Iterative Methods for Nonnegative and Box Constrained Least Squares Problems and Their Applications

Ning Zheng

Doctor of Philosophy

Department of Informatics
School of Multidisciplinary Sciences
SOKENDAI (The Graduate University for Advanced Studies)

Iterative Methods for Nonnegative and Box Constrained Least Squares Problems and Their Applications

by

Ning Zheng

Dissertation

submitted to the Department of Informatics

School of Multidisciplinary Sciences

in partial fulfillment of the requirements for the degree of

Doctor of Philosophy

SOKENDAI (The Graduate University for Advanced Studies)

March 2017

Supervisor

Professor Ken HAYAMI
National Institute of Informatics
The Graduate University for Advanced Studies

Subadvisors

Professor Takeaki UNO
National Institute of Informatics
The Graduate University for Advanced Studies

Doctor Nobutaka ONO
National Institute of Informatics
The Graduate University for Advanced Studies

Other Examiners

Professor Takashi TSUCHIYA
National Graduate Institute for Policy Studies

Doctor Hironobu GOTODA
National Institute of Informatics
The Graduate University for Advanced Studies

“And ye shall know the truth, and the truth shall make you free."

— John 8:32

“Because strait is the gate, and narrow is the way, which leadeth unto life, and few there be that find it."

— Matthew 7:14

ACKNOWLEDGMENTS

First, I would like to thank my supervisor, Professor Ken Hayami, for the patient guidance and extraordinary inspiration he has provided throughout many years. I have been extremely lucky to have a supervisor who cared so much about my research, and who provided insightful and invaluable advice to my questions and queries so promptly.

I am thankful to my subadvisors Professor Takeaki Uno and Dr. Nobutaka Ono, as well as other thesis examiners Professor Takashi Tsuchiya and Dr. Hironobu Gotoda. During the second intermediate presentation, Dr. Ono suggested to apply my proposed methods to solve nonnegative matrix factorization, which widen my research from various perspectives and eventually became a new chapter in this thesis. Professors Tsuchiya and Uno advised me to design the algorithms from the optimization viewpoint. In addition, Dr. Gotoda gave me a lots of insightful comments and encouragement during the intermediate presentations and final examinations.

I also would like to thank Professor Jun-Feng Yin, Drs. Keiichi Morikuni and Kota Sugihara for their valuable remarks and discussions. Dr. Morikuni's advices had greatly improved the first version of the paper on nonnegative constrained least squares problem.

Many thanks to National Institute of Informatics, for the financial support, and the great efforts from International Affairs and Education Support Team. Doing research here is one of my best memories in my life.

Last but not the least, I must express my gratitude to my family, for their continued support and encouragement.

ABSTRACT

In this thesis, we mainly focus on the iterative methods for the large sparse nonnegative constrained least squares (NNLS) and box constrained least squares (BLS) problems. The new iterative methods, especially based on the modulus variable transformation and active set strategy, are discussed and constructed. The theoretical analysis for the convergence of the proposed methods are established. Also, numerical experiments show that the proposed methods outperform previous methods in terms of computational time and storage requirement under suitable conditions with reasonable parameters.

First, we briefly review numerical methods for the solution of NNLS and BLS problems. Moreover, the advantages and the disadvantages of different methods are discussed, which motivate one to design new iterative algorithms for large sparse problems.

Secondly, we consider the solution of the general NNLS problem. A new iterative method is proposed which uses the CGLS method for the inner iteration and the modulus iterative method for the outer iteration to solve the linear complementarity problem resulting from the Karush-Kuhn-Tucker condition of the NNLS problem. Theoretical convergence analysis including the optimal choice of the parameter matrix is presented for the proposed method. In addition, the method can be further enhanced by incorporating the active set strategy, which contains two stages: the first stage consists of modulus iterations to identify the active set, while the second stage solves the reduced unconstrained least squares problems only on the inactive variables, and projects the solution into the nonnegative region. Numerical experiments show the efficiency of the proposed methods compared to projection gradient-type methods with less iteration steps and CPU time.

Thirdly, we consider the solution of large sparse BLS problems using a new class of iterative methods based on modulus transformation, which converts the solution of the BLS into a sequence of unconstrained least squares problems. The inner unconstrained least squares problems are solved by CGLS method for each outer iteration. We also discuss the solution of saddle point inner systems, and the choice of the parameter matrix. Numerical experiments show the efficiency of the proposed methods in comparison of the gradient projection methods.

Fourthly, we consider the solution of the nonnegative constrained ill-posed problem, especially the image restoration problem. The problem can be formulated as the NNLS problem, which can be solved by the methods proposed previously. Meanwhile, we consider the discrepancy principle and Tikhonov regularization. Computed examples illustrate the performances of these methods.

Fifthly, we consider the solution of nonnegative matrix factorization (NMF), which is a low rank matrix approximation problem with nonnegative constraints, using a new alternating least squares method by utilizing the modulus method to solve the nonnegative constrained least squares problem in each iteration. We review some existing methods for the solution of NMF, then construct the modulus method for NMF and compare the proposed method with the existing methods numerically on the synthetic data and ORL face image data.

Finally, the Anderson extrapolation is used to accelerate the algorithms for NMF. We proposed a new Anderson acceleration technique that can flexibly accelerate the fixed point iteration sequences, instead of accelerating the iteration sequence for every step. The numerical experiments show that the Anderson acceleration not only accelerates the classical stationary iterations for linear equations, but also outperform the previous methods on nonlinear problems like linear complementarity problem, NNLS and NMF with less iterations and CPU time.

CONTENTS

LIST OF FIGURES

LIST OF TABLES

CHAPTER 1

INTRODUCTION

1.1 Backgrounds

Linear least squares is an approach fitting a mathematical, statistical or engineering model to data in cases where the idealized value of the model at any data point is expressed linearly with respect to the unknown parameters of the model. It is known by different names in different academic areas. For examples [67], mathematicians regard the least squares problem as finding the closest point in a given subspace to a given point in a function space; Statisticians use linear regression which arises as a particular form of regression analysis to describe the model; Engineers may reach the problem as parameter estimation, filtering, or process identification. Linear least squares problem can be used to summarize the data, to predict unobserved values from the same system, to understand the mechanisms that may underlie the system, and to approximate the nonlinear least squares problems with linearization.

The problem of nonnegative constrained least squares (NNLS) is a constrained version of the least squares problem where the variables are not allowed to be negative [8]

$$\min_{x \in \mathbf{R}^n} \|Ax - b\|_2 \quad \text{subject to} \quad x \geq \mathbf{0}. \tag{1.1}$$

Another generalization of NNLS is box constrained least squares (BLS), with simultaneous upper and lower bounds on the variable [8]

$$\min_{x \in \mathbf{R}^n} \|Ax - b\|_2 \quad \text{subject to} \quad l \leq x \leq u. \tag{1.2}$$

Here, we assume that $A \in \mathbf{R}^{m \times n}$, $b \in \mathbf{R}^m$, $m \geq n$ and the inequalities are to be interpreted componentwise. The rank-deficient case is allowed, when the equality in $\text{rank} A \leq \min(m,n)$ does not hold. Not only do the NNLS and BLS problems arise in many scientific computing and engineering applications [8], e.g., image restoration [78], reconstruction problems in geodesy [19] and tomography, contact problems for mechanical systems [57], and the modeling of ocean circulation, but it is even argued that any minimization problem becomes realistic only when its variables are constrained within meaningful intervals [19].

The linear least squares problem, NNLS problem and BLS problem are convex and the global minimums can be computed. They are strictly convex, where the global minimum is unique, if and only if A is full column rank. There are two main challenges for the solution of NNLS and BLS. One challenge is that there is a growing tendency that the problems arising from the real applications are larger scale and sparser, which makes the previous sophisticated algorithms not efficient. It goes without saying that the new algorithms are desperately needed for the fast and accurate solution of the large problems. Another challenge is that even if there exist the efficient solvers for NNLS and BLS, they may not be efficient and suitable to apply to the nonconvex problems, e.g., nonnegative matrix factorization (NMF) problem, which treats the NNLS problem as subproblems. Therefore, a new framework for the algorithm is required.

1.2 Motivations

Algorithms for the solution of the unconstrained linear least squares problem

$$\min_{x \in \mathbf{R}^n} \|Ax - b\|_2 \tag{1.3}$$

fall into two classes: direct methods, which are usually based on some matrix factorizations and may not be so practical when the matrix A is large, sparse and does not have a special structure, and iterative methods, among which the (preconditioned) CGLS

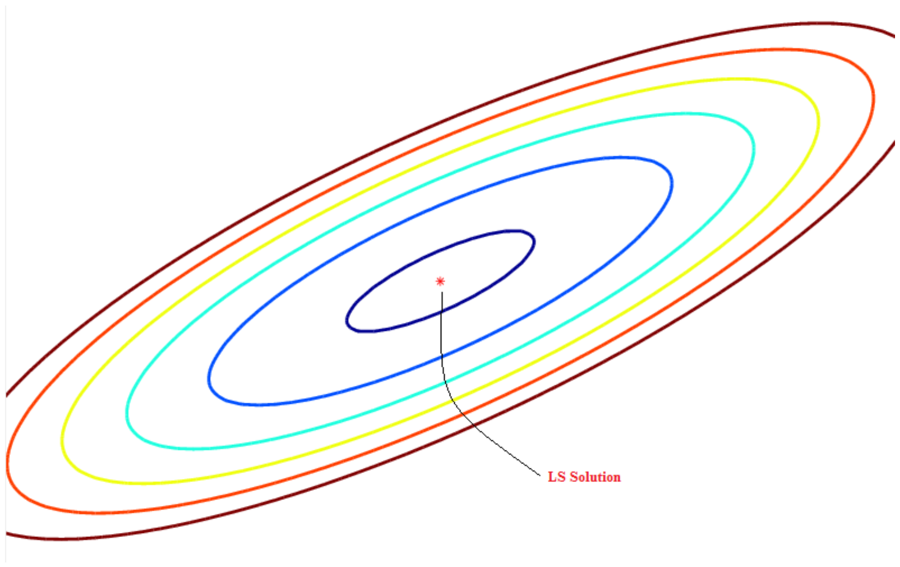


Figure 1.1: The contour of the least squares objective function and the solution of the least squares problem.

method [8], which is mathematically equivalent to the conjugate gradient (CG) method applied to the normal equation

$$A^{\mathrm{T}}Ax = A^{\mathrm{T}}b, \tag{1.4}$$

and the BA-GMRES (left preconditioned GMRES) method [53, 72] play important roles. However, the approximate solutions determined by the above methods are not guaranteed to satisfy the nonnegative constraints in (1.1). Therefore, special techniques must be added to the algorithms that handle the status of variables with respect to their nonnegativity.

1.3 Contributions

The main contribution of this thesis is a new class of inner outer iterative methods for nonnegative constrained least squares (NNLS) problem (1.1) was proposed based on the modulus transformation for the nonnegative variables. Thus, the solution of the NNLS problem (1.1) can be transformed into the solution of a sequence of unconstrained least squares problems. Similar techniques are also used for constructing the algorithms for box constrained least squares problem, nonnegative ill-posed problem and nonnegative

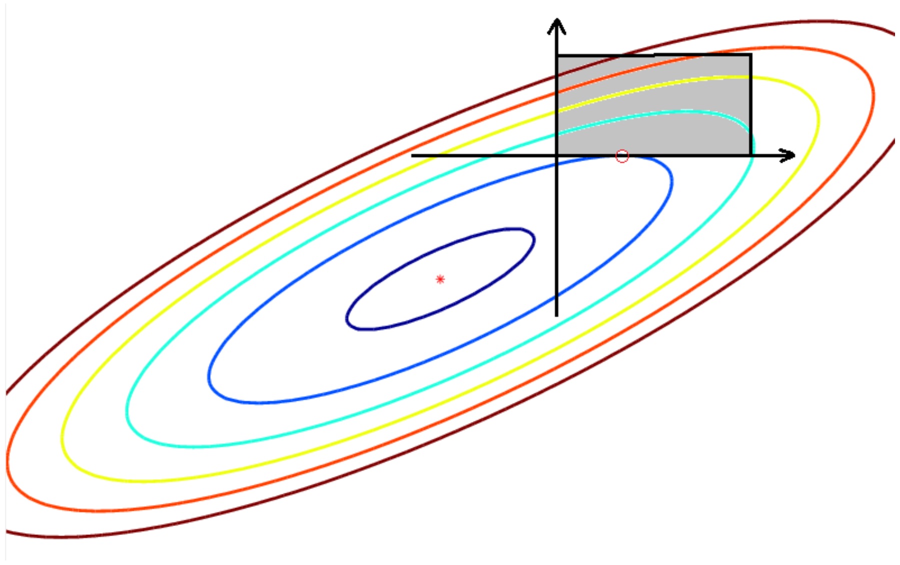


Figure 1.2: The contour of the least squares objective function and the solution of the box constrained least squares problem.

matrix factorization. Theoretical convergence analysis was presented when the inner system is solved either exactly or iteratively, and the choice of the parameter matrix was discussed for the proposed methods. Moreover, we proposed a two-stage hybrid modulus algorithm by incorporating the active set strategy, which contains two stages where the first stage consists of modulus iterations to identify the active set, while the second stage solves the reduced unconstrained least squares problems only on the inactive variables, and projects the solution into the nonnegative region. Numerical experiments show the efficiency of the proposed modulus methods compared to projection gradient-type methods with less iteration steps and CPU time for full column rank and rank deficient overdetermined NNLS problems. The modulus method is not only more efficient for identifying a suitable active set, but also outperforms projection gradient-type methods with less iteration steps and CPU time when the coefficient matrix has ill-determined rank with large condition number and the singular values cluster near zero.

The idea and techniques developed above can be further used for the solution of the box constrained least squares problem, the nonnegative matrix factorization, and the nonnegative ill-posed problem. For the solution of the different problems, we design the algorithm making use of the characteristic of each problem and exploit its optimal performance by choosing the parameters. We also applied a flexible

Anderson acceleration to the proposed algorithms, to further accelerate the convergence performance.

1.4 Outline

The rest of the thesis is organized as follows.

In Chapter 2, we first briefly review previous numerical methods for the solution of NNLS and BLS problems. Moreover, the advantage and the disadvantage of different methods are discussed and compared, which motivates one to design new iterative algorithms for large sparse problems.

In Chapter 3, we consider the solution of general NNLS problems. A new iterative method is proposed which uses the CGLS method for the inner iteration and the modulus iterative method for the outer iteration to solve the linear complementarity problem resulting from the Karush-Kuhn-Tucker condition of the NNLS problem. Theoretical convergence analysis including the optimal choice of the parameter matrix is presented for the proposed method. In addition, the method can be further enhanced by incorporating the active set strategy, which contains two stages: the first stage consists of modulus iterations to identify the active set, while the second stage solves the reduced unconstrained least squares problems only on the inactive variables, and projects the solution into the nonnegative region. Numerical experiments show the efficiency of the proposed methods compared to projection gradient-type methods with less iteration steps and CPU time.

In Chapter 4, we consider the solution of large sparse BLS problems using a new class of iterative methods based on modulus transformation, which converts the solution of the BLS into a sequence of the unconstrained least squares problems. The inner unconstrained least squares problems are solved by CGLS method for each outer iteration. We also discuss the solution of saddle point inner systems, and the choice of the parameter matrix. Numerical experiments show the efficiency of the proposed methods in comparison of the gradient projection methods.

In Chapter 5, we consider the solution of the nonnegative constrained ill-posed problem, especially the image restoration problem. The problem can be formulated as the NNLS problem, which can be solved by the methods proposed previously. Meanwhile, we consider the discrepancy principle and Tikhonov regularization. Computed examples illustrate the performances of these methods.

In Chapter 6, we consider the solution of nonnegative matrix factorization (NMF), which is a low rank matrix approximation problem with nonnegative constraints, using a new alternating least squares method by utilizing the modulus method to solve the nonnegative constrained least squares problem in each iteration. We review some existing methods for the solution of NMF, then construct the modulus method for NMF and compare the proposed method with the existing methods numerically on the synthetic data and ORL face image data.

In Chapter 7, the Anderson extrapolation is used to accelerate the algorithms for NMF. We proposed a new Anderson acceleration technique that can flexibly accelerate the fixed point iteration sequences, instead of accelerating the iteration sequence for every step. The numerical experiments show that the Anderson acceleration not only accelerate the classical stationary iterations for linear equations, but also outperform the previous methods on nonlinear problems like linear complementarity problem, NNLS and NMF with less iterations and CPU time.

Finally in Chapter 8, the concluding remarks are given and several ongoing works are discussed.

CHAPTER 2

PREVIOUS WORK

In this chapter, we briefly review almost all the previous work on the iterative solution of NNLS and BLS that arise from practical problems including contact and friction problems in rigid body mechanics, journal bearing lubrication, flow through a porous medium, and elastic-plastic torsion problems [77]. Some of the methods were originally proposed for the solution of quadratic programming with nonnegative or box constraints, which can be easily applied to solve NNLS and BLS. By comparing these methods in terms of the advantages and disadvantages, we show the reason why a new strategy is needed.

In Section 2.1, we propose an important theorem describing several equivalent conditions of NNLS and BLS. These equivalent conditions are the key to deriving the fixed-point iteration, e.g., the projection gradient method which will be introduced in Section 2.2. In Sections 2.3 and 2.4, the active set methods and the interior point methods are reviewed, respectively. In Sections 2.5, the reflective Newton method and the recent developed flexible Krylov subspace method are introduced. Finally in Section 2.6, we compare the previous methods and mention the necessity of a new strategy.

2.1 Equivalent Conditions

In this section, we list several equivalent conditions of BLS, which can be naturally extended for NNLS.

Let the quadratic objective function be

$$q(x) \equiv \frac{1}{2}\|Ax-b\|_2^2 = \frac{1}{2}x^{\mathrm{T}}(A^{\mathrm{T}}A)x - (A^{\mathrm{T}}b)^{\mathrm{T}}x + \frac{1}{2}b^{\mathrm{T}}b. \tag{2.1}$$

The gradient function of $q(x)$ is

$$\nabla q(x) = A^{\mathrm{T}}Ax - A^{\mathrm{T}}b.$$

The equivalent formulations of the BLS problem (1.2) can be established through the Karush-Kuhn-Tucker (KKT) optimality conditions, which is shown in the following theorem.

Theorem 2.1.1. *For the BLS problem* (1.2)*, the following assertions are all equivalent.*

(i) *x^* is a solution of the BLS problem* (1.2)*;*

(ii) *x^* is a solution of the linear complementarity problem (LCP)*

$$A^TAx - A^Tb - \begin{bmatrix} I \\ -I \end{bmatrix}^T \begin{bmatrix} \lambda_1 \\ \lambda_2 \end{bmatrix} = 0, \quad \begin{bmatrix} \lambda_1 \\ \lambda_2 \end{bmatrix} \geq 0,$$
$$\begin{bmatrix} I \\ -I \end{bmatrix} x - \begin{bmatrix} l \\ -u \end{bmatrix} \geq 0 \quad \text{and} \quad \begin{bmatrix} \lambda_1 \\ \lambda_2 \end{bmatrix}^T \left(\begin{bmatrix} I \\ -I \end{bmatrix} x - \begin{bmatrix} l \\ -u \end{bmatrix} \right) = 0 \tag{2.2}$$

(iii) *x^* satisfies*

$$\begin{cases} [\nabla q(x)]_i \geq 0, & \text{if} \quad x_i = l_i, \\ [\nabla q(x)]_i \leq 0, & \text{if} \quad x_i = u_i, \\ [\nabla q(x)]_i = 0, & \text{if} \quad l_i < x_i < u_i. \end{cases} \tag{2.3}$$

for $i = 1, 2, \ldots, n$.

(iv) *The variational inequality*

$$(x - x^*)^T \nabla q(x^*) \geq 0 \tag{2.4}$$

holds for any $l \leq x \leq u$, where x^* is a solution of the BLS (1.2)*.*

$$\begin{tikzcd}
\textbf{(i)} \arrow[r, Leftrightarrow] & \textbf{(ii)} \arrow[d, Rightarrow] & \\
\textbf{(iv)} \arrow[u, Rightarrow] & \textbf{(iii)} \arrow[l, Rightarrow] \arrow[r, Leftrightarrow] \arrow[d, Leftrightarrow] & \textbf{(v)} \\
& \textbf{(vi)} &
\end{tikzcd}$$

Figure 2.1: Equivalent relationship between the assertions in Theorem 2.1.1.

(v) *x^* is a solution of the fixed-point equation*

$$x = P(x - \alpha \nabla q(x)), \tag{2.5}$$

where $\alpha > 0$ is a scalar parameter, and the projection operator $P(\cdot)$ is defined as

$$[P(x)]_i = \begin{cases} l_i, & \text{if} \quad x_i < l_i \\ u_i, & \text{if} \quad x_i > u_i \\ x_i, & \text{if} \quad l_i \le x_i \le u_i \end{cases} \tag{2.6}$$

for $i = 1, 2, \ldots, n$.

(vi) *x^* is a solution of $\nabla_\Omega q(x) = 0$, where the projected gradient on $l \le x \le u$ is defined as*

$$[\nabla_\Omega q(x)]_i = \begin{cases} \min\left([\nabla q(x)]_i, 0\right), & \text{if} \quad x_i = l_i \\ \max\left([\nabla q(x)]_i, 0\right), & \text{if} \quad x_i = u_i \\ [\nabla q(x)]_i, & \text{if} \quad l_i \le x_i \le u_i \end{cases} \tag{2.7}$$

for $i = 1, 2, \ldots, n$.

Proof. The following proof is based on the equivalence relationship in Figure 2.1.

((i)$\Longleftrightarrow$(ii)) If x^* is a solution of LCP (2.2), it holds that

$$\begin{aligned} A^{\mathrm{T}}Ax^* - A^{\mathrm{T}}b - \lambda_1^* + \lambda_2^* = 0, \\ \lambda_1^* \geq 0, \quad \lambda_2^* \geq 0, \quad l \leq x^* \leq u, \\ (\lambda_1^*)^{\mathrm{T}}(x^* - l) = 0, \quad (\lambda_2^*)^{\mathrm{T}}(u - x^*) = 0 \end{aligned} \tag{2.8}$$

For any $l \leq x \leq u$,

$$\begin{aligned} & \|Ax-b\|_2^2 - \|Ax^*-b\|_2^2 \\ = \; & (x-x^*)^{\mathrm{T}}A^{\mathrm{T}}A(x-x^*) + 2(x-x^*)^{\mathrm{T}}(A^{\mathrm{T}}Ax^* - A^{\mathrm{T}}b) \\ = \; & (x-x^*)^{\mathrm{T}}A^{\mathrm{T}}A(x-x^*) + 2(x-l-x^*+l)^{\mathrm{T}}\lambda_1^* - 2(u-x^*-u+x)^{\mathrm{T}}\lambda_2^* \\ = \; & (x-x^*)^{\mathrm{T}}A^{\mathrm{T}}A(x-x^*) + 2(x-l)^{\mathrm{T}}\lambda_1^* + 2(u-x)^{\mathrm{T}}\lambda_2^* \geq 0. \end{aligned}$$

Hence, x^* is a minimization solution of BLS (1.2).

If x^* is a solution of BLS (1.2), then x^* satisfies the necessary KKT conditions as follows.

Stationarity

$$\begin{aligned} & \nabla_x \left(\frac{1}{2} x^{\mathrm{T}}(A^{\mathrm{T}}A)x - (A^{\mathrm{T}}b)^{\mathrm{T}}x - \lambda_1^{\mathrm{T}}(x-l) - \lambda_2^{\mathrm{T}}(u-x) \right) \Bigg|_{x=x^*} \\ = \; & A^{\mathrm{T}}Ax^* - A^{\mathrm{T}}b - \lambda_1 + \lambda_2 = 0. \end{aligned}$$

Primal and Dual feasibility

$$l \leq x^* \leq u, \quad \lambda_1 \geq 0, \quad \lambda_2 \geq 0$$

Complementarity slackness

$$\begin{bmatrix} \lambda_1 \\ \lambda_2 \end{bmatrix}^{\mathrm{T}} \begin{bmatrix} x^* - l \\ u - x^* \end{bmatrix} = 0.$$

By collecting the KKT conditions above, it is derived that x^* satisfies LCP (2.2).

((ii)$\Longrightarrow$(iii)) If x^* is a solution of LCP (2.2), then x^* satisfies (2.8), and thus

$$\begin{array}{ccccccc} x_i^* = l_i & \Longrightarrow & u_i - x_i^* > 0 & \Longrightarrow & (\lambda_2^*)_i = 0 & \Longrightarrow & (\lambda_1^*)_i = [\nabla q(x^*)]_i \geq 0; \\ x_i^* = u_i & \Longrightarrow & x_i^* - l_i > 0 & \Longrightarrow & (\lambda_1^*)_i = 0 & \Longrightarrow & (\lambda_2^*)_i = -[\nabla q(x^*)]_i \geq 0; \end{array}$$
$$l_i < x_i^* < u_i \Longrightarrow (\lambda_1^*)_i = (\lambda_2^*)_i = 0 \Longrightarrow [\nabla q(x^*)]_i = 0,$$

for $i = 1, 2, \ldots, n$. Collecting the analysis above, we can easily derive that (2.3) is valid for x^*.

((iii)$\Longrightarrow$(iv)) By the fact that x^* satisfies (2.3), for any $l \leq x \leq u$,

$$\begin{array}{ccccc} x_i^* = l_i & \Longrightarrow & x_i - x_i^* \geq 0, \ [\nabla q(x^*)]_i \geq 0 & \Longrightarrow & (x_i - x_i^*)\,[\nabla q(x^*)]_i \geq 0; \\ x_i^* = u_i & \Longrightarrow & x_i - x_i^* \leq 0, \ [\nabla q(x^*)]_i \leq 0 & \Longrightarrow & (x_i - x_i^*)\,[\nabla q(x^*)]_i \geq 0; \\ l_i < x_i^* < u_i & \Longrightarrow & [\nabla q(x^*)]_i = 0 & \Longrightarrow & (x_i - x_i^*)\,[\nabla q(x^*)]_i = 0, \end{array}$$

for $i = 1, 2, \ldots, n$, which imply that the variational inequality (2.4) is valid for x^*.

((iv)$\Longrightarrow$(i)) For any $l \leq x \leq u$, if there exists $l \leq x^* \leq u$ such that (2.4) holds, then

$$\begin{aligned} & \|Ax - b\|_2^2 - \|Ax^* - b\|_2^2 \\ = \ & (x - x^*)^{\mathrm{T}} A^{\mathrm{T}} A (x - x^*) + 2x^{\mathrm{T}} A^{\mathrm{T}} A x^* - 2(A^{\mathrm{T}} b)^{\mathrm{T}} x - 2(x^*)^{\mathrm{T}} A^{\mathrm{T}} A x^* + 2(A^{\mathrm{T}} b)^{\mathrm{T}} x^* \\ = \ & (x - x^*)^{\mathrm{T}} A^{\mathrm{T}} A (x - x^*) + 2(x - x^*)^{\mathrm{T}} (A^{\mathrm{T}} A x^* - A^{\mathrm{T}} b) \geq 0, \end{aligned}$$

which implies that x^* is a solution of BLS problem.

((iii)$\Longleftrightarrow$(v)) If x^* satisfies (2.3), then according to the definition of the projection operator $P(\cdot)$, when $x_i^* = l_i$,

$$[\nabla q(x^*)]_i \geq 0 \quad \Longleftrightarrow \quad x_i^* - \alpha\,[\nabla q(x^*)]_i \leq l_i \quad \Longleftrightarrow \quad P(x_i^* - \alpha\,[\nabla q(x^*)]_i) = l_i;$$

when $x_i^* = u_i$,

$$[\nabla q(x^*)]_i \leq 0 \quad \Longleftrightarrow \quad x_i^* - \alpha\,[\nabla q(x^*)]_i \geq u_i \quad \Longleftrightarrow \quad P(x_i^* - \alpha\,[\nabla q(x^*)]_i) = u_i;$$

when $l_i < x_i^* < u_i$,

$$[\nabla q(x^*)]_i = 0 \quad \Longleftrightarrow \quad x_i^* - \alpha\,[\nabla q(x^*)]_i = x_i^* \Longleftrightarrow P(x_i^* - \alpha\,[\nabla q(x^*)]_i) = x_i^*,$$

for $\alpha > 0$ and $i = 1, 2, \dots, n$, which is equivalent to the fact that x^* is a solution of the fixed-point equation (2.5).

(**(iii)$\Longleftrightarrow$(vi)**) If x^* satisfies (2.3), then

$$\begin{aligned} \text{when } x_i^* = l_i,\ [\nabla q(x^*)]_i \geq 0 \quad &\Longleftrightarrow \quad \min([\nabla q(x^*)]_i, 0) = 0; \\ \text{when } x_i^* = u_i,\ [\nabla q(x^*)]_i \leq 0 \quad &\Longleftrightarrow \quad \max([\nabla q(x^*)]_i, 0) = 0; \\ \text{when } l_i < x_i^* < u_i,\ [\nabla q(x^*)]_i &= 0, \end{aligned}$$

for $i = 1, 2, \dots, n$, which is equivalent to the fact that x^* is a solution of (2.7). ∎

We can easily obtain the corresponding equivalent assertions for nonnegative constrained case as follows.

Corollary 2.1.2. *For the NNLS problem* (1.1)*, the following assertions are all equivalent.*

(i) *x^* is a solution of the NNLS problem* (1.1)*;*

(ii) *there exists a solution x^* satisfying the LCP*

$$x \geq \mathbf{0}, \quad \lambda = A^T Ax - A^T b \geq \mathbf{0}, \quad \textit{and} \quad x^T \lambda = 0, \tag{2.9}$$

(iii) *x^* satisfies*

$$\begin{cases} [\nabla q(x)]_i \geq 0, & \text{if} \quad x_i = 0, \\ [\nabla q(x)]_i = 0, & \text{if} \quad x_i > 0. \end{cases} \tag{2.10}$$

for $i = 1, 2, \dots, n$.

(iv) *For any $x \in \Omega \equiv \{x \in \mathbf{R}^n : x \geq 0\}$, the variational inequality* (2.4) *holds, where x^* is a solution of NNLS* (1.1)*.*

(v) *x^* is a solution of the fixed-point equation $x = P(x - \alpha \nabla q(x))$, where $\alpha > 0$ is a scalar parameter, and the projection operator $P(\cdot)$ is defined as*

$$[P(x)]_i = \begin{cases} 0, & \text{if} \quad x_i < 0 \\ x_i, & \text{if} \quad x_i \geq 0 \end{cases} \tag{2.11}$$

for $i = 1, 2, \dots, n$.

(vi) *x^* is a solution of $\nabla_\Omega q(x) = 0$, where the projected gradient on $x \geq 0$ is defined as*

$$[\nabla_\Omega q(x)]_i = \begin{cases} \min\left([\nabla q(x)]_i, 0\right), & \text{if} \quad x_i = 0 \\ [\nabla q(x)]_i, & \text{if} \quad x_i \geq 0 \end{cases} \tag{2.12}$$

for $i = 1, 2, \ldots, n$.

Remark that due to the equivalence, the methods for LCP (2.9) and (2.2), and variational inequality (2.4) can be applied for NNLS and BLS, and vice versa. The **(ii)** and **(v)** are the basis for the modulus methods and the projected gradient methods, respectively, by utilizing fixed-point equations to generate an approximate solution sequence. In addition, the condition **(vi)** can be used as the stopping criteria of algorithms for solving NNLS and BLS.

2.2 Projection Gradient Method

In order to handle the constraints, one of the most popular techniques is the projection operations (2.11) and (2.6). For example, the projection gradient method proposed by Bertsekas [5] iteratively solves the fixed-point equation (2.5) by generating a sequence $\{x^k\}$ via

$$x^{k+1} = P(x^k - \alpha_k \nabla q(x^k)).$$

The parameter $\alpha_k > 0$ is chosen so that $q(x^{k+1}) < q(x^k)$. The method was applied to quadratic programming with box constraints in [27, 76, 77].

Algorithm 2.2.1. Projected Gradient Method

```
1.    Choose an initial approximate solution $x^0$ and compute $r^0 = b - Ax^0$.
2.    For $k = 0, 1, 2, \ldots$ until convergence
3.        Compute $s^k = A^T r^k$.
4.        Compute $\alpha_k = \|s^k\|_2^2 / \|As^k\|_2^2$.
5.        Set $x^{k+1} = P(x^k + \beta^m \alpha_k s^k)$, and find the smallest integer $m \geq 0$
          that satisfies
```

$$q(x^{k+1}) \leq q(x^k) + \mu \nabla q(x^k)^T (x^{k+1} - x^k), \tag{2.13}$$

```
            where $0 < \beta < 1$ and $0 \leq \mu < 1$.
6.          Compute $r^{k+1} = b - Ax^{k+1}$.
7.      Endfor
```

Here, (2.13) is called the sufficient decrease condition, or Armijo condition. One of the advantages of this procedure is that it produces an acceptable α_k with a finite number of evaluations of $q(x)$. Another advantage is that with this choice of α_k, any limit point x^* of $\{x^k\}$ is a stationary point of $q(x)$ [76].

Note that the theoretical convergence analysis of the projection gradient method was discussed by Bertsekas [5], Calamai and Moré [27], Burke and Moré [16], and Dunn [32].

Another projection method is the projected successive overrelaxation (SOR) method proposed by Cryer [22] for the solution of nonnegative constrained quadratic programming. The method solves the equivalent LCP (2.9) by the fixed-point iteration. More sophisticated projection iterative methods were discussed and analyzed when the coefficient matrix is either symmetric or nonsymmetric by Ahn [1], Mangasarian [70], Pang et. al. [28, 84], and Murty [71].

We can easily extend Cryer's method [22] to obtain the projected NR-SOR method, where the NR-SOR method is the SOR iteration for the normal equation (1.4). Below, we give their corresponding versions for the NNLS problem. For the projected NR-SOR method, we only list one forward sweep.

Algorithm 2.2.2. Projected NR-SOR with One Forward Sweep

```
1.      Given $x^k$ and $r^k (= b - Ax^k)$, $k \geq 0$.
2.      For $i = 1, 2, \ldots, n$ Do
3.          Compute $\delta_i = \omega(r^k, Ae_i)/\|Ae_i\|_2^2$, where $e_i$ is the ith column vector
            of the $n \times n$ identity matrix and $0 < \omega < 2$.
4.          Compute $x_i^{k+1} = P(x_i^k + \delta_i)$.
5.          Compute $r^k = r^k - (x_i^{k+1} - x_i^k)(Ae_i)$.
6.      Endfor
7.      Set $r^{k+1} = r^k$.
```

Unlike the Newton-type method where the Hessian matrix is required, the projection gradient method only use the gradient information and thus is easy to understand and implement. However, similar to the steepest descent method, it may suffer from slow convergence due to the zigzag phenomenon when approaching the minimizer if the condition number is large [102]. Therefore, instead of using it to solve the NNLS and BLS directly, the projection gradient method is applied to identify the optimal active constraints quickly. Once the optimal active constraints are identified, the minimization problem can be treated as an unconstrained problem. This is the idea of the active set strategy, which will be introduced in the following section.

2.3 Active Set Methods

A class of inner outer iterative methods is widely discussed for the solution of NNLS and BLS problems, where a series of unconstrained least squares problems are solved in the inner iteration, and the obtained solution is updated to satisfy the nonnegative constraints, and then the inner iteration is resumed for each outer iteration until convergence. Remark that the inner outer iteration methods contain two tasks including how to update the solution of the unconstrained least squares problem when some of its components violate the bounds, and when to terminate the inner iteration and start the next outer iteration.

In the following, we will discuss three classical active set inner outer iteration methods.

2.3.1 Lawson and Hanson's Method

The NNLS algorithm of Lawson and Hanson [67] is an active set method, and was the *de facto* method for solving (1.1) for many years [61]. This algorithm is implemented in MATLAB as the function “lsqnonneg". Bro and Jong [14] analyzed the NNLS algorithm and proposed a Fast-NNLS (FNNLS) method to accelerate it for the solution of nonnegative constrained least squares with multiple right-hand sides. A recent variant of FNNLS appropriately rearranges calculations to achieve further speedups in the presence of multiple right hand sides [15]. We consider the single right hand side case here, so only the NNLS algorithm proposed by Lawson and Hanson is reviewed.

For the NNLS problem (1.1), suppose x^k is the kth iterative solution. Then the active set and the corresponding set of free variables are defined as

$$\mathscr{A}(x^k) = \{j : x_j^k = 0\} \quad \text{and} \quad \mathscr{F}(x^k) = \{1,2,\dots,n\} \backslash \mathscr{A}(x^k). \tag{2.14}$$

The idea of active set method for NNLS (1.1) and BLS (1.2) is that if the constraints active at the final solution are known in advance, then the original problem can be solved by simply optimizing the objective function in an unconstrained manner over only the variables that correspond to the inactive constraints [61]. The NNLS algorithm [67] is described as follows.

Algorithm 2.3.1. NNLS Algorithm [67]

```
1.   Choose $x^0 = 0$ and compute $r^0 = b$ and $s^0 = A^T r^0$.
2.   Set $\mathscr{A}(x^0) = \{1,2,\dots,n\}$ and $\mathscr{F}(x^0) = \emptyset$.
3.   For $k = 0,1,2,\dots$ until either $\mathscr{A}(x^k) = \emptyset$ or $s^0 \le 0$ is satisfied
4.       Find index $t \in \mathscr{A}(x^k)$ such that $s_t^k = \max\{s_j^k,\ j \in \mathscr{A}(x^k)\}$.
5.       Move $t$ from $\mathscr{A}(x^k)$ to $\mathscr{F}(x^k)$.
6.       Solve the reduced unconstrained least squares problem

             $$\min_{y \in \mathbf{R}^{\check{n}}} \|A_{\mathscr{F}} y - b\|_2, \tag{2.15}$$

         where $\check{n}$ is the number of elements in $\mathscr{F}(x^k)$.
7.       If $y > 0$, set $x_{\mathscr{F}}^{k+1} = y$.
8.       Otherwise, find $q \in \mathscr{F}(x^k)$ such that

             $$\alpha = \frac{x_q^k}{x_q^k - y_q} = \min_{y_j \le 0} \left\{ \frac{x_j^k}{x_j^k - y_j},\ j \in \mathscr{F}(x^k) \right\}.$$

9.       Set $x_{\mathscr{F}}^{k+1} = x_{\mathscr{F}}^k + \alpha(y - x_{\mathscr{F}}^k)$.
10.      Compute $r^{k+1} = b - Ax^{k+1}$ and $s^{k+1} = A^T r^{k+1}$.
11.      Update $\mathscr{A}(x^{k+1})$ and $\mathscr{F}(x^{k+1})$.
12.  Endfor
```

Here, note that the update of $\mathscr{A}(x^{k+1})$ and $\mathscr{F}(x^{k+1})$ in Step 11 is implemented by moving all indices that $x_j^{k+1} = 0,\ j \in \mathscr{F}(x^k)$ from $\mathscr{F}(x^k)$ to $\mathscr{A}(x^k)$. Meanwhile, in order to handle the nonnegative constraints, Algorithm 2.3.1 choose the reduced step size α instead of utilizing the projection operator $P(\cdot)$.

We can easily extend the Algorithm 2.3.1 for the solution of BLS problem. The active set is defined as

$$\mathscr{A}(x^k) = \{j : x_j^k = l_i \text{ or } x_j^k = u_i\} \quad \text{and} \quad \mathscr{F}(x^k) = \{1,2,\dots,n\} \backslash \mathscr{A}(x^k). \tag{2.16}$$

The corresponding BLS algorithm is described as follows.

Algorithm 2.3.2. Lawson-Hanson Algorithm for BLS

```
1.   Choose $x^0$ and compute $r^0 = b - Ax^0$ and $s^0 = A^T r^0$.
2.   Compute $\mathscr{A}(x^0)$ and $\mathscr{F}(x^0)$ based on (2.16).
3.   For $k = 0,1,2,\dots$ until either $\mathscr{A}(x^k) = \emptyset$ or $s^0 \le 0$ is satisfied
4.       Find index $t \in \mathscr{A}(x^k)$ such that $s_t^k = \max\{s_j^k,\ j \in \mathscr{A}(x^k)\}$.
5.       Move $t$ from $\mathscr{A}(x^k)$ to $\mathscr{F}(x^k)$.
6.       Solve the reduced unconstrained least squares problem
             $$\min_{y \in \mathbf{R}^{\check{n}}} \|A_{\mathscr{F}} y - (b - A_{\mathscr{A}} x^k)\|_2, \tag{2.17}$$
         where $\check{n}$ is the number of elements in $\mathscr{F}(x^k)$.
7.       If $l_{\mathscr{F}} \le y \le u_{\mathscr{F}}$, set $x_{\mathscr{F}}^{k+1} = y$.
8.       Otherwise, find $q \in \mathscr{F}$ such that
             $$\alpha = \min\left\{ \min_{y_j \le l_j} \left\{ \frac{x_j^k - l_j}{x_j^k - y_j} \right\},\ \min_{y_j \ge u_j} \left\{ \frac{x_j^k - u_j}{x_j^k - y_j} \right\},\ j \in \mathscr{F} \right\}.$$
9.       Set $x_{\mathscr{F}}^{k+1} = x_{\mathscr{F}}^k + \alpha(y - x_{\mathscr{F}}^k)$.
10.      Compute $r^{k+1} = b - Ax^{k+1}$ and $s^{k+1} = A^T r^{k+1}$.
11.      Update $\mathscr{A}(x^{k+1})$ and $\mathscr{F}(x^{k+1})$.
12.  Endfor
```

The disadvantage of Lawson and Hanson's method is that typically only one variable between the active set and the passive set is exchanged in each iteration, which make it very slow for the sequence to converge.

2.3.2 Polyak and O'Leary's Method

Instead of projection-type methods, which orthogonally project the iterated solution into the feasible region by (2.6), Polyak [85] and O'Leary [81] proposed a generalized CG method for solving general box constrained quadratic programming problems with a symmetric positive definite matrix, by restricting the step size in each CG iteration to satisfy constraints, which can be naturally applied to solving NNLS problems. A similar algorithm called the restricted LSQR method was presented by Lötstedt [66], where LSQR is a stabilized version of CGLS, and Bierlaire, Toint and Tuyttens [19] introduced a variant of the algorithm.

Set $B = A^{\mathrm{T}}A$ and $c = A^{\mathrm{T}}b$. Polyak's algorithm [85] is as follows.

Algorithm 2.3.3. Polyak's Algorithm for BLS

```
1.  Initialization
2.  Choose an $x^0$ such that $l \le x \le u$ and set $k = 0$.
3.  Set $I = \{1, 2, \ldots, n\}$. This definition ensures that the first halting test in the
    outer iteration will work properly.
4.  Outer Iteration
5.  Let $k = k+1$, $x^k = x^{k+1}$, $y^k = Bx^k - c$, and $I_{k-1} = I$.
6.  Define $I_k = \{i : x_i^k = l_i \text{ and } y_i^k > 0\} \cup \{i : x_i^k = u_i \text{ and } y_i^k < 0\}$.
7.  If $I_k = I_{k-1}$, halt. The optimal solution has been found.
    Otherwise, set $I = I_k$ and begin the inner iteration.
8.  Inner Iteration
9.  Partition and rearrange the matrix system as
```

$$x^k \to \begin{bmatrix} x_I^k \\ x_J^k \end{bmatrix}, \quad b^k \to \begin{bmatrix} c_I^k \\ c_J^k \end{bmatrix}, \quad B \to \begin{bmatrix} B_{II} & B_{JI}^T \\ B_{JI} & B_{JJ} \end{bmatrix}$$

with B_{JJ} $s \times s$, symmetric positive definite. We initialize the conjugate gradient iteration to solve the equation

$$B_{JJ}x_J = c_J - B_{JI}x_I. \tag{2.18}$$

The sequence $\{z^q\}$ will be our approximations to the solution vector x_J. The vectors p^q will be search directions, and vectors r^q will be residuals for (2.18).

Set $q = 0$ and

$$z^0 = x_J^k, \quad p^0 = r^0 = c_J - B_{JI}x_I^k - B_{JJ}z^0.$$

10. *Calculate the new iterate and residual. We compute two step parameters: a_{cg} is the conjugate gradient step in the direction p^q, and $a_{\max}$ is the largest step in that direction which does not violate any bounds on the variables*

$$a_{cg} = \frac{(r^q, p^q)}{(p^q, B_{JJ}p^q)} = \frac{(r^q, r^q)}{(p^q, B_{JJ}p^q)}$$
$$a_{\max} = \min\left(\min_{p_j^q<0} \frac{l_j - z_j^q}{p_j^q}, \ \min_{p_j^q>0} \frac{u_j - z_j^q}{p_j^q}, \ j = 1, 2, \ldots, s\right).$$

The step taken is the smaller of these two positive numbers

$$a_q = \min(a_{cg}, \ a_{\max}),$$
$$z^{q+1} = z^q + a_q p^q, \quad r^{q+1} = r^q - a_q B_{JJ} p^q.$$

The vector y could also be updated at this stage to correspond to the current values x_I^k and z^{q+1}.

11. *Test for termination of the inner iteration:*
12. *If $r^{q+1} = 0$, set $x_J^k = z^{q+1}$ and restart the outer iteration.*
13. *If $\{j : z_j^{q+1} = l_j$ or $z_j^{q+1} = u_j\} = \emptyset$, proceed with Step 15.*
14. *Otherwise, set $x_J^k = z^{q+1}$ and $I = \{i : x_i^k = l_i$ or $x_i^k = u_i\}$. If $I = \{1, 2, \ldots, n\}$, then restart the outer iteration. Otherwise restart the inner iteration.*
15. *Calculate the new search direction p^{q+1}, B_{JJ}-conjugate to the old ones:*

$$b_q = -\frac{(B_{JJ}p^q, r^{q+1})}{(p^q, B_{JJ}p^q)} = \frac{(r^{q+1}, r^{q+1})}{(r^q, r^q)}.$$
$$p^{q+1} = r^{q+1} + b_q p^q.$$

Replace q by $q + 1$ and go to Step 10.

16. *The initialization of z^0, p^0, r^0, and q in Step 9 of the inner iteration, plus Step 10 and 15 with $a_q = a_{cg}$ and Step 11 replaced by*

```
17.   If $r^{q+1} = 0$, then halt with $x_J = z^{q+1}$, constitute the standard conjugate gradient
      algorithm for solving the linear system (2.18).
```

It can be proved that Polyak's algorithm terminates in a finite number of iterations [81]. In addition, the performance of the Polyak's algorithm can be enhanced by improving the convergence rate of the inner iteration. This can be accomplished by using the preconditioned conjugate gradient algorithm with matrix splitting [81]. O'Leary's algorithm [81] is as follows.

Algorithm 2.3.4. O'Leary's Algorithm for BLS

```
1.   Initialization
2.   Choose an $x^0$ such that $l \le x \le u$ and set $k = 0$.
3.   Set $I = \{1, 2, \ldots, n\}$. This definition ensures that the first halting test in the
     outer iteration will work properly.
4.   Outer Iteration
5.   Let $k = k+1$, $x^k = x^{k+1}$, $y^k = Bx^k - c$, and $I_{k-1} = I$.
6.   Define $I_k = \{i : x_i^k = l_i \text{ and } y_i^k > 0\} \cup \{i : x_i^k = u_i \text{ and } y_i^k < 0\}$.
7.   If $I_k = I_{k-1}$, halt. The optimal solution has been found. Otherwise, set $I = I_k$
     and begin the inner iteration.
8.   Inner Iteration
9.   Partition and rearrange the matrix system as
```

$$x^k \to \begin{bmatrix} x_I^k \\ x_J^k \end{bmatrix}, \quad b^k \to \begin{bmatrix} c_I^k \\ c_J^k \end{bmatrix}, \quad B \to \begin{bmatrix} B_{II} & B_{JI}^T \\ B_{JI} & B_{JJ} \end{bmatrix}$$

with B_{JJ} $s \times s$, symmetric positive definite. We initialize the conjugate gradient iteration to solve the equation

$$B_{JJ} x_J = c_J - B_{JI} x_I. \tag{2.19}$$

The sequence $\{z^q\}$ will be our approximations to the solution vector x_J. The vectors p^q will be search directions, and vectors r^q will be residuals for (2.19). Set $q = 0$ and

$$z^0 = x_J^k, \quad p^0 = r^0 = c_J - B_{JI} x_I^k - B_{JJ} z^0.$$

10. *Calculate the new iterate and residual. We compute two step parameters: a_{cg} is the conjugate gradient, or, equivalently for this step, the steepest descent parameter, and $a_{\max}$ is the largest step in that direction which does not violate any bounds on the variables*

$$a_{cg} = \frac{(r^0, r^0)}{(r^0, B_{JJ} r^0)}$$
$$a_{\max} = \min\left(\min_{p_j^0 < 0} \frac{l_j - z_j^0}{p_j^0},\ \min_{p_j^0 > 0} \frac{u_j - z_j^0}{p_j^0},\ j = 1, 2, \ldots, s \right).$$

The step taken is the smaller of these two positive numbers

$$a_0 = \min(a_{cg},\ a_{\max}),$$
$$z^1 = z^0 + a_0 r^0, \quad r^1 = r^0 - a_0 B_{JJ} p^0.$$

11. *If $r^1 = 0$, set $x_J^k = z^1$ and restart the outer iteration.*
12. *If $\{j : z_j^1 = l_j \text{ or } z_j^1 = u_j\} = \emptyset$, proceed with Step 15.*
13. *Otherwise, set $x_J^k = z^1$ and $I = \{i : x_i^k = l_i \text{ or } x_i^k = u_i\}$. If $I = \{1, 2, \ldots, n\}$, then restart the outer iteration. Otherwise repartition x, b, and B as in Step 9, set*

$$z^1 = x_J^k, \quad r^1 = b_J - B_{JI} x_I^k - z^1,$$

and continue with Step 14.

14. *Initialize the scaled (preconditioned) conjugate gradient algorithm. Choose M to scale the matrix B_{JJ}, set $q = 1$ and let*

$$p^1 = M^{-1} r^1.$$

15. *Calculate the new iterate and residual:*

$$\begin{aligned}
a_{cg} &= \frac{(r^q, p^q)}{(p^q, B_{JJ}p^q)} = \frac{(r^q, M^{-1}r^q)}{(p^q, B_{JJ}p^q)} \\
a_{\max} &= \min\left(\min_{p_j^q<0} \frac{l_j - z_j^q}{p_j^q},\ \min_{p_j^q>0} \frac{u_j - z_j^q}{p_j^q},\ j = 1,2,\dots,s \right). \\
a_q &= \min(a_{cg},\ a_{\max}), \\
z^{q+1} &= z^q + a_q p^q, \quad r^{q+1} = r^q - a_q B_{JJ} p^q.
\end{aligned}$$

16. *Test for termination of the inner iteration:*
17. *If $r^{q+1} = 0$, set $x_J^k = z^{q+1}$ and restart the outer iteration.*
18. *If $\{j : z_j^{q+1} = l_j$ or $z_j^{q+1} = u_j\} = \emptyset$, proceed with Step 15.*
19. *Otherwise, set $x_J^k = z^{q+1}$ and $I = \{i : x_i^k = l_i$ or $x_i^k = u_i\}$. If $I = \{1,2,\dots,n\}$, then restart the outer iteration. Otherwise restart the inner iteration.*
20. *Calculate the new search direction p^{q+1}, B_{JJ}-conjugate to the old ones:*

$$\begin{aligned}
b_q &= -\frac{(B_{JJ}p^q, M^{-1}r^{q+1})}{(p^q, B_{JJ}p^q)} = \frac{(r^{q+1}, M^{-1}r^{q+1})}{(r^q, M^{-1}r^q)}. \\
p^{q+1} &= r^{q+1} + b_q p^q.
\end{aligned}$$

Replace q by $q+1$ and go to Step 10.

21. *The initialization of z^0, p^0, r^0, and q in Step 9 of the inner iteration, plus Step 10 and 15 with $a_q = a_{cg}$ and Step 11 replaced by*
22. *If $r^{q+1} = 0$, then halt with $x_J = z^{q+1}$, constitute the scaled (preconditioned) conjugate gradient algorithm for solving the linear system* (2.18).

The advantage of O'Leary's algorithm is that it can terminate in a finite number of iterations and converges faster than Polyak's algorithm. However, the disadvantage is that the inner iteration is terminated as soon as a component of a computed iterate violates a constraint, which forces frequent resuming of the inner iteration and thus slows down the convergence. Another undesirable feature is that the active set type algorithm allows only one variable to leave a bound at a given outer iteration, which only allows to add or delete one index from the active set at a time. This is an inefficient feature when the number of variables is large.

2.3.3 Projected Quasi-Newton Method

Bertsekas proposed a projected Newton-type method [6] which is based on

$$x^{k+1} = P(x^k - \beta^k S^k \nabla q(x^k)), \tag{2.20}$$

where β^k is determined by an Armijo-like rule [5], called the Armijo along projection arc (APA) rule. In [6], the gradient scaling S^k is reset to the inverse of the Hessian $(\nabla^2 q(x^k))^{-1}$ at every iteration, followed by the modification

$$S^k(i,j) = S^k(j,i) = 0, \quad \forall i \in I^+, \quad j = 1,2,\ldots,n, \quad j \neq i.$$

The projected quasi-Newton method combines the gradient projection method (2.5) and the active set strategy. Consider the NNLS problem (1.1) first. Suppose x^k is the kth iterative solution. Then the binding set is defined as

$$\mathscr{B}(x^k) = \{j : x_j^k = 0,\ \lambda_j^k \geq 0\}, \tag{2.21}$$

where $\lambda^k = -A^{\mathrm{T}} r^k = -A^{\mathrm{T}}(b - Ax^k)$ is the Lagrange multiplier, which is also the gradient vector of the least-squares objective function in (1.3), or the negative residual of the normal equation $A^{\mathrm{T}}Ax = A^{\mathrm{T}}b$. More specifically, $\mathscr{B}(x^k)$ contains variables that satisfy the KKT conditions (2.10) at the current iteration, which makes them more likely to be active at the final solution. In contrast, for a component $x_j^k = 0$, for which $\left[\nabla q(x^k)\right]_j < 0$, it may be possible to optimize the objective further by making $x_j^k > 0$ and $\left[\nabla q(x^k)\right]_j = 0$ at the next iteration, thereby violating the notion of a fixed variable [61].

Denote $A_{\mathscr{F}}$ as the submatrix of A consisting of the columns of A whose indices belong to $\tilde{\mathscr{F}} \equiv \{1,2,\ldots,n\}\backslash\mathscr{B}$. Then, the reduced NNLS problem

$$\min_{y \in \mathbf{R}^{\tilde{n}}} q_k(y) \equiv \|A_{\tilde{\mathscr{F}}}y - b\|_2 \quad \text{subject to} \quad x \geq 0, \tag{2.22}$$

where $\tilde{n}$ is the number of elements in $\tilde{\mathscr{F}}$. Let $\tilde{y}$ denote the solution of (2.22), then we obtain the update $x_{\tilde{\mathscr{F}}}^{k+1} = \tilde{y}$.

Next, we show how to compute the solution $\tilde{y}$ of the reduced NNLS problem (2.22). We make use of the projected gradient method (2.5) and a non-diagonal gradient scaling matrix S^k, which approximates the inverse of the Hessian $\nabla^2 q = A^{\mathrm{T}}A$ at each iteration. The use of such non-diagonal gradient scaling matrices accelerates the rate of convergence, and it is central to the success of the popular Quasi-Newton method [61]. We remark that the "non-diagonality" helps by pointing out that the ordinary projection gradient method may be viewed as using $S^k = I$ at each iteration.

We compute the vector $\tilde{y}$ by using the following update

$$\tilde{y} = y^k + \alpha(\gamma^k(\beta; y^k) - y^k), \tag{2.23}$$

where $\alpha, \beta \geq 0$ are certain parameters, and the function $\gamma(\beta; y)$ is defined as the projection

$$\gamma^k(\beta; y^k) = P(y^k - \beta \bar{S}^k \nabla q(y^k)), \tag{2.24}$$

where $\bar{S}^k$ is an approximate principal submatrix of S^k, and $P(\cdot)$ denotes the orthogonal projection onto $\mathbb{R}^n_+$.

The overall method that encapsulates all the details mentioned above is presented as Algorithm 2.3.5.

Algorithm 2.3.5. Projected Quasi-Newton NNLS Algorithm [61]

```
1.    Choose $x^0$ and $S^0 = I$.
2.    Compute $r^0 = b - Ax^0$ and $s^0 = A^T r^0$.
3.    For $k = 0, 1, 2, \ldots$ until convergence
4.        Compute $\mathscr{B}(x^k)$ and $\tilde{\mathscr{F}}(x^k)$.
5.        Solve the reduced NNLS problem (2.22):
6.            Find appropriate $\alpha^k$ and $\beta^k$.
7.            Compute $\tilde{y}$ by (2.23) and (2.24).
8.        Update gradient scaling matrix $S^k$ to obtain $S^{k+1}$.
9.        Set $x^{k+1}_{\tilde{\mathscr{F}}} = \tilde{y}$ and compute $r^{k+1} = b - Ax^{k+1}$ and $s^{k+1} = A^T r^{k+1}$.
10.   Endfor
```

In the following we discuss the line search in Step 6 and gradient scaling update in Step 8.

Line Search

Naturally, the parameters β and α are crucial to the update (2.23), and subsequently to the convergence speed of overall algorithm. One of the most popular methods is the limited minimization rule, which computes an appropriate α for a pre-specified fixed value of β by performing the following single-variable minimization

$$\alpha^k = \operatorname{argmin}_{\alpha \in [0,1]} q_k(y^k + \alpha(\gamma^k(\beta; y^k) - y^k)). \tag{2.25}$$

We exploit the fact that the objective function in (2.25) is a continuous and quadratic univariate function of α. Letting $d = \gamma^k(\beta; y^k) - y^k$ and setting the derivative $dq_k/d\alpha = 0$, we have

$$\begin{aligned} \tfrac{dq_k}{d\alpha} &= d^{\mathrm{T}}\nabla q_k(y^k + \alpha d) = d^{\mathrm{T}}\bar{A}^{\mathrm{T}}\bar{A}y^k + \alpha\|\bar{A}d\|^2 - d^{\mathrm{T}}\bar{A}^{\mathrm{T}}b = 0 \\ \Longrightarrow \alpha &= \tfrac{d^{\mathrm{T}}(\bar{A}^{\mathrm{T}}b - \bar{A}^{\mathrm{T}}\bar{A}y^k)}{\|\bar{A}d\|^2}. \end{aligned}$$

With the above α, we can choose the current step size to be $\alpha^k = \operatorname{mid}(0, \alpha, 1)$.

Although the limited minimization rule above yields an analytic solution for a fixed value β, sometimes the convergence speed of the overall algorithm can be sensitive to this pre-specified inner step length β. Finding a good value of β could itself be time-consuming. One choice can be the Armijo step-size rule, which is simple and usually works well both for unconstrained and constrained problems. Bertsekas [5] proposed a variation of Armijo-like rule, called the Armijo along projection arc (APA) rule. Interestingly, unlike the above methods, APA fixes α in (2.23) and computes an appropriate β at each iteration. In our implementation of APA, we fix $\alpha = 1$, which leads the following changes to the update

$$\tilde{y} = \gamma^k(\beta^k; y^k) = P(y^k - \beta^k S^k \nabla q(y^k)).$$

Here, the computation of β^k takes the following form. Let m be the smallest nonnegative integer such that

$$q_k(y^k) - q_k(\gamma^k(s^m\sigma; y^k)) \geq \tau \nabla q_k(y^k)^{\mathrm{T}}(y^k - \gamma^k(s^m\sigma; y^k)), \tag{2.26}$$

where $\sigma > 0$, $0 < \sigma < 1$ and $0 < \tau < 0.5$ are fixed scalars. The resulting step size $\beta = s^m \sigma$.

BFGS Update

We consider how the gradient scaling update step can be implemented. In a traditional Newton method the inverse of the Hessian of the objective function serves as the gradient scaling matrix. However, a fundamental drawback of using the inverse Hessian for large scale problems is cost of computing it. For the NNLS objective function, this amounts to $O(n^3)$ in general, which can make the overall procedure unacceptably expensive. A further drawback, especially for large scale problems is that the Hessian or its inverse can be ill-conditioned. This not only causes numerical difficulties, but can also lead to a poor rate of convergence.

To circumvent some of the problems associated with the use of the full Hessian, researchers have used the idea of approximating it in various ways. Such an approximation leads to what is known as the Quasi-Newton or Variable Metric method – a popular and well-established method for nonlinear programming. Some common techniques for iteratively approximating the Hessian are the Powell-Symmetric-Broyden (PSB), Davidon-Fletcher-Powell (DFP), and the Broyden-Fletcher-Goldfarb-Shanno (BFGS) updates, where the latter is believed to be the most effective in general.

Suppose H^k is the current approximation to the Hessian. The BFGS update adds a rank-two correction to H^k to obtain

$$H^{k+1} = H^k - \frac{H^k u u^{\mathrm{T}} H^k}{u^{\mathrm{T}} H^k u} + \frac{w w^{\mathrm{T}}}{u^{\mathrm{T}} w}, \tag{2.27}$$

where w and u are defined as

$$w = \nabla q(x^{k+1}) - \nabla q(x^k) \quad \text{and} \quad u = x^{k+1} - x^k.$$

Let S^k denote the inverse of H^k. Then, by applying the Sherman-Morrison-Woodbury formula, update (2.27) amounts to the following

$$S^{k+1} = S^k + \left(1 + \frac{w^{\mathrm{T}} S^k w}{u^{\mathrm{T}} w}\right) \frac{u u^{\mathrm{T}}}{u^{\mathrm{T}} w} - \frac{S^k w u^{\mathrm{T}} + u w^{\mathrm{T}} S^k}{u^{\mathrm{T}} w}. \tag{2.28}$$

Now we use the fact that

$$w = \nabla q(x^{k+1}) - \nabla q(x^k) = A^{\mathrm{T}}A(x^{k+1} - x^k) = A^{\mathrm{T}}Au,$$

to rewrite (2.28) as

$$S^{k+1} = S^k + \left(1 + \frac{u^{\mathrm{T}}A^{\mathrm{T}}AS^kA^{\mathrm{T}}Au}{u^{\mathrm{T}}A^{\mathrm{T}}Au}\right)\frac{uu^{\mathrm{T}}}{u^{\mathrm{T}}A^{\mathrm{T}}Au} - \frac{S^kA^{\mathrm{T}}Auu^{\mathrm{T}} + uu^{\mathrm{T}}A^{\mathrm{T}}AS^k}{u^{\mathrm{T}}A^{\mathrm{T}}Au}. \tag{2.29}$$

The convergence analysis and numerical comparison of the above methods were shown in [61].

2.3.4 Gradient Projection Conjugate Gradient Method

In order to avoid the disadvantages of the active set methods mentioned previously, Dembo and Tulowitzki [36] proposed algorithms that allow to add or delete many indices at each iteration. Similar algorithms were presented by Yang and Tolle [101], in which they showed theoretically that the iterations converge to the solution in a finite number of steps. For the solution of ill-posed nonnegative least squares problems, Calvetti et al. [26] proposed a projected iteration method by allowing more consecutive iterations in the inner iteration. In addition, Morigi et al. [74] proposed an active set projected CG method for general box constrained ill-posed problems, which can be applied to nonnegative constrained problems, where the components of the solution that equal their bounds are referred to as the active set and identified in the outer iteration, and the reduced unconstrained least squares problem is solved in the inner iteration by keeping the identified components fixed. These methods are shown to require low storage and are easy to implement, and numerical examples arising from constrained linear ill-posed problems and image restoration indicate their fairly rapid convergence. However, there is few theoretical analysis to guarantee the convergence, and the norm of consecutively generated residual vectors may not be monotonically decreasing [73, 74].

Wright [98] proposed a hybrid two-stage algorithm by using the projected gradient method until a suitable active set is identified in the first stage, and then by applying the CG method to obtain a numerical solution for the current inactive variable set in the second stage. The idea was further developed by Moré and Toraldo [76, 77] for the solution of box constrained quadratic programming, in which the convergence can be guaranteed when the problem is nondegenerate. The active set strategy can be regarded as a subspace acceleration technique, and numerical results show the significant acceleration of convergence.

The algorithm uses conjugate gradient method to explore the face of the feasible region defined by the current iterate, and the gradient projection method to move to a different face. The algorithm [77] is proposed as follows.

Algorithm 2.3.6. Gradient Projection Conjugate Gradient Method

```
1.  For $k = 0, 1, 2, \ldots$ until convergence
2.      Generate gradient projection iterates $y^0, y^1, \ldots$ with $y^0 = x^k$ by
            $$y^{j+1} = P(y^j - \alpha_j \nabla q(y^j)).$$
3.      Set $x^k = y^{j_k}$, where $j_k$ is the first index $j$ that satisfies
            $$\mathscr{A}(y^j) = \mathscr{A}(y^{j-1}),$$
    or
            $$q(y^{j-1}) - q(y^j) \le \eta_2 \max\{q(y^{l-1}) - q(y^l) : 1 \le l < j\}.$$
4.      Generate conjugate gradient iterates $w^0, w^1, \ldots$ with $w^0 = 0$. Set
        $d^k = Z_k w^{j_k}$, where $j_k$ is the first index $j$ that satisfies
            $$q_k(w^{j-1}) - q_k(w^j) \le \eta_1 \max\{q_k(w^{l-1}) - q_k(w^l) : 1 \le l < j\},$$
        and
            $$q_k = \|A(x^k + Z_k w) - b\|_2,$$
        the matrix $Z_k \in \mathbf{R}^{n \times \check{n}}$ be the matrix whose $j$th column is the $i_j$th column
        of the $n \times n$ identity matrix, $j = 1, 2, \ldots, i_{\check{n}}$.
5.      Use a projected search to define $x^{k+1} = P(x^k + \alpha_k d^k)$.
6.      If $\mathscr{B}(x^{k+1}) = \mathscr{A}(x^{k+1})$, continue the conjugate gradient method.
7.  Endfor
```

Moreover, Bardsley and Vogel [20] extended the algorithm of Moré and Toraldo [77] to the solution of non-quadratic, but convex problems with nonnegative constraints in image reconstruction, by taking Newton steps in the inactive variables. The disadvantage of this hybrid method is that when the initial vector is far from the solution, the convergence can be slow as a large number of iterations are needed for the projected gradient method to identify a suitable active set.

2.4 Interior Point Method

Interior point method, or barrier method, is widely used for the solution of the constrained optimization problems. The method solve problems iteratively such that all iterates satisfy the inequality constraints strictly. They approach the solution from either the interior or exterior of the feasible region but never lie on the boundary of this region.

In this section, we will briefly introduce two classes of interior point methods. One is based on the primal-dual path-following methods, which is considered the most successful interior point methods. The other one is based on the penalty method.

2.4.1 Primal Dual Predictor Corrector Interior Point Method

Mehrotra's predictor corrector interior point method [94] is discussed as follows. The first order optimality conditions (2.9) and (2.2) for the NNLS and BLS problems, respectively, can be written as

$$F(x,\lambda) \equiv \begin{bmatrix} A^{\mathrm{T}}Ax - A^{\mathrm{T}}b - \lambda \\ X\Lambda e \end{bmatrix} \equiv \begin{bmatrix} r_d \\ r_{x\lambda} \end{bmatrix} = 0 \tag{2.30}$$

and

$$F(x,\lambda,s) \equiv \begin{bmatrix} A^{\mathrm{T}}Ax - A^{\mathrm{T}}b - \begin{bmatrix} I & -I \end{bmatrix} \lambda \\ s - \begin{bmatrix} x-l \\ u-x \end{bmatrix} \\ SXe \end{bmatrix} \equiv \begin{bmatrix} r_d \\ r_p \\ r_{x\lambda} \end{bmatrix} = 0. \tag{2.31}$$

Set $z = (x,\lambda)$ or $z = (x,\lambda,s)$, we can apply Newton method to solve $F(z) = 0$.

However, the solution of $F(z)=0$ can not guarantee to have the constrained solutions.

$$F(x,\lambda)=\begin{bmatrix}0\\ \tau e\end{bmatrix} \quad \text{and} \quad F(x,\lambda,s)=\begin{bmatrix}0\\0\\ \tau e\end{bmatrix}. \tag{2.32}$$

Applying Newton method to $F(z)-\tau\hat{e}=0$, we have the primal-dual interior point method.

Next, we apply damped Newton method to solve the nonlinear equations (2.30) and (2.31). Consider the solution of $F(x)=0$, the damped Newton method is based on the following iterations

$$F'(x^k)\Delta x=-F(x^k), \quad x^{k+1}=x^k+\alpha_k\Delta x, \quad k=0,1,2,\dots \tag{2.33}$$

The damped simplified Newton method is based on the following iterations

$$F'(x^0)\Delta x=-F(x^k), \quad x^{k+1}=x^k+\alpha_k\Delta x, \quad k=0,1,2,\dots \tag{2.34}$$

The level-m composite Newton method is based on the following iterations

$$\begin{aligned}
&F'(x^k)\Delta x^0=-F(x^k)\\
&F'(x^k)\Delta x^1=-F(x^k+\Delta x^0)\\
&F'(x^k)\Delta x^2=-F(x^k+\Delta x^0+\Delta x^1)\\
&\dots\dots\\
&F'(x^k)\Delta x^m=-F(x^k+\Delta x^0+\dots+\Delta x^{m-1})\\
&x^{k+1}=x^k+\alpha_k(\Delta x^0+\Delta x^1+\dots+\Delta x^m), \quad k=0,1,2,\dots
\end{aligned} \tag{2.35}$$

Specifically, the level-1 composite Newton method is widely used as follows.

$$\begin{aligned}
&F'(x^k)\Delta x^N=-F(x^k)\\
&F'(x^k)\Delta x^s=-F(x^k+\Delta x^N)\\
&x^{k+1}=x^k+\alpha_k(\Delta x^N+\Delta x^s), \quad k=0,1,2,\dots
\end{aligned} \tag{2.36}$$

Applying the level-1 composite Newton method to $F(z) - \tau\hat{e} = 0$, we have the predictor corrector primal-dual interior point method.

Remark that the Jacobian matrix are

$$J(x,\lambda) = \begin{bmatrix} A^{\mathrm{T}}A & -I \\ \Lambda & X \end{bmatrix} \quad \text{and} \quad J(x,\lambda,s) = \begin{bmatrix} A^{\mathrm{T}}A & -I & I & 0 & 0 \\ -I & 0 & 0 & I & 0 \\ I & 0 & 0 & 0 & I \\ 0 & S_{11} & S_{12} & \Lambda_{11} & \Lambda_{12} \\ 0 & S_{21} & S_{22} & \Lambda_{21} & \Lambda_{22} \end{bmatrix} \tag{2.37}$$

for NNLS and BLS problems, respectively.

2.4.2 Penalty-Type Interior Point Method

The following interior point trust-region-based method was proposed by Rojas and Steihaug [89], and was applied for the solution of constrained ill-posed problems in [75].

For $x = [\xi_1, \xi_2, \ldots, \xi_n] > 0$, define

$$f_\gamma(x) = \frac{1}{2}\|Ax - b\|_2^2 - \gamma \sum_{i=1}^{n} \log \xi_i, \tag{2.38}$$

where $\gamma \geq 0$. Consider the solution of the unconstrained minimization problem $\min f_\gamma(x)$, we apply the Newton method $\nabla^2 f_\gamma(x^k)\Delta x = -\nabla f_\gamma(x^k)$, where

$$\begin{aligned} \nabla f_\gamma(x) &= A^{\mathrm{T}}Ax - A^{\mathrm{T}}b - \gamma X^{-1}e \\ \nabla^2 f_\gamma(x) &= A^{\mathrm{T}}A + \gamma X^{-2}, \end{aligned}$$

and $X = \mathrm{diag}(x)$ and $e = [1, 1, \ldots, 1]^{\mathrm{T}}$. Hence, for each iteration k, we compute

$$(A^{\mathrm{T}}A + \gamma X_k^{-2})\Delta x = -(A^{\mathrm{T}}Ax^k - A^{\mathrm{T}}b - \gamma X_k^{-1}e)$$

and set $x^{k+1} = x^k + \alpha_k \Delta x$. This can be further written in the form of the first kind normal equations

$$\begin{bmatrix} A \\ \sqrt{\gamma}X_k^{-1} \end{bmatrix}^{\mathrm{T}} \begin{bmatrix} A \\ \sqrt{\gamma}X_k^{-1} \end{bmatrix} \Delta x = \begin{bmatrix} A \\ \sqrt{\gamma}X_k^{-1} \end{bmatrix}^{\mathrm{T}} \begin{bmatrix} b - Ax^k \\ \sqrt{\gamma}e \end{bmatrix}$$

or the equivalent least squares problem

$$\min \left\| \begin{bmatrix} A \\ \sqrt{\gamma} X_k^{-1} \end{bmatrix} \Delta x - \begin{bmatrix} b - Ax^k \\ \sqrt{\gamma} e \end{bmatrix} \right\|_2 .$$

The LSQR method, which is an implementation of the conjugate gradient method applied to the normal equation based on the Lanczos bidiagonalization, is used for solving the above least squares problem in [75]. In addition, a few auxiliary quantities are computed to facilitate the constraints and the evaluation of the stopping criteria.

2.5 Other Iterative Methods

In this section, we briefly introduce the reflective Newton method and the recent developed flexible Krylov subspaces method for the solution of NNLS and BLS.

2.5.1 Reflective Newton Method

Instead of using projection, the shrinkage of the step size or barrier penalty function to ensure feasibility, the reflective Newton method generates descent directions and follows a piecewise linear path, reflecting off constraints as they are encountered. The method was proposed by Coleman and Li for solving the general nonlinear box constrained problem [24] and the quadratic programming [25].

The advantage of the reflective Newton method is that it exhibits strong convergence properties, global and quadratic convergence, and appears to have significant practical potential to solve the large scale problem.

2.5.2 Flexible Krylov Subspaces Method

In order to ensure the nonnegative feasibility, Nagy and Strakos [78] proposed the modified residual norm steepest descent (MRNSD) algorithm for image restoration problems. The MRNSD algorithm utilize the nonlinear transformation

$$x = e^z$$

to preserve the nonnegativity, which is based on the EMLS algorithm developed by Kaufman [56]. The gradient descent method is used for the resulted unconstrained least squares problems, and the step size is chosen in order to impose nonnegativity of the approximate solution at each iteration. Hence, the convergence rate of the MRNSD method is similar to the other gradient descent method or steepest descent method, which is considerably slow.

Recently, Gazzola and Wiaux [43] proposed the flexible Krylov subspaces method for solving NNLS problem arising from the image restoration, by merging the ability of delivering high-quality approximations typical of the MRNSD methods, with a fast convergence typical of Krylov methods for unconstrained problems. By exploiting the potentialities of flexible Krylov subspaces, the new strategy embraces and improves the class of the MRNSD methods.

2.6 Concluding Remarks

We conclude this chapter by summarizing and comparing the above mentioned previous work on NNLS and BLS.

- As the first order methods, the gradient descent-type methods, including the projection gradient method and restricted gradient method, are easy to implement than the Newton-type methods since only the gradient is computed. However, similar to the steepest descent method, it may suffer from the slow convergence due to the zigzag phenomenon when approaching the minimizer if the condition number is large [102]. See Figure 2.2 for details. Therefore, instead of using them to solve the NNLS and BLS directly, the gradient descent-type methods are applied to identify the optimal active constraints quickly in the first stage of the active set methods.

- For the active set methods by Lawson and Hanson [67], Polyak [85] and O'Leary [81], even though the unconstrained least squares problem can be solved by the fast conjugate gradient method once the active set is determined, typically only one variable between active set and passive set is exchanged in each iteration, which is very slow for the sequence to converge. Specifically, the inner iteration is terminated as soon as a component of a computed iterate violates a constraint,

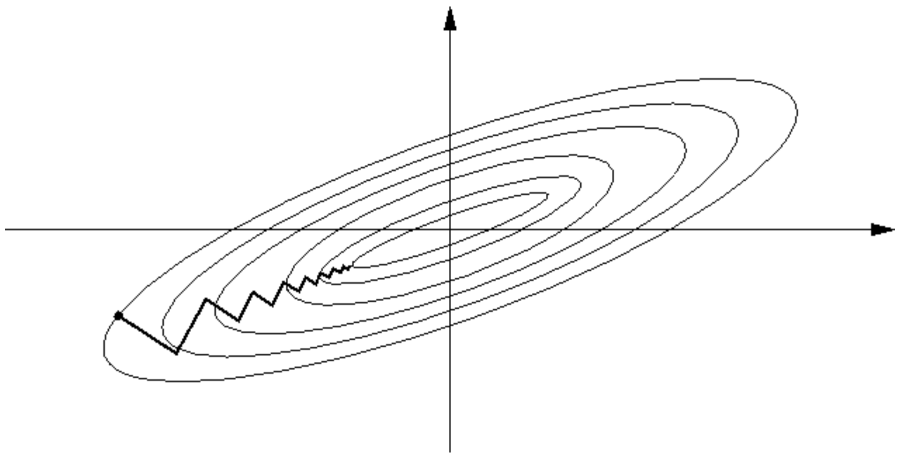


Figure 2.2: The zigzag phenomenon for the gradient descent methods.

which forces frequent resuming of the inner iteration and thus slows down convergence. Also the active set type algorithm allows only one variable to leave a bound at a given outer iteration, which allows to add or delete one index from the active set at a time. This is an inefficient feature when the number of variables is large. The advantage of Polyak and O'Leary's algorithm is that it can terminates in a finite number of iterations theoretically.

- For the the primal dual predictor corrector interior point method, the penalty interior point method, the reflective Newton method, and the flexible Krylov subspaces method, it is not easy to implement. The algorithm not only utilize the barrier function or nonlinear variable transformation to transform the constrained minimization problem to unconstrained minimization problem, but also choose the reasonable step size during each iteration to guarantee the interior property. The advantage of these methods is that they exhibits strong convergence properties, even global and quadratic convergence rate.

- The active set method proposed by Moré and Toraldo was efficient for the large sparse NNLS and BLS problems, since it allows more consecutive iterations for the inner iteration and many elements to exchange between the active set and inactive set for the outer iteration. The disadvantage is that the projection gradient method is used in the first stage to identify the active set, which may be slow. We will utilize the two stages hybrid idea of it and construct the new algorithm by replacing the projection gradient method with other efficient methods.

In next chapter, instead of using shrinking step size or the projection techniques, we apply a modulus transformation to constrain the nonnegativity of the variable, and the solution of NNLS problem can be replaced by the solution of a sequence of unconstrained least squares problems, for which numerous efficient solvers can be exploited. The modulus transformation is similar to the piecewise linear transformation introduced in the reflective Newton method [24, 25]. Therefore, a new class of inner outer iterative methods is proposed by using CGLS method for inner iterations and the modulus iterative method in the outer iterations for the solution of LCP (linear complementarity problem) resulting from the KKT (Karush-Kuhn-Tucker) conditions. Theoretical convergence analysis is presented, and the choice of the parameter matrix is discussed for the proposed method.

We also propose a corresponding hybrid algorithm by incorporating the active set strategy, which contains two stages where the first stage consists of modulus iterations to identify the active set, while the second stage solves the reduced unconstrained least squares problems only on the inactive variables, and projects the solution into the nonnegative region. Such active set method resembles the algorithm proposed by Moré and Toraldo [76, 77]and Bardsley and Vogel [20].

CHAPTER 3

NONNEGATIVE CONSTRAINED LEAST SQUARES PROBLEM

In this charter, we consider the solution of large sparse nonnegative constrained linear least squares (NNLS) problems. A new iterative method is proposed which uses the CGLS method for the inner iterations and the modulus iterative method for the outer iterations to solve the linear complementarity problem resulting from the Karush-Kuhn-Tucker condition of the NNLS problem. Theoretical convergence analysis including the optimal choice of the parameter matrix is presented for the proposed method. In addition, the method can be further enhanced by incorporating the active set strategy, which contains two stages: the first stage consists of modulus iterations to identify the active set, while the second stage solves the reduced unconstrained least squares problems only on the inactive variables, and projects the solution into the nonnegative region. Numerical experiments show the efficiency of the proposed methods compared to projection gradient-type methods with less iteration steps and CPU time.

The rest of the chapter is organized as follows. In Section 3.1, the modulus inner outer iteration method is proposed for the solution of the NNLS problem. In Section 3.3, convergence analysis of the proposed method is presented, and the choice of the parameter matrix is discussed. In Section 3.4, a hybrid method is proposed by alternately performing the modulus iterations and the active set CGLS iterations. In Section 3.5, numerical results are presented, and Section 3.6 gives concluding remarks.

This chapter is mainly based on the contents in [103].

3.1 Modulus Iterative Methods

In this section, we show that the solution of the NNLS problem can be transformed to a series of unconstrained least squares problems by applying a modulus transformation on the variables. First, the equivalence between the nonnegative constrained quadratic programming and the linear complementarity problem (LCP) is shown in the following theorem, when the coefficient matrix is symmetric positive semidefinite.

Theorem 3.1.1. *The nonnegative constrained quadratic programming problem* NNQP(B,c)

$$\min_{x\in\mathbf{R}^n}\left(\frac{1}{2}x^TBx+c^Tx\right)\quad \textit{subject to}\quad x\geq\mathbf{0} \tag{3.1}$$

is equivalent to the linear complementarity problem LCP(B,c)

$$x\geq\mathbf{0},\quad Bx+c\geq\mathbf{0},\quad \textit{and}\quad x^T(Bx+c)=0, \tag{3.2}$$

provided that B is a symmetric positive semidefinite matrix.

Proof. If x^* is a solution of LCP(B,c), then it holds that

$$x^*\geq\mathbf{0},\quad Bx^*+c\geq\mathbf{0},\quad \text{and}\quad (x^*)^{\mathrm{T}}(Bx^*+c)=0.$$

It is observed that for any $x\geq 0$,

$$\begin{aligned}
&\frac{1}{2}x^{\mathrm{T}}Bx+c^{\mathrm{T}}x-\left(\frac{1}{2}(x^*)^{\mathrm{T}}Bx^*+c^{\mathrm{T}}x^*\right)\\
=\;&\frac{1}{2}(x-x^*)^{\mathrm{T}}B(x-x^*)+x^{\mathrm{T}}(Bx^*+c)-(x^*)^{\mathrm{T}}(Bx^*+c)\\
=\;&\frac{1}{2}(x-x^*)^{\mathrm{T}}B(x-x^*)+x^{\mathrm{T}}(Bx^*+c)\geq 0.
\end{aligned}$$

The last inequality holds by the fact that B is symmetric positive semidefinite. Hence, we have

$$\frac{1}{2}x^{\mathrm{T}}Bx + c^{\mathrm{T}}x \geq \frac{1}{2}(x^*)^{\mathrm{T}}Bx^* + c^{\mathrm{T}}x^*,$$

which indicates that x^* is a minimization solution of $\mathrm{NNQP}(B,c)$.

If x^* is a solution of $\mathrm{NNQP}(B,c)$, then x^* satisfies the necessary KKT conditions as follows. There exists $f \in \mathbf{R}^n$, called KKT multipliers, such that

Stationarity

$$\nabla\left(\frac{1}{2}x^{\mathrm{T}}Bx + c^{\mathrm{T}}x - f^{\mathrm{T}}x\right)\bigg|_{x=x^*} = Bx^* + c - f = \mathbf{0},$$

Primal and Dual feasibility

$$x^* \geq \mathbf{0}, \quad f \geq \mathbf{0},$$

Complementarity slackness

$$(x^*)^{\mathrm{T}}f = 0.$$

By collecting the KKT conditions above, it is derived that x^* satisfies $\mathrm{LCP}(B,c)$. ■

Corollary 3.1.2. *If matrix B is symmetric positive definite, then both* $\mathrm{NNQP}(B,c)$ *and* $\mathrm{LCP}(B,c)$ *have the same unique solution.*

Corollary 3.1.3. *([8]) The NNLS problem* (1.1) *is equivalent to* $\mathrm{LCP}(A^TA, -A^Tb)$

$$x \geq \mathbf{0}, \quad \lambda \equiv A^TAx - A^Tb \geq \mathbf{0}, \quad \textit{and} \quad x^T\lambda = 0. \tag{3.3}$$

Proof. Set $B = A^{\mathrm{T}}A$ and $c = -A^{\mathrm{T}}b$ in Theorem 3.1.1. ■

Furthermore, the following theorem, which is a special case of Theorem 2.1 in [4], implies that $\mathrm{LCP}(A^{\mathrm{T}}A, -A^{\mathrm{T}}b)$ is equivalent to the implicit fixed-point equation

$$(\Omega + A^{\mathrm{T}}A)z = (\Omega - A^{\mathrm{T}}A)|z| + A^{\mathrm{T}}b \tag{3.4}$$

with modulus transformation $x = z + |z|$, where Ω is a positive diagonal parameter matrix. Hence, it is equivalent to solve the implicit fixed-point equation (3.4) for the solution of (1.1) by Corollary 3.1.3.

Theorem 3.1.4. *Let Ω be an $n \times n$ positive diagonal matrix. For the* $\mathrm{LCP}(A^TA, -A^Tb)$*, the following statements hold:*

(i) *if x is a solution of the* $\mathrm{LCP}(A^TA, -A^Tb)$*, then* $z = (x - \Omega^{-1}\lambda)/2$ *satisfies the implicit fixed-point equation* (3.4)*, where* $\lambda = A^TAx - A^Tb$*;*

(ii) *if z satisfies the implicit fixed-point equation* (3.4)*, then* $x = |z| + z$ *is a solution of the* $\mathrm{LCP}(A^TA, -A^Tb)$*. Moreover,* $\lambda = \Omega(|z| - z)$ *holds.*

Based on the equivalence in Theorem 4.3.1, the modulus-type iterative scheme

$$(\Omega + A^{\mathrm{T}}A)z^{k+1} = (\Omega - A^{\mathrm{T}}A)|z^k| + A^{\mathrm{T}}b \tag{3.5}$$

is naturally derived for the solution of the fixed-point equation (3.4). If z^* is a fixed point of (3.5), then by Corollary 3.1.3 and Theorem 4.3.1, the solution of the NNLS problem (1.1) can be obtained straightforwardly by $x^* = z^* + |z^*|$. Therefore, the solution of the NNLS problem (1.1) is transformed to the solution of a series of fixed-point equations (3.5), which can be solved directly by matrix decompositions, or by iterative methods, such as the preconditioned CG method as the coefficient matrix $\Omega + A^{\mathrm{T}}A$ is symmetric positive definite.

The modulus iteration method for NNLS problem (1.1) is described as follows.

Algorithm 3.1.5. Modulus Iteration Method

```
1.   Choose an initial approximate solution $z^0$ and a parameter matrix $\Omega$.
2.   Compute $x^0 = z^0 + |z^0|$.
3.   For $k = 0, 1, 2, \ldots$ until convergence
4.       Compute $z^{k+1}$ by solving equation (3.5).
5.       Compute $x^{k+1} = z^{k+1} + |z^{k+1}|$.
6.   Endfor
```

We remark that this modulus method derived from (3.4) is a special case of modulus-based matrix splitting methods with $M = A^{\mathrm{T}}A$ and $N = \mathbf{0}$ in [4]. For more numerical methods for LCP, see [104] and the references therein.

Another remark is that a similar idea that transforms the constrained minimization problem into an unconstrained problem using the parameterization $x = e^z$ instead of $x = z + |z|$ is proposed by Hanke, Nagy and Vogel [51], and is applied for image reconstruction problems in [78]. The main difference is the iterative methods constructed in [78] are based on the solution of fixed-point equation with respect to x, and thus line search is needed at each iteration to maintain nonnegativity, while it is not necessary for the modulus iteration, since the iteration is based on the unconstrained vector z.

Finally, it is noted that the iterative scheme (3.5) can be reorganized as the normal equations

$$\widetilde{A}^{\mathrm{T}}\widetilde{A}z^{k+1} = \widetilde{A}^{\mathrm{T}}\tilde{b}^{k}, \tag{3.6}$$

of the unconstrained least squares problem

$$\min_{z^{k+1}\in\mathbf{R}^n} \|\widetilde{A}z^{k+1} - \tilde{b}^{k}\|_2 \tag{3.7}$$

for any fixed $k = 0, 1, 2, ...$, where

$$\widetilde{A} = \begin{bmatrix} A \\ \Omega^{\frac{1}{2}} \end{bmatrix} \quad \text{and} \quad \tilde{b}^{k} = \begin{bmatrix} -A|z^k| + b \\ \Omega^{\frac{1}{2}}|z^k| \end{bmatrix},$$

Therefore, the solution of the NNLS problem (1.1) is transformed to the solution of a series of unconstrained least squares problems (3.7). This is the main idea of modulus method.

The modulus-type inner outer iteration method for NNLS problem (1.1) is described as follows.

Algorithm 3.1.6. Modulus-Type Inner Outer Iteration Method

```
1.  Choose an initial approximate solution $z^0$ and a parameter matrix $\Omega$.
2.  Compute $x^0 = z^0 + |z^0|$ and $r^0 = b - Ax^0$.
3.  Set
        $\widetilde{A} = \begin{bmatrix} A \\ \Omega^{\frac{1}{2}} \end{bmatrix}$ and $\tilde{r}^0 \equiv \tilde{b}^0 - \widetilde{A}z^0 = \begin{bmatrix} r^0 \\ \Omega^{\frac{1}{2}}(|z^0| - z^0) \end{bmatrix}$
4.  For $k = 0, 1, 2, \ldots$ until convergence
5.      Compute an approximate solution $w^{k+1}$ by solving
            $\min_{w\in\mathbf{R}^n} \|\widetilde{A}w - \tilde{r}^k\|_2.$   (3.8)
6.      Compute $z^{k+1} = z^k + w^{k+1}$.
7.      Compute $x^{k+1} = z^{k+1} + |z^{k+1}|$ and $r^{k+1} = b - Ax^{k+1}$.
```

```
8.          Set
                $\tilde{r}^{k+1} = \begin{bmatrix} r^{k+1} \\ \Omega^{\frac{1}{2}}(|z^{k+1}| - z^{k+1}) \end{bmatrix}$
9.      Endfor
```

Here, the iterative solution of the unconstrained least squares problems (3.8) for each $k = 0, 1, 2, \ldots$ is referred to as the inner iteration of the algorithm, while the for loop is referred to as the outer iteration. Note that Algorithms 3.1.5 and 3.1.6 are mathematically equivalent, since the solution of the inner unconstrained least squares problems (3.8) is equivalent to the solution of the normal equations (3.5), which can be solved by efficient solvers like (preconditioned) CGLS method [8], or BA-GMRES method [53] with inner iteration preconditioning [72].

For the iterative solution of the NNLS problem (1.1), we define the residual as

$$\mathrm{Res}(x^k) \equiv \min(\lambda^k, x^k) = \min(A^{\mathsf{T}}Ax^k - A^{\mathsf{T}}b, x^k), \tag{3.9}$$

and set the stopping criterion as

$$\frac{\|\mathrm{Res}(x^k)\|_2}{\|\mathrm{Res}(x^0)\|_2} < tol \tag{3.10}$$

with initial vector x^0 and given tolerance tol. The definition (3.9) shows that $\mathrm{Res}(x^*) = \mathbf{0}$ if and only if x^* is a solution of the NNLS problem (1.1) by Corollary 3.1.3. Meanwhile, for the iterative solution of the unconstrained least squares problems (3.8), the stopping criterion is set as

$$\frac{\|s^k\|_2}{\|s^0\|_2} = \frac{\|A^{\mathsf{T}}(b - Ax^k)\|_2}{\|A^{\mathsf{T}}(b - Ax^0)\|_2} < tol, \tag{3.11}$$

where $s^k = -\lambda^k$ is the residual of the normal equation (1.4). Note that (3.10) and (3.11) are used as stopping criteria of outer and inner iterations in Algorithm 3.1.6, respectively.

3.2 Review of Modulus Methods

Before establishing the convergence theory of Algorithm 3.1.5, in this section we briefly review and compare the different modulus methods, and show the advantages of methods proposed in the previous section.

By equivalently reformulating the LCP(B,c) (3.2) as an implicit fixed-point equation

$$(I+B)z=(I-B)|z|-c$$

with modulus transformation $x=z+|z|$, where I is a identity matrix, van Bokhoven [95] presented a modulus method which has polynomial complexity when A is symmetric positive definite. Kappel and Watson [60] extended the modulus method to a class of nonsymmetric matrix. see also Section 9.2 in Murty [71]. Moreover, Bai [4] presented a class of modulus-based matrix splitting methods which not only provided a general framework for the modified modulus method [33]

$$(\alpha I+B)z=(\alpha I-B)|z|-c$$

with modulus transformation $x=z+|z|$, and nonstationary extrapolated modulus algorithms [52]

$$(I+\alpha B)z=(I-\alpha B)|z|-c$$

with modulus transformation $x=\alpha(z+|z|)$, but also yielded a series of modulus-based relaxation methods which outperform the projected relaxation methods as well as the modified modulus method in computing efficiency.

Specifically, Bai [4] established the following implicit fixed-point equation

$$(M\Gamma+\Omega_1)z=(N\Gamma-\Omega_2)z+(\Omega-B\Gamma)|z|-c \tag{3.12}$$

to construct modulus-based matrix splitting iteration method for solving the LCP(B,c), where $B=M-N$ is a splitting of the matrix $A\in\mathbf{R}^{n\times n}$, $\Omega=\Omega_1+\Omega_2$ and Γ are $n\times n$ positive diagonal matrices. Let $B=D-L-U$, where D, L and U are the diagonal, the strictly lower-triangular and the strictly upper-triangular matrices of the matrix A. By setting $\Omega_1=\Omega$, $\Omega_2=0$ and $\Gamma=(1/\gamma)I$, equation (3.12) yields a series of modulus-based matrix splitting iteration methods. For instance, when $M=D$ and $N=L+U$, it gives the modulus-based Jacobi (MJ) iteration method

$$(D+\Omega)z^{(k+1)}=(L+U)z^{(k)}+(\Omega-B)|z^{(k)}|-\gamma c;$$

when $M=D-L$ and $N=U$, it gives the modulus-based Gauss-Seidel (MGS) iteration method

$$(D+\Omega-L)z^{(k+1)}=Uz^{(k)}+(\Omega-B)|z^{(k)}|-\gamma c;$$

when $M=(1/\alpha)D-L$ and $N=(1/\alpha-1)D+U$, it gives the modulus-based successive overrelaxation (MSOR) iteration method

$$(D+\alpha\Omega-\alpha L)z^{(k+1)}=[(1-\alpha)D+\alpha U]z^{(k)}+\alpha(\Omega-B)|z^{(k)}|-\alpha\gamma c.$$

For MSOR iteration method, suppose $z_j^{(k+1)}$, $j=1,2,\ldots,i-1$ were known already, then $z_i^{(k+1)}$ is calculated by

$$\begin{aligned}(a_{ii}+\alpha\omega_{ii})z_i^{(k+1)} &= (1-\alpha)a_{ii}z_i^{(k)}+\alpha(-\sum_{j=1}^{i-1}a_{ij}z_j^{(k+1)}-\sum_{j=i+1}^{n}a_{ij}z_j^{(k)}-\gamma c_i)\\ &+ \alpha\sum_{j=1}^{n}(\omega_{ij}-a_{ij})|z_j^{(k)}|.\end{aligned}$$

It would intuitively seem very attractive to use the latest estimates $|z_j^{(k+1)}|$, $j=1,2,\ldots,i-1$, in the right hand side of the subsequent computations

$$\begin{aligned}(a_{ii}+\alpha\omega_{ii})z_i^{(k+1)} &= (1-\alpha)a_{ii}z_i^{(k)}+\alpha(-\sum_{j=1}^{i-1}a_{ij}z_j^{(k+1)}-\sum_{j=i+1}^{n}a_{ij}z_j^{(k)}-\gamma c_i)\\ &+ \alpha\sum_{j=1}^{i-1}(\omega_{ij}-a_{ij})|z_j^{(k+1)}|+\alpha\sum_{j=i}^{n}(\omega_{ij}-a_{ij})|z_j^{(k)}|,\end{aligned}$$

which may accelerate the convergence performance of the MSOR method. In addition, new iterative method reduces the storage requirements by saving $z^{(k+1)}$ instead of saving both $z^{(k)}$ and $z^{(k+1)}$ in MSOR method.

Based on the fixed-point equation (3.12) and the idea mentioned above, a new general implicit fixed-point equation

$$(M_1\Gamma+\Omega_1)z=(N_1\Gamma-\Omega_2)z+(\Omega-M_2\Gamma)|z|+N_2\Gamma|z|-c \tag{3.13}$$

can be established by Zheng and Yin [104], where utilizing the matrix splitting, which is essential for establishing the accelerated modulus-based matrix splitting methods for solving the LCP(B,c). The equivalence of fixed-point equation (3.13) and LCP(B,c) is described in the following theorem, and its proof can be easily obtained by Theorem 2.1 in [4].

Theorem 3.2.1. *Let $B = M_1 - N_1 = M_2 - N_2$ be two splittings of the matrix $B \in \mathbf{R}^{n\times n}$, Ω_1 and Ω_2 be $n \times n$ nonnegative diagonal matrices, and Ω and Γ be $n \times n$ positive diagonal matrices such that $\Omega = \Omega_1 + \Omega_2$. For the* LCP$(B,c)$*, the following statements hold true:*

(i) *if (λ,x) is a solution of the* LCP(B,c)*, then $z = (\Gamma^{-1}x - \Omega^{-1}\lambda)/2$ satisfies the implicit fixed-point equation* (3.13)*;*

(ii) *if z satisfies the implicit fixed-point equation* (3.13)*, then*

$$x = \Gamma(|z| + z) \quad \textit{and} \quad \lambda = \Omega(|z| - z) \tag{3.14}$$

is a solution of the LCP(B,c).

Theorem 3.2.1 can be used to introduce a general matrix splitting iteration method for solving the LCP(B,c). However, as implicit fixed-point equation (3.13) involves many arbitrary parameters that are quite complicated to be determined in actual computation, it would be more convenient to obtain the simplified implicit fixed-point equation

$$(M_1 + \widetilde{\Omega})z = N_1 z + (\widetilde{\Omega} - M_2)|z| + N_2|z| - \gamma c$$

by specifically setting

$$\Omega_1 = \Omega, \quad \Omega_2 = 0 \quad \text{and} \quad \Gamma = \frac{1}{\gamma} I,$$

where $\gamma > 0$ and $\widetilde{\Omega} = \gamma\Omega$. Note that $\widetilde{\Omega}$ is also a positive diagonal matrix, it can be replaced by Ω for simplicity without causing any confusion. Based on this fixed-point equation, the following accelerated modulus-based matrix splitting iteration method for solving the LCP(B,c) is established.

Algorithm 3.2.2. *(***Accelerated modulus-based matrix splitting iteration method for** LCP(B,c)*)* *Let $B = M_1 - N_1 = M_2 - N_2$ be two splittings of the matrix $B \in \mathbf{R}^{n\times n}$. Given* *an initial vector $z^{(0)} \in \mathbf{R}^n$, for $k = 0, 1, 2, \dots$ until the iteration sequence $\{x^{(k)}\}_{k=0}^{+\infty}$ is convergent, compute $z^{(k+1)} \in \mathbf{R}^n$ by solving the linear system*

$$(M_1 + \Omega)z^{(k+1)} = N_1 z^{(k)} + (\Omega - M_2)|z^{(k)}| + N_2|z^{(k+1)}| - \gamma c \tag{3.15}$$

and set

$$x^{(k+1)} = \frac{1}{\gamma}(|z^{(k+1)}| + z^{(k+1)}).$$

Here, Ω is an $n \times n$ positive diagonal matrix and γ is a positive constant.

Algorithm 3.2.2 provides a general framework of accelerated modulus-based matrix splitting iteration methods for solving the LCP(B, c). Besides including the modulus-based matrix splitting iteration methods when $M_2 = B$ and $N_2 = 0$ studied in [4] as the special cases, it also yields a series of accelerated modulus-based matrix splitting methods. For example, when

$$M_1 = \frac{1}{\alpha}(D - \beta L),\ N_1 = \frac{1}{\alpha}[(1-\alpha)D + (\alpha - \beta)L + \alpha U],\ M_2 = D - U \quad \text{and} \quad N_2 = L,$$

it gives the accelerated modulus-based accelerated overrelaxation (AMAOR) iteration method

$$\begin{array}{rcl} (D + \alpha\Omega - \beta L)z^{(k+1)} & = & [(1-\alpha)D + (\alpha - \beta)L + \alpha U]z^{(k)} \\ & + & \alpha(\Omega - D + U)|z^{(k)}| + \alpha L|z^{(k+1)}| - \alpha\gamma c. \end{array}$$

It also gives the accelerated modulus-based successive overrelaxation (AMSOR) iteration method, the accelerated modulus-based Gauss-Seidel (AMGS) iteration method and the accelerated modulus-based Jacobi (AMJ) iteration method when $\alpha = \beta$, $\alpha = \beta = 1$ and $\alpha = 1$, $\beta = 0$, respectively. Moveover, it gives the accelerated modulus-based symmetric successive overrelaxation (AMSSOR) iteration method

$$\left\{ \begin{array}{rcl} (D + \alpha\Omega - \alpha L)z^{(k+\frac{1}{2})} & = & [(1-\alpha)D + \alpha U]z^{(k)} \\ & + & \alpha(\Omega - D + U)|z^{(k)}| + \alpha L|z^{(k+\frac{1}{2})}| - \alpha\gamma c \\ (D + \alpha\Omega - \alpha U)z^{(k+1)} & = & [(1-\alpha)D + \alpha L]z^{(k+\frac{1}{2})} \\ & + & \alpha(\Omega - D + L)|z^{(k+\frac{1}{2})}| + \alpha U|z^{(k+1)}| - \alpha\gamma c. \end{array} \right.$$

The convergence of Algorithm 3.2.2 when the system matrix B is either a positive definite matrix or an H_+-matrix is presented in [104, 105].

It is noted that for NNLS problem (1.1), since $B = A^{\mathrm{T}}A$ and $c = -A^{\mathrm{T}}b$, we do not use the explicit matrix splitting of B so that the symmetric structure can be preserved. Moreover, the general accelerated modulus-based matrix splitting iteration schemes can not be rewritten as the unconstrained least squares problems, and thus those efficient least squares solvers will not be applied.

3.3 Convergence Analysis

In this section, we establish the convergence theory of Algorithm 3.1.5 in which the inner unconstrained least squares problems (3.8) are solved based on the normal equations (3.5). Specifically, we will discuss the cases when the inner systems are solved exactly or inexactly, respectively, as well as the theoretically optimal choice of the iteration parameter matrix Ω.

Assume that $z^* \in \mathbf{R}^n$ is a solution of the implicit fixed-point equation (3.4), i.e.,

$$(\Omega + A^{\mathrm{T}}A)z^* = (\Omega - A^{\mathrm{T}}A)|z^*| + A^{\mathrm{T}}b, \tag{3.16}$$

and z^{k+1} is computed exactly from z^k by solving (3.5). After subtracting (3.16) from (3.5), we obtain

$$z^{k+1} - z^* = (\Omega + A^{\mathrm{T}}A)^{-1}(\Omega - A^{\mathrm{T}}A)(|z^k| - |z^*|), \tag{3.17}$$

provided that $\Omega + A^{\mathrm{T}}A$ is nonsingular. The relationship (3.17) is the basis for us to establish convergence theorems about Algorithm 3.1.5. The following analysis is based on the condition that A is of full column rank and thus $A^{\mathrm{T}}A$ is symmetric positive definite. Similar techniques were used to analyse other iterative methods in [12, 91].

3.3.1 Scalar matrix case

Consider the case when $\Omega = \omega I$ with $\omega > 0$. It follows from taking vector norm $\|\cdot\|_2$ of both sides of (3.17) that

$$\begin{aligned}\|z^{k+1} - z^*\|_2 &\le \|(\omega I + A^{\mathrm{T}}A)^{-1}(\omega I - A^{\mathrm{T}}A)\|_2 \| |z^k| - |z^*| \|_2 \\ &\le \|(\omega I + A^{\mathrm{T}}A)^{-1}(\omega I - A^{\mathrm{T}}A)\|_2 \|z^k - z^*\|_2\end{aligned}$$

It can be easily shown that $(\omega I + A^{\mathrm{T}}A)^{-1}(\omega I - A^{\mathrm{T}}A)$ is symmetric. Therefore,

$$\|(\omega I + A^{\mathrm{T}}A)^{-1}(\omega I - A^{\mathrm{T}}A)\|_2 = \max_{\lambda_i \in \sigma(A^{\mathrm{T}}A)} \left| \frac{\omega - \lambda_i}{\omega + \lambda_i} \right|,$$

where $\sigma(A^{\mathrm{T}}A)$ denotes the set of all eigenvalues of $A^{\mathrm{T}}A$. As A is of full column rank, it follows that $\lambda_i > 0$ and

$$\left| \frac{\omega - \lambda_i}{\omega + \lambda_i} \right| < 1,$$

for any i, and thus

$$\|(\omega I + A^{\mathrm{T}}A)^{-1}(\omega I - A^{\mathrm{T}}A)\|_2 < 1.$$

Consequently, the iteration sequence $\{z^k\}_{k=0}^{+\infty}$ generated by (3.5) converges to the unique solution z^* for any initial vector.

Let $\lambda_{\min}$ and $\lambda_{\max}$ be the minimum and maximum eigenvalues of $A^{\mathrm{T}}A$, respectively. It can be easily shown that the optimal ω^* is

$$\omega^* \equiv \arg\min_{\omega} \left\{ \max_{\lambda_{\min} \le \lambda \le \lambda_{\max}} \left| \frac{\omega - \lambda}{\omega + \lambda} \right| \right\} = \sqrt{\lambda_{\min}\lambda_{\max}}$$

and

$$\|(\omega^* I + A^{\mathrm{T}}A)^{-1}(\omega^* I - A^{\mathrm{T}}A)\|_2 = \frac{\sqrt{\lambda_{\max}} - \sqrt{\lambda_{\min}}}{\sqrt{\lambda_{\max}} + \sqrt{\lambda_{\min}}} = \frac{\sqrt{\kappa(A^{\mathrm{T}}A)} - 1}{\sqrt{\kappa(A^{\mathrm{T}}A)} + 1},$$

where $\kappa(A^{\mathrm{T}}A)$ denotes the spectral condition number of matrix $A^{\mathrm{T}}A$.

3.3.2 General positive diagonal matrix case

Consider the general case when Ω is a positive diagonal matrix. We define two vector norms and a matrix norm that are useful in the following discussions. For all $x \in \mathbf{R}^n$,

$\|x\|_Q \equiv \sqrt{x^{\mathrm{T}}Qx}$ and $\|x\|_{P,q} \equiv \|Px\|_q$ define vector norms on $\mathbf{R}^n$, where $Q \in \mathbf{R}^{n\times n}$ is an arbitrary symmetric positive definite matrix, $P \in \mathbf{R}^{n\times n}$ is an arbitrary nonsingular matrix and q is a positive integer. Moveover, if $X \in \mathbf{R}^{n\times n}$, then $\|X\|_{P,q} \equiv \|PXP^{-1}\|_q$; see [4, 82]. It follows from taking vector norm $\|\cdot\|_{\Omega^{1/2},2}$ of both sides of (3.17) that

$$\|z^{k+1} - z^*\|_{\Omega^{1/2},2} \le \|(\Omega + A^{\mathrm{T}}A)^{-1}(\Omega - A^{\mathrm{T}}A)\|_{\Omega^{1/2},2}\||z^k| - |z^*|\|_{\Omega^{1/2},2}. \tag{3.18}$$

Note that

$$\|z^{k+1} - z^*\|_{\Omega^{1/2},2} = \|\Omega^{1/2}(z^{k+1} - z^*)\|_2 = \|z^{k+1} - z^*\|_\Omega$$

and

$$\begin{aligned}
& \|(\Omega + A^{\mathrm{T}}A)^{-1}(\Omega - A^{\mathrm{T}}A)\|_{\Omega^{1/2},2} \\
= \; & \|\Omega^{-1/2}(I + (A\Omega^{-1/2})^{\mathrm{T}}(A\Omega^{-1/2}))^{-1}\Omega^{-1/2}\Omega^{1/2}(I - (A\Omega^{-1/2})^{\mathrm{T}}(A\Omega^{-1/2}))\Omega^{1/2}\|_{\Omega^{1/2},2} \\
= \; & \|(I + (A\Omega^{-1/2})^{\mathrm{T}}(A\Omega^{-1/2}))^{-1}(I - (A\Omega^{-1/2})^{\mathrm{T}}(A\Omega^{-1/2}))\|_2 \\
\equiv \; & \|(I + \widehat{A}^{\mathrm{T}}\widehat{A})^{-1}(I - \widehat{A}^{\mathrm{T}}\widehat{A})\|_2,
\end{aligned}$$

where $\widehat{A} \equiv A\Omega^{-1/2}$. Therefore, (3.18) gives

$$\begin{aligned}
\|z^{k+1} - z^*\|_\Omega &\le \|(I + \widehat{A}^{\mathrm{T}}\widehat{A})^{-1}(I - \widehat{A}^{\mathrm{T}}\widehat{A})\|_2\||z^k| - |z^*|\|_\Omega \\
&\le \|(I + \widehat{A}^{\mathrm{T}}\widehat{A})^{-1}(I - \widehat{A}^{\mathrm{T}}\widehat{A})\|_2\|z^k - z^*\|_\Omega.
\end{aligned}$$

Notice that $\widehat{A} \equiv A\Omega^{-1/2}$ is of full column rank as A is of full column rank and Ω is a positive diagonal matrix. Hence, $\widehat{A}^{\mathrm{T}}\widehat{A}$ is symmetric positive definite and

$$\|(I + \widehat{A}^{\mathrm{T}}\widehat{A})^{-1}(I - \widehat{A}^{\mathrm{T}}\widehat{A})\|_2 = \max_{\hat{\lambda}_i \in \sigma(\widehat{A}^{\mathrm{T}}\widehat{A})} \left|\frac{1 - \hat{\lambda}_i}{1 + \hat{\lambda}_i}\right| < 1.$$

Consequently, the iteration sequence $\{z^k\}_{k=0}^{+\infty}$ generated by (3.5) converges to the unique solution z^* for any initial vector.

Next, the choice of the parameter matrix Ω is discussed. Set $\Omega = \bar{\omega}D$, where $D \equiv \mathrm{diag}(A^{\mathrm{T}}A)$ denotes the diagonal part of $A^{\mathrm{T}}A$ and $\bar{\omega}$ is a positive scalar parameter.

Then $\widehat{A} = \bar{\varpi}^{-1/2} A D^{-1/2} \equiv \bar{\varpi}^{-1/2} \bar{A}$ and

$$\begin{aligned} & \|(I + \widehat{A}^{\mathrm{T}} \widehat{A})^{-1} (I - \widehat{A}^{\mathrm{T}} \widehat{A})\|_2 \\ = \; & \|(I + \bar{\varpi}^{-1} \bar{A}^{\mathrm{T}} \bar{A})^{-1} (I - \bar{\varpi}^{-1} \bar{A}^{\mathrm{T}} \bar{A})\|_2 \\ = \; & \|(\bar{\varpi} I + \bar{A}^{\mathrm{T}} \bar{A})^{-1} (\bar{\varpi} I - \bar{A}^{\mathrm{T}} \bar{A})\|_2. \end{aligned}$$

Similar to the previous analysis, the optimal parameter can be obtained by

$$\bar{\varpi}^* = \sqrt{\bar{\lambda}_{\min} \bar{\lambda}_{\max}},$$

where $\bar{\lambda}_{\min}$ and $\bar{\lambda}_{\max}$ are the minimum and maximum eigenvalues of $\bar{A}^{\mathrm{T}} \bar{A}$, respectively. In addition,

$$\|(\bar{\varpi}^* I + \bar{A}^{\mathrm{T}} \bar{A})^{-1} (\bar{\varpi}^* I - \bar{A}^{\mathrm{T}} \bar{A})\|_2 = \frac{\sqrt{\bar{\lambda}_{\max}} - \sqrt{\bar{\lambda}_{\min}}}{\sqrt{\bar{\lambda}_{\max}} + \sqrt{\bar{\lambda}_{\min}}} = \frac{\sqrt{\kappa(\bar{A}^{\mathrm{T}} \bar{A})} - 1}{\sqrt{\kappa(\bar{A}^{\mathrm{T}} \bar{A})} + 1},$$

where $\kappa(\bar{A}^{\mathrm{T}} \bar{A})$ denotes the spectral condition number of matrix $\bar{A}^{\mathrm{T}} \bar{A}$.

Remark that $\bar{A}^{\mathrm{T}} \bar{A} = D^{-1/2} A^{\mathrm{T}} A D^{-1/2}$ can be regarded as a symmetric diagonal scaling preconditioning of $A^{\mathrm{T}} A$. Hence, it may be more efficient to choose $\Omega = \omega D$ than to choose $\Omega = \omega I$ in the modulus iteration Algorithm 3.1.5.

3.3.3 Convergence of inexact inner iteration

Finally, the convergence analysis based on the inexact solution of the implicit fixed-point equation (3.5) is considered. Suppose z^k has already been computed. Then, z^{k+1} is computed by applying iterative methods, such as the PCG method, to (3.5). Thus, we have

$$(\Omega + A^{\mathrm{T}} A) z^{k+1} = (\Omega - A^{\mathrm{T}} A)|z^k| + A^{\mathrm{T}} b + e^k, \tag{3.19}$$

where e^k denotes the error of inner iteration. Note that $e^k = 0$ for some fixed k indicates that the inner iteration is solved exactly. In addition, we define the error of outer iteration

$$\varepsilon^k = (\Omega + A^{\mathrm{T}} A) z^k - (\Omega - A^{\mathrm{T}} A)|z^k| - A^{\mathrm{T}} b.$$

Note that if $\varepsilon^k = 0$ for some fixed k, then $x^* = x^k$ is an exact solution of the fixed-point equation (3.4).

Assume that $\|e^k\|_\Omega \le \gamma_k \|\varepsilon^k\|_\Omega$ with $\gamma_k < 1$, which indicates that the error of inner iteration is controlled by the error of outer iteration. Then, it follows by subtracting (3.16) from (3.19) that

$$\begin{aligned}
& \|z^{k+1} - z^*\|_\Omega \\
= \;& \|(\Omega + A^{\mathsf T}A)^{-1}(\Omega - A^{\mathsf T}A)(|z^k| - |z^*|) + (\Omega + A^{\mathsf T}A)^{-1} e^k\|_\Omega \\
\le \;& \|(\Omega + A^{\mathsf T}A)^{-1}(\Omega - A^{\mathsf T}A)\|_{\Omega^{1/2},2} \||z^k| - |z^*|\|_\Omega + \|(\Omega + A^{\mathsf T}A)^{-1}\|_{\Omega^{1/2},2} \|e^k\|_\Omega \\
\le \;& \|(\Omega + A^{\mathsf T}A)^{-1}(\Omega - A^{\mathsf T}A)\|_{\Omega^{1/2},2} \|z^k - z^*\|_\Omega + \gamma_k \|(\Omega + A^{\mathsf T}A)^{-1}\|_{\Omega^{1/2},2} \|\varepsilon^k\|_\Omega \\
= \;& \|(\Omega + A^{\mathsf T}A)^{-1}(\Omega - A^{\mathsf T}A)\|_{\Omega^{1/2},2} \|z^k - z^*\|_\Omega \\
+ \;& \gamma_k \|(\Omega + A^{\mathsf T}A)^{-1}\|_{\Omega^{1/2},2} \|(\Omega + A^{\mathsf T}A) z^k - (\Omega - A^{\mathsf T}A)|z^k| - A^{\mathsf T}b\|_\Omega \\
= \;& \|(\Omega + A^{\mathsf T}A)^{-1}(\Omega - A^{\mathsf T}A)\|_{\Omega^{1/2},2} \|z^k - z^*\|_\Omega \\
+ \;& \gamma_k \|(\Omega + A^{\mathsf T}A)^{-1}\|_{\Omega^{1/2},2} \|(\Omega + A^{\mathsf T}A)(z^k - z^*) - (\Omega - A^{\mathsf T}A)(|z^k| - |z^*|)\|_\Omega \\
\le \;& \|(\Omega + A^{\mathsf T}A)^{-1}(\Omega - A^{\mathsf T}A)\|_{\Omega^{1/2},2} \|z^k - z^*\|_\Omega \\
+ \;& \gamma_k \|(\Omega + A^{\mathsf T}A)^{-1}\|_{\Omega^{1/2},2} (\|\Omega + A^{\mathsf T}A\|_{\Omega^{1/2},2} + \|\Omega - A^{\mathsf T}A\|_{\Omega^{1/2},2}) \|z^k - z^*\|_\Omega \\
=: \;& L_k \|z^k - z^*\|_\Omega .
\end{aligned}$$

Hence, we only need to verify that $L_k \le \theta < 1$, where θ is a scalar constant independent of k.

Set

$$\begin{aligned}
\tau &\equiv \|(\Omega + A^{\mathsf T}A)^{-1}\|_{\Omega^{1/2},2} \|\Omega + A^{\mathsf T}A\|_{\Omega^{1/2},2}, \\
\delta &\equiv \|(\Omega + A^{\mathsf T}A)^{-1}(\Omega - A^{\mathsf T}A)\|_{\Omega^{1/2},2}, \\
\mu &\equiv \|(\Omega + A^{\mathsf T}A)^{-1}\|_{\Omega^{1/2},2} \|\Omega - A^{\mathsf T}A\|_{\Omega^{1/2},2}.
\end{aligned}$$

By the fact that $\delta < 1$, we have

$$\theta \equiv \alpha + (1 - \alpha)\delta < 1,$$

where $0 \leq \alpha < 1$. If there exists an integer k_0 such that for all $k \geq k_0$,

$$L_k = \delta + \gamma_k(\tau + \mu) \leq \theta \quad \Rightarrow \quad \gamma_k \leq \frac{\alpha(1-\delta)}{\tau + \mu},$$

then it follows that for $k \geq k_0$, $L_k \leq \theta < 1$, which guarantees the convergence of the iteration sequence $\{z^k\}_{k=0}^{+\infty}$ generated by inexact modulus iteration for any initial vector.

Combining the analysis above, we have the following theorem.

Theorem 3.3.1. *If A is of full column rank, then the iteration sequence $\{x^k\}_{k=0}^{\infty}$ generated by modulus-type inner outer iteration Algorithm 3.1.6 converges to the unique solution x^* for any initial vector when*

- *the inner system is solved exactly;*
- *or the inner system is solved iteratively with*

$$\|e^k\|_\Omega \leq \gamma_k \|\varepsilon^k\|_\Omega \quad \text{with} \quad \gamma_k \leq \frac{\alpha(1-\delta)}{\tau + \mu},$$

  for $k \geq k_0$, where k_0 is an integer and $0 \leq \alpha < 1$.

3.4 Two-Stage Hybrid Iterative Methods with Active Set Strategy

In this section, we propose a two-stage hybrid algorithm that resembles the algorithm of Moré and Toraldo [77] and Bardsley and Vogel [20] by using modulus iterations to identify the active set in the first stage, and the CGLS method to solve unconstrained least squares subproblem only on the current inactive variables in the second stage, and alternately performing these two stages until convergence.

Suppose x^k is the kth iterative solution. Then the active set is defined as

$$\mathscr{A}(x^k) = \{j : x_j^k = 0\}, \tag{3.20}$$

and the binding set is defined as

$$\mathscr{B}(x^k) = \{j : x_j^k = 0,\ \lambda_j^k \geq 0\}, \tag{3.21}$$

where $\lambda^k = -A^{\mathrm{T}} r^k = -A^{\mathrm{T}}(b - Ax^k)$ is the Lagrange multiplier, which is also the gradient vector of the least-squares objective function in (1.3), or the negative residual of the normal equation (1.4). The corresponding set of free variables is defined as

$$\mathscr{F}(x^k) = \{1, 2, ..., n\} \backslash \mathscr{A}(x^k). \tag{3.22}$$

It can be easily obtained from Corollary 3.1.3 that x^* is the solution of the NNLS problem (1.1) if and only if $\mathscr{A}(x^*) = \mathscr{B}(x^*)$ and $\forall j \notin \mathscr{B}(x^*)$, $x_j^* > 0$ and $\lambda_j^* = 0$.

In the first stage, the modulus inner outer iteration method in Algorithm 3.1.6 is applied to perform a fast update of the active set. By choosing $y^0 = x^{k+1}$, a sequence of iterates $\{y^j\}_{j=0}^{\infty}$ is generated until either the modulus iterations fails to update a new active set when

$$\mathscr{A}(y^j) = \mathscr{A}(y^{j-1}) \tag{3.23}$$

is satisfied, or it fails to make sufficient progress in decreasing the objective function value when

$$|l(y^{j-1}) - l(y^j)| \le \eta_1 \max\{|l(y^{i-1}) - l(y^i)| : 1 \le i < j\} \tag{3.24}$$

is satisfied, where $0 < \eta_1 < 1$ is a given tolerance and $l(y)$ is defined in (2.1).

Next, we discuss the details of the active set method in the second stage. Let $i_1, i_2, ..., i_{\check{n}}$ be the elements in $\mathscr{F}(x^k)$, and the matrix $Z_k \in \mathbf{R}^{n \times \check{n}}$ be the matrix whose jth column is the i_jth column of the $n \times n$ identity matrix, $j = 1, 2, ..., i_{\check{n}}$. Then, the CGLS method is used to compute the minimization subproblem

$$\min_{w \in \mathbf{R}^{\check{n}}} \|A(x^k + Z_k w) - b\|_2. \tag{3.25}$$

Denote $A_{\mathscr{F}} := AZ_k$ as the submatrix of A consisting of the columns of A whose indices belong to $\mathscr{F}$. Then, the subproblem (3.25) is equivalent to the reduced unconstrained least squares problem

$$\min_{w \in \mathbf{R}^{\check{n}}} l_k(w) \equiv \|A_{\mathscr{F}} w - r^k\|_2. \tag{3.26}$$

Note that if $\mathscr{A}(x^k) = \mathscr{A}(x^*)$, where x^* is a solution of the NNLS (1.1) and w^* is the solution of (3.26), then $x^* = x^k + Z_k w^*$. In a word, if the constraints active at the exact solution x^* are known in advance, then the NNLS problem can be solved by simply optimizing the least-square function in an unconstrained manner over only the variables that correspond to the inactive constraints.

Given an initial vector $w^{k+1,0}$, let the CGLS method generate a sequence of iterates until

$$l_k(w^{k+1,j-1}) - l_k(w^{k+1,j}) \le \eta_2 \max\{l_k(w^{k+1,i-1}) - l_k(w^{k+1,i}) : 1 \le i < j\} \tag{3.27}$$

is satisfied, where $0 < \eta_2 < 1$ is a given tolerance. The stopping criterion (3.27) indicates that the CGLS method fails to make sufficient progress at the jth step. After setting $w^{k+1} \equiv w^{k+1,j}$, in general we do not set $x^{k+1} = x^k + Z_k w^{k+1}$ since this may produce negative elements in x^{k+1}. Similar to the strategy used in Algorithm 2.2.1, we set

$$x^{k+1} = P(x^k + \beta^m Z_k w^{k+1}), \tag{3.28}$$

and find the smallest integer $m \ge 0$ that satisfies the sufficient decrease condition

$$\|b - Ax^{k+1}\|_2^2 \le \|b - Ax^k\|_2^2 - 2\mu (s^k)^{\mathrm{T}}(x^{k+1} - x^k), \tag{3.29}$$

where $0 < \beta < 1$ and $0 \le \mu < 1$. It can be easily calculated that (3.29) is equivalent to

$$(x^{k+1} - x^k)^{\mathrm{T}}((2\mu - 1)s^k - s^{k+1}) \le 0, \tag{3.30}$$

which is used in the practical algorithm to avoid evaluating the objective function.

Due to the projection operation in (3.28), more elements in x^{k+1} are constrained to zero and thus $\mathscr{A}(x^k) \subseteq \mathscr{A}(x^{k+1})$. It can be observed that if $\mathscr{A}(x^{k+1}) = \mathscr{A}(x^*)$, then $\mathscr{B}(x^{k+1}) = \mathscr{A}(x^{k+1})$. Otherwise there exists at least one index $i \in \mathscr{A}(x^{k+1})$ and $i \notin \mathscr{B}(x^{k+1})$, such that $x_i^{k+1} = 0$ and $\lambda_i^{k+1} < 0$. Note that it is possible to optimize the objective further by making $x_i^{k+1} > 0$ and $\lambda_i^{k+1} = 0$ at the next iteration, and thereby $x_i^* \neq 0$. Therefore, $\mathscr{B}(x^{k+1}) = \mathscr{A}(x^{k+1})$ is a necessary condition for $x^{k+1} = x^*$. In the second stage of the hybrid algorithm, if $\mathscr{B}(x^{k+1}) = \mathscr{A}(x^{k+1})$ holds, we update the active set and then resume the CGLS iterations until either the exact solution x^* of NNLS (1.1) is obtained, or the condition $\mathscr{B}(x^{k+1}) = \mathscr{A}(x^{k+1})$ is violated. If the latter case occurs, we go to the first stage.

The two-stage hybrid modulus active set CGLS method is described as follows.

Algorithm 3.4.1. Hybrid Modulus Active Set CGLS Method

```
1.    Choose an initial approximate solution $x^0$ and compute $r^0 = b - Ax^0$.
2.    For $k = 0, 1, 2, \dots$ until convergence
3.        First Stage Choose $y^0 = x^k$ and generate $\{y^j\}_{j=0}^{\infty}$ by modulus inner
            outer iterations until either (3.23) or (3.24) is satisfied.
4.          Set $x^k = y^j$ and compute $r^k = b - Ax^k$.
5.        Second Stage Update $\mathscr{A}(x^k)$ and $\mathscr{F}(x^k)$ and then solve the reduced
            subproblem (3.26) by CGLS method until (3.27) is satisfied.
6.          Compute $x^{k+1}$ using (3.28) with sufficient decrease condition (3.29).
7.        If $\mathscr{B}(x^{k+1}) = \mathscr{A}(x^{k+1})$, set $x^k = x^{k+1}$ and resume the Second Stage;
          otherwise go to the First Stage.
8.    Endfor
```

Here, the modulus iterations in the first stage and the CGLS iterations for the unconstrained least squares problems (3.26) in the second stage for each $k = 0, 1, 2, \dots$ are referred to as the inner iteration of the algorithm, while the for loop is referred to as the outer iteration. Since the modulus Algorithm 3.1.6 is an inner outer iteration method, the whole Algorithm 3.4.1 contains three-level iterations. The stopping criterion for the outer iteration is set as (3.10), while for inner iteration, (3.23), (3.24) and (3.27) are used for two stages, respectively.

Remark that if the modulus inner outer iteration method in the first stage is replaced by the projected gradient method, then the above algorithm is the gradient projection conjugate gradient (GPCG) method proposed by Moré and Toraldo [77].

The convergence of Algorithm 3.4.1 is proved in the following. The idea of the proof comes from the proof of convergence of the general multilevel algorithm for optimization problems. See [62] and the references therein.

Theorem 3.4.2. *If A is of full column rank, and the modulus Algorithm 3.1.6 generates a nonincreasing objective function sequence, then the iteration sequence $\{x^k\}_{k=0}^{\infty}$ generated by the hybrid modulus active set CGLS iteration Algorithm 3.4.1 converges to the unique solution x^* for any initial vector.*

Proof. Since A is of full column rank, $l(x)$ defined in (2.1) is a strictly convex quadratic function. If $\{x^k\}_{k=0}^{\infty}$ is the iteration sequence generated by hybrid modulus active set

CGLS iteration Algorithm 3.4.1, then $\{l(x^k)\}_{k=0}^{\infty}$ is a nonincreasing sequence by the fact that the modulus algorithm generates a nonincreasing objective function sequence and the sufficient decrease condition in (3.29). Therefore, $\{x^k\}_{k=0}^{\infty}$ is bounded. In addition, Theorem 3.3.1 guarantees that every limit point of $\{x^k\} \subseteq \mathscr{K}_{mod}$, where $\mathscr{K}_{mod}$ is the set of iterates generated by the modulus iteration, is a stationary point of the NNLS (1.1). Hence, there exists a subsequence $x^{k_j} \to x^*$, and thus $l(x^k) \to l(x^*)$.

Next we prove that $x^k \to x^*$. Note that the NNLS solution x^* satisfies the variational inequality

$$(x-x^*)^{\mathrm{T}}\nabla l(x^*) = (x-x^*)^{\mathrm{T}}(A^{\mathrm{T}}Ax^* - A^{\mathrm{T}}b) \geq 0,$$

for any $x \geq 0$. Hence,

$$\begin{aligned} l(x^k) - l(x^*) &= \frac{1}{2}(x^k - x^*)^{\mathrm{T}}A^{\mathrm{T}}A(x^k - x^*) + (x^k - x^*)^{\mathrm{T}}(A^{\mathrm{T}}Ax^* - A^{\mathrm{T}}b) \\ &\geq \frac{\lambda_{\min}}{2}\|x^k - x^*\|_2^2, \end{aligned}$$

where $\lambda_{\min} > 0$ is the minimum eigenvalues of $A^{\mathrm{T}}A$. We can easily obtain $x^k \to x^*$ for $k \to \infty$. ■

Remark that the assumption of generating a nonincreasing sequence in the modulus inner outer iteration method can be easily satisfied, if we modify step 6 in Algorithm 3.1.6 as $z^{k+1} = z^k + \beta^m w^{k+1}$ and choose the smallest integer $m \geq 0$ that satisfies the sufficient decrease condition (3.29).

3.5 Numerical Experiments

Finally, numerical experiment results are presented to show the performance of the modulus-type inner outer iteration Algorithm 3.1.6 and the two-stage hybrid modulus active set CG iteration Algorithm 3.4.1. We compare them to the projected gradient Algorithm 2.2.1, the projected NR-SOR Algorithm 2.2.2 and the GPCG method in [77] which replaces the modulus method in the first stage of Algorithm 3.4.1 with the projected gradient method, for overdetermined NNLS problems.

All the computations were run on a personal computer with 2.20 GHz CPU and 2 GB memory. The programming language is Matlab 7.8 with machine precision $\varepsilon = 1.1 \times 10^{-16}$. The initial vectors for the outer and inner iterations were chosen to be

Table 3.1: Abbreviations for the compared methods.

Method	Description
PG	Projected gradient method (Algorithm 2.2.1)
PSOR	Projected NR-SOR method (Algorithm 2.2.2)
Mod	Modulus method (Algorithm 3.1.6 with $\Omega = \omega I$)
GMod	Modulus method (Algorithm 3.1.6 with $\Omega = \omega \mathrm{diag}(A^{\mathsf{T}}A)$)
GPCG	Hybrid projected gradient active set CG method [77]
ModASCG	Hybrid modulus active set CG method (Algorithm 3.4.1 with $\Omega = \omega I$)
GModASCG	Hybrid modulus active set CG method (Algorithm 3.4.1 with $\Omega = \omega \mathrm{diag}(A^{\mathsf{T}}A)$)

zero vector. For the modulus-type iteration methods, the parameter matrix was chosen to be $\Omega = \omega I$ or $\Omega = \omega \mathrm{diag}(A^{\mathsf{T}}A)$, where ω is a positive parameter. The abbreviations for all the compared methods are listed in Table 3.1. Remark that all the inner least squares problems are solved by the CGLS method, which is easy to implement and needs small storage requirement.

As mentioned in (3.10), the stopping criterion for the outer iteration of all methods is chosen as $\|\mathrm{Res}(x^k)\|_2/\|\mathrm{Res}(x^0)\|_2 < tol$. For one stage methods including PG, PSOR, Mod and GMod, the stopping criterion for the inner unconstrained least squares problems is chosen as (3.11)

$$\frac{\|s^k\|_2}{\|s^0\|_2} = \frac{\|A^{\mathsf{T}}(b - Ax^k)\|_2}{\|A^{\mathsf{T}}(b - Ax^0)\|_2} < tol_{in} \equiv 10^{-2}/k,$$

which means the accuracy required for the solution of the inner systems is refined with the increase of the outer iterations k. Different tolerances tol are chosen for different numerical problems. For two-stage methods including GPCG, ModASCG and GModASCG, the stopping criteria for the inner iterations are chosen as (3.27), (3.23) and (3.24). In order to perform a fair comparison among different methods, the parameters in (3.27), (3.24) and the sufficient decrease condition (3.29) are chosen as

$$\eta_1 = \eta_2 = 0.1, \quad \mu = 0.1 \quad \text{and} \quad \beta = 0.9 \tag{3.31}$$

for all methods. In addition, the maximum number of iteration steps is restricted to be 10,000.

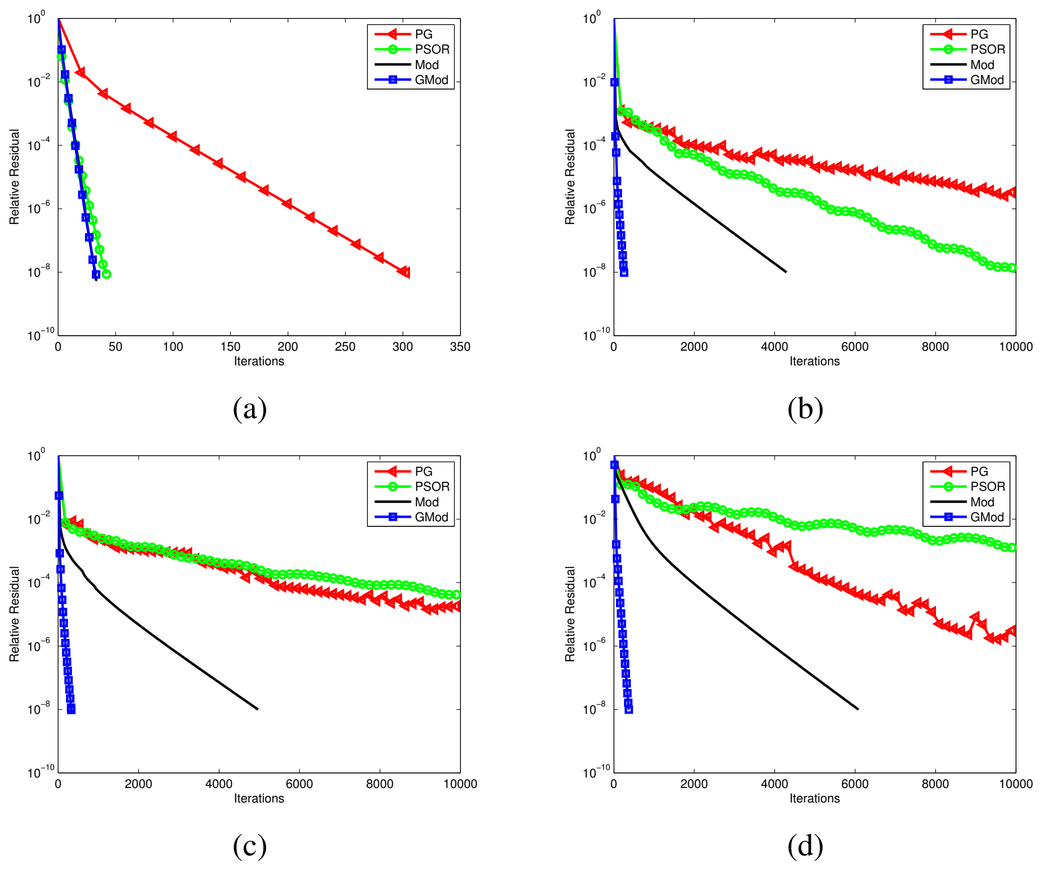
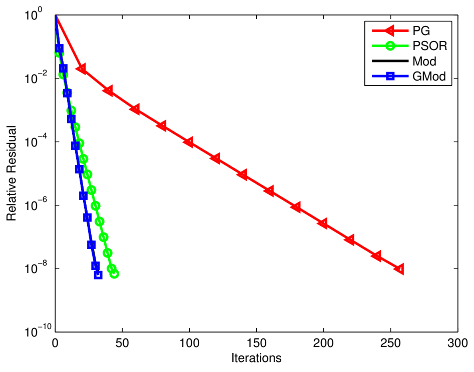


Figure 3.1: Relative residual vs. iterations for (a) $\rho = 1$, (b) $\rho = 0.9$, (c) $\rho = 0.8$ and (d) $\rho = 0.7$ with $\sigma_n = 0.01$ $(\kappa(A) = 100)$.

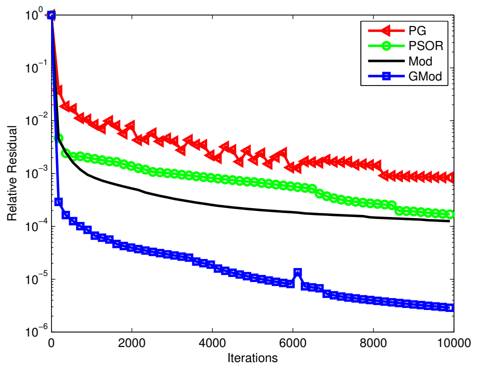
In the following, we compare the numerical methods on four examples, which are dense full rank case, sparse full rank case, sparse rank deficient case and nonnegative image restoration problems.

3.5.1 Dense full rank case

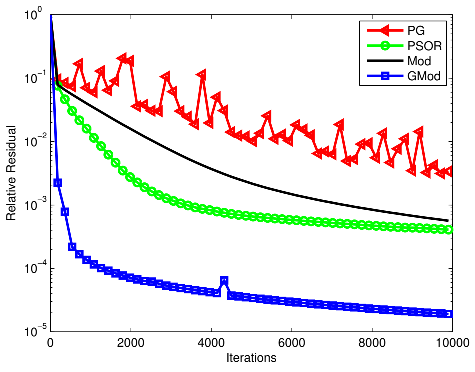
First, we show how the condition number and the distribution of singular values of A influence the convergence of the modulus-type and projection-type methods with a class of dense matrices of the form $A = U\Sigma V^{\mathsf{T}}$, where $U \in \mathbf{R}^{m\times m}$ and $V \in \mathbf{R}^{n\times n}$ are orthogonal matrices obtained from the QR decomposition of random matrices, and $\Sigma \in \mathbf{R}^{m\times n}$ is a rectangular diagonal matrix with diagonal entries $\sigma_1 > \sigma_2 > \ldots > \sigma_n$,



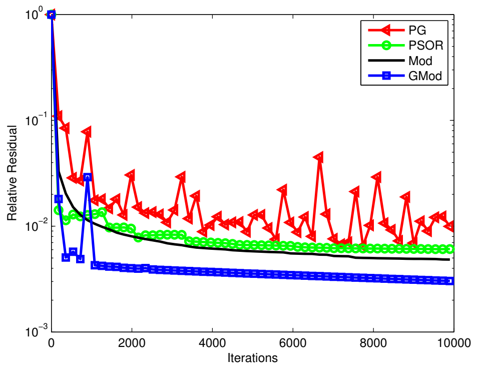
(a)



(b)



(c)



(d)

Figure 3.2: Relative residual vs. iterations for (a) $\rho = 1$, (b) $\rho = 0.9$, (c) $\rho = 0.8$ and (d) $\rho = 0.7$ with $\sigma_n = 0.0001$ ($\kappa(A) = 10^4$).

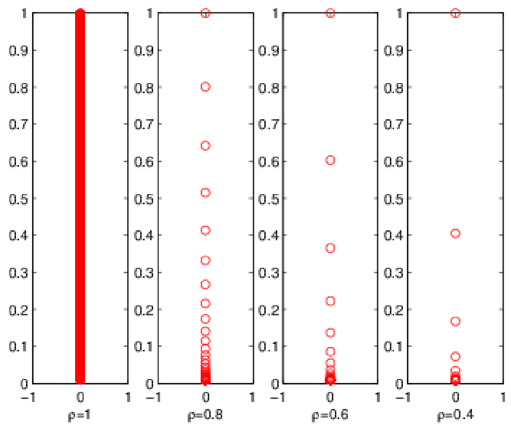


Figure 3.3: The singular values are tightly clustered towards the smallest singular value when ρ decreases.

where the ith smallest singular value is

$$\sigma_{n-i+1} = \sigma_n + \frac{i-1}{n-1}(\sigma_1 - \sigma_n)\rho^{n-i}, \quad i = 1, \dots, n,$$

with the parameter $\rho \in (0,1]$. Note that when ρ decreases, the singular values are tightly clustered towards the smallest singular value σ_n and are far apart towards the largest singular value σ_1. See Figure 3.3 for the illustration. The idea of generating this kind of matrices is from [46, 53].

In our numerical experiments, we set $m = 200,\ n = 100,\ \sigma_1 = 1,\ \sigma_n = 0.01$ or 0.0001, $\rho = 1,\ 0.9,\ 0.8,\ 0.7$, and form inconsistent NNLS problems where the elements of the vector b are generated randomly following the normal distribution with mean zero and variance 1, using the Matlab function $\text{randn}(m,1)$. The same b is used for all the cases.

We compared the four testing methods, PG, PSOR, Mod and GMod, where the inner unconstrained least squares problems (3.8) were solved by backslash “\" in Matlab, The relaxation parameter in the PSOR method were chosen as $\omega_{PSOR} = 1.2$, and the parameters in the Mod and GMod methods were chosen as $\omega = 0.1$.

In Figures 3.1 and 4.1, we depict the curves of the relative residual $\|\text{Res}(x^k)\|_2/\|\text{Res}(x^0)\|_2$ of the testing methods versus the number of iteration steps with $\sigma_n = 0.01$ and $\sigma_n = 0.0001$, respectively. In each figure, there are four diagrams denoted by (a),

Table 3.2: Comparison of the iterative methods (full rank and inconsistent problem with $\kappa(A) = 100$).

	PG	Mod	GMod	GPCG	ModASCG	GModASCG
(a)	302	33	32	2	2	2
	1.00	8.09	8.63	(4.00,5.00)	(3.50,4.50)	(3.50,4.50)
	908	602	618	150	177	181
	0.03	0.01	0.01	$*0.002$	0.004	0.01
(b)	–	4,294	256	20	16	26
		9.26	34.77	(33.45,1.90)	(12.56,1.88)	(4.65,1.69)
		88,144	18,316	5,985	2,446	3,332
		1.31	0.28	0.11	$*0.06$	0.07
(c)	–	4,971	322	20	13	12
		6.51	28.09	(32.20,2.25)	(6.00,2.54)	(4.00,2.67)
		74,632	18,736	7,441	1,671	1,336
		1.08	0.28	0.13	0.05	$*0.04$
(d)	–	6,084	367	18	8	10
		5.03	25.59	(10.89,2.06)	(7.75,3.88)	(4.20,2.70)
		73,332	19,518	1,961	1,175	1,035
		1.06	0.29	0.05	0.03	$*0.03$

First row: number of outer iterations.
Second row: average inner iterations (of the first stage, second stage).
Third row: number of matrix vector multiplications.
Fourth row: computational time in seconds.
($tol = 10^{-8}$, $tol_{in} = 10^{-2}/k$)

(b), (c) and (d) corresponding to $\rho = 1,\ 0.9,\ 0.8$ and 0.7, respectively. The tolerances were chosen as $tol = 10^{-8}$ and $tol = 10^{-5}$ for $\sigma_n = 0.01$ and $\sigma_n = 0.0001$, respectively.

From Figure 3.1, it is observed that the relative residual of the GMod method decreases much more rapidly than any other iterative method as the iteration steps increase for the case when the singular values cluster towards the smallest singular value in (b), (c) and (d). For the case when the singular values are uniformly distributed with $\rho = 1$, Mod and GMod methods show similar convergence performance. This shows that the modulus method with $\Omega = \omega \mathrm{diag}(A^{\mathsf{T}}A)$ outperforms projection-type iterative methods, and the choice of $\Omega = \omega \mathrm{diag}(A^{\mathsf{T}}A)$ in GMod is more efficient than the choice of $\Omega = \omega I$ in Mod, which confirms our convergence analysis.

Table 3.3: Comparison of the iterative methods (full rank and inconsistent problem with $\kappa(A) = 10^4$).

	PG	Mod	GMod	GPCG	ModASCG	GModASCG
	$tol = 10^{-8}$					
(a)	256 1.00 770 0.02	31 8.23 574 0.01	31 8.77 608 0.02	3 (2.33,3.67) 168 $*0.01$	2 (3.50,4.50) 181 0.01	2 (3.50,4.00) 168 $*0.01$
	$tol = 10^{-3}$			$tol = 10^{-8}$		
(b)	7,423 3.41 58,105 0.84	846 9.80 18,282 0.27	72 28.58 4,262 0.06	62 (432.66,2.35) 286,543 5.64	149 (12.17,1.75) 54,595 0.95	443 (3.61,1.84) 42,666 $*0.88$
(c)	–	7,315 7.74 127,820 1.84	261 33.95 18,246 0.27	4,400 (97.65,1.32) 2,462,239 63.36	3,516 (5.50,1.96) 414,452 8.18	842 (7.53,2.03) 246,024 $*4.45$
(d)	–	–	–	–	3,932 (5.84,1.42) 429,613 $*8.20$	2,824 (11.85,1.20) 741,330 13.68

First row: number of outer iterations.
Second row: average inner iterations (of the first stage, second stage).
Third row: number of matrix vector multiplications.
Fourth row: computational time in seconds.

Figure 4.1 shows similar convergence phenomena as in Figure 3.1. As the iteration steps increase, the relative residual of the GMod method decreases much more rapidly than any other iterative method. Note that the convergence behavior of all four methods deteriorate as the condition number of the matrix A increases and the singular values cluster towards σ_n, as can be seen by comparing Figures 3.1 and 4.1. PSOR, Mod and GMod show relatively smooth residual curves, whereas those of PG are oscillatory and the convergence behaviors are erratic.

When the inner least squares problems (3.8) are solved by the CGLS method, in Tables 3.2 and 4.2 we compare the six testing methods, PG, Mod, GMod, GPCG, ModASCG and GModASCG, from the aspects of outer iteration steps, average inner iteration steps, the number of matrix vector multiplications, and CPU time in seconds. Note that for two-stage methods, the average inner iterations of the first and second stages are shown separately. The tolerance for outer iteration is chosen as $tol = 10^{-8}$ except for the cases (b), (c) and (d) in Table 4.2, where the tolerance for PG, Mod and GMod is $tol = 10^{-3}$. The parameters in two-stage methods are chosen as (6.24). In addition, the parameters in Mod, GMod, ModASCG and GModASCG methods are chosen as $\omega = 0.1$.

In Tables 3.2 and 4.2, the symbol "$-$" indicates that the iterative method failed to converge within the maximum iteration steps (10,000), and "$*$" denotes the most efficient method with the least number of matrix vector multiplications and least CPU time among all the testing methods.

From Table 3.2, it is observed that GPCG is more efficient than other iterative methods with less matrix vector multiplications and CPU time when $\rho = 1$. For $\rho < 1$, the two-stage modulus methods including ModASCG and GModASCG outperform other iterative methods with less computational costs, and the PG method fails to converge within the maximum iteration steps. Similar phenomena can be observed in Table 4.2, namely, the modulus type methods require less matrix vector multiplications and CPU time than the projection type methods.

For the one stage methods, GMod converges much faster than PG and Mod with far less outer iterations, more inner iterations, but less computational costs except for the case $\rho = 1$ in Tables 3.2 and 4.2, where Mod requires slightly less matrix vector multiplications. For the two-stage methods, the modulus type methods outperform projection type methods in most cases. However, there is no optimal method between ModASCG and GModASCG. The reason is that the tolerance $\eta_1 = 0.1$ used in the first stage of the two-stage methods are quite large, and thus it is hard to show the advantage of the modulus method with $\Omega = \omega \mathrm{diag}(A^{\mathrm{T}}A)$.

Moreover, it can be observed from Tables 3.2 and 4.2 that the two-stage methods require much less computational costs than the corresponding one-stage methods. This shows that the active set strategy in the second stage of the hybrid algorithm enhances the performance of PG, Mod and GMod. In Table 4.2, when the singular values cluster towards 0, the PG, Mod and GMod methods fail to converge within the maximum iteration steps for $tol = 10^{-8}$. This is the reason why we set $tol = 10^{-3}$ in (b), (c) and

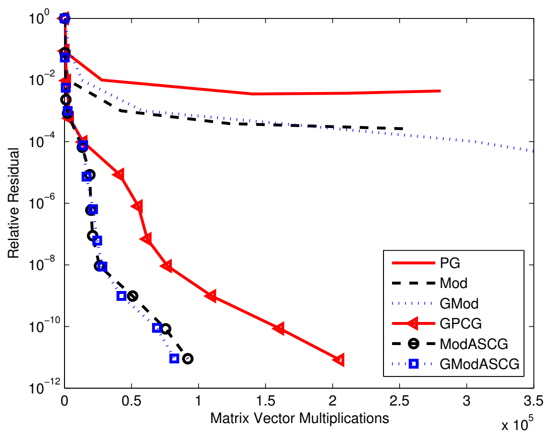


Figure 3.4: Relative residual vs. matrix vector multiplications for Randn_4 (inconsistent).

(d) for all the one-stage numerical methods. In addition, it is further confirmed in Tables 3.2 and 4.2 that the convergence behavior of all methods deteriorate as the condition number of the matrix A increases, as was shown in Figures 3.1 and 4.1. Note that generally the CPU time show positive correlation with matrix vector multiplications, since matrix vector multiplication is the main computational cost in the algorithms. Therefore, the iterative methods with least CPU time have least number of matrix vector multiplications.

3.5.2 Sparse full rank case

Next, we generate a class of large, sparse full column rank matrices, abbreviated as "Randn_i", $i = 1,2,3,4,5,6,7,8$, using the Matlab function "sprandn" with $m = 30,000$, $n = 3,000$, and the ratio of nonzero elements density$= 0.1\%$. The condition numbers of these matrices are specified as

$$\kappa(\text{Randn\_i}) = 10^i, \quad i = 1,2,3,4,5,6,7,8.$$

The nonzero element values were generated by a random number generator following the normal distribution, and the pattern of the nonzero elements is also determined by a random number generator. In these experiments, we form inconsistent NNLS problems where the elements of the right-hand side vector b are generated randomly using the

Matlab function randn$(m,1)$. The same b is used for all the cases.

The numerical results are shown in Table 3.4. The tolerance for outer iterations for PG, Mod and GMod is chosen as $tol = 10^{-5}$, while for GPCG, ModASCG and GModASCG the tolerance is chosen as $tol = 10^{-8}$. The iteration parameters used in modulus type methods are set to be $\omega = 0.1$.

Table 3.4 shows that the GPCG method outperforms the other iterative methods for “Randn_1" and “Randn_2" when the condition number is small, while the two-stage modulus methods achieve better performance than the other methods with less matrix vector multiplications and CPU time for the case when the condition number is larger. This shows that the modulus inner outer iterative method is more effective and efficient than projected gradient methods in identifying a suitable active set in the first stage of the hybrid algorithm. Moreover, the PG and Mod methods could not converge within the maximum outer iteration numbers except for “Randn_1" and “Randn_2". The GMod method converged for all the cases with $tol = 10^{-5}$, although it requires large computational costs compared to the two-stage methods. Similar to the previous numerical experiments, it can be concluded that the active set strategy accelerates the convergence behavior with far less iteration steps and CPU time, and the convergence behavior of all methods deteriorate as the condition number of the matrix A increases.

In Figure 3.4, we plot the relative residual $\|\text{Res}(x^k)\|_2/\|\text{Res}(x^0)\|_2$ of the testing methods versus the number of matrix vector multiplications for Randn_4 inconsistent problem. The residual curves of the two-stage modulus active set iterative methods decline much more rapidly than GPCG and other one-stage iterative methods.

3.5.3 Sparse rank deficient case

In the following, we test a class of rectangular matrices from the University of Florida Sparse Matrix Collection [30]. We construct the rank-deficient overdetermined systems by deleting all the zero rows and zero columns. The resulting number of rows m, columns n, nonzero elements nnz, as well as the rank, are given in Table 3.5.

For the consistent case, we form NNLS problems where the right-hand side vector $b = Ax^*$ and $x^* = [1,1,\dots,1]^{\mathrm{T}} \in \mathbf{R}^n$. The numerical results are shown in Table 3.6, where the tolerance for the outer iteration is chosen to be $tol = 10^{-6}$ for all methods. The optimal μ and ω were chosen for projected gradient type methods and modulus type methods, respectively, so that the number of matrix vector multiplications is minimized. We chose the optimal μ practically by changing it from 0.1 to 0.9, and chose the optimal ω by changing it from 0 to 4 with an interval of 0.1.

Table 3.6 shows that PG fails to converge for all the cases, and the two-stage modulus methods ModASCG or GModASCG require the least matrix vector multiplications and CPU time. Different from the dense full rank case and the sparse full rank case, Mod outperforms GMod with less average inner iterations and CPU time. For Maragal_8, Mod is faster than GPCG when $\omega = 1.3$ is chosen. Apart from Maragal_4, ModASCG is faster than GModASCG. Remark that although it is not guaranteed theoretically, the modulus type methods including Mod, GMod, ModASCG and GModASCG converge for the rank deficient problems. This can be explained as the conditions proposed in the previous convergence analysis are sufficient conditions but not necessary ones.

Figure 3.5 shows the number of outer iterations, inner iterations and matrix vector multiplications vs. ω for the Mod method for Maragal_3. The number of outer iterations decreases at first, and then increases, while the number of inner iterations always decreases as ω increases. The optimal parameter with the least number of matrix vector multiplications was $\omega^* = 0.7$. Note that the dependence on ω is mild for $0.5 \leq \omega \leq 2$. Hence, in practice one may set $\omega = 1.0$.

In Figure 3.6, we plot the relative residual $\|\mathrm{Res}(x^k)\|_2 / \|\mathrm{Res}(x^0)\|_2$ of the testing methods versus the number of matrix vector multiplications for Maragal_5 consistent problem. The residual curves of the two-stage modulus active set iterative methods decline much more rapidly than GPCG and other one-stage iterative methods as shown in Figure 3.4.

For the inconsistent case, we form inconsistent NNLS problems where the elements of the right-hand side vector b are generated randomly using the Matlab function $\mathrm{randn}(m,1)$. The numerical results are shown in Table 3.7, where the tolerance for the outer iterations is chosen to be $tol = 10^{-6}$.

Similar to the phenomena observed in Table 3.6, Table 3.7 shows that PG fails to converge within the maximum iteration steps for all the cases, and Mod outperforms GMod with less average inner iterations and CPU time. It can be concluded that modulus type methods including Mod, ModASCG and GModASCG outperform the projection type methods with less computational costs. When the practically optimal parameters are chosen, even Mod converges faster than GPCG for problems Maragal_4, Maragal_6

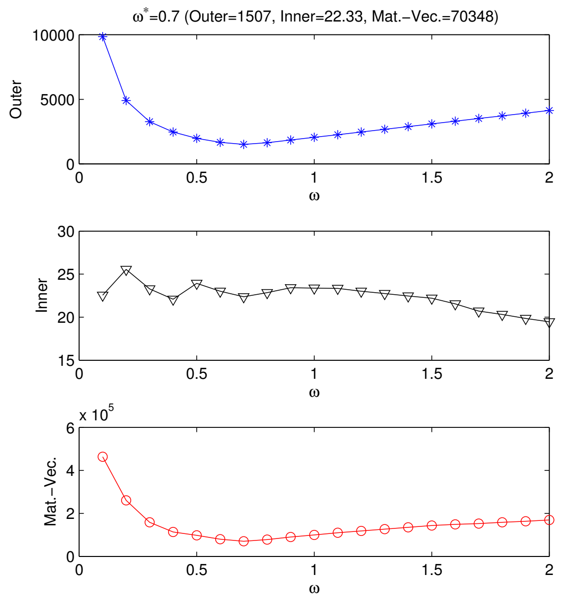


Figure 3.5: Number of outer iterations, average inner iterations and matrix vector multiplication vs. ω for Mod method in Maragal_3 (consistent).

and Maragal_8. Compared with the consistent problems in Table 3.6, the solution of inconsistent problems require more matrix vector multiplications and CPU time. In addition, the convergence behavior of all methods deteriorate as the problem size and the condition number of the matrix A increases.

3.6 Concluding Remarks

In this chapter, a new class of inner outer iterative methods for nonnegative constrained least squares (NNLS) problem (1.1) was proposed based on the modulus transformation for the nonnegative variables. Thus, the solution of the NNLS problem (1.1) can be transformed into the solution of a sequence of unconstrained least squares problems. Theoretical convergence analysis was presented when the inner system is solved either exactly or iteratively, and the choice of the parameter matrix was discussed for the proposed methods. Moreover, we proposed a two-stage hybrid modulus algorithm by incorporating the active set strategy, which contains two stages where the first stage

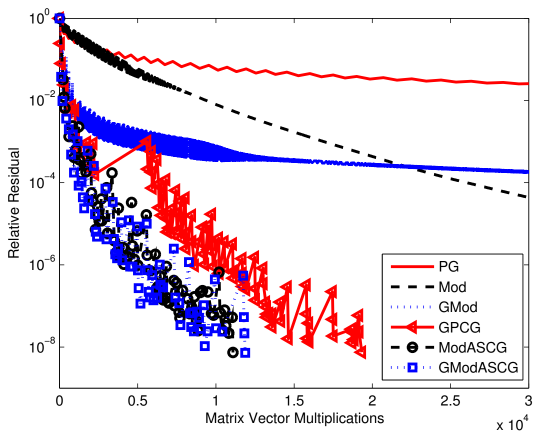


Figure 3.6: Relative residual vs. matrix vector multiplications for Maragal_5 (consistent).

consists of modulus iterations to identify the active set, while the second stage solves the reduced unconstrained least squares problems only on the inactive variables, and projects the solution into the nonnegative region. Numerical experiments show the efficiency of the proposed modulus methods compared to projection gradient-type methods with less iteration steps and CPU time for full column rank and rank deficient overdetermined NNLS problems. The modulus method is not only more efficient for identifying a suitable active set, but also outperforms projection gradient-type methods with less iteration steps and CPU time when the coefficient matrix has ill-determined rank with large condition number and the singular values cluster near zero. We also applied our modulus methods to nonnegative constrained ill-posed image restoration problems, and the numerical results showed that the proposed method gives more accurate results compared to the projected gradient type methods.

In Chapters 5 and 6, we will apply the modulus-type iteration methods for the solution of nonnegative constrained ill-posed problem and the nonnegative matrix factorization problem, respectively. Also, in Chapter 7, we will use an acceleration technique to further accelerate the convergence of modulus-type methods.

Table 3.4: Comparison of the iterative methods (full rank and inconsistent problem).

Problem	PG	Mod	GMod	GPCG	ModASCG	GModASCG
	$tol = 10^{-5}$			$tol = 10^{-8}$		
Randn_1	353	49	16	2	2	2
	6.39	5.08	13.63	(3.50,4.00)	(7.50,4.50)	(2.50,4.00)
	4,869	598	470	179	229	202
	1.61	0.19	0.14	*0.07	0.10	0.09
Randn_2	–	2,618	93	7	4	8
		5.02	60.85	(5.00,4.86)	(19.25,7.75)	(4.25,4.13)
		31,546	11,506	1,419	1,597	1,485
		9.02	3.49	*0.43	0.46	0.45
Randn_3	–	–	540	30	7	4
			143.92	(5.50,6.53)	(33.57,10.14)	(11.75,10.75)
			156,518	11,103	8,458	7,588
			43.21	3.05	2.08	*1.94
Randn_4	–	–	1,523	12	2	3
			146.04	(7.33,27.17)	(263.00,40.50)	(11.67,22.33)
			447,896	60,244	27,919	30,387
			123.02	13.17	*6.66	7.52
Randn_5	–	–	524	137	20	3
			832.20	(2.12,8.99)	(24.30,22.10)	(16.00,38.00)
			873,194	77,495	41,358	19,466
			239.72	19.54	10.03	*5.03
Randn_6	–	–	450	44	22	10
			485.65	(2.55,64.95)	(47.91,37.05)	(6.70,34.90)
			437,988	134,208	99,409	39,087
			120.39	33.87	23.15	*9.40
Randn_7	–	–	1,933	120	88	173
			308.59	(2.15,31.29)	(8.13,21.51)	(4.87,12.93)
			1,196,884	109,427	70,037	85,397
			424.00	31.55	*19.55	24.58
Randn_8	–	–	2,137	26	6	4
			205.73	(3.81,12.03)	(44.50,24.33)	(13.50,27.75)
			883,554	24,088	20,242	17,633
			302.65	5.88	4.74	*4.42

First row: number of outer iterations.
Second row: average inner iterations (of the first stage, second stage).
Third row: number of matrix vector multiplication.
Fourth row: computational time in seconds.

Table 3.5: Information on the practical test matrices.

Problem	m	n	nnz	dens. [%]	rank	$\kappa(A)$
Maragal_3	1,682	858	18,391	1.27	613	1.10×10^3
Maragal_4	1,964	1,027	26,719	1.32	801	9.33×10^6
Maragal_5	4,654	3,296	93,091	0.61	2,147	1.19×10^5
Maragal_6	21,251	10,144	537,694	0.25	8,331	2.91×10^6
Maragal_7	46,845	26,525	1,200,537	0.10	20,843	8.98×10^6
Maragal_8	60,845	33,093	1,308,415	0.06	15,343	1.76×10^9

Table 3.6: Comparison of the iterative methods (rank-deficient and consistent problem).

Problem	PG	Mod	GMod	GPCG	ModASCG	GModASCG
Maragal_3	–	1,507	505	48	39	36
		22.33	664.97	(7.21,1.00)	(4.62,1.00)	(4.56,1.00)
		70,348	672,628	6,346	5,406	6,076
		3.72	35.07	0.42	*0.35	0.40
		0.7	0.1	0.3	1.4	0.7
Maragal_4	–	801	446	26	30	12
		20.14	470.94	(7.31,1.00)	(4.37,1.00)	(4.00,1.00)
		33,870	420,970	2,832	2,746	2,368
		2.37	28.89	0.24	0.24	*0.21
		0.7	0.1	0.1	0.5	0.8
Maragal_5	–	1,106	546	41	28	21
		23.43	2,664.70	(50.20,1.00)	(5.00,1.00)	(6.48,1.00)
		54,040	2,910,980	8,634	3,766	3,738
		12.47	724.04	2.20	*0.95	1.01
		1.4	0.2	0.5	2	0.5
Maragal_6	–	1,972	–	73	93	49
		32.71		(13.93,1.00)	(5.94,1.00)	(6.84,1.00)
		132,966		18,379	13,334	13,958
		174.47		24.58	*18.07	19.22
		0.6		0.3	1.1	0.5
Maragal_7	–	1,927	–	138	79	81
		38.48		(23.77,1.00)	(6.96,1.00)	(10.01,1.00)
		152,142		19,584	11,094	13,612
		510.06		64.24	*35.16	47.35
		0.6		0.7	1.3	0.2
Maragal_8	–	745	–	39	41	36
		20.02		(322.97,1.00)	(8.83,1.00)	(8.89,1.00)
		31,326		44,110	5,908	5,360
		111.63		164.78	*21.91	26.63
		1.3		0.2	0.8	0.4

First row: number of outer iterations.
Second row: average inner iterations (of the first stage, second stage).
Third row: number of matrix vector multiplication.
Fourth row: computational time in seconds.
Fifth row: optimal parameters μ^* for GPCG and ω^* for modulus-type methods.
($tol = 10^{-6}$, $tol_{in} = 10^{-2}/k$)

Table 3.7: Comparison of the iterative methods (rank-deficient and inconsistent problem).

Problem	PG	Mod	GMod	GPCG	ModASCG	GModASCG
Maragal_3	–	875	590	24	13	17
		34.63	578.56	(86.33,1.00)	(16.15,1.08)	(22.94,1.00)
		62,346	683,884	28,446	10,146	13,158
		3.26	35.24	1.57	$*$0.57	0.77
		0.6	0.1	0.1	0.6	0.1
Maragal_4	–	822	436	66	33	26
		35.06	889.40	(152.56,1.00)	(4.82,1.00)	(5.62,1.00)
		59,278	776,430	129,807	5,602	7,424
		4.08	53.17	9.61	$*$0.43	0.56
		0.5	0.1	0.5	3	0.9
Maragal_5	–	649	569	24	14	15
		37.38	2,092.10	(94.58,1.00)	(13.71,1.00)	(26.07,1.00)
		49,818	2,381,966	30,736	12,622	20,090
		11.45	547.63	6.91	$*$2.79	4.64
		0.8	0.2	0.1	0.2	0.1
Maragal_6	–	1,105	–	110	40	63
		43.43		(70.54,1.00)	(4.46,1.00)	(4.92,1.00)
		98,182		123,023	7,548	10,764
		127.39		161.32	$*$9.71	14.64
		0.6		0.8	3.5	0.7
Maragal_7	–	1,210	–	136	51	62
		42.33		(42.97,1.00)	(9.65,1.00)	(4.52,1.00)
		104,864		67,490	19,062	11,674
		359.51		238.37	57.22	$*$39.92
		0.8		0.6	3	0.5
Maragal_8	–	735	–	122	3,716	2,121
		39.34		(182.32,1.00)	(2.59,1.00)	(4.55,1.00)
		59,296		381,421	116,542	106,202
		$*$223.67		2,508.94	519.81	476.51
		0.8		0.3	1.3	1.9

First row: number of outer iterations.
Second row: average inner iterations (of the first stage, second stage).
Third row: number of matrix vector multiplication.
Fourth row: computational time in seconds.
Fifth row: optimal parameters μ^* for GPCG and ω^* for modulus-type methods.
$(tol = 10^{-6},\ tol_{in} = 10^{-2}/k)$

CHAPTER 4

BOX CONSTRAINED LEAST SQUARES PROBLEM

In this chapter, we consider the solution of large sparse box constrained least squares problems (BLS). A new class of iterative methods is proposed by utilizing modulus transformation, which converts the solution of the BLS into a sequence of the unconstrained least squares problems. The efficient Krylov subspace methods with suitable preconditioners are applied to solve the inner unconstrained least squares problems for each outer iteration. Similar to the NNLS case in Chapter 3, the method can be further enhanced by incorporating the active set strategy, which contains two stages where the first stage consists of modulus iterations to identify the active set, while the second stage solves the reduced unconstrained least squares problems only on the inactive variables, and projects the solution into the feasible region. Numerical experiments show the efficiency of the proposed methods in comparison of the gradient projection methods.

4.1 Modulus Methods

In Section 2.1 of Chapter 2, we have shown the equivalence of BLS

$$\min_{x \in \mathbf{R}^n} \frac{1}{2} \|Ax - b\|_2 \quad \text{subject to} \quad l \leq x \leq u$$

with the linear complementarity problem (LCP($\mathscr{A}$, $\mathbf{b}$)) (2.2)

$$A^{\mathrm{T}}Ax - A^{\mathrm{T}}b - \lambda_1 + \lambda_2 = 0$$
$$\begin{bmatrix}\lambda_1\\ \lambda_2\end{bmatrix} \geq 0, \quad \begin{bmatrix}I\\ -I\end{bmatrix} x - \begin{bmatrix}l\\ -u\end{bmatrix} \geq 0 \quad \text{and} \quad \begin{bmatrix}\lambda_1\\ \lambda_2\end{bmatrix}^{\mathrm{T}} \left(\begin{bmatrix}I\\ -I\end{bmatrix} x - \begin{bmatrix}l\\ -u\end{bmatrix} \right).$$

Similar to modulus transformation strategy used on NNLS, set

$$\begin{bmatrix}I\\ -I\end{bmatrix} x - \begin{bmatrix}l\\ -u\end{bmatrix} = \begin{bmatrix}z_1 + |z_1|\\ z_2 + |z_2|\end{bmatrix} \quad \text{and} \quad \begin{bmatrix}\lambda_1\\ \lambda_2\end{bmatrix} = \begin{bmatrix}\Omega_1(|z_1| - z_1)\\ \Omega_2(|z_2| - z_2)\end{bmatrix},$$

where Ω_1 and Ω_2 are positive diagonal matrices. Then we have

$$\begin{aligned} x = z_1 + |z_1| + l = u - z_2 - |z_2| \\ A^{\mathrm{T}}A(z_1 + |z_1| + l) - A^{\mathrm{T}}b - |z_1| + z_1 + |z_2| - z_2 = 0 \\ A^{\mathrm{T}}A(u - z_2 - |z_2|) - A^{\mathrm{T}}b - |z_1| + z_1 + |z_2| - z_2 = 0 \end{aligned}$$

The last two equations can be written as a fixed point equation in matrix form

$$\begin{bmatrix}\Omega_1 + A^{\mathrm{T}}A & -\Omega_2\\ -\Omega_1 & \Omega_2 + A^{\mathrm{T}}A\end{bmatrix} \begin{bmatrix}z_1\\ z_2\end{bmatrix} = \begin{bmatrix}\Omega_1 - A^{\mathrm{T}}A & -\Omega_2\\ -\Omega_1 & \Omega_2 - A^{\mathrm{T}}A\end{bmatrix} \begin{bmatrix}|z_1|\\ |z_2|\end{bmatrix} + \begin{bmatrix}A^{\mathrm{T}}b - A^{\mathrm{T}}Al\\ -A^{\mathrm{T}}b + A^{\mathrm{T}}Au\end{bmatrix}.$$

If we apply Jacobi iteration method to solve the fixed point equation, we have

$$\begin{aligned} (\Omega_1 + A^{\mathrm{T}}A)z_1^{k+1} = (\Omega_1 - A^{\mathrm{T}}A)|z_1^k| + A^{\mathrm{T}}b - A^{\mathrm{T}}Al - \Omega_2(|z_2^k| - z_2^k) \\ (\Omega_2 + A^{\mathrm{T}}A)z_2^{k+1} = (\Omega_2 - A^{\mathrm{T}}A)|z_2^k| - A^{\mathrm{T}}b + A^{\mathrm{T}}Au - \Omega_1(|z_1^k| - z_1^k) \end{aligned}$$

where $k = 0, 1, 2, \ldots$. Note that they can be regarded as the normal equations of the first kind

$$\begin{bmatrix}A\\ \Omega_1^{1/2}\end{bmatrix}^{\mathrm{T}} \begin{bmatrix}A\\ \Omega_1^{1/2}\end{bmatrix} z_1^{k+1} = \begin{bmatrix}A\\ \Omega_1^{1/2}\end{bmatrix}^{\mathrm{T}} \begin{bmatrix}-A|z_1^k| + b - Al\\ \Omega_1^{1/2}|z_1^k| - \Omega_1^{-1/2}\Omega_2(|z_2^k| - z_2^k)\end{bmatrix}$$
$$\begin{bmatrix}A\\ \Omega_2^{1/2}\end{bmatrix}^{\mathrm{T}} \begin{bmatrix}A\\ \Omega_2^{1/2}\end{bmatrix} z_2^{k+1} = \begin{bmatrix}A\\ \Omega_2^{1/2}\end{bmatrix}^{\mathrm{T}} \begin{bmatrix}-A|z_2^k| - b + Au\\ \Omega_2^{1/2}|z_2^k| - \Omega_2^{-1/2}\Omega_1(|z_1^k| - z_1^k)\end{bmatrix}$$

which is equivalent to the least squares problems

$$\min \left\| \begin{bmatrix} A \\ \Omega_1^{1/2} \end{bmatrix} z_1^{k+1} - \begin{bmatrix} -A|z_1^k| + b - Al \\ \Omega_1^{1/2}|z_1^k| - \Omega_1^{-1/2}\Omega_2(|z_2^k| - z_2^k) \end{bmatrix} \right\|_2$$
$$\min \left\| \begin{bmatrix} A \\ \Omega_2^{1/2} \end{bmatrix} z_2^{k+1} - \begin{bmatrix} -A|z_2^k| - b + Au \\ \Omega_2^{1/2}|z_2^k| - \Omega_2^{-1/2}\Omega_1(|z_1^k| - z_1^k) \end{bmatrix} \right\|_2$$

where $k = 0, 1, 2, ...$ The right hand side of the second equation can use z_1^{k+1} instead of z_1^k.

$$\begin{aligned} (\Omega_1 + A^{\mathrm{T}}A)z_1^{k+1} &= (\Omega_1 - A^{\mathrm{T}}A)|z_1^k| + A^{\mathrm{T}}b - A^{\mathrm{T}}Al - \Omega_2(|z_2^k| - z_2^k) \\ (\Omega_2 + A^{\mathrm{T}}A)z_2^{k+1} &= (\Omega_2 - A^{\mathrm{T}}A)|z_2^k| - A^{\mathrm{T}}b + A^{\mathrm{T}}Au - \Omega_1(|z_1^{k+1}| - z_1^{k+1}) \end{aligned}$$

where $k = 0, 1, 2, ...$

Method 2:

$$\begin{aligned} (\Omega_1 + A^{\mathrm{T}}A)z_1^{k+1} &= (\Omega_1 - A^{\mathrm{T}}A)|z_1^k| + A^{\mathrm{T}}b - A^{\mathrm{T}}Al - \Omega_2(|z_2^k| - z_2^k) \\ z_2^{k+1} &= u - (z_1^{k+1} + |z_1^{k+1}| + l) - |z_2^k|. \end{aligned}$$

where $k = 0, 1, 2, ...$

Method 3:

$$\begin{aligned} (\Omega_1 + A^{\mathrm{T}}A)z_1^{k+1} &= (\Omega_1 - A^{\mathrm{T}}A)|z_1^k| + A^{\mathrm{T}}b - A^{\mathrm{T}}Al - \Omega_2(|z_2^k| - z_2^k) \\ z_1^{k+1} &= (u - z_2^{k+1} - |z_2^{k+1}|) - |z_1^k| - l. \end{aligned}$$

where $k = 0, 1, 2, ...$

The modulus inner outer iteration method 1 is described as follows.

Algorithm 4.1.1. Modulus Inner Outer Iteration Method 1

```
1.   Choose an initial solution $l \le x^0 \le u$, $r^0 = b - Ax^0$ and $s^0 = A^{\mathrm{T}}r^0$
2.   Set $x_1^0 = x_2^0 = x^0$, $s_1^0 = s_2^0 = s^0$, $z_1^0 = (x^0 - l)/2$ and $z_2^0 = (u - x^0)/2$.
3.   For $k = 0, 1, 2, \ldots$ until convergence
```

```
4.          Solve

            $$(\Omega + A^{T}A)w_1^{k+1} = \Omega(|z_1^k| - z_1^k) - \Omega(|z_2^k| - z_2^k) + s_1^k$$

5.          Set $z_1^{k+1} = z_1^k + w_1^{k+1}$ and $x_1^{k+1} = z_1^{k+1} + |z_1^{k+1}| + l$.
6.          Compute $r_1^{k+1} = b - Ax_1^{k+1}$ and $s_1^{k+1} = A^{T}r_1^{k+1}$
7.          Solve

            $$(\Omega + A^{T}A)w_2^{k+1} = \Omega(|z_2^k| - z_2^k) - \Omega(|z_1^{k+1}| - z_1^{k+1}) - s_2^k$$

8.          Set $z_2^{k+1} = z_2^k + w_2^{k+1}$ and $x_2^{k+1} = u - z_2^{k+1} - |z_2^{k+1}|$.
9.          Compute $r_2^{k+1} = b - Ax_2^{k+1}$ and $s_2^{k+1} = A^{T}r_2^{k+1}$
10.     Endfor
```

The modulus inner outer iteration method 2 is described as follows.

Algorithm 4.1.2. Modulus Inner Outer Iteration Method 2

```
1.      Choose an initial solution $l \leq x^0 \leq u$, $r^0 = b - Ax^0$ and $s^0 = A^{T}r^0$
2.      Set $x_1^0 = x_2^0 = x^0$, $s_1^0 = s_2^0 = s^0$, $z_1^0 = (x^0 - l)/2$ and $z_2^0 = (u - x^0)/2$.
3.      For $k = 0, 1, 2, \ldots$ until convergence
4.          Solve

            $$(\Omega + A^{T}A)w_1^{k+1} = \Omega(|z_1^k| - z_1^k) - \Omega(|z_2^k| - z_2^k) + s_1^k$$

5.          Set $z_1^{k+1} = z_1^k + w_1^{k+1}$ and $x_1^{k+1} = z_1^{k+1} + |z_1^{k+1}| + l$.
6.          Compute $r_1^{k+1} = b - Ax_1^{k+1}$ and $s_1^{k+1} = A^{T}r_1^{k+1}$
7.          Set
                $$z_2^{k+1} = u - x_1^{k+1} - |z_2^k|$$
8.          Set $x_2^{k+1} = u - z_2^{k+1} - |z_2^{k+1}|$.
9.          Compute $r_2^{k+1} = b - Ax_2^{k+1}$ and $s_2^{k+1} = A^{T}r_2^{k+1}$
10.     Endfor
```

4.2 Inner Iterations for Saddle Point Problems

We utilize matrix splitting to construct iterative methods for the solution of LCP (2.2), which is equivalent to the solution of BLS problem (1.2). Let $\mathscr{A} = \mathscr{M}_1 - \mathscr{N}_1 = \mathscr{M}_2 - \mathscr{N}_2$ be two splittings of the matrix $\mathscr{A}$. With modulus transformation

$$\mathbf{x} = \mathbf{z} + |\mathbf{z}|, \tag{4.1}$$

the following theorem implies that LCP$(\mathscr{A}, \mathbf{b})$ is equivalent to the implicit fixed-point equation

$$(\Omega + \mathscr{A})\mathbf{z} = (\Omega - \mathscr{A})|\mathbf{z}| + \mathbf{b}, \tag{4.2}$$

or a more general implicit fixed-point equation

$$(\Omega + \mathscr{M}_1)\mathbf{z} = \mathscr{N}_1\mathbf{z} + (\Omega - \mathscr{M}_2)|\mathbf{z}| + \mathscr{N}_2|\mathbf{z}| + \mathbf{b}, \tag{4.3}$$

where Ω is a positive diagonal parameter matrix. It can be observed that (4.2) is a special case of (4.3) when $\mathscr{M}_2 = \mathscr{A}$ and $\mathscr{N}_2 = \mathbf{0}$. In addition, it is more efficient to choose the iterative scheme based on (4.3) when the matrix $\mathscr{M}_1$ is an (block) upper or a (block) lower triangular matrix [104].

Theorem 4.2.1. *([104]) Let Ω be an $n \times n$ positive diagonal matrix. For the* LCP$(\mathscr{A}, \mathbf{b})$*, the following statements hold:*

(i) *if $(\mathbf{x}, \mathbf{f})$ is a solution of the* LCP$(\mathscr{A}, \mathbf{b})$*, then $\mathbf{z} = (\mathbf{x} - \Omega^{-1}\mathbf{f})/2$ satisfies the implicit fixed-point equation (4.3);*

(ii) *if $\mathbf{z}$ satisfies the implicit fixed-point equation (4.3), then*

$$\mathbf{x} = |\mathbf{z}| + \mathbf{z} \quad \textit{and} \quad \mathbf{f} = \Omega(|\mathbf{z}| - \mathbf{z})$$

is a solution of the LCP$(\mathscr{A}, \mathbf{b})$.

Based on Theorem 4.3.1, the accelerated modulus-based matrix splitting iterative schemes

$$(\Omega + \mathscr{A})\mathbf{z}^{k+1} = (\Omega - \mathscr{A})|\mathbf{z}^k| + \mathbf{b}, \tag{4.4}$$

and

$$(\Omega+\mathscr{M}_1)\mathbf{z}^{k+1}=\mathscr{N}_1\mathbf{z}^k+(\Omega-\mathscr{M}_2)|\mathbf{z}^k|+\mathscr{N}_2|\mathbf{z}^{k+1}|+\mathbf{b}, \tag{4.5}$$

for the solution of the implicit fixed-point equations (4.2) and (4.3), are naturally derived for the solution of BLS problem (1.2), respectively. That is to say, if $\mathbf{z}^*$ is a fixed-point of (4.4) or (4.5), then the solution of BLS problem (1.2) can be obtained straightforward by $\mathbf{x}^*=\mathbf{z}^*+|\mathbf{z}^*|$. Therefore, the solution of BLS problem (1.2) is transformed into the solution of a series of fixed-point equations (4.4) or (4.5), which can be solved directly by matrix decompositions, or by efficient iterative methods.

The modulus-type restarted iterative methods for BLS problem (1.2) is described as follows.

Algorithm 4.2.2. Modulus Type Restarted Iteration Method

```
1.    Choose an initial solution $\mathbf{z}^0$ and a parameter matrix $\Omega$;
2.    For $k=0,1,2,\ldots$ until convergence
3.        Compute a solution $\mathbf{z}^{k+1}$ by solving the fixed-point equations (4.4) or (4.5);
4.        Compute $\mathbf{x}^{k+1}=\mathbf{z}^{k+1}+|\mathbf{z}^{k+1}|$;
5.    Endfor
```

Here, the for loop is referred as the outer iteration, while the solution of (4.4) or (4.5) is referred as the inner iteration. Remark that the coefficient matrix of (4.4) or (4.5) is a two-by-two saddle point matrix $\Omega+\mathscr{M}_1$. The convergence of the fixed point iteration is presented in the following.

Theorem 4.2.3. *Let $B=\mathscr{M}_1-\mathscr{N}_1$, $C=\mathscr{M}_2-\mathscr{N}_2$ be two matrix splittings. Assume that $\Omega\in\mathbf{R}^{n\times n}$ be a positive diagonal matrix. Define*

$$\xi(\Omega)=\|\mathscr{M}_1^{-1}\mathscr{N}_1\|,\quad \eta(\Omega)=\|\mathscr{M}_1^{-1}\mathscr{N}_2\|\quad \textit{and}\quad \mu(\Omega)=\|\mathscr{M}_1^{-1}\mathscr{M}_2\|$$

where $\|\cdot\|$ is an arbitrary matrix norm and

$$\delta(\Omega)=\mu(\Omega)+2\xi(\Omega)+2\eta(\Omega).$$

If the parameter matrix Ω satisfies $\delta(\Omega)<1$, then the modulus inner outer iteration sequence $\{z^{(k)}\}_{k=0}^{+\infty}$ converges to the unique solution z^* for any initial vector $z^{(0)}$.

We would briefly review some stationary methods for the solution of saddle point system (4.5) in the following.

4.2.1 MINRES

First, we consider the Krylov subspace method for the solution of (4.4), which can be rewritten as

$$\begin{bmatrix} \Omega_1 + A^{\mathrm{T}}A & I \\ I & -\Omega_2 \end{bmatrix} \begin{bmatrix} z_1^{k+1} \\ z_2^{k+1} \end{bmatrix} = \begin{bmatrix} \Omega_1 - A^{\mathrm{T}}A & -I \\ -I & -\Omega_2 \end{bmatrix} \begin{bmatrix} |z_1^{k+1}| \\ |z_2^{k+1}| \end{bmatrix} + \begin{bmatrix} A^{\mathrm{T}}(b - Al) \\ -l + u \end{bmatrix} \tag{4.6}$$

by multiplying -1 in the second equations. Remark that

$$\begin{aligned} &\begin{bmatrix} \Omega_1 + A^{\mathrm{T}}A & I \\ I & -\Omega_2 \end{bmatrix} \\ = &\begin{bmatrix} I & 0 \\ (\Omega_1 + A^{\mathrm{T}}A)^{-1} & I \end{bmatrix} \begin{bmatrix} \Omega_1 + A^{\mathrm{T}}A & 0 \\ 0 & -\Omega_2 - (\Omega_1 + A^{\mathrm{T}}A)^{-1} \end{bmatrix} \begin{bmatrix} I & (\Omega_1 + A^{\mathrm{T}}A)^{-1} \\ 0 & I \end{bmatrix}, \end{aligned}$$

so the coefficient matrix in (4.6) is symmetric indefinite and thus symmetric solvers, such as MINRES and SymmLQ can be applied.

4.2.2 Preconditioned CG

Next, we introduce (preconditioned) conjugate gradient method for the solution of (4.4), which can be rewritten as

$$\begin{bmatrix} \Omega_1 + A^{\mathrm{T}}A & I \\ -I & \Omega_2 \end{bmatrix} \begin{bmatrix} z_1^{k+1} \\ z_2^{k+1} \end{bmatrix} = \begin{bmatrix} \Omega_1 - A^{\mathrm{T}}A & -I \\ I & \Omega_2 \end{bmatrix} \begin{bmatrix} |z_1^{k+1}| \\ |z_2^{k+1}| \end{bmatrix} + \begin{bmatrix} A^{\mathrm{T}}(b - Al) \\ l - u \end{bmatrix}. \tag{4.7}$$

Set

$$\begin{bmatrix} z_1^{k+1} \\ z_2^{k+1} \end{bmatrix} = \begin{bmatrix} z_1^{k} \\ z_2^{k} \end{bmatrix} + \begin{bmatrix} w_1^{k+1} \\ w_2^{k+1} \end{bmatrix},$$

then (4.7) can be transformed to

$$\begin{aligned}\begin{bmatrix}\Omega_1+A^{\mathrm{T}}A & I\\ -I & \Omega_2\end{bmatrix}\begin{bmatrix}w_1^{k+1}\\ w_2^{k+1}\end{bmatrix} &= \begin{bmatrix}\Omega_1(|z_1^k|-z_1^k)-A^{\mathrm{T}}A(z_1^k+|z_1^k|+l)-(z_2^k+|z_2^k|+A^{\mathrm{T}}b)\\ \Omega_2(|z_2^k|-z_2^k)+(z_1^k+|z_1^k|+l)-u\end{bmatrix}\\ &= \begin{bmatrix}\Omega_1(|z_1^k|-z_1^k)+A^{\mathrm{T}}(b-Ax^k)-y^k\\ \Omega_2(|z_2^k|-z_2^k)+x^k-u\end{bmatrix}\equiv\begin{bmatrix}\bar r_1^k\\ \bar r_2^k\end{bmatrix}.\end{aligned}$$

Note that from the first equation

$$w_1^{k+1}=(\Omega_1+A^{\mathrm{T}}A)^{-1}\bar r_1^k-(\Omega_1+A^{\mathrm{T}}A)^{-1}w_2^{k+1}.$$

It can be utilized to eliminate the variable w_1^{k+1} in the second equation, then we have

$$\begin{cases}((\Omega_1+A^{\mathrm{T}}A)^{-1}+\Omega_2)w_2^{k+1}=(\Omega_1+A^{\mathrm{T}}A)^{-1}\bar r_1^k+\bar r_2^k\\ (\Omega_1+A^{\mathrm{T}}A)w_1^{k+1}=\bar r_1^k-w_2^{k+1}\end{cases}\tag{4.8}$$

which can be solved by (preconditioned) CG method since the coefficient matrices $(\Omega_1+A^{\mathrm{T}}A)^{-1}+\Omega_2$ and $\Omega_1+A^{\mathrm{T}}A$ are symmetric positive definite with suitable parameter matrices Ω_1 and Ω_2.

4.2.3 Stationary Iteration

Note that the stationary iteration is based on the way of matrix splitting of the coefficient matrix. In the following, we introduce two classes of matrix splitting.

4.2.4 Matrix Splitting of $\mathscr{A}$

Let $\mathscr{A}=\mathscr{D}-\mathscr{L}-\mathscr{U}$, where

$$\mathscr{D}=\begin{bmatrix}\Omega_1+A^{T}A & 0\\ 0 & Q\end{bmatrix},\quad \mathscr{L}=\begin{bmatrix}0 & 0\\ I & 0\end{bmatrix}\quad\text{and}\quad \mathscr{U}=\begin{bmatrix}0 & -I\\ 0 & Q\end{bmatrix}.\tag{4.9}$$

Table 4.1: The choice of parameter matrix Q

Parameters	Algorithm
$Q = \frac{1}{\alpha} I,\ \omega = 1$	Uzawa algorithm
$Q = I$	SOR algorithm 1 (Golub, etc. '01)
$Q = (A^{\mathrm{T}}A)^{-1}$	SOR algorithm 2 (Golub, etc. '01)
$Q = \alpha I$	SOR algorithm 3 (Golub, etc. '01)

The generalized successive overrelaxation (GSOR) matrix splitting [104, 105] is

$$\begin{aligned}\mathscr{M} &= \frac{1}{\omega}(\mathscr{D} - \omega\mathscr{L}) = \frac{1}{\omega}\begin{bmatrix} A^{\mathrm{T}}A & 0 \\ -\omega I & Q \end{bmatrix} \\ \mathscr{N} &= \frac{1}{\omega}(\mathscr{D} - \omega\mathscr{L}) = \frac{1}{\omega}\begin{bmatrix} (1-\omega)A^{\mathrm{T}}A & -\omega I \\ 0 & Q \end{bmatrix}.\end{aligned}$$

Substituting it to the scheme (4.5), we have

$$\begin{aligned}\begin{bmatrix} \Omega_1 + \frac{1}{\omega}A^{\mathrm{T}}A & 0 \\ -I & \Omega_2 + \frac{1}{\omega}Q \end{bmatrix}\begin{bmatrix} z_1^{k+1} \\ z_2^{k+1} \end{bmatrix} &= \begin{bmatrix} \Omega_1 + \frac{1}{\omega}A^{\mathrm{T}}A & 0 \\ -I & \Omega_2 + \frac{1}{\omega}Q \end{bmatrix}\begin{bmatrix} z_1^{k} \\ z_2^{k} \end{bmatrix} + \begin{bmatrix} A^{\mathrm{T}}(b - Al) \\ l - u \end{bmatrix} \\ &+ \begin{bmatrix} 0 & 0 \\ I & 0 \end{bmatrix}\begin{bmatrix} |z_1^{k+1}| \\ |z_2^{k+1}| \end{bmatrix} + \begin{bmatrix} \Omega_1 - A^{\mathrm{T}}A & -I \\ 0 & \Omega_2 \end{bmatrix}\begin{bmatrix} |z_1^{k}| \\ |z_2^{k}| \end{bmatrix}\end{aligned} \tag{4.10}$$

Set

$$\begin{bmatrix} z_1^{k+1} \\ z_2^{k+1} \end{bmatrix} = \begin{bmatrix} z_1^{k} \\ z_2^{k} \end{bmatrix} + \begin{bmatrix} w_1^{k+1} \\ w_2^{k+1} \end{bmatrix},$$

we have

$$\begin{cases} (\omega\Omega_1 + A^{\mathrm{T}}A)w_1^{k+1} = \omega(\Omega_1(|z_1^k| - z_1^k) + A^{\mathrm{T}}(b - Ax^k) - y^k), \\ z_1^{k+1} = z_1^k + w_1^{k+1}, \quad x^{k+1} = z_1^{k+1} + |z_1^{k+1}| + l, \\ (\omega\Omega_2 + Q)w_2^{k+1} = \omega(\Omega_2(|z_2^k| - z_2^k) + x^{k+1} - u), \\ z_2^{k+1} = z_2^k + w_2^{k+1}, \quad y^{k+1} = z_2^{k+1} + |z_2^{k+1}|, \end{cases} \tag{4.11}$$

For the Choice of Q, we refer to the Table 4.1 [44].

Table 4.2: Comparison of the iterative methods (full rank and inconsistent problem with $\kappa(A) = 10^4$).

ρ	PG	Mod1	Mod2	GPCG	ModCG
0.7	14,741	2,451	1,224	1	5
	1.00	2.74	2.73	(1.00,9.00)	(3.40,3.00)
	4.42×10^4	1.83×10^4	9.13×10^3	1.59×10^2	6.06×10^2

First row: number of outer iterations.
Second row: average inner iterations (of the first stage, second stage).
Third row: number of matrix vector multiplications.
$(tol = 10^{-5},\ tol_{in} = 0.1)$

4.3 Numerical Experiments

4.3.1 Dense full rank case

First, we show how the condition number and the distribution of singular values of A influence the convergence of the modulus-type and projection-type methods with a class of dense matrices of the form $A = U\Sigma V^{\mathrm{T}}$, where $U \in \mathbf{R}^{m\times m}$ and $V \in \mathbf{R}^{n\times n}$ are orthogonal matrices obtained from the QR decomposition of random matrices, and $\Sigma \in \mathbf{R}^{m\times n}$ is a rectangular diagonal matrix with diagonal entries $\sigma_1 > \sigma_2 > ... > \sigma_n$, where the ith smallest singular value is

$$\sigma_{n-i+1} = \sigma_n + \frac{i-1}{n-1}(\sigma_1 - \sigma_n)\rho^{n-i}, \quad i = 1, ..., n,$$

with the parameter $\rho \in (0,1]$. Note that when ρ decreases, the singular values are tightly clustered towards the smallest singular value σ_n and are far apart towards the largest singular value σ_1. The idea of generating this kind of matrices is from [46, 53].

In our numerical experiments, we set $m = 200,\ n = 100,\ \sigma_1 = 1,\ \sigma_n = 0.01$ or 0.0001, $\rho = 1,\ 0.9,\ 0.8,\ 0.7$, and form inconsistent BLS problems where the elements of the vector b are generated randomly following the normal distribution with mean zero and variance 1, using the MATLAB function randn$(m, 1)$.

Table 4.3: Comparison of the iterative methods (full rank and inconsistent problem with $\kappa(A) = 10^4$).

ρ	PG	Mod1	Mod2	GPCG	ModCG
0.7	19,556	3,012	1,508	1	3
	1.00	2.76	2.76	(1.00,8.00)	(1.67,6.67)
	5.87×10^4	2.26×4	1.13×10^4	1.33×10^2	6.46×10^2

First row: number of outer iterations.
Second row: average inner iterations (of the first stage, second stage).
Third row: number of matrix vector multiplications.
$(tol = 10^{-5},\ tol_{in} = 0.1)$

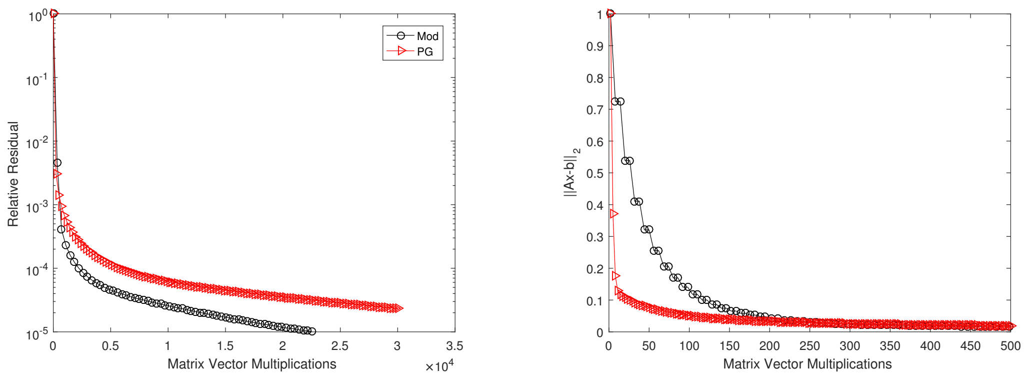


Figure 4.1: Relative residual vs. matrix vector multiplications (left); objective function value vs. matrix vector multiplications (right).

4.3.2 Box constrained ill-posed problem

Consider the Fredholm integral equation of the first kind

$$\int_{-6}^{6} \kappa(\tau,\sigma)x(\sigma)d\sigma = b(\tau), \quad -6 \le \tau \le 6,$$

Table 4.4: Comparison of the numerical methods on box constrained ill-posed problems

		ProjCG	ProjGrad	Mod-GSOR2	Mod-GSOR3
10^{-1}	IT(Inner)	8(13.33)	8(10.50)	8(2.63)	6(2.83)
	Mat.-Vec	112	84	21	17
	CPU	0.12	0.11	0.03	0.06
10^{-2}	IT(Inner)	> 1000	13(45.77)	15(8.13)	17(7.18)
	Mat.-Vec		595	122	122
	CPU		0.37	0.12	0.16
10^{-3}	IT(Inner)	> 1000	49(145.73)	35(13.34)	43(12.19)
	Mat.-Vec		7141	467	524
	CPU		3.46	0.31	0.44
10^{-4}	IT(Inner)	> 1000	134(434.01)	106(21.80)	149(18.42)
	Mat.-Vec		58157	2311	2745
	CPU		26.33	1.29	1.95
10^{-5}	IT(Inner)	> 1000	> 1000	275(38.30)	379(31.51)
	Mat.-Vec			10532	11944
	CPU			5.34	6.97

discussed by Phillips. Its solution, kernel, and right-hand side are given by

$$\begin{aligned} x(\sigma) &= \begin{cases} 1+\cos(\frac{\pi}{3}\sigma), & \text{if} -3<\sigma<3, \\ 0, & \text{otherwise}, \end{cases} \\ \kappa(\tau,\sigma) &= x(\tau-\sigma) \\ b(\tau) &= (6-|\tau|)(1+\frac{1}{2}\cos(\frac{\pi}{3}\tau))+\frac{9}{2\pi}\sin(\frac{\pi}{3}|\tau|) \end{aligned}$$

We use the MATLAB code phillips from Hansen to discretize the integral equation by a Galerkin method with orthonormal box functions as test and trial functions. $A \in R^{300\times 300}$ is a symmetric indefinite matrix and right-hand side $b \in R^{300}$. The condition number of A is $\kappa(A) = \|A\|\|A^{-1}\| = 2.142 \times 10^8$.

4.3.3 Sparse full column rank

We generate a class of sparse full column rank matrices, abbreviated as “Randn_i", $i = 1,2,3,4,5$, using the MATLAB function “sprandn" with $m = 30,000$, $n = 3,000$, Ratio of nonzero elements density$= 0.1\%$. The condition numbers of these matrices are

Table 4.5: Comparison of the numerical methods on sparse full column rank problems

Problem		ProjCG	ProjGrad	Mod-GSOR2	Mod-GSOR3
Randn_1	IT(Inner)	> 1000	6(23.33)	32(13.81)	32(12.00)
	Mat.-Vec		180	891	733
	CPU		0.08	0.43	0.36
Randn_2	IT(Inner)	> 1000	376(72.10)	123(69.90)	204(34.91)
	Mat.-Vec		32,711	19,711	17,429
	CPU		10.51	4.10	2.34
Randn_3	IT(Inner)	> 1000	701(212.76)	676(211.92)	660(122.67)
	Mat.-Vec		342,136	242,136	163,941
	CPU		102.70	60.70	25.85
Randn_4	IT(Inner)	> 1000	936(152.89)	784(102.46)	785(67.55)
	Mat.-Vec		1,124,731	394,647	200,784
	CPU		834.01	431.94	210.13
Randn_5	IT(Inner)	> 1000	> 1000	788(185.13)	792(111.88)
	Mat.-Vec			419,483	234,239
	CPU			494.38	241.42

specified as

$$\kappa(\text{Randn\_i}) = 10^i, \quad i = 1,2,3,4,5.$$

The nonzero element values were generated by a random number generator following the normal distribution, and the pattern of the nonzero elements is also determined by a random number generator. We form inconsistent BLS problems where the elements of the right-hand side vector b are generated randomly using the Matlab function randn$(m,1)$. Same b is used for all the cases.

4.4 Concluding Remarks

In this chapter, we consider the solution of large sparse BLS problems using a new class of iterative methods based on modulus transformation, which converts the solution of the BLS into a sequence of the unconstrained least squares problems. The efficient Krylov subspace methods with suitable preconditioners are applied to solve the inner unconstrained least squares problems for each outer iteration. We also discuss the solution of saddle point inner systems, and the choice of the parameter matrix. Numerical experiments show the efficiency of the proposed methods in comparison of the gradient projection methods.

CHAPTER 5

NONNEGATIVE CONSTRAINED ILL-POSED PROBLEM

Many questions in science and engineering give rise to linear discrete ill-posed problems. Often it is desirable that the computed approximate solution satisfies certain constraints, e.g., that some or all elements of the computed solution be nonnegative [74]. In this chapter, we employ the modulus type inner outer iterative method with regularization, including the discrepancy principle and Tikhonov regularization, which is one of the most popular methods for the solution of linear discrete ill-posed problems, for the solution of large scale problems of this kind. Since the desired solution is known to lie in the nonnegative cone. It is then natural to require that the approximate solution determined by regularization also lies in this cone.

The rest of the chapter is organized as follows. In Section 5.1, we briefly introduce the constrained ill-posed problem. In Section 5.2, the modulus-type inner outer iteration method is proposed combined with the active set strategy directly. In sections 5.3 and 5.4, the modulus-type inner outer iteration method is applied directly to solve the image restoration ill-posed problem with discrepancy principle and Tikhonov regularization, respectively. Finally in Section 5.5, the concluding remarks are given.

5.1 Introduction

The discretization of linear ill-posed problems gives rise to linear systems of equations with constraints

$$Ax = b, \quad x \geq \mathbf{0}, \tag{5.1}$$

where $A \in \mathbf{R}^{m\times n}$ is a large matrix whose singular values "cluster" at the origin and the vector $b \in \mathbf{R}^m$ is contaminated by error. In particular, the matrix is severely ill-conditioned and may be rank-deficient. In many linear discrete ill-posed problems that arise in science and engineering, such as the restoration of an image, the right hand side vector is contaminated by blur and noise [26]. Hence, (5.1) is generally inconsistent and thus one has to solve a NNLS problem of the form

$$\min_{x\in\mathbf{R}^n} \|Ax-b\|_2. \tag{5.2}$$

Matrices of this kind arise from the discretization of linear ill-posed problems such as Fredholm integral equations of the first kind and, therefore, the minimization problem (5.2) is referred to as a linear discrete ill-posed problem. Applications include remote sensing and image restoration. In the latter application the kernel of the integral equation is known as the point-spread function (PSF) and describes the blur-contamination of an unavailable image that one would like to restore. The vector b represents known error-contaminated data and can be expressed as

$$b = b_{\text{true}} + \eta, \tag{5.3}$$

where b_{true} is an unknown error-free vector associated with b and η represents the error in b. We will refer to η as *noise*. In image restoration applications, b_{true} represents an unavailable blur-contaminated, but noise-free, image, while b represents an available image that has been contaminated by both blur and noise. The noise may stem from measurement and/or discretization errors. We will assume that a fairly sharp bound,

$$\|\eta\|_2 \leq \delta, \tag{5.4}$$

for the error is available.

Let $A^\dagger$ denote the Moore–Penrose pseudoinverse of A. We would like to determine

$x_{\text{true}} = A^{\dagger} b_{\text{true}}$. The minimum norm solution of (5.2) can be expressed as $A^{\dagger} b$. Due to the severe ill-conditioning of A and the presence of the error η in b, the vector

$$A^{\dagger} b = A^{\dagger} b_{\text{true}} + A^{\dagger} \eta = x_{\text{true}} + A^{\dagger} \eta$$

typically is dominated by the propagated error $A^{\dagger} \eta$ and then is not a useful approximation of x_{true}. Generally, a much better approximation of x_{true} can be determined by first replacing the least-squares problem (5.2) by a nearby minimization problem that is less sensitive to the error η in b. One of the most popular replacement methods is *Tikhonov regularization*, which in its simplest form yields the minimization problem

$$\min_{x \in \mathbf{R}^n} \left\{ \|Ax - b\|_2^2 + \mu \|x\|_2^2 \right\}, \tag{5.5}$$

where the scalar $\mu > 0$ is referred to as the *regularization parameter*. The normal equations associated with this minimization problem are given by

$$(A^{\mathrm{T}} A + \mu I_n) x = A^{\mathrm{T}} b, \tag{5.6}$$

which shows that (5.5) has the unique solution

$$x_{\mu} = (A^{\mathrm{T}} A + \mu I_n)^{-1} A^{\mathrm{T}} b. \tag{5.7}$$

for any fixed $\mu > 0$; see, e.g., [38, 45] for properties of Tikhonov regularization. I_n is the identity matrix of order n.

The Tikhonov minimization problem (5.5) has two terms: the first one is a *fidelity term* that ensures that the solution x_{μ} approximately fits the observed data, and the second one is a *regularization term* that penalizes the Euclidean norm of x_{μ}. The balance between data fitting and penalization is determined by the regularization parameter μ. An imprudent choice of μ makes x_{μ} a poor approximation of x_{true}: if μ is too small, then the error η will be propagated and amplified in x_{μ}, while if μ is too large, then x_{μ} will be over-smoothed without displaying details that x_{true} may possess.

The *discrepancy principle* provides an approach to determine a suitable value of μ.

It prescribes that $\mu > 0$ be chosen so that

$$\left\| A x_\mu - b \right\| = \tau \delta, \tag{5.8}$$

where $\tau > 1$ is a user-chosen parameter independent of δ. This is a nonlinear equation for μ, which can be solved, e.g., by Newton's method. The discrepancy principle requires a bound (5.4) for the error η to be available, as well as the error-free minimization problem associated with (5.2) to be consistent; see [38] for discussions. We will use the discrepancy principle in the computed examples reported in Section 5.3. However, the solution methods discussed in this paper also can be applied in conjunction with other techniques for determining a suitable value of $\mu > 0$, including the L-curve criterion and generalized cross validation; see [39] for discussions and comparisons of many methods for determining a suitable value of the regularization parameter.

Due to the fact that many singular values of the matrix A cluster at the origin, the least-squares problem (5.2) may be numerically rank-deficient. Therefore, it is generally beneficial to impose constraints on the computed solution that the desired solution x_{true} is known to satisfy. For instance, in image restoration problems the entries of the vector (5.7) represent pixel values of the image. Pixel values are nonnegative and, therefore, it is generally meaningful to solve the constraint minimization problem

$$x_\mu^+ = \arg\min_{x \geq 0} \left\{ \|Ax - b\|_2^2 + \mu \|x\|_2^2 \right\} \tag{5.9}$$

instead of (5.5). Here $x \geq 0$ is intended component-wise. A closed form of the solution x_μ^+ generally is not available.

An approximation of x_μ^+ is obtained by

$$x^+ = P(x_\mu) = P\left((A^{\mathrm{T}} A + \mu I)^{-1} A^{\mathrm{T}} b \right). \tag{5.10}$$

When $x_{\text{true}} \geq 0$, the vector x^+ generally is a better approximation of x_{true} than x_μ. However, typically x_μ^+ is a much more accurate approximation of x_{true} than x^+. This depends, at least in part, on the fact that the condition number of the matrix A restricted to Ω typically is smaller than the condition number of A; the latter is defined as the ratio between the largest and smallest singular values of A; see, e.g., [41].

In the following, we will apply the modulus type inner outer iteration method for solving nonnegative constrained ill-posed problem with discrepancy principle and Tikhonov regularization, respectively. Before that, the active set modulus inner outer iteration method is derived in the next section.

5.2 Modulus Inner Outer Iteration Method with Active Set Strategy

As mentioned in Chapter 2, the disadvantage of some active set method is that the iteration in the inner iteration is terminated as soon as a component of a computed iterate violates a constraint, which forces frequent restart of the outer iteration and thus slows down convergence. Another undesirable feature is that the active set type algorithm allows only one variable to leave a bound at a given outer iteration, which allows to add or delete one index from the active set at a time. This is a very inefficient feature when the number of variables is large.

In order to avoid these disadvantages, Calvetti et al. [26] proposed a projected restarted iteration method for nonnegative constrained ill-posed problems by allowing more consecutive iterations in the inner iteration. The algorithm is given as follows.

Algorithm 5.2.1. Projected Restarted Iteration Method

```
1.  Choose an initial approximate solution $x^0$ and compute $r^0 = b - Ax^0$.
2.  For $k = 0, 1, 2, \ldots$ until convergence
3.      Compute an approximate solution $w^k$ by an iterative method
            $$\min_{w \in \mathbf{R}^n} \|Aw - r^k\|_2.$$
4.      Compute $\hat{x}^{k+1} = x^k + w^k$ and project it on the nonnegative region
            $$x_j^{k+1} = \begin{cases} 0, & \hat{x}_j^{k+1} < 0; \\ \hat{x}_j^{k+1}, & \hat{x}_j^{k+1} \geq 0. \end{cases} \qquad j = 1, 2, \ldots, n.$$
5.      Compute $r^{k+1} = b - Ax^{k+1}$.
6.  Endfor
```

Here, the unconstrained least squares problem for each loop is solved with CGLS, GMRES and RRGMRES iterative methods until the stopping criterion is satisfied. In addition, Morigi et al. [74] proposed an active set restarted projected CG method for general box constrained ill-posed problems, which can be applied to nonnegative constrained problems, where the components of the solution that equal their bounds are referred to as the active set and identified in the outer iteration, and the reduced unconstrained least squares problem is solved in the inner iteration by keeping the identified components fixed. The nonnegative constrained version of the algorithm is given as follows.

Algorithm 5.2.2. Active Set Projected Restarted Iteration Method

```
1.  Choose an initial approximate solution $x^0$ and compute $r^0 = b - Ax^0$.
2.  For $k = 0, 1, 2, \dots$ until convergence
3.      Define Lagrange multipliers $\lambda^k = -A^T r^k$.
4.      Define active set $\mathscr{B}$ and free variable set $\mathscr{F}$
            $\mathscr{B} = \{j : x_j^k = 0,\ \lambda_j^k \geq 0\}, \quad \mathscr{F} = \{1,2,\dots,n\} \backslash \mathscr{B}.$
5.      Compute an approximate solution $w^k$ by an iterative method
            $\min_{w \in \mathbf{R}^{\check{n}}} \|A_{\mathscr{F}} w - r^k\|_2.$
6.      Compute
            $\hat{x}_{\mathscr{F}}^{k+1} = x_{\mathscr{F}}^k + w^k,\ \hat{x}_{\mathscr{B}}^{k+1} = x_{\mathscr{B}}^k,$
        and project it on the nonnegative region
            $x_j^{k+1} = \begin{cases} 0, & \hat{x}_j^{k+1} < 0; \\ \hat{x}_j^{k+1}, & \hat{x}_j^{k+1} \geq 0. \end{cases} \quad j = 1,2,\dots,n.$
7.      Compute $r^{k+1} = b - Ax^{k+1}$.
8.  Endfor
```

Here, $\check{n}$ denotes the number of elements in set $\mathscr{F}$, and $A_{\mathscr{F}}$ denotes the submatrix of A consisting of the columns of A whose indices belong to $\mathscr{F}$. These methods are shown to require low storage requirement and are easy to implement, and numerical examples arising from constrained linear ill-posed problems as well as image restoration indicate their fairly rapid convergence. However, there is no theoretical analysis to guarantee the convergence, and the norm of consecutively generated residual vectors might not be monotonically decreasing.

Similar to Algorithm 5.2.2 [74], the corresponding active set version of modulus iterative algorithm is given as follows.

Algorithm 5.2.3. Active Set Modulus-Type Inner Outer Iteration Method

```
1.  Choose an initial approximate solution $z^0$ and a parameter matrix $\Omega$.
2.  Compute $x^0 = z^0 + |z^0|$ and $r^0 = b - Ax^0$.
3.  Set
        $$\widetilde{A} = \begin{bmatrix} A \\ \Omega^{1/2} \end{bmatrix} \quad \text{and} \quad \tilde{b}^0 = \begin{bmatrix} -A|z^0| + b \\ \Omega^{1/2}|z^0| \end{bmatrix}.$$
4.  Compute
        $$\tilde{r}^0 = \begin{bmatrix} r^0 \\ \Omega^{1/2}(|z^0| - z^0) \end{bmatrix} (= \tilde{b}^0 - \widetilde{A}z^0).$$
5.  For $k = 0, 1, 2, \ldots$ until convergence
6.      Define Lagrange multipliers $\lambda^k = -A^T r^k$.
7.      Define active set $\mathscr{B}$ and free variable set $\mathscr{F}$
            $$\mathscr{B} = \{j : x_j^k = 0,\ \lambda_j^k \geq 0\}, \quad \mathscr{F} = \{1, 2, ..., n\} \backslash \mathscr{B}.$$
        Let $\check{n}$ be the number of elements in set $\mathscr{F}$.
8.      Compute an approximate solution $w^k$ by an iterative method
            $$\min_{w \in \mathbf{R}^{\check{n}}} \|\tilde{A}_{\mathscr{F}} w - \tilde{r}^k\|_2,$$
        where $\tilde{A}_{\mathscr{F}}$ denotes the submatrix of $\tilde{A}$ consisting of the columns of $\tilde{A}$
        whose indices belong to $\mathscr{F}$.
9.      Compute $z_{\mathscr{F}}^{k+1} = z_{\mathscr{F}}^k + w^k$, $z_{\mathscr{B}}^{k+1} = z_{\mathscr{B}}^k$.
10.     Compute $x^{k+1} = z^{k+1} + |z^{k+1}|$ and $r^{k+1} = b - Ax^{k+1}$.
11.     Set
            $$\tilde{b}^{k+1} = \begin{bmatrix} -A|z^{k+1}| + b \\ \Omega^{1/2}|z^{k+1}| \end{bmatrix}.$$
```

```
12.        Compute
```

$$\tilde{r}^{k+1} = \begin{bmatrix} r^{k+1} \\ \Omega^{1/2}(|z^{k+1}| - z^{k+1}) \end{bmatrix} (= \tilde{b}^{k+1} - \widetilde{A}z^{k+1}).$$

```
13.    Endfor
```

Similar to the non-active-set algorithms, the unconstrained least squares problem is required to solve for each outer iteration. For the active set method, only the reduced system with fewer column needed to be solve. Consequently, the computational cost will be saved.

5.3 Numerical Experiments: Discrepancy Principle

We test the numerical methods PCGLS and MCGLS2 on image restoration, where the data consists of the noise- and blur-free images shown in Figure 5.1, which come from Nagy's Matlab toolbox “RestoreTools" [17]. Some basic definitions in image restoration are shown in Table 5.1. Note that the matrix A is determined by the point spread function. Vector e is generated with normally distributed entries with zero mean by Matlab. The iteration is terminated when the numerical solution x^k satisfies the discrepancy principle as

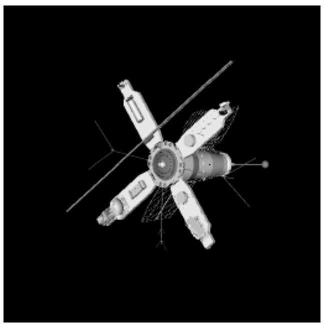
$$||b - Ax^k||_2 \leq \eta\delta, \quad \delta = ||e||_2 = ||b - \hat{b}||_2, \tag{5.11}$$

where $\eta \geq 1$ is a specified constant, or k reaches the maximal number of iteration steps, e.g., 10. The noise level δ is set to be 5% and the discrepancy factor $\eta = 1$ for all cases.

In Figures 5.2 and 5.3, the medium and large blurred and noisy image, the restored images of satellite by PCGLS and MCGLS2 are shown, respectively. The relative error of the image is defined as

$$\text{Error} = \frac{||x^k - \hat{x}||_2}{||\hat{x}||_2}.$$

The figures show that the MCGLS2 method could obtain more accurate numerical solutions and thus clearer images compared to the PCGLS method with the same computational costs.


(a)

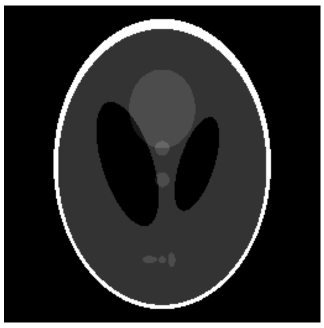

(b)

Figure 5.1: Exact image for (a) satellite and (b) variant motion.

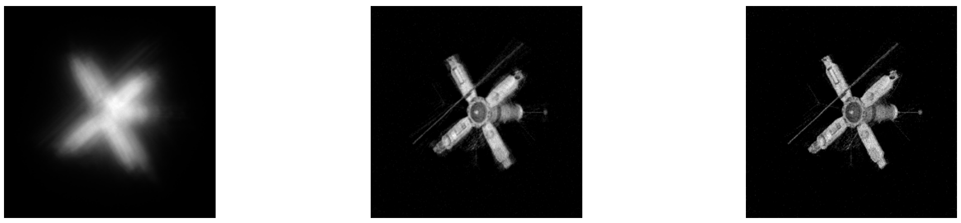


Figure 5.2: Medium blurred and noisy image (left), restored image by PCGLS (middle), and restored image by MCGLS2 (right) of satellite image.

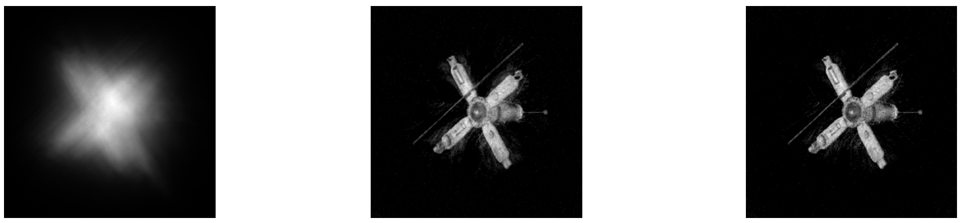


Figure 5.3: Large blurred and noisy image (left), restored image by PCGLS (middle), and restored image by MCGLS2 (right) of satellite image.

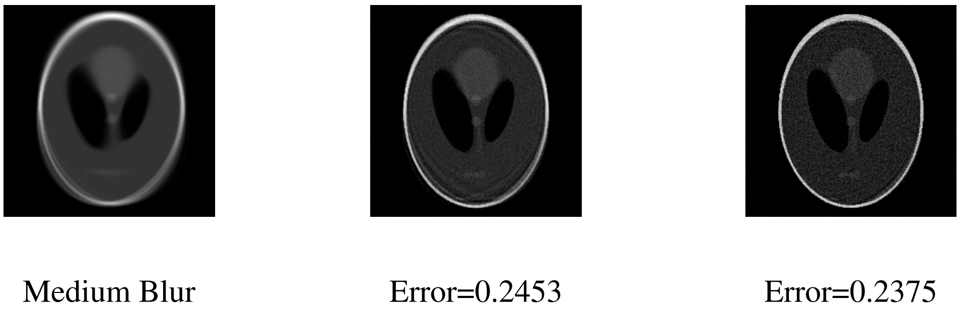


Figure 5.4: Medium blurred and noisy image (left), restored image by PCGLS (middle), and restored image by MCGLS2 (right) of variant motion image.

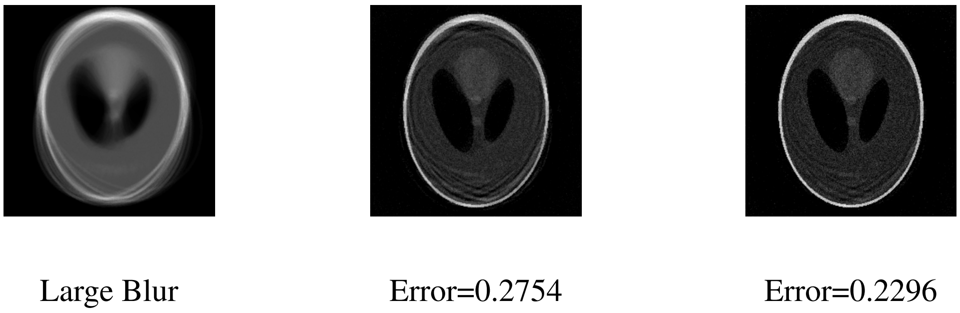


Figure 5.5: Large blurred and noisy image (left), restored image by PCGLS (middle), and restored image by MCGLS2 (right) of variant motion image.

In Figures 5.4 and 5.5, the medium and large blurred and noisy image, the restored images of variant motion by PCGLS and MCGLS2 are shown, respectively.

From the figures, the same conclusion can be reached that the MCGLS2 method could obtain more accurate numerical solutions than the PCGLS method with the same computational costs.

Table 5.1: Definitions in image restoration.

A	blurring operator
$\hat{x}$	noise- and blur-free image
$\hat{b} = A\hat{x}$	blurred noise-free image
e	noise
$b = \hat{b} + e$	blurred and noisy image
$\gamma = \|e\|_2 / \|\hat{b}\|_2$	noise level

5.4 Numerical Experiments: Tikhonov Regularization

We first rewrite the minimization problem (5.9) in the form of the previous section. Thus,

$$\begin{aligned} & \min_{x \geq 0} \left\{ \|Ax - b\|^2 + \mu \|x\|^2 \right\} \\ = \; & \min_{x \geq 0} \left\| \begin{bmatrix} A \\ \sqrt{\mu} I_n \end{bmatrix} x - \begin{bmatrix} b \\ 0 \end{bmatrix} \right\|^2 \\ = \; & \min_{x \geq 0} \|\tilde{A}x - \tilde{b}\|^2, \end{aligned} \tag{5.12}$$

where we assume that $\mu > 0$. Then the matrix $\tilde{A} \in \mathbf{R}^{(m+n)\times n}$ is of full column rank and the minimization problem (5.12) satisfies the convergence conditions. Therefore, the iterates determined by Algorithm 3.1.6 and 3.4.1 will converge.

We test the numerical methods PG, Mod, GPCG and ModASCG on image restoration problems, which come from Nagy's Matlab toolbox “RestoreTools" [17]. The modulus type methods with $\Omega = \omega \mathrm{diag}(A^{\mathrm{T}}A)$ are not applied here since it requires large computational costs to obtain the diagonal elements of $A^{\mathrm{T}}A$. Some basic definitions in image restoration are shown in Table 5.1. Note that the matrix A is determined by the point spread function. Vector e is generated with normally distributed entries with zero mean by Matlab. The noise level γ is set to be 5% and the regularization parameter is set to be $\mu = 10^{-4}$. The parameters in Mod and ModASCG are set to be $\omega = 0.1$. The relative error of the restored image is defined as

$$\text{Error} = \frac{\|x^k - \hat{x}\|_2}{\|\hat{x}\|_2}.$$

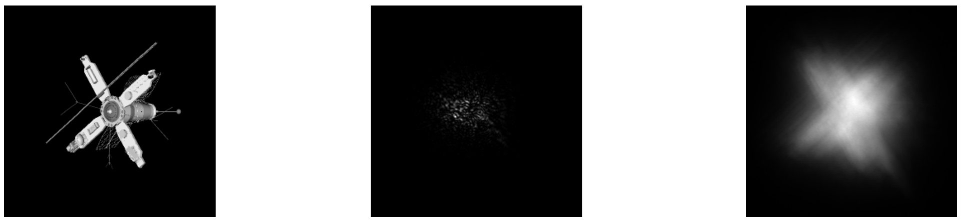


Figure 5.6: The exact image (left), PSF function (middle) and large blurred and noisy image (right) of test problem “AtmosphericBlur".

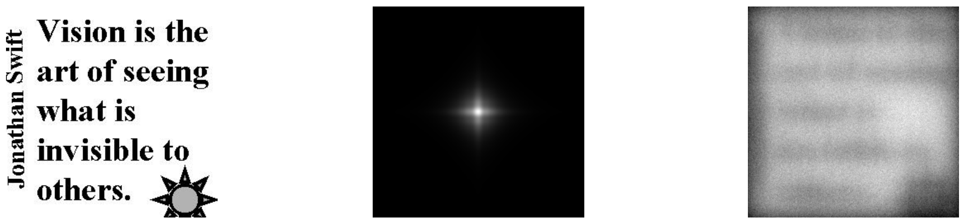


Figure 5.7: The exact image (left), PSF function (middle) and large blurred and noisy image (right) of test problem “Text".

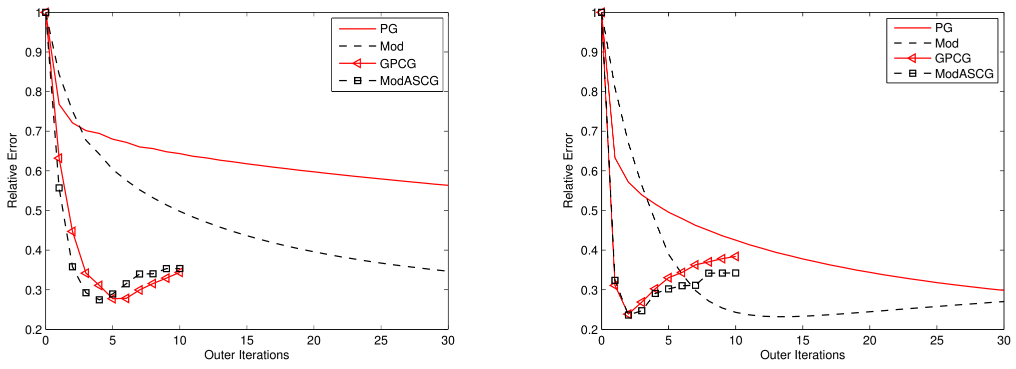
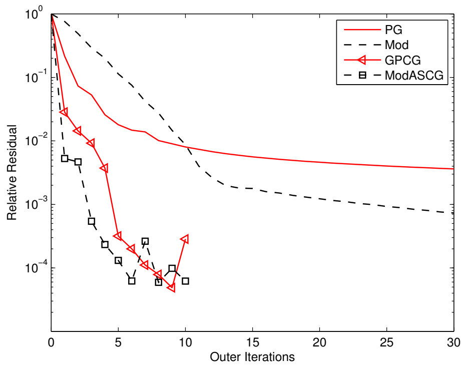


Figure 5.8: Relative error vs. outer iterations for test problems “AtmosphericBlur" (left) and “Text" (right).



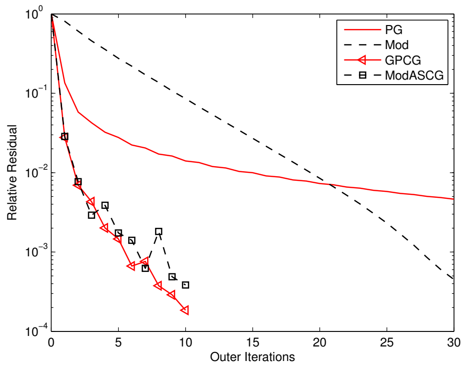
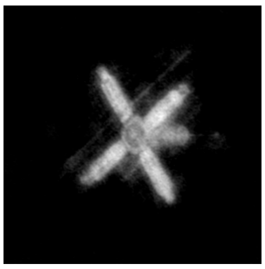
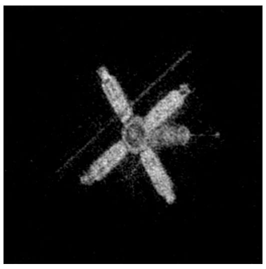


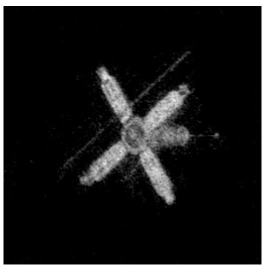
Figure 5.9: Relative residual vs. outer iterations for test problems “AtmosphericBlur" (left) and “Text" (right).



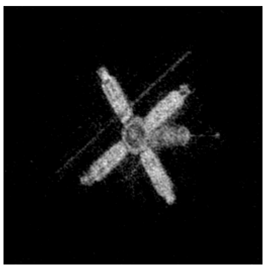
(a) Error=0.3814



(b) Error=0.2783

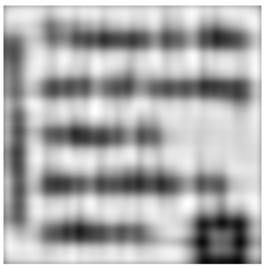


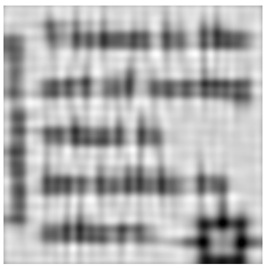
(c) Error=0.2775

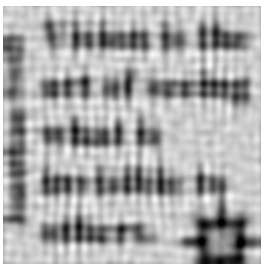


(d) Error=0.2747

Figure 5.10: Restored images by (a) PG, (b) Mod, (c) GPCG and (d) ModASCG methods for test problem “AtmosphericBlur".


(a) Error=0.3198


(b) Error=0.2945


(c) Error=0.2762

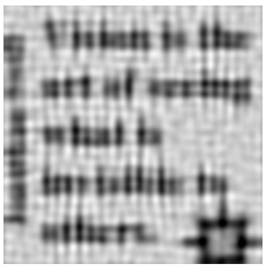

(d) Error=0.2763

Figure 5.11: Restored images by (a) PG, (b) Mod, (c) GPCG and (d) ModASCG methods for test problem “Text".

In Figures 5.6 and 5.7, the exact image and the blurred and noisy image of test problems “AtmosphericBlur" and “Text" are shown, respectively. Moreover, in Figures 5.8 and 5.9, we depict the relative error and relative residual curves of four testing methods versus the number of outer iterations for the two image restoration problems, respectively. In Figures 5.10 and 5.11, the restored images are shown.

From Figure 5.8, it is observed that the ModASCG method obtains the smallest error, and thus the most accurate restored images at the fewest outer iterations. For test problem “AtmosphericBlur", the relative error curves of PG and Mod can not reach the minimum point within 30 steps. The relative error of the two-stage methods decreases in the first few iterations, then increases with more iterations. The optimal number of outer iterations for GPCG and ModASCG are 5 and 4, respectively. Similar phenomena can be observed in test problem “Text". The modulus type methods outperform projection type methods by obtaining more accurate solutions with same computational costs.

5.5 Concluding Remarks

This chapter applies modulus-based iterative methods to nonnegative discrepancy principle and Tikhonov regularization. The discrepancy principle is used to determine the regularization parameter. Efficient solution methods are described. Several numerical examples illustrate the efficiency of the proposed methods.

In the future work, we will consider the l_1-norm regularization, which is more commonly used for the ill-posed image restoration problem. We will also consider more general case $l_p - l_q$ minimization, which requires to construct the new modulus-type algorithms and to analysis the convergence.

CHAPTER 6

NONNEGATIVE MATRIX FACTORIZATION

In this chapter, we consider the solution of nonnegative matrix factorization, which is a low rank matrix approximation problem with nonnegative constraints, using a new alternating least squares method by utilizing modulus-type inner outer iteration method to solve the nonnegative constrained least squares problem with multiple right hand sides at each iteration. Theoretical convergence that the limit of the sequence generated by the proposed method is a stationary point of NMF can be guaranteed. Moreover, by considering that the methods for NMF can be regarded as the fixed-point iteration, which the rate of convergence is at best linear, we also propose a new matrix-based active set method and employ Anderson acceleration to further speed up the algorithms. Finally, we show that the proposed methods can be extended to solve the sparse NMF and regularized NMF. Numerical experiments on the synthetic data and ORL face image data show that the proposed methods converges faster than the gradient descent methods.

In Section 6.1, a brief introduction of nonnegative matrix factorization is given. In Section 6.2 we review some existing methods for the solution of NMF, then construct modulus method for NMF in Section 6.3. In Section 6.6 we compare the proposed method with the existing methods numerically on the synthetic data and ORL face image data. and finally in Section 6.7 the concluding remarks are given.

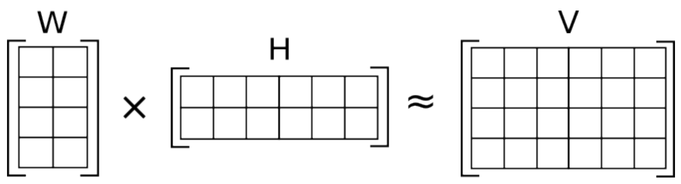


Figure 6.1: Nonnegative matrix factorization. [1]

6.1 Introduction

Given a nonnegative matrix $V \in \mathbf{R}^{m\times n}$ and a positive integer $r \ll \min(m,n)$, nonnegative matrix factorization (NMF) finds two nonnegative matrices $W \in \mathbf{R}^{m\times r}$ and $H \in \mathbf{R}^{r\times n}$ such that

$$V \approx WH. \tag{6.1}$$

One of the most common ways to formulate the approximation in (6.1) is

$$\min f(W,H) := \frac{1}{2}\|V - WH\|_F^2, \tag{6.2}$$

where $\|\cdot\|_F$ represents the Frobenius norm, or Euclidean norm, of the corresponding matrix. There are other measurements for the approximation (6.1) such as Kullback-Leibler divergence, Itakura-Saito divergence, etc., where different cost functions are used depending on the various applications and purposes. Also, the appropriate choice of r is critical and often problem dependent. The low rank approximation and the nonnegativity properties make NMF applicable for many scientific computing and engineering applications, e.g., image processing, text mining, spectral data analysis, audio signal separation, document clustering, recommender systems, and so on. The NMF was first proposed by Paatero and Tapper [88] as positive matrix factorization, and had obtained wide attention after Lee and Seung's work [68].

6.2 Existing Methods

The main framework for the solution of NMF is alternating nonnegative least squares (ANLS) method, block coordinate descent method, or block Gauss-Seidel method as listed in Algorithm 6.2.1, which iteratively fixes W (or H), and solves the nonnegative constrained least squares (NNLS) subproblems with multiple right hand sides with respect to H (or W) alternatively. Note that the NMF problem (6.2) is nonconvex and thus only the stationary points are expected to be found, while the NNLS subproblems are convex and the global minimizers can be computed. For the convergence of Algorithm 6.2.1, Grippo and Sciandrone [42] proved that every limit point of the sequences $\{(H^{k+1}, W^{k+1})\}_{k=0}^{+\infty}$ is a stationary point of NMF (6.2), where for each k, H^{k+1} and W^{k+1} are the global minimizers of the subproblems. Note that the uniqueness of the global minimizer for each subproblem, which is not a must for the convergence of Algorithm 6.2.1, may not be guaranteed, since W^k and $(H^{k+1})^\top$ are not necessary to be full column rank. See [7] for more convergence conditions of general block coordinate descent method. Consequently, the solution of NMF problem (6.2) is transformed to the solution of a series of NNLS problems with multiple right hand sides, which can be solved by the efficient iterative methods, e.g., gradient descent methods, active set methods, quasi-Newton methods, etc. We will briefly review some classical and fast existing NMF methods as follows. See [10, 23, 29] for more detailed review and survey.

Algorithm 6.2.1. Alternating Nonnegative Least Squares (ANLS)

```
1.    Choose initial matrices $H^0$ and $W^0$.
2.    For $k = 0, 1, 2, \ldots$ until convergence
3.        Compute $H^{k+1}$ by solving
              $$\min \|V - W^k H\|_F^2 \quad \text{subject to} \quad H \geq \mathbf{0}. \qquad (6.3)$$
4.        Compute $W^{k+1}$ by solving
              $$\min \|V - W H^{k+1}\|_F^2 \quad \text{subject to} \quad W \geq \mathbf{0}. \qquad (6.4)$$
5.    Endfor
```

The gradient descent methods are widely used for the NMF problem due to the fact that the monotonically nonincreasing of the objective function $f(W,H)$ can be guaranteed along the negative gradient direction with sufficient small step size, and it is easy to implement than the Newton-type method as only the gradient is computed. The first well known NMF method is multiplicative update (MU) method proposed by Lee and Seung [69], which can be regarded as diagonally rescaled gradient descent method. Unfortunately, Gonzales and Zhang [47] showed that the MU method may fail to converge to a stationary point and presented an accelerated MU method. Lin further presented a modified MU method [64] to overcome the disadvantages of MU method and analyzed its convergence. Although they are slow to converge in terms of the iteration numbers, the MU method and its variants require small computational costs per iteration. Therefore, they had become benchmarks against which the newer methods are compared. The original MU method [69] is listed in Algorithm 6.2.2, where '$\circ$' and '$\oslash$' represent for the Hadamard (componentwise) product and division, respectively.

Algorithm 6.2.2. Multiplicative Update (MU)

```
1.    Choose initial matrices $H^0$ and $W^0$.
2.    For $k = 0,1,2,\ldots$ until convergence
3.        $H^{k+1} = H^k \circ ((W^k)^T V) \oslash ((W^k)^T W^k H^k)$.
4.        $W^{k+1} = W^k \circ (V(H^{k+1})^T) \oslash (W^k H^{k+1} (H^{k+1})^T)$.
5.    Endfor
```

Instead of choosing the step size to maintain the nonnegativity in MU methods, the projected gradient method, which is another common gradient descent method, sets the negative components in the updated matrices to be zero. For example, Lin presented projected gradient method in [63] for NMF problem, which was originally proposed by Bertsekas [5] for general constrained convex optimization. The convergence can be proved under certain assumptions and the nonincreasing property can be obtained by sufficient decrease condition, or Armijo condition. For the gradient descent methods, it is known that the convergence can be slow and very sensitive to the choice of step size. Also similar to the steepest descent method, they may suffer from the zigzag phenomenon if the condition number is large.

The active set methods are also widely discussed and proposed for the solution of NMF problem. The idea of active set method is if the constraints active at the exact solution are known in advance, then the constrained problem can be solved by simply optimizing the objective function in an unconstrained manner over only the variables that correspond to the inactive constraints. Lawson and Hanson proposed a classical active set algorithm [67] for the solution of NNLS problem, which was further developed by Bro and De Jong [11] and van Brethem and Keenan [97] for NNLS with multiple right hand sides. Kim and Park [58] applied their active set method directly for the solution of subproblems in the ANLS framework. Although the strategies including precomputing the terms in the normal equations and grouping the columns with the same active set are proposed to save the computational costs, such active set method had an inefficient feature that typically only one variable index was exchanged between the active set and inactive set at each iteration. In order to avoid the disadvantage, various sophisticated active set methods for NNLS with single right hand side have been proposed by allowing more variable indices to exchange. For example, Júdice and Pires proposed a block principal pivoting (BPP) method [55] for solving the linear complementarity problem resulted from the equivalent Karush-Kuhn-Tucker (KKT) conditions of the NNLS problem. Moré and Toraldo [76, 77] proposed a gradient projection conjugate gradient (GPCG) method for the solution of box constrained quadratic programming. Moreover, Hager and Zhang proposed a new active set method [54] for general box constrained optimization. For the extensions of these active set methods to NMF problem in the ANLS framework, Kim and Park combined the BPP with the grouping idea, Zhang et al. [102] proposed a matrix-based variant of Hager and Zhang's method [54], and Cichocki et al. [29] applied the GPCG method directly to each column of the matrices. However, these extensions may be inefficient when the size of the problem as well as r are large.

In this paper, based on the ANLS framework, we first introduce the variant of modulus-type inner outer iteration method [103] for the solution of NNLS subproblems. Theoretical convergence that the limit of the sequence generated by the proposed method is a stationary point of NMF can be guaranteed. Moreover, by considering that the methods for NMF can be regarded as the fixed-point iteration, which the rate of convergence is at best linear, we also propose a new matrix-based active set method and employ Anderson acceleration, respectively, to further speed up the algorithms. Finally, we show that the proposed methods can be extended to solve the sparse NMF and regularized NMF. Numerical experiments on the synthetic data and ORL face image data show that the proposed methods converges faster than the gradient descent methods.

Remark that an open question that deserves further attention for NMF is the initialization. Most methods start the iterations with random nonnegative matrices. In order to speed up the convergence and to avoid the randomness, Wild et al. [99] employed a spherical k-means clustering approach and Boutsidis and Gallopoulos [13] proposed a partial SVD-based initialization method, respectively. Even though the suitable initial approximate matrices are of importance for the convergence in solving

NMF problem due to its nonconvex property, the details of the initialization will not be discussed here.

6.3 Modulus-Type Inner Outer Iteration Method

In this section, we apply an inner outer iterative method with modulus variable transformation, which originally proposed for the solution of NNLS [103], to solve NMF in the ANLS framework.

According to the KKT conditions, (W,H) is a stationary point of NMF (6.2) if and only if

$$H \geq 0, \quad \nabla_H f(W,H) \geq 0, \quad \langle H, \nabla_H f(W,H) \rangle = 0, \tag{6.5}$$

$$W \geq 0, \quad \nabla_W f(W,H) \geq 0, \quad \langle W, \nabla_W f(W,H) \rangle = 0, \tag{6.6}$$

where the gradient of objective function $f(W,H)$ with respect to W and H are

$$\nabla_W f(W,H) = (WH - V)H^{\mathrm{T}} \quad \text{and} \quad \nabla_H f(W,H) = W^{\mathrm{T}}(WH - V),$$

respectively. Here, $\langle A,B \rangle := \sum_i \sum_j A_{ij} B_{ij}$ refers to the inner product of two matrices. Note that (6.5) and (6.6) are also the KKT conditions of the subproblems (6.3) and (6.4), respectively. In order to handle the nonnegative constraints, the modulus variable transformation set

$$H = Z + |Z| \quad \text{and} \quad \nabla_H f(W,H) = \Omega_1(|Z| - Z), \tag{6.7}$$

where $Z \in \mathbf{R}^{r \times n}$ and $\Omega_1 \in \mathbf{R}^{r \times r}$ is a positive diagonal matrix. Hence, the KKT conditions (6.5) are equivalent to an implicit fixed-point equation

$$(\Omega_1 + W^{\mathrm{T}}W)Z = (\Omega_1 - W^{\mathrm{T}}W)|Z| + W^{\mathrm{T}}V,$$

which can be solved by the fixed-point iteration FPI(W, V, Ω_1)

$$(\Omega_1 + W^{\mathrm{T}}W)Z^{i+1} = (\Omega_1 - W^{\mathrm{T}}W)|Z^i| + W^{\mathrm{T}}V, \tag{6.8}$$

where $i = 1, 2, \ldots$. It is noted that the iterative scheme (6.8) can be reorganized as the normal equations

$$\widetilde{W}^{\mathrm{T}}\widetilde{W}Z^{i+1} = \widetilde{W}^{\mathrm{T}}\widetilde{V}^{i},$$

of the unconstrained least squares problem $\mathrm{LS}(\widetilde{W}, \widetilde{V}^{i})$

$$\min_{Z^{i+1}\in\mathbf{R}^{r\times n}} \left\| \widetilde{W}Z^{i+1} - \widetilde{V}^{i} \right\|_2 \tag{6.9}$$

for $i = 0, 1, 2, \ldots$, where

$$\widetilde{W} = \begin{bmatrix} W \\ \Omega_1^{\frac{1}{2}} \end{bmatrix} \quad \text{and} \quad \widetilde{V}^{i} = \begin{bmatrix} -W|Z^{i}| + V \\ \Omega_1^{\frac{1}{2}}|Z^{i}| \end{bmatrix}.$$

Therefore, the solution of the NNLS subproblem (6.3) can be transformed to the solution of a series of unconstrained least squares problems $\mathrm{LS}(\widetilde{W}, \widetilde{V}^{i})$, for which the efficient iterative methods can be applied.

Similarly by setting

$$W = (Y + |Y|)^{\mathrm{T}} \quad \text{and} \quad \nabla_W f(W,H) = [\Omega_2(|Y| - Y)]^{\mathrm{T}}, \tag{6.10}$$

where $Y \in \mathbf{R}^{r\times m}$ and $\Omega_2 \in \mathbf{R}^{r\times r}$ is a positive diagonal matrix, the KKT conditions (6.6) is equivalent to an implicit fixed-point equation

$$(\Omega_2 + HH^{\mathrm{T}})Y = (\Omega_2 - HH^{\mathrm{T}})|Y| + HV^{\mathrm{T}},$$

which can be solved by the fixed-point iteration $\mathrm{FPI}(H^{\mathrm{T}}, V^{\mathrm{T}}, \Omega_2)$

$$(\Omega_2 + HH^{\mathrm{T}})Y^{i+1} = (\Omega_2 - HH^{\mathrm{T}})|Y^{i}| + HV^{\mathrm{T}}, \tag{6.11}$$

where $i = 1, 2, \ldots$. It is noted that the iterative scheme (6.11) can be reorganized as the normal equations

$$\widehat{H}\widehat{H}^{\mathrm{T}}Y^{i+1} = \widehat{H}(\widehat{V}^{i})^{\mathrm{T}},$$

of the unconstrained least squares problem $\mathrm{LS}(\widehat{H}^{\mathrm{T}}, (\widehat{V}^i)^{\mathrm{T}})$

$$\min_{Y^{i+1}\in\mathbf{R}^{r\times m}} \left\|\widehat{H}^{\mathrm{T}}Y^{i+1} - (\widehat{V}^i)^{\mathrm{T}}\right\|_2 \tag{6.12}$$

for $i = 0, 1, 2, \ldots$, where

$$\widehat{H} = \begin{bmatrix} H & \Omega_2^{\frac{1}{2}} \end{bmatrix} \quad \text{and} \quad \widehat{V}^i = \begin{bmatrix} -|(Y^i)^{\mathrm{T}}|H + V & |(Y^i)^{\mathrm{T}}|\Omega_2^{\frac{1}{2}} \end{bmatrix}.$$

Similarly, the solution of the NNLS subproblem (6.4) can be transformed to the solution of a series of unconstrained least squares problems $\mathrm{LS}(\widehat{H}^{\mathrm{T}}, (\widehat{V}^i)^{\mathrm{T}})$.

The modulus-type inner outer iteration method in the ANLS framework for NMF problem is described as follows.

Algorithm 6.3.1. ANLS - Modulus-Type Inner Outer Iteration

```
1.   Choose initial matrices $H^0$ and $W^0$, and parameter matrices $\Omega_1$ and $\Omega_2$.
2.   For $k = 0, 1, 2, \ldots$ until convergence
3.       For $i = 0, 1, 2, \ldots$ until obtaining $Z^*$
4.           Compute $Z^{i+1}$ by solving $\mathrm{LS}(\widetilde{W}^k, \widetilde{V}^i)$.
5.       Endfor
6.       Compute $H^{k+1} = Z^* + |Z^*|$.
7.       For $i = 0, 1, 2, \ldots$ until obtaining $Y^*$
8.           Compute $Y^{i+1}$ by solving $\mathrm{LS}((\widehat{H}^{k+1})^T, (\widehat{V}^i)^T)$.
9.       Endfor
10.      Compute $W^{k+1} = Y^* + |Y^*|$.
11.  Endfor
```

Here, lines 3-6 and 7-10 correspond to the solution of the NNLS subproblems (6.3) and (6.4), respectively. The iterative solution of the $\mathrm{LS}(\widetilde{W}^k, \widetilde{V}^i)$ in Step 4 or $\mathrm{LS}((\widehat{H}^{k+1})^{\mathrm{T}}, (\widehat{V}^i)^{\mathrm{T}})$ in Step 8 for each i is referred to as the inner iteration. For the single right hand side case, the overdetermined least square problem can be solved by the (preconditioned) CGLS, LSQR, LSMR, BA-GMRES [53] methods, etc. In the following, we derive a simple extension of CGLS method for the overdetermined least squares problem with multiple right hand sides $\mathrm{LS}(A,B)$

$$\min \|AX - B\|_F, \tag{6.13}$$

where $A \in \mathbf{R}^{m\times n}$ and $B \in \mathbf{R}^{m\times p}$ are given, and $X \in \mathbf{R}^{n\times p}$ is unknown. The extension for other solvers can be derived similarly.

It is noted that (6.13) is equivalent to

$$\min \|(I \otimes A)\mathrm{vec}(X) - \mathrm{vec}(B)\|_2, \tag{6.14}$$

where $I \in \mathbf{R}^{p\times p}$ is an identity matrix, $\otimes$ denotes the Kronecker product between two matrices, and $\mathrm{vec}(\cdot)$ denotes the vectorization of a matrix by stacking its column vectors on top of one another. Since $I \otimes A$ is a $mp \times np$ matrix, and $\mathrm{vec}(X)$ and $\mathrm{vec}(B)$ are $np \times 1$ and $mp \times 1$ vectors, respectively, we can apply the CGLS method to solve (6.14) directly. Then, by reorganizing the algorithm in the matrix form with the inverse of the vectorization, the CGLS method for $\mathrm{LS}(A,B)$ can be obtained and listed as follows.

Algorithm 6.3.2. CGLS Method for LS(A,B)

```
1.    Choose initial $X^0$.
2.    Compute $R^0 = B - AX^0$, $S^0 = A^T R^0$ and $P^0 = S^0$.
3.    For $k = 0,1,2,\ldots$ until convergence
4.        $\alpha_k = \langle S^k, S^k\rangle / \langle AP^k, AP^k\rangle$
5.        $X^{k+1} = X^k + \alpha_k P^k$
6.        $R^{k+1} = R^k - \alpha_k AP^k$
7.        $S^{k+1} = A^T R^{k+1}$
8.        $\beta_{k+1} = \langle S^{k+1}, S^{k+1}\rangle / \langle S^k, S^k\rangle$
9.        $P^{k+1} = S^{k+1} + \beta_{k+1} P^k$
10.   Endfor
```

Remark that a more standard derivation of conjugate gradient method for the symmetric positive definite linear equations with multiple right hand sides can be found in [21]. The difference is the computation of step size α_k and β_{k+1} in lines 3 and 8.

As mentioned previously, for Algorithm 6.3.1, the merit is by modulus variable transformations (6.7) and (6.10), the solution of NMF problem can be transformed to the solution of a series of unconstrained least squares problems (6.9) and (6.12), or fixed point equations (6.8) and (6.11), at each inner iteration, which can be solved efficiently. However, the demerit is that the outer iteration, which can be regarded as the fixed-point iteration, is at best linear convergence rate. Hence, we consider the active set strategy and the Anderson extrapolation to accelerate the NMF algorithms in the following.

6.4 Active Set Method for NNLS with Multiple Right Hand Sides

In this section, we consider the active set method for NNLS problem with multiple right hand sides.

First, we review active set method for NNLS problem with single right hand side

$$\min \|Ax-b\|_2 \quad \text{subject to} \quad x \geq \mathbf{0},$$

where $A \in \mathbf{R}^{m\times n}$ and $b \in \mathbf{R}^{m\times 1}$ are given, and $x \in \mathbf{R}^{n\times 1}$ is the unknown. Active set and free variables (inactive) set for the kth step iterative solution are defined as

$$\mathscr{B}(x^k) = \{j:\ x_j^k = 0,\ \lambda_j^k \geq 0\} \quad \text{and} \quad \mathscr{F}(x^k) = \{1,2,\ldots,n\} \backslash \mathscr{B}(x^k),$$

where $\lambda^k = -A^{\mathrm{T}}r^k = -A^{\mathrm{T}}(b-Ax^k)$ is the gradient of the quadratic function in (1.1). The idea of the active set method is that if the active set of x^* is known, where x^* is the global minimizer of (1.1), then the NNLS can be solved by simply optimizing the quadratic function in an unconstrained manner over only the free variables set

$$\min \|A_{\mathscr{F}} x_{\mathscr{F}}^{k+1} - b\|_2, \tag{6.15}$$

where $A_{\mathscr{F}}$ is the submatrix of A consisting of the columns of A whose indices belong to $\mathscr{F}$, and $x_{\mathscr{F}}$ is the reduced vector of x consisting of the elements of x whose indices belong to $\mathscr{F}$. One significant task for the active set method is when to terminate the solution of the unconstrained least squares problem (6.15) and update the active set $\mathscr{B}$ and free variable set $\mathscr{F}$ if some of its components violate the nonnegative constraints. For example, the active set methods proposed by O'Leary [81] and Lawson and Hanson

[?] terminate the inner iteration as soon as a component violates a constraint, which forces frequent resuming of the iteration and typically only one index from active set and free variable set is exchanged at a time. In order to avoid these disadvantages, the active set methods proposed by Moré and Toraldo [76, 77] and Morigi et al. [74] allow more consecutive iterations to solve (6.15) and more elements to update in the active set.

In terms of the NNLS problem with multiple right hand sides

$$\min \|AX - B\|_2 \quad \text{subject to} \quad X \geq \mathbf{0}, \tag{6.16}$$

where $B = \begin{bmatrix} b_1 & b_2 & \cdots & b_p \end{bmatrix} \in \mathbf{R}^{m\times p}$ are given and $X = \begin{bmatrix} x_1 & x_2 & \cdots & x_p \end{bmatrix} \in \mathbf{R}^{n\times p}$ is the unknown, the straightforward way is to decompose (6.16) as p NNLS problems with single right hand side

$$\min \|Ax_i - b_i\|_2 \quad \text{subject to} \quad x_i \geq \mathbf{0},$$

where $i = 1,2,\ldots,p$, and then apply the active set method to each NNLS separately. Since the active sets correspond to each right hand side b_i are updated differently, it is inefficient to handle (6.16) as p NNLS problems independently with active set method comparing to the nonactive set method. In order to save the computational costs, Bro and De Jong [11] and van Brethem and Keenan [97] proposed a grouping technique in Lawson and Hanson's active set method [67], and Kim and Park [59] utilize the same idea in the block principal pivoting method for NMF. For each iteration, those columns that correspond to the same active sets are grouped and the resulting unconstrained least squares problems are solved together by apply Cholesky factorization for the normal equations. The grouping idea is illustrated in Figure 6.2, which comes from [59].

The grouping technique may be inefficient when the number of the groups is large. In other words, the grouping fails to work if the active sets for each right hand sides are different with each other. Also, the frequent change of the groups at each iteration may become time consuming. In the following, we propose a new active set strategy to avoid these inefficient features.

For (1.1), by introducing $D^k = \operatorname{diag}(d_1^k, d_2^k, \ldots, d_n^k)$ with entries

$$d_i^k = \begin{cases} 1, & \textbf{if } i \in \mathscr{F}(x^k); \\ 0, & \textbf{if } i \in \mathscr{B}(x^k), \end{cases}$$

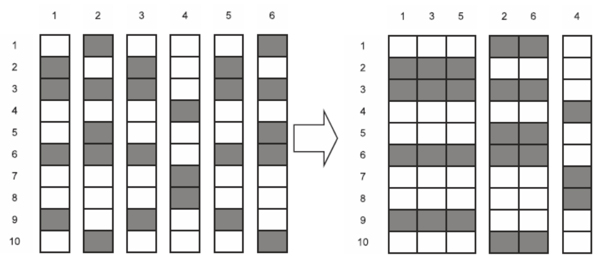


Figure 6.2: ([59]) An example in of the grouping of right hand sides when $n = 10$ and $p = 6$. Dark cells indicate variables with indices in $\mathscr{F}$, which need to be computed by (6.15). By grouping the columns that have a common $\mathscr{F}$ set, i.e., columns $\{1,3,5\}$, $\{2,6\}$, and $\{4\}$, we can avoid redundant computation for Cholesky factorization in solving the normal equation of (6.15).

it can be obtained that $A_{\mathscr{F}} x_{\mathscr{F}}^{k+1} = AD^k x^{k+1}$ and thus (6.15) is equivalent to

$$\min \|AD^k x^{k+1} - b\|_2. \tag{6.17}$$

Such formulation was proposed by Morigi et al. [74] by assuming that the matrix can only be accessed through the evaluation of matrix vector products with A and A^{T}, which is a common situation in image restoration problems. Furthermore, by using Hadamard product, the (6.17) is also equivalent to

$$\min \|A(d^k \circ x^{k+1}) - b\|_2, \tag{6.18}$$

where $d^k = \begin{bmatrix} d_1 & d_2 & \cdots & d_n \end{bmatrix}^{\mathrm{T}}$. Note that $d^k \circ x^{k+1}$ vanishes components whose indices belong to the active set $\mathscr{B}(x^k)$.

Finally, we are ready to generalize this idea for (6.16) by introducing the matrix $\Gamma \in \mathbf{R}^{n\times p}$ with the entries

$$\Gamma_{ij}^k = \begin{cases} 0, & \textbf{if } (i,j) \in \mathscr{B}(X^k); \\ 1, & \textbf{otherwise}, \end{cases} \tag{6.19}$$

where $\mathscr{B}(X^k) = \{(i,j) : X_{ij}^k = 0, \ \Lambda_{ij}^k \geq 0\}$ and $\Lambda^k = -A^{\mathrm{T}} R^k = -A^{\mathrm{T}}(B - AX^k)$ is the gradient of the quadratic function in (6.16). Then, similar to (6.18) in the single right hand side case, the unconstrained least squares problem

$$\min \|A(\Gamma^k \circ X^{k+1}) - B\|_2, \tag{6.20}$$

can be solved over the free variable set since $\Gamma^k \circ X^{k+1}$ vanishes the components whose indices belong to the active set $\mathscr{B}(X^k)$.

The active set method for NNLS with multiple right hand sides (6.16) is described as follows.

Algorithm 6.4.1. Active Set Method for (6.16)

```
1.    Choose initial $X^0$.
2.    Compute $R^0 = B - AX^0$ and $\Lambda^0 = -A^T R^0$.
3.    For $k = 0, 1, 2, \ldots$ until convergence
4.        Update $\Gamma^k$ according to (6.19).
5.        Compute $N^{k+1}$ by solving
              $$\min \|A(\Gamma^k \circ N^{k+1}) - R^k\|_2, \tag{6.21}$$
6.        Set $X^{k+1} = P(X^k + \beta^m(\Gamma^k \circ N^{k+1}))$, and find the smallest integer
          $m \geq 0$ that satisfies the sufficient decrease condition
              $$\|B - AX^{k+1}\|_F^2 \leq \|B - AX^k\|_F^2 - 2\mu\langle -\Lambda^k, X^{k+1} - X^k\rangle, \tag{6.22}$$
          where $0 < \beta < 1$ and $0 \leq \mu < 1$.
7.        Compute $R^{k+1} = B - AX^{k+1}$
8.        Compute $\Lambda^{k+1} = -A^T R^{k+1}$
9.    Endfor
```

Algorithm 6.4.2. CGLS Method for (6.21)

```
1.    Choose initial $X^0$.
2.    Compute $R^0 = B - A(\Gamma \circ X^0)$, $S^0 = \Gamma \circ (A^T R^0)$ and $P^0 = S^0$.
3.    For $k = 0, 1, 2, \ldots$ until convergence
4.        $\alpha_k = \langle S^k, S^k\rangle / \langle A(\Gamma \circ P^k), A(\Gamma \circ P^k)\rangle$
5.        $X^{k+1} = X^k + \alpha_k P^k$
6.        $R^{k+1} = R^k - \alpha_k A(\Gamma \circ P^k)$
```

```
7.            $S^{k+1} = \Gamma \circ (A^T R^{k+1})$
8.            $\beta_k = \langle S^{k+1}, S^{k+1} \rangle / \langle S^k, S^k \rangle$
9.            $P^{k+1} = S^{k+1} + \beta_k P^k$
10.     Endfor
```

6.5 Sparse and Regularized NMF

Penalty constraint with Frobenius norm:

$$\min \|V - WH\|_F^2 + \alpha \|W\|_F^2 + \beta \|H\|_F^2,$$

Alternating nonnegative least squares method

$$\begin{aligned}
\min \|V - W^k H\|_F^2 + \beta \|H\|_F^2 &= \min \left\| \begin{bmatrix} V \\ 0 \end{bmatrix} - \begin{bmatrix} W^k \\ \sqrt{\beta} I \end{bmatrix} H \right\|_F^2 \\
&:= \min \|\bar{V} - \bar{W}^k H\|_F^2 \\
\min \|V - WH^{k+1}\|_F^2 + \alpha \|W\|_F^2 &= \min \left\| \begin{bmatrix} V & 0 \end{bmatrix} - W \begin{bmatrix} H^{k+1} & \sqrt{\alpha} I \end{bmatrix} \right\|_F^2 \\
&:= \min \|\tilde{V} - W \tilde{H}^{k+1}\|_F^2,
\end{aligned}$$

where I is an identity matrix.

Sparsity constraint with l_1-norm:

$$\min \|V - WH\|_F^2 + \alpha \sum_{i=1}^{m} \|w_i\|_1^2 + \beta \sum_{j=1}^{n} \|h_j\|_1^2,$$

where w_i^{T} and h_j are ith row vector of W and jth column vector of W, respectively.

Alternating nonnegative least squares method

$$\begin{aligned}
\min \|V - W^k H\|_F^2 + \beta \sum_{j=1}^{n} \|h_j\|_1^2 &= \min \left\| \begin{bmatrix} V \\ 0 \end{bmatrix} - \begin{bmatrix} W^k \\ \sqrt{\beta}\mathbf{e}^{\mathrm{T}} \end{bmatrix} H \right\|_F^2 \\
&:= \min \|\bar{V} - \bar{W}^k H\|_F^2 \\
\min \|V - WH^{k+1}\|_F^2 + \alpha \sum_{i=1}^{m} \|w_i\|_1^2 &= \min \left\| \begin{bmatrix} V & 0 \end{bmatrix} - W \begin{bmatrix} H^{k+1} & \sqrt{\alpha}\mathbf{e} \end{bmatrix} \right\|_F^2 \\
&:= \min \|\tilde{V} - W\tilde{H}^{k+1}\|_F^2,
\end{aligned}$$

where $\mathbf{e}$ is a column vector with all components equal to one.

6.6 Numerical Experiments

Finally, we compare the proposed modulus (Mod) method with the existing methods including multiplicative update (MU) method, projected gradient (PG) method, projected gradient method with Armijo condition (PGA). The testing problems contain

Synthetic data: Consider matrix V is randomly generated by the normal distribution with mean 0 and standard deviation 1

$$V_{ij} = |N(0,1)|.$$

The initial matrices are also constructed randomly. The size of the problem is $(m,r,n) = (100,20,500)$. Such generation of matrix can also be seen in [**?**].

Image data: ORL face image database [3].

The programming language is MATLAB 7.8 with machine precision $\varepsilon = 1.1 \times 10^{-16}$. The initial matrices were chosen to be random matrices. For the modulus-type iteration methods, the parameter matrix was chosen to be $\Omega = \omega I$, where ω is a positive parameter. the stopping criterion for the outer iteration of all methods is chosen as

$$\frac{|f(W^{k+1},H^{k+1}) - f(W^k,H^k)|}{f(W^0,H^0)} < tol. \tag{6.23}$$

Table 6.1: Comparison of the iterative methods for random problem.

Methods	Iterations	$f(W,H)$	CPU
MU	3468	111.90	10.09
PG	54	115.47	0.64
PGA	47	112.93	5.52
Mod	52	111.88	5.14

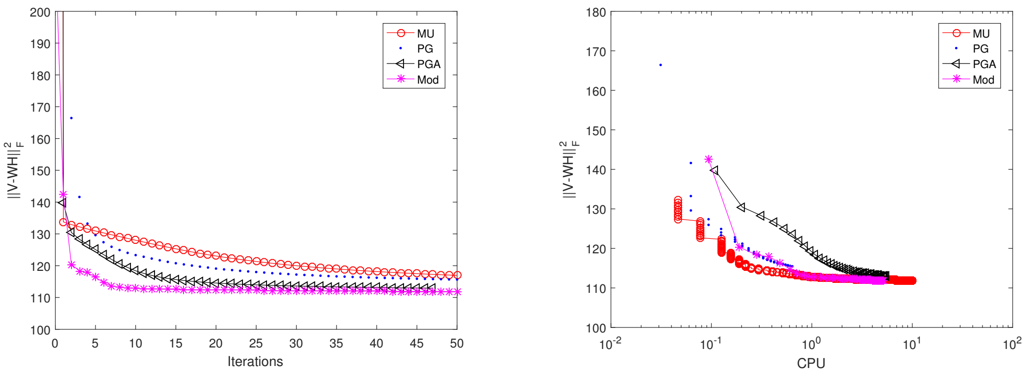


Figure 6.3: Objective function value versus iterations (left) and CPU time in seconds (right), respectively, for random problem.

In order to perform a fair comparison among different methods, the parameters are chosen as

$$\mu = 0.1, \quad \beta = 0.9 \quad \text{and} \quad \omega = 1 \tag{6.24}$$

for all testing problems. In addition, the maximum number of iteration steps is restricted to be 5,000.

In Table 6.1, we show the numerical results of four methods for synthetic data from the aspect of the number of iterations ("Iterations"), the objective function value ("$f(W,H)$") and the CPU time in seconds ("CPU"). Meanwhile, in Figure 6.3, we show the objective function value with respect to the iterations and the CPU time. In Figure 6.4, the condition numbers of W and H with respect to the iterations are depicted, respectively.

From Table 6.1, it is observed that the Mod obtains the least objective function value among the four methods, and requires less CPU time than MU and PGA. Although the PG method is the fastest method, the objective function value is relative higher than the

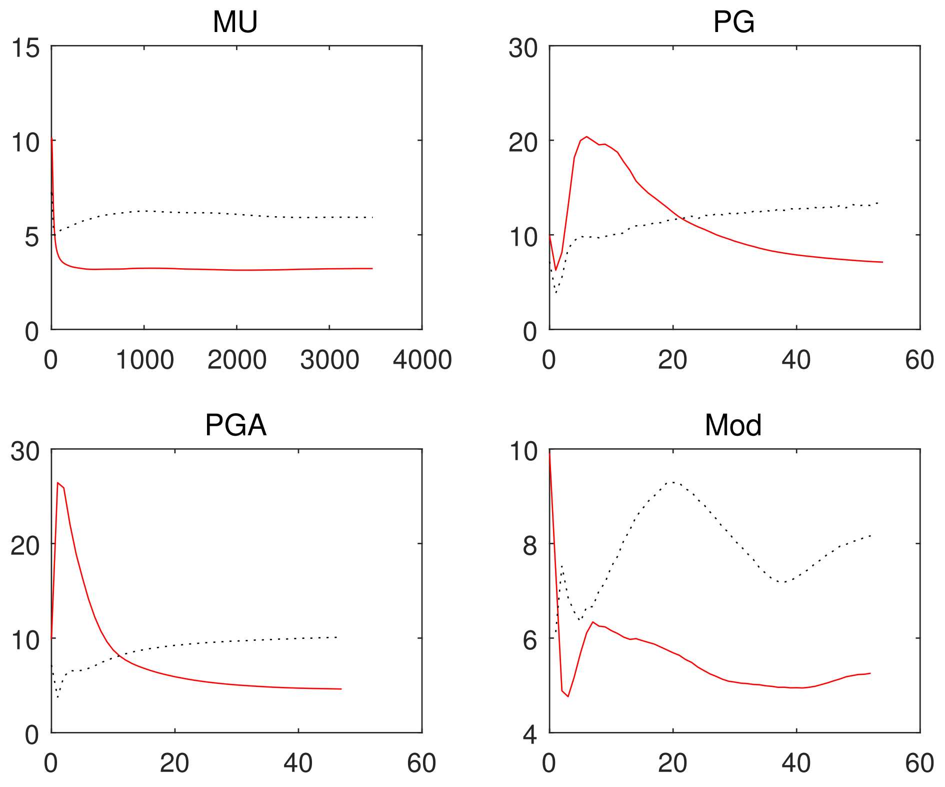


Figure 6.4: Condition numbers of W (red line) and H (black dot line) versus iterations.

other methods.

In Table 6.2, we show the numerical results of four methods for synthetic data from the aspect of the number of iterations (“Iterations"), the objective function value (“$f(W,H)$") and the CPU time in seconds (“CPU"). Meanwhile, in Figure 6.5, we show the objective function value with respect to the iterations and the CPU time. In Figure 6.6, the condition numbers of W and H with respect to the iterations are depicted, respectively. In Figure 6.7, the face images of four methods and the original images are shown.

We can reach the conclusion that the proposed alternating modulus method is quite competitive among all the testing methods. It outperforms the other methods with less iterations and CPU time. In addition, the parameter in modulus method is not sensitive since the condition numbers of W and H is small during the iterations.

Table 6.2: Comparison of the iterative methods for ORL facedata problem.

Methods	Iterations	$f(W,H)$	CPU
MU	88	50346.85	14.28
PG	114	45296.18	63.02
PGA	17	45372.30	92.58
Mod	19	45900.87	21.83

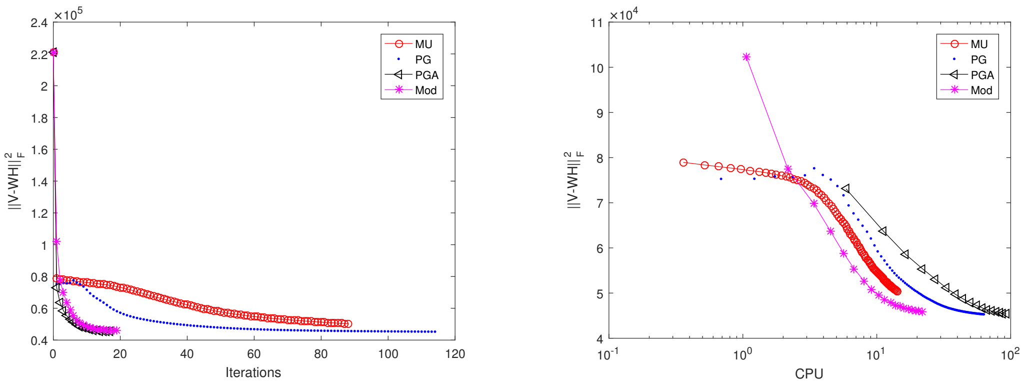


Figure 6.5: Objective function value versus iterations (left) and CPU time in seconds (right), respectively, for random problem.

6.7 Concluding Remarks

In this chapter, we consider a new alternating nonnegative least squares method using modulus-type inner outer iteration method for the nonnegative constrained least squares subproblem. Numerical experiments on the synthetic data and ORL face image data show that the proposed methods converges faster than the gradient descent methods.

In the future work, we will consider the NMF with sparsity constraints on W or H. One way is to add penalty terms to the NMF objective function (Hoyer, 02')

$$\min \frac{1}{2}\|V-WH\|_F^2+\alpha\|W\|_F^2+\beta\|H\|_F^2,$$

where α and β are positive parameters.

The alternating nonnegative least squares method can be applied similarly as follows.

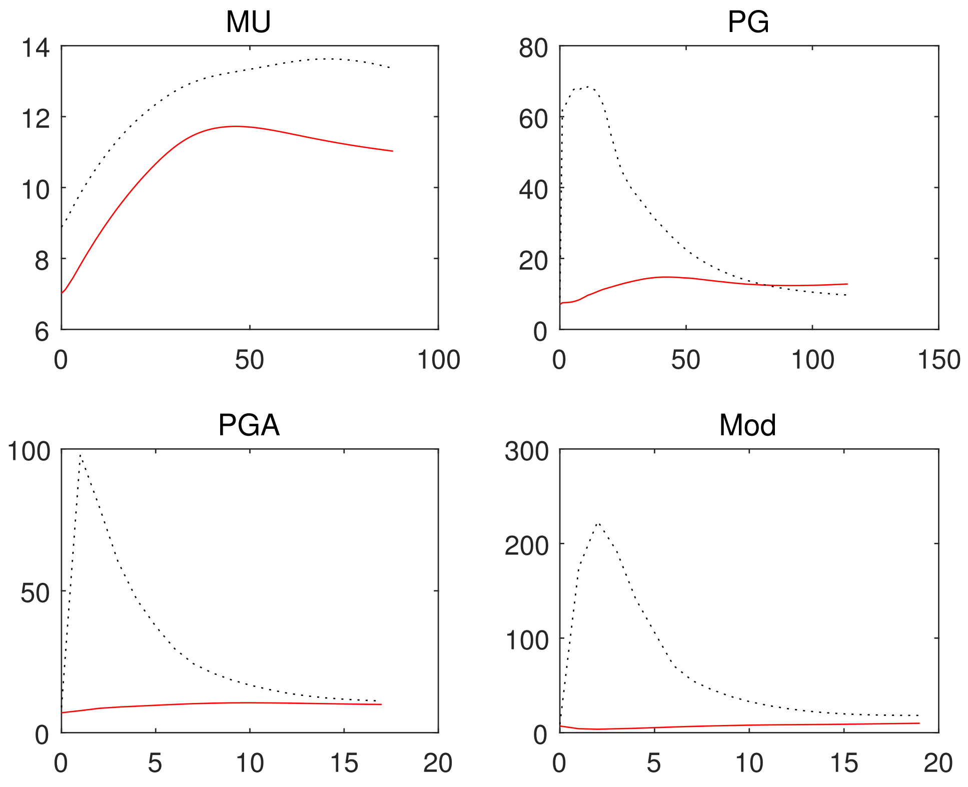


Figure 6.6: Condition numbers of W (red line) and H (black dot line) versus iterations.

$$\begin{aligned}
\min \|V - W^k H\|_F^2 + \beta \|H\|_F^2 &= \min \left\| \begin{bmatrix} V \\ 0 \end{bmatrix} - \begin{bmatrix} W^k \\ \sqrt{\beta} I \end{bmatrix} H \right\|_F^2 \\
&:= \min \|\bar{V} - \bar{W}^k H\|_F^2 \\
\min \|V - W H^{k+1}\|_F^2 + \alpha \|W\|_F^2 &= \min \left\| \begin{bmatrix} V & 0 \end{bmatrix} - W \begin{bmatrix} H^{k+1} & \sqrt{\alpha} I \end{bmatrix} \right\|_F^2 \\
&:= \min \|\tilde{V} - W \tilde{H}^{k+1}\|_F^2,
\end{aligned}$$

where I is an identity matrix.

Another way is to introduce L_1-norm based sparsity constraints

$$\min \|V - WH\|_F^2 + \alpha \sum_{i=1}^{m} \|w_i\|_1^2 + \beta \sum_{j=1}^{n} \|h_j\|_1^2,$$

where w_i^{T} and h_j are ith row vector of W and jth column vector of W, respectively. Similarly, alternating nonnegative least squares method can be applied as follows.

$$\begin{aligned}
\min \|V - W^k H\|_F^2 + \beta \sum_{j=1}^{n} \|h_j\|_1^2 &= \min \left\| \begin{bmatrix} V \\ 0 \end{bmatrix} - \begin{bmatrix} W^k \\ \sqrt{\beta}\mathbf{e}^{\mathrm{T}} \end{bmatrix} H \right\|_F^2 \\
&:= \min \|\bar{V} - \bar{W}^k H\|_F^2 \\
\min \|V - WH^{k+1}\|_F^2 + \alpha \sum_{i=1}^{m} \|w_i\|_1^2 &= \min \left\| \begin{bmatrix} V & 0 \end{bmatrix} - W \begin{bmatrix} H^{k+1} & \sqrt{\alpha}\mathbf{e} \end{bmatrix} \right\|_F^2 \\
&:= \min \|\tilde{V} - W\tilde{H}^{k+1}\|_F^2,
\end{aligned}$$

where $\mathbf{e}$ is a column vector with all components equal to one. We will apply the proposed methods to solve the above sparse NMF problem, and show the numerical results.

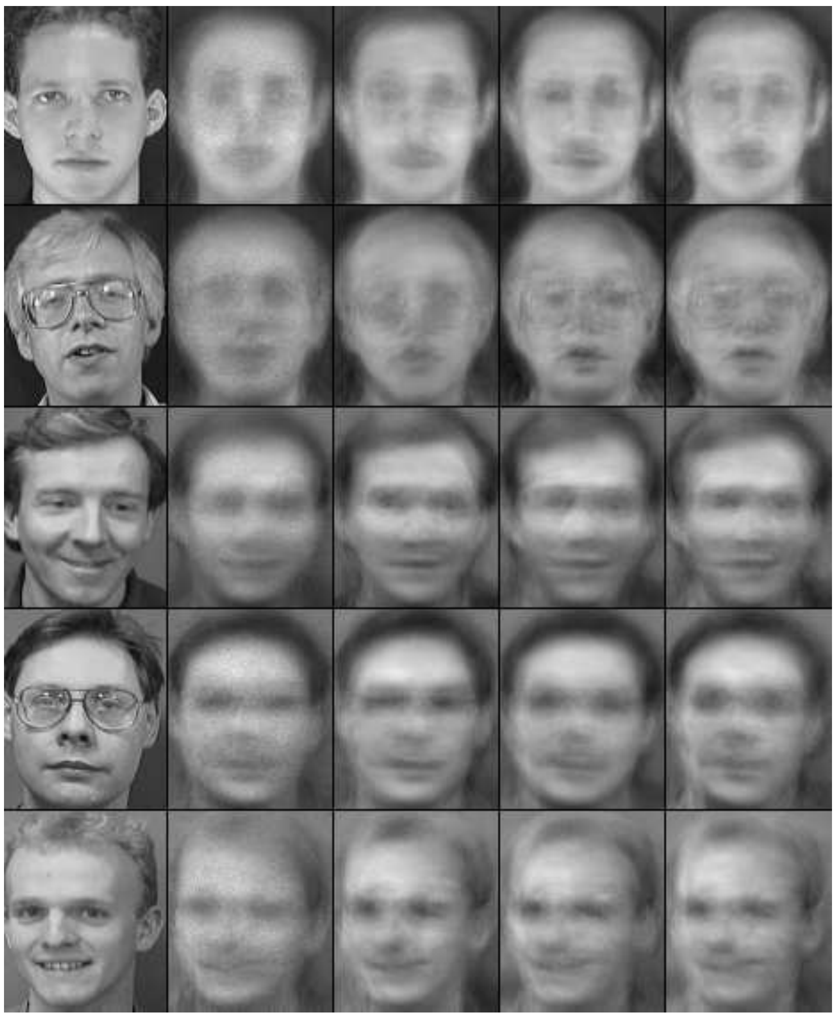


Figure 6.7: Images of the original facedata (first column), MU (second column), PG (third column), PGA (fourth column) and Mod (fifth column).

CHAPTER 7

ANDERSON ACCELERATION

In this chapter, we attempt to accelerate the numerical methods that proposed in the previous chapters, by employing Anderson acceleration [2]. The Anderson acceleration, also known as Anderson mixing, which is designed for fixed-point problems. While fixed-point iteration uses only the current iterate to define the next one, Anderson acceleration uses the additional information from the m_k previous iterations and computes the new iterate as a specific linear combination of these $m_k + 1$ quantities. The selected history length m_k is usually small. A discussion that puts Anderson acceleration in context with other acceleration methods can be found in [100].

The rest of the chapter is organized as follows. In Section 7.1, we briefly review the Anderson acceleration scheme. In Sections 7.2 and 7.3, we apply the Anderson acceleration to the stationary method for solving linear equations and the modulus-based methods for solving linear complementarity problems, respectively. The solution of the latter problem implies that the Anderson acceleration can also be applied for the solution of NNLS and BLS problems. In Sections 7.4, the Anderson acceleration is used to accelerate the alternating nonnegative least squares methods for NMF. Finally in Section 7.5, the concluding remarks are given.

7.1 Introduction

The fixed-point equation $x = g(x)$ is fixed-point iteration

$$x_{k+1} = g(x_k), \tag{7.1}$$

where $g : \mathbb{R}^n \to \mathbb{R}^n$ and x_0 is given. It also called as the Richardson iteration or Picard iteration. In general the convergence is at a linear rate. A method to accelerate convergence is Anderson acceleration, which redefines x_{k+1} to make use of the information from the previous m_k steps. We first briefly outline the original method derived by Anderson [2].

Algorithm 7.1.1. Original Anderson acceleration

```
1.    Given $x_0$ and an integer $m \geq 1$.
2.    Set $x_1 = g(x_0)$
3.    For $k = 1, 2, \ldots$ until convergence
4.        $m_k = \min(m, k)$.
5.        Determine $\theta^k = [\theta_1^k, ..., \theta_{m_k}^k]^T$ that minimizes $\|u_k - v_k\|_2^2$, where
```

$$u_k = x_k + \sum_{j=1}^{m_k} \theta_j (x_{k-j} - x_k) \quad \textit{and} \quad v_k = g(x_k) + \sum_{j=1}^{m_k} \theta_j (g(x_{k-j}) - g(x_k)).$$

```
6.        Set $x_{k+1} = u_k + \beta_k (v_k - u_k)$ using the parameters from $\theta^k$.
5.    Endfor
```

Here, the usual choice in the literature is $\beta_k = 1$.

Next, we introduce an equivalent form of the method that stores in two matrices the differences of the successive iterates and their function values. These matrices are related by simple update formulae that can be exploited for an efficient implementation. This variant is given in [40, 100, 50]. Here, Anderson acceleration is applied to the equivalent problem $f(x) = 0$, where $f(x) = g(x) - x$.

Algorithm 7.1.2. Anderson acceleration

```
1.    Given $x_0$ and an integer $m \geq 1$.
2.    Set $x_1 = x_0 + f(x_0)$
```

```
3.    For $k = 1, 2, \dots$ until convergence
4.        $m_k = \min(m, k)$.
5.        Compute $\gamma^k = [\gamma^k_{k-m_k}, \dots, \gamma^k_{k-1}]^T$ that solves $\min_{\gamma \in \mathbf{R}^{\mathbf{m_k}}} \|f_k - \mathscr{F}_k \gamma\|$.
6.        Set $\bar{x}_k = x_k - \sum_{i=k-m_k}^{k-1} \gamma^k_i \Delta x_i = x_k - \mathscr{X}_k \gamma^k$.
7.        Set $\bar{f}_k = f_k - \sum_{i=k-m_k}^{k-1} \gamma^k_i \Delta f_i = f_k - \mathscr{F}_k \gamma^k$.
8.        Set $x_{k+1} = \bar{x}_k + \bar{f}_k$.
9.    Endfor
```

Here, the following notation is used

$$\Delta x_i = x_{i+1} - x_i, \quad \mathscr{X}_k = [\Delta x_{k-m_k}, \dots, \Delta x_{k-1}]$$
$$f_i = f(x_i), \quad \Delta f_i = f_{i+1} - f_i, \quad \mathscr{F}_k = [\Delta f_{k-m_k}, \dots, \Delta f_{k-1}].$$

Also, the main computations is to solving an unconstrained least squares problem in Line 5 of Algorithm 7.1.2. The usual way is to apply a QR factorization updating method [100].

In quantum chemistry Anderson accelration is known as Pulay mixing or direct inversion in the iterative subspace (DIIS) [86] and it has been widely used in electronic structure computations; see [90] and the references therein. Anderson acceleration is related to multisecant methods (extensions of quasi-Newton methods involving multiple secant conditions). Eyert [37] proves that it is equivalent to the so-called “bad" Broyden’s method [9], and a similar analysis is done by Fang and Saad [40] and Rohwedder and Schneider [90]. For linear systems, if $m_k = k$ for each k then Anderson acceleration is essentially equivalent to the generalized minimal residual (GMRES) method [92], as shown by Rohwedder and Schneider [90], and Walker and Ni [100]. For nonlinear problems, it is shown in [90] that Anderson acceleration is locally linearly convergent under certain conditions. Adding to the above convergence analysis is the recent work by Toth and Kelley [93] concerning Anderson acceleration with $m_k = \min(m, k)$, for a fixed m, applied to contractive mappings.

Even though there are no general guarantees of its convergence, Anderson acceleration has a successful record of use in electronic structure computations. Furthermore, it significantly improved the performance of several domain decomposition methods presented in [100] and has proved to be very efficient on various examples in the above references. Hence Anderson acceleration has great potential for enhancing the convergence of algorithms proposed in the previous chapters.

7.2 Linear Equations

In this section, we apply the Anderson acceleration to the stationary iterative methods for linear equations. Specifically, we apply the Anderson acceleration to Jacobi iteration and successive overrelaxation (SOR) iteration.

Recall that the stationary iteration methods for solving the systems of linear equations

$$Ax = b \tag{7.2}$$

are based on the matrix splitting $A = M - N$, where $A \in \mathbf{R}^{n\times n}$ and $M \in \mathbf{R}^{n\times n}$ are nonsingular. Hence, the fixed-point equation

$$Mx = Nx + b \quad \text{or} \quad x = M^{-1}Nx + M^{-1}b \tag{7.3}$$

can be solved iteratively by setting

$$Mx^{k+1} = Nx^k + b \quad \text{or} \quad x^{k+1} = M^{-1}Nx^k + M^{-1}b.$$

Let $A = D - L - U$, where D, L and U are the diagonal, the strictly lower-triangular and the strictly upper-triangular matrices of the matrix A. For the Jacobi and SOR iteration, the matrix splitting are

$$\begin{aligned} M &= D & \text{and} \qquad N &= L + U, \\ M &= \frac{1}{\omega}(D - \omega L) & \text{and} \qquad N &= \frac{1}{\omega}[(1-\omega)D + \omega U], \end{aligned}$$

respectively. The relaxation parameter $0 < \omega < 2$ and when $\omega = 1$ the SOR is reduced to Gauss-Seidel iteration.

The Jacobi and SOR iteration methods are proposed as follows.

Algorithm 7.2.1. Jacobi Iteration Method

1. Choose the initial approximation x^0.

```
2.    For $k = 0, 1, 2, \ldots$ until convergence
3.        Set $x^{k+1} = D^{-1}(L+U)x^k + D^{-1}b$
4.        Set $r^{k+1} = b - Ax^{k+1}$
5.    Endfor
```

Algorithm 7.2.2. SOR Iteration Method

```
1.    Choose the initial approximation $x^0$ and parameter $\omega$.
2.    For $k = 0, 1, 2, \ldots$ until convergence
3.        Solve $(D - \omega L)x^{k+1} = [(1-\omega)D + \omega U]x^k + \omega b$
4.        Set $r^{k+1} = b - Ax^{k+1}$
5.    Endfor
```

We can rewrite the Algorithms 7.2.1 and 7.2.2 as the fixed-point iteration schemes as (7.1)

$$x^{k+1} = g_{\text{jacobi}}(x^k) \quad \text{and} \quad x^{k+1} = g_{\text{sor}}(x^k).$$

Then, combined with the Anderson acceleration Algorithm 7.1.2, we have the Jacobi iteration with Anderson acceleration method and the SOR iteration with Anderson acceleration method.

Algorithm 7.2.3. Jacobi Iteration with Anderson Acceleration Method

```
1.    Choose the initial approximation $x^0$.
2.    For $k = 0, 1, 2, \ldots$ until convergence
3.        Generate sequence through $x^{k+1} = g_{\text{jacobi}}(x^k)$
4.        Apply Anderson acceleration Algorithm 7.1.2 for the sequence.
5.    Endfor
```

Algorithm 7.2.4. SOR Iteration with Anderson Acceleration Method

```
1.    Choose the initial approximation $x^0$.
2.    For $k = 0, 1, 2, \ldots$ until convergence
3.        Generate sequence through $x^{k+1} = g_{\text{sor}}(x^k)$
4.        Apply Anderson acceleration Algorithm 7.1.2 for the sequence.
5.    Endfor
```

Next, we construct numerical experiments for the system of linear equations, to test the performance of Algorithms 7.2.3 and 7.2.4. The linear systems $Ax = b$ are constructed where

$$A = \text{tridiag}(-lI, S, -rI) = \begin{pmatrix} S & -rI & 0 & \cdots & 0 & 0 \\ -lI & S & -rI & \cdots & 0 & 0 \\ 0 & -lI & S & \cdots & 0 & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & 0 & \cdots & S & -rI \\ 0 & 0 & 0 & \cdots & -lI & S \end{pmatrix} \in \mathbf{R}^{n \times n}$$

is a block-tridiagonal matrix,

$$S = \text{tridiag}(-l, 4, -r) = \begin{pmatrix} 4 & -r & 0 & \cdots & 0 & 0 \\ -l & 4 & -r & \cdots & 0 & 0 \\ 0 & -l & 4 & \cdots & 0 & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & 0 & \cdots & 4 & -r \\ 0 & 0 & 0 & \cdots & -l & 4 \end{pmatrix} \in \mathbf{R}^{m \times m}$$

is a tridiagonal matrix. Right hand side b is all one vector and the initial approximation vector is zero vector for all the algorithms. When $l = r = 1$, the coefficient matrix A is symmetric, which is the finite difference discretization matrix from the Poisson equation. While when $l \neq r$, the coefficient matrix A is nonsymmetric, which is the finite difference discretization matrix from the convection diffusion equation.

For the symmetric case, we compare the Jacobi, SOR, CG, Jacobi with Anderson acceleration, and SOR with Anderson acceleration methods from the aspect of number of iteration steps (denoted by ‘IT’) and elapsed CPU time in seconds (denoted by ‘CPU’).

For the nonsymmetric case, we compare the Jacobi, SOR, GMRES, Jacobi with Anderson acceleration, and SOR with Anderson acceleration methods from the aspect of number of iteration steps (denoted by ‘IT’) and elapsed CPU time in seconds (denoted by ‘CPU’).

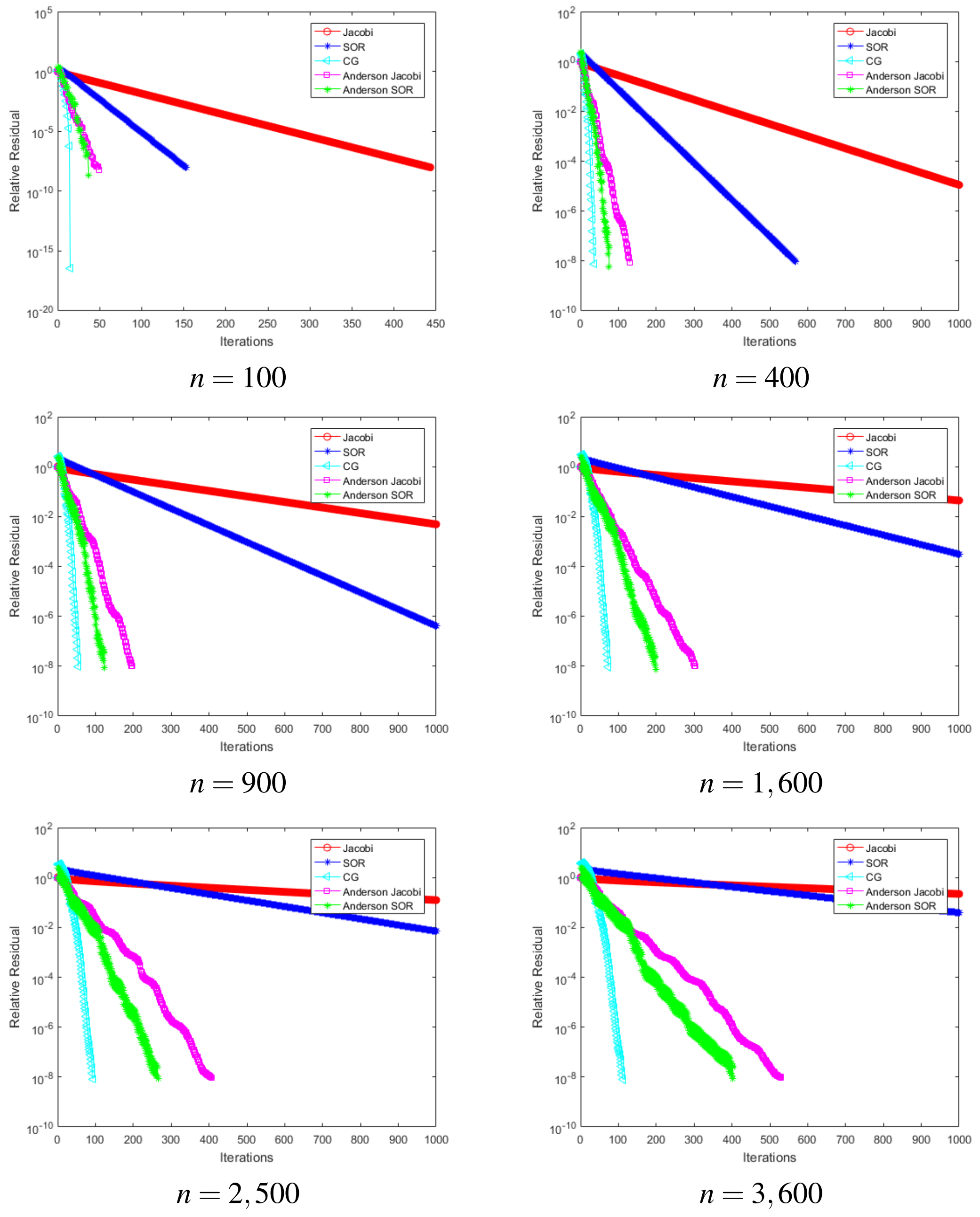


Figure 7.1: Relative residual versus iterations for symmetric problem (Run Anderson acceleration for every 3 steps).

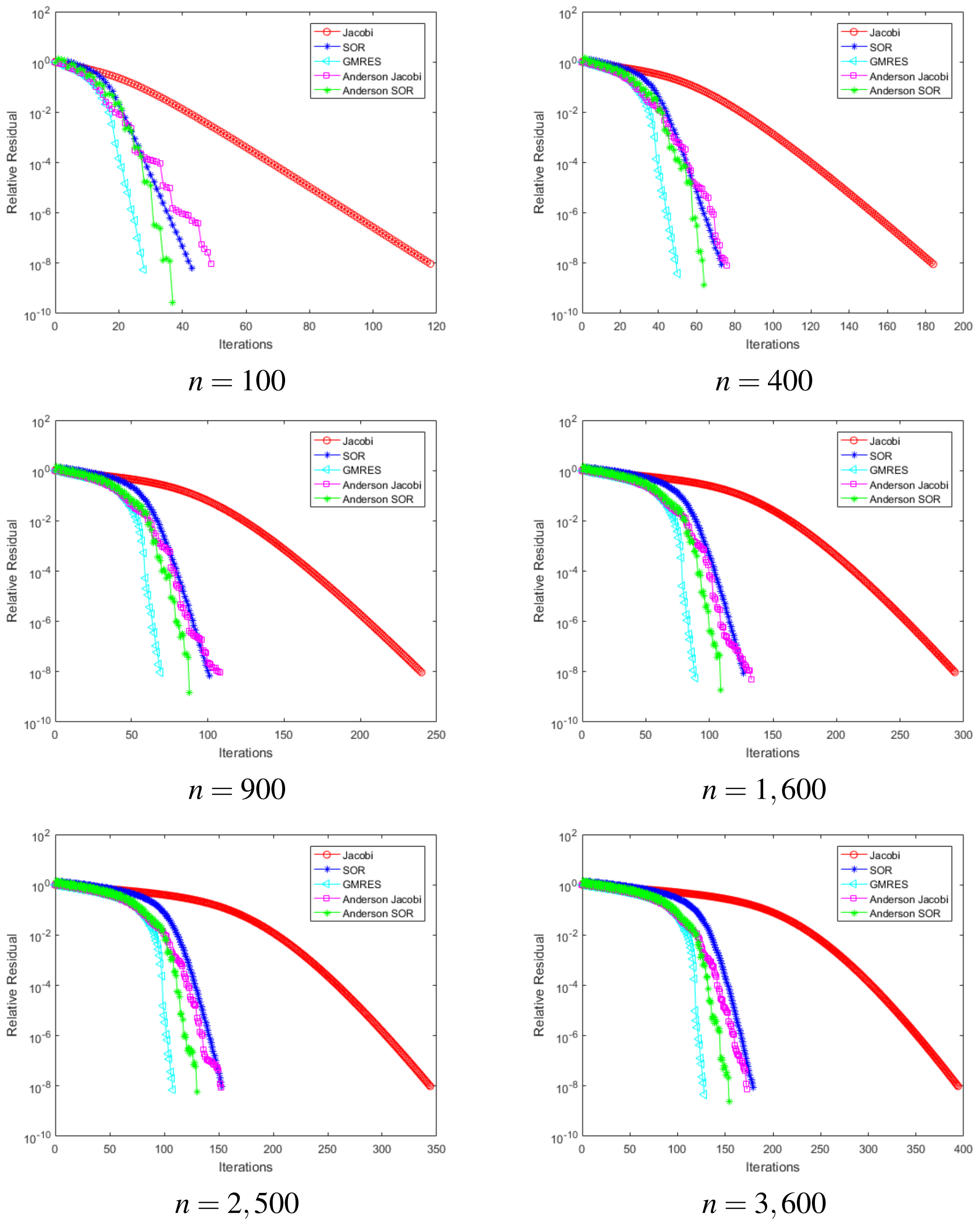
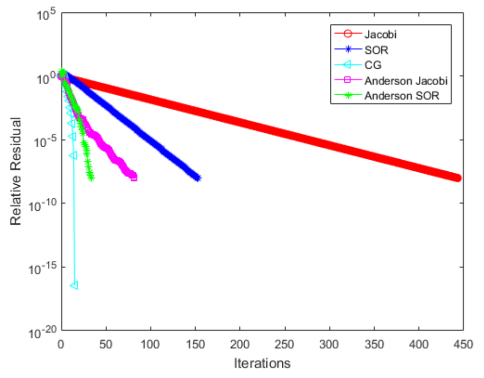
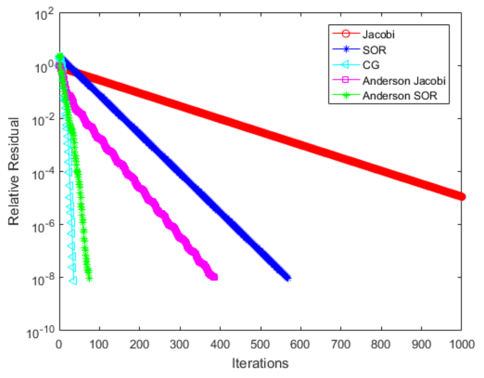


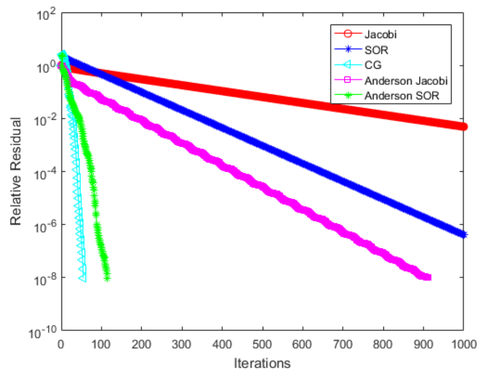
Figure 7.2: Relative residual versus iterations for nonsymmetric problem (Run Anderson acceleration for every 3 steps).



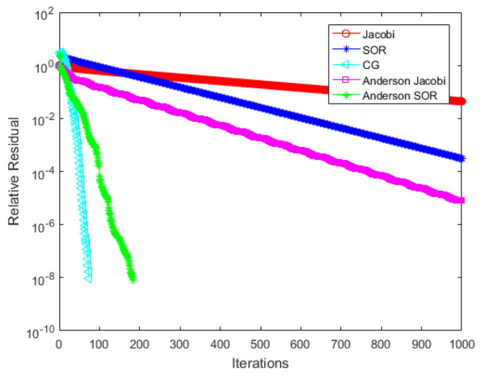
$n = 100$



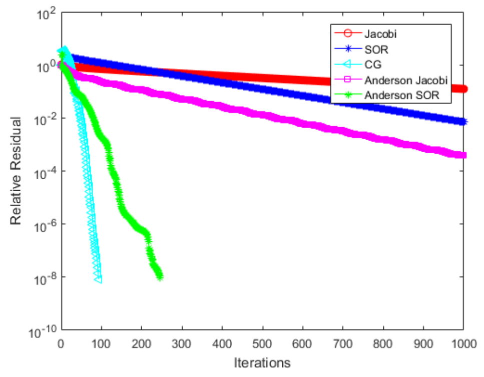
$n = 400$



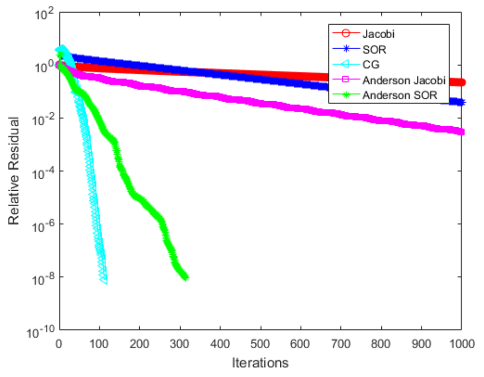
$n = 900$



$n = 1,600$

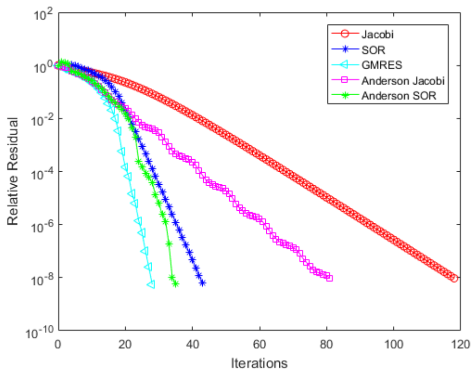


$n = 2,500$

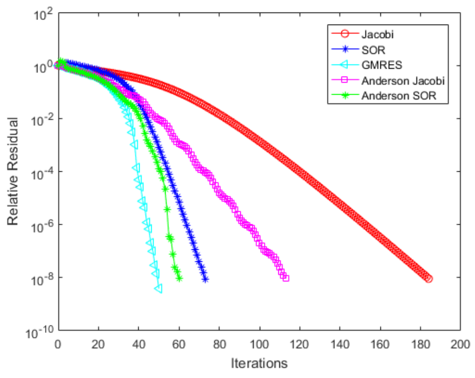


$n = 3,600$

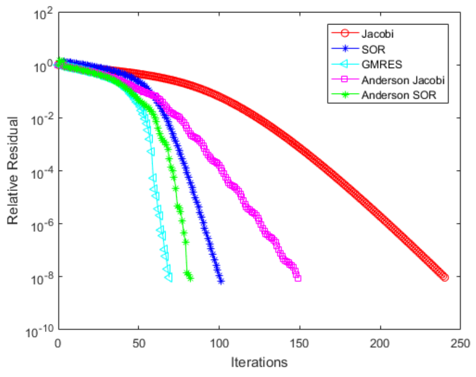
Figure 7.3: Relative residual versus iterations for symmetric problem (Run Anderson acceleration for every steps).



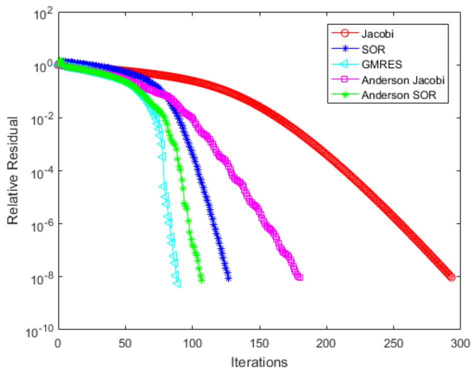
$n = 100$



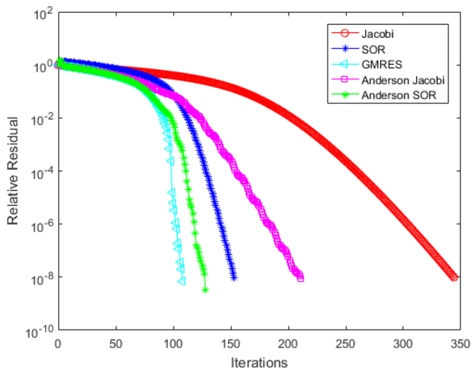
$n = 400$



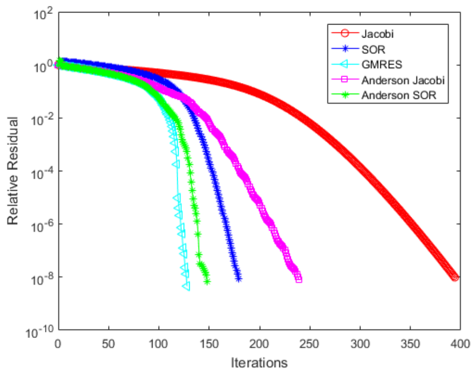
$n = 900$



$n = 1,600$



$n = 2,500$



$n = 3,600$

Figure 7.4: Relative residual versus iterations for nonsymmetric problem (Run Anderson acceleration for every steps).

7.3 Nonnegative Constrained Least Squares Problem

In this section, we apply the Anderson acceleration to the modulus-based matrix splitting iteration methods for linear complementarity problem (2.9)

$$x \geq \mathbf{0}, \quad \lambda = A^{\mathrm{T}}Ax - A^{\mathrm{T}}b \geq \mathbf{0}, \quad \text{and} \quad x^{\mathrm{T}}\lambda = 0,$$

which is equivalent to NNLS problem (1.1). Without loss of generality, we assume the standard LCP as LCP(A,q)

$$w := Az + q \geq 0, \quad z \geq 0 \quad \text{and} \quad z^{\mathrm{T}}w = 0. \tag{7.4}$$

Bai [4] established the following implicit fixed-point equation

$$(M\Gamma + \Omega_1)x = (N\Gamma - \Omega_2)x + (\Omega - A\Gamma)|x| - q \tag{7.5}$$

to construct modulus-based matrix splitting iteration method for solving the LCP(A,q), where $A = M - N$ is a splitting of the matrix $A \in \mathbf{R}^{n\times n}$, $\Omega = \Omega_1 + \Omega_2$ and Γ are $n \times n$ positive diagonal matrices. By setting $\Omega_1 = \Omega$, $\Omega_2 = 0$ and $\Gamma = (1/\gamma)I$, equation (7.5) yields a series of modulus-based matrix splitting iteration methods. For instance, when $M = D$ and $N = L + U$, it gives the modulus-based Jacobi (MJ) iteration method

$$(D + \Omega)x^{(k+1)} = (L + U)x^{(k)} + (\Omega - A)|x^{(k)}| - \gamma q;$$

when $M = (1/\omega)D - L$ and $N = (1/\omega - 1)D + U$, it gives the modulus-based successive overrelaxation (MSOR) iteration method

$$(D + \omega\Omega - \omega L)x^{(k+1)} = [(1-\omega)D + \omega U]x^{(k)} + \omega(\Omega - A)|x^{(k)}| - \omega\gamma q;$$

and when $\alpha = 1$, it gives the modulus-based Gauss-Seidel (MGS) iteration method. Therefore, the modulus-based Jacobi iteration with Anderson acceleration method and the modulus-based successive overrelaxation (MSOR) with Anderson acceleration method can be derived as follows.

Algorithm 7.3.1. Modulus-Based Jacobi Iteration with Anderson Acceleration

1. Choose the initial approximation z^0.

```
2.    For $k = 0, 1, 2, \ldots$ until convergence
3.        Generate sequence through $z^{k+1} = g_{\mathrm{mj}}(z^k)$
4.        Apply Anderson acceleration Algorithm 7.1.2 for the sequence.
5.    Endfor
```

Algorithm 7.3.2. Modulus-Based SOR Iteration with Anderson Acceleration

```
1.    Choose the initial approximation $z^0$.
2.    For $k = 0, 1, 2, \ldots$ until convergence
3.        Generate sequence through $z^{k+1} = g_{\mathrm{msor}}(z^k)$
4.        Apply Anderson acceleration Algorithm 7.1.2 for the sequence.
5.    Endfor
```

Next, we construct numerical experiments for the system of linear equations, to test the performance of Algorithms 7.3.1 and 7.3.2. The linear systems LCP(A, q) are constructed similar to the Section 7.2.

7.4 Nonnegative Matrix Factorization

In this section, we apply the Anderson acceleration to the alternating least squares methods for NMF.

First, we rewrite the multiplicative update Algorithm 6.2.2 as the fixed-point iteration scheme.

Algorithm 7.4.1. Multiplicative Update Fixed-Point Iteration

```
1.    Choose initial matrices $H^0$ and $W^0$.
2.    For $k = 0, 1, 2, \ldots$ until convergence
3.        $[H^{k+1}, W^{k+1}] = g_{mu}([H^k, W^k])$
4.    Endfor
```

Note that the input and output of multiplicative update are matrices. In order to utilize Anderson acceleration for the multiplicative update, we need to vectorize the matrix by stacking the columns of a matrix one on top of the other. We denote by vec the operation of vectorization and unvec the inverse operation to vec.

Methods	Iterations	$f(W,H)$	CPU
MU	88	50346.85	14.28
MU-AA	65	48576.58	14.03
PG	17	45372.30	92.58
Mod	19	45900.87	21.83

Algorithm 7.4.2. Multiplicative Update Fixed-Point Iteration

```
1.    Choose initial matrices $H^0$, $W^0$ and $Z^0 = [H^0, W^0]$.
2.    For $k = 0, 1, 2, \dots$ until convergence
3.        Set $z^k = vec(Z^k)$
4.        $z^{k+1} = g_{mu}(z^k)$
5.        $[H^{k+1}, W^{k+1}] = Z^{k+1} = unvec(z^{k+1})$
6.    Endfor
```

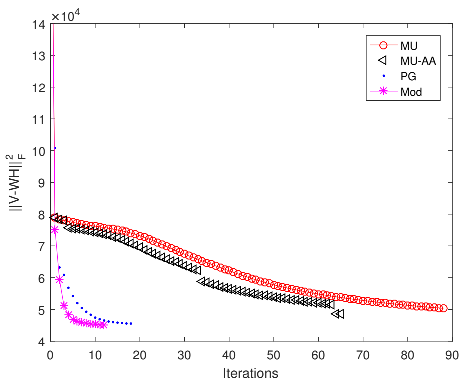
Then, the Anderson acceleration can be applied for Algorithm 7.4.3.

Algorithm 7.4.3. Multiplicative Update Fixed-Point Iteration

```
1.    Choose initial matrices $H^0$, $W^0$ and $Z^0 = [H^0, W^0]$.
2.    For $k = 0, 1, 2, \dots$ until convergence
3.        Set $z^k = vec(Z^k)$
4.        $z^{k+1} = g_{mu}(z^k)$
5.        Apply Anderson acceleration Algorithm 7.1.2 for the sequence.
6.        $[H^{k+1}, W^{k+1}] = Z^{k+1} = unvec(z^{k+1})$
7.    Endfor
```

7.5 Concluding Remarks

We can reach the conclusion that the Anderson extrapolation not only can accelerate the linear fixed-point iteration, but also accelerate the modulus-type and matrix-based nonlinear fixed-point iteration. How to further exploit the optimal acceleration technique by controlling the times and the number of interval vectors will be the ongoing research work.



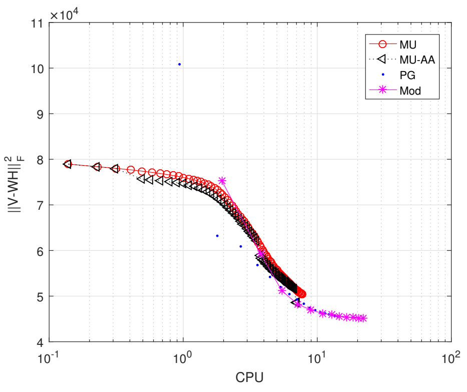


Figure 7.5: Objective function value versus iterations (left) and CPU time in seconds (right), respectively, for random problem.

CHAPTER 8

CONCLUDING REMARKS

We give the concluding remarks for the overall thesis as follows.

1. A new class of inner outer iterative methods for nonnegative constrained least squares (NNLS) problem (1.1) was proposed based on the modulus transformation for the nonnegative variables. Thus, the solution of the NNLS problem (1.1) can be transformed into the solution of a sequence of unconstrained least squares problems. Theoretical convergence analysis was presented when the inner system is solved either exactly or iteratively, and the choice of the parameter matrix was discussed for the proposed methods. Moreover, we proposed a two-stage hybrid modulus algorithm by incorporating the active set strategy, which contains two stages where the first stage consists of modulus iterations to identify the active set, while the second stage solves the reduced unconstrained least squares problems only on the inactive variables, and projects the solution into the nonnegative region. Numerical experiments show the efficiency of the proposed modulus methods compared to projection gradient-type methods with less iteration steps and CPU time for full column rank and rank deficient overdetermined NNLS problems. The modulus method is not only more efficient for identifying a suitable active set, but also outperforms projection gradient-type methods with less iteration steps and CPU time when the coefficient matrix has ill-determined rank with large condition number and the singular values cluster near zero. We also applied our modulus

methods to nonnegative constrained ill-posed image restoration problems, and the numerical results showed that the proposed method gives more accurate results compared to the projected gradient type methods.

2. We consider the solution of large sparse BLS problems using a new class of iterative methods based on modulus transformation, which converts the solution of the BLS into a sequence of the unconstrained least squares problems. The efficient Krylov subspace methods with suitable preconditioners are applied to solve the inner unconstrained least squares problems for each outer iteration. We also discuss the solution of saddle point inner systems, and the choice of the parameter matrix. Numerical experiments show the efficiency of the proposed methods in comparison of the gradient projection methods.

3. We applies modulus-based iterative methods to nonnegative Tikhonov regularization. The discrepancy principle is used to determine the regularization parameter. Efficient solution methods are described. The given linear discrete ill-posed problem is reduced to a small problem by a Krylov subspace method, and then the reduced Tikhonov regularization problems so obtained is solved. Several numerical examples in one and two space-dimensions illustrate the efficacy of the proposed methods.

4. We consider a new alternating nonnegative least squares method using modulus-type inner outer iteration method for the nonnegative constrained least squares subproblem. Numerical experiments on the synthetic data and ORL face image data show that the proposed methods converges faster than the gradient descent methods.

5. Finally we attempt to accelerate the numerical methods that proposed in the previous chapters using Anderson acceleration. The accelerated version of the previous algorithms are proposed and analyzed. Numerical experiments show the efficiency of the Anderson acceleration for linear equations, nonlinear NNLS and NMF problems.

BIBLIOGRAPHY

[1] B. H. AHN, *Solutions of nonsymmetric linear complementarity problems by iterative methods*, Journal of Optimization Theory and Applications, 33 (1981), pp. 175–185.

[2] D. G. ANDERSON, *Iterative procedures for nonlinear integral equations*, J. Assoc. Comput. Mach., 12 (1965), pp. 547–560.

[3] AT&T Laboratories Cambridge ORL Database of Faces, Available at http://www.cl.cam.ac.uk/research/dtg/attarchive/facedatabase.html.

[4] Z.-Z. BAI, *Modulus-based matrix splitting iteration methods for linear complementarity problems*, Numer. Linear Algebra Appl., 6 (2010), pp. 917–933.

[5] D. P. BERTSEKAS, *On the Goldstein-Levitin-Polyak gradient projection method*, IEEE Trans. Automat. Control, 21 (1976), pp. 174–184.

[6] D. P. BERTSEKAS, *Projected Newton methods for optimization problems with simple constraints*, SIAM Journal on Control and Optimization, 20 (1982), pp. 221–246.

[7] D. P. BERTSEKAS, *Nonlinear Programming*, 2nd ed., Athena Scientific, Belmont, MA, 1999.

[8] Å. BJÖRCK, *Numerical Methods for Least Squares Problems*, SIAM, Philadelphia, 1996.

[9] C. G. BROYDEN, *A class of methods for solving nonlinear simultaneous equations*, Math. Comp., 19 (1965), pp. 577–593.

[10] M. W. BERRY, M. BROWNE, A. N. LANGVILLE, V. P. PAUCA, AND R. J. PLEMMONS, *Algorithms and applications for approximate nonnegative matrix factorization*, Computational statistics & data analysis, 52 (2007), pp. 155–173.

[11] R. BRO AND S. DE JONG, *A fast non-negativity-constrained least squares algorithm*, J. Chemometrics, 11 (1997), pp. 393–401.

[12] Z.-Z. BAI, G. H. GOLUB, AND M. K. NG, *Hermitian and skew-Hermitian splitting methods for non-Hermitian positive definite linear systems*, SIAM J. Matrix Anal. Appl., 24 (2003), pp. 603–626.

[13] C. BOUTSIDIS AND E. GALLOPOULOS, *SVD based initialization: A head start for nonnegative matrix factorization*, Pattern Recognition, 41 (2008), pp. 1350-ÍC1362.

[14] R. BRO AND S. D. JONG, *A fast non-negativity-constrained least squares algorithm*, Journal of Chemometrics, 11 (1997), pp. 393–401.

[15] M. H. VAN BENTHEM AND M. R. KEENAN, *Fast algorithm for the solution of large-scale non-negativity-constrained least squares problems*, Journal of Chemometrics, 18 (2004), pp. 441–450.

[16] ON THE IDENTIFICATION OF ACTIVE CONSTRAINTS, SIAM J. Numer. Anal., 25 (1988), pp. 1197–1211.

[17] S. BERISHA AND J. G. NAGY, *Iterative Methods for Image Restoration*, (2012), http://www.mathcs.emory.edu/~nagy/RestoreTools.

[18] J. BAGLAMA AND L. REICHEL, *Augmented implicitly restarted Lanczos bidiagonalization methods*, SIAM J. Sci. Comput., 27 (2005), pp. 19–42.

[19] M. Bierlaire, Ph. L. Toint, and D. Tuyttens, *On iterative algorithms for linear least squares problems with bound constraints*, Linear Algebra Appl. 143 (1991), pp. 111–143.

[20] J. M. Bardsley and C. R. Vogel, *A nonnegative constrained convex programming method for image reconstruction*, SIAM J Sci. Comput., 25 (2004), pp. 1326–1343.

[21] R. Cockett, *The block conjugate gradient for multiple right hand sides in a direct current resistivity inversion*. Available online at http://www.row1.ca/s/pdfs/courses/BlockCG.pdf.

[22] C. W. Cryer, *The solution of a quadratic programming using systematic overrelaxation*, SIAM J. Control 9 (1971), pp. 385–392 .

[23] M. T. Chu, F. Diele, R. J. Plemmons, and S. Ragni, *Optimality, computation and interpretation of nonnegative matrix factorizations*, (2005). Available online at http://www4.ncsu.edu/~mtchu/Research/Papers/nnmf.pdf.

[24] T. F. Coleman and Y. Li, *On the convergence of interior-reflective Newton methods for nonlinear minimization subject to bounds*, Mathematical programming, 67 (1994), pp. 189–224.

[25] T. F. Coleman and Y. Li, *A reflective Newton method for minimizing a quadratic function subject to bounds on some of the variables*, SIAM Journal on Optimization, 6 (1996), pp. 1040–1058.

[26] D. Calvetti, G. Landi, L. Reichel, and F. Sgallari, *Nonnegativity and iterative methods for ill-posed problems*, Inverse Problems, 20 (2004), pp. 1747–1758.

[27] P. H. Calamai and J. J. Moreé, *Projected gradient methods for linearly constrained problems*, Math. Programming, 39 (1987), pp. 93–116.

[28] R. W. Cottle, J.-S. Pang and R. E. Stone, *The linear complementarity problem*, Academic, SanDiego, 1992.

[29] A. CICHOCKI, R. ZDUNEK, A. H. PHAN, AND S. I. AMARI, *Nonnegative matrix and tensor factorizations: applications to exploratory multi-way data analysis and blind source separation*, John Wiley & Sons, 2009.

[30] T. DAVIS, *The University of Florida Sparse Matrix Collection*, available online at http://www.cise.ufl.edu/research/sparse/matrices.

[31] Z. DOSTÁL, *Box constrained quadratic programming with proportioning and projections*, SIAM J. Optim., 7 (1997), pp. 871–887.

[32] J. C. DUNN, *Global and asymptotic convergence rate estimates for a class of projected gradient processes*, SIAM J. Control Optim., 19 (1981), pp. 368–400.

[33] J.-L. DONG AND M.-Q. JIANG, *A modified modulus method for symmetric positive-definite linear complementarity problems*, Numer. Linear Algebra Appl., 16 (2009), pp. 129–143.

[34] M. Donatelli, C. Estatico, A. Martinelli, and S. Serra–Capizzano, Improved image deblurring with anti-reflective boundary conditions and re-blurring. Inverse Problems, 22 (2006), pp. 2035–2053.

[35] Z. DOSTÁL AND J. SCHÖBERL, *Minimizing quadratic functions subject to bound constraints with the rate of convergence and finite termination*, Computational Optimization and Applications, 30 (2005), pp. 23–43.

[36] R. S. DEMBO AND U. TULOWITZKI, *On the minimization of quadratic functions subject to box constraints*, Working paper 71, School of Organization and Management, Yale University, New Haven, CT, 1983.

[37] V. EYERT, *A comparative study on methods for convergence acceleration of iterative vector sequences*, J. Comput. Phys., 124 (1996), pp. 271–285.

[38] H. W. ENGL, M. HANKE AND A. NEUBAUER, *Regularization of Inverse Problems*, Kluwer, Dordrecht, 1996.

[39] C. Fenu, L. Reichel, and G. Rodriguez, GCV for Tikhonov regularization via global Golub–Kahan decomposition, Numer. Linear Algebra Appl., 23 (2016), pp. 467–484.

[40] H. R. Fang and Y. Saad, *Two classes of multisecant methods for nonlinear acceleration*, Numer. Linear Algebra Appl., 16 (2009), pp. 197–221.

[41] G. H. Golub and C. F. Van Loan, Matrix Computations, 4th ed., Johns Hopkins University Press, Baltimore, 2013.

[42] L. Grippo and M. Sciandrone, *On the convergence of the block nonlinear Gauss-Seidel method under convex constraints*, Oper. Res. Lett., 26 (2000), pp. 127–136.

[43] S. Gazzola and Y. Wiaux, *Fast nonnegative least squares through flexible Krylov subspaces*, arXiv preprint arXiv:1511.06269, (2015).

[44] G. H. Golub, X. Wu and J.-Y. Yuan, *SOR-like Methods for Augmented Systems*, BIT Numerical Mathematics, 41 (2001), pp. 71–85.

[45] C. W. Groetsch, The Theory of Tikhonov Regularization for Fredholm Equations of the First Kind, Pitman, Boston, 1984.

[46] A. Greenbaum, *Iterative Methods for Solving Linear Systems*, Frontiers Appl. Math. 17, SIAM, Philadelphia, 1997.

[47] E. F. Gonzalez and Y. Zhang, *Accelerating the Lee-Seung algorithm for non-negative matrix factorization*, Dept. Comput. Appl. Math., Rice Univ., Houston, TX, Tech. Rep. TR-05-02, 2005.

[48] P. C. Hansen, *Regularization tools: A Matlab package for analysis and solution of discrete ill-posed problems*, Numer. Algorithms 6 (1994), pp. 1–35.

[49] P. C. Hansen, J. G. Nagy and D. P. O'Leary, *Deblurring Images: Matrices, Spectra, and Filtering*, SIAM, Philadelphia, 2006.

[50] N. J. Higham and N. Strabić, *Anderson acceleration of the alternating projections method for computing the nearest correlation matrix*, Numerical Algorithms, 72 (2016), pp. 1021–1042.

[51] M. Hanke, J. G. Nagy and C. Vogel, *Quasi-Newton approach to nonnegative image restorations*, Linear Algebra Appl., 316 (2000), pp. 223–236.

[52] A. HADJIDIMOS AND M. TZOUMAS, *Nonstationary extrapolated modulus algorithms for the solution of the linear complementarity problem*, Linear Algebra Appl., 431 (2009), pp. 197–210.

[53] K. HAYAMI, J.-F. YIN, AND T. ITO, *GMRES methods for least squares problems*, SIAM J. Matrix Anal. Appl., 31 (2010), pp. 2400–2430.

[54] W. W. HAGER AND H. ZHANG, *A new active set algorithm for box constrained optimization*, SIAM J. Optim., 17 (2006), pp. 526–557.

[55] J. J. JÚDICE AND F. M. PIRES, *A block principal pivoting algorithm for large-scale strictly monotone linear complementarity problems*, Comput. Oper. Res., 21 (1994), pp. 587–596.

[56] L. KAUFMAN, *Maximum likelihood, least squares, and penalized least squares for PET*, IEEE Transactions on Medical Imaging, 12 (1993), pp. 200–214.

[57] A. KLARBRING, *Quadratic programs in frictionless contact problems*, Internat. J. Engrg. Sci., 24 (1986), pp. 1207–1217.

[58] H. KIM AND H. PARK, *Nonnegative matrix factorization based on alternating nonnegativity constrained least squares and active set method*, SIAM J. Matrix Anal. Appl., 30 (2008), pp. 713–730.

[59] J. KIM AND H. PARK, *Fast nonnegative matrix factorization: an active-set-like method and comparisons*, SIAM J. Sci. Comput., 33 (2011), pp. 3261–3281.

[60] N. W. KAPPEL AND L. T. WATSON, *Iterative algorithms for the linear complementarity problems*, Int. J. Comput. Math., 19 (1986), pp. 273–297.

[61] D. KIM, S. SRA AND I. S. DHILLON, *A new projected quasi-newton approach for solving nonnegative least squares problem*, Technical Report CS-TR-06-54, The University of Texas at Austin, 2007.

[62] M. KOČVARA AND J. ZOWE, *An iterative two-step algorithm for linear complementarity problems*, Numer. Math., 68 (1994), pp. 95–106.

[63] C.-J. LIN, *Projected gradient methods for non-negative matrix factorization*, Neural Computation, 19 (2007), pp. 2756–2779.

[64] C.-J. LIN, *On the convergence of multiplicative update algorithms for nonnegative matrix factorization*, IEEE Transactions on Neural Networks, 18 (2007), pp. 1589–1596.

[65] L. LIN, *Alternative gradient algorithms with applications to nonnegative matrix factorizations*, Applied Mathematics and Computation, 216 (2010), pp. 1763–1770.

[66] P. LÖTSTEDT, *Soving the minimal least squares problem subject to bounds on the variables*, BIT Numerical Mathematics, 24 (1984), pp. 206–224.

[67] C. L. LAWSON AND R. J. HANSON, *Solving Least Squares Problems*, Prentice-Hall, 1974.

[68] D. D. LEE AND H. S. SEUNG, *Learning the parts of objects by non-negative matrix factorization*, Nature, 401 (1999), pp. 788–791.

[69] D. D. LEE AND H. S. SEUNG, *Algorithms for non-negative matrix factorization*, Advances in Neural Information Processing Systems, 13 (2001), pp. 556–562.

[70] O. MANGASARIAN, *Solutions of symmetric linear complementarity problems by iterative methods*, Journal of Optimization Theory and Applications, 22 (1977), pp. 465–485.

[71] K. MURTY, *Linear Complementarity, Linear and Nonlinear Programming*, Heldermann, Berlin, 1988.

[72] K. MORIKUNI AND K. HAYAMI, *Inner-iteration Krylov subspace methods for least squares problems*, SIAM J. Matrix Anal. Appl., 34 (2013), pp. 1–22.

[73] S. MORIGI, R. PLEMMONS, L. REICHEL, AND F. SGALLARI, *A hybrid multilevel-active set method for large box-constrained linear discrete ill-posed problems*, Calcolo, 48 (2011), pp. 89–105.

[74] S. MORIGI, L. REICHEL, F. SGALLARI, AND F. ZAMA, *An iterative method for linear discrete ill-posed problems with box constraints*, J. Comput. Appl. Math., 198 (2007), pp. 505–520.

[75] S. MORIGI, L. REICHEL AND F. SGALLARI, *An interior-point method for large constrained discrete ill-posed problems*, J. Comput. Appl. Math., 233 (2010), pp. 1288–1297.

[76] J. J. MORÉ AND G. TORALDO, *Algorithms for bound constrained quadratic programming problems*, Numer. Math., 55 (1989), pp. 377–400.

[77] J. J. MORÉ AND G. TORALDO, *On the solution of large quadratic programming problems with bound constraints*, SIAM J. Optimization, 1 (1991), pp. 93–113.

[78] J. NAGY AND Z. STRAKOŠ, *Enforcing nonnegativity in image reconstruction algorithms*, in Mathematical Modeling, Estimation and Imaging, ed. D. C. Wilson et al., of the Society of Photo-Optical Instrumentation Engineers (SPIE), Vol. 4121, The International Society for Optical Engineering, Bellingham, WA, 2000, pp. 182–190.

[79] J. NOCEDAL AND S. J. WRIGHT, *Numerical Optimization*, 2nd ed., Springer, New York, 2006.

[80] M. K. Ng, R. H. Chan, W. C. Tang, A fast algorithm for deblurring models with Neumann boundary conditions, SIAM J. Sci. Comp., 21 (1999), pp. 851–866.

[81] D. P. O'LEARY, *A generalized conjugate gradient algorithm for solving a class of quadratic programming problems*, Linear Algebra and its Applications, 34 (1980), pp. 371–399.

[82] J. ORTEGA AND W. RHEINBOLDT, *Iterative Solution of Nonlinear Equations in Several Variables*, Academic Press: New York, 1970.

[83] E. ONUNWOR AND L. REICHEL, *On the computation of a truncated SVD of a large linear discrete ill-posed problem*, submitted for publication.

[84] J.-S. PANG, *Necessary and sufficient conditions for the convergence of iterative methods for the linear complementarity problem*, Journal of Optimization Theory and Applications, 42 (1984), pp. 1–17

[85] B. T. POLYAK, *The conjugate gradient method in extremal problems*, U.S.S.R. Computational Mathematics and Mathematical Physics, 9 (1969), pp. 94–112.

[86] P. PULAY, *Convergence acceleration of iterative sequences. The case of SCF iteration*, Chem. Phys. Lett., 73 (1980), pp. 393–398.

[87] P. P. PRATAPA, P. SURYANARAYANA AND J. E. PASK, *Anderson accelration of the Jacobi iterative method: an efficient alternative to Krylov methods for large, sparse linear systems*, J. Comput. Phys., 306 (2016), pp. 43–54.

[88] P. PAATERO AND U. TAPPER, *Positive matrix factorization: A non-negative factor model with optimal utilization of error estimates of data values*, Environmetrics, 5 (1994), pp. 111–126

[89] M. ROJAS AND T. STEIHAUG, *An interior-point trust-region-based method for large-scale non-negative regularization*, Inverse Problems, 18 (2002), pp. 1291–1307.

[90] T. ROHWEDDER AND R. SCHNEIDER, *An analysis for the DIIS acceleration method used in quantum chemistry calculations*, J. Math. Chem., 49 (2011), pp. 1889–1914.

[91] Y. SAAD, *Iterative Methods for Sparse Linear Systems*, 2nd ed., SIAM, Philadelphia, 2003.

[92] Y. SAAD AND M. H. SCHULTZ, *GMRES: a generalized minimal residual algorithm for solving nonsymmetric linear systems*, SIAM J. Sci. Statist. Comput., 7 (1986), pp. 856–869.

[93] A. TOTH AND C. T. KELLEY, *Convergence analysis for Anderson acceleration*, SIAM J. Numer. Anal., 53 (2015), pp. 805–819.

[94] R. A. Tapia, Y. Zhang, M. J. Saltzman and A. Weiser, *The Mehrotra predictor-corrector interior-point method as a perturbed composite Newton method*, SIAM Journal on Optimization, 6 (1996), pp. 47–56.

[95] W. M. G. van Bokhoven, *A Class of Linear Complementarity Problems is Solvable in Polynomial Time*, unpublished paper, Dept. of Electrical Engineering, University of Technology, The Netherlands (1980).

[96] W. M. G. van Bokhoven, *Piecewise-linear modelling and analysis*, Proeschrift, Eindhoven (1981).

[97] M. H. van Benthem and M. R. Keenan, *Fast algorithm for the solution of large-scale non-negativity-constrained least squares problems*, J. Chemometrics, 18 (2004), pp. 441–450.

[98] S. J. Wright, *Implementing proximal point methods for linear programming*, Report MCS-P45-0189, Algonne National Laboratory, Algonne, IL, 1989.

[99] S. Wild, J. Curry, and A. Dougherty, *Improving non-negative matrix factorizations through structured intitialization*, Pattern Recognition, 37 (2004), pp. 2217–2232.

[100] H. F. Walker and P. Ni, *Anderson acceleration for fixed-point iterations*, SIAM J. Numer. Anal., 49 (2011), pp. 1715–1735.

[101] E. K. Yang and J. W. Tolle, *A class of methods for solving large convex quadratic programs subject to box constraints*, preprint, Department of Operations Research, University of North Carolina, Chapel Hill, NC, 1988.

[102] C. Zhang, L. Jing and N. Xiu, *A new active set method for nonnegative matrix factorization*, SIAM J. Sci. Comput., 36 (2014), pp. A2633–A2653.

[103] N. Zheng, K. Hayami and J.-F. Yin, *Modulus-type inner outer iteration methods for nonnegative constrained least squares problems*, SIAM J. Matrix Anal. Appl., 37 (2016), pp. 1250–1278.

[104] N. ZHENG AND J.-F. YIN, *Accelerated modulus-based matrix splitting iteration methods for linear complementarity problems*, Numer. Algorithms 64 (2013), pp. 245–262.

[105] N. ZHENG AND J.-F. YIN, *Convergence of accelerated modulus-based matrix splitting iteration methods for linear complementarity problem with an H_+-matrix*, J. Comput. Appl. Math. 260 (2014), pp. 281–293.