ISSN 1346-5597

National Institute of Informatics

NII Technical Report

Modulus-Type Inner Outer Iteration Methods for Nonnegative Constrained Least Squares Problems (Revised Version)

Ning Zheng, Ken Hayami, and Jun-Feng Yin

NII-2016-001E
Jan. 2016

URL: http://www.nii.ac.jp/ Tokyo 101-8430, JAPAN +81-3-4212-2000 FAX: +81-3-3556-1916

MODULUS-TYPE INNER OUTER ITERATION METHODS FOR NONNEGATIVE CONSTRAINED LEAST SQUARES PROBLEMS*

NING ZHENG†, KEN HAYAMI‡, AND JUN-FENG YIN§

Abstract. For the solution of large sparse nonnegative constrained 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 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 methods compared to projection gradient-type methods with less iteration steps and CPU time.

Key words. least squares problems, nonnegative constraints, convergence, modulus method, active set method, CGLS method

AMS subject classifications. 65F10, 65F20, 65F22, 65F50

1. Introduction.

Consider the nonnegative constrained linear least squares problem [3], abbreviated as NNLS,

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

where $A\in\mathbf{R}^{m\times n}$, $\boldsymbol{b}\in\mathbf{R}^{m\times 1}$, $m\geq n$ or $m<n$, 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 problems arise in many scientific computing and engineering applications [3], e.g., image restoration [26], reconstruction problems in geodesy [5] and tomography, contact problems for mechanical systems [17], 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 [5].

Algorithms for the solution of the unconstrained linear least squares problem

$$\min_{\boldsymbol{x}\in\mathbf{R}^n} \|A\boldsymbol{x}-\boldsymbol{b}\|_2 \tag{1.2}$$

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)

*This work was supported by the Grant-in-Aid for Scientific Research (C) of the Ministry of Education, Culture, Sports, Science and Technology, Japan, and the National Natural Science Foundation of China (No. 11271289).

†Department of Informatics, School of Multidisciplinary Sciences, SOKENDAI (The Graduate University for Advanced Studies), 2-1-2 Hitotsubashi, Chiyoda-ku, Tokyo, 101-8430, Japan/Department of Mathematics, Tongji University, 1239 Siping Road, Shanghai, 200092, People's Republic of China (nzheng@nii.ac.jp).

‡National Institute of Informatics/Department of Informatics, School of Multidisciplinary Sciences, SOKENDAI (The Graduate University for Advanced Studies), 2-1-2 Hitotsubashi, Chiyoda-ku, Tokyo, 101-8430, Japan (hayami@nii.ac.jp).

§Department of Mathematics, Tongji University, 1239 Siping Road, Shanghai, 200092, People's Republic of China (yinjf@tongji.edu.cn).

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

$$A^{\mathrm{T}}A\boldsymbol{x} = A^{\mathrm{T}}\boldsymbol{b}, \tag{1.3}$$

and (preconditioned) GMRES method by Hayami, Yin and Morikuni [16, 21] 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.

One of the most popular techniques is the projection operation

$$P(\boldsymbol{x}) = \max(\boldsymbol{x}, \boldsymbol{0}), \tag{1.4}$$

where $\max(\boldsymbol{x}, \boldsymbol{0})$ is the vector whose ith component is the maximum of x_i and 0. Projection is widely used for the solution of the NNLS problem (1.1). For example, by iteratively solving the fixed-point equation

$$\boldsymbol{x} = P(\boldsymbol{x} - \alpha\nabla l(\boldsymbol{x})), \quad l(\boldsymbol{x}) \equiv \frac{1}{2}\min_{\boldsymbol{x}\in\mathbf{R}^n} \|A\boldsymbol{x} - \boldsymbol{b}\|_2^2, \tag{1.5}$$

which is equivalent to the NNLS problem (1.1), the projected gradient method was proposed by Bertsekas [2], and was applied to quadratic programming with box constraints in [9, 24, 25]. Another projection-based method is the projected SOR method proposed by Cryer [7], and we can easily extend it to be the projected NR-SOR method, where NR-SOR method is SOR iteration for the normal equation (1.3). Below, we give their corresponding versions for the NNLS problem. For the projected NR-SOR method, we only list one forward sweep.

ALGORITHM 1.1. **Projected Gradient Method**

1. *Choose an initial approximate solution* $\boldsymbol{x}^0$ *and compute* $\boldsymbol{r}^0 = \boldsymbol{b} - A\boldsymbol{x}^0$.
2. *For* $k = 0, 1, 2, \ldots$ *until convergence*
3. *Compute* $\boldsymbol{s}^k = A^{\mathrm{T}}\boldsymbol{r}^k$.
4. *Compute* $\alpha_k = \|\boldsymbol{s}^k\|_2^2/\|A\boldsymbol{s}^k\|_2^2$.
5. *Find the smallest integer* $m \geq 0$ *that satisfies the sufficient decrease condition*

$$\|\boldsymbol{b} - A\boldsymbol{x}^{k+1}\|_2^2 \leq \|\boldsymbol{b} - A\boldsymbol{x}^k\|_2^2 - 2\mu(\boldsymbol{s}^k)^{\mathrm{T}}(\boldsymbol{x}^{k+1} - \boldsymbol{x}^k),$$

where $0 < \beta < 1$, $0 \leq \mu < 1$ *and*

$$\boldsymbol{x}^{k+1} = P(\boldsymbol{x}^k + \beta^m\alpha_k\boldsymbol{s}^k).$$

6. *Compute* $\boldsymbol{r}^{k+1} = \boldsymbol{b} - A\boldsymbol{x}^{k+1}$.
7. *Endfor*

ALGORITHM 1.2. **Projected NR-SOR with One Forward Sweep**

1. *Given* $\boldsymbol{x}^k$ *and* $\boldsymbol{r}^k (= \boldsymbol{b} - A\boldsymbol{x}^k)$, $k \geq 0$.
2. *For* $i = 1, 2, \ldots, n$ *Do*
3. *Compute* $\delta_i = \omega(\boldsymbol{r}^k, A\boldsymbol{e}_i)/\|A\boldsymbol{e}_i\|_2^2$, *where* $\boldsymbol{e}_i$ *is the* i*th 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* $\boldsymbol{r}^k = \boldsymbol{r}^k - (x_i^{k+1} - x_i^k)(A\boldsymbol{e}_i)$.

6. *Endfor*
7. *Set* $\boldsymbol{r}^{k+1} = \boldsymbol{r}^k$.

In addition, a class of inner outer iterative methods is widely discussed for the solution of NNLS 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. For example, by restricting the step size in each CG iteration to satisfy constraints, Polyak [29] and O'Leary [27] proposed a generalized CG method for solving general box constrained quadratic programming problems with a symmetric positive definite matrix, which can be naturally applied to solving NNLS problems. Similar algorithm called the restricted LSQR method was presented by Lötstedt [20], where LSQR is a stabilized version of CGLS, and Bierlaire, Toint and Tuyttens [5] introduced a variant of the algorithm. Moreover, instead of shrinking the step size, some researchers considered projection-type methods, which orthogonally project the iterated solution into the feasible region by (1.4). For example, Bierlaire et al. [5] proposed projected gradient methods with active set strategy. 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 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 an inefficient feature when the number of variables is large.

In order to avoid these disadvantages, Dembo and Tulowitzki [12] proposed algorithms that allow to add or delete many indices at each iteration. Similar algorithms were presented by Yang and Tolle [31], in which they showed theoretically that the iterations converge to the solution in a finite number of steps. For the solution of nonnegative constrained ill-posed problem, which is equivalent to solving the NNLS problem with regularization, Calvetti et al. [8] proposed a projected iteration method by allowing more consecutive iterations in the inner iteration. In addition, Morigi et al. [23] 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 requirement 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 [22, 23].

Wright [30] 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 [24, 25] 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. Moreover, Bardsley and Vogel [6] extended the algorithm of Moré and Toraldo [25] 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.

In this paper, 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 (1.1) can be replaced by the solution of a sequence of unconstrained least squares problems, for which numerous efficient solvers can be exploited. Therefore, a new class of inner outer iterative methods is proposed by using CGLS method for inner iterations and the modulus-based iterative method in the outer iterations for the solution of LCP (linear complementarity problem) resulting from the KKT (Karush-Kuhn-Tucker) conditions of the NNLS. 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. Numerical experiments show the efficiency of the proposed methods compared to projection-type methods with less iteration steps and CPU time. Moreover, the modulus method is more efficient for identifying a suitable active set compared to projected gradient methods.

The rest of the paper is organized as follows. In Section 2, the modulus inner outer iteration method is proposed for the solution of the NNLS problem. In Section 3, convergence analysis of the proposed method is presented, and the choice of the parameter matrix is discussed. In Section 4, a hybrid method is proposed by alternately performing the modulus iterations and the active set CGLS iterations. In Section 5, numerical results are presented, and Section 6 concludes the paper.

2. 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 2.1. *The nonnegative constrained quadratic programming problem* NNQP(B, c)

$$\min_{\boldsymbol{x}\in\mathbf{R}^n}\left(\frac{1}{2}\boldsymbol{x}^T B\boldsymbol{x}+\boldsymbol{c}^T\boldsymbol{x}\right), \quad \textit{subject to} \quad \boldsymbol{x}\geq\mathbf{0} \tag{2.1}$$

is equivalent to the linear complementarity problem LCP$(B, \boldsymbol{c})$

$$\boldsymbol{x}\geq\mathbf{0}, \quad B\boldsymbol{x}+\boldsymbol{c}\geq\mathbf{0}, \quad \textit{and} \quad \boldsymbol{x}^T(B\boldsymbol{x}+\boldsymbol{c})=0, \tag{2.2}$$

provided that B is a symmetric positive semidefinite matrix.

Proof. See Appendix A. ■

COROLLARY 2.2. *If matrix B is symmetric positive definite, then both* NNQP$(B, \boldsymbol{c})$ *and* LCP$(B, \boldsymbol{c})$ *have the same unique solution.*

COROLLARY 2.3. *([3]) The NNLS problem* (1.1) *is equivalent to* LCP$(A^TA, -A^T\boldsymbol{b})$

$$\boldsymbol{x} \geq \mathbf{0}, \quad \boldsymbol{\lambda} \equiv A^T A\boldsymbol{x} - A^T\boldsymbol{b} \geq \mathbf{0}, \quad \textit{and} \quad \boldsymbol{x}^T\boldsymbol{\lambda} = 0. \tag{2.3}$$

Proof. Set $B = A^{\mathrm{T}}A$ and $\boldsymbol{c} = -A^{\mathrm{T}}\boldsymbol{b}$ in Theorem 2.1. ■

Furthermore, the following theorem, where the proof can be easily obtained by Theorem 2.1 in [1], implies that $\mathrm{LCP}(A^{\mathrm{T}}A, -A^{\mathrm{T}}\boldsymbol{b})$ is equivalent to the implicit fixed-point equation

$$(\Omega + A^{\mathrm{T}}A)\boldsymbol{z} = (\Omega - A^{\mathrm{T}}A)|\boldsymbol{z}| + A^{\mathrm{T}}\boldsymbol{b} \tag{2.4}$$

with modulus transformation $\boldsymbol{x} = \boldsymbol{z} + |\boldsymbol{z}|$, where Ω is a positive diagonal parameter matrix. Hence, it is equivalent to solve the implicit fixed-point equation (2.4) for the solution of (1.1) by Corollary 2.3.

THEOREM 2.4. *Let Ω be an $n\times n$ positive diagonal matrix. For the* $\mathrm{LCP}(A^TA, -A^T\boldsymbol{b})$, *the following statements hold:*

(i) *if $\boldsymbol{x}$ is a solution of the* $\mathrm{LCP}(A^TA, -A^T\boldsymbol{b})$, *then $\boldsymbol{z} = (\boldsymbol{x} - \Omega^{-1}\boldsymbol{\lambda})/2$ satisfies the implicit fixed-point equation* (2.4), *where $\boldsymbol{\lambda} = A^TA\boldsymbol{x} - A^T\boldsymbol{b}$;*

(ii) *if $\boldsymbol{z}$ satisfies the implicit fixed-point equation* (2.4), *then $\boldsymbol{x} = |\boldsymbol{z}| + \boldsymbol{z}$ is a solution of the* $\mathrm{LCP}(A^TA, -A^T\boldsymbol{b})$. *Moreover, $\boldsymbol{\lambda} = \Omega(|\boldsymbol{z}| - \boldsymbol{z})$ holds.*

Based on the equivalence in Theorem 2.4, the modulus-type iterative scheme

$$(\Omega + A^{\mathrm{T}}A)\boldsymbol{z}^{k+1} = (\Omega - A^{\mathrm{T}}A)|\boldsymbol{z}^k| + A^{\mathrm{T}}\boldsymbol{b} \tag{2.5}$$

is naturally derived for the solution of the fixed-point equation (2.4). If $\boldsymbol{z}^*$ is a fixed point of (2.5), then by Theorems 2.1 and 2.4, the solution of the NNLS problem (1.1) can be obtained straightforwardly by $\boldsymbol{x}^* = \boldsymbol{z}^* + |\boldsymbol{z}^*|$. Therefore, the solution of the NNLS problem (1.1) is transformed to the solution of a series of fixed-point equations (2.5), 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 2.1. **Modulus Iteration Method**

1. *Choose an initial approximate solution $\boldsymbol{z}^0$ and a parameter matrix Ω.*
2. *Compute $\boldsymbol{x}^0 = \boldsymbol{z}^0 + |\boldsymbol{z}^0|$.*
3. *For $k = 0, 1, 2, \ldots$ until convergence*
4. *Compute $\boldsymbol{z}^{k+1}$ by solving equation* (2.5).
5. *Compute $\boldsymbol{x}^{k+1} = \boldsymbol{z}^{k+1} + |\boldsymbol{z}^{k+1}|$.*
6. *Endfor*

We remark that this modulus method derived from (2.4) is a special case of modulus-based matrix splitting methods with $M = A^{\mathrm{T}}A$ and $N = \mathbf{0}$ in [1]. For more numerical methods for LCP, see [32] and the references therein.

Another remark is that a similar idea that transforms the constrained minimization problem into an unconstrained problem using the parameterization $\boldsymbol{x} = e^{\boldsymbol{z}}$ instead of $\boldsymbol{x} = \boldsymbol{z} + |\boldsymbol{z}|$ is proposed by Hanke, Nagy and Vogel [15], and is applied for image reconstruction problems in [26]. The main difference is the iterative methods constructed in [26] are based on the solution of fixed-point equation with respect to $\boldsymbol{x}$, 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 $\boldsymbol{z}$.

Finally, it is noted that the iterative scheme (2.5) can be reorganized as the normal equations

$$\widetilde{A}^{\mathrm{T}}\widetilde{A}\boldsymbol{z}^{k+1} = \widetilde{A}^{\mathrm{T}}\tilde{\boldsymbol{b}}^{k}, \tag{2.6}$$

of the unconstrained least squares problem

$$\min_{\boldsymbol{z}^{k+1}\in\mathbf{R}^n} \|\widetilde{A}\boldsymbol{z}^{k+1} - \tilde{\boldsymbol{b}}^{k}\|_2 \tag{2.7}$$

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

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

Therefore, the solution of the NNLS problem (1.1) is transformed to the solution of a series of unconstrained least squares problems (2.7). 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 2.2. **Modulus-Type Inner Outer Iteration Method**

1. *Choose an initial approximate solution* $\boldsymbol{z}^0$ *and a parameter matrix* Ω.
2. *Compute* $\boldsymbol{x}^0 = \boldsymbol{z}^0 + |\boldsymbol{z}^0|$ *and* $\boldsymbol{r}^0 = \boldsymbol{b} - A\boldsymbol{x}^0$.
3. *Set*

$$\widetilde{A} = \begin{bmatrix} A \\ \Omega^{\frac{1}{2}} \end{bmatrix} \quad \textit{and} \quad \tilde{\boldsymbol{r}}^0 = \begin{bmatrix} \boldsymbol{r}^0 \\ \Omega^{\frac{1}{2}}(|\boldsymbol{z}^0| - \boldsymbol{z}^0) \end{bmatrix}$$

4. *For* $k = 0, 1, 2, \ldots$ *until convergence*
5. *Compute an approximate solution* $\boldsymbol{w}^{k+1}$ *by solving*

$$\min_{\boldsymbol{w}\in\mathbf{R}^n} \|\widetilde{A}\boldsymbol{w} - \tilde{\boldsymbol{r}}^{k}\|_2. \tag{2.8}$$

6. *Compute* $\boldsymbol{z}^{k+1} = \boldsymbol{z}^{k} + \boldsymbol{w}^{k+1}$.
7. *Compute* $\boldsymbol{x}^{k+1} = \boldsymbol{z}^{k+1} + |\boldsymbol{z}^{k+1}|$ *and* $\boldsymbol{r}^{k+1} = \boldsymbol{b} - A\boldsymbol{x}^{k+1}$.
8. *Set*

$$\tilde{\boldsymbol{r}}^{k+1} = \begin{bmatrix} \boldsymbol{r}^{k+1} \\ \Omega^{\frac{1}{2}}(|\boldsymbol{z}^{k+1}| - \boldsymbol{z}^{k+1}) \end{bmatrix}$$

9. *Endfor*

Here, the iterative solution of the unconstrained least squares problems (2.8) for each $k = 0, 1, 2, ...$ is referred to as the inner iteration of the algorithm, while the for loop is referred to as the outer iteration. Note that Algorithms 2.1 and 2.2 are mathematically equivalent, since the solution of the inner unconstrained least squares problems (2.8) is equivalent to the solution of the normal equations (2.5), which can be solved by efficient solvers like (preconditioned) CGLS method, or BA-GMRES method [16] with inner iteration preconditioning [21].

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

$$\mathrm{Res}(\boldsymbol{x}^k) \equiv \min(\boldsymbol{\lambda}^k, \boldsymbol{x}^k) = \min(A^{\mathrm{T}}A\boldsymbol{x}^k - A^{\mathrm{T}}\boldsymbol{b}, \boldsymbol{x}^k), \tag{2.9}$$

and set the stopping criterion as

$$\frac{\|\mathrm{Res}(\boldsymbol{x}^k)\|_2}{\|\mathrm{Res}(\boldsymbol{x}^0)\|_2} < tol \tag{2.10}$$

with initial vector $\boldsymbol{x}^0$ and given tolerance tol. The definition (2.9) shows that $\mathrm{Res}(\boldsymbol{x}^*) = \boldsymbol{0}$ if and only if $\boldsymbol{x}^*$ is a solution of the NNLS problem (1.1) by Corollary 2.3. Meanwhile, for the iterative solution of the unconstrained least squares problems (2.8), the stopping criterion is set as

$$\frac{\|\boldsymbol{s}^k\|_2}{\|\boldsymbol{s}^0\|_2} = \frac{\|A^{\mathrm{T}}(\boldsymbol{b} - A\boldsymbol{x}^k)\|_2}{\|A^{\mathrm{T}}(\boldsymbol{b} - A\boldsymbol{x}^0)\|_2} < tol, \tag{2.11}$$

where $\boldsymbol{s}^k = -\boldsymbol{\lambda}^k$ is the residual of the normal equation (1.3). Note that (2.10) and (2.11) are used as stopping criteria of outer and inner iterations in Algorithm 2.2, respectively.

3. Convergence Analysis. In this section, we establish the convergence theory of Algorithm 2.1 in which the inner unconstrained least squares problems (2.8) are solved based on the normal equations (2.5). 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 $\boldsymbol{z}^* \in \mathbf{R}^n$ is a solution of the implicit fixed-point equation (2.4), i.e.,

$$(\Omega + A^{\mathrm{T}}A)\boldsymbol{z}^* = (\Omega - A^{\mathrm{T}}A)|\boldsymbol{z}^*| + A^{\mathrm{T}}\boldsymbol{b}, \tag{3.1}$$

and $\boldsymbol{z}^{k+1}$ is computed exactly from $\boldsymbol{z}^k$ by solving (2.5). After subtracting (3.1) from (2.5), we obtain

$$\boldsymbol{z}^{k+1} - \boldsymbol{z}^* = (\Omega + A^{\mathrm{T}}A)^{-1}(\Omega - A^{\mathrm{T}}A)(|\boldsymbol{z}^k| - |\boldsymbol{z}^*|), \tag{3.2}$$

provided that $\Omega + A^{\mathrm{T}}A$ is nonsingular. The relationship (3.2) is the basis for us to establish convergence theorems about Algorithm 2.1. 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.

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 (3.2) that

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

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 $\{\boldsymbol{z}^k\}_{k=0}^{+\infty}$ generated by (2.5) converges to the unique solution $\boldsymbol{z}^*$ 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 $A^{\mathrm{T}}A$.

3.2. 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 $\boldsymbol{x} \in \mathbf{R}^n$, $\|\boldsymbol{x}\|_Q \equiv \sqrt{\boldsymbol{x}^{\mathrm{T}}Q\boldsymbol{x}}$ and $\|\boldsymbol{x}\|_{P,q} \equiv \|P\boldsymbol{x}\|_q$ 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 [1, 28]. It follows from taking vector norm $\|\cdot\|_{\Omega^{1/2},2}$ of both sides of (3.2) that

$$\|\boldsymbol{z}^{k+1} - \boldsymbol{z}^*\|_{\Omega^{1/2},2} \le \|(\Omega + A^{\mathrm{T}}A)^{-1}(\Omega - A^{\mathrm{T}}A)\|_{\Omega^{1/2},2} \||\boldsymbol{z}^k| - |\boldsymbol{z}^*|\|_{\Omega^{1/2},2}. \tag{3.3}$$

Note that

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

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, (3.3) gives

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

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 $\{\boldsymbol{z}^k\}_{k=0}^{+\infty}$ generated by (2.5) converges to the unique solution $\boldsymbol{z}^*$ 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 $\widehat{A} = \bar{\omega}^{-1/2}AD^{-1/2} \equiv \bar{\omega}^{-1/2}\bar{A}$ and

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

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

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

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

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

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

Remark that $\bar{A}^{\mathrm{T}}\bar{A} = D^{-1/2}A^{\mathrm{T}}AD^{-1/2}$ 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 2.1.

3.3. Convergence of inexact inner iteration. Finally, the convergence analysis based on the inexact solution of the implicit fixed-point equation (2.5) is considered. Suppose $\boldsymbol{z}^k$ has already been computed. Then, $\boldsymbol{z}^{k+1}$ is computed by applying iterative methods, such as the PCG method, to (2.5). Thus, we have

$$(\Omega + A^{\mathrm{T}}A)\boldsymbol{z}^{k+1} = (\Omega - A^{\mathrm{T}}A)|\boldsymbol{z}^k| + A^{\mathrm{T}}\boldsymbol{b} + \boldsymbol{e}^k, \tag{3.4}$$

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

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

Note that if $\boldsymbol{\varepsilon}^k = 0$ for some fixed k, then $\boldsymbol{x}^* = \boldsymbol{x}^k$ is an exact solution of the fixed-point equation (2.4).

Assume that $\|\boldsymbol{e}^k\| \leq \gamma_k\|\boldsymbol{\varepsilon}^k\|$, which indicates that the error of inner iteration is controlled by the error of outer iteration. Then, it follows by subtracting (3.1) from

(3.4) that

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

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

Set

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

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

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

where $\alpha \in [0,1)$. If there exists an integer k_0 such that for all $k \ge k_0$,

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

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

Combining the analysis above, we have the following theorem.

THEOREM 3.1. *If A is of full column rank, then the iteration sequence $\{\boldsymbol{x}^k\}_{k=0}^{\infty}$ generated by modulus-type inner outer iteration Algorithm 2.2 converges to the unique solution $\boldsymbol{x}^*$ for any initial vector when*

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

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

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

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 [25] and Bardsley and Vogel [6] 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 $\boldsymbol{x}^k$ is the kth iterative solution. Then the active set is defined as

$$\mathcal{A}(\boldsymbol{x}^k) = \{j : x_j^k = 0\}, \tag{4.1}$$

and the binding set is defined as

$$\mathcal{B}(\boldsymbol{x}^k) = \{j : x_j^k = 0,\ \lambda_j^k \geq 0\}, \tag{4.2}$$

where $\boldsymbol{\lambda}^k = -A^{\mathrm{T}}\boldsymbol{r}^k = -A^{\mathrm{T}}(\boldsymbol{b} - A\boldsymbol{x}^k)$ is the Lagrange multiplier, which is also the gradient vector of the least-squares objective function in (1.2), or the negative residual of the normal equation (1.3). The corresponding set of free variables is defined as

$$\mathcal{F}(\boldsymbol{x}^k) = \{1, 2, ..., n\} \backslash \mathcal{A}(\boldsymbol{x}^k). \tag{4.3}$$

It can be easily obtained from Corollary 2.3 that $\boldsymbol{x}^*$ is the solution of the NNLS problem (1.1) if and only if $\mathcal{A}(\boldsymbol{x}^*) = \mathcal{B}(\boldsymbol{x}^*)$ and $\forall j \notin \mathcal{B}(\boldsymbol{x}^*)$, $x_j^* > 0$ and $\lambda_j^* = 0$.

In the first stage, the modulus inner outer iteration method in Algorithm 2.2 is applied to perform a fast update of the active set. By choosing $\boldsymbol{y}^0 = \boldsymbol{x}^{k+1}$, a sequence of iterates $\{\boldsymbol{y}^j\}_{j=0}^{\infty}$ is generated until either the modulus iterations fails to update a new active set when

$$\mathcal{A}(\boldsymbol{y}^j) = \mathcal{A}(\boldsymbol{y}^{j-1}) \tag{4.4}$$

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

$$|l(\boldsymbol{y}^{j-1}) - l(\boldsymbol{y}^j)| \leq \eta_1 \max\{|l(\boldsymbol{y}^{i-1}) - l(\boldsymbol{y}^i)| : 1 \leq i < j\} \tag{4.5}$$

is satisfied, where η_1 is a given tolerance and $l(\boldsymbol{y})$ is defined in (1.5).

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 $\mathcal{F}(\boldsymbol{x}^k)$, 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

$$\min_{\boldsymbol{w}\in\mathbf{R}^{\check{n}}} \|A(\boldsymbol{x}^k + Z_k\boldsymbol{w}) - \boldsymbol{b}\|_2. \tag{4.6}$$

Denote $A_{\mathcal{F}} := AZ_k$ as the submatrix of A consisting of the columns of A whose indices belong to $\mathcal{F}$. Then, the subproblem (4.6) is equivalent to the reduced unconstrained least squares problem

$$\min_{\boldsymbol{w}\in\mathbf{R}^{\check{n}}} l_k(\boldsymbol{w}) \equiv \|A_{\mathcal{F}}\boldsymbol{w} - \boldsymbol{r}^k\|_2. \tag{4.7}$$

Note that if $\mathcal{A}(\boldsymbol{x}^k) = \mathcal{A}(\boldsymbol{x}^*)$, where $\boldsymbol{x}^*$ is a solution of the NNLS (1.1) and $\boldsymbol{w}^*$ is the solution of (4.7), then $\boldsymbol{x}^* = \boldsymbol{x}^k + Z_k\boldsymbol{w}^*$. In a word, if the constraints active at the exact solution $\boldsymbol{x}^*$ 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 $\boldsymbol{w}^{k+1,0}$, let the CGLS method generate a sequence of iterates until

$$l_k(\boldsymbol{w}^{k+1,j-1}) - l_k(\boldsymbol{w}^{k+1,j}) \leq \eta_2 \max\{l_k(\boldsymbol{w}^{k+1,i-1}) - l_k(\boldsymbol{w}^{k+1,i}) : 1 \leq i < j\} \tag{4.8}$$

is satisfied, where η_2 is a given tolerance. The stopping criterion (4.8) indicates that the CGLS method fails to make sufficient progress at the jth step. After setting $\boldsymbol{w}^{k+1} \equiv \boldsymbol{w}^{k+1,j}$, in general we do not set $\boldsymbol{x}^{k+1} = \boldsymbol{x}^k + Z_k \boldsymbol{w}^{k+1}$ since this may produce negative elements in $\boldsymbol{x}^{k+1}$. Similar to the strategy used in Algorithm 1.1, we set

$$\boldsymbol{x}^{k+1} = P(\boldsymbol{x}^k + \beta^m Z_k \boldsymbol{w}^{k+1}), \tag{4.9}$$

with the smallest integer $m \geq 0$ that satisfies the sufficient decrease condition

$$\|\boldsymbol{b} - A\boldsymbol{x}^{k+1}\|_2^2 \leq \|\boldsymbol{b} - A\boldsymbol{x}^k\|_2^2 - 2\mu(\boldsymbol{s}^k)^{\mathrm{T}}(\boldsymbol{x}^{k+1} - \boldsymbol{x}^k), \tag{4.10}$$

where $0 < \beta < 1$ and $0 \leq \mu < 1$. It can be easily calculated that (4.10) is equivalent to

$$(\boldsymbol{x}^{k+1} - \boldsymbol{x}^k)^{\mathrm{T}}((2\mu - 1)\boldsymbol{s}^k - \boldsymbol{s}^{k+1}) \leq 0, \tag{4.11}$$

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

Due to the projection operation in (4.9), more elements in $\boldsymbol{x}^{k+1}$ are constrained to zero and thus $\mathcal{A}(\boldsymbol{x}^k) \subseteq \mathcal{A}(\boldsymbol{x}^{k+1})$. It can be observed that if $\mathcal{A}(\boldsymbol{x}^{k+1}) = \mathcal{A}(\boldsymbol{x}^*)$, then $\mathcal{B}(\boldsymbol{x}^{k+1}) = \mathcal{A}(\boldsymbol{x}^{k+1})$. Otherwise there exists at least one index $i \in \mathcal{A}(\boldsymbol{x}^{k+1})$ and $i \notin \mathcal{B}(\boldsymbol{x}^{k+1})$, 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, $\mathcal{B}(\boldsymbol{x}^{k+1}) = \mathcal{A}(\boldsymbol{x}^{k+1})$ is a necessary condition for $\boldsymbol{x}^{k+1} = \boldsymbol{x}^*$. In the second stage of the hybrid algorithm, if $\mathcal{B}(\boldsymbol{x}^{k+1}) = \mathcal{A}(\boldsymbol{x}^{k+1})$ holds, we update the active set and then resume the CGLS iterations until either the exact solution $\boldsymbol{x}^*$ of NNLS (1.1) is obtained, or the condition $\mathcal{B}(\boldsymbol{x}^{k+1}) = \mathcal{A}(\boldsymbol{x}^{k+1})$ 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 4.1. **Hybrid Modulus Active Set CGLS Method**

Choose an initial approximate solution $\boldsymbol{x}^0$ *and compute* $\boldsymbol{r}^0 = \boldsymbol{b} - A\boldsymbol{x}^0$.
For $k = 0, 1, 2, \ldots$ *until convergence*
  First Stage *Choose* $\boldsymbol{y}^0 = \boldsymbol{x}^k$ *and generate* $\{\boldsymbol{y}^j\}_{j=0}^{\infty}$ *by modulus inner outer iterations until either* (4.4) *or* (4.5) *is satisfied.*
  Set $\boldsymbol{x}^k = \boldsymbol{y}^j$ *and compute* $\boldsymbol{r}^k = \boldsymbol{b} - A\boldsymbol{x}^k$.
  Second Stage *Update* $\mathcal{A}(\boldsymbol{x}^k)$ *and* $\mathcal{F}(\boldsymbol{x}^k)$ *and then solve the reduced subproblem* (4.7) *by CGLS method until* (4.8) *is satisfied.*
  Compute $\boldsymbol{x}^{k+1}$ *using* (4.9) *with sufficient decrease condition* (4.10).
  If $\mathcal{B}(\boldsymbol{x}^{k+1}) = \mathcal{A}(\boldsymbol{x}^{k+1})$, *set* $\boldsymbol{x}^k = \boldsymbol{x}^{k+1}$ *and resume the* **Second Stage**; *otherwise go to the* **First Stage**.
Endfor

Here, the modulus iterations in the first stage and the CGLS iterations for the unconstrained least squares problems (4.7) in the second stage for each $k = 0, 1, 2, ...$ 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 2.2 is an inner outer iteration

method, the whole Algorithm 4.1 contains three-level iterations. The stopping criterion for the outer iteration is set as (2.10), while for inner iteration, (4.4), (4.5) and (4.8) 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 [25].

The convergence of Algorithm 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 [19] and the references therein.

THEOREM 4.1. *If A is of full column rank, and the modulus Algorithm 2.2 generates a nonincreasing objective function sequence, then the iteration sequence $\{\boldsymbol{x}^k\}_{k=0}^{\infty}$ generated by the hybrid modulus active set CGLS iteration Algorithm 4.1 converges to the unique solution $\boldsymbol{x}^*$ for any initial vector.*

Proof. Since A is of full column rank, $q(\boldsymbol{x})$ is a strictly convex quadratic function. If $\{\boldsymbol{x}^k\}_{k=0}^{\infty}$ is the iteration sequence generated by hybrid modulus active set CGLS iteration Algorithm 4.1, then $\{q(\boldsymbol{x}^k)\}_{k=0}^{\infty}$ is a nonincreasing sequence by the fact that the modulus algorithm generates a nonincreasing objective function sequence and the sufficient decrease condition in (4.10). Therefore, $\{\boldsymbol{x}^k\}_{k=0}^{\infty}$ is bounded. In addition, Theorem 3.1 guarantees that every limit point of $\{\boldsymbol{x}^k\} \subseteq \mathcal{K}_{mod}$, where $\mathcal{K}_{mod}$ is the set of iterates generated by the modulus iteration, is a stationary point of the NNLS (1.1). Hence, there exists a subsequence $\boldsymbol{x}^{k_j} \to \boldsymbol{x}^*$, and thus $q(\boldsymbol{x}^k) \to q(\boldsymbol{x}^*)$.

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

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

for any $\boldsymbol{x} \geq 0$. Hence,

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

where $\lambda_{\min} > 0$ is the minimum eigenvalues of $A^{\mathrm{T}}A$. We can easily obtain $\boldsymbol{x}^k \to \boldsymbol{x}^*$ 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 2.2 as $\boldsymbol{z}^{k+1} = \boldsymbol{z}^k + \beta^m \boldsymbol{w}^{k+1}$ and choose the smallest integer $m \geq 0$ that satisfies the sufficient decrease condition (4.10).

5. Numerical Experiments. Finally, numerical experiment results are presented to show the performance of the modulus-type inner outer iteration Algorithm 2.2 and the two-stage hybrid modulus active set CG iteration Algorithm 4.1. We compare them to the projected gradient Algorithm 1.1, the projected NR-SOR Algorithm 1.2 and the GPCG method in [25] which replaces the modulus method in the first stage of Algorithm 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 $\epsilon = 1.1 \times 10^{-16}$. The initial vectors for the outer and inner iterations were chosen to be zero vector. For the modulus-type iteration methods, the parameter matrix was chosen to be $\Omega = \omega I$ or $\Omega = \omega \mathrm{diag}(A^{\mathrm{T}}A)$, where ω is a positive parameter. The abbreviations for all the compared methods are listed in Table 1. Remark that

TABLE 1
Abbreviations for the compared methods.

Method	Description
PG	Projected gradient method (Algorithm 1.1)
PSOR	Projected NR-SOR method (Algorithm 1.2)
Mod	Modulus method (Algorithm 2.2 with $\Omega = \omega I$)
GMod	Modulus method (Algorithm 2.2 with $\Omega = \omega \mathrm{diag}(A^{\mathrm{T}}A)$)
GPCG	Hybrid projected gradient active set CG method [25]
ModASCG	Hybrid modulus active set CG method (Algorithm 4.1 with $\Omega = \omega I$)
GModASCG	Hybrid modulus active set CG method (Algorithm 4.1 with $\Omega = \omega \mathrm{diag}(A^{\mathrm{T}}A)$)

all the inner least squares problems are solved by the CGLS method, which is easy to implement and needs small storage requirement. However, for ill-conditioned and rank-deficient problems, one would recommend the BA-GMRES method [16] with suitable preconditioners [21] for the solution of unconstrained least squares problems in order to achieve better convergence with less iteration steps and CPU time.

As mentioned in (2.10), the stopping criterion for the outer iteration of all methods is chosen as $\|\mathrm{Res}(\boldsymbol{x}^k)\|_2/\|\mathrm{Res}(\boldsymbol{x}^0)\|_2 < tol$. For one stage methods including PG, PSOR, Mod and GMod, the stopping criterion for the inner unconstrained least squares problems is chosen as (2.11)

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

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 and tol_{in} 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 (4.8), (4.4) and (4.5). In order to perform a fair comparison among different methods, the parameters in (4.8), (4.5) and the sufficient decrease condition (4.10) are chosen as

$$\eta_1 = \eta_2 = 0.1, \quad \mu = 0.1 \quad \text{and} \quad \beta = 0.9 \tag{5.1}$$

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

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.

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 $A = U\Sigma V^{\mathrm{T}}$, 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 [13, 16].

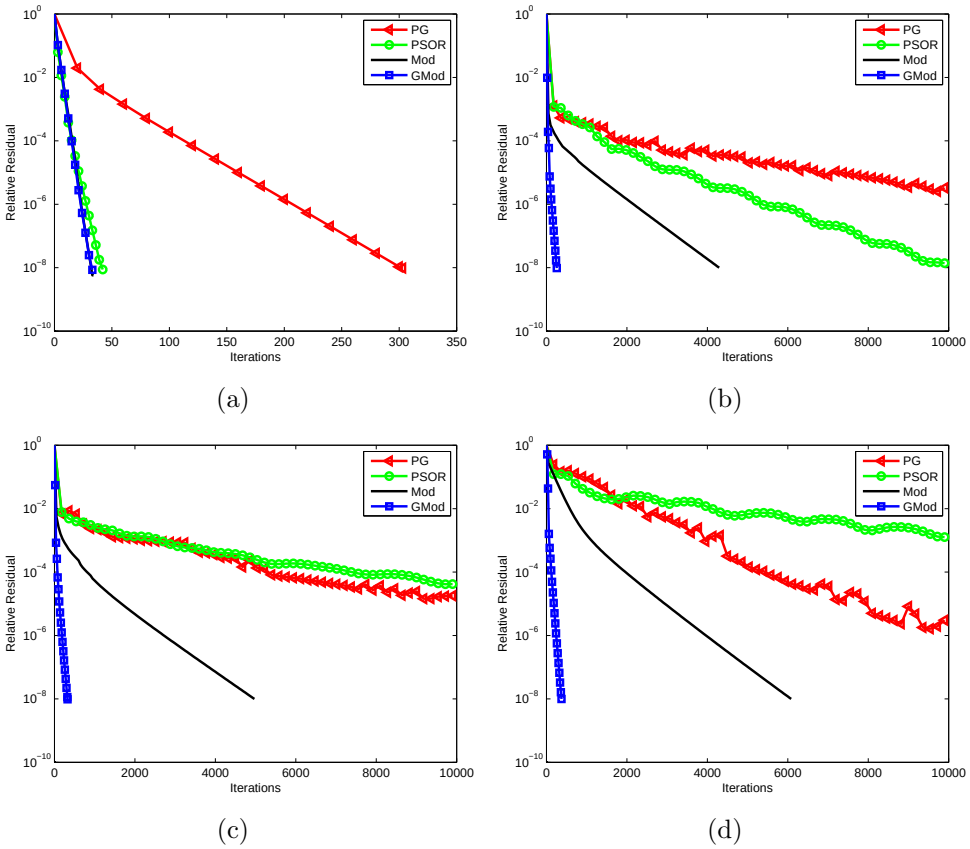


FIG. 1. *Relative residual vs. iterations for (a)* $\rho = 1$*, (b)* $\rho = 0.9$*, (c)* $\rho = 0.8$ *and (d)* $\rho = 0.7$ *with* $\sigma_n = 0.01$ $(\kappa(A) = 100)$.

In 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 $\boldsymbol{b}$ are generated randomly following the normal distribution with mean zero and variance 1, using the Matlab function randn$(m, 1)$. The same $\boldsymbol{b}$ is used for all the cases.

We compared the four testing methods, PG, PSOR, Mod and GMod, where the inner unconstrained least squares problems (2.8) 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 1 and 2, we depict the curves of the relative residual $\|\mathrm{Res}(\boldsymbol{x}^k)\|_2/\|\mathrm{Res}(\boldsymbol{x}^0)\|_2$ 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), (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 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 dis-

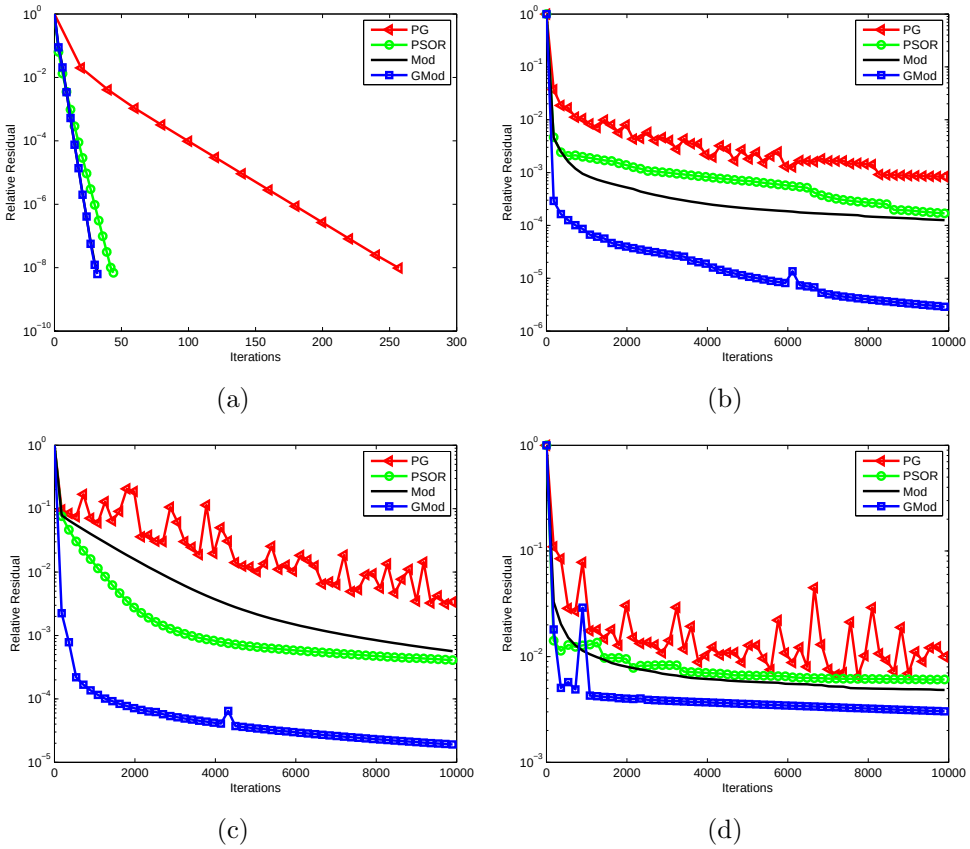


FIG. 2. *Relative residual vs. iterations for (a)* $\rho = 1$*, (b)* $\rho = 0.9$*, (c)* $\rho = 0.8$ *and (d)* $\rho = 0.7$ *with* $\sigma_n = 0.0001$ $(\kappa(A) = 10^4)$.

tributed with $\rho = 1$, Mod and GMod methods show similar convergence performance. This shows that the modulus method with $\Omega = \omega \mathrm{diag}(A^{\mathrm{T}}A)$ outperforms projection-type iterative methods, and the choice of $\Omega = \omega \mathrm{diag}(A^{\mathrm{T}}A)$ in GMod is more efficient than the choice of $\Omega = \omega I$ in Mod, which confirms our convergence analysis.

Figure 2 shows similar convergence phenomena as in Figure 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 1 and 2. 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 (2.8) are solved by the CGLS method, in Tables 2 and 3 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 3, where the tolerance for PG, Mod and

TABLE 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	∗0.00	0.00	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.
($tol = 10^{-8},\ tol_{in} = 10^{-2}/k$)

GMod is $tol = 10^{-3}$. The tolerance for inner iteration is chosen as $tol_{in} = 10^{-2}/k$ for all the one stage methods, and the parameters in two-stage methods are chosen as (5.1). In addition, the parameters in Mod, GMod, ModASCG and GModASCG methods are chosen as $\omega = 0.1$.

In Tables 2 and 3, 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 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 3, 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 2 and 3, 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 $\Omega = \omega \mathrm{diag}(A^{\mathrm{T}}A)$.

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

Moreover, it can be observed from Tables 2 and 3 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 3, 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 (d) for all the one-stage numerical methods. In addition, it is further confirmed in Tables 2 and 3 that the convergence behavior of all methods deteriorate as the condition number of the matrix A increases, as was shown in Figures 1 and 2. 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.

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

TABLE 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.

problems where the elements of the right-hand side vector $\boldsymbol{b}$ are generated randomly using the Matlab function randn$(m, 1)$. The same $\boldsymbol{b}$ is used for all the cases.

The numerical results are shown in Table 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 $\omega = 0.1$.

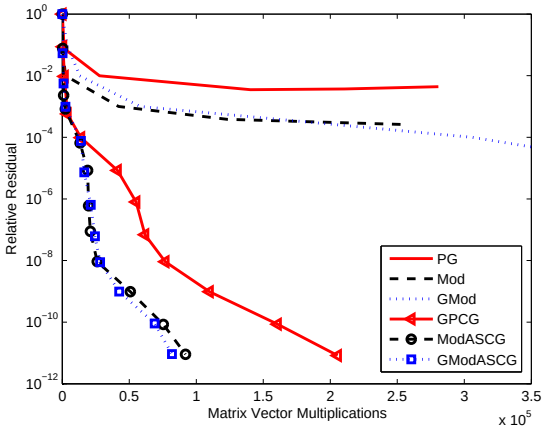


FIG. 3. *Relative residual vs. matrix vector multiplications for Randn_4 (inconsistent).*

TABLE 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 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, we plot the relative residual $\|\mathrm{Res}(\boldsymbol{x}^k)\|_2/\|\mathrm{Res}(\boldsymbol{x}^0)\|_2$ 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.

5.3. Sparse rank deficient case. In the following, we test a class of rectangular matrices from the University of Florida Sparse Matrix Collection [10]. We construct the rank-deficient overdetermined systems by deleting all the zero rows and

TABLE 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 iteration parameters.
($tol = 10^{-6},\ tol_{in} = 10^{-2}/k$)

zero columns. The resulting number of rows m, columns n, nonzero elements nnz, as well as the rank, are given in Table 5.

For the consistent case, we form NNLS problems where the right-hand side vector $\boldsymbol{b} = A\boldsymbol{x}^*$ and $\boldsymbol{x}^* = [1, 0, 1, 0, ...]^{\mathrm{T}} \in \mathbf{R}^n$. The numerical results are shown in Table 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, by minimizing the number of matrix vector

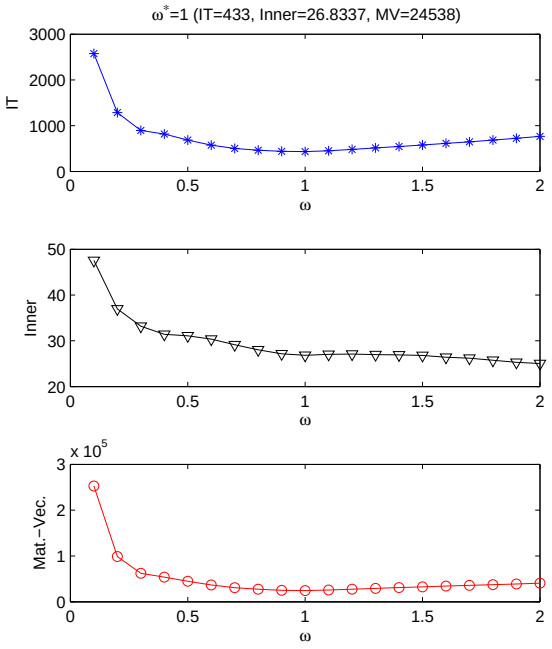


FIG. 4. *Number of outer iterations, inner iterations and matrix vector multiplication vs. ω for Mod method in Maragal_3.*

multiplications. We chose the optimal μ practically by changing it from 0.1 to 2, and chose the optimal ω by changing it from 0 to 0.9 with an interval of 0.1.

Table 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 4 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 5, we plot the relative residual $\|\mathrm{Res}(\boldsymbol{x}^k)\|_2/\|\mathrm{Res}(\boldsymbol{x}^0)\|_2$ 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.

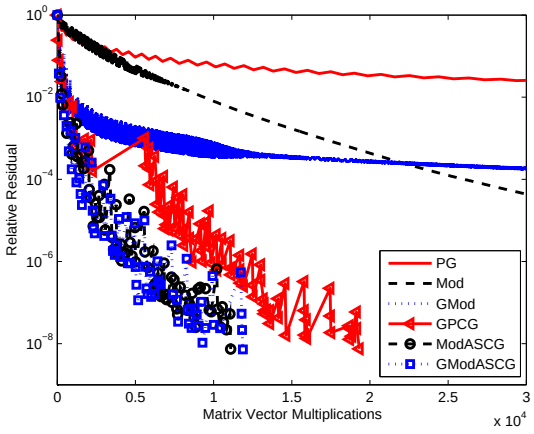


FIG. 5. *Relative residual vs. matrix vector multiplications for Maragal_5 (consistent).*

For the inconsistent case, we form inconsistent NNLS problems where the elements of the right-hand side vector $\boldsymbol{b}$ are generated randomly using the Matlab function randn$(m, 1)$. The numerical results are shown in Table 7, where the tolerance for the outer iterations is chosen to be $tol = 10^{-6}$.

Similar to the phenomena observed in Table 6, Table 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 and Maragal_8. Compared with the consistent problems in Table 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.

5.4. Image restoration. Lastly, we apply the proposed method to the solution of nonnegative constrained ill-posed problems

$$A\boldsymbol{x} = \boldsymbol{b} \quad \text{subject to} \quad \boldsymbol{x} \geq \boldsymbol{0}, \tag{5.2}$$

where $A \in \mathbf{R}^{n\times n}$ is a matrix with ill-determined rank and has many singular values of different orders of magnitude close to the origin. In many linear discrete ill-posed problems that arise in science and engineering, the right hand side vector is contaminated by measurement errors [8]. For example, in image restoration problems, the right hand side vector is contaminated by noise, and the desired solution $\boldsymbol{x}$ whose entries represent pixels values is known to be nonnegative. Hence, (5.2) is generally inconsistent and thus one has to solve a NNLS problem with some regularization. Here, we use the Tikhonov regularization (i.e., stabilization with an additive penalty term [14])

$$\min_{\boldsymbol{x}\in\mathbf{R}^n} \|A\boldsymbol{x} - \boldsymbol{b}\|_2^2 + \alpha\|\boldsymbol{x}\|_2^2 \equiv \left\| \begin{bmatrix} A \\ \sqrt{\alpha}I \end{bmatrix} \boldsymbol{x} - \begin{bmatrix} \boldsymbol{b} \\ 0 \end{bmatrix} \right\|_2^2 \quad \text{subject to} \quad \boldsymbol{x} \geq \boldsymbol{0}, \tag{5.3}$$

TABLE 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 iteration parameter ω.
($tol = 10^{-6},\ tol_{in} = 10^{-2}/k$)

where $\alpha > 0$ is the regularization parameter, since it is convenient for comparing the efficiency of the different NNLS solvers.

We test the numerical methods PG, Mod, GPCG and ModASCG on image restoration problems, which come from Nagy's Matlab toolbox "RestoreTools" [4]. 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 8. Note that the matrix A is deter-

TABLE 8
Definitions in image restoration.

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

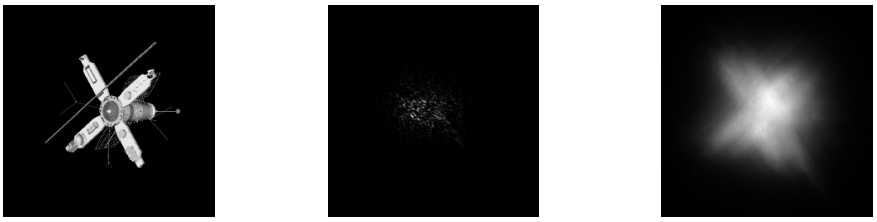


FIG. 6. *The exact image (left), PSF function (middle) and large blurred and noisy image (right) of test problem "AtmosphericBlur".*

mined by the point spread function. Vector $\boldsymbol{e}$ is generated with normally distributed entries with zero mean by Matlab. The noise level δ is set to be 5% and the regularization parameter is set to be $\alpha = 10^{-4}$. The parameters in Mod and ModASCG are set to be $\omega = 0.1$. The relative error of the restored image is defined as

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

In Figures 6 and 7, the exact image and the blurred and noisy image of test problems "AtmosphericBlur" and "Text" are shown, respectively. Moreover, in Figure 8, we depict the relative error curves of four testing methods versus the number of outer iterations for the two image restoration problems, respectively. In Figures 9 and 10, the restored images are shown.

From Figure 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 "AtmosphericBlur" problem, 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 "Text" problem. The modulus type methods outperform projection type methods by obtaining more accurate solutions with same computational costs.

6. Concluding Remarks. 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

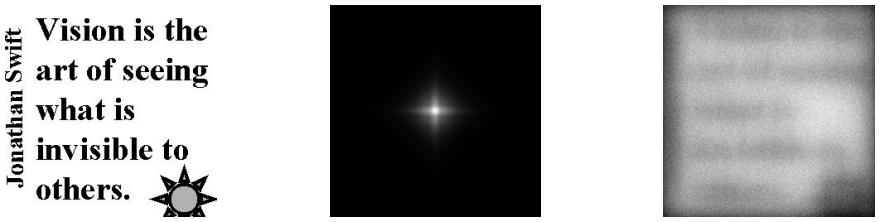


FIG. 7. *The exact image (left), PSF function (middle) and large blurred and noisy image (right) of test problem "Text".*

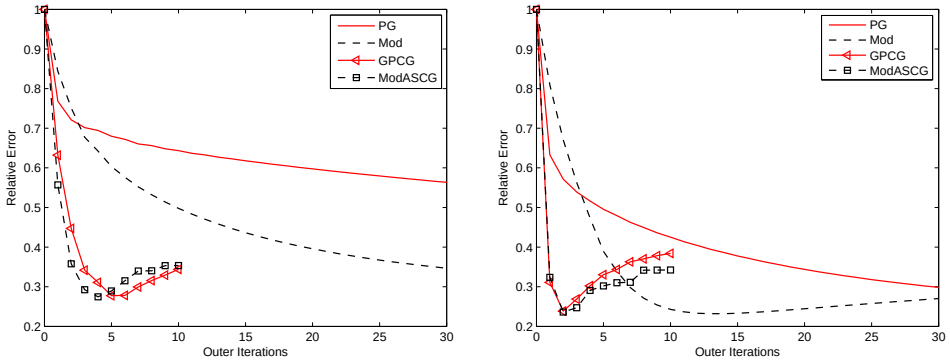
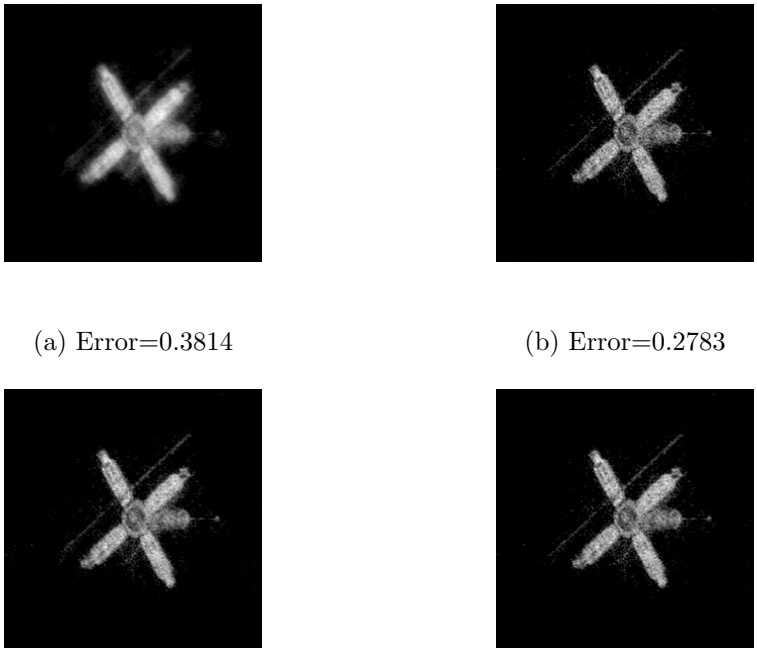


FIG. 8. *Relative error vs. outer iterations for test problems "AtmosphericBlur" (left) and "Text" (right).*

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.

Appendix A. Equivalence between the nonnegative constrained quadratic programming and the linear complementarity problem in Theorem 2.1. If $\boldsymbol{x}^*$ is a solution of LCP$(B, \boldsymbol{c})$, then it holds that

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



(a) Error=0.3814 (b) Error=0.2783

(c) Error=0.2775 (d) Error=0.2747

FIG. 9. *Restored images by (a) PG, (b) Mod, (c) GPCG and (d) ModASCG methods for test problem "AtmosphericBlur".*

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

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

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

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

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

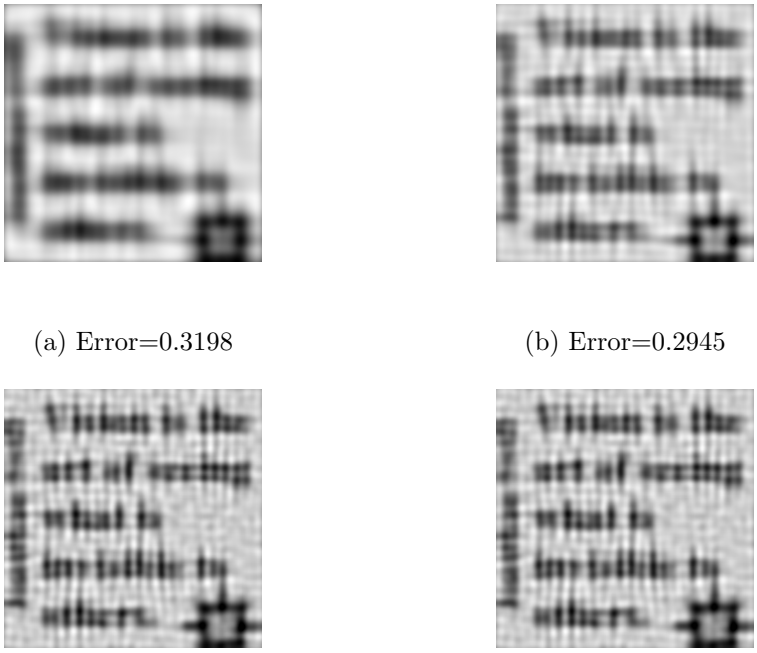
If $\boldsymbol{x}^*$ is a solution of NNQP$(B, \boldsymbol{c})$, then $\boldsymbol{x}^*$ satisfies the necessary KKT conditions as follows. There exists $\boldsymbol{f} \in \mathbf{R}^n$, called KKT multipliers, such that

Stationarity

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

Primal and Dual feasibility

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



(a) Error=0.3198 (b) Error=0.2945

(c) Error=0.2762 (d) Error=0.2763

FIG. 10. *Restored images by (a) PG, (b) Mod, (c) GPCG and (d) ModASCG methods for test problem "Text".*

Complementarity slackness

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

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

Acknowledgement. The authors would like to thank Dr. Keiichi Morikuni for valuable remarks which greatly improved the quality of the paper. The authors also would like to thank Professors Lothar Reichel, Per Christian Hansen, James Nagy, Zhong-Zhi Bai, Wei Xu, Xiao-Ming Yuan, Ms. Yiran Cui and Mr. Kota Sugihara for valuable discussions. The first author would like to express his gratitude to Professor Ken Hayami for his courteous attention during the six-month internship in Tokyo.

We would also like to thank the referees for valuable remarks which helped to greatly improve this paper.

REFERENCES

[1] Z.-Z. BAI, *Modulus-based matrix splitting iteration methods for linear complementarity problems*, Numer. Linear Algebra Appl., 6 (2010), pp. 917–933.
[2] D. P. BERTSEKAS, *On the Goldstein-Levitin-Polyak gradient projection method*, IEEE Trans. Automat. Control, 21 (1976), pp. 174–184.
[3] Å. BJÖRCK, *Numerical Methods for Least Squares Problems*, SIAM, Philadelphia, 1996.
[4] S. BERISHA AND J. G. NAGY, *Iterative Methods for Image Restoration*, (2012), `http://www.mathcs.emory.edu/~nagy/RestoreTools`.

[5] 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.
[6] 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.
[7] C. W. Cryer, *The solution of a quadratic programming using systematic overrelaxation*, SIAM J. Control 9 (1971), pp. 385–392 .
[8] D. Calvetti, G. Landi, L. Reichel, and F. Sgallari, *Nonnegativity and iterative methods for ill-posed problems*, Inverse Problems, 20 (2004), pp. 1747–1758.
[9] P. H. Calamai and J. J. Moreé, *Projected gradient methods for linearly constrained problems*, Math. Programming, 39 (1987), pp. 93–116.
[10] T. Davis, *The University of Florida Sparse Matrix Collection*, available online at `http://www.cise.ufl.edu/research/sparse/matrices`.
[11] Z. Dostál, *Box constrained quadratic programming with proportioning and projections*, SIAM J. Optim., 7 (1997), pp. 871–887.
[12] 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.
[13] A. Greenbaum, *Iterative Methods for Solving Linear Systems*, Frontiers Appl. Math. 17, SIAM, Philadelphia, 1997.
[14] P. C. Hansen, *Regularization tools: A Matlab package for analysis and solution of discrete ill-posed problems*, Numer. Algorithms 6 (1994), pp. 1–35.
[15] M. Hanke, J. G. Nagy and C. Vogel, *Quasi-Newton approach to nonnegative image restorations*, Linear Algebra Appl., 316 (2000), pp. 223–236.
[16] K. Hayami, J.-F. Yin, and T. Ito, *GMRES methods for least squares problems*, SIAM J. Matrix Anal. Appl., 31 (2010), pp. 2400–2430.
[17] A. Klarbring, *Quadratic programs in frictionless contact problems*, Internat. J. Engrg. Sci., 24 (1986), pp. 1207–1217.
[18] N. W. Kappel and L. T. Watson, *Iterative algorithms for the linear complementarity problems*, Int. J. Comput. Math., 19 (1986), pp. 273–297.
[19] M. Kočvara and J. Zowe, *An iterative two-step algorithm for linear complementarity problems*, Numer. Math., 68 (1994), pp. 95–106.
[20] P. Lötstedt, *Soving the minimal least squares problem subject to bounds on the variables*, BIT Numerical Mathematics, 24 (1984), pp. 206–224.
[21] K. Morikuni and K. Hayami, *Inner-iteration Krylov subspace methods for least squares problems*, SIAM J. Matrix Anal. Appl., 34 (2013), pp. 1–22.
[22] 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.
[23] 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.
[24] J. J. Moré and G. Toraldo, *Algorithms for bound constrained quadratic programming problems*, Numer. Math., 55 (1989), pp. 337–400.
[25] 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.
[26] 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.
[27] 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.
[28] J. Ortega and W. Rheinboldt, *Iterative Solution of Nonlinear Equations in Several Variables*, Academic Press: New York, 1970.
[29] B. T. Polyak, *The conjugate gradient method in extremal problems*, U.S.S.R. Computational Mathematics and Mathematical Physics, 9 (1969), pp. 94–112.
[30] S. J. Wright, *Implementing proximal point methods for linear programming*, Report MCS-P45-0189, Algonne National Laboratory, Algonne, IL, 1989.
[31] 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.
[32] N. Zheng and J.-F. Yin, *Accelerated modulus-based matrix splitting iteration methods for linear complementarity problems*, Numer. Algorithms 64 (2013), pp. 245–262.