THE APPLICATION OF OPTIMIZATION
TECHNIQUES IN MACHINE LEARNING

HAOTIAN PANG

A DISSERTATION
PRESENTED TO THE FACULTY
OF PRINCETON UNIVERSITY
IN CANDIDACY FOR THE DEGREE
OF DOCTOR OF PHILOSOPHY

RECOMMENDED FOR ACCEPTANCE
BY THE DEPARTMENT OF
ELECTRICAL ENGINEERING
ADVISER: PROFESSOR VANDERBEI

SEPTEMBER 2017
Abstract

The past decades have witnessed significantly progress in machine learning, and solving these problems requires the advancing in optimization techniques. High dimensional sparse learning has imposed a great computational challenge to large scale data analysis. In this dissertation, parametric simplex method is applied to solve a broad class of sparse learning approaches, which can be formulated as linear programs parametrized by a regularization parameter. There are serious drawbacks to the existing methods for solving these types of problems as tuning the parameter for the desired solution is very inefficient. The customized parametric simplex method is introduced and it uses the unknown weighting factor as the parameter and provides a powerful and efficient way to address these shortcomings. Although the simplex method has an exponential complexity in the worse case, it has been shown that the parametric simplex method is an appropriate method for these cases when the expected solution is sparse. An R package named fastclime which efficiently solves a variety machine learning problems by the customized parametric simplex method is developed. A convex optimization method named Inexact Peaceman-Rachford Splitting Method (IPRSM) is studied. Similar to the Alternating Direction Method of Multiplier (ADMM), the strictly contractive Peaceman-Rachford Splitting Method (PRSM) is also used to solve a convex minimization problem with linear constraints and a separable objective function. In many applications, it is quite expensive to obtain exact solutions to these subproblems. The inexact methods intend to solve the iterative subproblems when the exact solutions do not exist or they are hard to obtain. Finally, the new graph Perceptron algorithm, a graph estimation method which performs on online binary classification problems is proposed. The new graph Perceptron algorithm is a new kernel based Perceptron derived from online class-action and extend to online graph estimation with a new kernel trick.
Acknowledgements

Foremost, I would like to express my deepest gratitude to my advisor, Professor Robert Vanderbei, whose immense knowledge, wisdom and encouragement has made my past five years an amazing life-changing experience. I am grateful for his constant guidance, wholehearted support and endless patience at every stage of my Ph.D. study. My special thanks goes to Professor Han Liu, who has led me into the fascinating world of statistical learning research during the past few years. Through our numerous intense but inspiring discussions, I was greatly influenced by his broad knowledge, critical thinking and uncompromising standards towards research. Being extraordinary researchers and great mentors, they are always ready to help and encourage me to overcome difficulties and frustrations in research.

I thank both Professor Emmanuel Abbe and Professor Sun-Yuan Kung for sitting on my committee. Sincere thanks to Professor Paul Cuff and Professor Peter Ramadge, my dissertation reader, for his support and valuable comments on my work. I thank Prof Emmanuel Abbe and Professor Peter Ramadge, for serving as committee of my general exam. I would like to thank the faculty of the Department of Electrical Engineering as well as the Department of Operational Research and Financial Engineering for their teaching and support during the past few years. During the course of my graduate studies, I have benefited from some of the best lectures, taught by Professor Peter Ramadge, Professor Paul Cuff and Professor Sebastien Bubeck. I would like to thank them for sharing their knowledge, intuition and enthusiasm of their fields. Besides, I have been fortunate to work with a few great young scholars: Professor Xiaoming Yuan and Professor Quanquan Gu, who I collaborated with remotely. Their expertise has enriched my research experience, and their devotion has contributed to the fruitful results of our projects. In no particular order, I would like to thank all the lovely ORFE people, including Xingyuan Fang, Yan Li, Junwei Lu, Elahesadat Naghib, Weichen Wang, Zhaoran Wang, Tianqi Zhao and Tuo Zhao, as
well as my dearest friends, including Yaosheng Fu, Xiaoyu Tang, Sen Tao, Yun Wang, Tianwei Zhang and Yanqi Zhou in Princeton and other parts of the world, for being an endless source of inspiration and happiness in my student life.

Finally, I am eternally grateful to my parents, Jie Pang and Xiaohua Wang, who brought me to this wonderful world and offered my guidance and support in each step of my life, and taught me the value of kindness, enterprise, honor and integrity. I am also extremely blessed to have found my special one, Jin Zhang, to accompany me throughout my Ph.D. journey, and the unknown but exciting adventures ahead of me. To them I dedicate this dissertation.
To my parents.
# Contents

Abstract ........................................................................................................ iii

Acknowledgements ....................................................................................... iv

List of Tables .................................................................................................. x

List of Figures ................................................................................................ xi

1 The Parametric Simplex Method ................................................................. 1

1.1 Introduction .............................................................................................. 1

1.2 The Simplex Method ................................................................................ 4

1.3 The Parametric Simplex Method .............................................................. 8

1.4 Applications of Parametric Simplex Method .......................................... 16

1.4.1 Sparse Linear Regression ................................................................... 17

1.4.2 Sparse Linear Discriminant Analysis ................................................ 19

1.4.3 Sparse Precision Matrix Estimation .................................................... 20

1.4.4 \( \ell_1 \)-norm SVM ............................................................................ 22

1.4.5 Differential Network and its Sparse Formulation ............................... 24

1.5 Initialization of the Simplex Method ........................................................ 27

1.6 Theoretical Justification ......................................................................... 28

1.7 Numerical Experiment ............................................................................ 34

1.7.1 Solution path of Dantzig selector ....................................................... 34

1.7.2 Feasibility of Dantzig Selector ......................................................... 35

1.7.3 Performance Benchmark of Dantzig Selector ................................... 36
1.7.4 Performance Benchmark of Differential Network

2 The Fastclime Package

2.1 Design, Implementation and Examples

2.2 Performance Benchmark

2.3 Conclusions

3 The Inexact Peachman-Rachford Splitting Method

3.1 Introduction

3.2 Preliminaries

3.2.1 Variational Reformulation

3.2.2 Matrix Notations

3.3 Convergence Analysis

3.4 Ergodic Worst-case $O(1/t)$ Convergence Rate

3.5 Nonergodic Worst-case $O(1/t)$ Convergence Rate

3.6 Numerical Results

3.6.1 Statistical Models and Implementation Details

3.6.2 Experiment setup and results

3.7 Conclusion

4 The New Graph Perceptron Algorithm

4.1 Introduction to Online Learning Framework

4.1.1 Classical Perceptron Algorithm

4.2 Online Learning on Graph Estimation

4.2.1 Graph Estimation Problem Setup

4.2.2 Graph Perceptron Algorithm

4.2.3 Learning with Local and Global Consistency

4.3 The New Graph Perceptron Algorithm

4.3.1 Preliminaries
4.3.2 The New Graph Perceptron Algorithm . . . . . . . . . . . . . 106
4.3.3 Regret Bound and Mistake Bound . . . . . . . . . . . . . . . 108

5 Conclusions and Future Work .......................................... 112

Bibliography .................................................................. 115
# List of Tables

1.1 Average feasibility violation with standard errors along the solution path  

1.2 Average timing performance (in seconds) with standard errors in the parentheses on Dantzig selector  

1.3 Average timing performance (in seconds) and iteration numbers with standard errors in the parentheses on differential network  

2.1 Average Timing Performance of Three Solvers  

3.1 Quantitative time comparison (in seconds) among the exact ADMM, the exact strictly contractive PRSM, IPRSM1 and IPRSM2  

3.2 Quantitative iteration number comparison among the exact ADMM, the exact strictly contractive PRSM, IPRSM1 and IPRSM2  

3.3 Quantitatively time comparison (in seconds) between the inner and outer iterations stopping criterion for the LASSO model, $v_k = \frac{1}{n^2}$  

3.4 Quantitatively time comparison (in seconds) between the inner and outer iterations stopping criterion for the sparse logistic regression model, $v_k = \frac{1}{n^2}$  

3.5 Sensitivity test on the underdetermined relaxation factor $\alpha$
List of Figures

1.1 Dantzig selector method: (a) The solution path of the parametric simplex method (b) The parameter path of the parametric simplex method. 35

2.1 Graphs estimated by fastclime .................................................. 40

2.2 The solution path of the stock data ............................................. 42

2.3 Graphs estimated by fastclime based on the stock data .................... 42

3.1 LASSO model: evolution of the objective function, primal residual and dual residual .................................................. 88

3.2 Sparse logistic regression model: evolution of the objective function, primal redisplay and dual residual ........................................ 89

3.3 Sensitivity test on the underdetermined relaxation factor $\alpha$ for IPRSM 92
Chapter 1

The Parametric Simplex Method

1.1 Introduction

A broad spectrum of problems in the areas of statistics and machine learning can be formulated as high-dimensional optimization problems involving some parameter whose appropriate value is not known \textit{a priori}. As an example, Lasso-type problems start with a traditional statistical problem in which some error term is minimized and then encourage a sparse solution by adding a weighted sparsity-inducing term to the minimization problem. Unfortunately, an appropriate choice of the weighting factor is not known ahead of time. Hence, one assumes that the algorithm will be tuned to a specific type of problem so that with future instances one can guess a reasonable starting value for the weighting parameter and get to the desired solution in a small number of trials. However, each trial involves solving a problem from scratch and is therefore time-consuming on high-dimensional problems. There is a chance that the result obtained is not actually the desired result. These are serious drawbacks to the existing methods for solving these types of problems. We will show in this chapter that the \textit{parametric simplex method} using the unknown weighting factor as
the “parameter” provides a powerful and efficient way to address these shortcomings in cases where the underlying optimization problem is linear.

Geometrically speaking, simplex methods move from one intersection of hyperplanes to an “adjacent” intersection in search of an optimal solution. If the intersection is feasible for the problem, then it is a vertex of the polytope of feasible solutions. However, many variants of the simplex method, including the parametric simplex method, do not insist on feasibility at every iteration. Algebraically, the algorithm involves moving from one partition of the variables into basic and nonbasic variables to another. Each partition deviates from the previous in that one basic variable gets swapped with one nonbasic variable in a process called *pivoting*. Although in the worst-case, the simplex method can take an exponential number of pivots to find an optimal solution, in practice, the number of pivots is roughly linear in the number of variables in the problem. Even better, for Lasso-type problems, the desired sparse solution can often be found in very few pivots. Although there is no guaranteed that the number of pivot iterations will be small, we show numerically that desired sparse solutions can be found in a very few iterations.

In addition to the Lasso-type sparseness problem, there is a broad spectrum of applications in machine learning problems in which there is a regularization parameter used to tune the desired solution. We will extend the ideas described above to a variety of problems in machine learning that can be written as a linear programming problem with a regularization parameter.

To be specific, we are interested in parametric linear programming problems with equality or inequality constraints. The equality parametric linear programming is defined as the following:

\[
\max_{x} \langle c + \lambda \bar{c} \rangle^T x \quad \text{subject to: } Ax = b + \lambda \bar{b}, \ x \geq 0, \quad (1.1.1)
\]
The inequality parametric linear programming is defined as:

$$\max_{x} (c + \lambda \bar{c})^T x \quad \text{subject to:} \quad Ax \leq b + \lambda \bar{b}, \quad x \geq 0. \quad (1.1.2)$$

It is well-known that, if a problem can be cast as a parametric linear programming problem, then the parametric simplex method can solve the problem for all values of the parameter in the same time as another variant of the simplex method would require to solve for just one value of the parameter [61, 52]. Recently, [66] applied the “parametric cost linear programming” idea with a perturbation on the right-hand-side to some problems in statistics. We extend this method to perturbation on both right-hand-side and the objective as described in [61] so that a variety of problems in machine learning can benefit from it.

In addition to the computational benefits achieved by the parametric simplex method, some statistical problems also benefit from factoring a dense constraint matrix into a product of sparse matrices. Further dramatic computational speedups can be obtained by exploiting this factorization.

For completeness, we start with a review of the primal simplex method. We first present a detailed description of the parametric simplex method that starts from the (infeasible) all zero solution and proceeds to the desired sparse answer. Next, we apply the parametric simplex method to the Dantzig selector, CLIME estimator used for precision matrix estimation as well as the differential network problem. At the end of this chapter, we present our numerical experiments.

We start with some basic notations and definitions. We use boldface for vectors and matrices. In particular, $\mathbf{1}$ is the vector of all ones and $\mathbf{0}$ is the zero vector. For a vector $\mathbf{a} = (a_1, ..., a_d)^T \in \mathbb{R}^d$, we let $\|\mathbf{a}\|_p = \left(\sum_j |a_j| \right)^{1/p}$ denote the $p$-norm. In particular, the 0-“norm”, $\|\mathbf{a}\|_0$, is the number of nonzero elements of $\mathbf{a}$, $\|\mathbf{a}\|_1 = \sum_j |a_j|$, $\|\mathbf{a}\|_2 = \sum_j a_j^2$, and $\|\mathbf{a}\|_\infty = \max_j |a_j|$. When comparing vectors, “$\geq$” and
“≤” are defined component-wise. For a square matrix $A \in \mathbb{R}^{d \times d}$ with entries $a_{jk}$, we use $\|A\|$ to denote element-wise norms and $\|A\|$ to denote matrix norms. So, $\|A\|_0$ is the number of nonzero entries in $A$, $\|A\|_1 = \sum_{j,k} |a_{jk}|$, and $\|A\|_\infty = \max_{j,k} |a_{jk}|$. For the matrix norms, we define $\|A\|_1 = \max_k \sum_j |a_{jk}|$, $\|A\|_\infty = \max_j \sum_k |a_{jk}|$, $\|A\|_2 = \max_{\|a\|_2 \leq 1} \|Aa\|_2$ and $\|A\|_F^2 = \sum_{j,k} a_{jk}^2$. We denote $A_{\setminus i \setminus j}$ as the submatrix of $A$ with $i$-th row and $j$-th column removed. We denote $A_{\setminus i \setminus j}$ as the $i$-th row of $A$ with its $j$-th entry removed and $A_{\setminus i \setminus j}$ as the $j$-th column of $A$ with its $i$-th entry removed.

For any subset $G$ of $\{1, 2, \ldots, d\}$, we let $A_G$ denote the submatrix of $A \in \mathbb{R}^{p \times d}$ consisting of the corresponding columns of $A$. The notation $A \geq 0$ means all of $A$’s entries are nonnegative. Similarly, for a vector $a \in \mathbb{R}^d$, we let $a_G$ denote the subvector of $a$ associated with the indices in $G$. Finally, $I_d$ denotes the $d$-dimensional identity matrix and $e_i$ denotes vector that has a one in as it’s $i$-th element and is zero elsewhere. In a large matrix, we leave a submatrix blank when all of its entries are zeros.

1.2 The Simplex Method

We now review the primal simplex method for solving the standard linear programming problem:

$$\max_x c^T x \quad \text{subject to: } Ax = b, \quad x \geq 0 \quad x \in \mathbb{R}^n,$$  \hspace{1cm} (1.2.1)

where $A \in \mathbb{R}^{m \times n}$, $b \in \mathbb{R}^m$ and $c \in \mathbb{R}^n$ are given. Without loss of generality, we assume that $m \leq n$ and matrix $A$ has full row rank $m$. Throughout our analysis, we assume that an optimal solution exists (it need not be unique). The primal simplex method starts from a basic feasible solution (to be defined shortly—but geometrically can be thought of as any vertex of the feasible polytope) and proceeds step-by-step (vertex-by-vertex) to the optimal solution. Various techniques exist to find the first
feasible solution. In many variants of the simplex method these techniques involve a so-called Phase I method. See [61, 52, 15].

Algebraically, a basic solution corresponds to a partition of the indices \{1, \ldots, n\} into \(m\) basic indices denoted \(B\) and \(n - m\) non-basic indices denoted \(N\). Note that not all partitions are allowed—the submatrix of \(A\) consisting of the columns of \(A\) associated with the basic indices, denoted \(A_B\), must be invertible. The submatrix of \(A\) corresponding to the nonbasic indices is denoted \(A_N\). Suppressing the fact that the columns have been rearranged, we can write

\[
A = \begin{bmatrix} A_N & A_B \end{bmatrix}.
\]

If we rearrange the rows of \(x\) and \(c\) in the same way, we can introduce a corresponding partition of these vectors:

\[
x = \begin{bmatrix} x_N \\ x_B \end{bmatrix}, \quad c = \begin{bmatrix} c_N \\ c_B \end{bmatrix}.
\]

From the commutative property of addition, it is easy to see that \(Ax = b\) can be written as

\[
\begin{bmatrix} A_N & A_B \end{bmatrix} \begin{bmatrix} x_N \\ x_B \end{bmatrix} = A_Nx_N + A_Bx_B = b.
\]

Since the matrix \(A_B\) is assumed to be invertible, we can express \(x_B\) in terms of \(x_N\) as follows:

\[
x_B = A_B^{-1}b - A_B^{-1}A_Nx_N = x^*_B - A_B^{-1}A_Nx_N, \quad (1.2.2)
\]

where we have written \(x^*_B\) as an abbreviation for \(A_B^{-1}b\). This rearrangement of the equality constraints is called a dictionary because the basic variables are defined as functions of the nonbasic variables.
Denoting the objective $c^T x$ by $\zeta$, then we also can write:

\[
\zeta = c^T x = c_B^T x_B + c_N^T x_N = c_B^T (A_B^{-1} b - A_A^{-1} A_N x_N) + c_N^T x_N 
\]

\[
= c_B^T A_B^{-1} b - ((A_B^{-1} A_N)^T c_B - c_N)^T x_N
\]

\[
= \zeta^* - (z_N^*)^T x_N,
\]

where $\zeta^* = c_B^T A_B^{-1} b$, $x_B^* = A_B^{-1} b$ and $z_N^* = (A_B^{-1} A_N)^T c_B - c_N$.

We call equations (1.2.2) and (1.2.3) the *primal dictionary* associated with the current basis $B$. Corresponding to each dictionary, there is a *basic solution* (also called a dictionary solution) obtained by setting the nonbasic variables to zero and reading off values of the basic variables:

\[
x_N = 0, \quad x_B = x_B^*.
\]

This particular “solution” satisfies the equality constraints of the problem by construction. To be a feasible solution one only needs to check that the values of the basic variables are nonnegative. So, we say that a basic solution is a *basic feasible solution* if $x_B^* \geq 0$.

An iteration of the simplex method, also called a *pivot*, is accomplished by swapping one basic variable with one nonbasic variable updating the dictionary appropriately. Geometrically, a pivot corresponds to jumping from one vertex to an adjacent vertex in the polytope of feasible solutions. The pair of indices that are swapped are chosen so that the objective function increases (since we are maximizing) and the dictionary solution remains feasible. These conditions guide but do not uniquely determine the indices to swap. One can employ various selection methods. The
particular method chosen is called a *pivot rule*. Different pivot rules correspond to different variants of the simplex method.

The dual of (1.2.1) is given by

\[
\max_y -b^T y \quad \text{subject to: } A^T y - z = c, \quad z \geq 0 \quad z \in \mathbb{R}^n, y \in \mathbb{R}^m. \quad (1.2.4)
\]

In this case, we separate variable \( z \) into basic and nonbasic parts as before:

\[
\begin{bmatrix}
    z \\
    z_B
\end{bmatrix} =
\begin{bmatrix}
    z_N \\
    z_B
\end{bmatrix}.
\]

Now (1.2.4) becomes

\[
A_N^T y - z_N = c_N \quad (1.2.5)
\]

and

\[
A_B^T y - z_B = c_B \quad (1.2.6)
\]

We can use (1.2.6) to solve for the “free” (i.e., not constrained to be nonnegative) variable \( y \):

\[
y = (A_B^T)^{-1}(c_B + z_B).
\]

We plug this back to (1.2.5) and the dual objective. The corresponding dual dictionary is given by:

\[
z_N = (A_B^{-1}A_N)^T c_B - c_N + (A_B^{-1}A_N)^T z_B = z_N^* + (A_B^{-1}A_N)^T z_B, \quad (1.2.7)
\]

\[
-\xi = -c_B^T A_B^{-1} b + (A_B^{-1}b)^T z_B = -\zeta^* - (x_B^*)^T z_B, \quad (1.2.8)
\]

where \( \xi \) denotes the objective function in the (1.2.4), \( \zeta^* = c_B^T A_B^{-1} b, x_B^* = A_B^{-1} b \) and \( z_N^* = (A_B^{-1}A_N)^T c_B - c_N. \)
For each dictionary, we set $x_N$ and $z_B$ to 0 (complementarity) and read off the solutions to $x_B$ and $z_N$ according to (1.2.2) and (1.2.7). Next, we update the dictionary by removing one basic index and replacing it with a nonbasic index, then we get an updated dictionary. The simplex method produces a sequence of steps to adjacent bases such that the value of the objective function is always increasing at each step. Primal feasibility requires that $x_B \geq 0$, so while we update the dictionary, primal feasibility must always be satisfied. This process will stop when $z_N \geq 0$ (dual feasibility), and this is the optimality condition since it satisfies primal feasibility, dual feasibility and complementarity.

1.3 The Parametric Simplex Method

We introduce the parametric simplex method used to find the full solution path while solving the parametric linear programming problem only once. Different variant of the simplex method is defined by the rule for choosing the pair of variables to swap at each iteration. Here we describe the rule used by the parametric simplex method: we add some positive perturbations ($\bar{b}$ and $\bar{c}$) times a positive parameter $\lambda$ to both objective function and the right hind side of the primal problem. The purpose of doing this is to guarantee the primal and dual feasibility when $\lambda$ is large. Since the problem is already primal feasible and dual feasible, there is no phase I stage required for the parametric simplex method. Furthermore, if the $i$-th entry of $b$ or the $j$-th entry of $c$ has already satisfied the feasibility condition ($b_i \geq 0$ or $c_j \leq 0$), then the corresponding perturbation $\bar{b}_i$ or $\bar{c}_j$ to that entry is allowed to be 0. With these perturbations, (1.2.1) becomes:

$$\max (c + \lambda \bar{c})^T x \quad \text{subject to: } Ax = b + \lambda \bar{b}, \quad x \geq 0 \quad x \in \mathbb{R}^n. \quad (1.3.1)$$
We separate the perturbation vectors into basic and nonbasic parts as well and write down the dictionary with perturbations corresponding to (1.2.2), (1.2.3), (1.2.7) and (1.2.8) as:

\[ x_B = (x_B^* + \lambda x_B) - A_B^{-1} A_N x_N, \]  
\[ \zeta = \zeta^* - (z_N^* + \lambda z_N)^T x_N, \]  
\[ z_N = (z_N^* + \lambda z_N) + (A_B^{-1} A_N)^T z_B, \]  
\[ - \xi = - \zeta^* - (x_B^* + \lambda x_B)^T z_B, \]

where \( x_B^* = A_B^{-1} b, \) \( z_N^* = (A_B^{-1} A_N)^T c_B - c_N, \) \( x_B = A_B^{-1} \bar{b} \) and \( z_N = (A_B^{-1} A_N)^T \bar{c}_B - \bar{c}_N. \)

Assume that initially \( x_B \) and \( z_N \) are nonnegative, then when \( \lambda \) is large, the dictionary will be both primal and dual feasible (\( x_B^* + \lambda x_B \geq 0 \) and \( z_N^* + \lambda z_N \geq 0 \)). The corresponding primal solution is simple: \( x_B = x_B^* + \lambda x_B \) and \( x_N = 0 \). This solution is valid until \( \lambda \) hits a lower bound which breaks the feasibility. The smallest value of \( \lambda \) without breaking any feasibility is given by

\[ \lambda^* = \min \{ \lambda : z_N^* + \lambda z_N \geq 0 \text{ and } x_B^* + \lambda x_B \geq 0 \}. \]  

The minimum \( \lambda \) for the initial dictionary is given by (1.3.6).

It is not hard to derive the optimal range of \( \lambda \) for a general dictionary in the form of (1.3.2) to (1.3.5). The dictionary and its corresponding solution \( x_B = x_B^* + \lambda x_B \) and \( x_N = 0 \) is optimal for the value of \( \lambda \in [\lambda^*, \lambda_{\text{max}}] \), where

\[ \lambda^* = \max \left( \max_{j \in N, z_{N_j} > 0} \frac{z_{N_j}^*}{z_{N_j}}, \max_{i \in B, x_{B_i} > 0} \frac{x_{B_i}^*}{x_{B_i}} \right), \]  
\[ \lambda_{\text{max}} = \min \left( \min_{j \in N, z_{N_j} < 0} \frac{z_{N_j}^*}{z_{N_j}}, \min_{i \in B, x_{B_i} < 0} \frac{x_{B_i}^*}{x_{B_i}} \right). \]
Note that although initially the perturbations are nonnegative, as the dictionary gets updated, the perturbation does not necessarily maintain nonnegativity. For each dictionary, there is a corresponding interval of $\lambda$ given by (1.3.7) and (1.3.8). We have characterized the optimal solution for this interval, and these together give us the solution path of the original parametric linear programming problem. Next, we show how the dictionary gets updated as the leaving variable and entering variable swap.

We expect that after swapping the entering variable $j$ and leaving variable $i$, the new solution in the dictionary (1.3.2) and (1.3.4) would slightly change to:

\[ x_j^* = t, \quad \bar{x}_j = \bar{t}, \]
\[ z_i^* = s, \quad \bar{z}_i = \bar{s}, \]
\[ x^*_B \leftarrow x^*_B - t\Delta x_B, \quad \bar{x}_B \leftarrow \bar{x}_B - \bar{t}\Delta x_B, \]
\[ z^*_N \leftarrow z^*_N - s\Delta z_N, \quad \bar{z}_N \leftarrow \bar{z}_N - \bar{s}\Delta z_N, \]

where $t$ and $\bar{t}$ are the primal step length for the primal basic variables and perturbations, $s$ and $\bar{s}$ are the dual step length for the dual nonbasic variables and perturbations, $\Delta x_B$ and $\Delta z_N$ are the primal and dual step directions, respectively. We explain how to find these values in details now.

There is either a $j \in \mathcal{N}$ for which $z^*_N + \lambda \bar{z}_N = 0$ or an $i \in \mathcal{B}$ for which $x^*_B + \lambda \bar{x}_B = 0$ in (1.3.6). If it corresponds to a nonbasic index $j$, then we do one step of the primal simplex. In this case, we declare $j$ as the entering variable, then we need to find the primal step direction $\Delta x_B$. After the entering variable $j$ has been selected, $x_N$ changes from $0$ to $te_j$, where $t$ is the primal step length, then according to (1.3.2), we have that

\[ x_B = (x^*_B + \lambda \bar{x}_B) - A_B^{-1}A_N t e_j. \]

The step direction $\Delta x_B$ is given by $\Delta x_B = A_B^{-1}A_N e_j$. We select the leaving variable next. In order to maintain primal feasibility, we need
to keep $x_B \geq 0$, therefore, the leaving variable $i$ is selected such that $i \in B$ achieves
the maximal value of $\frac{x_i}{x_i^* + \lambda^* x_i}$. It only remains to show how $z_N$ changes. Since $i$
the leaving variable, according to (1.3.4), we have $\Delta z_N = -(A_B^{-1} A_N)^T e_i$. After we
know the entering variables, the primal and dual step directions, the primal and dual
step lengths can be found as

$$t = \frac{x_i^*}{\Delta x_i}, \quad \bar{t} = \frac{x_i^*}{\Delta x_i},$$

$$s = \frac{z_j^*}{\Delta z_j}, \quad \bar{s} = \frac{z_j^*}{\Delta z_j}.$$

If, on the other hand, the constraint in (1.3.6) corresponds to a basic index $i$, we
declare $i$ as the leaving variable, then similar calculation can be made based on the
dual simplex method (apply the primal simplex method to the dual problem). Since
it is very similar to the primal simplex method, we omit the detailed description.

The algorithm will terminate whenever $\lambda^* \leq 0$. The corresponding solution is
optimal since our dictionary always satisfies primal feasibility, dual feasibility and
complementary slackness condition. The only concern during the entire process of
the parametric simplex method is that $\lambda$ does not equal zero, so as long as $\lambda$ can be
set to be zero, we have the optimal solution to the original problem. We summarize
the parametric simplex method for a given parametric linear programming problem
in Algorithm 1.

The following theorem shows that the updated basic and nonbasic partition still
gives the optimal solution.

**Theorem 1.3.1.** For a given dictionary with parameter $\lambda$ in the form of (1.3.2),
(1.3.3), (1.3.4) and (1.3.5), let $B$ be a basic index set and $N$ be an nonbasic index
set. Assume this dictionary is optimal for $\lambda \in [\lambda^*, \lambda_{max}]$, where $\lambda^*$ and $\lambda_{max}$ are given
by (1.3.7) and (1.3.8), respectively. The updated dictionary with basic index set $B^*$
Write down the dictionary as in (1.3.2), (1.3.3), (1.3.4) and (1.3.5);
Find $\lambda^*$ given by (1.3.7);
while $\lambda^* > 0$ do
  if the constraint in (1.3.7) corresponds to an index $j \in \mathcal{N}$ then
    Declare $x_j$ as the entering variable;
    Compute primal step direction. $\Delta x_B = A_{\mathcal{N}}^{-1}A_{\mathcal{B}}e_j$;
    Select leaving variable. Need to find $i \in \mathcal{B}$ that achieves the maximal value of $\Delta x_i^* = x_i^* + \lambda^* \bar{x}_i$;
    Compute dual step direction. It is given by $\Delta z_{\mathcal{N}} = -(A_{\mathcal{B}}^{-1}A_{\mathcal{N}})^T e_i$;
  else if the constraint in (1.3.7) corresponds to an index $i \in \mathcal{B}$ then
    Declare $z_i$ as the leaving variable;
    Compute dual step direction. $\Delta z_{\mathcal{N}} = -(A_{\mathcal{B}}^{-1}A_{\mathcal{N}})^T e_i$;
    Select entering variable. Need to find $j \in \mathcal{N}$ that achieves the maximal value of $\Delta z_j^* = z_j^* + \lambda^* \bar{z}_j$;
    Compute primal step direction. It is given by $\Delta x_B = A_{\mathcal{B}}^{-1}A_{\mathcal{N}}e_j$;
  Compute the dual and primal step lengths for both variables and perturbations:
    $t = \frac{x_i^*}{\Delta x_i}$, $\bar{t} = \frac{\bar{x}_i}{\Delta x_i}$, $s = \frac{z_j^*}{\Delta z_j}$, $\bar{s} = \frac{\bar{z}_j}{\Delta z_j}$.
  Update the primal and dual solutions:
    $x_j^* = t$, $\bar{x}_j = \bar{t}$, $z_i^* = s$, $\bar{z}_i = \bar{s}$,
    $x_B^* \leftarrow x_B^* - t \Delta x_B$, $\bar{x}_B \leftarrow \bar{x}_B - \bar{t} \Delta x_B$, $z_{\mathcal{N}}^* \leftarrow z_{\mathcal{N}}^* - s \Delta z_{\mathcal{N}}$, $\bar{z}_{\mathcal{N}} \leftarrow \bar{z}_{\mathcal{N}} - \bar{s} \Delta z_{\mathcal{N}}$.
  Update the basic and nonbasic index sets $\mathcal{B} := \mathcal{B} \setminus \{i\} \cap \{j\}$ and $\mathcal{N} := \mathcal{N} \setminus \{j\} \cap \{i\}$. Write down the new dictionary and compute $\lambda^*$ given by (1.3.7);
end
Set the nonbasic variables as 0s and read the values of the basic variables.

**Algorithm 1:** The parametric simplex method
and nonbasic index set $N^*$ given by the parametric simplex method is still optimal at $\lambda = \lambda^*$.

**Proof.** In order to prove that the new dictionary is still optimal, we only need to show that new dictionary is still primal and dual feasible: $x^*_{B^*} + \lambda^* \bar{x}_B \geq 0$ and $z^*_{N^*} + \lambda^* \bar{z}_{N^*} \geq 0$.

Case I. When calculating $\lambda^*$ given by (1.3.7), if the constraint corresponds to an index $i \in B$, then $z^*_{N^*} + \lambda^* \bar{z}_{N^*} \geq 0$ is guaranteed by the way of choosing entering variable. It remains to show the primal solution is not changed: $x^*_{B^*} + \lambda^* \bar{x}_B = x^*_{B^*} + \lambda^* \bar{x}_{B^*}$.

We observe that $A_{B^*}$ is obtained by changing one column of $A_B$ to another column vector from $A_N$, and we assume the difference of these two vectors are $u$. Without loss of generality, we assume that the $k$-th column of $A_B$ is replaced, now we have $A_{B^*} = A_B + u e_k^T$. Sherman-Morrison formula says that

$$A_B A_B^{-1} = I - \frac{u e_k^T A_B^{-1}}{1 + e_k^T A_B^{-1} u} = I - \beta u e_k^T A_B^{-1},$$

where $\beta = \frac{1}{1 + e_k^T A_B^{-1} u}$. Now consider the following term:

$$A_B [x^*_B - x^*_{B^*} + \lambda^* (\bar{x}_B - \bar{x}_{B^*})] = A_B (A_B^{-1} - A_{B^*}^{-1}) b + \lambda^* (\bar{b} - A_B A_{B^*}^{-1} \bar{b})$$

$$= b - A_B A_{B^*}^{-1} b + \lambda^* (\bar{b} - A_B A_{B^*}^{-1} \bar{b})$$

$$= (\beta u e_k^T A_B^{-1}) b + \lambda^* (\beta u e_k^T A_B^{-1}) \bar{b}$$

$$= \beta (u e_k^T A_B^{-1} b + \lambda^* u e_k^T A_B^{-1} \bar{b}).$$

Recall in this case, we have

$$\lambda^* = \max_{i \in B, x_{B_i} > 0} \frac{-x^*_B}{x_{B_i}} = -\frac{e_k^T A_B^{-1} b}{e_k^T A_B^{-1} \bar{b}}. \quad (1.3.11)$$
Substitute the definition of $\lambda^*$ from (1.3.11) into (1.3.10), we notice that the expression in (1.3.10) is 0. Since $A_B$ is invertible, we have $x_B^* + \lambda^*x_B = x_B^* + \lambda^*x_B^*$, and thus the new dictionary is still optimal at $\lambda^*$.

Case II. When calculating $\lambda^*$ given by (1.3.7), if, on the other hand, the constraint corresponds to an index $j \in \mathcal{N}$, then $x_B^* + \lambda^*x_B \geq 0$ is guaranteed by the way we choose leaving variable. It remains to show that it is still dual feasible.

Again, we observe that $A_B$ is obtained by changing one column of $A_B$ (say, $a_i$) to another column vector from $A_N$ (say, $a_j$), and we denote $u = a_j - a_i$ as the difference of these two vectors. Without loss of generality, we assume the replacement occurs at the $k$-th column of $A_B$. Sherman-Morrison formula gives

$$
A_B^{-1}A_B = I - \frac{A_B^{-1}ue_k^T}{1 + e_k^TA_B^{-1}u} = I - \frac{pe_k^T}{1 + e_k^Tp} = \begin{pmatrix}
1 & -\frac{p_1}{1+p_k} \\
\vdots & \ddots \\
-\frac{p_m}{1+p_k} & \ddots & \ddots \\
\end{pmatrix},
$$

(1.3.12)

where $p = A_B^{-1}u$, and $p_l$ denotes the $l$-th entry of $p$. Observe that in (1.3.12), only the $k$-th column is different from the identity matrix.

Dual feasible requires that $z_N^* = (A_B^{-1}A_N)^Tc_B - c_N \geq 0$. Since $(A_B^{-1}A_B)^Tc_B - c_B = 0$, we slightly change the dual feasible condition to: $(A_B^{-1}A)^Tc_B - c \geq 0$. In the parametric linear programming sense, $c \leftarrow c + \lambda\bar{c}$ and $c_B \leftarrow c_B + \lambda\bar{c}_B$. We only need to show that $(A_B^{-1}A)^T(c_B + \lambda^*\bar{c}_B) - (c + \lambda^*\bar{c}) = (A_B^{-1}A)^T(c_B^* + \lambda^*\bar{c}_B^*) - (c + \lambda^*\bar{c})$. 

14
Consider the following term:

\[
(A_B^{-1} A)^T c_{B^*} - c + \lambda^* [(A_B^{-1} A)^T \bar{c}_{B^*} - \bar{c}] - \{(A_B^{-1} A)^T c_{B} - c + \lambda^* [(A_B^{-1} A)^T \bar{c}_{B} - \bar{c}] \}
\]

\[
= A^T (A_B^{-1})^T (c_{B^*} + \lambda^* \bar{c}_{B^*}) - A^T (A_B^{-1})^T (c_{B} + \lambda^* \bar{c}_{B})
\]

\[
= A^T (A_B^{-1})^T (A_B^{-1} A_B) (c_{B^*} + \lambda^* \bar{c}_{B^*}) - A^T (A_B^{-1})^T (c_{B} + \lambda^* \bar{c}_{B})
\]

\[
= -\alpha A^T (A_B^{-1})^T e_k,
\]

where \(\alpha\) is a constant. According to (1.3.12), we have

\[
\alpha = \sum_{i \in B^* \setminus j} \frac{(c_i + \lambda^* \bar{c}_i)p_i}{1 + p_k} - \frac{c_j + \lambda^* \bar{c}_j}{1 + p_k} + c_i + \lambda^* \bar{c}_i
\]

\[
= \sum_{i \in B} \frac{(c_i + \lambda^* \bar{c}_i)p_i}{1 + p_k} + \frac{c_i + \lambda^* \bar{c}_i - c_j + \lambda^* \bar{c}_j}{1 + p_k}
\]

\[
= (c_B + \lambda^* \bar{c}_B)^T A_B^{-1} u + c_i + \lambda^* \bar{c}_i - c_j - \lambda^* \bar{c}_j
\]

\[
= (c_B + \lambda^* \bar{c}_B)^T A_B^{-1} (a_j - a_i) + c_i + \lambda^* \bar{c}_i - c_j - \lambda^* \bar{c}_j
\]

\[
= (c_B + \lambda^* \bar{c}_B)^T A_B^{-1} (a_j - e_k) + c_i + \lambda^* \bar{c}_i - c_j - \lambda^* \bar{c}_j
\]

\[
= (c_B + \lambda^* \bar{c}_B)^T (A_B^{-1} a_j - c_j - \lambda^* \bar{c}_j)
\]

\[
= (c_B^T A_B^{-1} a_j - c_j) + \lambda^* (\bar{c}_B^T A_B^{-1} a_j - \bar{c}_j)
\]

\[
= (c_B^T A_B^{-1} a_j - c_j) + \lambda^* (\bar{c}_B^T A_B^{-1} a_j - \bar{c}_j)
\]

(1.3.14)

where \(c_i\) and \(c_j\) are the entries in \(c\), with indices corresponding to \(a_i\) and \(a_j\), and \(\bar{c}_i\) and \(\bar{c}_j\) are the entries in \(\bar{c}\) and defined similarly.

Recall in this case

\[
\lambda^* = \max_{j \in N, z_{N_j} > 0} \frac{z_{N_j}^*}{z_{N_j}^*} = -\frac{(A_B^{-1} a_j)^T c_B - c_j}{(A_B^{-1} a_j)^T c_B - \bar{c}_j}.
\]

(1.3.15)
Substitute the definition of $\lambda^*$ from (1.3.15) into (1.3.14), we observe that $\alpha = 0$ and thus the dual feasible is guaranteed in the new dictionary. This proves Theorem 1.3.1.

During each iteration, there is an optimal solution corresponding to $\lambda \in [\lambda^*, \lambda_{\text{max}}]$. Notice each of these $\lambda$ ranges is determined by a partition between basic and nonbasic variables, and the number of the partition into basic and nonbasic variables is finite, so after finite steps, we must find the optimal solution corresponding to all $\lambda$ values.

1.4 Applications of Parametric Simplex Method

Many sparse learning problems are formulated as convex programs in a generic form:

$$\min_{\beta} \|\beta\|_1 \quad \text{subject to} \quad \|\nabla L(\beta)\|_{\infty} \leq \lambda,$$  \hspace{1cm} (1.4.1)

where $L(\beta)$ is a smooth convex loss function, $\nabla L(\beta)$ is the gradient of $L(\beta)$, and $\lambda > 0$ is a regularization parameter controlling the trade-off between $\|\beta\|_1$ and $\|\nabla L(\beta)\|_{\infty}$. Alternatively, if we set the convex loss function in infinity norm and the regularization term in $\ell_1$ norm, we can write the same problem in a regularization form:

$$\min_{\beta} \|\nabla L(\beta)\|_{\infty} + \lambda \|\beta\|_1.$$  \hspace{1cm} (1.4.2)

In this case, we replace the normal $l_2$-loss function by the non-smooth $\ell_1$-loss function and $\|\beta\|_1$ serves as a convex penalty function which measures the model complexity. Both (1.4.1) and (1.4.2) are naturally suited for our method. The complete piecewise-linear solution path as a function of the regularization parameter can be computed by a variant of the parametric simplex method.

We will introduce several loss functions associated with various different sparse learning problems.
1.4.1 Sparse Linear Regression

We start with the sparse linear regression problem. Let \( y \in \mathbb{R}^n \) be a response vector and let \( X \in \mathbb{R}^{n \times d} \) be the design matrix. We consider a linear model \( y = X\beta^0 + \epsilon \), where \( \beta^0 \in \mathbb{R}^d \) is the unknown regression coefficient vector, and \( \epsilon \) is the observational noise vector. In high dimensional statistical inference, \( d \) is much larger than \( n \): \( d \gg n \). Even though \( d \) is large, it is assumed that in reality most entries in \( \beta^0 \) are zero: \( \beta^0 \) is a sparse vector: \( \| \beta^0 \|_0 = s \ll n \). Statisticians have proposed many different regression methods to obtain sparse estimators of \( \beta^0 \) including Lasso [60], Elastic Net [72], Dantzig Selector [9] and LAD-Lasso [63]. Of these methods, we will focus on the Dantzig selector and LAD-Lasso as these can be computed by solving a simple linear program with a tuning parameter.

The Dantzig selector is defined as the solution to the following convex program:

\[
\min_{\beta} \| \beta \|_1 \quad \text{subject to} \quad \| X^T (y - X\beta) \|_\infty \leq \lambda. \tag{1.4.3}
\]

Using a common modeling trick, (1.4.3) can be rewritten as a parametric linear program: let \( \beta = \beta^+ - \beta^- \) where \( \beta^+ \) is the vector of positive parts of the elements of \( \beta \) and \( \beta^- \) is the corresponding vector of negative parts. By definition, the positive and negative parts are nonnegative and complementary to each other. In addition, it is easy to see that \( \| \beta \|_1 = 1^T \beta^+ + 1^T \beta^- \). It turns out that we can drop the complementarity condition because, at optimality, at least one of the two parts will be zero. Hence, (1.4.3) can be rewritten as

\[
\min_{\beta^+, \beta^-} 1^T (\beta^+ + \beta^-) \tag{1.4.4}
\]

subject to

\[
\begin{pmatrix}
X^T X & -X^T X \\
-X^T X & X^T X
\end{pmatrix}
\begin{pmatrix}
\beta^+ \\
\beta^-
\end{pmatrix}
\leq
\begin{pmatrix}
\lambda 1 + X^T y \\
\lambda 1 - X^T y
\end{pmatrix}, \quad \beta^+, \beta^- \geq 0.
\]
Now we see clearly that (1.4.4) fits into the parametric linear program form as (1.1.2) with

\[
A = \begin{pmatrix}
X^T X & -X^T X \\
-X^T X & X^T X
\end{pmatrix}, \quad b = \begin{pmatrix}
X^T y \\
-X^T y
\end{pmatrix}, \quad c = -1,
\]

\[
\bar{b} = 1, \quad \bar{c} = 0, \quad x = \begin{pmatrix}
\beta^+ \\
\beta^-
\end{pmatrix}.
\]

Alternatively, the regression problem can be solved in a LAD-Lasso form:

\[
\min_\beta \|X\beta - y\|_1 + \lambda \|eta\|_1. \tag{1.4.5}
\]

Letting \( t = X\beta - y \) and separating both \( t \) and \( \beta \) into positive and negative parts, the problem becomes

\[
\min_{\beta^+, \beta^-, t^+, t^-} 1^T(t^+ + t^-) + \lambda 1^T(\beta^+ + \beta^-) \tag{1.4.6}
\]

subject to

\[
\begin{pmatrix}
X & -X & -I & I
\end{pmatrix}
\begin{pmatrix}
\beta^+ \\
\beta^- \\
t^+ \\
t^-
\end{pmatrix} = y \quad \beta^+, \beta^-, t^+, t^- \geq 0.
\]
Clearly, (1.4.6) fits into form (1.1.1) with

$$
A = \begin{pmatrix} X & -X & -I & I \end{pmatrix}, \quad b = y, \quad c = \begin{pmatrix} 0 \\ -1 \end{pmatrix}.
$$

\[ b = 0, \quad c = \begin{pmatrix} -1 \\ 0 \end{pmatrix}, \quad x = \begin{pmatrix} \beta^+ \\ \beta^- \\ t^+ \\ t^- \end{pmatrix}. \]

### 1.4.2 Sparse Linear Discriminant Analysis

The second problem we consider is the sparse linear discriminant analysis (LDA) for classification problem. Here, we assume that there are two classes of observations generated from a \(d\)-dimensional normal distribution with different means \(\mu_1^0\) and \(\mu_2^0\), but the same covariance matrix \(\Sigma^0\) and precision matrix \(\Omega^0 = (\Sigma^0)^{-1}\). Let \(x_1, \ldots, x_{n_1} \in \mathbb{R}^d\) be \(n_1\) independent and identically distributed samples from \(N(\mu_1^0, \Sigma^0)\) (class 1) and \(y_1, \ldots, y_{n_2} \in \mathbb{R}^d\) be \(n_2\) independent and identically distributed samples from \(N(\mu_2^0, \Sigma^0)\) (class 2). We then calculate sample means by

\[ \bar{x} = \frac{1}{n_1} \sum_{i=1}^{n_1} x_i, \quad \bar{y} = \frac{1}{n_2} \sum_{i=1}^{n_2} y_i, \] (1.4.7)

and we define

\[ \bar{\delta} = (\bar{x} - \bar{y}). \] (1.4.8)

The sample covariance matrix is given by

\[ S = \frac{1}{n_1 + n_2} \left( \sum_{j=1}^{n_1} (x_j - \bar{x})(x_j - \bar{x})^T + \sum_{j=1}^{n_2} (y_j - \bar{y})(y_j - \bar{y})^T \right). \] (1.4.9)
Given a new sample \( z \in \mathbb{R}^d \), Fisher’s linear discriminant rule classifies it to class 1 if
\[
(z - \mu^0)^T \Omega^0 \delta^0 \geq 0,
\]
where \( \mu^0 = (\mu_1 + \mu_2)/2 \) and \( \delta^0 = \mu_1 - \mu_2 \), and otherwise to class 2. This rule is proposed based on equal prior probabilities for both classes. In high dimensions, we have \( d \gg (n_1 + n_2) \). To make the estimation manageable, we assume that \( \| \beta^0 \|_0 = s \ll (n_1 + n_2) \), where \( \beta^0 = \Omega^0 \delta^0 \). This leads to the following convex optimization problem:
\[
\min_{\beta} \| \beta \|_1 \quad \text{subject to} \quad \| S\beta - \delta \|_\infty \leq \lambda.
\]
(1.4.11)

Similar to sparse linear regression, (1.4.11) yields a sparse estimator. We then adopt the same reparametrization trick, and rewrite (1.4.11) as a parametric linear programming problem:
\[
\min_{\beta^+,\beta^-} 1^T (\beta^+ + \beta^-)
\]
subject to
\[
\begin{pmatrix}
S & -S \\
-S & S
\end{pmatrix}
\begin{pmatrix}
\beta^+ \\
\beta^-
\end{pmatrix}
\leq
\begin{pmatrix}
\lambda 1 + \delta \\
\lambda 1 - \delta
\end{pmatrix},
\quad \beta^+,\beta^- \geq 0,
\]
with
\[
A = \begin{pmatrix} S & -S \\ -S & S \end{pmatrix}, \quad b = \begin{pmatrix} \delta \\ -\delta \end{pmatrix}, \quad c = -1, \quad \bar{b} = 1, \quad \bar{c} = 0, \quad x = \begin{pmatrix} \beta^+ \\ \beta^- \end{pmatrix}.
\]

1.4.3 Sparse Precision Matrix Estimation

The third problem of interest is the sparse precision matrix estimation problem. In a typical graphical model setting, a \( d \)-dimensional random vector \( x = (x^1, ..., x^d)^T \) can
be represented by an undirected graph $G = (V, E)$, where graph node set $V$ represents the $d$ variables in $\mathbf{x}$ and the edge set $E$ represents the conditional independence relationship among $x^1, \ldots, x^d$. We say that the graphical model is a Gaussian graphical model when the random vector $\mathbf{x}$ is assumed to be Gaussian: $\mathbf{x} \sim N_d(\mu^0, \Sigma^0)$. We call the inverse of the covariance matrix the precision matrix: $\Theta^0 = (\Sigma^0)^{-1}$. Under the Gaussian assumption, the edge between $x^i$ and $x^j$ is excluded from $E$ if and only if $\Theta^0_{ij} = 0$ [17]. Let $\{x_1, \ldots, x_n\}$ be $n$ independent and identically distributed samples from the distribution of $\mathbf{x}$. We are interested in estimating the support of the true precision matrix. We denote the sample covariance matrix $S = \frac{1}{n} \sum_{j=1}^{n} (x_j - \bar{x})(x_j - \bar{x})^T$, where $\bar{x} = \frac{1}{n} \sum_{j=1}^{n} x_j$. Most methods used to estimate the precision matrix are based on maximum-likelihood estimation. For example, see [2, 22, 21, 23] for the details of these MLE methods. Other methods include neighborhood selection [50] and linear programming approaches. A first linear programming approach is introduced in [68]. The estimation is decomposed into $d$ subproblems, and each column of $\Omega^0$ is estimated separately. For the $i$-th subproblem, we first solve the following convex optimization problem:

$$
\min_{\beta} \| \beta \|_1 \quad \text{subject to:} \quad \| S_{i\setminus i} \beta - S_{ii} \|_\infty \leq \lambda \quad \text{and} \quad \beta \in \mathbb{R}^{d-1}. \tag{1.4.13}
$$

Let $\hat{\theta}^i$ denote the optimizer to (1.4.13), then we set the estimator to the $i$-th column of the precision matrix as the following: $\Omega_{ii} = (S_{ii} - 2\hat{\theta}^iT S_{i\setminus i} + \hat{\theta}^iT S_{i\setminus i} \hat{\theta}^i)^{-1}$ and $\Omega_{i\setminus i} = -\Omega_{ii} \hat{\theta}^i$. This process is repeated for $i = 1, \cdots, d$.

Another similar linear programming approach used to estimate sparse precision matrix, named Constrained $\ell_1$ minimization Estimation (CLIME) [8], also decomposes the estimation steps column-wise. It initially solves the following optimization problem:

$$
\min_{\Omega} \| \Omega \|_1 \quad \text{subject to:} \quad \| S \Omega - I_d \|_\infty \leq \lambda \quad \text{and} \quad \Omega \in \mathbb{R}^{d \times d}, \tag{1.4.14}
$$

21
where $S$ is the sample covariance matrix. We further decompose (1.4.14) into $d$ smaller problems:

For the $i$-th subproblem, we recover the $i$-th column of $\Omega$, denoted as $\beta$, by solving

$$
\min_{\beta} \|\beta\|_1 \quad \text{subject to: } \|S\beta - e_i\|_\infty \leq \lambda \text{ and } \beta \in \mathbb{R}^d.
$$

(1.4.15)

Similar to the previous two models, equation (1.4.13) and (1.4.15) and can be both casted into the parametric linear programming form. Take (1.4.15) as an example:

$$
\min_{\beta^+ , \beta^-} 1^T(\beta^+ + \beta^-)
$$

(1.4.16)

subject to \( \begin{pmatrix} S & -S \\ -S & S \end{pmatrix} \begin{pmatrix} \beta^+ \\ \beta^- \end{pmatrix} \leq \begin{pmatrix} \lambda 1 + e_i \\ \lambda 1 - e_i \end{pmatrix} \), \( \beta^+ , \beta^- \geq 0 \),

with

$$
A = \begin{pmatrix} S & -S \\ -S & S \end{pmatrix}, \quad b = \begin{pmatrix} e_i \\ -e_i \end{pmatrix}, \quad c = -1, \quad b = 1, \quad \bar{c} = 0, \quad x = \begin{pmatrix} \beta^+ \\ \beta^- \end{pmatrix}.
$$

A software package that solves CLIME using the parametric simplex method has already been developed and documented [54]. We are interested in extending the application of this package to other problems such as those described above. Chapter 2 gives a comprehensive description of this R package.

1.4.4 $\ell_1$-norm SVM

In the standard two-class classification problem, we are given a sequence of training data set \((x_1, y_1), ..., (x_n, y_n)\), where $x_i \in \mathbb{R}^d$ is the input and $y_i \in \{1,-1\}$ is the binary output. Each entries of a data point $x_i$ is called a feature. The Support
Vector Machine (SVM) is a powerful supervised learning method introduced in [12] used for classification problems. It finds the hyperplane that maximizes the distance to the nearest training data point in a certain space so that the generalization error of the classifier is minimized. By changing the kernel functions and mapping the data points to different spaces, this method is able to classify data in a high-dimensional feature space. The exact solution to the linear space $l_2$-norm SVM can be found by Quadratic Programming. We are interested in the linear space $l_1$-norm SVM [73], which usually results in a better classification when there is redundant noise in features. The constrained $l_1$-norm SVM solves the following problem:

$$\min_{\beta_0, \beta} \sum_{i=1}^{n} [1 - y_i(\beta_0 + \beta^T x_i)]_+ \quad \text{subject to: } ||\beta||_1 \leq \lambda, \quad (1.4.17)$$

where $\beta_0 \in \mathbb{R}$, $\beta \in \mathbb{R}^d$. When new data $x_k$ comes, we use the sign of $\beta_0 + \beta^T x_k$ to classify it. Let $z_i = 1 - y_i(\beta_0 + \beta^T x_i)$, for $i = 1, ..., n$. $z_i$ can be expressed as $z_i = z_i^+ - z_i^-$. Notice $[1 - y_i(\beta_0 + \beta^T x_i)]_+$ can be represented by $z_i^+$. We split $\beta$ and $\beta_0$ into positive and negative parts as well: $\beta = \beta^+ - \beta^-$ and $\beta_0 = \beta_0^+ + \beta_0^-$ and add slack variable $w$ to the constraint so that the constraint becomes equality: $\beta^+ + \beta^- + w = \lambda 1$, $w \geq 0$. Now we are ready to cast the problem into the equality parametric simplex form (1.1.1). We identify each component of (1.1.1) as the following:

$$x = \left( z^+ \quad z^- \quad \beta^+ \quad \beta^- \quad \beta_0^+ \quad \beta_0^- \quad w \right)^T \in \mathbb{R}^{(n+1) \times (2n+3d+2)}, x \geq 0,$$

$$c = \left( -1^T \quad 0^T \quad 0^T \quad 0^T \quad 0 \quad 0 \quad 0^T \right)^T \in \mathbb{R}^{2n+3d+2}, \quad \bar{c} = 0 \in \mathbb{R}^{2n+3d+2}.$$
\[ b = \begin{pmatrix} 1^T & 0 \end{pmatrix}^T \in \mathbb{R}^{n+1}, \quad \bar{b} = \begin{pmatrix} 0^T & 1 \end{pmatrix}^T \in \mathbb{R}^{n+1}, \]

and finally

\[ A = \begin{pmatrix} I_n & -I_n & Z & -Z & y & -y \\ 1^T & 1^T & 1^T \end{pmatrix} \in \mathbb{R}^{(n+1) \times (2n + 3d + 2)}, \]

where

\[ Z = \begin{pmatrix} y_1 x_1^T \\ \vdots \\ y_n x_n^T \end{pmatrix} \in \mathbb{R}^{n \times d}. \]

### 1.4.5 Differential Network and its Sparse Formulation

We consider the problem of direct estimation of differential networks in high dimensional graphical models. Many recent different applications of differential networking analysis can be founded in the biological literature \[39, 1, 40\]. The changes in graph structure has been modeled in \[45\] between two different conditions as a differential network which encodes the conditional dependency information. Sometimes, it is more interesting to analyze the magnitude difference of two graphs when they have the same structure \[69, 14\]. Let \( x = (x^1, \ldots, x^d)^T \) and \( y = (y^1, \ldots, y^d)^T \) be two different \( d \)-dimensional random vectors, where \( x \sim N_d(\mu_x^0, \Sigma_x^0) \) and \( y \sim N_d(\mu_y^0, \Sigma_y^0) \). Let \( \{x_1, \ldots, x_{n_1}\} \) be \( n_1 \) independent and identically distributed samples from the distribution of \( x \), and similarly, let \( \{y_1, \ldots, y_{n_2}\} \) be \( n_2 \) independent and identically distributed samples from the distribution of \( y \). We are interested in estimating the difference of the precision matrices: \( \Delta^0 = (\Sigma_x^0)^{-1} - (\Sigma_y^0)^{-1} \). We define the sample covariance matrix in the usual way: \( S_x = \frac{1}{n_1} \sum_{j=1}^{n_1} (x_j - \bar{x})(x_j - \bar{x})^T \), where \( \bar{x} = \frac{1}{n_1} \sum_{j=1}^{n_1} x_j \). The sample covariance matrix \( S_y \) for \( y \) is defined similarly.
The most natural and straightforward way is to estimate the difference is to estimate the precision matrices separately and subtract. All recent sparse precision matrix estimation methods we discussed in the CLIME subsection can be used such estimation. But, another approach is to estimate the two precision matrices jointly using the assumption that they have similar features [11, 29, 14]. It has been proposed in [69] that the difference matrix $\Delta^0$ can be directly estimated by an approach similar to CLIME. Compared with separate estimation of the two matrices, this direct estimation does not require both precision matrices to be sparse; instead, only relative sparsity is important in this approach. This advantage allows each network to contain a hub node structure. This structure is very important in real-world problems since normally a differential network is considered as changes in the graph structure which should be relatively sparse. The estimator proposed in [69], solves the following problem:

$$\min_{\Delta} \|\Delta\|_1 \quad \text{subject to: } \|S_x\Delta S_y - S_x + S_y\|_{\infty} \leq \lambda, \quad \Delta \in \mathbb{R}^{d \times d}, \quad (1.4.18)$$

where $S_x$ and $S_y$ are the sample covariance matrices.

Problem (1.4.18) can be formulated as a parametric linear programming problem as we’ll now explain in slightly more generality. In general, we are interested in solving:

$$\min_{D} \|D\|_1 \quad \text{subject to: } \|XDZ - Y\|_{\infty} \leq \lambda, \quad D \in \mathbb{R}^{n_1 \times n_2}, \quad (1.4.19)$$

where $X \in \mathbb{R}^{m_1 \times n_1}$, $Z \in \mathbb{R}^{n_2 \times m_2}$ and $Y \in \mathbb{R}^{m_1 \times m_2}$ are given data matrix.

Instead of directly writing the problem as a double matrix product $XDZ$, we break the product into a two-step process: $C = XD$ and $CZ$. Although the number of the unknown variables increases, we get a sparse rather than dense constraint matrix and this sparsity increases the speed of the algorithm significantly. To address the nonlinearity of the matrix one-norm, we write the matrix $D$ as $D = D^+ - D^-$, where

25
the components of the matrices \( \mathbf{D}^+ \) and \( \mathbf{D}^- \) are all nonnegative and represent the positive and negative parts of matrix \( \mathbf{D} \) at optimality. Now problem (1.4.19) becomes

\[
\min_{\mathbf{D}^+, \mathbf{D}^-} 1^T (\mathbf{D}^+ + \mathbf{D}^-) \mathbf{1}
\]

subject to: \( \| \mathbf{CZ} - \mathbf{Y} \|_\infty \leq \lambda, \quad \mathbf{X}(\mathbf{D}^+ - \mathbf{D}^-) = \mathbf{C}, \quad \mathbf{D}^+, \mathbf{D}^- \geq \mathbf{0} \).

The next step is to decompose the matrices \( \mathbf{D}^+, \mathbf{D}^-, \mathbf{C} \) and \( \mathbf{Y} \) column by column.

Let \( \text{vec}(\mathbf{D}^+), \text{vec}(\mathbf{D}^-), \text{vec}(\mathbf{C}) \) and \( \text{vec}(\mathbf{Y}) \) be the vectors obtained by stacking the columns of matrices \( \mathbf{D}^+, \mathbf{D}^- \mathbf{C} \) and \( \mathbf{Y} \), respectively. We write down (1.4.20) in the equality parametric linear programming form by adding positive slack variables to the inequality constraint:

\[
\max_x \mathbf{c}^T \mathbf{x} \quad \mathbf{A} \mathbf{x} = \bar{\mathbf{b}} \lambda + \mathbf{b}.
\]

After some careful calculation, it is straightforward to check matrix \( \mathbf{A} \) in (1.4.21) has the following form:

\[
\mathbf{A} = \begin{pmatrix}
\mathbf{X}^0 & -\mathbf{X}^0 & -\mathbf{I}_{m_1n_2} \\
\mathbf{Z}^0 & \mathbf{I}_{m_1m_2} \\
-\mathbf{Z}^0 & \mathbf{I}_{m_1m_2}
\end{pmatrix}
\]

with

\[
\mathbf{X}^0 = \begin{pmatrix}
\mathbf{X} \\
\vdots \\
\mathbf{X}
\end{pmatrix} \in \mathbb{R}^{m_1n_2 \times n_1n_2},
\]

\[
\mathbf{Z}^0 = \begin{pmatrix}
z_{11} \mathbf{I}_{m_1} & \cdots & z_{n_2} \mathbf{I}_{m_1} \\
\vdots & \ddots & \vdots \\
-z_{1m_2} \mathbf{I}_{m_1} & \cdots & -z_{n_2m_2} \mathbf{I}_{m_1}
\end{pmatrix} \in \mathbb{R}^{n_1m_2 \times m_1n_2},
\]
where \( z_{ij} \) denotes the \((i, j)\) entry of matrix \( Z \).

\[
x = \begin{bmatrix} \text{vec}(D^+) & \text{vec}(D^-) & \text{vec}(C) & w \end{bmatrix}^T \in \mathbb{R}^{2n_1n_2 + m_1n_2 + 2m_1m_2},
\]

\(\text{vec}(D^+), \text{vec}(D^-), w \geq 0,\)

where \( w \in \mathbb{R}^{2m_1m_2} \) is nonnegative slack variable vector used to make the inequality become an equality. Note that the \( \text{vec}(C) \) component does not have the nonnegative constraint but we can always declare it to be the basic variables. We identify the vectors \( x, b, c \) and \( \bar{b} \) in (1.4.21) as the following:

\[
b = \begin{bmatrix} 0^T & \text{vec}(Y) & -\text{vec}(Y) \end{bmatrix}^T \in \mathbb{R}^{m_1n_2 + 2m_1m_2},
\]

where the first \( m_1n_2 \) components of vector \( b \) are 0 and the rest components are from matrix \( Y \);

\[
\bar{b} = \begin{bmatrix} 0^T & 1^T & 1^T \end{bmatrix}^T \in \mathbb{R}^{m_1n_2 + 2m_1m_2},
\]

where the first \( m_1n_2 \) components of vector \( \bar{b} \) are 0 and the rest \( 2m_1m_2 \) components are 1;

\[
c = \begin{bmatrix} -1^T & -1^T & 0^T & 0^T \end{bmatrix}^T \in \mathbb{R}^{2n_1n_2 + m_1n_2 + 2m_1m_2},
\]

where the first \( 2n_1n_2 \) components of vector \( c \) are \(-1\) and the rest \( m_1n_2 + 2m_1m_2 \) components are 0.

### 1.5 Initialization of the Simplex Method

It is important to declare the initial basic and nonbasic variables correctly. For the inequality form (1.1.2), it is very natural to add slack variables \( w \) to make the constraint be an equality and the initial slack variables \( w \) can be chosen as the basic variables since an identity matrix is obviously invertible. If the problem is in equality
form (1.1.1), we need to manually select the basic variables so that the matrix \( A_B \) is invertible. For the LAD-Lasso problem in (1.4.6), it is very natural to select \( t^- \) as the basic variables. For the differential network problem, we are interested in solving (1.4.21). We explain how to select the basic variables in this case.

First, recall that the nonbasic variables are used to represent the basic variables. As a result, the coefficient of the matrix associated with the basic variables should be sparse so it can be “easily” inverted. In fact, we choose the first \( 2n_1n_2 \) variables from \( \text{vec}(D^+) \) and \( \text{vec}(D^-) \) to be nonbasic variables and the rest \( m_1n_2 + 2m_1m_2 \) variables from \( \text{vec}(C) \) to be basic variables. The slack variables added at the end in order to make the constraints be equality, of course, are basic variables. Second, the variables \( \text{vec}(D^+) \), \( \text{vec}(D^-) \) and slack variables \( w \) are nonnegative, but \( \text{vec}(C) \) are free variables. While we are implementing the parametric simplex method, we simply declare that the variables \( \text{vec}(C) \) are always basic. That means they never swap with other variables and this trick improves the speed of the algorithm significantly. Despite the fact that this matrix has large dimension, this formulation can still be solved efficiently because the constraint matrix and the vectors are quite sparse.

### 1.6 Theoretical Justification

We present our theoretical analysis on solving Dantzig selector using PSM. This analysis guarantees the sparsity of the primal and dual solutions within each update of PSM. Specifically, given \( X \in \mathbb{R}^{n \times d}, y \in \mathbb{R}^n \), we consider a linear model \( y = X\theta^* + \epsilon \), where \( \theta^* \) is the unknown sparse regression coefficient vector with \( \|\theta^*\|_0 = s^* \), and \( \epsilon \sim N(0, \sigma^2 I_n) \). We show that PSM always maintains a pair of sparse primal and dual solutions. Therefore, the computation cost within each iteration of PSM can be significantly reduced. Before we proceed with our main result, we introduce two
assumptions. The first assumption requires the regularization factor to be sufficiently large.

**Assumption 1.6.1.** Suppose that PSM solves (1.4.3) for a regularization sequence \( \{\lambda_K\}_{K=0}^N \). The smallest regularization factor \( \lambda_N \) satisfies

\[
\lambda_N = C\sigma \sqrt{\frac{\log d}{n}} \geq 4\|X^\top \epsilon\|_{\infty} \text{ for a generic constant } C.
\]

Existing literature has extensively studied Assumption 1.6.1 for high dimensional statistical theories. Such an assumption enforces all regularization parameters to be sufficiently large to eliminate irrelevant coordinates along the regularization path. Note that Assumption 1.6.1 is deterministic for any given \( \lambda_N \). Existing literature has verified that for sparse linear regression models, given \( \epsilon \sim N(0, \sigma^2 I_n) \), Assumption 1.6.1 holds with overwhelming probability.

Before we proceed, we define the largest and smallest \( s \)-sparse eigenvalues of \( n^{-1}X^\top X \) as follows.

**Definition 1.6.2.** Given an integer \( s \geq 1 \), we define

\[
\rho_+(s) = \sup_{\|\Delta\|_0 \leq s} \frac{\Delta^T X^\top X \Delta}{n\|\Delta\|_2^2},
\]

\[
\rho_-(s) = \inf_{\|\Delta\|_0 \leq s} \frac{\Delta^T X^\top X \Delta}{n\|\Delta\|_2^2}.
\]

We then introduce our second assumption.

**Assumption 1.6.3.** Given \( \|\theta^*\|_0 \leq s^* \), there exists an integer \( \tilde{s} \) such that

\[
\tilde{s} \geq 100\kappa s^*, \quad \rho_+(s^* + \tilde{s}) < +\infty, \quad \text{and} \quad \tilde{\rho}_-(s^* + \tilde{s}) > 0,
\]

where \( \kappa \) is defined as \( \kappa = \rho_+(s^* + \tilde{s})/\tilde{\rho}_-(s^* + \tilde{s}) \).

Assumption 1.6.3 guarantees that \( n^{-1}X^\top X \) satisfies the sparse eigenvalue conditions as long as the number of active irrelevant blocks never exceeds \( \tilde{2}s \) along the
solution path. That is closely related to the restricted isometry property (RIP) and restricted eigenvalue (RE) conditions, which have been extensively studied in existing literature.

We then characterize the sparsity of the primal and dual solutions within each iteration.

**Theorem 1.6.4** (Primal and Dual Sparsity). Suppose that Assumptions 1.6.1 and 1.6.3 hold. We consider an alternative formulation to the Dantzig selector,

\[
\hat{\theta}^{\lambda'} = \arg\min_{\theta} \|\theta\|_1 \quad \text{s.t.} \quad -|\nabla_j \mathcal{L}(\theta)| \leq \lambda'.
\]

(1.6.1)

Let \(\hat{\mu}^{\lambda'} = [\hat{\mu}_1^{\lambda'}, \ldots, \hat{\mu}_d^{\lambda'}, \hat{\gamma}_{d+1}^{\lambda'}, \ldots, \hat{\gamma}_{2d}^{\lambda'}]^T\) denote the optimal dual variables to (1.6.1). For any \(\lambda' \geq \lambda\), we have \(\|\mu^{\lambda'}\|_0 + \|\gamma^{\lambda'}\|_0 \leq s^* + \tilde{s}\). Moreover, given design matrix satisfying

\[
\|X^\top S X S (X^\top S X S)^{-1}\|_\infty \leq 1 - \zeta,
\]

where \(\zeta > 0\) is a generic constant, \(S = \{j \mid \theta_j^* \neq 0\}\) and \(\overline{S} = \{j \mid \theta_j^* = 0\}\), we have \(\|\hat{\theta}^{\lambda'}\|_0 \leq s^*\).

We provide the proof of Theorem 1.6.4. Theorem 1.6.4 shows that within each iteration, both primal and dual variables are sparse, i.e., the number of nonzero entries are far smaller than \(d\). Therefore, the computation cost within each iteration of PSM can be significantly reduced by a factor of \(O(d/s^*)\). This partially justifies the superior performance of PSM in sparse learning.

For notational simplicity, we omit the superscript \(\lambda'\) in \(\hat{\mu}\). Before we proceed with the statistical properties of the Dantzig selector, we first introduce the following lemmas.
Lemma 1.6.5 ([6]). Suppose that Assumptions 1.6.1 and 1.6.3 hold. Define $\tilde{\Delta} = \hat{\theta} - \theta^*$. We have

$$\|\tilde{\Delta}\|_1 \leq \|\hat{\Delta}\|_1. \quad (1.6.2)$$

Moreover, we have

$$\min_{\|\Delta\|_1 \leq \|\hat{\Delta}\|_1} \frac{\Delta^T \nabla^2 \mathcal{L}(\theta) \Delta}{\|\Delta\|_2^2} \geq \frac{\rho_-(s^* + 2\bar{s})}{4}. \quad (1.6.3)$$

The proof of Lemma 1.6.5 is provided in [6], and therefore is omitted. Note that (1.6.2) in Lemma 1.6.5 implies that $\hat{\theta}$ lies in a restricted cone-shape set, and (1.6.3) implies that (1.6.2) combined with Assumption 1.6.3 implies the restricted eigenvalue condition. The next lemma presents the statistical rates of convergence of the Dantzig selector.

Lemma 1.6.6 ([9]). Suppose that Assumptions 1.6.1 and 1.6.3 hold. We have

$$\|\Delta\|_2 = \frac{C_1 \sqrt{s^* \lambda}}{\rho_-(s^* + 2\bar{s})} \quad \text{and} \quad \|\Delta\|_1 = \frac{C_2 s^* \lambda}{\rho_-(s^* + 2\bar{s})}. \quad (1.6.4)$$

The proof of Lemma 1.6.6 is provided in [9], and therefore is omitted. Based on Lemmas 1.6.5 and 1.6.6, we can further characterize the statistical properties of $\nabla \mathcal{L}(\hat{\theta})$ in the following lemma.

Lemma 1.6.7. Suppose that Assumptions 1.6.1 and 1.6.3 hold. We have

$$\left| \left\{ j \mid |\nabla_j \mathcal{L}(\hat{\theta})| \geq \frac{3\lambda}{4}, \ j \in \mathcal{S} \right\} \right| \leq \bar{s}. \quad (1.6.5)$$

Proof. By Assumption 1.6.1 we have $\lambda \geq 8\|\nabla \mathcal{L}(\theta^*)\|_{\infty}$, which further implies

$$\left| \left\{ j \mid |\nabla_j \mathcal{L}(\theta^*)| \geq \lambda/8, \ j \in \mathcal{S} \right\} \right| = 0. \quad (1.6.6)$$
We then consider an arbitrary set $S'$ such that

$$S' = \{ j \mid |\nabla_j \mathcal{L}(\hat{\theta}) - \nabla_j \mathcal{L}(\theta^*)| \geq 5\lambda/8, \ j \in \mathcal{S} \}.$$ 

Let $s' = |S'|$. Then there exists $v$ such that

$$\|v\|_\infty = 1, \ \|v\|_0 \leq s', \ \text{and} \ 5s'\lambda/8 \leq v^T(\nabla \mathcal{L}(\hat{\theta}) - \nabla \mathcal{L}(\theta^*)).$$

Since $\mathcal{L}(\hat{\theta})$ is twice differentiable, then by the mean value theorem, there exists some $z_1 \in [0, 1]$ such that

$$\ddot{\theta} = z_1 \theta + (1 - z_1)\theta^* \ \text{and} \ \nabla \mathcal{L}(\hat{\theta}) - \nabla \mathcal{L}(\theta^*) = \nabla^2 \mathcal{L}(\ddot{\theta}) \Delta.$$ 

Then we have

$$\frac{5s'\lambda}{8} \leq v^T \nabla^2 \mathcal{L}(\ddot{\theta}) \Delta \leq \sqrt{v^T \nabla^2 \mathcal{L}(\ddot{\theta}) v} \sqrt{\Delta^T \nabla^2 \mathcal{L}(\ddot{\theta}) \Delta}.$$ 

Since we have $\|v\|_0 \leq s'$, then we obtain

$$\frac{3s'\lambda}{4} \leq \sqrt{\rho_+(s') \sqrt{s'} \Delta^T (\nabla \mathcal{L}(\hat{\theta}) - \nabla \mathcal{L}(\theta^*))} \leq \sqrt{\rho_+(s') \sqrt{s'} \|\Delta\|_1 \cdot \|\nabla \mathcal{L}(\hat{\theta}) - \nabla \mathcal{L}(\theta^*)\|_\infty} \leq \sqrt{\rho_+(s') \sqrt{s'} \|\Delta\|_1 (\|\nabla \mathcal{L}(\hat{\theta})\|_\infty + \|\nabla \mathcal{L}(\theta^*)\|_\infty)} \leq \sqrt{\rho_+(s') \sqrt{s'} \frac{115s^*\lambda^2}{12\rho_-(s^* + \hat{s})}}.$$ 

By simple manipulation, we have

$$\frac{5\sqrt{s'}}{8} \leq \sqrt{\rho_+(s')} \sqrt{\frac{115s^*}{12\rho_-(s^* + \hat{s})}}.$$ 

which implies

\[ s' \leq \frac{184\rho_+(s')}{15\rho_-(s^* + \tilde{s})} \cdot s*. \]

Since \( s' = |S'| \) attains the maximum value such that \( s' \leq \tilde{s} \) for arbitrary defined subset \( S' \), we obtain \( s' \leq \tilde{s} \). Then by simple manipulation, we have

\[ |\{ j \mid |\nabla_j \mathcal{L}(\hat{\theta}) - \nabla_j \mathcal{L}(\theta^*)| \geq 5\lambda/8, \ j \in \overline{S}\}| \leq 13\kappa s^* < \tilde{s}. \quad (1.6.7) \]

Thus, (1.6.6) and (1.6.7) imply

\[ |\{ j \mid |\nabla_j \mathcal{L}(\hat{\theta})| \geq 3\lambda/4, \ j \in \overline{S}\}| \leq \tilde{s}. \]

By the complementary slackness, we have \( \hat{\mu}_j(\nabla_j \mathcal{L}(\hat{\theta}) - \lambda) = 0 \) and \( \hat{\gamma}_j(-\nabla_j \mathcal{L}(\hat{\theta}) - \lambda) = 0 \). By (1.6.7), we know

\[ |\{ j \mid \hat{\mu}_j \neq 0 \ or \ \hat{\gamma}_j \neq 0, \ j \in \overline{S}\}| \leq \tilde{s}. \quad (1.6.8) \]

Thus, we show that the optimal dual variables are sparse. The cardinality is at most \( 2s^* + \tilde{s} \).

To control the sparsity of the primal variables, we directly use the following lemma.

**Lemma 1.6.8** ([26]). Suppose that Assumptions 1.6.1 and 1.6.3 hold. Given the design matrix satisfying

\[ \|X_S^\top X_S(X_S^\top X_S)^{-1}\|_\infty \leq 1 - \zeta, \]

where \( \zeta > 0 \) is a generic constant, we have \( \hat{\theta}_j = 0 \) for any \( j \in \overline{S} \).
The proof of Lemma \(\text{(1.6.8)}\) is provided in [26]. Lemma \(\text{1.6.8}\) guarnatees that \(\hat{\theta}\) does not select any irrelevant coordinates. Thus, we complete the proof.

1.7 Numerical Experiment

In this section, we present some numerical experiments and give some insights about how the parametric simplex method solves different linear programming problems. We verify the following assertions: (1) The parametric simplex method requires very few iterations to identify the nonzero component if the original problem is sparse. (2) The parametric simplex method is able to find the full solution path by solving the problem only once. It is fast, efficient and precise. (3) The parametric simplex method maintains the feasibility of the problem up to machine precision along the solution path.

1.7.1 Solution path of Dantzig selector

We start with a simple example that illustrates how the recovered solution path of the Dantzig selector model changes as the parametric simplex method iterates. We adopt the example used in [9]. The design matrix \(X\) has \(n = 100\) rows and \(d = 250\) columns. The entries of \(X\) are generated from an array of independent Gaussian random variables that are then normalized so that each column has a given norm. We randomly select \(s = 8\) entries from the response vector \(\beta^0\), and set them as \(\beta_i^0 = s_i(1 + a_i)\), where \(s_i = 1\) or \(-1\), with probability \(1/2\) and \(a_i \sim \mathcal{N}(0, 1)\). The other entries of \(\beta^0\) are set to zero. We form \(y = X\beta^0 + \epsilon\), where \(\epsilon_i \sim \mathcal{N}(0, \sigma)\), with \(\sigma = 1\). We stop the parametric simplex method when \(\lambda \leq \sigma n \sqrt{\log d/n}\). The solution path of the result is shown in Figure 1.1(a). We see that our method correctly identifies all nonzero entries of \(\beta\) in less than 10 iterations. Some small overestimations occur in a few iterations after all nonzero entries have been identified. We also show how
1.7.2 Feasibility of Dantzig Selector

Another advantage of the parametric simplex method is that the solution is always feasible along the path while other estimating methods usually generate infeasible solutions along the path. We compare our algorithm with the R package “flare” [48] which uses the Alternating Direction Method of Multipliers (ADMM) using the same example described above. We compute the values of $\|X^T X \beta^i - X^T y\|_{\infty} - \lambda_i$ along the solution path, where $\beta^i$ is the $i$-th basic solution (with corresponding $\lambda_i$).
obtained while the parametric simplex method is iterating. Without any doubts, we always obtain 0s during each iteration. We plug the same list of $\lambda_i$ into “flare” and compute the solution path for this list as well. As expected, Table 1.1 shows that the parametric simplex method is always feasible along the path since it is solving each iteration up to machine precision; while the solution path of the ADMM is almost always breaking the feasibility by a large amount, especially in the first few iterations which correspond to large $\lambda$ values. Each experiment is repeated for 100 times.

Table 1.1: Average feasibility violation with standard errors along the solution path

<table>
<thead>
<tr>
<th></th>
<th>Maximum violation</th>
<th>Minimum Violation</th>
</tr>
</thead>
<tbody>
<tr>
<td>ADMM</td>
<td>498(122)</td>
<td>143(73.2)</td>
</tr>
<tr>
<td>PSM</td>
<td>0(0)</td>
<td>0(0)</td>
</tr>
</tbody>
</table>

1.7.3 Performance Benchmark of Dantzig Selector

In this part, we compare the timing performance of our algorithm with R package “flare”. We fix the sample size $n$ to be 200 and vary the data dimension $d$ from 100 to 5000. Again, each entries of $X$ is independent Gaussian and normalized so that the column has uniform norm. We randomly select 2% entries from vector $\beta$ to be nonzero and each entry is chosen as $\sim \mathcal{N}(0, 1)$. We compute $y = X\beta + \epsilon$, with $\epsilon_i \sim \mathcal{N}(0, 1)$ and try to recover vector $\beta$, given $X$ and $y$. Our method stops when $\lambda$ is less than $2\sigma \sqrt{\log d/n}$, so that the full solution path for all the values of $\lambda$ up to this value is computed by the parametric simplex method. In “flare”, we estimate $\beta$ when $\lambda$ equals that number in the Dantzig selector model. This means “flare” has much less computation task than the parametric simplex method. As we can see in Table 1, our method has a much better performance than “flare” in terms of speed. We compare and present the timing performance of the two algorithms in seconds and each experiment is repeated for 100 times. In practice, only very few iterations is required when the response vector $\beta$ is sparse.
Table 1.2: Average timing performance (in seconds) with standard errors in the parentheses on Dantzig selector

<table>
<thead>
<tr>
<th>Size (d)</th>
<th>500</th>
<th>1000</th>
<th>2000</th>
<th>5000</th>
</tr>
</thead>
<tbody>
<tr>
<td>Flare</td>
<td>19.5(2.72)</td>
<td>44.4(2.54)</td>
<td>142(11.5)</td>
<td>1500(231)</td>
</tr>
<tr>
<td>PSM</td>
<td>2.40(0.220)</td>
<td>29.7(1.39)</td>
<td>47.5(2.27)</td>
<td>649(89.8)</td>
</tr>
</tbody>
</table>

1.7.4 Performance Benchmark of Differential Network

We now apply this optimization method to the Differential Network model. We need two covariance matrices and the difference of their inverse has to be sparse. We generate $\Sigma^0_x = U^T \Lambda U$, where $\Lambda \in \mathbb{R}^{d \times d}$ is a diagonal matrix and its entries are i.i.d. and uniform on $[1,2]$, and $U \in \mathbb{R}^{d \times d}$ is a random matrix with i.i.d. entries from $\mathcal{N}(0,1)$. Let $D_1 \in \mathbb{R}^{d \times d}$ be a random sparse symmetric matrix with a certain sparsity level. Each entry of $D_1$ is i.i.d. and from $\mathcal{N}(0,1)$. We set $D = D_1 + 2|\lambda_{\min}(D_1)|I_d$ in order to guarantee the positive definiteness of $D$, where $\lambda_{\min}(D_1)$ is the smallest eigenvalue of $D_1$. Finally, we let $\Omega^0_x = (\Sigma^0_x)^{-1}$ and $\Omega^0_y = \Omega^0_x + D$.

We then generate data of sample size $n = 100$. The corresponding sample covariance matrices $S_x$ and $S_y$ are also computed based on the data. Unfortunately, we are not able to find other software which can efficiently solve this problem, so we only list the timing performance of our algorithm as dimension $d$ varies from 25 to 200 in Table 2. We stop our algorithm whenever the solution achieved the desired sparsity level. When $d = 25$, 50 and 100, the sparsity level of $D_1$ is set to be 0.02 and when $d = 150$ and 200, the sparsity level of $D_1$ is set to be 0.002. Each experiment is repeated by 100 times.

Table 1.3: Average timing performance (in seconds) and iteration numbers with standard errors in the parentheses on differential network

<table>
<thead>
<tr>
<th>Size (d)</th>
<th>25</th>
<th>50</th>
<th>100</th>
<th>150</th>
<th>200</th>
</tr>
</thead>
<tbody>
<tr>
<td>Timing</td>
<td>0.0185(0.00689)</td>
<td>0.376(0.124)</td>
<td>6.81(2.38)</td>
<td>13.41(1.26)</td>
<td>46.88(7.24)</td>
</tr>
<tr>
<td>Iteration Number</td>
<td>15.5(7.00)</td>
<td>55.3(18.8)</td>
<td>164(58.2)</td>
<td>85.8(16.7)</td>
<td>140(26.2)</td>
</tr>
</tbody>
</table>
Chapter 2

The Fastclime Package

In this chapter, we introduce an R package called fastclime, which provides a generic linear programming solver and a parameterized linear programming solver. It also contains a solver used for an important sparse precision matrix estimator named CLIME as well as a solver used for an important linear regression estimator called the Dantzig selector. This package efficiently implements the parametric simplex method. Originally, this package was written for solving CLIME and for that reason we named it fastclime. We later added more features to this package and now it is extremely popular in the statistical learning community. Compared with the existing packages clime and flare used for precision matrix estimation, fastclime has certain extra advantages: (1) it gives a full piece-wise linear regularization solution path of the CLIME tuning parameter for each column; (2) it only requires a few iterations to recover each sparse precision matrix column; (3) the entire solver is written in C and is easy to modify.

2.1 Design, Implementation and Examples

The package fastclime consists of three major parts: a series of functions used to estimate a precision matrix: data generator, graph estimation, graph visualization, a
Dantzig selector solver and a pair of linear programming solvers. We first focus on the design of the precision matrix estimator CLIME.

For the data generation part, the function `fastclime.generator` is able to generate multivariate Gaussian data with different graph structures. The first two parameters are $n$ and $d$, and $n$ is the size of the data and $d$ is the dimension of each data point. It is able to generate “random”, “hub”, “cluster” and “band” four different data structures used for testing purpose.

`Fastclime` is the main function used to estimate the graph. It takes three parameters. The first parameter can be either a data matrix or a sample covariance matrix. The second parameter is the smallest value of $\lambda$ the user would like to reach. The third parameter is the designed iteration numbers required for each column (the length of the solution path). The function estimates the precision matrix column by column, and stops when the required sparsity level has been reached at each column or the designed iteration number has been reached. The estimator is based on a parametric simplex linear programming solver written in C. In order to maintain the speed of the program, the default path length is set to be 50. The user must be very careful when asking for a large solution path when the dimension $d$ is large, as storing a list of large matrices could easily cause memory issues. This estimator is designed to take advantage of the sparsity of the precision matrix, so when the precision matrix is not sparse, it could take a very long time to recover the precision matrix and the result is therefore meaningless.

The output object of the main function `fastclime` has several components. The list of precision matrices is stored in `icovlist`. For each column, it stores the parameter value in `lambdamtx` when the solution changes, with zero filled in when the sparsity level has been achieved in that column. `maxnlambda` is the length of the solution path.
Fastclime.selector is the selector function of fastclime. It takes three parameters: icovalist, maxnlamba and lambda. The first two are taken directly from the output object of fastclime and the third one is the user-specified parameter $\lambda$. The selector gives the estimated precision matrix corresponding to this $\lambda$.

The plotting functions fastclime.plot provides a method to plot the undirected graph at each path. Please note only the adjacency matrix is allowed for this function.

We illustrate the user interface by two examples. The first one is based on the data generated by fastclime.generator():

```r
> library(fastclime) # Load the package
> L = fastclime.generator(n=200,d=50,graph="hub") # Generate the data
> out0 = fastclime(L$data,0.05)
> out = fastclime.selector(out0$lambdamtx, out0$icovlist, 0.1)
> fastclime.plot(out$adaj)
```

(a) Truth  (b) Graph obtained in 3 iterations  (c) Graph obtained in 4 iterations

Figure 2.1: Graphs estimated by fastclime

In this example, we first generate 200 samples from a 50-dimensional Gaussian distribution with hub structure. All information, such as the true covariance matrix, the sample covariance matrix, the true precision matrix and the adjacency matrix are stored in object $L$. We estimate the inverse covariance matrix by fastclime, and stops when $\lambda$ reaches 0.05 in each column. The estimator we obtain is stored in an
object called \textit{out0}. We use \texttt{fastclime.selector} to determine the precision matrix at the desired value of $\lambda$. The true sparsity level corresponds to $\lambda = 0.06$. After three
iterations, it recovers all the tree correlation between the variables. The user has to
be aware that as the path length increases, it is likely to obtain additional undirected
edges in the graph, as shown in Figure 2.1(c).

The second example is based on some stock market data which is contained in
this package. The data contains stock price from S&P 500 during the period between
Jan 1st, 2003 and Jan 1st, 2008. It gives 1258 samples (n) from 452 stocks (d). We
let an approximate sparsity ratio be 0.05. The program automatically calculates the
solution path and it reaches the sparsity level 0.05 after 30 iterations. The sparsity
level versus the parameter $\lambda$ is shown in Figure 2.2 and a few examples of estimated
graphs with corresponding sparsity level labeled are shown in Figure 2.3.

```R
> data(stockdata)
> Y = log(stockdata$data[2:1258,]/stockdata$data[1:1257,])
> out = fastclime(Y,0.006,200)
> out0 = fastclime.selector(out$lambdamtx, out$icovlist, 0.06)
> fastclime.plot(out0$adaj)
> out0 = fastclime.selector(out$lambdamtx, out$icovlist, 0.05)
> fastclime.plot(out0$adaj)
> out0 = fastclime.selector(out$lambdamtx, out$icovlist, 0.04)
> fastclime.plot(out0$adaj)
```

Our package can also be used for solving Dantzig selector. \texttt{dantzig} is the main
function used to solve the linear regression problem. It takes four parameters. The
first two are the regressor matrix and the response vector. Next two parameters are
the same as in \texttt{fastclime}. They are the smallest value of $\lambda$ and the number of
iterations the user would like to reach. \texttt{dantzig.selector} is the selector function.
used to select the solution to the Dantzig selector for any given value of $\lambda$. The first two arguments are taken directly from the output of \texttt{dantzig}. Here is an example:

\begin{verbatim}
> d = 250
> n = 200
> sparsity = 8
> sigma0 = 1e-1
> L = dantzig.generator(d, n, sparsity, sigma0)
> out0 = dantzig(L$X0, L$y, lambda=sigma0*sqrt(log(d))*sqrt(n), 1500)
> out = dantzig.selector(out0$lambdalist, out0$BETA0, 6.0)
\end{verbatim}
The linear programming solver \texttt{fastlp} is used to solve a general linear programming problem in standard inequality form. The solver automatically gives a random perturbation and solves it by the parametric simple method. If the original problem is not in this form, the user has to convert it into the standard inequality form. For example, if one expression is an equation instead of an inequality, then this equation shall be separated into two inequalities in order to convert the original problem into standard inequality form. \texttt{fastlp} has four parameters. The first three are the objective vector (length \( n \)), the constraint matrix (dimension \( m \times n \)) and the right hand side vector (length \( m \)). The last parameter \texttt{lambda} is used to specify the stopping rule for the linear programming solver. Whenever the calculated parameter \( \lambda \) in the solver is smaller than this value, the program stops at that point and the function gives the optimal solution corresponding to that \( \lambda \). The default value of \( \lambda \) is zero, which corresponds to the optimal value of the original linear programming problem. The function will also indicate if the input problem is unbounded or infeasible. For example, we generate a random feasible linear programming problem and solve it by \texttt{fastlp} as follows:

\begin{verbatim}
> n=100
> A=matrix(mvrnorm(n^2,0, 1),nrow=n)
> x=runif(n, min = 0, max = 5)
> y=runif(n, min = 0, max = 5)
> w=runif(n, min = 0, max = 5)
> z=runif(n, min = 0, max = 5)
> b=A%*%x+w
> c=t(A)%*%y-z
> out1<-fastlp(c,A,b)
\end{verbatim}

The other linear programming solver \texttt{paralp} is used to solve a parameterized linear programming problem in [1.1.2]. It takes six parameters. The only difference
between `paralp` and `fastlp` is that this time the user can specify the two perturbation vectors $\bar{b}$ and $\bar{c}$ in (1.1.2). The rest four parameters are the same as in `fastlp`. Again, the function stops at the value of `lambda` provided by the user and gives the optimal solution corresponding to that value. Note that the perturbation vector $\bar{b}$ must be nonnegative and $\bar{c}$ must be nonpositive in order to correctly apply the parametric simplex method. Here is a simple example:

```r
> c<-c(1,2,1,1)
> b<-c(8,12,18)
> A<-matrix(c(2,2,3,1,2,1,5,0,2,1,4,0),nrow=4)
> b_bar<-c(1,1,1)
> c_bar<-c(1,1)
> opt2<-paralp(c,A,b,c_bar,b_bar)
```

### 2.2 Performance Benchmark

Since CLIME has many advantages over MLE-type methods, we focus on the comparison between packages based on the CLIME method. We evaluate the timing performance of our package with comparison to the packages `flare` and `clime` while estimating the precision matrix. `Flare` [47] uses the Alternating Direction Method of Multiplier (ADMM) algorithm to evaluate CLIME and `clime` solves a series of LP problems for different values of $\lambda$. We simulate the data from several multivariate normal distributions. We fix the sample size $n$ to be 200 and vary the data dimension $d$ from 50 to 800. We generate our data by `fastclime.generator`, without any particular data structure.

The error tolerance (primal and dual gap) used for `clime` and `fastclime` is $10^{-5}$ and the error tolerance (differences between iterations) used for `flare` is also $10^{-5}$. Our package `fastclime` calculates its own $\lambda$ sequences while solving the LP only.
once, the $\lambda$ sequence is stored in the output. **Fastclime** solves the problem for the entire continuum of lambda values. The discrete list of values simply indicate where the primal and/or dual vectors change. As shown in Table 2.1, we list the average timing performance of three solvers and the numbers in parenthesis are standard deviations: **fastclime** performance is significantly faster than **clime** when $d$ is 50 or 100. When $d$ is large, we are not able to obtain results directly from **clime** within one hour. We also notice that, in most cases, **fastclime** performs better than **flare**, and it has a smaller deviation compared with **flare**. The units are in seconds and all experiments are carried on a PC with Intel Core i5-3320 2.6GHz processor and the R version used is 2.15.0.

<table>
<thead>
<tr>
<th>solve</th>
<th>$d=50$</th>
<th>$d=100$</th>
<th>$d=200$</th>
<th>$d=400$</th>
<th>$p=800$</th>
</tr>
</thead>
<tbody>
<tr>
<td>clime</td>
<td>103.52(9.11)</td>
<td>937.37(6.77)</td>
<td>N/A</td>
<td>N/A</td>
<td>N/A</td>
</tr>
<tr>
<td>flare</td>
<td>0.632(0.335)</td>
<td>1.886(0.755)</td>
<td>10.770(0.184)</td>
<td>74.106(33.940)</td>
<td>763.632(135.724)</td>
</tr>
<tr>
<td>fastclime</td>
<td>0.248(0.0148)</td>
<td>0.928(0.0268)</td>
<td>9.928(3.702)</td>
<td>53.038(1.488)</td>
<td>386.880(58.210)</td>
</tr>
</tbody>
</table>

### 2.3 Conclusions

We developed a new package named **fastclime** for generic and parameterized linear programming problems as well as conditional independence graph estimation and Dantzig selector estimation. The package is based on the parametric simplex method. Compared with the existing package **clime** and **flare**, it has many additional features when it is applied to solve for CLIME: it is much faster and it can recover the full piece-wise linear regularization solution path. We hope the CRAN community could benefit from our contributions. We plan to maintain and support this package in the future.
Chapter 3

The Inexact Peachman-Rachford Splitting Method

3.1 Introduction

We consider the following linearly constrained convex minimization problem with a separable objective in a finite dimensional space:

\[
\min \theta_1(x) + \theta_2(y) \quad (3.1.1)
\]

subject to: \( Ax + By = b \)

\( x \in \mathcal{X}, y \in \mathcal{Y}, \)

where \( \mathcal{X} \subset \mathbb{R}^{n_1} \) and \( \mathcal{Y} \subset \mathbb{R}^{n_2} \) are given closed and convex sets; \( \theta_1 : \mathcal{X} \to \mathbb{R} \) and \( \theta_2 : \mathcal{Y} \to \mathbb{R} \) are closed proper convex (but not necessarily smooth) functions; \( A \in \mathbb{R}^{m \times n_1} \) and \( B \in \mathbb{R}^{m \times n_2} \). We assume that the solution set of (3.1.1) is nonempty throughout this chapter.

46
Since the objective function has a separable structure, most solvers intend to solve the subproblems involving $\theta_1$ and $\theta_2$ separately. Douglas-Rachford Alternating Direction Method of Multiplier (ADMM), proposed in [27] (see also [10, 25]), as the application of proximal point algorithm to the dual of (3.1.1) (see [24]), is a very popular splitting method and have a variety of applications in image processing, engineering and statistical learning. The highlight of a few representative examples in different areas can be found in [4]. We summarize some basic notations used in this dissertation. We denote $\|x\| = \sqrt{x^T x}$ as the normal Euclidean-norm, and $\|A\|$ as the induced matrix norm of $A$, defined as $\max \|Ax\|$, with $\|x\| = 1$. $\|x\|_A$ is defined as $\sqrt{x^T Ax}$.

We first discuss how to formulate ADMM as a special case of the dual proximal point algorithm [3]. Consider the minimization of a closed proper convex function $\theta(x) : \mathcal{X} \to \mathbb{R}$, where $\mathcal{X} \subset \mathbb{R}^n$. The proximal algorithm is given by:

$$x^{k+1} = \arg \min_{x \in \mathcal{X}} \{ \theta(x) + \frac{1}{2\beta} \|x - x^k\|^2 \}, \quad (3.1.2)$$

where $x_0$ is an arbitrary starting point and $\beta$ is a positive scalar parameter. The quadratic regularization term $\|x - x^k\|^2$ makes the optimization problem strictly convex and guaranteeing that each update is unique and well-defined. The dual of (3.1.2) is given by

$$\min \theta^*(\lambda) - (x^k)^T \lambda + \frac{\beta}{2} \|\lambda\|^2, \quad (3.1.3)$$

where $\theta^*$ is the conjugate of $\theta$, defined as $\theta^*(\lambda) = \max\{x^T \lambda - \theta(x)\}$. The dual implementation of the proximal point algorithm is given by

$$\begin{cases}
\lambda^{k+1} = \arg \min_{\lambda} \{ \theta^*(\lambda) - (x^k)^T \lambda + \frac{\beta}{2} \|\lambda\|^2 \}, \\
x^{k+1} = x^k - \beta \lambda^{k+1}.
\end{cases} \quad (3.1.4)$$

47
Assuming \( x \in \mathcal{X}, y \in \mathcal{Y} \). Consider the corresponding primal and dual functions

\[
p(v) = \min_{Ax - b = v} \theta(x), \quad q(\lambda) = \min L(x, \lambda),
\]

where \( L(x, \lambda) = \theta(x) + \lambda^T (Ax - b) \) is the Lagrangian function. After applying the proximal point algorithm to the dual problem of maximizing \( q \), it has the following form:

\[
\lambda^{k+1} = \arg \max_\lambda \{q(\lambda) - \frac{1}{\beta} \|\lambda - \lambda^k\|^2\}, \quad (3.1.5)
\]

Since \( p \) and \( q \) are so it can be easily seen that dual proximal point algorithm takes the form:

\[
v^{k+1} = \arg \min_v \{p(v) - (\lambda^k)^T v + \frac{\beta}{2} \|v\|^2\}. \quad (3.1.6)
\]

Using the definition of \( p \), we see that finding \( v^{k+1} \) is equivalent to minimize the argument Lagrangian.

In viewing of (3.1.4), the iteration of the dual algorithm takes the form

\[
\begin{align*}
x^{k+1} &= \arg \min_x \theta(x) + \lambda^T (Ax - b) + \frac{\beta}{2} \|Ax - b\|^2 \\
\lambda^{k+1} &= \lambda^k + \beta(Ax^{k+1} - b).
\end{align*} \quad (3.1.7)
\]

By symmetry, if we add the steps of updating the variable \( y^k \), we see that ADMM has the following iterative scheme:

\[
\begin{align*}
x^{k+1} &= \arg \min_{x} \theta_1(x) - (\lambda^k)^T (Ax + By^k - b) + \frac{\beta}{2} \|Ax + By^k - b\|^2, \\
y^{k+1} &= \arg \min_{y} \theta_2(y) - (\lambda^k)^T (Ax^{k+1} + By - b) + \frac{\beta}{2} \|Ax^{k+1} + By - b\|^2, \\
\lambda^{k+1} &= \lambda^k - \beta(Ax^{k+1} + By^{k+1} - b),
\end{align*} \quad (3.1.8)
\]

where \( \lambda \in \mathbb{R}^m \) is the Lagrange multiplier related to the linear constraint and \( \beta > 0 \) is a penalty parameter. In addition to ADMM, there is another splitting method called
the Peaceman-Rachford Splitting Method (PRSM) which has the following iterative scheme:

\[
\begin{align*}
    x^{k+1} &= \arg\min \{ \theta_1(x) - (\lambda^k)^T(Ax + By^k - b) + \frac{\beta}{2} \| Ax + By^k - b \|^2 \}, \\
    \lambda^{k+\frac{1}{2}} &= \lambda^k - \beta(Ax^{x+1} + By^k - b), \\
    y^{k+1} &= \arg\min \{ \theta_2(y) - (\lambda^{k+\frac{1}{2}})^T(Ax^{k+1} + By - b) + \frac{\beta}{2} \| Ax^{k+1} + By - b \|^2 \}, \\
    \lambda^{k+1} &= \lambda^{k+\frac{1}{2}} - \beta(Ax^{x+1} + By^{k+1} - b).
\end{align*}
\]

(3.1.9)

It is obvious that the only difference between ADMM scheme and PRSM scheme is the addition of the intermediate update of the multiplier \( \lambda \) before updating \( y \). Although the PRSM scheme is less likely to converge unless it has some restrictive assumptions, whenever it converges, it converges faster than the ADMM [24]. Fortunately, it has been shown that with a slight modification in the algorithm scheme, the PRSM establishes a worst-case \( O(1/t) \) convergence rate in both ergodic and nonergodic senses [31]. An underdetermined relaxation factor \( \alpha \in (0, 1) \) could be used to multiply the penalty parameter \( \beta \) in the iterative scheme. The resultant iterative scheme is called the strictly contractive PRSM. Strictly contractive PRSM for (3.1.1) gives the following iterative scheme:

\[
\begin{align*}
    x^{k+1} &= \arg\min \{ \theta_1(x) - (\lambda^k)^T(Ax + By^k - b) + \frac{\beta}{2} \| Ax + By^k - b \|^2 \}, \\
    \lambda^{k+\frac{1}{2}} &= \lambda^k - \alpha \beta(Ax^{x+1} + By^k - b), \\
    y^{k+1} &= \arg\min \{ \theta_2(y) - (\lambda^{k+\frac{1}{2}})^T(Ax^{k+1} + By - b) + \frac{\beta}{2} \| Ax^{k+1} + By - b \|^2 \}, \\
    \lambda^{k+1} &= \lambda^{k+\frac{1}{2}} - \alpha \beta(Ax^{x+1} + By^{k+1} - b),
\end{align*}
\]

(3.1.10)

where \( \alpha \in (0, 1) \) and is called an underdetermined relaxation factor.

In many cases, the exact solution to \( x \) and \( y \) updates in (3.1.8) and (3.1.9) is difficult to obtain. There are several versions of inexact ADMM [30, 53, 46]. As
suggested in literature [30], it will be much more powerful to include a regularized proximal term to both subproblems in (3.1.8), so that the solution will be stable, unique and well-conditioned. We take this idea and present the regularized version of strictly contractive PRSM scheme here.

Let \( L \) be defined as the augmented Lagrangian function of (3.1.1):

\[
L(x, y, \lambda, \beta) = \theta_1(x) + \theta_2(y) - (\lambda^k)^T(Ax + by - b) + \frac{\beta}{2}||Ax + By - b||^2. \quad (3.1.11)
\]

The exact solution to the strictly contractive PRSM scheme with regularized proximal terms can be described as

\[
\begin{align*}
x_{\text{exact}}^{k+1} &= \arg \min \{L(x, y^k, \lambda^k, \beta) + \frac{1}{2\tau}||x - x^k||^2 | x \in \mathcal{X}\}, \\
\lambda^{k+\frac{1}{2}} &= \lambda^k - \alpha \beta (Ax^{k+1} + By^k - b), \\
y_{\text{exact}}^{k+1} &= \arg \min \{L(x_{\text{exact}}^{k+1}, y, \lambda^{k+\frac{1}{2}}, \beta) + \frac{1}{2\tau}||y - y^k||^2 | y \in \mathcal{Y}\}, \\
\lambda^{k+1} &= \lambda^{k+\frac{1}{2}} - \alpha \beta (Ax^{k+1} + By^{k+1} - b),
\end{align*} \quad (3.1.12)
\]

Let \( \tau > 0 \) be the regularization parameter of the proximal terms (3.1.12), and define

\[
f_k(x) := f(x) - A^T[\lambda^k - \beta(Ax + By^k - b)] + \frac{1}{\tau}(x - x^k), \quad (3.1.13)
\]

and

\[
g_k(y) := g(y) - B^T[\lambda^{k+\frac{1}{2}} - \beta(Ax^{k+1} + By - b)] + \frac{1}{\tau}(y - y^k), \quad (3.1.14)
\]

where \( f(x) \in \partial \theta_1(x) \) and \( g(y) \in \partial \theta_2(y) \) are the subgradients of \( \theta_1(x) \) and \( \theta_2(y) \). Let \( (x^0, y^0, \lambda^0) \in \mathcal{X} \times \mathcal{Y} \times \mathbb{R}^m \) be an initial triplet. For a given triplet \( (x^k, y^k, \lambda^k) \in (\mathcal{X} \times \mathcal{Y} \times \mathbb{R}^m) \), the new triplet \( (x^{k+1}, y^{k+1}, \lambda^{k+1}) \) for (3.1.12) is generated by the iterative scheme:
Step 1. Find the unique inexact solution $x^{k+1} \in \mathcal{X}$, such that
\[(x - x^{k+1})^T f_k(x^{k+1}) \geq 0, \forall x \in \mathcal{X}. \tag{3.1.15}\]

Step 2. Update
\[\lambda^{k+\frac{1}{2}} = \lambda^k - \alpha \beta (Ax^{k+1} + By^k - b). \tag{3.1.16}\]

Step 3. Find the unique exact solution $y^{k+1} \in \mathcal{Y}$, such that
\[(y - y^{k+1})^T g_k(y^{k+1}) \geq 0, \forall y \in \mathcal{Y}. \tag{3.1.17}\]

Step 4. Update
\[\lambda^{k+1} = \lambda^{k+\frac{1}{2}} - \alpha \beta (Ax^{k+1} + By^{k+1} - b). \tag{3.1.18}\]

Now we present the inexact version of the iteration scheme for (3.1.15) to (3.1.18).

Let $(x^0, y^0, \lambda^0) \in \mathcal{X} \times \mathcal{Y} \times \mathbb{R}^m$ be an initial triplet. For a given triplet $(x^k, y^k, \lambda^k) \in (\mathcal{X} \times \mathcal{Y} \times \mathbb{R}^m)$, the new triplet $(x^{k+1}, y^{k+1}, \lambda^{k+1})$ for (3.1.12) in generated inexactly by the following iterative scheme:

Step 1. Find $x^{k+1} \in \mathcal{X}$ and $\xi^k_x \in \mathbb{R}^n_1$, such that
\[(x - x^{k+1})^T (f_k(x^{k+1}) - \frac{1}{\tau} \xi^k_x) \geq 0, \forall x \in \mathcal{X}. \tag{3.1.19}\]

Step 2. Update
\[\lambda^{k+\frac{1}{2}} = \lambda^k - \alpha \beta (Ax^{k+1} + By^k - b). \tag{3.1.20}\]

Step 3. Find $y^{k+1} \in \mathcal{Y}$ and $\xi^k_y \in \mathbb{R}^n_2$, such that
\[(y - y^{k+1})^T (g_k(y^{k+1}) - \frac{1}{\tau} \xi^k_y) \geq 0, \forall y \in \mathcal{Y}. \tag{3.1.21}\]
Step 4. Update

\[
\lambda^{k+1} = \lambda^{k+\frac{1}{2}} - \alpha \beta (Ax^{k+1} + By^{k+1} - b).
\]  

(3.1.22)

The sequences \(\{\xi^k_x\}\) and \(\{\xi^k_y\}\) are called the error terms and should satisfy the following generic inexactness criterion:

Criterion 1: \(\|\xi^k_x\| \leq \nu_k, \|\xi^k_y\| \leq \nu_k, \text{ and } \sum_{k=0}^{\infty} \nu_k < \infty,\)  

(3.1.23)

or

Criterion 2: \(\|\xi^k_x\| \leq \nu_k \|x^k - x^{k+1}\|, \|\xi^k_y\| \leq \nu_k \|y^k - y^{k+1}\|, \text{ and } \sum_{k=0}^{\infty} \nu_k^2 < \infty.\)  

(3.1.24)

We notice that (3.1.23) and (3.1.24) can be easily satisfied when doing numerical experiment. In the following analysis, (3.1.23) and (3.1.24) are denoted as IPRSM1 and IPRSM2, respectively. Unlike many other inexact methods which usually require comparing the difference between the inexact solution with the exact solution, IPRSM1 and IPRSM2 are implementable in a way that only the difference between each iteration needs to be calculated. This advantage allows us to track the inexactness criteria quite easily. Both absolute error criterion and relative error criterion will be provided in the following analysis.

As suggested in [53], for the \(x\)-update, the error term can be chosen in several steps:

Step i. Using a certain iterative method for strongly monotone VI to calculate \(x^{k+1}_{\text{exact}}\) in (3.1.15).
Step ii. Once an iterate $x_j^{k+1}$ is generated, we find

$$\tilde{x}_j^{k+1} = P_{\mathcal{X}}[x_j^{k+1} - f_k(x_j^{k+1})],$$

(3.1.25)

where $P_{\mathcal{X}}$ is a projection operator mapping to $\mathcal{X}$.

Step iii. Choose

$$\xi_k = f_k(\tilde{x}_j^{k+1}) - f_k(x_j^{k+1}) + x_j^{k+1} - \tilde{x}_j^{k+1},$$

(3.1.26)

and check if (3.1.23) or (3.1.24) is satisfied. If yes, go to Step 2 of the main scheme; if not, go back to Step i and do one more iteration. The $y$-update has the similar steps, so we omit the details here.

We give some useful preliminaries in Section 3.2. In Section 3.3 we give the convergence analysis proof. In section 3.4 and 3.5 we establish a worst-case $O(1/t)$ convergence rate in ergodic sense and in non-ergodic senses for both IPRSM1 and IPRSM2, respectively. Numerical efficiency for the inexact PRSM will be provided in Section 3.6.

### 3.2 Preliminaries

In this section, we give some useful preliminaries and prove some useful lemmas for the convergence analysis.

#### 3.2.1 Variational Reformulation

We need a variational inequality (VI) reformulation of (3.1.1). Given a subset $\Omega$ of the Euclidean $d$-dimensional space $\mathbb{R}^d$ and a mapping $H$ from $\Omega$ to $\mathbb{R}^d$, the variational
inequality (VI) is to find \( u^* \in \Omega \) such that

\[
(u - u^*)^T H(u^*) \geq 0, \quad \forall u \in \Omega. \tag{3.2.1}
\]

In our case, let \( w^* = (x^*, y^*, \lambda^*) \in \Omega := \mathcal{X} \times \mathcal{Y} \times \mathbb{R}^m \). The solution to (3.1.1) is characterized by the following variational inequality:

**Lemma 3.2.1.** Solving (3.1.1) is equivalent to find \( w^* \) such that

\[
(w - w^*)^T Q(w^*) \leq 0, \quad \forall w \in \Omega, \tag{3.2.2}
\]

where

\[
Q(w^*) = \begin{pmatrix}
f(x^*) - A^T \lambda \\
g(y^*) - B^T \lambda \\
Ax^* + By^* - b
\end{pmatrix}
\tag{3.2.3}
\]

and \( f(x) \in \partial \theta_1(x) \) and \( g(y) \in \partial \theta_2(y) \).

**Proof.** This is a direct application of VI to the unconstrained optimization problem. The optimization problem is obtained by attaching a Lagrange multiplier vector \( \lambda \) to the linear constraints in (3.1.1).

In the next lemma, we elaborate the VI used in our analysis as MVI(\( \Omega, F, \theta \)) and give the definition. We denote the solution set of MVI as \( \Omega^* \) throughout this paper. By assumption, \( \Omega^* \) is non-empty under our assumption.

**Lemma 3.2.2.** Denote \( \Omega^* \) by the solution set of MVI(\( \Omega, F, \theta \)) and assume it is non-empty. If \( w^* = (x^*, y^*, \lambda^*) \in \Omega^* \), then solving (3.1.1) is equivalent to find \( w^* \) such that

\[
\text{MVI}(\Omega, F, \theta) : \theta(u) - \theta(u^*) + (w - w^*)^T \leq 0, \quad \forall w \in \Omega, \tag{3.2.4}
\]

\[54\]
where
\[
\begin{align*}
  u &= \begin{pmatrix} x \\ y \end{pmatrix}, \\
  w &= \begin{pmatrix} x \\ y \\ \lambda \end{pmatrix}, \\
  F(w) &= \begin{pmatrix} -A^T \lambda \\ -B^T \lambda \\ Ax + By - b \end{pmatrix}
\end{align*}
\] (3.2.5)

and
\[
\theta(u) = \theta_1(x) + \theta_2(y). 
\] (3.2.6)

**Proof.** This follows directly from the previous lemma and the convexity of \( \theta \). \( \square \)

Note that the mapping \( F(w) \) defined above is affine with a skew-symmetric matrix so that it is monotone.

**Lemma 3.2.3.** The mapping \( P_\Omega : \mathbb{R}^d \rightarrow \Omega \) is called the projection operator and a basic inequality of this projection operator is
\[
\|P_\Omega(v) - P_\Omega(u)\| \leq \|v - u\|, \quad \forall v, u \in \mathbb{R}^d. 
\] (3.2.7)

**Proof.** The proof of this inequality can be found in [20]. \( \square \)

**Lemma 3.2.4.** Let \( \gamma \) be any positive constant and \( \text{VI}(\Omega, H) \) be defined in [3.2.1], then solving \( \text{VI}(\Omega, H) \) is equivalent to finding a zero point of
\[
e(u, \Omega, \gamma H) := u - P_\Omega[u^* - \gamma H(u^*)]. 
\] (3.2.8)

**Proof.** See [20]. \( \square \)

In our analysis, solving \( \text{MVI}(\Omega, F, \theta) \) is equivalent to finding a zero point of
\[
e(w, \Omega, Q) = w - P_\Omega[w - Q(w)] \begin{pmatrix} x - P_X[x - f(x) + A^T\lambda] \\ y - P_Y[y - g(y) + B^T\lambda] \\ Ax + By - b \end{pmatrix}. 
\] (3.2.9)

55
3.2.2 Matrix Notations

Now we define some matrices which will be used in the proof

\[
Q := \begin{pmatrix}
\frac{1}{\tau} I_{n_1} & 0 & 0 \\
0 & \beta B^T B + \frac{1}{\tau} I_{n_2} & -\alpha B^T \\
0 & -B & \frac{1}{\beta} I_m
\end{pmatrix},
\]  
(3.2.10)

\[
M := \begin{pmatrix}
I_{n_1} & 0 & 0 \\
0 & I_{n_2} & 0 \\
0 & -\alpha \beta B & 2\alpha I_m
\end{pmatrix},
\]  
(3.2.11)

\[
H := \frac{1}{2} \begin{pmatrix}
\frac{2}{\tau} I_{n_1} & 0 & 0 \\
0 & (2 - \alpha)\beta B^T B + \frac{2}{\tau} I_{n_2} & -B^T \\
0 & -B & \frac{1}{\alpha \beta} I_m
\end{pmatrix}.
\]  
(3.2.12)

In these matrices, “0” denotes a matrix with appropriate dimensionality having all zero entries. \(\alpha \in (0, 1]\). The next lemma is to simplify the presentation of the proof of convergence analysis.

**Lemma 3.2.5.** Let the matrices \(Q\), \(M\) and \(H\) be defined in (3.2.10), (3.2.11) and (3.2.12), respectively, then we have the following relationships:

\[
HM = Q,
\]  
(3.2.13)

\[
Q^T + Q - M^T HM \succeq \frac{1 - \alpha}{2(1 + \alpha)} M^T HM.
\]  
(3.2.14)

If \(\alpha \in (0, 1)\), \(H\) is positive definite. If \(\alpha = 1\), \(H\) is positive semi-definite.
Proof.

\[ HM = \frac{1}{2} \begin{pmatrix} \frac{2}{\tau} I_{n_1} & 0 & 0 \\ 0 & (2 - \alpha)\beta B^T B + \frac{2}{\tau} I_{n_2} & -B^T \\ 0 & -B & \frac{1}{\alpha \beta} I_m \end{pmatrix} \begin{pmatrix} I_{n_1} & 0 & 0 \\ 0 & I_{n_2} & 0 \\ 0 & -\alpha \beta B & 2\alpha I_m \end{pmatrix} \]

\[ = \frac{1}{2} \begin{pmatrix} \frac{2}{\tau} I_{n_1} & 0 & 0 \\ 0 & 2\beta B^T B + \frac{2}{\tau} I_{n_2} & -2\alpha B^T \\ 0 & -2B & \frac{2}{\beta} I_m \end{pmatrix} = Q. \]

Note that

\[ H \succeq \frac{1}{2} \begin{pmatrix} \frac{2}{\tau} I_{n_1} & 0 & 0 \\ 0 & (2 - \alpha)\beta B^T B & -B^T \\ 0 & -B & \frac{1}{\alpha \beta} I_m \end{pmatrix} \]

\[ = \frac{1}{2} \begin{pmatrix} I_{n_1} & 0 & 0 \\ 0 & \sqrt{\beta} B^T & 0 \\ 0 & 0 & \sqrt{\frac{1}{\beta} I_m} \end{pmatrix} \begin{pmatrix} \frac{2}{\tau} I_{n_1} & 0 & 0 \\ 0 & (2 - \alpha) I_m & -I_m \\ 0 & -I_m & \frac{1}{\alpha I_m} \end{pmatrix} \begin{pmatrix} I_{n_1} & 0 & 0 \\ 0 & \sqrt{\beta} B & 0 \\ 0 & 0 & \sqrt{\frac{1}{\beta} I_m} \end{pmatrix} \]

and if \( \alpha \in (0, 1) \), \( H \) is positive definite. If \( \alpha = 1 \), \( H \) is positive semi-definite. To prove (3.2.14), first we have

\[ M^T HM = M^T Q \]

\[ = \begin{pmatrix} I_{n_1} & 0 & 0 \\ 0 & I_{n_2} & -\alpha \beta B^T \\ 0 & 0 & 2\alpha I_m \end{pmatrix} \begin{pmatrix} \frac{1}{\tau} I_{n_1} & 0 & 0 \\ 0 & \beta B^T B + \frac{1}{\tau} I_{n_2} & -\alpha B^T \\ 0 & -B & \frac{1}{\beta} I_m \end{pmatrix} \]

\[ = \begin{pmatrix} \frac{1}{\tau} I_{n_1} & 0 & 0 \\ 0 & (1 + \alpha)\beta B^T B + \frac{1}{\tau} I_{n_2} & -2\alpha B^T \\ 0 & -2\alpha B & \frac{2\alpha}{\beta} I_m \end{pmatrix}, \]

57
then we have

\[ Q + Q^T - \frac{3 + \alpha}{2(1 + \alpha)} M^T H M = \begin{pmatrix}
\frac{3\alpha + 1}{2\tau(1 + \alpha)} I_{n_1} & 0 & 0 \\
0 & \frac{1 - \alpha}{2} \beta B^T B + \frac{3\alpha + 1}{2\tau(1 + \alpha)} I_{n_2} & -\frac{1 - \alpha}{1 + \alpha} B^T \\
0 & -\frac{1 - \alpha}{1 + \alpha} B & \frac{(\alpha + 2)(1 - \alpha)}{\beta(1 + \alpha)} I_m
\end{pmatrix} \]

\[ \succeq \frac{1 - \alpha}{2(1 + \alpha)} \begin{pmatrix}
\frac{3\alpha + 1}{2\tau(1 - \alpha)} I_{n_1} & 0 & 0 \\
0 & 0 & (1 + \alpha) \beta B^T B - 2B^T \\
0 & 0 & -2B - \frac{4 + 2\alpha}{\beta} I_m
\end{pmatrix} \]

\[ \succeq 0. \]

The last relation holds because

\[ \begin{pmatrix} 1 + \alpha & 2 \\
-2 & 4 + 2\alpha \end{pmatrix} \succeq 0 \quad \begin{pmatrix} \beta B^T B & -B^T \\
-B & \frac{1}{\beta} \end{pmatrix} \succeq 0, \]

and this proves (3.2.14).

3.3 Convergence Analysis

In this section, we start with a few lemmas and then we prove the convergence of the sequences generated by the IPRSM1 and IPRSM2 at the end.

To alleviate the notation, we define an auxiliary sequence \( \{\tilde{w}^k\} \) as

\[ \tilde{w}^k := \begin{pmatrix} \tilde{x}^k \\
\tilde{y}^k \\
\tilde{\lambda}^k \end{pmatrix} = \begin{pmatrix} x^{k+1} \\
y^{k+1} \\
\lambda^k - \beta(Ax^{k+1} + By^k - b) \end{pmatrix}, \quad (3.3.1) \]

where \((x^k, y^k, \lambda^k)\) is generated by either IPRSM1 or IPRSM2. Note according to (3.3.1), we immediately have \( \tilde{x}^k = x^{k+1}, \tilde{y}^k = y^{k+1} \) and \( \tilde{\lambda}^k = \lambda^k - \beta(Ax^{k+1} + By^k - b) \).
The first lemma in this section establishes the relation between \(w^k - w^{k+1}\) and \(w^k - \tilde{w}^k\).

**Lemma 3.3.1.** Using the above notation, and letting \(w^k = (x^k, y^k, \lambda^k)\) be the sequence generated by IPRSM1 or IPRSM2, we have

\[
w^{k+1} = w^k - M(w^k - \tilde{w}^k).
\]

**(3.3.2)**

**Proof.** By (3.1.20) and (3.3.1), we have

\[
\lambda^{k + \frac{1}{2}} = \lambda^k - \alpha(\lambda^k - \bar{\lambda}^k).
\]

**(3.3.3)**

By (3.1.22), (3.3.1) and (3.3.3), we get

\[
\lambda^{k+1} = \lambda^{k + \frac{1}{2}} - \alpha \beta (Ax^{k+1} + By^{k+1} - b)
\]

\[
= \lambda^k - \alpha(\lambda^k - \bar{\lambda}^k) - \alpha[(\lambda^k - \bar{\lambda}^k) + \beta B(y^{k+1} - \bar{y}^k)]
\]

\[
= \lambda^k - 2\alpha(\lambda^k - \bar{\lambda}^k) + \alpha\beta B(y^k - \bar{y}^k).
\]

**(3.3.4)**

The proof of the lemma follows directly from the definition of \(M\) in (3.2.11).

\[\square\]

We start to prove some properties for the sequence defined in (3.3.1) and use these properties to prove the main theorem presented at the end of this section. Remember, our purpose is to prove the convergence and analyze the convergence rate based on (3.2.4). We need to estimate how accurate the point \(\tilde{w}^k\) is to a solution point of \(\text{MVI}(\Omega, F, \theta)\); therefore, we need an upper bound of the quantity of \(\theta(\tilde{w}^k) - \theta(u) + (\tilde{w}^k - w)F(\tilde{w}^k)\). The first lemma is used to give an upper bound of \(\theta(\tilde{w}^k) - \theta(u) + (\tilde{w}^k - w)F(\tilde{w}^k)\), for all \(w \in \Omega\), in terms of a quadratic term involving the matrix \(Q\).
Lemma 3.3.2. Let \( \{w^k\} \) be generated by the IPRSM1 or IPRSM2 scheme and \( \{\tilde{w}^k\} \) be defined in (3.3.1), then we have

\[
\theta(u) - \theta(\tilde{u}^k) + (w - \tilde{w}^k)^T F(\tilde{w}^k) \geq (w - \tilde{w}^k)^T Q(w^k - \tilde{w}^k) - \frac{1}{\tau} (w - \tilde{w}^k)^T \xi^k, \quad \forall w \in \Omega,
\]

(3.3.5)

where the matrix \( Q \) is defined in (3.2.10) and \( \xi^k = ((\xi^k_x)^T, (\xi^k_y)^T, 0) \).

Proof. Since \( x^{k+1} = \tilde{x}^k \), from (3.1.13) and the convexity of \( \theta_1(x) \), we have

\[
\theta_1(x) - \theta_1(\tilde{x}^k) + (x - \tilde{x}^k)^T \{ A^T [\beta(A\tilde{x}^k + By^k - b) - \lambda^k] - \frac{1}{\tau} (x^k - \tilde{x}^k) - \frac{1}{\tau} \xi^k_x \} \\
\geq 0, \quad \forall x \in \mathcal{X}.
\]

(3.3.6)

From (3.3.1), we get

\[
\theta_1(x) - \theta_1(\tilde{x}^k) + (x - \tilde{x}^k)^T [ -A^T \tilde{\lambda}^k - \frac{1}{\tau} (x^k - \tilde{x}^k) ] \geq \frac{1}{\tau} (x - \tilde{x}^k)^T \xi^k_x, \quad \forall x \in \mathcal{X}.
\]

(3.3.7)

Similarly, from (3.1.19) and the convexity of \( \theta_2(y) \), we have

\[
\theta_2(y) - \theta_2(\tilde{y}^k) + (y - \tilde{y}^k)^T B^T \{ \beta(B\tilde{x}^k + By^k - b) - [\lambda^k - \alpha(\lambda^k - \tilde{\lambda}^k)] \} \\
+ \frac{1}{\tau} (y - \tilde{y}^k)^T [ - (y^k - \tilde{y}^k) - \xi^k_y ] \geq 0, \quad \forall y \in \mathcal{Y}.
\]

(3.3.8)

Applying (3.3.1) again, we get

\[
\theta_2(y) - \theta_2(\tilde{y}^k) + (y - \tilde{y}^k)^T B^T [ - \tilde{\lambda}^k + \beta B(\tilde{y}^k - y^k) + \alpha(\lambda^k - \tilde{\lambda}^k) ] \\
+ \frac{1}{\tau} (y - \tilde{y}^k)^T [ - (y^k - \tilde{y}^k) ] \geq \frac{1}{\tau} (y - \tilde{y}^k)^T \xi^k_y, \quad \forall y \in \mathcal{Y}.
\]

(3.3.9)

From (3.3.1), we also have

\[
(A\tilde{x}^k + By^k - b) - B(\tilde{y}^k - y^k) + \frac{1}{\beta}(\lambda^k - \tilde{\lambda}^k) = 0.
\]

(3.3.10)
Combining (3.3.7), (3.3.9) and (3.3.10), we get

$$\theta(u) - \theta(\tilde{u}^k) + \left( x - \tilde{x}^k \right)^T \left( \begin{array}{c} -A^T \tilde{\lambda}^k \\ -B^T \tilde{\lambda}^k \\ A\tilde{x}^k + B\tilde{y}^k - b \end{array} \right) +$$

$$\left( x - \tilde{x}^k \right)^T \left( \begin{array}{c} \frac{1}{\tau}(\tilde{x}^k - x^k) \\ \frac{1}{\tau}(\tilde{y}^k - y^k) - \alpha B^T(\tilde{\lambda}^k - \lambda^k) \\ -B(\tilde{y}^k - y^k) + \frac{1}{\beta}(\tilde{\lambda}^k - \lambda^k) \end{array} \right) \geq \frac{1}{\tau}(w - \tilde{w}^k)^T \xi^k, \quad \forall w \in \Omega.$$  

(3.3.11)

Now (3.3.5) follows directly from (3.3.12), the definition of $F(w)$ and the definition of $Q$.

In the next lemma, we further bound the quadratic term $(w - \tilde{w}^k)^T Q(w^k - \tilde{w}^k)$ from the previous lemma by the difference of two quadratic terms involving consecutive iterates of $\{w^k\}$. This lemma is very important since we need them to establish the convergence rate in both ergodic and nonergodic senses for IPRSM1 or IPRSM2 later.

**Lemma 3.3.3.** Let $\{w^k\}$ be generated by the IPRSM1 or IPRSM2 scheme and $\{\tilde{w}^k\}$ be defined in (3.3.1); $H$ and $Q$ be defined in (3.2.12) and (3.2.10), respectively, then

$$\theta(u) - \theta(\tilde{u}^k) + (w - \tilde{w}^k) F(\tilde{w}^k) \geq \frac{1}{2}(\|w - w^{k+1}\|_H^2 - \|w - w^k\|_H^2)$$

$$+ \frac{1 - \alpha}{4(1 + \alpha)} \|w^k - w^{k+1}\|_H^2 - \frac{1}{\tau}(w - \tilde{w}^k)\xi^k, \quad \forall w \in \Omega.$$  

(3.3.12)

**Proof.** We only need to show that

$$(w - \tilde{w}^k)^T Q(w^k - \tilde{w}^k) \geq$$

$$\frac{1}{2}(\|w - w^{k+1}\|_H^2 - \|w - w^k\|_H^2) + \frac{1 - \alpha}{4(1 + \alpha)} \|w^k - w^{k+1}\|_H^2, \quad \forall w \in \Omega.$$  

(3.3.13)
By using \( Q = HM \) and (3.3.2), we have

\[(w - \tilde{w}^k)^T H (w^k - w^{k+1}) = (w - \tilde{w}^k)^T Q (w^k - \tilde{w}^k)^T. \tag{3.3.14}\]

The following identity holds for the vectors \( a, b, c \) and \( d \) in the same space and a matrix \( H \) with appropriate dimensionality

\[(a - b)^T H (c - d) = \frac{1}{2} \{||a - d||_H^2 - ||a - c||_H^2\} + \frac{1}{2} \{||c - b||_H^2 - ||d - b||_H^2\}. \tag{3.3.15}\]

Taking \( a = w, \ b = \tilde{w}^k, \ c = w^k, \ d = w^{k+1} \), and combining it with (3.3.14), we get

\[(w - \tilde{w}^k)^T Q (w^k - \tilde{w}^k)^T = \frac{1}{2} \{||w - w^{k+1}||_H^2 - ||w - w^k||_H^2\} + \frac{1}{2} \{||w^k - w^k||_H^2 - ||w^{k+1} - w^k||_H^2\}. \tag{3.3.16}\]

For the second component on the right side of (3.3.16), we found the lower bound by applying Lemma 3.2.5 and Lemma 3.3.1

\[\|w^k - \tilde{w}^k\|_H^2 - \|w^{k+1} - \tilde{w}^k\|_H^2\]

\[= \|w^k - \tilde{w}^k\|_H^2 - \|(w^k - \tilde{w}^k) - (w^{k+1} - \tilde{w}^k)\|_H^2\]

\[= \|w^k - \tilde{w}^k\|_H^2 - \|(w^k - \tilde{w}^k) - M(w^k - \tilde{w}^k)\|_H^2\]

\[= 2 (w^k - \tilde{w}^k)^T (Q^T + Q - M^T H M) (w^k - \tilde{w}^k) \]

\[\geq \frac{1 - \alpha}{2(1 + \alpha)} \|w^k - w^{k+1}\|_H^2. \tag{3.3.17}\]

Combining (3.3.16) and (3.3.17), we finish the proof of this lemma. \( \square \)
The next two lemmas are used to bound the term \( \|w^{k+1} - w^*\|_H^2 \) by \( \|w^k - w^*\|_H^2 \), for IPRSM1 and IPRSM2, respectively, so that the recursion in the main theorem can be applied.

**Lemma 3.3.4.** Let \( w^* = (x^*, y^*, \lambda^*) \in W^* \) be any solution point of MVI(\( \Omega, F, \theta \)). Let \( (x^k, y^k, \lambda^k) \) be the sequence generated by IPRSM1. Let \( 0 < \alpha < 1 \), then we have

\[
(1 - \nu_k)\|w^{k+1} - w^*\|_H^2 \leq \|w^k - w^*\|_H^2 - c_1\|w^k - w^{k+1}\|_H^2 + c_2\nu_k, \quad (3.3.18)
\]

where

\[
c_1 = \frac{1 - \alpha}{2(1 + \alpha)} \geq 0, \quad c_2 = \frac{1}{\sigma\tau^2} > 0, \quad (3.3.19)
\]

and \( w^* \) is an arbitrary solution point. \( \sigma \), which is the smallest eigenvalue of \( H \), is only related to \( \alpha, \beta, \) and \( \tau \).

**Proof.** (3.3.12) says that

\[
\theta(u) - \theta(\tilde{w}^k) + (w - \tilde{w}^k)^TF(\tilde{w}^k) \geq \frac{1}{2}(\|w - w^{k+1}\|_H^2 - \|w - w^k\|_H^2) + \frac{1 - \alpha}{4(1 + \alpha)}\|w^k - w^{k+1}\|_H^2 - \frac{1}{\tau}(w - \tilde{w}^k)^T\xi_k, \quad \forall w \in \Omega. \quad (3.3.20)
\]

The monotonicity of \( F(w) \) says that

\[
(w - \tilde{w}^k)^TF(w) - (w - \tilde{w}^k)^TF(\tilde{w}^k) \geq 0, \quad \forall w \in \Omega. \quad (3.3.21)
\]

Therefore, by (3.3.20), we have

\[
\theta(u) - \theta(\tilde{w}^k) + (w - \tilde{w}^k)^TF(w) \geq \frac{1}{2}(\|w - w^{k+1}\|_H^2 - \|w - w^k\|_H^2) + \frac{1 - \alpha}{4(1 + \alpha)}\|w^k - w^{k+1}\|_H^2 - \frac{1}{\tau}(w - \tilde{w}^k)^T\xi_k, \quad \forall w \in \Omega. \quad (3.3.22)
\]
Let $w = w^*$ in (3.3.22) and apply Lemma 3.2.2, we get

$$0 \geq \|w^* - w^{k+1}\|_H^2 - \|w^* - w^k\|_H^2 + \frac{1 - \alpha}{2(1 + \alpha)}\|w^k - w^{k+1}\|_H^2 - \frac{2}{\tau}(w^* - \bar{w}^k)^T \xi^k.$$  (3.3.23)

Note that $\xi^k$ has the form $((\xi_x^K)^T, (\xi_y^K)^T, 0)^T$ and $\sigma$ is the smallest eigenvalue of $H$, then by criterion 1, we know that

$$2(w^* - \bar{w}^k)^T \xi^k = 2(w^* - w^{k+1})^T \xi^k$$

$$\leq 2\|w^* - w^{k+1}\| \cdot \|\xi^k\|$$

$$\leq p\nu_k\|w^* - w^{k+1}\|_H^2 + \frac{1}{p\nu_k}\|\xi^k\|^2$$  (3.3.24)

$$\leq \frac{p\nu_k}{\sigma}\|w^* - w^{k+1}\|_H^2 + \frac{1}{p\nu_k}\|\xi^k\|^2$$

$$\leq \frac{p\nu_k}{\sigma}\|w^* - w^{k+1}\|_H^2 + \frac{1}{p\nu_k},$$

where $p$ is some constant to be determined. Setting $p = \tau\sigma$, we get

$$\|w^* - w^k\|_H^2 - (1 - \nu_k)\|w^* - w^{k+1}\|_H^2 \geq \frac{1 - \alpha}{2(1 + \alpha)}\|w^k - w^{k+1}\|_H^2 - \frac{1}{\tau^2\sigma}\nu_k.$$  (3.3.25)

\[\square\]

**Lemma 3.3.5.** Let $w^* = (x^*, y^*, \lambda^*) \in \mathcal{W}^*$ be any solution point of MVI($\Omega, F, \theta$). Let $(x^k, y^k, \lambda^k)$ be the sequence generated by IPRSM2. Let $0 < \alpha < 1$, then we have

$$(1 - \nu_k^2)\|w^{k+1} - w^k\|_H^2 \leq \|w^k - w^*\|_H^2 - c_3\|w^{k+1} - w^k\|_H^2,$$  (3.3.26)

where

$$c_3 = \frac{1 - \alpha}{2(1 + \alpha)} - \frac{1}{\sigma^2\tau^2},$$

and $w^*$ is an arbitrary solution point. $\sigma$, which is the smallest eigenvalue of $H$, is only related to $\alpha, \beta,$ and $\tau$. 

64
Proof. Note that $\sigma$ is the smallest eigenvalue of $H$, then by criterion 2, we have

$$2(w^* - \tilde{w}^k)^T \xi^k = 2(w^* - w^{k+1})^T \xi^k$$

\begin{align}
&\leq 2\|w^* - w^{k+1}\| \cdot \|\xi^k\| \\
&\leq \nu_k^2 \tau \sigma \|w^* - w^{k+1}\|^2 + \frac{1}{\tau \sigma} \|w^k - w^{k+1}\|^2 \\
&\leq \nu_k^2 \tau \|w^* - w^{k+1}\|^2 + \frac{1}{\sigma^2 \tau} \|w^k - w^{k+1}\|^2_H.
\end{align}

(3.3.27)

The proof for (3.3.23) still holds in this case since we have not used criterion 1 until (3.3.24). Substitute (3.3.27) into (3.3.23), we obtain

$$(1 - \nu_k)\|w^{k+1} - w^*\|^2_H \leq \|w^k - w^*\|^2_H - c_3\|w^{k+1} - w^k\|^2_H,$$

(3.3.28)

where

$$c_3 = \frac{1 - \alpha}{2(1 + \alpha)} - \frac{1}{\sigma^2 \tau^2}.$$ 

For $0 < \alpha < 1$, we can choose $\tau$ such that $c_3 > 0$.

The next lemma is used to upper bound $\|e(w^{k+1}, \Omega, Q)\|$ so that we can use (3.2.4) to prove the main theorem in this section.

Lemma 3.3.6. Let $w^* = (x^*, y^*, \lambda^*) \in \mathcal{W}^*$ be any solution point of MVI($\Omega, F, \theta$). Let $(x^k, y^k, \lambda^k) \in \Omega$ be the sequence generated by IPRSM1 or IPRSM2, then we have

$$\|e(w^{k+1}, \Omega, Q)\| \leq \left\| \frac{1}{\tau}(x^k - x^{k+1}) - \frac{\beta}{2} A^T B(y^k - y^{k+1}) + \frac{2}{2\alpha} A^T (\lambda^k - \lambda^{k+1}) + \frac{1}{\tau} \xi^k \right\|

\begin{align}
&= \left( \frac{(1-\alpha)\beta}{2} B^T B + \frac{1}{\tau} I \right)(y^k - y^{k+1}) + \frac{\alpha - 1}{2\alpha} B^T (\lambda^k - \lambda^{k+1}) + \frac{1}{\tau} \xi^k \\
&- \frac{1}{2} B(y^k - y^{k+1}) + \frac{1}{2\alpha \beta}(\lambda^k - \lambda^{k+1})
\end{align}

(3.3.29)
Proof. First recall the definition of \( f_k(x) \) and \( g_k(y) \)

\[
f_k(x) = f(x) - A^T[\lambda^k - \beta(Ax + By^k - b)] + \frac{1}{\tau}(x - x^k), \tag{3.3.30}
\]

and

\[
g_k(y) = g(y) - B^T[\lambda^{k+1/2} - \beta(Ax^{k+1} + By - b)] + \frac{1}{\tau}(y - y^k) \tag{3.3.31}
\]

\[
= g(y) - B^T[\lambda^k - \alpha(\lambda^k - \bar{\lambda}) + \beta(Ax^k + 1 + By - b) + \frac{1}{\tau}(y - y^k)].
\]

It follows from (3.2.9) that

\[
x^{k+1} = P_X[x^{k+1} - f(x^{k+1} + 1\tau\xi)] \tag{3.3.32}
\]

and

\[
y^{k+1} = P_Y[y^{k+1} - g(y^{k+1} + 1\tau\xi)]. \tag{3.3.33}
\]

Note that

\[
\|e(w^{k+1}, \Omega, Q)\| = \left\| \begin{array}{c} x^{k+1} - P_X[x^{k+1} - f(x^{k+1}) + A^T\lambda^{k+1}] \\ y^{k+1} - P_Y[y^{k+1} - g(y^{k+1}) + B^T\lambda^{k+1}] \\ Ax^{k+1} + By^{k+1} - b \end{array} \right\|. \tag{3.3.34}
\]
If follows from (3.3.34) and (3.2.7) that,

\[ \|e(w^{k+1}, \Omega, Q)\| = \left\| P_x[x^{k+1} - f_k(x^{k+1}) + \frac{1}{\tau} \xi^k] - P_x[x^{k+1} - f(x^{k+1}) + A^T \lambda^{k+1}] \right\| \\
= \left\| P_y[y^{k+1} - g_k(y^{k+1}) + \frac{1}{\tau} \xi^k] - P_y[y^{k+1} - g(y^{k+1}) + B^T \lambda^{k+1}] \right\| \\
= \left\| A x^{k+1} + B y^{k+1} - b \right\| \\
\leq \left\| f(x^{k+1}) - f_k(x^{k+1}) - A^T \lambda^{k+1} + \frac{1}{\tau} \xi^k \right\| \\
\leq \left\| g(y^{k+1}) - g_k(y^{k+1}) - B^T \lambda^{k+1} + \frac{1}{\tau} \xi^k \right\|.
\]

Now we analyze each of the three terms separately.

\[ \|f(x^{k+1}) - f_k(x^{k+1}) - A^T \lambda^{k+1} + \frac{1}{\tau} \xi^k\| \]
\[ = |\| A^T \lambda^k - A^T \beta(A x^{k+1} + B y - b) - \frac{1}{\tau} (x^{k+1} - x^k) - A^T \lambda^{k+1} + \frac{1}{\tau} \xi^k\| | \\
= |\| A^T (\lambda^k - \lambda^{k+1}) - A^T (\lambda^k - \tilde{\lambda}^k) + \frac{1}{\tau} (x^k - x^{k+1}) + \frac{1}{\tau} \xi^k\| | \\
= |\| \frac{1}{\tau} (x^k - x^{k+1}) - \frac{\beta}{2} A^T B (y^k - y^{k+1}) + \frac{2\alpha - 1}{2\alpha} A^T (\lambda^k - \lambda^{k+1}) + \frac{1}{\tau} \xi^k\| |. \]

where in the last equality, (3.3.4) has been used. Similarly,

\[ \|g(y^{k+1}) - g_k(y^{k+1}) - B^T \lambda^{k+1} + \frac{1}{\tau} \xi^k\| \]
\[ = |\| B^T (\lambda^{k+\frac{1}{2}} - \lambda^{k+1}) - B^T \beta(A x^{k+1} + B y^{k+1} - b) + \frac{1}{\tau} (y^k - y^{k+1}) + \frac{1}{\tau} \xi^k\| | \\
= |\| B^T [\alpha \beta(A x^{k+1} + B y^{k+1} - b) - \beta(A x^{k+1} + B y^{k+1} - b)] + \frac{1}{\tau} (y^k - y^{k+1}) + \frac{1}{\tau} \xi^k\| | \\
= |\| B^T (\alpha - 1) [(\lambda^k - \tilde{\lambda}^k) - \beta B(y^k - y^{k+1})] + \frac{1}{\tau} (y^k - y^{k+1}) + \frac{1}{\tau} \xi^k\| | \\
= |\| [\frac{1}{\tau} - \frac{\beta}{2} (\alpha - 1) B^T B](y^k - y^{k+1}) + \frac{\alpha - 1}{2\alpha} B^T (\lambda^k - \lambda^{k+1}) + \frac{1}{\tau} \xi^k\| |. \]

(3.3.35)

(3.3.36)

(3.3.37)
We also have
\[ \|Ax^{k+1} + By^{k+1} - b\| = \frac{1}{\beta}(\lambda^k - \tilde{\lambda}^k - \beta B(y^k - y^{k+1})) \]
\[ = \frac{1}{\beta^2} 2\alpha (\lambda^k - \lambda^{k+1}) + \frac{\beta}{2} B(y^k - y^{k+1}) - \beta B(y^k - y^{k+1}) \]
\[ = -\frac{1}{2} B(y^k - y^{k+1}) + \frac{1}{2\alpha\beta}(\lambda^k - \lambda^{k+1}). \tag{3.3.38} \]
Combining (3.3.36), (3.3.37) and (3.3.38), we finish the proof of this lemma. \[ \square \]

The next two lemmas are used to further bound the \( e(w^{k+1}, \Omega, Q) \) term by \( \|w^{k+1} - w^k\| \) for IPRSM1 and IPRSM2, respectively.

**Lemma 3.3.7.** Let \( w^* = (x^*, y^*, \lambda^*) \in W^* \) be any solution point of MVI(\( \Omega, F, \theta \)). Let \( (x^k, y^k, \lambda^k) \) be the sequence generated by IPRSM1, then there exists constant \( a_1 > 0 \), such that \( \forall k \geq 0 \), we have
\[ \|e(w^{k+1}, \Omega, Q)\| \leq a_1\|w^{k+1} - w^k\| + \frac{2}{\tau}v_k. \tag{3.3.39} \]

**Proof.** From the previous lemma and (3.1.23), we have
\[ \|e(w^{k+1}, \Omega, Q)\| \leq \frac{1}{\tau}\|x^k - x^{k+1}\| \]
\[ + (\frac{\beta}{2}\|ATB\| + \frac{\beta(1 - \alpha)}{2}\|BTB\| + \frac{1}{\tau} + \frac{1}{2}\|B\|)\|y^k - y^{k+1}\| \]
\[ + (\frac{2\alpha - 1}{2\alpha}\|AT\| + \frac{1 - \alpha}{2\alpha}\|BT\| + \frac{1}{2\alpha\beta})\|\lambda^k - \lambda^{k+1}\| + \frac{2}{\tau}v_k. \tag{3.3.40} \]
Take \( a_1 = \max\{\frac{\beta}{2}\|ATB\| + \frac{\beta(1 - \alpha)}{2}\|BTB\| + \frac{1}{\tau} + \frac{1}{2}\|B\|, \frac{2\alpha - 1}{2\alpha}\|AT\| + \frac{1 - \alpha}{2\alpha}\|BT\| + \frac{1}{2\alpha\beta}\} \), the lemma is proved. \[ \square \]

Similarly, we have

**Lemma 3.3.8.** Let \( w^* = (x^*, y^*, \lambda^*) \in W^* \) be any solution point of MVI(\( \Omega, F, \theta \)). Let \( (x^k, y^k, \lambda^k) \) be the sequence generated by IPRSM2, then there exists constant
\( a_2 > 0 \), such that \( \forall k \geq 0 \), we have

\[
\|e(w^{k+1}, \Omega, \mathcal{Q})\| \leq a_2\|w^{k+1} - w^k\|. \tag{3.3.41}
\]

**Proof.** Again, from Lemma 3.3.6 and (3.1.24), we have

\[
\|e(w^{k+1}, \Omega, \mathcal{Q})\| \leq \left(\frac{1}{\tau} + \frac{1}{\tau}v_k\right)\|x^k - x^{k+1}\|
+
\left(\frac{\beta}{2}\|A^TB\| + \frac{\beta(1-\alpha)}{2}\|B^TB\| + \frac{1}{\tau} + \frac{1}{\tau}B\| + \frac{1}{\tau}v_k\right)\|y^k - y^{k+1}\|
+
\left((\frac{2\alpha - 1}{2\alpha})\|A^T\| + \frac{1 - \alpha}{2\alpha}\|B^T\| + \frac{1}{2\alpha\beta}\right)\|\lambda^k - \lambda^{k+1}\|.
\tag{3.3.42}
\]

Since \( \sum_{k=0}^{\infty} v_k^2 < \infty \), there exists a number \( \delta \), such that \( \delta \geq v_k \), for all \( k \). It follows that

\[
\|e(w^{k+1}, \Omega, \mathcal{Q})\| \leq \left(\frac{1}{\tau} + \frac{1}{\tau}\delta\right)\|x^k - x^{k+1}\|
+
\left(\frac{\beta}{2}\|A^TB\| + \frac{\beta(1-\alpha)}{2}\|B^TB\| + \frac{1}{\tau} + \frac{1}{\tau}B\| + \frac{1}{\tau}\delta\right)\|y^k - y^{k+1}\|
+
\left((\frac{2\alpha - 1}{2\alpha})\|A^T\| + \frac{1 - \alpha}{2\alpha}\|B^T\| + \frac{1}{2\alpha\beta}\right)\|\lambda^k - \lambda^{k+1}\|.
\tag{3.3.43}
\]

Let \( a_2 = \max\{\frac{\beta}{2}\|A^TB\| + \frac{\beta(1-\alpha)}{2}\|B^TB\| + \frac{1}{\tau} + \frac{1}{\tau}B\| + \frac{1}{\tau}\delta, \frac{\beta}{2}\|A^T\| + \frac{\beta(1-\alpha)}{2}\|B^T\| + \frac{1}{\tau} + \frac{1}{\tau}B\| + \frac{1}{\tau}\delta\} \), the lemma is proved.

\[
\square
\]

Now we are ready to present the main theorems and prove the convergence of IPRSM1 and IPRSM2.

**Theorem 3.3.9.** Let \( w^* = (x^*, y^*, \lambda^*) \in \mathcal{W}^* \) be any solution point of MVI(\( \Omega, F, \theta \)).
Let \( (x^k, y^k, \lambda^k) \) be the sequence generated by IPRSM1, then the sequence \( \{w^k\} \) converges to some \( w^\infty \in \mathcal{W}^* \).

**Proof.** Define

\[
C_1 := \sum_{k=0}^{\infty} v_k \text{ and } C_2 := \prod_{k=0}^{\infty}(1 + 2v_k).
\tag{3.3.44}
\]

69
By our assumption of criterion 1 on \( v_k \), we have

\[
0 \leq C_1 < \infty \quad \text{and} \quad 0 \leq C_2 < \infty.
\]

(3.3.45)

Without loss of generality, we assume \( 0 \leq v_k \leq \frac{1}{2} \) for all \( k \), then we have

\[
(1 - v_k)^{-1} \leq (1 + 2v_k), \quad \forall k.
\]

(3.3.46)

From Lemma 3.3.4 we have

\[
\|w^{k+1} - w^*\|^2_H \leq (1 + 2v_k)\|w^k - w^*\|^2_H + (1 + 2v_k)c_2v_k - c_1(1 + 2v_k)\|w^k - w^{k+1}\|^2_H
\]

\[
\leq (1 + 2v_k)\|w^k - w^*\|^2_H + (1 + 2v_k)c_2v_k
\]

\[
\leq (1 + 2v_k)(1 + 2v_{k-1})\|w^{k-1} - w^*\|^2_H + C_2c_2(v_k + v_{k-1})
\]

\[
\leq \cdots
\]

\[
\leq \prod_{i=k_0}^k (1 + 2v_i)\|w^{k_0} - w^*\|^2_H + C_2c_2 \sum_{i=k_0}^k v_i.
\]

(3.3.47)

This holds for any \( k \geq k_0 \). Take \( k_0 = 0 \), we have

\[
\|w^{k+1} - w^*\|^2_H \leq C_2\|w^0 - w^*\|^2_H + C_2C_1c_2, \quad \forall k.
\]

(3.3.48)

Therefore, the sequence \( \{w^k\} \) is bounded. From Lemma 3.3.4 and (3.3.48), we have

\[
c_1\|w^k - w^{k+1}\|^2_H \leq \|w^k - w^*\|^2_H - \|w^{k+1} - w^*\|^2_H + v_k\|w^{k+1} - w^*\|^2_H + c_2v_k
\]

\[
\leq \|w^k - w^*\|^2_H - \|w^{k+1} - w^*\|^2_H + v_k(C_2\|w^0 - w^*\|^2_H + C_2C_1c_2)
\]

\[+ c_2v_k.
\]

(3.3.49)
Summing over $k$, we get

$$
\sum_{k=0}^{\infty} c_1(\|w^{k+1} - w^k\|_H^2)
\leq \|w^0 - w^*\|_H^2 + C_1(C_2\|w^0 - w^*\|_H^2 + C_2C_1c_2) + C_1c_2
\leq \|w^0 - w^*\|_H^2 + C_1(\epsilon_1)^2
\leq \infty.
$$

(3.3.50)

Due to the equivalence of norms in finite dimensional space, we have

$$
\lim_{k \to \infty} \|w^{k+1} - w^k\| = 0.
$$

(3.3.51)

As $v_k \to 0$, from (3.3.39) we obtain

$$
\lim_{k \to \infty} \|e(w^{k+1}, \Omega, Q)\| = 0.
$$

(3.3.52)

Since $\{w^k\}$ is a bounded sequence, so it has a cluster point in $\Omega$. By Bolzano-Weierstrass Theorem, every bounded sequence has a convergent subsequence. We denote the subsequence by $\{w^{k_j}\}$ and its limit by $w_\infty$. Since $e(w^{k+1}, \Omega, Q)$ is continuous in $w$, it follows that

$$
\|e(w_\infty, \Omega, Q)\| = \lim_{j \to \infty} \|e(w^{k_j}, \Omega, Q)\| = 0.
$$

(3.3.53)

By (3.2.9), $w_\infty$ is a solution of MVI($\Omega, F, \theta$), and that means $w_\infty \in \Omega^*$. For any given $\epsilon > 0$, since $w^{kj} \to w_\infty$, there exists a positive integer $l$ such that

$$
\|w^{kj} - w_\infty\|_H^2 \leq \frac{\epsilon^2}{4}, \quad \sum_{i=k_l}^{\infty} v_i \leq \frac{\epsilon^2}{2C_2c_2}, \quad \prod_{i=k_l}^{\infty} (1 + 2v_i) \leq 2.
$$

(3.3.54)
For $k \geq k_l$, it follows from (3.3.47) that
\[
\|w^k - w^\infty\|_H^2 \leq \prod_{i=k_l}^{k-1} (1 + 2\nu_i)\|w_i - w^\infty\|_H^2 + C_2 c_2 \sum_{i=k_l}^{k-1} \nu_i
\]
\[
\leq \prod_{i=k_l}^\infty (1 + 2\nu_i)\|w_i - w^\infty\|_H^2 + C_2 c_2 \sum_{i=k_l}^\infty \nu_i
\]
\[
\leq \epsilon^2.
\]

Hence $\|w^k - w^\infty\|_H < \epsilon$ for $k \geq k_l$. Therefore, the sequence $w^k$ converges to $w^\infty \in \mathcal{W}^*$.

\[\square\]

**Theorem 3.3.10.** Let $w^* = (x^*, y^*, \lambda^*) \in \mathcal{W}^*$ be any solution point of MVI($\Omega, F, \theta$). Let $(x^k, y^k, \lambda^k)$ be the sequence generated by IPRSM2, then the sequence $\{w^k\}$ converges to some $w^\infty \in \mathcal{W}^*$.

**Proof.** Define
\[
D_1 := \sum_{k=0}^\infty \nu_k^2 \quad \text{and} \quad D_2 := \prod_{k=0}^\infty (1 + 2\nu_k^2).
\]
(3.3.56)

By our assumption of criterion 2 on $\nu_k$, we have
\[
0 \leq D_1 < \infty \quad \text{and} \quad 1 \leq D_2 < \infty.
\]
(3.3.57)

Again, without loss of generality, we assume $0 \leq \nu_k \leq \frac{1}{2}$ for all $k$, then we have
\[
(1 - \nu_k^2)^{-1} \leq (1 + 2\nu_k^2), \quad \forall k.
\]
(3.3.58)

It follows from Lemma 3.3.5 that
\[
\|w^{k+1} - w^*\|_H^2 \leq (1 + 2\nu_k^2) - c_3 (1 + 2\nu_k^2)\|w^{k+1} - w^k\|_H^2
\]
\[
\leq \cdots
\]
\[
\leq \prod_{i=k_0}^{k} (1 + 2\nu_i^2)\|w^{k_0} - w^*\|_H^2.
\]
(3.3.59)
This holds for any $k \geq k_0$. Take $k_0 = 0$ and it follows that

$$
\|w^{k+1} - w^*\|_H^2 \leq \prod_{i=0}^{k-1} (1 + 2\nu_i^2) \|w^0 - w^*\|_H^2 \leq D_2 \|w^0 - w^*\|_H^2.
$$

(3.3.60)

Therefore, $\{w^k\}$ is bounded. From Lemma 3.3.5 and (3.3.60), we have

$$
c_3\|w^{k+1} - w^k\|_H^2 \leq \|w^k - w^*\|_H^2 - (1 - \nu_k^2)\|w^{k+1} - w^*\|_H^2 \\
= \|w^k - w^*\|_H^2 - \|w^{k+1} - w^*\|_H^2 + \nu_k^2\|w^{k+1} - w^*\|_H^2 \\
\leq \|w^k - w^*\|_H^2 - \|w^{k+1} - w^*\|_H^2 + \nu_k^2D_2\|w^0 - w^*\|_H^2.
$$

(3.3.61)

Summing over $k$, we get

$$
\sum_{k=0}^{\infty} c_3(\|w^{k+1} - w^k\|_H^2) \leq \|w^0 - w^*\|_H^2 + \sum_{k=0}^{\infty} \nu^2(D_2\|w^0 - w^*\|_H^2) \\
= \|w^0 - w^*\|_H^2 + D_1D_2\|w^0 - w^*\|_H^2 \\
< \infty.
$$

(3.3.62)

Once again, due to the equivalence of norms in finite dimensional space, we have

$$
\lim_{k \to \infty} \|w^{k+1} - w^k\| = 0.
$$

(3.3.63)

As $\nu_k \to 0$, from (3.3.41) we obtain

$$
\lim_{k \to \infty} \|e(w^{k+1}, \Omega, Q)\| = 0.
$$

(3.3.64)

Since $\{w^k\}$ is a bounded sequence, so it has a cluster point in $\Omega$. By Bolzano-Weierstrass Theorem, every bounded sequence has a convergent subsequence. We denote the subsequence by $\{w^{k_j}\}$ and its limit by $w_\infty$. Since $e(w^{k+1}, \Omega, Q)$ is contin-
uous in \( w \), it follows that
\[
\|e(w^\infty, \Omega, Q)\| = \lim_{j \to \infty} \|e(w^{kj}, \Omega, Q)\| = 0. \tag{3.3.65}
\]

By (3.2.9), \( w^\infty \) is a solution of MVI(\( \Omega, F, \theta \)), and that means \( w^\infty \in \Omega^* \). For any given \( \epsilon > 0 \), since \( w^{kj} \to w^\infty \), there exists a positive integer \( l \) such that
\[
\|w^{kl} - w^\infty\|_H^2 \leq \frac{\epsilon^2}{4}, \quad \prod_{i=kl}^\infty (1 + 2\nu_i^2) \leq 4. \tag{3.3.66}
\]

For \( k \geq kl \), it follows from (3.3.59) that
\[
\|w^k - w^\infty\|_H^2 \leq \prod_{i=kl}^{k-1} (1 + 2\nu_i^2) \|w^{kl} - w^\infty\|_H^2 \leq \prod_{i=kl}^\infty (1 + 2\nu_i^2) \|w^{kl} - w^\infty\|_H^2 \leq \epsilon^2. \tag{3.3.67}
\]

Hence \( \|w^k - w^\infty\|_H < \epsilon \) for \( k \geq kl \). Therefore, the sequence \( w^k \) converges to \( w^\infty \in \mathcal{W}^* \).

\[ \square \]

### 3.4 Ergodic Worst-case \( O(1/t) \) Convergence Rate

In this section, we establish a worst-case \( O(1/t) \) convergence rate for IPRSM1 and IPRSM2, respectively. We find an approximate solution to MVI(\( \Omega, F, \theta \)) with an accuracy of \( O(1/t) \) based on the first \( t \) iterations of IPRSM1 and IPRSM2. We first define matrix \( P \) as
\[
P = \begin{pmatrix}
I_{n_1} & 0 & 0 \\
0 & I_{n_2} & 0 \\
0 & 0 & 0
\end{pmatrix}, \tag{3.4.1}
\]

74
so that we can express the sum of Euclidean-norm which only involves $x$ and $y$ terms easily.

**Theorem 3.4.1.** Let $\{w^k\}$ be generated by IPRSM1 with $\alpha = 1$ and $\{\tilde{w}^k\}$ be defined by (3.3.1). Let $\{\tilde{w}^t\}$ be defined as

$$\tilde{w}_t = \frac{1}{t + 1} \sum_{k=0}^{t} \tilde{w}^k, \quad (3.4.2)$$

then, for any integer number $t > 0$, $\tilde{w}_t \in \Omega$, we have

$$\theta(u) - \theta(\tilde{u}_k) + (w - \tilde{w}_k)^T F(w) \leq$$

$$\frac{1}{2(t + 1)}[\|w - w^0\|_H^2 + \frac{1}{\tau} \sum_{k=0}^{t} \nu_k (2 + \|w - w^{k+1}\|_H^2)], \quad \forall w \in \Omega. \quad (3.4.3)$$

**Proof.** We first use Cauchy-Schwartz Inequality to bound the inexactness criterion error terms:

$$(\tilde{x}^k - x)^T \xi_x^k \geq -\frac{\nu_k}{2} \|\tilde{x}^k - x\|^2 - \frac{1}{2\nu_k} \|\xi_x\|^2 \geq -\frac{\nu_k}{2} \|\tilde{x}^k - x\|^2 - \frac{\nu_k}{2}, \quad \forall x \in \mathcal{X}. \quad (3.4.4)$$

Similarly, we have

$$(\tilde{y}^k - y)^T \xi_y^k \geq -\frac{\nu_k}{2} \|\tilde{y}^k - y\|^2 - \frac{\nu_k}{2}, \quad \forall y \in \mathcal{Y}. \quad (3.4.5)$$

These two inequalities will be also used in establishing the nonergodic convergence rate. Setting $\alpha = 1$ in (3.3.12), we have

$$\theta(u) - \theta(\tilde{u}_k) + (w - \tilde{w}_k)^T F(\tilde{w}_k) \geq$$

$$\frac{1}{2} (\|w - w^{k+1}\|_H^2 - \|w - w^k\|_H^2) - \frac{1}{\tau} (w - \tilde{w}_k)^T \xi_k, \quad \forall w \in \Omega, \quad (3.4.6)$$
Substituting (3.4.4) and (3.4.5) into (3.4.6), and noticing that $\xi_k = ((\xi^k_x)^T, (\xi^k_y)^T, 0)$, we get

$$
\theta(u) - \theta(\tilde{u}^k) + (w - \tilde{w}^k)^T F(\tilde{w}^k) \geq \frac{1}{2}(\|w - w^{k+1}\|_H^2 - \|w - w^k\|_H^2)
- \frac{v_k}{\tau} - \frac{v_k}{2\tau}(\|x^{k+1} - x\|^2 + \|y^{k+1} - y\|^2), \quad \forall w \in \Omega.
$$

(3.4.7)

Since $F$ is monotone, applying (3.3.21) to (3.4.7), we have

$$
\theta(u) - \theta(\tilde{u}^k) + (w - \tilde{w}^k)^T F(w) \geq \frac{1}{2}(\|w - w^{k+1}\|_H^2 - \|w - w^k\|_H^2)
- \frac{v_k}{\tau} - \frac{v_k}{2\tau}(\|x^{k+1} - x\|^2 + \|y^{k+1} - y\|^2), \quad \forall w \in \Omega.
$$

(3.4.8)

Summing this inequality over $k = 0, 1, \ldots, t$, we obtain

$$
(t + 1)\theta(u) - \sum_{k=0}^{t} \theta(\tilde{u}^k) + [(t + 1)w - \sum_{k=0}^{t} \tilde{w}^k]^T F(w) \geq \frac{1}{2}(\|w - w^{t+1}\|_H^2 - \|w - w^0\|_H^2)
- \frac{1}{\tau} \sum_{k=0}^{t} v_k [1 + \frac{1}{2}(\|x^{k+1} - x\|^2 + \|y^{k+1} - y\|^2)] \quad \forall w \in \Omega.
$$

(3.4.9)

Since $\theta(u)$ is convex and

$$
\tilde{u}_t = \frac{1}{t + 1} \sum_{k=0}^{t} \tilde{u}^k,
$$

we have

$$
\theta(\tilde{u}_t) \leq \frac{1}{t + 1} \sum_{k=0}^{t} \theta(\tilde{u}^k).
$$

Substituting it into (3.4.9), the assertion (3.4.3) follows immediately. \qed

It follows from the previous convergence analysis that the sequence \{w^k\} generated by IPRSM1 is bounded. Let $w^0 = (x^0, y^0, \lambda^0)$ be the initial iterate. For a given compact set $\mathcal{D} \in \mathcal{X} \times \mathcal{Y} \times \mathbb{R}^m$, let $d_1 = \sup\{\|w - w^0\|_H | w \in \mathcal{D}\}$, and let $d_2 = \sup\{\|w - w^k\|_P | w \in \mathcal{D}, \forall k\}$. We also assume $C_1 = \sum_{k=0}^{\infty} v_k$. Therefore, after $t$ iterations
of IPRSM1, the point \( \tilde{w}_t \) satisfies

\[
\sup_{w \in D} \{ \theta(\tilde{u}_t) - \theta(u) + (\tilde{w}_t - w)^T F(w) \} \leq \frac{1}{2(t+1)} [d_1^2 + \frac{C_1}{\tau} (1 + d_2^2)].
\]

(3.4.10)

This completes the proof since \( \tilde{w}_t \) is an approximate solution of MVI(\( \Omega, F, \theta \)) with an accuracy of \( O(1/t) \).

Note that we take \( \alpha = 1 \) in the proof. When \( \alpha \in (0,1) \), the proof is still valid since the sum of the extra term \( \frac{1-\alpha}{4(1+\alpha)} \| w^k - w^{k+1} \|^2_H \) in (3.3.12) is bounded as shown in the convergence analysis. Thus, we omit the proof and conclude that a worst-case \( O(1/t) \) convergence rate can be established easily for IPRSM1 when \( \alpha \in (0,1) \).

Now we establish a worst-case \( O(1/t) \) convergence rate in ergodic sense for IPRSM2.

**Theorem 3.4.2.** Let \( \{w^k\} \) be generated by IPRSM2 with \( \alpha = 1 \) and \( \{\tilde{w}^k\} \) be defined by (3.3.1). Let \( \tilde{w}^t \) be defined as

\[
\tilde{w}_t = \frac{1}{t+1} \sum_{k=0}^{t} \tilde{w}^k,
\]

(3.4.11)

then, for any integer number \( t > 0 \), \( \tilde{w}_t \in \Omega \), we have

\[
\theta(u) - \theta(\tilde{u}_k) + (w - \tilde{w}_k)^T F(w) \leq \frac{1}{2(t+1)} \| w - w^0 \|^2_H + \frac{1}{\tau} \left( 2 \sum_{k=0}^{t} u_k^2 \| w^k - w^{k+1} \|^2_P + \frac{1}{2} \sum_{k=0}^{t} \| w^k - w^{k+1} \|^2_P \right), \quad \forall w \in \Omega.
\]

(3.4.12)

**Proof.** We first use Cauchy-Schwartz Inequality to bound the inexactness criterion error terms:

\[
(\bar{x}^k - x)^T \xi_x^k \geq -\frac{2u_k^2}{2} \| \bar{x}^k - x \|^2 - \frac{1}{4u_k^2} \| \xi_x^k \|^2 \geq -u_k^2 \| x^{k+1} - x \|^2 - \frac{1}{4} \| x^k - x^{k+1} \|^2, \quad \forall x \in \mathcal{X}.
\]

(3.4.13)
Similarly, we have
\begin{align*}
(\tilde{y}^k - y)^T \xi_y^k \geq -v_k^2 \| y^{k+1} - y \|^2 - \frac{1}{4} \| y^k - x^{y+1} \|^2, \quad \forall y \in \mathcal{Y}.
\end{align*}
(3.4.14)

These two inequalities will be also used in establishing the nonergodic convergence rate. Setting $\alpha = 1$ in (3.3.12), we have
\begin{align*}
\theta(u) - \theta(\tilde{u}^k) + (w - \tilde{w}^k)^T F(\tilde{w}^k) &\geq \frac{1}{2} (\| w - w^{k+1} \|^2_H - \| w - w^k \|^2_H) \\
&- \frac{1}{\tau} (w - \tilde{w}^k) \xi^k, \quad \forall w \in \Omega.
\end{align*}
(3.4.15)

Adding the above three inequalities, and noticing that $F$ is monotone, we get
\begin{align*}
\theta(u) - \theta(\tilde{u}^k) + (w - \tilde{w}^k)^T F(w) &\geq \frac{1}{2} (\| w - w^{k+1} \|^2_H - \| w - w^k \|^2_H) \\
&- \frac{1}{\tau} (v^2_k (w^{k+1} - w) \| F(w) \|^2_p + \frac{1}{4} \| w^{k+1} - w^k \|^2_p), \quad \forall w \in \Omega.
\end{align*}
(3.4.16)

Summing this inequality over $k = 0, 1, \cdots, t$, we obtain
\begin{align*}
(t + 1) \theta(u) - \sum_{k=0}^t \theta(\tilde{u}^k) + [(t + 1) w - \sum_{k=0}^t \tilde{w}^k]^T F(w) \geq &\ \frac{1}{2} \| w - w^{t+1} \|^2_H \\
- &\ \| w - w^0 \|^2_H - \frac{1}{\tau} \sum_{k=0}^t v^2_k \| w^{k+1} - w \|^2_p - \frac{1}{4\tau} \sum_{k=0}^t \| w^{k+1} - w^k \|^2_p, \quad \forall w \in \Omega.
\end{align*}
(3.4.17)

Since $\theta(u)$ is convex and
\begin{align*}
\tilde{u}_t = \frac{1}{t+1} \sum_{k=0}^t \tilde{u}^k,
\end{align*}
we have
\begin{align*}
\theta(\tilde{u}_t) \leq \frac{1}{t+1} \sum_{k=0}^t \theta(\tilde{u}^k).
\end{align*}

Substituting it into (3.4.17), the assertion (3.4.12) follows immediately. \qed

Notice that all terms \((\| w - w^0 \|^2_H, \sum_{k=0}^t v^2_k, \sum_{k=0}^t \| w^{k+1} - w \|^2_p \) and \(\sum_{k=0}^t \| w^{k+1} - w^k \|^2_p)\) on the right side of (3.4.12) are bounded as we showed in the convergence
analysis; and of course, the proof is still valid when $\alpha \in (0, 1)$. Therefore, We conclude that a worst-case $O(1/t)$ convergence rate in ergodic sense for IPRSM2 is established.

### 3.5 Nonergodic Worst-case $O(1/t)$ Convergence Rate

In this section, we establish a worst-case $O(1/t)$ convergence rate in nonergodic sense for IPRSM1 and IPRSM2, respectively.

We first establish the nonergodic convergence rate for IPRSM1. We notice that the sequence $\{w^k\}$ generated by IPRSM1 converges to the solution, so that it is bounded. Therefore, for any given $w^* \in \Omega^*$, there is a constant $C_{w^*}$ such that for any $k \geq 0$, we have $\|w^k - w^*\|_P \leq C_{w^*}$, where $P$ is defined in (3.4.1).

**Theorem 3.5.1.** Let the sequence $\{w^k\}$ be generated by IPRSM1, and the sequence $\{\tilde{w}^k\}$ be defined by (3.3.1), then for any $w^* \in \Omega^*$, we have

$$\frac{1 - \alpha}{2(1 + \alpha)} \min_{k \in \{0, \ldots, t\}} \|w^k - \tilde{w}^k\|_{M^T HM} \leq \frac{1}{t + 1} (\|w^* - w^0\|_H^2 + \frac{2C_1}{\tau} + \frac{C_{w^*}^2 C_1}{\tau}),$$  

where $C_1 = \sum_{k=0}^{\infty} u_k$, $M$ and $H$ are defined in (3.2.11) and (3.2.12), respectively.

**Proof.** Summing (3.3.12), (3.4.4) and (3.4.5), we have

$$\theta(u) - \theta(\tilde{u})^k + (w - \tilde{w}^k)^T F(\tilde{w}^k) \geq \frac{1}{2} (\|w - w^{k+1}\|_H^2 - \|w - w^k\|_H^2)$$

$$+ \frac{1 - \alpha}{4(1 + \alpha)} \|w^k - w^{k+1}\|_H^2 - \frac{u_k}{\tau} (1 + \frac{1}{2} \|w^{k+1} - w^*\|_P^2), \quad \forall w \in \Omega. \tag{3.5.2}$$

Setting $w = w^*$ in (3.5.2), and using (3.3), we get:

$$\theta(w^*) - \theta(\tilde{w}) + (w^* - \tilde{w})^T F(\tilde{w}) \geq \frac{1}{2} (\|w^* - w^{k+1}\|_H^2 - \|w^* - w^k\|_H^2)$$

$$+ \frac{1 - \alpha}{4(1 + \alpha)} \|w^k - w^{k+1}\|_{M^T HM}^2 - \frac{u_k}{\tau} (1 + \frac{1}{2} \|w^{k+1} - w^*\|_P^2). \tag{3.5.3}$$
Since $F$ is monotone and $w^* \in \Omega^*$, we have

$$0 \geq \theta(u^*) - \theta(\tilde{u}^k) + (w^* - \tilde{w}^k)F(w^*) \geq \theta(u^*) - \theta(\tilde{u}^k) + (w^* - \tilde{w}^k)F(\tilde{w}^k). \quad (3.5.4)$$

Combining (3.5.3) and (3.5.4), we get

$$1 - \frac{\alpha}{2(1 + \alpha)} \sum_{k=0}^{t} \|w^k - \tilde{w}^k\|_{M^T H M}^2 \leq \|w^* - w^0\|_H^2 + \sum_{k=0}^{t} 2\left(\frac{\nu}{\tau} + \|w^{k+1} - w^*\|_P^2\right). \quad (3.5.5)$$

Summing this inequality over $k = 0, 1, \cdots, t$, we obtain

$$1 - \frac{\alpha}{2(1 + \alpha)} \sum_{k=0}^{t} \|w^k - \tilde{w}^k\|_{M^T H M}^2 \leq \|w^* - w^0\|_H^2 + \sum_{k=0}^{t} 2\left(\frac{\nu_{\ell}}{\tau} + \|w^{k+1} - w^*\|_P^2\right). \quad (3.5.6)$$

The assertion follows directly from the above inequality.

If $\|w^k - \tilde{w}^k\|_H = 0$, we can easily verify that $w^k \in \Omega$ can be regarded as a $\nu_{\ell}$-approximation solution of MVI($\Omega, F, \theta$). Therefore, we conclude that $\|w^k - \tilde{w}^k\|_H$ can be viewed as an error measurement in terms of the distance to the solution set of MVI($\Omega, F, \theta$) for the $(k + 1)$-th iteration of IPRSM1 and this establishes a worst-case $O(1/t)$ convergence rate in nonergodic sense.

Now we establish a worst-case $O(1/t)$ convergence rate in nonergodic sense for IPRSM2. Again, we know that the sequence $\{w^k\}$ generated by IPRSM2 converges to the solution, so that it is bounded. Therefore, for any given $w^* \in \Omega^*$, we conclude that there is a constant $C_{w^*}$, such that for any $k \geq 0$, we have $\|w^k - w^*\|_P \leq D_{w^*}$, where $P$ is defined in (3.4.1).

**Theorem 3.5.2.** Let the sequence $\{w^k\}$ be generated by IPRSM2, and the sequence $\{\tilde{w}^k\}$ be defined by (3.3.1), then for any $w^* \in \Omega^*$, we have

$$\frac{\alpha^*}{2} \min_{k \in \{0, \cdots, t\}} \|w^k - \tilde{w}^k\|_L \leq \frac{1}{t + 1}(\|w^* - w^0\|_H^2 + \frac{2D_{w^*}^2 D_{w^*}}{\tau \alpha^*}), \quad (3.5.7)$$
where $D_1 := \sum_{k=0}^{\infty} u_k^2$, $M$ and $H$ are defined in (3.2.11) and (3.2.12). $\alpha^*$ is defined as $\alpha^* = \frac{1-\alpha}{1+\alpha}$ and matrix $L$ is defined as $L = M^T H M - \frac{1}{\tau} P$ in order to alleviate the notation.

**Proof.** We first use Cauchy-Schwartz Inequality to bound the inexactness criterion error terms:

$$\begin{align*}
(\tilde{x}^k - x)^T \xi^k &\geq -\frac{\nu_k^2}{\alpha^*} \|\tilde{x}^k - x\|^2 - \frac{\alpha^*}{4\nu_k^2} \|\xi^k\|^2 \\
&\geq -\frac{\nu_k^2}{\alpha^*} \|x^{k+1} - x\|^2 - \frac{\alpha^*}{4} \|x^k - x^{k+1}\|^2, \quad \forall x \in X.
\end{align*}
$$

(3.5.8)

Similarly, we have

$$
(\tilde{y}^k - y)^T \xi^k \geq -\frac{\nu_k^2}{\alpha^*} \|y^{k+1} - y\|^2 - \frac{1}{4\alpha^*} \|y^k - y^{k+1}\|^2, \quad \forall y \in Y.
$$

(3.5.9)

Combining (3.3.12), (3.5.8) and (3.5.9), we have

$$\begin{align*}
\theta(u) - \theta(\tilde{u}^k) + (w - \tilde{w}^k)^T F(\tilde{w}^k) &\geq \frac{1}{2}(\|w - w^{k+1}\|_H^2 - \|w - w^k\|_H^2) \\
+ \frac{\alpha^*}{4} \|w^k - w^{k+1}\|_H^2 - \frac{\nu_k^2}{\tau} \|w^{k+1} - w\|_P^2 - \frac{\alpha^*}{4\tau} \|w^k - \tilde{w}^k\|_P^2, \quad \forall w \in \Omega.
\end{align*}
$$

(3.5.10)

Setting $w = w^*$ in (3.5.10), and using (3.3.2), we get:

$$\begin{align*}
\theta(u^*) - \theta(\tilde{u}^k) + (w^* - \tilde{w}^k)^T F(\tilde{w}^k) &\geq \frac{1}{2}(\|w^* - w^{k+1}\|_H^2 - \|w^* - w^k\|_H^2) \\
+ \frac{\alpha^*}{4} \|w^k - \tilde{w}^k\|^2_{M^T H M} - \frac{\nu_k^2}{\tau} \|w^{k+1} - w^*\|_P^2 - \frac{\alpha^*}{4\tau} \|w^k - \tilde{w}^k\|_P^2, \quad \forall w \in \Omega.
\end{align*}
$$

(3.5.11)

Since $F$ is monotone and $w^* \in \Omega^*$, we have

$$
0 \geq \theta(u^*) - \theta(\tilde{u}^k) + (w^* - \tilde{w}^k) F(w^*) \geq \theta(u^*) - \theta(\tilde{u}^k) + (w^* - \tilde{w}^k) F(\tilde{w}^k).
$$

(3.5.12)
Combining (3.5.11) and (3.5.12), we get
\[
\frac{\alpha^*}{2} \|w^k - \tilde{w}^k\|^2_{M^T H M} \leq \|w^* - w^k\|^2_H - \|w^* - w^{k+1}\|^2_H \\
+ \frac{2\nu_k^2}{\tau \alpha^*} \|w^{k+1} - w^*\|^2_P + \frac{\alpha^*}{2\tau} \|w^k - \tilde{w}^k\|^2_P.
\] (3.5.13)

We notice \( L = M^T H M - \frac{1}{\tau} P = \begin{pmatrix} 0 & 0 & 0 \\ 0 & (1 + \alpha)\beta B^T B & -2\alpha B^T \\ 0 & -2\alpha B & \frac{2\alpha}{\beta} I_m \end{pmatrix} \), and it is very easy to check that \( L \) is positive semidefinite. Now (3.5.13) becomes
\[
\frac{\alpha^*}{2} \|w^k - \tilde{w}^k\|^2_L \leq \|w^* - w^k\|^2_H - \|w^* - w^{k+1}\|^2_H + \frac{2\nu_k^2}{\tau \alpha^*} \|w^{k+1} - w^*\|^2_P.
\] (3.5.14)

Summing the above inequality over \( k = 0, 1, \ldots, t \), we have
\[
\frac{\alpha^*}{2} \sum_{k=0}^t \|w^k - \tilde{w}^k\|^2_L \leq \|w^* - w^0\|^2_H + \sum_{k=0}^t \frac{2\nu_k^2}{\tau \alpha^*} \|w^{k+1} - w^*\|^2_P.
\] (3.5.15)

The assertion follows directly from the above inequality.

If \( \|w^k - \tilde{w}^k\|_H = 0 \), then it indicates that \( w^k \in \Omega \) is a solution of \( \text{MVI}(\Omega, F, \theta) \). Therefore, we conclude that \( \|w^k - \tilde{w}^k\|_H \) can be viewed as an error measurement in terms of the distance to the solution set of \( \text{MVI}(\Omega, F, \theta) \) for the \((k+1)\)-th iteration of IPRSM2 and this establishes a worst-case \( O(1/t) \) convergence rate in nonergodic sense.

### 3.6 Numerical Results

In this section, we focus on the applications of the inexact PRSM methods to some statistical learning problems. We use the LASSO model and the \( l_1 \)-regularized sparse
logistic regression model as the examples to illustrate the advantages of our proposed methods. We make the following assertions and verify them in the experiments:

1). IPRSM1 and IPRSM2 are both fast and convergent. If properly implemented, they can be much faster than the exact ADMM and exact strictly contractive PRSM.

2). The stopping criterion for the PRSM iterative scheme and the sequence \( \{ u_k \} \), used to control the stopping criterion of each iteration, they together play an important role in maintaining the speed and efficiency of the algorithm.

3). The empirically ideal values of \( \alpha (\alpha \in [0.8, 0.9]) \) \(^{31}\) for the exact strictly contractive PRSM still hold for IPRSM1. This underdetermined relaxation factor \( \alpha \) works for a wide range of values.

All numerical experiments were written in MATLAB 2012 and conducted on a computer with a 2.6GHz processor and 8GB memory. In the first subsection, we introduce the statistical learning models and their iterative schemes. We give the details of our experiments and present the result of our experiments in the second subsection.

### 3.6.1 Statistical Models and Implementation Details

1). The LASSO model, proposed in \(^{60}\), is designed to solve the following problem:

\[
\min \left\{ \frac{1}{2} \| Dx - r \|_2^2 + \gamma \| x \|_1, \ x \in \mathbb{R}^d \right\},
\]

where \( r \) is a \( n \)-dimensional response vector, \( D \) is a \( n \times d \) dimensional design matrix. Note that \( n \) is the number of data points while \( d \) is the number of the features and \( \gamma \) is the regularization parameter. For the applications of the LASSO model, see \(^{23} 49, 65\).

Applying (3.1.10) to the LASSO model, with \( \theta_1(x) = \frac{1}{2} \| Dx - r \|_2^2, \theta_2(y) = \gamma \| y \|_1, \) \( n_1 = n_2 = d, \ A = I_d, \ B = -I_d, \ b = 0, \ X = \mathbb{R}^d, \ Y = \mathbb{R}^d, \) the strictly contractive
PRSM method has the following iterative scheme:

\[
\begin{align*}
  x^{k+1} &= (D^T D + \beta I)^{-1} (D^T r + \beta y^k + \lambda^k), \\
  \lambda^{k+\frac{1}{2}} &= \lambda^k - \alpha \beta (x^{k+1} - y^k), \\
  y^{k+1} &= S_{\gamma/\beta} (x^{k+1} - \lambda^{k+\frac{1}{2}}/\beta), \\
  \lambda^{k+1} &= \lambda^{k+\frac{1}{2}} - \alpha \beta (x^{k+1} - y^{k+1}),
\end{align*}
\]

(3.6.2)

where \( S_b(a) \) is the soft-thresholds operator. For a \( d \)-dimensional vector \( a \) and a positive number \( b \), the soft-thresholds operator is defined as \( S_b(a)_i = (1 - b |a_i|)_+ \).

The iterative step for solving \( y^k \) involves a shrinkage operation and there is no need to do it inexactly since the exact method is already accurate and efficient; therefore, we only focus on the application of inexact PRSM scheme to the \( x \) update. Based on (3.1.12), the inexact method needs the solution to:

\[
  x^{k+1} = [D^T D + (\beta + \frac{1}{\tau}) I]^{-1} (D^T r + \beta y^k + \lambda^k + \frac{1}{\tau} x^k),
\]

(3.6.3)

for the \( x^k \) iterative step.

It is not practical to invert the huge matrix \( D^T D + (\beta + \frac{1}{\tau}) I \). We propose to use the Jacobi Method to solve the linear system inexactly and thus IPRSM1 and IPRSM2 can be used here. The Jacobi Method stops when the inexactness criterion is satisfied. To be specific, for each iteration, when the Jacobi Method generates \( x_j^{k+1} \), we use (3.1.25) to find \( \tilde{x}_j^{k+1} \). Next, we calculate \( \xi^k_x \) by (3.1.26). Finally, we check if \( \xi^k_x \) satisfies IPRSM1 or IPRSM2 for some sequences \( \{v_k\} \).

In our experiment, the entries of \( D \) are drawn from \( \mathcal{N}(0,1) \) and normalized. The random sparse vector \( x \) has 100 nonzero component and each of them is also drawn from \( \mathcal{N}(0,1) \). The noise vector \( \epsilon \sim \mathcal{N}(0,10^{-3} I) \), and we set vector \( r = D x + \epsilon \).
2). The $l_1$-regularized sparse logistic regression model, as discussed in [43, 44, 55], solves:

$$
\min \left\{ \frac{1}{n} \sum_{i=1}^{n} \log(1 + \exp(-r_i(a_i^T x + x_0))) + \gamma \|x\|_1, \quad x \in \mathbb{R}^d, x \in \mathbb{R} \right\},
$$

(3.6.4)

where $r_i \in \{1, -1\}$ are the corresponding labels, $a_i$ is a $d$-dimensional training data vector, $n$ is the number of data points, $d$ is the dimension of the data, and $\gamma$ is a regularization parameter. We are trying to minimize over $x$ (the coefficient $d$-dimensional vector) and $x_0$ (the intercept scalar).

Applying (3.1.10) to the sparse logistic regression model, with $\theta_1(x) = \frac{1}{n} \sum_{i=0}^{n} \log(1 + \exp(-r_i(a_i^T x + x_0)))$, $\theta_2(y) = \gamma \|y\|_1$, $n_1 = n_2 = d$, $A = I_d$, $B = -I_d$, $b = 0$, $\mathcal{X} = \mathbb{R}^d$, $\mathcal{Y} = \mathbb{R}^d$, the strictly contractive PRSM method has the following iterative scheme:

$$
\begin{align*}
&x^{k+1} = \arg \min \frac{1}{n} \sum_{i=1}^{n} \log(1 + \exp(-r_i(a_i^T x + x_0))) + \beta \|x - y^k - \lambda^k / \beta\|_2^2, \\
&\lambda^{k+\frac{1}{2}} = \lambda^k - \alpha \beta (x^{k+1} - y^k), \\
y^{k+1} = S_{\gamma / \beta}(x^{k+1} - \lambda^{k+\frac{1}{2}} / \beta), \\
&\lambda^{k+1} = \lambda^{k+\frac{1}{2}} - \alpha \beta (x^{k+1} - y^{k+1}),
\end{align*}
$$

(3.6.5)

Since the $y^k$ iteration has an exact solution, we only focus on the $x^k$ iteration and use the inexact PRSM methods to solve it. Based on (3.1.12), for the $x^k$ iterative step, the inexact method needs to solve:

$$
x^{k+1} = \arg \min \frac{1}{n} \sum_{i=1}^{n} \log(1 + \exp(-r_i(a_i^T x + x_0))) + \frac{\beta}{2} \|x - y^k - \lambda^k / \beta\|_2^2 + \frac{1}{2\tau} \|x - x^k\|_2^2.
$$

(3.6.6)
We implement Newton’s method for the $x^k$ update and stop whenever the inexactness criterion is satisfied. The details of how this algorithm works in this case can be found in [5]. To be specific, for each iteration, whenever Newton’s method generates $x_j^{k+1}$, we use (3.1.25) to find $\tilde{x}_j^{k+1}$. Next, we calculate $\xi^k_x$ by (3.1.26). Again, we check if $\xi^k_x$ satisfy IPRSM1 or IPRSM2 with some fixed sequences $\{\upsilon_k\}$.

We specify some implantation details for the sparse logistic regression model as follows: We follow most of the conventions used in the original strictly contractive PRSM paper [31]. The response vector $r_i$ is generated as $r_i = \text{sign}(a_i^T x + x_0 + \epsilon_i)$, where $a_i$ has 10 nonzero entries from $\mathcal{N}(0,1)$, vector $x$ has 100 nonzero entries drawn from $\mathcal{N}(0,1)$, $x_0$ is a real number from $\mathcal{N}(0,1)$, noise vector $\epsilon$ is drawn from $\mathcal{N}(0,0.1)$. The regularization parameter for the sparse logistic regression model is set according to [43].

### 3.6.2 Experiment setup and results

In this subsection, we study the properties of IPRSM1 and IPRSM2 via some numerical experiments. We first verify the speed of both methods and then we study the relations between the main iterative scheme and the inexact iterative scheme. Finally, we confirm the acceleration effect of the relaxation factor $\alpha$ for our methods. We set the Augmented Lagrangian penalty parameter $\beta = 1$ throughout all experiments.

1). The first experiment is designed to verify the speed of IPRSM1 or IPRSM2. We compare the speeds and the number of iterations for exact ADMM, the exact strictly contractive PRSM, IPRSM1 and IPRSM2. We set the contractive constant $\alpha = 0.8$ since this is the best accelerated parameter we find for our models. The stopping criterion for the iterative PRSM scheme requires both primal and dual residuals to be relatively small. Namely, $\beta\|y^k - y^{k+1}\|_2 \leq \sqrt{d}\epsilon$ and $\frac{1}{\beta}\|\lambda^k - \lambda^{k+1}\|_2 \leq \sqrt{d}\epsilon$. In addition, the convergence sequence $\{\upsilon_k\}$ is set to be $\frac{1}{n\gamma}$, with $\gamma = 1.1$. We compare the performance among exact strictly contractive PRSM, exact ADMM, IPRSM1
and IPRSM2 and list the CPU time and the iteration number required for each experiment. For the LASSO model, we set the feature number \( d = 1000 \) and the data points \( n = 1000 \). For the sparse logistic regression model, we set the dimension \( d = 400 \) and the example number \( n = 1000 \). The result of timing performance and the iteration numbers are shown in Table 3.1 and Table 3.2, respectively.

Table 3.1: Quantitative time comparison (in seconds) among the exact ADMM, the exact strictly contractive PRSM, IPRSM1 and IPRSM2

<table>
<thead>
<tr>
<th>Model</th>
<th>Exact ADMM</th>
<th>Exact PRSM</th>
<th>IPRSM1</th>
<th>IPRSM2</th>
</tr>
</thead>
<tbody>
<tr>
<td>Lasso</td>
<td>2.82</td>
<td>2.58</td>
<td>1.32</td>
<td>1.85</td>
</tr>
<tr>
<td>Sparse Logistic Regression</td>
<td>2.54</td>
<td>1.70</td>
<td>1.51</td>
<td>1.58</td>
</tr>
</tbody>
</table>

Table 3.2: Quantitative iteration number comparison among the exact ADMM, the exact strictly contractive PRSM, IPRSM1 and IPRSM2

<table>
<thead>
<tr>
<th>Model</th>
<th>Exact ADMM</th>
<th>Exact PRSM</th>
<th>IPRSM1</th>
<th>IPRSM2</th>
</tr>
</thead>
<tbody>
<tr>
<td>Lasso</td>
<td>10</td>
<td>9</td>
<td>17</td>
<td>16</td>
</tr>
<tr>
<td>Sparse Logistic Regression</td>
<td>11</td>
<td>10</td>
<td>13</td>
<td>13</td>
</tr>
</tbody>
</table>

For both models, we show that IPRSM1 and IPRSM2 outperform the exact ADMM as well as the exact strictly contractive PRSM in terms of time. Although the inexact PRSM methods do require a bit more iterations, the total amount of time is still less than the exact methods. This is due to the fact that each update in the main iterative scheme is calculated inexactly, a few more iterations are required to achieve convergence. We visualize the evolution of the primal residual, dual residual and the objective value when these methods are applied to the LASSO model and the sparse logistic regression model in Figure 3.1 and 3.2, respectively. We omit IPRSM2 when doing comparison since it shares very similar properties with IPRSM1. Both figures indicate that less iterations are required for both ADMM and strictly contractive PRSM; however, IPRSM1 is much faster and more efficient in terms of total time.

2). Next, we study the relations between two iterations: the main iterative scheme (3.1.19) to (3.1.22) and the inexactness iterative scheme. We call the main iterative
scheme the outer iteration and the inexactness iterative scheme the inner iteration. In this part, we start to investigate the relations between the stopping criterion of the two iterative schemes. Since IPRMS1 and IPRSM2 both behave in a similar way, we only focus on IPRSM1 in this experiment.

For the LASSO model, we fix the data point $n = 1000$ and the dimension $d = 1000$. For the sparse logistic regression model, $n = 1000$ and $d = 200$. For the inner iteration stopping criterion, the sequence $\{\nu_k\}$ is set to be $1/n^\gamma$, with $\gamma$ varying from 1 to 2. Both primal residual $\beta\|y^k - y^{k+1}\|_2$ and dual residual $\frac{1}{\beta}\|\lambda^k - \lambda^{k+1}\|_2$ are ranging from $1 \times 10^{-1}$

Figure 3.1: LASSO model: evolution of the objective function, primal residual and dual residual.
Figure 3.2: Sparse logistic regression model: evolution of the objective function, primal redisplay and dual residual.

to $1 \times 10^{-4}$. The underdetermined relaxation factor $\alpha$ is set as 0.8 in this experiment. The timing performances are listed in Table 3.3 and Table 3.4.

For the LASSO model, as shown in Table 3.3, we notice that the timing performance is relatively more sensitive to exponential number $\gamma$ in $v_k$ than the sparse logistic regression model. A typical sequence of the norm of the error term $\|\xi^k\|$ has the following form: 0.77, 0.41, 0.25, 0.16, 0.10 0.067. This error term generated by the Jacobi method is relatively stable. This explains why the timing performance in Table 3.3 is relatively more sensitive to $\gamma$: as $\gamma$ increases, a few more iterations are
Table 3.3: Quantitatively time comparison (in seconds) between the inner and outer iterations stopping criterion for the LASSO model, $v_k = \frac{1}{n^\gamma}$

<table>
<thead>
<tr>
<th>$\gamma$</th>
<th>$1 \times 10^{-1}$</th>
<th>$1 \times 10^{-2}$</th>
<th>$1 \times 10^{-3}$</th>
<th>$1 \times 10^{-4}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1.06</td>
<td>1.45</td>
<td>1.73</td>
<td>2.01</td>
</tr>
<tr>
<td>1.1</td>
<td>1.16</td>
<td>1.57</td>
<td>1.86</td>
<td>2.15</td>
</tr>
<tr>
<td>1.2</td>
<td>1.23</td>
<td>1.67</td>
<td>1.97</td>
<td>2.26</td>
</tr>
<tr>
<td>1.4</td>
<td>1.37</td>
<td>1.89</td>
<td>2.32</td>
<td>2.59</td>
</tr>
<tr>
<td>1.6</td>
<td>1.48</td>
<td>2.13</td>
<td>2.69</td>
<td>2.95</td>
</tr>
<tr>
<td>1.8</td>
<td>1.59</td>
<td>2.35</td>
<td>3.03</td>
<td>3.35</td>
</tr>
<tr>
<td>2</td>
<td>1.65</td>
<td>2.47</td>
<td>3.32</td>
<td>4.02</td>
</tr>
</tbody>
</table>

For the sparse logistic regression model, as shown in Table 3.4, the timing performance is relatively less sensitive to the exponential number $\gamma$ in $v_k$. However, it is very sensitive to the stopping criterion used in the main iterative scheme. We further observe that one typical sequence of the norm of the error term $\|\xi_k\|$ has the following form: 144730, 390, 169, 87, 30, 0.97, 5.67 $\times 10^{-4}$. This indicates that Newton’s method for the inner iteration decreases the value of the error sequence significantly and thus the inner iteration stopping criterion sequence $\{v_k\}$ needs to be decreased much more significantly as well. This is the reason why the timing performance is

Table 3.4: Quantitatively time comparison (in seconds) between the inner and outer iterations stopping criterion for the sparse logistic regression model, $v_k = \frac{1}{n^\gamma}$

<table>
<thead>
<tr>
<th>$\gamma$</th>
<th>$1 \times 10^{-1}$</th>
<th>$1 \times 10^{-2}$</th>
<th>$1 \times 10^{-3}$</th>
<th>$1 \times 10^{-4}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1.69</td>
<td>2.71</td>
<td>3.88</td>
<td>5.91</td>
</tr>
<tr>
<td>1.1</td>
<td>1.69</td>
<td>2.71</td>
<td>3.88</td>
<td>5.91</td>
</tr>
<tr>
<td>1.2</td>
<td>1.71</td>
<td>2.73</td>
<td>3.90</td>
<td>5.93</td>
</tr>
<tr>
<td>1.4</td>
<td>1.71</td>
<td>2.74</td>
<td>3.91</td>
<td>5.94</td>
</tr>
<tr>
<td>1.6</td>
<td>1.73</td>
<td>2.75</td>
<td>3.91</td>
<td>5.97</td>
</tr>
<tr>
<td>1.8</td>
<td>1.73</td>
<td>2.75</td>
<td>3.91</td>
<td>6.00</td>
</tr>
<tr>
<td>2</td>
<td>1.75</td>
<td>2.78</td>
<td>3.91</td>
<td>6.05</td>
</tr>
</tbody>
</table>
less sensitive to the exponential number $\gamma$ in $\nu_k$: as $\gamma$ increases, the required iteration number in the inner iteration keeps the same.

For both models, we notice that as the outer iteration stopping criterion becomes smaller, the timing performance difference between $\gamma = 1$ and $\gamma = 2$ becomes larger. Based on this observation, we conjecture that when the outer iteration is far from convergence, the stopping criterion for the inner iteration does not play a very important role; when the primal and dual values in the outer iteration start to converge, the stopping criterion sequence $\{\nu_k\}$ becomes much more important and needs to be smaller in order to maintain the speed and efficiency of the algorithm. We confirm this by setting $\{\nu_k\}$ to be a sequence in the form: 100,\ldots,100,50,\ldots,50,1,\ldots,1. In this sequence, the first few numbers are large numbers 100 and 50, so that rough estimated values from Newton’s method are enough for the inner iteration; the last few numbers in this sequence are small, so more accurate estimated values from Newton’s method are needed as the algorithm goes. It only takes 1.22 second to finish the experiment.

In conclusion, the inexactness in the iterative scheme provides a method to efficiently implement the algorithms: in the first few iterations, a roughly estimated $\nu_k$ value for the inner iteration can be used in order to maintain the speed and efficiency; while the outer iteration is close to convergence, a smaller $\nu_k$ is required in order to obtain accurate result and maintain the convergence in the outer iteration.

3). Strictly contractive PRSM scheme requires an underdetermined relaxation factor $\alpha \in (0, 1)$ to guarantee the convergence. Therefore, we would like to investigate the acceleration effect of $\alpha$ in IPRSM1 as well. In Table 3.5 we show the sensitivity test on the underdetermined relaxation factor $\alpha$. In the test, we set dimension $d = 200$ and data point $m = 1000$, and focus on IPRSM1 for the sparse logistic regression model only. The convergence sequence $\{\nu_k\}$ and the stopping criterion are the same as in part 1). We notice that the strictly contractive constant $\alpha$ has a similar acceleration
Figure 3.3: Sensitivity test on the underdetermined relaxation factor $\alpha$ for IPRSM1

effect in IPRSM1 as in the original strictly contractive PRSM$^{[31]}$. We also plot the
computing time and the number of iterations with respect to different $\alpha$ values in
Figure 3.3. We conclude that a wide range of $\alpha$ values work for our example so that
this relaxation factor in our inexact methods can be easily chosen. We also notice
that $\alpha = 0.8$ is preferred for our tested cases; and thus we use this value in the first
two experiments.

Table 3.5: Sensitivity test on the underdetermined relaxation factor $\alpha$

<table>
<thead>
<tr>
<th>$\alpha$</th>
<th>iteration number</th>
<th>time (s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.1</td>
<td>50</td>
<td>5.42</td>
</tr>
<tr>
<td>0.2</td>
<td>27</td>
<td>3.14</td>
</tr>
<tr>
<td>0.3</td>
<td>20</td>
<td>2.34</td>
</tr>
<tr>
<td>0.4</td>
<td>17</td>
<td>1.98</td>
</tr>
<tr>
<td>0.5</td>
<td>16</td>
<td>1.87</td>
</tr>
<tr>
<td>0.6</td>
<td>14</td>
<td>1.63</td>
</tr>
<tr>
<td>0.7</td>
<td>13</td>
<td>1.53</td>
</tr>
<tr>
<td>0.8</td>
<td>13</td>
<td>1.51</td>
</tr>
<tr>
<td>0.9</td>
<td>15</td>
<td>1.77</td>
</tr>
<tr>
<td>0.95</td>
<td>23</td>
<td>2.68</td>
</tr>
<tr>
<td>1.0</td>
<td>54</td>
<td>5.81</td>
</tr>
</tbody>
</table>

3.7 Conclusion

In this chapter, we studied how to solve the resulting subproblems of the Peachman
Rachford Splitting Method (PRSM) subject to different inexactness criteria. We
demonstrate how to develop inexact versions of PRSM and show that such inexact criteria are most popular in the literature and easily implementable. The convergence of such inexact PRSMs have also been established. Numerical results on several different machine learning problems are also given to illustrate the effectiveness of our proposed inexact PRSMs. It deserves further research on embedding other inexactness criteria into the subproblems of PRSM and thus developing other algorithms of the same kind as those in this chapter.
Chapter 4

The New Graph Perceptron Algorithm

4.1 Introduction to Online Learning Framework

Online learning is a well-established technique used in machine learning when it is not computationally practical to train over the entire data set. It is also popular when the data set presents new patterns and requires dynamically updating in the algorithm. A typical feature of online learning is that its training phase and its testing phase are interleaved. Normally online learning algorithms update parameters after each input, and therefore are much faster than traditional batch optimization methods during each iteration. However, online learning algorithms are in general much noisy than batch optimization methods and as a trade-off, they are less accurate than batch methods [19]. Applications of online learning on large scale data sets include web ranking, spam filtering, and graph estimation [57, 33, 28].

This chapter is organized as follows. We start by presenting the general online learning framework and introducing the classical Perceptron algorithm used for binary online classification problems in Section 4.1. In Section 4.2, we begin with the setup of
the online graph estimation problem and introduce some useful background on online graph estimation including the well-known graph Perceptron algorithm. In Section 4.3, we derive the new graph Perceptron algorithm which applies a new kernel setting to graph learning problems. We also prove the regret bound and the mistake bound of the new graph Perceptron algorithm.

4.1.1 Classical Perceptron Algorithm

Throughout this chapter, we focus on binary classification. Assume each presented input \( x_t \in \mathbb{R}^n \) has a label \( y_t \in \{\pm 1\} \). We use a linear classifier \( w_t \) to predict the label of each input. Online learning operates in a sequence of rounds. In the first round, nature presents an input \( x_1 \), and we predict \( \hat{y}_1 \), then nature reveals the true label \( y_1 \); we update the weight vector; nature presents an input \( x_2 \), and we predict \( \hat{y}_2 \), then nature reveals the true label \( y_2 \), and so on. In round \( t \), the learning algorithm maintains a weight vector \( w_t \in \mathbb{R}^n \) as a linear classifier and predicts \( \hat{y}_t = \text{sgn}(w_t \cdot x_t) \). Next, the predictor receives \( y_t \) and if \( y_t \neq \hat{y}_t \), the algorithm updates the weight vector.

\( w_t \) has the following update rule

\[
\begin{align*}
    w_{t+1} &= \begin{cases} 
        w_t & \text{if } y_t w_t \cdot x_t > 0, \\
        w_t + \eta y_t x_t & \text{otherwise.}
    \end{cases} 
\end{align*}
\] (4.1.1)

Now we have the classical Perceptron algorithm for online learning [56]:
initialization $w_1 = 0$;

for $t = 1, 2, \cdots$ do

receive $x_t$;

predict $\hat{y}_t = \text{sgn}(w_t \cdot x_t)$;

receive correct label $y_t$;

if $y_t w_t \cdot x_t < 0$ then

$w_{t+1} = w_t + \eta y_t x_t$;

else

$w_{t+1} = w_t$;

end

Algorithm 2: Classical Perceptron Algorithm

There are two ways to measure the error of a prediction. First we can check whether $y_t = \hat{y}_t$ or not. However, this does not give a quantitative measure of the confidence of the linear predictor. The second way is using the hinge-loss function $\ell$ to measure the cost paid for inaccurate prediction in round $t$:

$$
\ell(w, (x_t, y_t)) = \begin{cases} 
0 & \text{if } y_t w \cdot x_t > 0, \\
1 - y_t w \cdot x_t & \text{otherwise.}
\end{cases}
$$ (4.1.2)

The regret is a very common measure when performing analysis of online learning algorithms. We would like to know how competitive our predictor vector $w$ is with respect to a fixed vector $u$. The regret of a predictor with respect to a fixed vector $u$ is defined as the cumulative difference between the loss incurred by the predictor and the loss incurred by the vector $u$. The regret $R$ of the algorithm relative to the vector $u$ when running on a sequence of $T$ examples is defined as
Definition 4.1.1.

\[ \mathcal{R}(u) = \sum_{t=1}^{T} \ell(w_t, (x_t, y_t)) - \sum_{t=1}^{T} \ell(u, (x_t, y_t)). \] (4.1.3)

It can be found in literature that the regret of the Perceptron learning algorithm is bounded by \[58\]

\[ \mathcal{R}(u) \leq \frac{1}{2\eta} \|u\|_2^2 + \frac{\eta}{2} T \sum_{t=1}^{T} \|x_t\|_2^2, \] (4.1.4)

Let \( M \) denote the total number of mistakes made by Perceptron and \( d = \max_t \|x_t\|_2 \), then \( M \) is bounded by \[51\]

\[ M \leq \sum_{t=1}^{T} \ell(u, (x_t, y_t)) + \|u\| \sqrt{d \sum_{t=1}^{T} \ell(u, (x_t, y_t)) + d\|u\|_2^2}. \] (4.1.5)

It is clear that if there exists a perfect \( u \) which never makes a mistake, then the total number of mistakes \( M \) is bounded by

\[ M \leq d\|u\|_2^2. \] (4.1.6)

The upper bound of the total number of mistakes is controlled by three factors: the loss generated by \( u \), the norm of \( u \) as well as the maximum value of the norm of \( x_t \). We use this Perceptron framework as a starting point for the Perceptron for online learning on graph estimation in the next section.

### 4.2 Online Learning on Graph Estimation

As we know, networks in the real world are usually massive; therefore, when the dimension of a network becomes sufficiently large, it is not possible to apply traditional batch learning algorithms on networks. We model the networks as graphs and focus on the application of online learning on graph estimation. In this section, we first set
up the problem and then review existing nonparametric methods used to solve these problems.

4.2.1 Graph Estimation Problem Setup

We are given an undirected graph $G = (V,E)$, consisting of the set $V$ of nodes and set $E$ of edges, which are unordered pairs of elements of $V$. Each node $v_i$ has a binary label $y_i \in \{\pm 1\}$. We are looking for a classifier function $f$ from $V$ to $\pm 1$.

The assumption behind graph estimation is called homophily \[62, 28\]. Homophily assumption says that if two nodes $v_i$ and $v_j$ are connected in the graph, then the labels of these two nodes have the tendency to be the same.

Online learning over a graph operates in a sequence of rounds. Consider the following online learning problem for predicting the label of each node: in the first round, nature presents a node $v_1$, we predict the label of the node $\hat{y}_1$, then nature reveals the true label $y_1$; we update the classifier function; nature presents the second node $v_2$ and we predict the label of the node $\hat{y}_2$, then nature reveals both the true label $y_2$ and the edge information between $v_2$ and node $v_1$, and so on. After the prediction, we update the classifier function based on the difference between the predicted label and the true label. At this point, we may use the known connectivity of the graph to improve the existing classifier function for future predictions. The goal is to predict the labels of incoming unlabeled nodes so that the total number of mistakes made is minimized.

We take the image segmentation problem as an example \[18\]. The user is given an image consisting of many pixels. An image is modeled as a graph and each pixel is therefore a node. Each pixel has a label, either “background” or “foreground”. Pixel connectivity is encoded in the graph’s topology. Homophily says that if two pixels are connected, then these two pixels tend to be either both “background” or both “foreground”. In each round, a pixel is presented to the user and the user is asked to
label the pixel either “background” or “foreground” based on the previous labelling as well as the currently known connectivity of the graph.

### 4.2.2 Graph Perceptron Algorithm

Traditional parametric online algorithms use vector-based data \cite{32, 71, 67}, and there is no edge information between the new incoming node and the currently known graph. Recently, many online learning methods take advantage of the link information to predict the label of the new node in a nonparametric way. The Graph Perceptron algorithm (GPA) \cite{37} is the application of the Perceptron algorithm to the graph estimation problem described in the previous subsection. We give a brief description of this algorithm.

We first give some definitions. Let the graph $G = (V, E)$ have a total of $n$ nodes and let $W \in \mathbb{R}^{n \times n}$ denote the adjacency matrix of the graph: $W_{ij} = 1$ when there is an edge between node $i$ and node $j$, and $W_{ij} = 0$ otherwise. Let $D = \text{diag}(d_1, \cdots, d_n)$, where $d_i$ is the degree of node $i$: $d_i = \sum_{j=1}^{n} W_{ij}$. The graph Laplacian $L$ is defined as

$$L = D - W. \quad (4.2.1)$$

We also define the coordinate spanning set \cite{37}

$$\mathcal{V}_L := \{v_i := L^+ e_i : i = 1, \cdots, n\}, \quad (4.2.2)$$

where $e_i$ is the $i$-th coordinate vector of $\mathbb{R}^n$, and $L^+$ is the pseudo-inverse of the graph Laplacian $L$. Each node $v_i$ has a label $y_i \in \{\pm 1\}$ and we are interested in predicting it. The vector $v_i$ will be used as the input to the graph Perceptron algorithm and it is similar to the data vector $x_i$ in the classical Perceptron algorithm. To be specific, in round $t$, we predict $\hat{y}_t = \text{sgn}(w_t \cdot v_t)$. The rest of the algorithm is the same as the classical Perceptron algorithm.
We formally present the graph Perceptron algorithm as

\begin{verbatim}
initialization $w_1 = 0$;
for $t = 1, 2, \cdots$ do
    receive $v_t$;
    predict $\hat{y}_t = \text{sgn}(w_t \cdot v_t)$;
    receive correct label $y_t$;
    if $\hat{y}_t = y_t$ then
        $w_{t+1} = w_t$;
    else
        $w_{t+1} = w_t + y_t v_t$;
end
\end{verbatim}

**Algorithm 3: Graph Perceptron Algorithm**

Due to the attractive theoretic guarantee as well as the competitive empirical performance, the graph Perceptron algorithm is the most popular existing online learning algorithm on graph estimation. There are many further theoretical developments of the graph Perceptron algorithm. For example, a new graph kernel matrix is introduced in order to improve the mistake bound [37]. Further analysis of the graph Perceptron algorithm can be found in the literature: the predicting vector $w$ can be calculated such that it minimizes the $p$-seminorm of a given linear mapping [35]; for large diameter graphs, a new kernel method called “spine” is used for online learning on graphs [36].

### 4.2.3 Learning with Local and Global Consistency

The graph Perceptron algorithm can be seen as a first-order online learning method and it has been shown that second-order online learning algorithms perform better than first-order online learning algorithms [61]. The intuition behind this argument
is that the classifier is smoother in second-order learning algorithms and therefore
the nodes which are labelled incorrectly by first-order methods could be possibly
smoothed by second-order methods. Learning with Local and Global Consistency
(LLGC) [70] is the most well-known second-order learning algorithm and it can be
seen as a ridge regression in the Hilbert space of functions defined on graphs. We
give a brief description of LLGC.

Let the graph $G = (V, E)$ have a total of $n$ nodes. Let $f$ be a classifier function
from $V$ to $\{\pm 1\}$, so $f(v_i)$ is the predicted label of the node $v_i$. Homophily assumption
can be formally written as [59]

$$\frac{1}{2} \sum_{i,j=1}^{n} (f(v_i) - f(v_j))^2 W_{ij} = f^T L f, \quad (4.2.3)$$

where $W_{ij}$ is the entry of the adjacency matrix $W$, $L$ is the graph Laplacian matrix
defined in (4.2.1) and $f = [f(v_1), \cdots, f(v_n)]^T$. LLGC changes the equation in (4.2.3)
to an objective function. It asks for a classifier function $f$ by solving the following
optimization problem [70]

$$\min_{f} \frac{1}{2} \|f - y\|_2^2 + \frac{\mu}{2} f^T L f, \quad (4.2.4)$$

where $y = [y_1, \cdots, y_n]^T$ is the true label vector and $\mu$ is a scalar regularization
parameter used to control the trade-off between smoothness of the classifier and the
loss of the prediction. A comprehensive review of online learning methods of graphs in
terms of theory and algorithms is given in [28], including an online version of LLGC.

A common theme of these papers is that they represent functions defined on the
graph by a Hilbert space associated with the graph Laplacian. There are two major
drawbacks to this approach. First, in order to calculate the graph Laplacian, we
need to know the entire graph structure in advance and this is often not possible in
practice. Second, if we do not know the graph in advance, then updating the graph
Laplacian for each online step is computationally intensive as it requires computing the inverse of a Laplacian matrix in each round. In other words, the existing literature about online learning on graphs are not really “online”.

As the dimension of the graph grows, retraining the model with the new incoming data together with the existing graph becomes almost impossible. It is desirable to develop a new method which can update the model incrementally while new data is coming and new relations are emerging. The limitations of existing algorithms motivate the development of new online learning algorithms. In next section, we change the graph Perceptron algorithm and make it suitable for the setting of online learning on a graph. The algorithm is computationally efficient and we can achieve a relatively low mistake bound similar to the classical Perceptron algorithm.

4.3 The New Graph Perceptron Algorithm

In this section, we derive a method called the new graph Perceptron algorithm. We start with some useful preliminaries.

4.3.1 Preliminaries

Let the graph $G = (V,E)$ have a total of $n$ nodes. We slightly change the definition of the adjacency matrix. Let $W \in \mathbb{R}^{n \times n}$ denote the adjacency matrix of the graph: for $i \neq j$, $W_{ij} = 1$ when there is an edge between node $i$ and node $j$, and $W_{ij} = 0$ otherwise. We further define the diagonal elements $W_{ii} = d$, where $d$ is the maximum degree over all nodes.

**Theorem 4.3.1.** $W$ is positive semi-definite.

*Proof.* Since we focus on undirected graphs, it is obvious that $W$ is symmetric. We claim that a hermitian matrix has real eigenvalues. To see this, we assume $A$ is hermitian so that $A^* = A$. Let $x$ be an eigenvector of $A$ and $\lambda$ be its corresponding
eigenvalue. We then have $Ax = \lambda x$ and $x^*A^* = \lambda^*x^*$. Multiplying $x$ from the right on the second equation, we have

$$
\lambda^*x^*x = x^*A^*x
= x^*Ax
= \lambda x^*x.
$$

Therefore $\lambda = \lambda^*$, so that a hermitian matrix has real eigenvalues.

Now we only need to show that the eigenvalues are all nonnegative. By Gershgorin circle theorem [38], every eigenvalue of $W$ lies within the Gershgorin discs with radius $R_i$ and centered at point $(d, 0)$ on the complex plane, where $R_i$ equals the sum of the absolute value of the non-diagonal entries in the $i$-th row. By definition, we know that $R_i \leq d$. Therefore, we conclude that each eigenvalue of $W$ is real and nonnegative, thus the matrix is guaranteed to be positive semi-definite.

The prediction in each round $t$ is determined by an internal function $f_t$. This internal function is also known as a hypothesis in literature [16, 58, 41]. Each internal function takes a node as input and outputs a real number, i.e., $f_t$ is a function from $V$ to $\mathbb{R}$. The sign of $f_t(v_t)$ indicates the actual binary prediction and the absolute value of $f_t(v_t)$ is the confidence of this prediction. The term $y_t f_t(v_t)$ is then positive whenever the prediction agrees with the true label. The hinge-loss function in this case is defined by

$$
\ell(f_t(v_t)) = \begin{cases} 
0 & \text{if } y_t f_t(v_t) \geq 0, \\
1 - y_t f_t(v_t) & \text{otherwise}.
\end{cases}
$$

In round $t$, the active set is a subset of $\{1, \cdots, (t-1)\}$. It consists of those indices for which the previous predictions made mistakes. Whenever the loss is zero, we
assume the predictive model is good and therefore we do not need these rounds to update the predictive model. We say that node $v_i$ is active in round $t$ if $i \in A_t$.

Let $\alpha = [\alpha_1, \cdots, \alpha_n]^T$. We consider the family of classifiers of the form:

$$f_{\alpha}(v_t) = \sum_{i \in V} \alpha_i W_{it}, \quad (4.3.3)$$

where $\alpha_1, \alpha_2, \ldots$ are any real valued weights. According to the definition of the adjacency matrix, $W_{it}$ is 1 when there is an edge between the existing node $i$ and the incoming node $t$, and is 0 otherwise.

Define an inner product between the functions $f_{\alpha}(v_t) = \sum_{i \in V} \alpha_i W_{it}$ and $f_{\beta}(v_t) = \sum_{i \in V} \beta_i W_{it}$ by

$$\langle f_{\alpha}, f_{\beta} \rangle = \sum_{i \in V} \sum_{j \in V} \alpha_i \beta_j W_{ij} = \alpha^T W \beta.$$

We claim that $K(v_i, v_j) = W_{ij}$ is a kernel. We first show that $f$ induces an inner product.

**Lemma 4.3.2.** The family of $f_{\alpha} : V \rightarrow \mathbb{R}$ with functions $f_{\alpha}(v_t) = \sum_{i \in V} \alpha_i W_{it}$ is a finite dimension vector space with inner product

$$\langle f_{\alpha}, f_{\beta} \rangle = \sum_{i \in V} \sum_{j \in V} \alpha_i \beta_j W_{ij}, \quad (4.3.4)$$

**Proof.** We have $\{f_{\alpha} : V \rightarrow \mathbb{R}\} \equiv \mathbb{R}^n$, so that the family of $f_{\alpha}$ is a vector space and $\langle f_{\alpha}, f_{\beta} \rangle = \alpha^T W \beta$, thus $\alpha^T W \beta$ is an inner product on $\mathbb{R}^n$. \qed

The inner product induces a norm defined by $\|f_{\alpha}\| = \langle f_{\alpha}, f_{\alpha} \rangle^{1/2}$ and a metric $\|f_{\alpha} - g_{\beta}\| = (\langle f_{\alpha}, f_{\alpha} \rangle - 2\langle f_{\alpha}, g_{\beta} \rangle + \langle g_{\beta}, g_{\beta} \rangle)^{1/2}$. Let $\mathcal{H}$ denote the set of all functions of the form in (4.3.3). Based on Lemma 4.3.2, $\mathcal{H}$ is a closed real inner product space. Since every Cauchy sequence in $\mathcal{H}$ has a limit that is also in $\mathcal{H}$, $\mathcal{H}$ is also complete. Thus, by definition, $\mathcal{H}$ is a Hilbert space.

104
**Theorem 4.3.3.** Let $K(\cdot, \cdot) : V \times V \to \mathbb{R}$ be a kernel function. We define $K(\cdot, \cdot) = W$. (this means that for any node $v_s$ and $v_t$, $K(v_s, v_t) = W_{st}$), then $K(\cdot, \cdot)$ is a reproducing kernel of Hilbert Space $\mathcal{H}$.

**Proof.** We define the evaluation functional as $L_v(f_\alpha) := f_\alpha(v)$. The evaluation functional $L_v$ is clearly bounded. The reproducing property requires that for an arbitrary node $v_s$, $\langle f_\alpha, W_s \rangle = f_\alpha(v_s) = L_v(f_\alpha), \forall f_\alpha \in \mathcal{H}$. We verify this property by the following argument: $\forall f_\alpha \in \mathcal{H}$, given an arbitrary node $v_t$, we have

$$\langle f_\alpha(v_t), W_{st} \rangle = \langle \sum_{i \in V} \alpha_i W_{it}, W_{st} \rangle = \sum_{i \in V} \alpha_i W_{is} = f_\alpha(v_s).$$

Since our kernel has the reproducing property and it spans $\mathcal{H}$, we conclude that $\mathcal{H}$ is a Reproducing Kernel Hilbert Space (RKHS).

We now restrict the kernel functions in (4.3.3) to the functions that are defined over the current active set $A_t$. The function $f_t$ that is used in round $t$ is defined by

$$f_{\alpha,t}(v_t) = \sum_{i \in A_t} \eta y_i W_{it}, \quad (4.3.5)$$

where $\eta$ is a scalar parameter, $A_t$ is the current active set, and $y_i$ is the true label of the node $v_i$. In this case, $\alpha_i = \eta y_i$ if $v_i \in A_t$ and $\alpha_i = 0$ otherwise. Therefore, the function $f_{\alpha,t}$ is still defined over all nodes in $V$ and takes value in $\mathbb{R}$.

As we use the label $y_i$ multiplying a parameter $\eta$ as the weight assigned to each active node, it is very similar to the classical Perceptron algorithm. This is the reason that we define the algorithm generated from the classifier in the form (4.3.5) as the new graph Perceptron algorithm.
In order to simplify the notation, we drop the subscript vector $\alpha$ from $f_{\alpha}$ in the following analysis. The new graph Perceptron initializes $A_1$ with an empty set and assumes that $f_1$ has value 0.

### 4.3.2 The New Graph Perceptron Algorithm

**Data:** Graph $G(V, E)$

**Result:** Prediction function $f$

Initialization $A_1 = \emptyset$, $f_1 = 0$;

for $t = 1, 2, \cdots$ do

- receive a vertex $v_t$ and edge information between $v_t$ and the existing graph;
- predict $\hat{y}_t = \text{sgn}(f_t(v_t))$;
- receive the true label $y_t$;

if $y_t f_t(v_t) > 0$ then

- $A_{t+1} = A_t$;
- $f_{t+1} = f_t$;

else

- $A_{t+1} = A_t \cup \{t\}$;
- $f_{t+1}(v_{t+1}) = \sum_{i \in A_{t+1}} \eta y_i W_{it+1}$ for node $v_{t+1}$;

end

**Algorithm 4:** The New Graph Perceptron Algorithm

Each function $f_t$ has a distance between itself and a fixed function $g \in \mathcal{H}$. The first lemma is used to show that whenever the new graph Perceptron makes a prediction mistake, the updated function $f_{t+1}$ moves closer to a fixed function $g$.

**Lemma 4.3.4.** Let $(v_t, y_t)$ be an example node, where $v_t \in V$ and $y_t \in \{\pm 1\}$. Let $f$, $g$ be any functions in $\mathcal{H}$, and define $f' = f + \eta y_t W_t$. (This means for an arbitrary
node $v_i$, we have $f'(v_i) = f(v_i) + \eta y_t W_t$. It holds that

$$\|f - g\|^2 - \|f' - g\|^2 = -2\eta y_t f(v_t) + 2\eta y_t g(v_t) - \eta^2 d. \quad (4.3.6)$$

**Proof.** Since all these functions are defined on a finite dimension vector space, therefore the calculation rules are similar to the rules on $\mathbb{R}^n$. Since $W$ is symmetric, we have $W_t = W$. We expand the term $\|f' - g\|^2$ in (4.3.6) and have

$$\|f' - g\|^2 = \| f - g + \eta y_t W_t \|^2 \quad (4.3.7)$$

By the reproducing property, we know that $\langle f, W_t \rangle = f(v_t)$, $\langle g, W_t \rangle = g(v_t)$ and $\langle W_t, W_t \rangle = W_{tt}$. Therefore, (4.3.7) becomes

$$\|f - g + \eta y_t W_t\|^2 = \| f - g \|^2 + 2\eta y_t \langle f, W_t \rangle - 2\eta y_t \langle g, W_t \rangle + \eta y_t^2 W_{tt}$$

$$= \| f - g \|^2 + 2\eta y_t f(v_t) - 2\eta y_t g(v_t) + \eta^2 y_t^2 W_{tt}$$

$$= \| f - g \|^2 + 2\eta y_t f(v_t) - 2\eta y_t g(v_t) + \eta^2 d. \quad (4.3.8)$$

Finally, we see that

$$\|f - g\|^2 - \|f' - g\|^2 = \| f - g \|^2 - \| f + \eta y_t W_t - g \|^2$$

$$= -2\eta y_t f(v_t) + 2\eta y_t g(v_t) - \eta^2 d. \quad (4.3.9)$$

$\Box$
4.3.3 Regret Bound and Mistake Bound

The term \( \|f - g\|^2 - \|f' - g\|^2 \) is a very important term when doing online algorithm analysis [13, 16, 34, 42] as it measures how much the updated function moves closer to a fixed function \( g \). We will denote it by \( \Delta \) in the later analysis.

Similar to the definition from (4.1.4), we define the regret \( R \) of the new graph Perceptron as

**Definition 4.3.5.**

\[
R = \sum_{t=1}^{T} \ell(f_t(v_t)) - \inf_{g \in H} \sum_{t=1}^{T} \ell(g(v_t)).
\]

(4.3.10)

At this stage, we are ready to prove the following regret bound and this regret bound reconciles with the regret bound of the classical Perceptron Algorithm [58].

**Theorem 4.3.6.** Let \((v_1, y_1), \ldots, (v_T, y_T)\) be a sequence of example nodes. Let \( M \) be the total number of mistakes made by the new graph Perceptron, then the regret made by Algorithm 4 is bounded by

\[
R \leq \frac{1}{2} \left( \frac{\|g\|^2}{\eta} + M \eta d \right),
\]

(4.3.11)

where \( g = \arg \min_{g \in H} \sum_{t=1}^{T} \ell(g(v_t)) \).

**Proof.** Let \((v_t, y_t)\) be an example node, where \( v_t \in V, y_t \in \{1\} \) and \( y_t f_t(v_t) < 1 \). Assume in round \( t \), the predictor receives \((v_t, y_t)\). Let \( \ell_{g_t} = \ell(g(v_t)) \), \( \ell_{f_t} = \ell(f_t(v_t)) \), and let \( \Delta_t = \|f_t - g\|^2 - \|f_{t+1} - g\|^2 \). By Lemma 4.3.4 we have

\[
\Delta_t = \|f_t - g\|^2 - \|f_{t+1} - g\|^2
\]

\[
= -2\eta y_t f_t(v_t) + 2\eta y_t g(v_t) - \eta^2 d
\]

\[
\geq -2\eta y_t f_t(v_t) - 2\eta (\ell_{g_t} - 1) - \eta^2 d
\]

\[
= -2\eta (1 - \ell_{f_t}) - 2\eta (\ell_{g_t} - 1) - \eta^2 d
\]

\[
= 2\eta (\ell_{f_t} - \ell_{g_t}) - \eta^2 d.
\]

(4.3.12)
We sum $\Delta_t$ over $t = 1, \cdots, T$ and by noticing that $\Delta_t = 0$ when $y_t f_t(v_t) \geq 0$, we get
\[
\sum_{t=1}^{T} \Delta_t \geq 2\eta R - M\eta^2 d. \quad (4.3.13)
\]

On the other hand, we need to find the upper bound of the term $\sum_{t=1}^{T} \Delta_t$. We notice that the term $\sum_{t=1}^{T} \Delta_t$ is a telescoping sum, so
\[
\sum_{t=1}^{T} \Delta_t = \|f_1 - g\|^2 - \|f_{T+1} - g\|^2. \quad (4.3.14)
\]

By noticing that $f_1$ has value 0 and $\|f_{T+1} - g\|^2 \geq 0$, we know that $\sum_{t=1}^{T} \Delta_t \leq \|g\|^2$. Finally we have
\[
\|g\|^2 \geq 2\eta R - M\eta^2 d. \quad (4.3.15)
\]

The theorem is proved.

\[\square\]

This regret bound is very similar to the regret bound of the classical Perceptron algorithm in (4.1.3). Now we derive the mistake bound of the new graph Perceptron algorithm.

**Theorem 4.3.7.** Let $(v_1, y_1), \cdots, (v_T, y_T)$ be a sequence of example nodes and $g = \arg \min_{g^* \in H} \sum_{t=1}^{T} \ell(g^*(v_t))$, then the total number of mistakes $M$ made by the new graph Perceptron is bounded by
\[
M \leq \sum_{t=1}^{T} \ell(g(v_t)) + \|g\| \sqrt{d \sum_{t=1}^{T} \ell(g(v_t)) + d\|g\|^2}. \quad (4.3.16)
\]

**Proof.** We know that the loss function is larger than 1 when a mistake is made, so we have $\sum_{t=1}^{T} \ell(f_t(v_t)) \geq M$. We obtain
\[
M - \sum_{t=1}^{T} \ell(g(v_t)) \leq \frac{1}{2} \left(\frac{\|g\|^2}{\eta} + M\eta d\right). \quad (4.3.17)
\]
Setting $\eta = \frac{\|g\|}{\sqrt{Md}}$ and rearranging (4.3.17), we obtain

$$M - \sqrt{Md}\|g\| - \sum_{t=1}^{T} \ell(g(v_t)) \leq 0. \quad (4.3.18)$$

This is a quadratic inequality in $\sqrt{M}$ of the form $M - b\sqrt{M} - c \leq 0$, where $b = \sqrt{d}\|g\|$ and $c = \sum_{t=1}^{T} \ell(g(v_t))$. We can easily see that

$$\sqrt{M} \leq \frac{b + \sqrt{b^2 + 4c}}{2}. \quad (4.3.19)$$

Therefore, we have

$$M \leq \frac{b^2}{2} + c + \frac{b\sqrt{b^2 + 4c}}{2} \leq b^2 + c + b\sqrt{c} \quad (4.3.20)$$

$$= d\|g\|^2 + \sum_{t=1}^{T} \ell(g(v_t)) + \|g\| \sqrt{d \sum_{t=1}^{T} \ell(g(v_t))}.$$ 

□

If there exists a best function $g$ such that $\sum_{t=1}^{T} \ell(g(v_t))$ is zero for all $t$, then we have

$$M \leq d\|g\|^2. \quad (4.3.21)$$

As shown in Theorem 4.3.1, $W$ is positive semi-definite since this matrix is dominated by its diagonal entries $d$, which are defined as the maximum degree over all nodes. Therefore, the mistake bound of the new graph Perceptron algorithm is closely related to this quantity $d$ as well as the norm of the best function $g$ in the vector space. Of course, the value $d$ can be decreased as long as the kernel matrix is still positive semi-definite. In order to minimize the total number of mistakes made by
the algorithm, the value $d$ must be small. This is achieved whenever the graph is sparse and there is no obvious hub structure.

We compare this bound with the mistake bound of the classical Perceptron algorithm in (4.1.6) which has the form $d\|u\|_2^2$. Recall in (4.1.6), $d = \max_t \|x_t\|_2^2$ and $\|u\|$ is the norm of the best weight vector in Perceptron algorithm. We observe that both $\|g\|$ and $\|u\|$ represent the norms of the best function in different spaces: $\|u\|$ is in $\mathbb{R}^n$ and $\|g\|$ is in the Hilbert space defined in (4.3.3). The quantity $d$ is closely related to the data points in both algorithms: the one in the classical Perceptron algorithm is the maximum norm of the input vector, and the other is the maximum degree over all input nodes. Therefore, we conclude that we have found meaningful regret bound and mistake bound of the new graph Perceptron algorithm under the assumption that the graph is sparse.
Chapter 5

Conclusions and Future Work

This thesis presented a variety of optimization methods used for machine learning problems. In Chapter 1, we studied the parametric simplex method and showed that it can be customized for specific learning problems by finding the entire solution path in one shot when the case is linear. The features of the parametric simplex method allow us to start with the largest value of the parameter and decrease it to zero. The advantage of the parametric simplex method over other linear programming methods is that it solves the entire regularization path in terms of the parameter. In addition, we proved that both primal and dual solution paths of Dantzig selector using the parametric simplex method are guaranteed to be sparse. We expect this method advances the field of $\ell_1$ type learning as well as compressed sensing problems. In Chapter 2, we introduced our R package `fastclime` which takes advantage of the parametric simplex method and solves CLIME, Dantzig selector as well as generic and parametric linear programming problems. In Chapter 3, we explored the inexact version of PRSM method, which is a variation of the famous Douglas-Rachford Alternating Direction Method of Multiplier algorithm. We first added one more dual update in the ADMM subproblems to form PRSM framework. Next, we added proximal regularization terms to the PRSM’s subproblems and accordingly established
an algorithmic framework of the inexact version of PRSM. We proved the $O(1/t)$ convergence rate of this algorithm in both ergodic and non-ergodic cases. Finally, in Chapter 4 we introduced the new graph Perceptron algorithm, a kernel-based online learning method used on a graph for binary node classification when the network is huge. Instead of using the graph Laplacian as the kernel matrix, we took advantage a novel kernel matrix which does not require intensive computation during each iteration. A relative low regret bound and a mistake bound were given and they are in the same form as the bounds achieved by the classical Perceptron algorithm.

In the future, there are several different directions which are worth to explore. The role of the parametric simplex method in machine learning may lead to many new research directions. For a given statistical learning problem, we would like to see the relation between statistical properties of this problem and the optimization method. For example, once the solution path indexed by the parameter is given, we are curious whether it is possible to pick a parameter in order to directly satisfy some certain statistical properties. For the theoretical justification, the sparsity of the primal and dual solution paths needs to be proved for each $\ell_1$ learning problem. We are dedicated to maintaining our R package fastclime on CRAN and extending its application to new learning problems. Continuing the topic of nonparametric graph estimation by the new graph Perceptron, we hope that it can be extended to semi-parametric mechanism. Current algorithm only utilizes the link information, so we aim to study the online algorithm which can take advantage of both the link and the content information of the graph. We believe that the semi-parametric methods would be more powerful than the nonparametric methods used for online graph learning. Another problem arises from the size of the graph. Kernel-based learning algorithms usually suffer from the lack of memory required to store the existing data. The incoming nodes and the functions may keep growing indefinitely as the algorithm updates. The running time is another issue as the network grows. Our next goal
is to derive and analyze an online prediction algorithm which resolves the problems when the network is huge and a predefined budget size of the active set is enforced. The idea is simple. Whenever the size of the active set is below a threshold and the Perceptron algorithm suffers from some loss, we add that node to the active set. The active set grows until it eventually reaches a threshold. From this time, whenever the algorithm suffers from a loss, it first adds a node to the active set, and then it reduces the size of the active set by removing a node from the active set.
Bibliography


119