the work that is presented in this paper.

5.1. Support Recovery and the Need for Hyper-parameter $t$ to Converge to Zero

In Theorem 3.1, we assume that there is a constant $\tau > 0$, such that $t_k \geq \tau$ for all $k$. Such a condition prevents the hyper-parameter $t$ from converge to zero. Therefore, our result just applies to the warm-up stage of a homotopic approach in solving the Lasso problem. In this subsection, we show that under some standard conditions that have appeared in the literature, as long as we set $\tau$ to be small enough, the associated algorithm will find a solution that both has small “prediction error” and “estimation error”. The mathematical meaning of “prediction error” is

\[
\frac{1}{n} \left\| X \left( \tilde{\beta} - \hat{\beta} \right) \right\|_2^2,
\]

where $\tilde{\beta} = \arg \min_{\beta} \frac{1}{2n} \left\| y - X\beta \right\|_2^2 + \lambda f_t(\beta)$ for a general $t$ and $f_t(\beta)$ defined in (5), and $\hat{\beta} = \arg \min_{\beta} \frac{1}{2n} \left\| y - X\beta \right\|_2^2 + \lambda \left\| \beta \right\|_1$. And the mathematical meaning of “estimation error” in our paper is

\[
\left\| \tilde{\beta} - \hat{\beta} \right\|^2_2.
\]

In the remaining of this section, we will give two propositions, where we develop the conditions where we would have small prediction error and estimation error, respectively.

We begin with the prediction error, i.e., $\frac{1}{n} \left\| X \left( \tilde{\beta} - \hat{\beta} \right) \right\|_2^2$. In the Proposition 5.1, we declare that there is no additional conditions needed to guarantee the small prediction error. That is, as long as we converge $t \to 0$, our proposed algorithm can guarantee the prediction error goes to zero as well.
Proposition 5.1. For our proposed algorithm, when $t \to 0$, we have the prediction error $\frac{1}{n} \left\| X \left( \tilde{\beta} - \bar{\beta} \right) \right\|^2_2 \to 0$, where $\tilde{\beta} = \text{arg min}_\beta \frac{1}{2n} \left\| y - X\beta \right\|^2_2 + \lambda f_t(\beta)$ for a general $t$ and $f_t(\beta)$ defined in (5). And $\hat{\beta} = \text{arg min}_\beta \frac{1}{2n} \left\| y - X\beta \right\|^2_2 + \lambda \left\| \beta \right\|_1$.

Proof. See Appendix C.6.

After developing the prediction error, we now discuss the estimation error. A nice property of Lasso is that, it can potentially achieve the sparse estimation when $n < p$, i.e., most of the entries in the Lasso estimator $\hat{\beta}$ are zero, and only few of them are non-zero. The index set of these non-zero entries are called support set, i.e., $S = \{ i : \hat{\beta}_i \neq 0 \forall i = 1, 2, \ldots, p \}$. To show how Lasso can realize the sparse estimation, we take $X = I$ as an illustration example, where $I$ is the identity matrix. (More complicated model matrix $X$ can also be used, but here we use $X = I$ to create an example.) Then we have the linear regression model as

$$y = \beta + w,$$

where $y$ is the response vector, and $w$ is the white noise. The Lasso estimator of the above linear regression model is

$$\hat{\beta} = \text{arg min}_\beta \frac{1}{2n} \left\| y - X\beta \right\|^2_2 + \lambda \left\| \beta \right\|_1.$$

It can be verified that

$$\tilde{\beta}_i = \begin{cases} \text{sign}(y_i)(|y_i| - n\lambda), & \text{if } |y_i| > n\lambda; \\ 0, & \text{otherwise}, \end{cases}$$

is the solution of the Lasso problem. Note that $\tilde{\beta}$ is sparse if $y$ has many components with small magnitudes. However, if we consider $f_t(\beta)$, instead of the $\ell_1$ penalty $\| \beta \|_1$, we have

$$\bar{\beta} = \text{arg min}_\beta \frac{1}{2n} \left\| y - X\beta \right\|^2_2 + \lambda f_t(\beta).$$

We can show that $\tilde{\beta}_i = 0$ if and only if $y_i = 0$. This shows that $\tilde{\beta}$ is not guaranteed to be sparse.

Although $\tilde{\beta}$ is not sparse, we can still verify that $\tilde{\beta}$ has very small estimation error under some specific assumptions of the model matrix $X$.

Proposition 5.2. Suppose the model matrix $X$ in the Lasso problem has the following three properties:

1. $\left\| (X'_S X_S)^{-1} X'_S \right\|_F$ can be bounded by a constant, where $S = \{ i : \tilde{\beta}_i \neq 0, \forall i = 1, 2, \ldots, p \}$ with $\tilde{\beta} = \frac{1}{n} \left\| y - X\beta \right\|^2_2 + \lambda \left\| \beta \right\|_1$. And $\left\| \cdot \right\|_F$ is the Frobenius norm defined as $\left\| A_{m \times n} \right\|_F = \sqrt{\sum_{i=1}^m \sum_{j=1}^n |a_{ij}|^2}$, where $a_{ij}$ is the $(i, j)$th entry in matrix $A$. 


(b) $\|X_{sc}^\dagger\|_F$ can be bounded by a constant, where $S^c$ is the complement set of $S$. And $X_{sc}^\dagger$, is the pseudo-inverse of matrix $X_{sc}$. The mathematical meaning of pseudo-inverse is that, suppose $X_{sc} = U\Sigma V$, which is the singular value decomposition (SVD) of $X_{sc}$. Then $X_{sc}^\dagger = V^\prime \Sigma^\dagger U^\prime$. For the rectangular diagonal matrix $\Sigma$, we get $\Sigma^\dagger$ by taking the reciprocal of each non-zero elements on the diagonal, leaving the zeros in place, and then transposing the matrix.

(c) $\sigma_{\text{max}}(\Sigma_1) < \min \{2, 2\sigma_{\text{min}}(\Sigma_2)\}$, where $\sigma_{\text{max}}(\Sigma_1)$ returns the maximal absolute diagonal values of matrix $\Sigma_1$, and $\sigma_{\text{min}}(\Sigma_2)$ returns the minimal absolute diagonal values of matrix $\Sigma_2$. Matrix $\Sigma_1$ is the diagonal matrix in the SVD of matrix $(X_S^\dagger X_S)^{-1} X_S^\dagger X_S + (X_{sc}^\dagger X_{sc})'$, i.e., $(X_S^\dagger X_S)^{-1} X_S^\dagger X_S + (X_{sc}^\dagger X_{sc})' = U_1 \Sigma_1 V_1$. Matrix $\Sigma_2$ is the diagonal matrix of the SVD of matrix $\frac{1}{2}X_{sc}^\dagger X_{sc} + \frac{1}{2}(X_{sc}^\dagger X_{sc})'$, i.e., $\frac{1}{2}X_{sc}^\dagger X_{sc} + \frac{1}{2}(X_{sc}^\dagger X_{sc})' = U_2 \Sigma_2 V_2$.

Then we have $\|\tilde{\beta} - \hat{\beta}\|_2^2 \to 0$ when $t \to 0$.

**Proof.** See Appendix C.7.

We notice that the above proposition requires a strong condition on the model matrix $X$ in order to achieve the support recovery. Releasing the conditions in the above proposition is an interesting future research topic.

### 5.2. Other Related Homotopic Ideas

It is worth noting that, in the recent research, some researchers also realize the log-polynomial order of complexity (see Xiao and Zhang (2013); Liu and Xiao (2014); Wang et al. (2014); Zhao et al. (2018); Pang et al. (2017)) in a framework similar to Lasso-algorithms. However, we would like to clarify that, there are some essential differences between our paper and these papers. First, the problem formulation in these papers is totally different from ours. The problem formulation these papers solve is that, they start at some initial objective function:

$$\frac{1}{2n}\|y - X\beta\|_2^2 + \lambda^{(0)}\|\beta\|_1,$$

and then they gradually decrease the large $\lambda^{(0)}$ until the target regularization $\lambda^{(\text{target})}$ is reached. When the $\lambda^{(\text{target})}$ is reached, the algorithm is stopped. However, this algorithmic solution is not the optimal in (11). In other words, the solution of these papers is not exactly the Lasso solution. While in our paper, our objective function stay the same as (2) from the beginning to the end of our algorithm. Therefore, the solution we iteratively calculated is the minimizer of the Lasso problem in (2). In addition to the difference of the objective function, the assumptions between our algorithm and these papers are also different. Specifically, these papers require more additional assumptions than us, such as the restricted isometry property (RIP), which is used to ensure that the all solution path is sparse. Finally, through both our paper and these papers are called
“homotopic” method, the definition of the “homotopic” is different. Specifically, these papers use the homotopic path in the penalty parameter $\lambda$: they start from a very large $\lambda$ and then shrinkage to the target $\lambda$. This type of method is also called “path following” in other papers, such as [Rosset and Zhu (2007), Allen (2013), Allen et al. (2013)] and etc, instead of “homotopic path”. However, our paper use the homotopic path in the $\ell_1$ penalty $\lambda \| \beta \|_1$: we replace the $\ell_1$ regularization term with a surrogate function, and then by adjusting the parameters in the surrogates, to get the surrogate approximates more close to the original $\ell_1$ regularization term.

References
Allen, G. I. (2013). Automatic feature selection via weighted kernels and regularization. *Journal of Computational and Graphical Statistics* 22(2), 284–299.

Allen, G. I., C. Peterson, M. Vannucci, and M. Maletić-Savatić (2013). Regularized partial least squares with an application to nmr spectroscopy. *Statistical Analysis and Data Mining: The ASA Data Science Journal* 6(4), 302–314.

Beck, A. and M. Teboulle (2009). A fast iterative shrinkage-thresholding algorithm for linear inverse problems. *SIAM journal on imaging sciences* 2(1), 183–202.

Beck, A. and L. Tetruashvili (2013). On the convergence of block coordinate descent type methods. *SIAM journal on Optimization* 23(4), 2037–2060.

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

Boyd, S. and L. Vandenberghe (2016). Numerical linear algebra background.

Brègman, L. M. (1966). Relaxation method for finding a common point of convex sets and its application to optimization problems. In *Doklady Akademii Nauk*, Volume 171, pp. 1019–1022. Russian Academy of Sciences.

Bubeck, S., M. B. Cohen, Y. T. Lee, and Y. Li (2018). An homotopy method for $\ell_p$ regression provably beyond self-concordance and in input-sparsity time. In *Proceedings of the 50th Annual ACM SIGACT Symposium on Theory of Computing*, pp. 1130–1137. ACM.

Daubechies, I., M. Defrise, and C. De Mol (2004). An iterative thresholding algorithm for linear inverse problems with a sparsity constraint. *Communications on Pure and Applied Mathematics: A Journal Issued by the Courant Institute of Mathematical Sciences* 57(11), 1413–1457.

Figueiredo, M. A. (2003). Adaptive sparseness for supervised learning. *IEEE transactions on pattern analysis and machine intelligence* 25(9), 1150–1159.

Friedman, J., T. Hastie, H. Höfling, R. Tibshirani, et al. (2007). Pathwise coordinate optimization. *The annals of applied statistics* 1(2), 302–332.
Friedman, J., T. Hastie, and R. Tibshirani (2010). Regularization paths for generalized linear models via coordinate descent. *Journal of statistical software* 33(1), 1.

Fu, W. J. (1998). Penalized regressions: the bridge versus the lasso. *Journal of computational and graphical statistics* 7(3), 397–416.

Ghadimi, S. and G. Lan (2016). Accelerated gradient methods for nonconvex nonlinear and stochastic programming. *Mathematical Programming* 156(1-2), 59–99.

Hestenes, M. R. (1969). Multiplier and gradient methods. *Journal of optimization theory and applications* 4(5), 303–320.

Hildreth, C. (1957). A quadratic programming procedure. *Naval research logistics quarterly* 4(1), 79–85.

Hunter, D. R. and R. Li (2005). Variable selection using mm algorithms. *Annals of statistics* 33(4), 1617.

Lan, G. (2019). Lectures on optimization. methods for machine learning. *H. Milton Stewart School of Industrial and Systems Engineering, Georgia Institute of Technology, Atlanta, GA*.

Lange, K., D. R. Hunter, and I. Yang (2000). Optimization transfer using surrogate objective functions. *Journal of computational and graphical statistics* 9(1), 1–20.

Li, H. and Z. Lin (2015). Accelerated proximal gradient methods for nonconvex programming. In *Advances in neural information processing systems*, pp. 379–387.

Lin, Q. and L. Xiao (2014). An adaptive accelerated proximal gradient method and its homotopy continuation for sparse optimization. In *International Conference on Machine Learning*, pp. 73–81.

Mukherjee, S. and C. S. Seelamantula (2016). Convergence rate analysis of smoothed Lasso. In *Communication (NCC), 2016 Twenty Second National Conference on*, pp. 1–6. IEEE.

Nesterov, Y. (2005). Smooth minimization of non-smooth functions. *Mathematical programming* 103(1), 127–152.

Nesterov, Y. (2013). Gradient methods for minimizing composite functions. *Mathematical Programming* 140(1), 125–161.

Nesterov, Y. E. (1983). A method for solving the convex programming problem with convergence rate o (1/k^2). In *Dokl. akad. nauk Sssr*, Volume 269, pp. 543–547.

Pang, H., H. Liu, R. J. Vanderbei, and T. Zhao (2017). Parametric simplex method for sparse learning. In *Advances in Neural Information Processing Systems*, pp. 188–197.

Polson, N. G., J. G. Scott, B. T. Willard, et al. (2015). Proximal algorithms in statistics and machine learning. *Statistical Science* 30(4), 559–581.
A. Review of some State-of-the-art Algorithms

In this section, we will show the algorithm mechanism of these four representative we select, namely ISTA (Daubechies et al., 2004) in Section A.1, FISTA Beck and Teboulle (2009) in Section A.2, CD (Friedman et al., 2010) in Section A.3, and SL (Mukherjee and Seelamantula, 2016) in Section A.4. For each algorithm, we show (i) their number of operations in an iteration, (ii) the number of iterations to meet the $\epsilon$-precision in equation (3), (iii) and their according order of complexity.

A.1. Iterative Shrinkage-Thresholding Algorithms (ISTA)

ISTA aims at the minimization of a summation of two functions, $g + f$, where the first function $g : \mathbb{R}^p \rightarrow \mathbb{R}$ is continuous convex and the other function $f : \mathbb{R}^p \rightarrow \mathbb{R}$ is smooth convex with a Lipschitz continuous gradient. Recall the definition of Lipschitz continuous gradient as follows:

$$\|\nabla f(x) - \nabla f(y)\|_2 \leq L\|x - y\|_2.$$
If we let \( g(\beta) = \lambda \|\beta\|_1 \) and \( f(\beta) = \frac{1}{2n} \| Y - X\beta \|_2^2 \) with the Lipschitz continuous gradient \( L \) taking the largest eigenvalue of matrix \( X'X/n \), noted as \( \sigma_{\text{max}}(X'X/n) \), then Lasso is a special case of ISTA.

The key point of ISTA lies in the updating rule from \( \beta^{(k)} \) to \( \beta^{(k+1)} \), i.e., \( \beta^{(k)} \rightarrow \beta^{(k+1)} \). It is realized by updating \( \beta^{(k+1)} \) through the quadratic approximation function of \( f(\beta) \) at value \( \beta^{(k)} \):

\[
\beta^{(k+1)} = \arg \min_{\beta} f(\beta^{(k)}) + \langle \beta - \beta^{(k)}, \nabla f(\beta^{(k)}) \rangle + \frac{\sigma_{\text{max}}(X'X/n)}{2} \| \beta - \beta^{(k)} \|_2^2 + \lambda \|\beta\|_1. \tag{12}
\]

Simple algebra shows that (ignoring constant terms in \( \beta \)), minimization of equation (12) is equivalent to the minimization problem in the following equation:

\[
\beta^{(k+1)} = \arg \min_{\beta} \frac{\sigma_{\text{max}}(X'X/n)}{2} \left( \| \beta - \beta^{(k)} - \frac{1}{\sigma_{\text{max}}(X'X/n)} (X'X\beta^{(k)} - X'y) \|_2^2 + \lambda \|\beta\|_1 \right), \tag{13}
\]

where the soft-thresholding function in equation (14) can be used to solve the problem in equation (13):

\[
S(x, \alpha) = \begin{cases} 
    x - \alpha, & \text{if } x \geq \alpha, \\
    x + \alpha, & \text{if } x \leq -\alpha, \\
    0, & \text{otherwise.}
\end{cases} \tag{14}
\]

The summary of ISTA algorithm is presented in Algorithm 3.

**Algorithm 3: Iterative Shrinkage-Thresholding Algorithms (ISTA)**

- **Input:** \( y_{n \times 1}, X_{n \times p}, L = \sigma_{\text{max}}(X'X/n) \)
- **Output:** an estimator of \( \beta \) satisfies the \( \epsilon \)-precision, noted as \( \beta^{(k)} \)

1. **initialization;**
2. \( \beta^{(0)}, k = 0 \)
3. **while** \( F(\beta^{(k)}) - F(\hat{\beta}) > \epsilon \) **do**
4. \( \beta^{(k+1)} = S(\beta^{(k)} - \frac{1}{\sigma_{\text{max}}(X'X/n)} X'X\beta^{(k)} - X'y), \lambda/L \)
5. \( k = k + 1 \)

It can be seen from line 4 in Algorithm 3 that the number of operations in one iteration of ISTA is \( O(p^2) \). This is because that the main computation of each iteration in ISTA is the matrix multiplication in \( X'X\beta^{(k)} \). Note that the matrix \( X'X \) can be pre-calculated and saved, therefore, the order of computational complexity is \( p(2p - 1) \) (Boyd and Vandenberghe, 2010).

In addition to the operations in each iteration, we also develop the convergence analysis of ISTA in the following equation (Beck and Teboulle, 2009, Theorem 3.1). To make it more clear, we list (Beck and Teboulle, 2009, Theorem 3.1) below with several changes of notation. The notations are changed to be consistent with the terminology that are used in this paper.

**Theorem A.1.** Let \( \{ \beta^{(k)} \} \) be the sequence generated by Line 4 in Algorithm 3. Then for any \( k \geq 1 \), we have

\[
F(\beta^{(k)}) - F(\hat{\beta}) \leq \frac{\sigma_{\text{max}}(X'X/n) \|\beta^{(0)} - \hat{\beta}\|_2^2}{2k}. \tag{15}
\]
Therefore, to achieve the $\epsilon$-precision, i.e., $F(\beta^{(k)}) - F(\hat{\beta}) \leq \epsilon$, at least $\frac{2\sigma_{\max}(X'X/n)\|\beta^{(0)} - \hat{\beta}\|^2}{2\epsilon}$ iterations are required, which leads to the order of complexity $O(\frac{\sigma_{\max}(X'X/n)\|\beta^{(0)} - \hat{\beta}\|^2 p^2}{2\epsilon}) = O(p^2/\epsilon)$.

A.2. Fast Iterative Shrinkage-Thresholding Algorithms (FISTA)

Motivated by ISTA, Beck and Teboulle (2009) developed another algorithm called Fast Iterative Shrinkage-Thresholding Algorithms (FISTA). The main difference of ISTA and FISTA is that FISTA employs an auxiliary variable $\alpha^{(k)}$ to update from $\beta^{(k)}$ to $\beta^{(k+1)}$ in the second-order Taylor expansion step (i.e., the one in equation (12)); More specifically, they have the same number of operation in one iteration, FISTA has improved convergence rate than ISTA, which is shown in the following theorem (Beck and Teboulle, 2009, Theorem 4.4).

**Algorithm 4:** Fast Iterative Shrinkage-Thresholding Algorithms (FISTA)

**Input:** $y_{n \times 1}, X_{n \times p}, L = \sigma_{\max}(X'X/n)$

**Output:** an estimator of $\beta$, noted as $\beta^{(k)}$, which satisfies the $\epsilon$-precision.

1. initialization;
2. $\beta^{(0)}, t_1 = 1, k = 0$
3. while $F(\beta^{(k)}) - F(\hat{\beta}) > \epsilon$ do
4. $\beta^{(k)} = S(\alpha^{(k)} - \frac{1}{\mu L}(X'X\alpha^{(k)} - X'y), \lambda / L)$
5. $t_{k+1} = \frac{1+\sqrt{1+4t_k^2}}{2}$
6. $\alpha^{(k+1)} = \beta^{(k)} + \frac{t_k - 1}{t_{k+1}}(\beta^{(k)} - \beta^{(k-1)})$
7. $k = k + 1$

Obviously, the main computational effort in both ISTA and FISTA remains the same, namely, in the soft-thresholding operation of line 4 in Algorithm A and [3]. The number of operations in each iterations of FISTA is still $O(p^2)$. Although for both ISTA and FISTA, they have the same number of operation in one iteration, FISTA has improved convergence rate than ISTA, which is shown in the following theorem (Beck and Teboulle, 2009, Theorem 4.4).

**Theorem A.2.** Let $\{\alpha^{(k)}\}$, $\{\beta^{(k)}\}$ be a sequence generated by Line 3 and Line 4 in Algorithm 4, respectively. Then for any $k \geq 1$, we have that

$$F(\beta^{(k)}) - F(\hat{\beta}) \leq \frac{2\sigma_{\max}(X'X/n)\|\beta^{(0)} - \hat{\beta}\|^2}{(k+1)^2}.$$
Consequently, FISTA has a faster convergence rate than ISTA, which improves from $O(1/k)$ to $O(1/k^2)$. This is because that, to update from $\beta^{(k-1)}$ to $\beta^{(k)}$, ISTA only considers $\beta^{(k-1)}$, however, FISTA takes both $\beta^{(k-1)}$ and $\beta^{(k-2)}$ into account. To achieve the precision $F(\beta^{(k)}) - F(\hat{\beta}) \leq \epsilon$, at least $\frac{2\sigma_{\max}(X'X/n)\|\beta^{(0)} - \hat{\beta}\|^2}{\sqrt{\epsilon}}$ iterations are required, which leads to an order of complexity of $O\left(\frac{2\sigma_{\max}(X'X/n)\|\beta^{(0)} - \hat{\beta}\|^2}{\sqrt{\epsilon}}p^2\right) = O(p^2/\sqrt{\epsilon})$.

### A.3. Coordinate Descent (CD)

The updating rule in both ISTA and FISTA involve all coordinates simultaneously. In contrast, Friedman et al. (2010) proposed a Lasso-algorithm that cyclically chooses one coordinate at a time and performs a simple analytical update. Such an approach is called coordinate gradient descent.

The updating rule (from $\beta^{(k)}$ to $\beta^{(k+1)}$) in CD is that, it optimizes with respect to only the $j$th entry of $\beta^{(k+1)}$ ($j = 1, \cdots, p$) where the gradient at $\beta^{(k)}_j$ in the following equation is used for the updating process:

$$\frac{\partial}{\partial \beta_j} F(\beta^{(k)}) = \frac{1}{n} \left( e_j' X X \beta^{(k)} - y' X e_j \right) + \lambda \text{sign}(\beta_j)$$

where $e_j$ is a vector of length $p$, whose entries are all zero expect that the $j$th entry is equal to 1. Imposing the gradient in equation (18) to be 0, we can solve for $\beta^{(k+1)}_j$ as follows:

$$\beta^{(k+1)}_j = S \left( y' X e_j - \sum_{l \neq j} (X'X)_{jl} \beta^{(k)}_l, n\lambda \right) / (X'X)_{jj},$$

where $S(\cdot)$ is the soft-thresholding function defined in equation (14). This algorithm has been implemented into the R package, glmnet, and we summarize it in Algorithm 5.

### Algorithm 5: Coordinate Descent (CD)

**Input:** $y_{n \times 1}$, $X_{n \times p}$, $\lambda$

**Output:** an estimator of $\beta$, noted as $\beta^{(k)}$, which satisfies the $\epsilon$-precision.

**1 initialization:**

**2** $\beta^{(0)}, k = 0$

**3** while $F(\beta^{(k)}) - F(\hat{\beta}) > \epsilon$ do

**4** **for** $j = 1 \cdots p$ **do**

**5** **[**

$$\beta^{(k+1)}_j = S \left( y' X e_j - \sum_{l \neq j} (X'X)_{jl} \beta^{(k)}_l, n\lambda \right) / (X'X)_{jj},$$

**[**

After reviewing the algorithm of CD, we develop the order of complexity of CD. Firstly, the number of operations in each iteration of CD is $O(p^2)$. It can be explained by the following two reasons. (i) While updating $\beta^{(k+1)}_j$ (line 5 in Algorithm 5), it costs $O(p)$ operations because of $\sum_{l \neq j} (X'X)_{jl} \beta^{(k)}_l$. (ii) From line 4 in Algorithm 5, we can see that all $p$ entries of $\beta^{(k+1)}$ are updated one by one. Combining (i) and (ii), we can see that the number of operations need in one iteration of CD is of the order $O(p^2)$.
The convergence rate of CD is derived as a corollary in [Beck and Tetruashvili, 2013, Corollary 3.8] and here we list the corollary as a theorem below. We changed several notations to adopt the terminology in this paper:

**Theorem A.3.** Let \( \{ \beta(k) \} \) be the sequence generated by the Line 5 in Algorithm 5. Then we have that

\[
F(\beta(k)) - F(\hat{\beta}) \leq \frac{4\sigma_{\text{max}}(X'X/n)(1 + p)\|\beta^{(0)} - \hat{\beta}\|_2^2}{k + (8/p)}.
\]

(19)

The above equation shows that, to achieve the precision \( \epsilon \)-precision, at least \( \frac{4\sigma_{\text{max}}(X'X/n)(1 + p)\|\beta^{(0)} - \hat{\beta}\|_2^2}{\epsilon} - \frac{8}{p}p^2 \) iterations are required, which leads to an order of complexity of \( O\left(\frac{p^2}{\epsilon} - 8p\right) = O\left(\frac{p^2}{\epsilon}\right) \).

### A.4. Smooth Lasso (SL)

The aforementioned Lasso-algorithms all aim exactly at minimizing the function \( F(\beta) \). On the contrary, Mukherjee and Seelamantula (2016) used an approximate objective function to solve the Lasso. Their method is called a Smooth-Lasso (SL) algorithm. The main idea of SL is that it uses a smooth function—\( \phi_\alpha(u) = \frac{2}{\alpha}u \log(1 + e^{\alpha u}) - u \)—to approximate the \( \ell_1 \) penalty, and Accelerated Gradient Descent (AGD) algorithm is applied after the replacement. Therefore, the objective function of SL becomes \( F_\alpha(\beta) = \frac{1}{2n}\|y - X\beta\|_2^2 + \lambda \sum_{i=1}^p \phi_\alpha(\beta_i) \). The pseudo code of SL is displayed in Algorithm 6.

**Algorithm 6: Smooth Lasso (SL)**

Input: \( y_{n \times 1}, X_{n \times p}, \mu = \left[ \sigma_{\text{max}}^2(X'X/n) + \lambda \alpha / 2 \right]^{-1} \)

Output: an estimator of \( \beta \), noted as \( \beta^{(k)} \), which satisfies the \( \epsilon \)-precision.

1. initialization;
2. \( \beta^{(0)} \), \( k = 0 \)
3. while \( F(\beta^{(k)}) - F(\hat{\beta}) > \epsilon \) do
4. \( w^{(k+1)} = \beta^{(k)} + \frac{k-1}{k+1}(\beta^{(k)} - \beta^{(k-1)}) \)
5. \( \beta^{(k+1)} = w^{(k+1)} - \mu \nabla F_\alpha(w^{(k)}) \)
6. \( k = k + 1 \)

For the computational effort, it mainly lies in the calculation of \( \nabla F_\alpha(w) = \frac{X'X}{n}w - \frac{X'y}{n} + v \), where the \( v \) is a vector of length \( p \), whose \( i \)th entry is \( \frac{-2}{w_i} \log(1 + e^{\alpha w_i}) + \frac{2e^{\alpha w_i}}{w_i(1 + e^{\alpha w_i})} - 1 \). Accordingly, the main computational effort of each iteration of SL is the matrix multiplication in \( X'Xw^{(k)} \), which cost \( O(p^2) \) operations. On the other side, proved by Mukherjee and Seelamantula (2014), the approximation error of \( \beta^{(k)} \) in SL is shown in equation (20).
Theorem A.4. Let \( \{ \beta(k) \} \) be a sequence generated as in Line 5 of Algorithm 6. Then we have

\[
F(\beta(k)) - F(\hat{\beta}) \leq \frac{4\|\beta(0) - \hat{\beta}\|^2_2 \sigma_{\max}(X)}{k^2} + \frac{4\sqrt{2\lambda n \log 2} \|\beta(0) - \hat{\beta}\|_2}{k}.
\]

(20)

So to achieve the \( \epsilon \)-precision, SL needs \( O(1/\epsilon) \), which results in the order of complexity \( O(p^2/\epsilon) \).

B. An Important Theorem

Our proof will rely on a result on the number of steps in achieving certain accuracy in using the accelerate gradient descent (AGD) when the objective function is strongly convex. The result is the Theorem 3.7 in Lan (2019). We represent the theorem here for readers’ convenience. We introduce some notations first. Suppose one wants to minimize a convex function \( f: X \to \mathbb{R} \) in a feasible closed convex set \( X \in \mathbb{R}^p \). We further assume that \( f \) is a differentiable convex function with Lipschitz continuous gradients \( L \), i.e., \( \forall x, y \in X \), we have

\[
\| \nabla f(x) - \nabla f(y) \|_2 \leq L \| x - y \|_2,
\]

where \( \nabla f(x) \) represents the gradient of function \( f(x) \). Furthermore, we assume that \( f \) is a strongly convex function, i.e., \( \forall x, y \in X \), there exist \( \mu > 0 \), such that we have

\[
f(y) \geq f(x) + \nabla f(x)(y - x) + \frac{\mu}{2} \| y - x \|^2_2.
\]

This type of function \( f \) is called the \( L \)-smooth and \( \mu \)-strongly convex function. Recall that our objective is to solve the following problem:

\[
\min_{x \in X} f(x).
\]

In the following, we present one version of the accelerated gradient descent (AGD) algorithm. Given \( (x^{(t-1)}, \bar{x}^{(t-1)}) \in X \times X \) for \( t = 1, 2, \ldots \), we set

\[
\bar{x}^{(t)} = (1 - q_t)\bar{x}^{(t-1)} + q_t x^{(t-1)}
\]

(21)

\[
x^{(t)} = \arg \min_{x \in X} \left\{ \gamma_t \left[ x' \nabla f \left( \bar{x}^{(t)} \right) + \mu V \left( \bar{x}^{(t)}, x \right) \right] + V \left( x^{(t-1)}, x \right) \right\}
\]

(22)

\[
\bar{x}^{(t)} = (1 - \alpha_t)\bar{x}^{(t-1)} + \alpha_t x^{(t)},
\]

(23)

for some \( q_t \in [0,1], \gamma_t \geq 0 \), and \( \alpha_t \in [0,1] \). And here \( V(x, z) \) is the prox-function (or Bregman’s distance), i.e.,

\[
V(x, z) = v(z) - v(x) + (z - x)' \nabla v(x),
\]

with \( v(x) = \| x \|^2_2 / 2 \). By applying AGD as shown in (21)-(23), the following theorem presents an inequality that can be utilized to determine the number of iterations when certain precision of a solution is given.
THEOREM B.1. Let \((x(t), x(t), \bar{x}(t)) \in X \times X \times X\) be generated by accelerated gradient descent method in (21). If \(\alpha = \alpha, \gamma = \gamma\) and \(q = q\), for \(t = 1, \ldots, k\), satisfy \(\alpha \geq q\), \(L(\alpha - q) \leq \mu, L(\alpha - q) \leq \frac{1}{q}\), and \(\frac{1}{\gamma(1 - \alpha)} \leq \mu + \frac{1}{\gamma}\), then for any \(x \in X\), we have
\[
f(\bar{x}(k)) - f(x) + \alpha \left(\mu + \frac{1}{\gamma}\right) V(x^{(k-1)}, x) \leq (1 - \alpha)^k \left[f(\bar{x}(0)) - f(x) + \alpha \left(\mu + \frac{1}{\gamma}\right) V(x^{(1)}, x)\right].
\]
In particular, if
\[
\alpha = \sqrt{\frac{\mu}{L}}, q = \frac{\alpha - \mu/L}{1 - \mu/L}, \gamma = \frac{\alpha}{\mu(1 - \alpha)},
\]
then for any \(x \in X\), we have
\[
f(\bar{x}(k)) - f(x) + \alpha \left(\mu + \frac{1}{\gamma}\right) V(x^{(k-1)}, x) \leq \left(1 - \sqrt{\frac{\mu}{L}}\right)^k \left[f(\bar{x}(0)) - f(x) + \alpha \left(\mu + \frac{1}{\gamma}\right) V(x^{(1)}, x)\right].
\]

The above theorem gives a convergence rate of AGD under the scenario when the objective function is strongly convex. This result will be utilized in the proof of Theorem 3.1, which can be found in Appendix C.3.

C. Proofs

C.1. Proof of a Lemma

The proof of Lemma 2.4 is as follows.

PROOF. In this proof, we will do two parts.

First, we will prove the existence of the initial point \(t_0\) stated in (4). We know that
\[
\lim_{t \to +\infty} \frac{\sum_{j=1}^p M(t)_{ij}(X'y/n)_j}{t} = \lim_{t \to +\infty} \sum_{j=1}^p \left(\left[\frac{X'X}{n} + \frac{\lambda \log(1 + t)^2}{3t^2}\right]^{-1}\right)_{ij} \left(\frac{X'y}{n}\right)_j \frac{1}{t}
\]
\[
= \lim_{t \to +\infty} \sum_{j=1}^p \left(\left[\frac{X'X}{n} + \frac{\lambda \log(1 + t)^2}{3t^2}\right]^{-1}\right)_{ij} \left(\frac{X'y}{n}\right)_j
\]
\[
= 0.
\]
The above indicates that when \(t\) is very large, the \(t_0\) defined in (4) will exist.

Next, we will verify that, if \(t_0\) is chosen as shown in (4), i.e,
\[
t_0 \in \left\{t : \sum_{j=1}^p M(t)_{ij}(X'y/n)_j \leq t, \forall i = 1, \ldots, p\right\},
\]
we have \(|\beta^{(0)}_i| < t_0\). It can be verified that,
\[
M(t)\frac{X'y}{n} = \arg \min_{\beta} \left\{\frac{1}{2n}\|y - X\beta\|^2 + \frac{1}{3t^2} \left[\log(1 + t)^2 \beta'\beta\right]\right\}, \quad (25)
\]
where \( G(\beta) \) is a special case of \( F_t(\beta) \) when \( t \) is large enough to include all the coefficient \( \beta_i \) into the interval \([-t, t] \). Utilizing the above fact that the minimizer in (25) when \( t = t_0 \) satisfies the condition that its coordinates are within \([-t_0, t_0] \), we have

\[
|\beta_i(t_0)| = \left| \left( M(t_0) \frac{X'y}{n} \right)_i \right| = \left| \sum_{j=1}^{p} M(t_0)_{ij} \left( X'y/n \right)_j \right| \leq t_0.
\]

Thus, if we choose \( t_0 \) as shown in (4), i.e.,

\[
t_0 \in \left\{ t : \left| \sum_{j=1}^{p} M(t)_{ij} \left( X'y/n \right)_j \right| \leq t, \forall i = 1, \ldots, p \right\},
\]

we can verify that \( \forall i = 1, 2, \ldots, p, |\beta_i(t_0)| \leq t_0 \), i.e., \( |\beta_i^{(0)}| \leq t_0 \).

\[\square\]

### C.2. Proof of a Lemma

The Proof of Lemma 2.5 is as follows.

**Proof.** Because \( f_t(x) \) is a even function, we only consider the positive \( x \) in the remaining of the proof.

First, when \( 0 \leq x \leq t \), one has

\[
f_t(x) - x = \frac{1}{3t^3} (\log(1 + t))^2 x^2 - x,
\]

which is a quadratic function with the axis of symmetry, \( \frac{3t^3}{2(\log(1 + t))^2} \) being larger than \( t \). Therefore, one has

\[
\frac{1}{3t}(\log(1 + t))^2 t - t \leq f_t(x) - x \leq \frac{1}{3t} (\log(1 + t))^2 t.
\]

Then we discuss the scenario when \( x > t \), where

\[
f_t(x) - x = \left[ \frac{\log(1 + t)}{t} \right]^2 - 1 \left[ x + \frac{1}{3x} (\log(1 + t))^2 - \frac{1}{t} (\log(1 + t))^2 \right],
\]

which is a decreasing function of variable \( x \). Therefore,

\[
f_t(B) - B = [f_t(x) - x]_{x=B} \leq f_t(x) - x \leq [f_t(x) - x]_{x=t} = f_t(t) - t,
\]

where we further have \( f_t(t) - t = \frac{1}{3t} (\log(1 + t))^2 t - t \leq 0 \).

By the combination of two scenario \((x \leq t \text{ and } x > t)\), we prove the statement in equation (8). \[\square\]
C.3. Proof of a Theorem

The proof of Theorem 3.1 can be found below.

**Proof.** To begin with, we revisit some notations in linear algebra. For matrix $A$, we use $A_{ij}$ to indicate the $(i, j)$th entry in matrix $A$. Besides, its maximal/minimal eigenvalue is $\lambda_{\text{min}}(A)/\lambda_{\text{min}}(A)$, respectively.

It is known that, the condition number of function $F_{t_k}(\beta) = \frac{1}{2n} \|y - X\beta\|_2^2 + \lambda f_{t_k}(\beta)$ is defined by the ratio between the maximal and minimal eigenvalue of its Hessian. Recall that, the $(i, j)$th entry of the Hessian matrix of the surrogate function $f_{t_k}(\beta)$, noted as $H_{t_k,i,j}$, is

$$H_{t_k,i,j} = \begin{cases} \frac{1}{2} \log(1 + t_k)^2 \max\{|\beta|_i, t_k\}^{-3}, & \text{if } i = j, \\ 0, & \text{otherwise.} \end{cases}$$

Note that the Hessian matrix $H_{t_k}$ is diagonal and positive definite; therefore one can easily find its minimum and maximum eigenvalues. So the condition number of function $F_{t_k}(\beta) = \frac{1}{2n} \|y - X\beta\|_2^2 + \lambda f_{t_k}(\beta)$, noted as $\kappa_k$, is

$$\kappa_k = \frac{\lambda_{\text{max}}(X'X_n + \lambda H_{t_k})}{\lambda_{\text{min}}(X'X_n + \lambda H_{t_k})}$$

$$\leq \frac{\lambda_{\text{max}}(X'X_n) + \lambda \lambda_{\text{max}}(H_{t_k})}{\lambda_{\text{min}}(X'X_n) + \lambda \lambda_{\text{min}}(H_{t_k})}$$

$$\leq \frac{\lambda_{\text{max}}(X'X_n) + \lambda \lambda_{\text{min}}(H_{t_k})}{\lambda_{\text{min}}(H_{t_k})}$$

$$= \frac{\lambda_{\text{max}}(X'X_n) + \frac{1}{2t_k} \log(1 + t_k)^2}{\lambda_{\text{min}}(H_{t_k})}$$

$$= \frac{3x^3 \lambda_{\text{max}}(X'X_n)}{2\lambda \log(1 + t_k)^2 + \frac{x^3}{t_k}}$$

$$\leq \frac{3B^3 \lambda_{\text{max}}(X'X_n)}{2\lambda \log(1 + \tau)^2 + \left(\frac{B}{\tau}\right)^3}.$$  

Equation (26) is due to the definition of the condition number. Inequality (27) is because of the two fact. First, for the maximal eigenvalue of summation of two matrix $A + B$, i.e., $\lambda_{\text{max}}(A + B)$, is no more than summation of maximal eigenvalue separately, $\lambda_{\text{max}}(A) + \lambda_{\text{max}}(B)$. Second, similar to the maximal eigenvalue, the minimal eigenvalue follows the similar rule that $\lambda_{\text{min}}(A + B) \geq \lambda_{\text{min}}(A) + \lambda_{\text{min}}(B)$. The equality in (28) is due to the fact that matrix $H_{t_k}$ is diagonal with positive diagonal entries. The $x$ in (28) refers to $x = \max\{|\beta|_i: \beta_i \text{ is the } i\text{th entry in } \beta\}$.

Inequality (29) is because that $t_k \geq \tau$ and we assume that throughout the algorithm, all elements in $\beta^{(k)}(k = 1, 2, . . .)$ is bounded by $B$.

By calling Theorem 3.7 in Lan (2019) (i.e., the theorem in Appendix B in this paper), we can prove the statement in Theorem 3.1. The details of the proof are listed as
follows. Recall that we want to minimize $F_{t_k}(\beta)$ for a fixed $k$. From the previous analysis, we can find that $F_{t_k}(\beta)$ is $L_k$-smooth and $\mu_k$-strongly convex, where $L_k = \lambda_{\max}\left(\frac{X'X}{n} + \lambda H_{t_k}\right)$ and $\mu_k = \lambda_{\min}\left(\frac{X'X}{n} + \lambda H_{t_k}\right)$. Consequently, the condition number in the $k$th outer-iteration $\kappa_k = \frac{\lambda_{\max}}{\mu_k}$ can be upper bounded by $\frac{3B^2\lambda_{\max}(\frac{X'X}{n})^2}{2\lambda_{\max}(1+\tau)^2} + \left(\frac{\tau}{2}\right)^3$ for any $\{k = 0, 1, 2, \ldots : t_k \geq \tau\}$.

For a fixed $k$, when applying AGD to minimize $F_{t_k}(\beta)$, our steps, which are line 8-10 in Algorithm 2, follow the AGD steps that are presented in (21)-(23), by setting $\alpha_k = \sqrt{\frac{\mu_k}{\mu_k}}, q_k = \frac{\alpha_k - \mu_k/L_k}{1 - \mu_k/L_k}$, $\gamma_k = \frac{\alpha_k}{\mu_k(1 - \alpha_k)}$. According to (21) in Theorem B.1, if

$$
(1 - \alpha_k)^s \left[ F_{t_k}(\beta^{(k)[s]}) - F_{k,\min} + \alpha_k \left( \mu_k + \frac{1}{\gamma_k} \right) V(\beta^{(k-1)[1]}, \hat{\beta}_k) \right]_{C_k}
- \alpha_k \left( \mu_k + \frac{1}{\gamma_k} \right) V(\beta^{(k-1)[s-1]}, \hat{\beta}_k) \leq \bar{\epsilon}_k,
$$

then $F_{t_k}(\beta^{(k)[s]}) - F_{k,\min} \leq \bar{\epsilon}_k$ with a given $\bar{\epsilon}_k$. Here in (30), we have $\hat{\beta}_k = \arg \min_{\beta} F_{t_k}(\beta)$ and function $V(\cdot, \cdot)$ has been defined in Appendix B.

We then solve the inequality in (30) to get an explicit formula for the quantity $\bar{\epsilon}_k$. To achieve this goal, we simplify (30) first. Note that quantities $C_k$ and $D_k$ are defined via underlining in (30). It can be verified that $D_k \geq 0$. This is because $v(x) = ||x||_2^2/2$ (recall the definition of $v(x)$ in Appendix B) is a convex function, i.e., we have

$$
V(\beta^{(k-1)[s-1]}, \hat{\beta}_k) = v(\hat{\beta}_k) - \left[ v(\beta^{(k-1)[s-1]}) + (\hat{\beta}_k - \beta^{(k-1)[s-1]})' \nabla v(\beta^{(k-1)[s-1]}) \right] \geq 0.
$$

Since $D_k > 0$, if we have

$$
(1 - \alpha_k)^s C_k \leq \bar{\epsilon}_k,
$$

then the inequality in (30) will be satisfied. By introducing simple linear algebra, the above inequality can be rewritten as

$$
(1 - \alpha_k)^s \leq \frac{\bar{\epsilon}_k}{C_k}.
$$

By taking logarithm of both sides, we have

$$
s \log (1 - \alpha_k) \leq \log \left( \frac{\bar{\epsilon}_k}{C_k} \right),
$$

which gives

$$
s \geq -\frac{-\log \left( \frac{\bar{\epsilon}_k}{C_k} \right)}{-\log (1 - \alpha_k)} = \frac{\log \left( \frac{\bar{\epsilon}_k}{C_k} \right)}{-\log (1 - \alpha_k)}.
$$

(31)

Furthermore, we know $\log \left( \frac{1}{1-x} \right) \geq x$ for $0 < x < 1$, so if

$$
s \geq \frac{\log \left( \frac{\bar{\epsilon}_k}{C_k} \right)}{\alpha_k},
$$

(32)
then the inequality in (31) holds. In summary, if we have (32), then we have \( F_t(\hat{\beta}_k) < \tilde{\epsilon}_k \).

Now we will show that, both 
\[
\frac{1}{\alpha_k} = \sqrt{\frac{L_k}{\mu_k}} \text{ and } \log(C_k) \text{ in (32) can be bounded by a constant that does not depend on } k \text{ (or equivalently, } t_k) \text{. First, we prove that } \frac{1}{\alpha_k} = \sqrt{\frac{L_k}{\mu_k}} \text{ can be bound. This is essentially the argument that have been used in the step (29). Second, we prove that } C_k \text{ is also bounded. Because we have}
\]
\[
C_k = \frac{F_t(\beta^{(k)[0]}) - F_{k,\min}}{C_{k,1}} + \alpha_k \left( \mu_k + \frac{1}{\gamma_k} \right) V(\beta^{(k-1)[1]}, \hat{\beta}_k). \]

Note that quantities \( C_{k,1}, C_{k,2}, \) and \( C_{k,3} \) are defined via underlining in the above equation. It is evident that \( C_{k,1} \) and \( C_{k,3} \) are bounded. For \( C_{k,2} \), we have
\[
C_{k,2} = \mu_k,
\]
because we set \( \gamma_k = \frac{\alpha_k}{\mu_k (1 - \alpha_k)} \). Since \( \mu_k \) is bounded above by a constant, quantity \( C_{k,2} \) is bounded as well. By combining the above several block, we know \( \log(C_k) \) is bounded.

In conclusion, after \( C_1 \log(1/\tilde{\epsilon}_k) \) inner-iterations, one is guaranteed to achieve the following precision
\[
F_t(\beta^{(k)}) - F_{\min, k} \leq \tilde{\epsilon}_k,
\]
where \( \tilde{\epsilon}_k = \frac{\lambda p}{2 B} [\log(1 + t_k)]^2 \) and \( C_1 \) is a constant that does not depend on the value of \( t_k \) (or \( k \)).

\( \square \)

C.4. Proof of a Theorem

The proof of Theorem [3.4] is as follows.

PROOF. We start by showing that, for any \( t \geq 0 \), one has
\[
F(\beta^{(k)}) - F(\hat{\beta}) \leq \lambda p(2B + 1)t_k.
\]
This is because of the following sequence of inequalities for any $\beta \in \mathbb{R}^p$:

$$F(\hat{\beta}(k)) = \frac{1}{2n} \left\| y - X\hat{\beta}(k) \right\|^2_2 + \lambda \left\| \hat{\beta}(k) \right\|_1$$

$$= \frac{1}{2n} \left\| y - X\hat{\beta}(k) \right\|^2_2 + \lambda \sum_{i=1}^{p} \left| \hat{\beta}_i(k) \right|$$

$$\leq \frac{1}{2n} \left\| y - X\hat{\beta}(k) \right\|^2_2 + \lambda \sum_{i=1}^{p} f_{t_k}(\hat{\beta}_i(k))$$

$$= \frac{1}{2n} \left\| y - X\beta(k) \right\|^2_2 + \lambda \sum_{i=1}^{p} f_{t_k}(\beta_i(k)) + \lambda p B \left[ 1 - \left( \frac{\log(1 + t_k)}{t_k} \right)^2 \right] - \frac{\lambda p}{3B} \log(1 + t_k)^2 + \frac{\lambda p}{t_k} \log(1 + t_k)^2$$

$$\leq \frac{1}{2n} \left\| y - X\beta(k) \right\|^2_2 + \lambda \sum_{i=1}^{p} f_{t_k}(\beta_i(k)) + \lambda p B \left[ 1 - \left( \frac{1}{1 + t_k} \right)^2 \right] - \frac{\lambda p}{3B} \log(1 + t_k)^2 + \lambda pt_k$$

$$\leq \frac{1}{2n} \left\| y - X\beta(k) \right\|^2_2 + \lambda \sum_{i=1}^{p} f_{t_k}(\beta_i(k)) + 2\lambda p B t_k - \frac{\lambda p}{3B} \log(1 + t_k)^2 + \lambda p t_k$$

$$= \frac{1}{2n} \left\| y - X\beta(k) \right\|^2_2 + \lambda \sum_{i=1}^{p} f_{t_k}(\beta_i(k)) + \lambda p(2B + 1)t_k - \frac{\lambda p}{3B} \log(1 + t_k)^2$$

$$\leq \frac{1}{2n} \left\| y - X\beta(k) \right\|^2_2 + \lambda \sum_{i=1}^{p} f_{t_k}(\beta_i(k)) + \lambda p(2B + 1)t_k - \frac{\lambda p}{3B} \log(1 + t_k)^2$$

$$= \frac{1}{2n} \left\| y - X\beta(k) \right\|^2_2 + \lambda \sum_{i=1}^{p} f_{t_k}(\beta_i) + \lambda p(2B + 1)t_k$$

$$\leq \frac{1}{2n} \left\| y - X\beta(k) \right\|^2_2 + \lambda \sum_{i=1}^{p} f_{t_k}(\beta_i) + \lambda p(2B + 1)t_k$$

$$\leq \frac{1}{2n} \left\| y - X\beta(k) \right\|^2_2 + \lambda \left\| \beta \right\|_1 + \lambda p(2B + 1)t_k$$

$$= F(\hat{\beta}) + \lambda p(2B + 1)t_k$$

where inequality (33) is due to the left side hand of inequality (33), i.e., $[f_{t_k}(x) - |x|]_{x=B} \leq f_{t_k}(x) - |x|$. And equation (33) is by plugging in the value of $[f_{t_k}(x) - |x|]_{x=\bar{t}_k}$. Inequality (34) utilizes the inequality that $\frac{\lambda p B}{t_k} \leq \log(1 + t_k)$ and inequality $\log(1 + t_k) \leq t_k$. Inequality (35) uses inequality $1 - \left( \frac{1}{1 + t_k} \right)^2 \leq 2t_k$. Inequality (36) is because that we assume the precision in kth inner-iteration is $F_{t_k}(\beta(k)) - F_{t_k}(\hat{\beta}(k)) \leq \bar{t}_k$. Equation (38) is owing to the fact that we set $\bar{t}_k = \frac{\lambda p}{3B} \log(1 + t_k)^2$. Inequality (40) is due to the right
hand side of inequality (8), i.e., $f_t(x) - |x| \leq 0$. Inequality (39) is because $\hat{\beta}^{(k)}$ is the minimizer of $F_{t_k}(\beta)$, so $F_{t_k}(\hat{\beta}^{(k)}) < F_{t_k}(\hat{\beta})$. Inequality (40) is because $f_{t_k}(x) - |x| \leq 0$ in Lemma 2.5.

Through the above series of equalities and inequalities, we know that

$$F(\beta^{(k)}) - F(\hat{\beta}) \leq \lambda p (2B + 1) t_k.$$ 

(41)

Besides, in the statement of the theorem, we have

$$k \geq \frac{-1}{\log(1 - h)} \log \left( \frac{\lambda p (2B + 1) t_0}{\epsilon} \right),$$

which is equivalent to

$$\lambda p (2B + 1) t_k \leq \epsilon.$$

So the right side of inequality (41) isn’t larger than $\epsilon$. Thus, we prove that, when $k \geq \frac{-1}{\log(1 - h)} \log \left( \frac{\lambda p (2B + 1) t_0}{\epsilon} \right)$, we have $F(\beta^{(k)}) - F(\hat{\beta}) \leq \epsilon$. □

C.5. Proof of a Theorem

The proof of Theorem 3.5 is as follows.

**Proof.** The total number of numeric operations is determined by three factors, namely (1) the number of out-iterations, (2) the number of inner-iterations, and (3) the number of numeric operations in each inner-iteration. We adopt the assumption that different basic operations can be treated equally. We have discussed (1) and (2) in Section 3.1 and 3.2, and we discuss (3) briefly here. The main computational cost of an inner-iteration in our proposed algorithm lies in Line 9 of Algorithm 2, which is the matrix multiplication in $\frac{\partial}{\partial \beta^{(k)[s]}} F_{t_k}(\beta^{(k)[s]}) = \frac{X_n^t X_n}{n} \beta^{(k)[s]} - \frac{X_n^t y_n}{n} + \frac{\partial}{\partial \beta^{(k)[s]}} f_{t_k}(\beta^{(k)[s]})$. With matrix $\frac{X_n^t X_n}{n}$, $\frac{X_n^t y_n}{n}$ being pre-calculated and stored at the beginning of the execution, the calculation of $\frac{\partial}{\partial \beta^{(k)[s]}} F_{t_k}(\beta^{(k)[s]})$ requires $O(p^2)$ operations.

Now we count the total number of numerical operations that are need in our proposed method to achieve the $\epsilon$ precision. We know that to achieve $F(\beta^{(k)}) - F_{\min} < \epsilon$, we need at least (Theorem 3.4)

$$N \Delta = \frac{-1}{\log(1 - h)} \log \left( \frac{\lambda p (2B + 1) t_0}{\epsilon} \right)$$

outer-iterations. Furthermore, we know that the number inner-iteration in an inner-loop $k$ is $O(\log(\frac{1}{\epsilon_t}))$ with a hidden constant which can be universally bounded, and the number of operations in each inner-iteration is $p^2$. Therefore, the total number of numerical operations to get the estimator $\beta^{(k)}$ with precision $F(\beta^{(k)}) - F(\hat{\beta}) \leq \epsilon$ can be
upper bounded by the following quantity:

\[
p^2 \sum_{k=1}^{N} \log \left( \frac{1}{\tilde{\epsilon}_k} \right) = p^2 \sum_{k=1}^{N} \log \left( \frac{3B}{\lambda p} \frac{1}{[\log(1 + t_k)]^2} \right)^{-1} \quad (42)
\]

\[
= p^2 \sum_{k=1}^{N} \log \left( \frac{3B}{\lambda p} \right) - p^2 \sum_{k=1}^{N} \log \left( [\log(1 + t_k)]^2 \right)
\]

\[
= p^2 N \log \left( \frac{3B}{\lambda p} \right) - 2p^2 \sum_{k=1}^{N} \log (1 + t_k)
\]

\[
\leq p^2 N \log \left( \frac{3B}{\lambda p} \right) - 2p^2 \sum_{k=1}^{N} \log \left( \frac{t_k}{1 + t_k} \right) \quad (43)
\]

\[
= p^2 N \log \left( \frac{3B}{\lambda p} \right) - 2p^2 \sum_{k=1}^{N} \log (t_k) + 2p^2 \sum_{k=1}^{N} \log (1 + t_k)
\]

\[
= p^2 N \log \left( \frac{3B}{\lambda p} \right) - 2p^2 \sum_{k=1}^{N} \log \left( t_0(1 - h)^k \right) + 2p^2 \sum_{k=1}^{N} \log (1 + t_k)
\]

\[
\leq p^2 N \log \left( \frac{3B}{\lambda p} \right) - 2p^2 \sum_{k=1}^{N} \log \left( t_0(1 - h)^k \right) + 2p^2 \sum_{k=1}^{N} t_k \quad (44)
\]

\[
= p^2 N \log \left( \frac{3B}{\lambda p} \right) - 2p^2 \sum_{k=1}^{N} \left[ \log (t_0) + k \log (1 - h) \right] + 2p^2 \sum_{k=1}^{N} t_0(1 - h)^k
\]

\[
= p^2 N \log \left( \frac{3B}{\lambda p} \right) - 2p^2 N \log (t_0) - 2p^2 \log (1 - h) \sum_{k=1}^{N} k + 2p^2 \sum_{k=1}^{N} t_0(1 - h)^k
\]

\[
= p^2 N \log \left( \frac{3B}{\lambda p} \right) - 2p^2 N \log (t_0) - 2p^2 \log (1 - h) \frac{(N + 1)N}{2} + 2p^2 t_0 \left[ 1 - (1 - h)^N \right] \frac{1}{h}
\]

\[
= O(N^2)
\]

where equality (42) is derived by plugging in that \( \tilde{\epsilon}_k = \frac{3B}{\lambda p} \). To be more exactly, there is a hidden constant related to the big O notation in \( O \left( \log \left( \frac{1}{\tilde{\epsilon}_k} \right) \right) \) in equality (42), however, as mention in the proof of Theorem 3.1 this hidden constant can be bounded universally. So in equality (42), we omit this hidden constant. Inequality (43) is derived due to the inequality that \( \log(1 + x) \geq \frac{1}{1 + x} \) for \( x \geq 0 \). Inequality (44) is derived due to the inequality that \( \log(1 + x) \leq x \) for \( x \geq 0 \).

\[
\square
\]

C.6. Proof of a Proposition

The proof of Proposition 5.1 is as follows.
The above is equivalent to
\[
\nabla f_t (\tilde{\beta}) = 0
\]
where \(\tilde{\beta}\) is the minimizer of \(\frac{1}{n} \|y - X\beta\|^2_2 + \lambda \|\beta\|_1\), we can get its first-order condition as:
\[
\frac{1}{n} (X'X\tilde{\beta} + X'y) + \lambda \text{sign} (\tilde{\beta}) = 0 \tag{45}
\]
And because \(\tilde{\beta}\) is the minimizer of \(\frac{1}{n} \|y - X\beta\|^2_2 + \lambda f_t (\beta)\), we can get its first-order condition as:
\[
\frac{1}{n} (X'X\tilde{\beta} + X'y) + \lambda \nabla f_t (\tilde{\beta}) = 0, \tag{46}
\]
where \(\nabla f_t (\tilde{\beta})\) is the gradient of \(f_t (\tilde{\beta})\). By subtracting (45) from (46), we have
\[
\frac{1}{n} X'X (\tilde{\beta} - \beta) + \lambda \left[ \nabla f_t (\beta) - \text{sign} (\beta) \right] = 0.
\]
By left multiplying \((\tilde{\beta} - \beta)'\) on both sides of the above equation, we have
\[
\frac{1}{n} (\tilde{\beta} - \beta)' X'X (\tilde{\beta} - \beta) + \lambda (\tilde{\beta} - \beta)' \left[ \nabla f_t (\beta) - \text{sign} (\beta) \right] = 0.
\]
The above is equivalent to
\[
\frac{1}{n} (\tilde{\beta} - \beta)' X'X (\tilde{\beta} - \beta) = -\lambda (\tilde{\beta} - \beta)' \left[ \nabla f_t (\beta) - \text{sign} (\beta) \right] \\
= -\lambda (\tilde{\beta} - \beta)' \nabla f_t (\tilde{\beta}) + \lambda (\tilde{\beta} - \beta)' \text{sign} (\tilde{\beta}) \\
= -\lambda (\tilde{\beta} - \beta)' \nabla f_t (\tilde{\beta}) + \lambda \tilde{\beta}' \text{sign} (\tilde{\beta}) - \lambda \beta' \text{sign} (\beta) \\
= -\lambda (\tilde{\beta} - \beta)' \nabla f_t (\tilde{\beta}) + \lambda \tilde{\beta}' \text{sign} (\tilde{\beta}) - \lambda \|\beta\|_1.
\]
Because \(f_t (\beta)\) is a convex function, we have
\[
\frac{1}{n} (\tilde{\beta} - \beta)' X'X (\tilde{\beta} - \beta) \leq -\lambda \left[ f_t (\tilde{\beta}) - f_t (\beta) \right] + \lambda \tilde{\beta}' \text{sign} (\tilde{\beta}) - \lambda \|\tilde{\beta}\|_1.
\]
So we have
\[
\frac{1}{n} \|X (\tilde{\beta} - \beta)\|^2_2 \leq -\lambda \left[ f_t (\tilde{\beta}) - f_t (\beta) \right] + \lambda \|\beta\|_1 - \lambda \|\tilde{\beta}\|_1.
\]
When \(t \to 0\), we have \(f_t (\beta)\) very close to \(\|\beta\|_1\), so we have \(\frac{1}{n} \|X (\tilde{\beta} - \beta)\|^2_2 \to 0\). □

C.7. Proof of a Proposition

The proof of Proposition 5.2 is as follows.

**Proof.** From Proposition 5.1 we know that
\[
\|X (\tilde{\beta} - \beta)\|^2_2 \to 0
\]
when \( t \to 0 \), where \( \hat{\beta} = \arg \min_{\beta} \frac{1}{2n} \|y - X\beta\|_2^2 + \lambda \|\beta\|_1 \), and \( \widetilde{\beta} = \arg \min_{\beta} \frac{1}{2n} \|y - X\beta\|_2^2 + \lambda f_1(\beta) \). The above can be written as
\[
X_S \left( \tilde{\beta}_S - \hat{\beta}_S \right) + X_{S^c} \tilde{\beta}_{S^c} = \delta, \tag{47}
\]
where \( S \) is the support set of \( \hat{\beta} \) and \( \|\delta\|_2 \approx 0 \). By left multiplying \( (X_S^\dagger X_S)^{-1} X_S' \) on both sides of (47), we have
\[
\left( \tilde{\beta}_S - \hat{\beta}_S \right) + \left( X_S' X_S \right)^{-1} X_S' X_{S^c} \tilde{\beta}_{S^c} = \left( X_S' X_S \right)^{-1} X_S' \delta, \tag{48}
\]
By left multiplying \( X_{S^c}^\dagger \) on both sides of (47), we have
\[
X_{S^c}^\dagger X_S \left( \tilde{\beta}_S - \hat{\beta}_S \right) + X_{S^c}^\dagger X_{S^c} \tilde{\beta}_{S^c} = X_{S^c}^\dagger \delta, \tag{49}
\]
where \( X_{S^c}^\dagger \) is the pseudo-inverse of matrix \( X_{S^c} \). The mathematical meaning of pseudo-inverse is that, suppose \( X_{S^c} = US\Sigma V' \), which is the singular value decomposition (SVD) of \( X_{S^c} \). Then \( X_{S^c}^\dagger = V\Sigma^\dagger U' \). For the rectangular diagonal matrix \( \Sigma \), we get \( \Sigma^\dagger \) by taking the reciprocal of each non-zero elements on the diagonal, leaving the zeros in place, and then transposing the matrix.

By reorganizing (48) and (49) into block matrix, we have
\[
\begin{pmatrix}
I \\
X_{S^c}^\dagger X_S \\
X_{S^c}^\dagger X_{S^c}
\end{pmatrix}
\begin{pmatrix}
\tilde{\beta}_S - \hat{\beta}_S \\
X_{S^c}^\dagger X_S \tilde{\beta}_{S^c}
\end{pmatrix}
= \begin{pmatrix}
\left( X_S^\dagger X_S \right)^{-1} X_S' \delta \\
X_{S^c}^\dagger \delta
\end{pmatrix}.
\]
Through this system of equations, we can solve \( \left( \begin{pmatrix} \tilde{\beta}_S - \hat{\beta}_S \\ X_{S^c}^\dagger X_{S^c} \tilde{\beta}_{S^c} \end{pmatrix} \right) \) as
\[
\left\| \begin{pmatrix} \tilde{\beta}_S - \hat{\beta}_S \\ X_{S^c}^\dagger X_{S^c} \tilde{\beta}_{S^c} \end{pmatrix} \right\|_2 \leq \left\| M^{-1} \left( \begin{pmatrix} \left( X_S^\dagger X_S \right)^{-1} X_S' \delta \\ X_{S^c}^\dagger \delta \end{pmatrix} \right) \right\|_2.
\]
Because for a matrix \( A \) and vector \( x \), we have \( \|Ax\|_2 \leq \|A\|_F \|x\|_2 \), we can bound \( \left\| \tilde{\beta}_S - \hat{\beta}_S \right\|_2 + \left\| \tilde{\beta}_{S^c} \right\|_2 \) as
\[
\left\| \tilde{\beta}_S - \hat{\beta}_S \right\|_2 + \left\| \tilde{\beta}_{S^c} \right\|_2 \leq \left\| M^{-1} \right\|_F \left( \begin{pmatrix} \left( X_S^\dagger X_S \right)^{-1} X_S' \delta \\ X_{S^c}^\dagger \delta \end{pmatrix} \right) \leq \left\| M^{-1} \right\|_F \left( \begin{pmatrix} \left( X_S^\dagger X_S \right)^{-1} X_S' \delta \\ X_{S^c}^\dagger \delta \end{pmatrix} \right) \leq \left\| M^{-1} \right\|_F \left( \begin{pmatrix} \left( X_S^\dagger X_S \right)^{-1} X_S' \delta \\ X_{S^c}^\dagger \delta \end{pmatrix} \right) \|X_S', \delta\|_2^2.
\]
Because \( \|M^{-1}\|_F \leq \sqrt{\text{rank}(M^{-1})} \|M^{-1}\|_2 \), we can further bound \( \left\| \tilde{\beta}_S - \hat{\beta}_S \right\|_2 + \left\| \tilde{\beta}_{S^c} \right\|_2 \)
where as

\[
\| \tilde{\beta} - \hat{\beta} \|_2^2 + \| \tilde{\beta}^* \|_2^2
\]

\[
\leq \text{rank}(M^{-1}) \| M^{-1} \|_2^2 \left( \left\| (X_\delta' X_\delta)^{-1} X_\delta' \|_2^2 + \| X_\delta' \delta \|_2^2 \right) \right)
\]

\[
= \text{rank}(M^{-1}) \left[ \frac{1}{\lambda_{\text{min}}(M)} \right]^2 \left( \left\| (X_\delta' X_\delta)^{-1} X_\delta' \|_2^2 + \| X_\delta' \delta \|_2^2 \right) \right].
\]

For \( \left\| (X_\delta' X_\delta)^{-1} X_\delta' \delta \right\|_2^2 \) in (50), we have

\[
\left\| \frac{(X_\delta' X_\delta)^{-1} X_\delta' \delta}{Q} \right\|_2^2 = \| Q \delta \|_2^2
\]

\[
= \sum_{i=1}^{q_i(\delta)}^2 \leq \sum_{i=1}^{\left| S \right|} \| q_i \|_2^2 \| \delta \|_2^2
\]

\[
= \| Q \|_F^2 \| \delta \|_2^2,
\]

where \( q_i \) denotes the \( i \)th row in matrix \( Q \), and \( Q \) denotes \( (X_\delta' X_\delta)^{-1} X_\delta' \). Because \( \| Q \|_F^2 \)

is bounded and \( \| \delta \|_2^2 \to 0 \), we have \( \left\| (X_\delta' X_\delta)^{-1} X_\delta' \delta \right\|_2^2 \to 0 \).

For \( \left\| X_\delta' \delta \right\|_2^2 \) in (50), following the similar logic, we have

\[
\left\| X_\delta' \delta \right\|_2^2 \leq \left\| X_\delta' \|_F^2 \| \delta \|_2^2,
\]

Because \( \left\| X_\delta' \|_F^2 \) is bounded and \( \| \delta \|_2^2 \to 0 \), we have \( \left\| X_\delta' \delta \right\|_2^2 \).

For \( \lambda_{\text{min}}(M) \) in (50), let’s start with a general eigenvalue of matrix \( M \), and we denote the eigenvalue of \( M \) as \( \lambda(M) \). If we prove that all the eigenvalue of matrix \( M \) is strictly larger than 0, than \( \frac{1}{\lambda_{\text{min}}(M)} \) can be bounded. This is equivalent to prove that \( M - \lambda(M) I \) is positive semidefinite for any eigenvalue \( \lambda(M) \).

If we denote \( M^* = \frac{M + M^*}{2} \), then we notice that \( \lambda(M) = \lambda(M^*) \). We will verify that \( M^* - \lambda(M) I \) is positive semidefinite under the conditions of Proposition 5.2. To verify it, we know that for any \( \alpha, \beta \), we have

\[
\left( \begin{array}{cc} \alpha' & \beta' \end{array} \right) M^* \left( \begin{array}{c} \alpha \\ \beta \end{array} \right)
\]

\[
= \left( \begin{array}{cc} \alpha' & \beta' \end{array} \right) \left( (1 - \lambda)I + \frac{A + B'}{2}X_\delta' X_\delta' + \frac{A + B'}{2} \left( X_\delta' X_\delta' \right)' - \lambda I \right) \left( \begin{array}{c} \alpha \\ \beta \end{array} \right)
\]

\[
= (1 - \lambda) \| \alpha \|_2^2 + \beta' \left[ \frac{1}{2} X_\delta' X_\delta + \frac{1}{2} \left( X_\delta' X_\delta \right)' - \lambda I \right] \beta + \alpha' (A + B') \beta,
\]
where $A = (X_S'X_S)^{-1}X_S'X_{S'}$, $B = X_{S'}'X_S$. For the last term in (51), we can apply SVD to $A + B'$, i.e., $A + B' = U_1\Sigma_1V_1$, then we have

$$|\alpha'(A + B')\beta| = \alpha'U_1\Sigma_1V_1\beta$$

$$\leq \sigma_{\max}(\Sigma_1)|\alpha'U_1|_2\|V_1\beta\|_2$$

$$\leq \sigma_{\max}(\Sigma_1)\|\alpha'\|_2\|\beta\|_2$$

$$\leq \frac{1}{2}\sigma_{\max}(\Sigma_1)\left(\|\alpha'\|^2_2 + \|\beta\|^2_2\right),$$

where $\sigma_{\max}(\Sigma_1)$ is the maximal absolute value in the diagonal entry of $\Sigma_1$.

By plugging the above result into (51), we have

$$(\alpha' \beta')M^*\left(\begin{array}{c}
\alpha \\
\beta
\end{array}\right) \geq (1 - \lambda)\|\alpha\|^2_2 + \beta'\left[\frac{1}{2}(X_{S'}'X_{S'})' - \lambda I\right]\beta$$

$$-|\alpha'(A + B')\beta|$$

$$\geq (1 - \lambda)\|\alpha\|^2_2 + \beta'\left[\frac{1}{2}(X_{S'}'X_{S'})' - \lambda I\right]\beta$$

$$-\frac{1}{2}\sigma_{\max}(\Sigma_1)\left(\|\alpha'\|^2_2 + \|\beta\|^2_2\right)$$

$$= \left(1 - \lambda - \frac{1}{2}\sigma_{\max}(\Sigma_1)\right)\|\alpha\|^2_2 +$$

$$\beta'\left[\frac{1}{2}(X_{S'}'X_{S'})' + \frac{1}{2}\left(X_{S'}'X_{S'}\right) - \left(\lambda + \frac{1}{2}\sigma_{\max}(\Sigma_1)\right)I\right]\beta,$$

where $\sigma_{\max}(\Sigma_1)$ is the maximal absolute diagonal value of matrix $\Sigma_1$. Because we have $\sigma(\Sigma_1) < 2$, so the first term in (52) is greater than 0. Besides, because the minimal singular value of $\frac{1}{2}(X_{S'}'X_{S'})' + \frac{1}{2}\left(X_{S'}'X_{S'}\right)$ is larger than $\frac{1}{2}\sigma_{\max}(\Sigma_1)$, i.e., $\frac{1}{2}(X_{S'}'X_{S'})' = U_2\Sigma_2V_2$ and $2\sigma_{\min}(\Sigma_2) > \sigma_{\max}(\Sigma_1)$, the second term in (52) is also greater than 0. Thus, we prove that $M^*$ is a positive semidefinite matrix, whose eigenvalue would be strictly larger than 0. According, $M$, which shares the same eigenvalue with $M^*$ also has eigenvalues strictly larger than 0. So we have $\frac{1}{\lambda_{\min}(M)}$ bounded.

In conclusion, because $\lambda_{\min}(M)$ is bounded, $\|X_S'X_S^{-1}X_{S'}\delta\|_2^2 \to 0$, and $\|X_{S'}\delta\|_2^2 \to 0$, we have

$$\|\tilde{\beta} - \beta\|_2^2 = \|\tilde{\beta}_S - \beta_S\|_2^2 + \|\tilde{\beta}_{S'}\|_2^2 \to 0.$$