ISSN 1346-5597

National Institute of Informatics

NII Technical Report

Inner-Iteration Krylov Subspace Methods for Least Squares Problems

Keiichi Morikuni and Ken Hayami

NII-2011-001E
Apr. 2011

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

INNER-ITERATION KRYLOV SUBSPACE METHODS FOR LEAST SQUARES PROBLEMS *

KEIICHI MORIKUNI† AND KEN HAYAMI‡

Abstract. Stationary inner iterations in combination with Krylov subspace methods are proposed for least squares problems. The inner iterations are efficient in terms of computational work and memory, and serve as powerful preconditioners also for ill-conditioned and rank-deficient least squares problems. Theoretical justifications for using the inner iterations as preconditioners are presented. Numerical experiments for overdetermined least squares problems including ill-conditioned and rank-deficient problems show that the proposed methods outperform previous methods.

Key words. least squares problems, iterative method, preconditioner, inner-outer iteration, Krylov subspace method, GMRES method, CG method, Cimmino's method, Kaczmarz's method

AMS subject classifications. 65F08, 65F10, 65F20, 65F50

1. Introduction.

Consider solving large sparse linear least squares problems

$$\min_{\boldsymbol{x}\in\mathbf{R}^n} \|\boldsymbol{b} - A\boldsymbol{x}\|_2, \tag{1.1}$$

where $A \in \mathbf{R}^{m\times n}$, $\boldsymbol{b} \in \mathbf{R}^m$, and A is not necessarily full rank. (1.1) is equivalent to the normal equation

$$A^T A\boldsymbol{x} = A^T\boldsymbol{b}. \tag{1.2}$$

For the under-determined case, $m < n$, the problem of computing the minimum norm solution $\min_{\boldsymbol{x}\in\mathbf{R}^n} \|\boldsymbol{x}\|_2$ subject to $A\boldsymbol{x} = \boldsymbol{b}$ is equivalent to

$$AA^T\boldsymbol{u} = \boldsymbol{b}, \tag{1.3}$$

where $A^T\boldsymbol{u} = \boldsymbol{x}$, if it exists.

Direct methods for solving the problems (1.2) or (1.3) are regarded as expensive for large sparse problems. On the other hand, since $\kappa(A^TA) = \kappa(AA^T) = \kappa(A)^2$, iterative methods may be slow to converge. Here, $\kappa(A) = \sigma_{\max}/\sigma_{\min}$ is the condition number, where $\sigma_{\max}$ and $\sigma_{\min}$ are the largest and smallest positive singular values of A, respectively. Hence, when iterative methods are used, good preconditioners are necessary to achieve better convergence with small storage requirement [7]. For this purpose, preconditioners such as [21], [26], [36], [4], [6], [11], [37], [18], [12] have been proposed for the iterative solution of least squares problems.

For solving systems of linear equations, inner iterations can be applied inside the Krylov subspace methods instead of preconditioning matrices explicitly. Such techniques are often called inner-outer iteration methods. Inner-outer iteration conjugate gradient (CG) methods [3], [17], [23] were proposed for the symmetric case. For

*This work was supported by the Grants-in-Aid for Scientific Research (C) of the Ministry of Education, Culture, Sports, Science and Technology, Japan.

†Department of Informatics, School of Multidisciplinary Sciences, The Graduate University for Advanced Studies, Sokendai, 2-1-2 Hitotsubashi, Chiyoda-ku, Tokyo, 101-8430, Japan (morikuni@nii.ac.jp).

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

the nonsymmetric case, methods for preconditioning the generalized minimal residual (GMRES) method using inner iterations were proposed, such as the flexible GMRES (FGMRES) [27] and GMRESR [33]. See also [16], [24]. In [27], the CG normal residual (CGNR) method, the GMRES method preconditioned by an incomplete LU factorization, and the GMRES method itself, and in [24], the GMRES method itself and the symmetric successive over-relaxation (SSOR) method are applied as inner iterations, respectively. The variable preconditioned generalized conjugate residual (VPGCR) method [1] employs the successive over-relaxation (SOR) method as the inner iterations for the preconditioning. A similar inner-iteration technique can be found in [14]. Some success with the biorthogonal Lanczos-based methods have been reported as outer solvers combined with the same type of methods as the inner iterations [31], [35]. General flexible Krylov subspace methods where Krylov subspace inner iterations are applied to Krylov subspace methods for solving linear systems of equations were analysed in [30].

As for least squares problems, SOR-like inner iterations for preconditioning GCR for square matrices were proposed especially for singular systems and least squares problems in [2]. In this paper, we propose using Jacobi-type (Cimmino [10]) and SOR-type (Kaczmarz [22]) stationary iterative methods specifically designed for least squares problems, as inner-iteration preconditioners for CG and GMRES. These stationary iterative methods are also discussed in [28, Chapter 8], and are applicable to (1.2) or (1.3) without explicitly constructing A^TA and AA^T. A significant advantage of such inner-iteration preconditioners is that one can avoid explicitly computing and storing the preconditioning matrix.

The main motivation for proposing the new preconditioners is to reduce CPU time and memory significantly and to broaden the scope of problems which can be solved. Note that previously BA-GMRES with RIF was comparable with, but not definitely superior to, reorthogonalized CGLS with RIF in terms of CPU time for ill-conditioned problems [18]. Moreover, RIF for least squares problems [6] will break down for rank-deficient matrices. We aim to improve these points.

The rest of the paper is organized as follows. In section 2, we describe the preconditioning framework for solving least squares problems using Krylov subspace methods with inner iterations. In section 3, we present and analyse the iterative methods used for the inner iteration. In section 4, we show numerical experiment results on each combination for the outer and inner iterations. Section 5 summarizes the main conclusions of the paper.

2. Preconditioning framework for inner iteration. This section presents several frameworks for applying inner iteration to the Krylov subspace methods for least squares problems.

In [18], the right-preconditioned and left-preconditioned GMRES methods for least squares problems were proposed. The former is called the AB-GMRES and the latter is called the BA-GMRES. AB-GMRES applies GMRES [29] to $\min_{\boldsymbol{u}\in\mathbf{R}^m} \|\boldsymbol{b} - AB\boldsymbol{u}\|_2$, whereas BA-GMRES applies GMRES to $\min_{\boldsymbol{x}\in\mathbf{R}^n} \|B\boldsymbol{b} - BA\boldsymbol{x}\|_2$, where $B \in \mathbf{R}^{n\times m}$ denotes the preconditioning matrix. Conditions for the preconditioner B for the GMRES methods for least squares problems were investigated. Among the theorems, the following are relevant to the present paper.

THEOREM 2.1. $\min_{\boldsymbol{x}\in\mathbf{R}^n} \|\boldsymbol{b} - A\boldsymbol{x}\|_2 = \min_{\boldsymbol{z}\in\mathbf{R}^m} \|\boldsymbol{b} - AB\boldsymbol{z}\|_2$ *holds for all* $\boldsymbol{b} \in \mathbf{R}^m$ *if and only if* $\mathcal{R}(AB) = \mathcal{R}(A)$.

THEOREM 2.2. $\min_{\boldsymbol{x}\in\mathbf{R}^n} \|\boldsymbol{b} - A\boldsymbol{x}\|_2$ *and* $\min_{\boldsymbol{x}\in\mathbf{R}^n} \|B\boldsymbol{b} - BA\boldsymbol{x}\|_2$ *are equivalent for all*

$\boldsymbol{b} \in \mathbf{R}^m$ *if and only if* $\mathcal{R}(A) = \mathcal{R}(B^T BA)$.

THEOREM 2.3. *If* $\mathcal{R}(A^T) = \mathcal{R}(B)$ *holds, then AB-GMRES determines a least squares solution of* $\min_{\boldsymbol{x}\in\mathbf{R}^n} \|\boldsymbol{b} - A\boldsymbol{x}\|_2$ *for all* $\boldsymbol{b} \in \mathbf{R}^m$ *and for all* $\boldsymbol{x}_0 \in \mathbf{R}^n$ *without breakdown if and only if* $\mathcal{R}(A) = R(B^T)$.

THEOREM 2.4. *If* $\mathcal{R}(A) = \mathcal{R}(B^T)$ *holds, then BA-GMRES determines a least squares solution of* $\min_{\boldsymbol{x}\in\mathbf{R}^n} \|\boldsymbol{b} - A\boldsymbol{x}\|_2$ *for all* $\boldsymbol{b} \in \mathbf{R}^m$ *and for all* $\boldsymbol{x}_0 \in \mathbf{R}^n$ *without breakdown if and only if* $\mathcal{R}(A^T) = R(B)$.

Here $\mathcal{R}(A)$ denotes the range space of A. In the paper [18], the robust incomplete factorization (RIF) [6] was used to construct the factors of the preconditioner B. RIF is guaranteed to work for full rank matrices. The preconditioning matrices are explicitly constructed, saved, and applied at each step of GMRES.

2.1. AB-GMRES method. Consider using AB-GMRES [18] to solve least squares problems. Instead of forming an explicit preconditioner matrix B, we propose applying several steps of a certain iterative method inside the GMRES whenever B is needed in the AB-GMRES. Therefore, our strategy can be considered as a preconditioned GMRES with an implicit B. If $\mathcal{R}(A) = \mathcal{R}(B^T)$ and $\mathcal{R}(A^T) = \mathcal{R}(B)$ hold, then, from Theorem 2.3, GMRES solves $\min_{\boldsymbol{u}\in\mathbf{R}^n} \|\boldsymbol{b} - AB\boldsymbol{u}\|_2$ and determines a solution to $\min_{\boldsymbol{x}\in\mathbf{R}^n} \|\boldsymbol{b} - A\boldsymbol{x}\|_2$ without breakdown, where $\boldsymbol{x} = B\boldsymbol{u}$. When the number of inner iterations is constant, the Krylov subspace $\mathcal{K}_j(AB, \boldsymbol{r}_0) = \text{span}\left\{\boldsymbol{r}_0, AB\boldsymbol{r}_0, \ldots, (AB)^{j-1}\boldsymbol{r}_0\right\}$ is involved. Note that B is not formed or stored explicitly. Similar to FGMRES [27] and GMRESR [33], the number of inner iterations can be changed for each outer-iteration.

Algorithm 1 shows the general framework for this approach. Note that l is the number of the restart cycle. In the following and hereafter, $(\boldsymbol{a}, \boldsymbol{b})$ denotes the inner product $\boldsymbol{a}^T\boldsymbol{b}$ between vectors $\boldsymbol{a}$ and $\boldsymbol{b}$.

Algorithm 2.1 AB-GMRES(l) method with inner iterations.

1. *Let* $\boldsymbol{x}_0$ *be the initial approximate solution.*
2. *Compute* $\boldsymbol{r}_0 = \boldsymbol{b} - A\boldsymbol{x}_0, \beta = \|\boldsymbol{r}_0\|_2, \boldsymbol{v}_1 = \boldsymbol{r}_0/\beta$
3. *For* $j = 1, 2, \ldots, l$, *Do*
4. *Roughly solve* $\min_{\boldsymbol{u}_j\in\mathbf{R}^m} \|\boldsymbol{v}_j - AA^T\boldsymbol{u}_j\|_2$ *to obtain* $\boldsymbol{z}_j = A^T\tilde{\boldsymbol{u}}_j = B_j\boldsymbol{v}_j$ *by using an inner iteration.*
5. $\boldsymbol{w}_j = A\boldsymbol{z}_j (= AB_j\boldsymbol{v}_j)$
6. *For* $i = 1, 2, \ldots, j$, *Do*
7. $h_{i,j} = (\boldsymbol{w}_j, \boldsymbol{v}_i)$
8. $\boldsymbol{w}_j = \boldsymbol{w}_j - h_{i,j}\boldsymbol{v}_i$
9. *EndDo*
10. $h_{j+1,j} = \|\boldsymbol{w}_j\|_2$
11. $\boldsymbol{v}_{j+1} = \boldsymbol{w}_j/h_{j+1,j}$
12. *Find* $\boldsymbol{y}_j \in \mathbf{R}^j$ *that minimizes* $\|\beta\boldsymbol{e}_1 - \bar{H}_j\boldsymbol{y}\|_2$.
13. $\boldsymbol{x}_j = \boldsymbol{x}_0 + [\boldsymbol{z}_1, \boldsymbol{z}_2, \ldots, \boldsymbol{z}_j]\,\boldsymbol{y}_j$
14. *If* $\|A^T(\boldsymbol{b} - A\boldsymbol{x}_j)\|_2 < \varepsilon\|A^T\boldsymbol{r}_0\|_2$ *, then stop.*
15. *EndDo*
16. $\boldsymbol{x}_0 = \boldsymbol{x}_l$ *and go to 2*

Here, $\bar{H}_j \equiv \{h_{pq}\} \in \mathbf{R}^{(j+1)\times j}$ and $\boldsymbol{e}_1 \equiv (1, 0, \ldots, 0)^T$. The method is said to break down when $h_{j+1,j} = 0$.

The idea behind line 4 is as follows. First consider the problem:

$$\text{Minimize } \|\boldsymbol{z}\|_2 \quad \text{subject to} \quad A\boldsymbol{z} = \boldsymbol{v} \in \mathcal{R}(A). \tag{2.1}$$

This problem is equivalent to

$$AA^T\boldsymbol{u} = \boldsymbol{v} \in \mathcal{R}(A), \quad \boldsymbol{z} = A^T\boldsymbol{u}, \tag{2.2}$$

which, in turn, is equivalent to

$$AB\boldsymbol{u}' = \boldsymbol{v} \in \mathcal{R}(A), \quad \boldsymbol{z} = B\boldsymbol{u}', \tag{2.3}$$

if $\mathcal{R}(A^T) = \mathcal{R}(B)$. Therefore, if we roughly solve (2.2) to obtain $\tilde{\boldsymbol{u}}$ such that $AA^T\tilde{\boldsymbol{u}} \simeq \boldsymbol{v}$ and set $\boldsymbol{z} = A^T\tilde{\boldsymbol{u}} \equiv B\boldsymbol{v}$, we have $AB\boldsymbol{v} \simeq \boldsymbol{v}$, so that B will serve as a preconditioner for solving (2.1)–(2.3).

Next, consider the problem

$$\min_{\boldsymbol{z}\in\mathbf{R}^n} \|\boldsymbol{v} - A\boldsymbol{z}\|_2, \tag{2.4}$$

where $\boldsymbol{v} \in \mathcal{R}(A)$ does not necessarily hold. Now,

$$\min_{\boldsymbol{u}\in\mathbf{R}^m} \|\boldsymbol{v} - AB\boldsymbol{u}\|_2, \quad \boldsymbol{z} = B\boldsymbol{u} \tag{2.5}$$

is equivalent to (2.4) if and only if $\mathcal{R}(A) = \mathcal{R}(AB)$ [18]. Note that $\mathcal{R}(A^T) = \mathcal{R}(B)$ implies $\mathcal{R}(A) = \mathcal{R}(AB)$ [18]. Hence, if we roughly solve $\min_{\boldsymbol{u}\in\mathbf{R}^m} \|\boldsymbol{v} - AA^T\boldsymbol{u}\|_2$ to obtain $\|\boldsymbol{v}-AA^T\tilde{\boldsymbol{u}}\|_2 \simeq \min_{\boldsymbol{u}\in\mathbf{R}^m} \|\boldsymbol{v} - AA^T\boldsymbol{u}\|_2$, and if we let $\boldsymbol{z} = A^T\tilde{\boldsymbol{u}} \equiv B\boldsymbol{v}$, we have $\min_{\boldsymbol{u}\in\mathbf{R}^m} \|\boldsymbol{v}-AB\boldsymbol{u}\|_2 \simeq \|\boldsymbol{v} - AB\boldsymbol{v}\|_2$. Therefore, in this case also, B will serve as a preconditioner for solving (2.4). Note that we have denoted B as B_j, to allow the possibility that B_j may be different for each j. For instance, the number of inner iterations may be different for each outer iteration.

Thus, line 4 can be considered as approximately solving another least squares problem: $\min_{\boldsymbol{z}\in\mathbf{R}^n} \|\boldsymbol{v}_j - A\boldsymbol{z}\|_2$. Here, it is solved roughly by using some iterative method. We call this inner iteration. Compared to the inner iteration, the outer solver, GMRES in this case, is called the outer iteration. In order to approximately and efficiently solve the inner least squares problems, we propose using simple stationary iterative methods implicitly related to the normal equation, i.e., without computing AA^T.

2.2. BA-GMRES method. Similar to Algorithm 1, next we consider applying inner iterations in the BA-GMRES method [18]. If $\mathcal{R}(A) = \mathcal{R}(B^T)$ and $\mathcal{R}(A^T) = \mathcal{R}(B)$ hold, from Theorem 2.4, then GMRES determines a solution of $\min_{\boldsymbol{x}\in\mathbf{R}^n} \|B\boldsymbol{b} - BA\boldsymbol{x}\|_2$, which is also a solution of $\min_{\boldsymbol{x}\in\mathbf{R}^n} \|\boldsymbol{b} - A\boldsymbol{x}\|_2$, without breakdown. The Krylov subspace at the jth step becomes $\mathcal{K}_j(BA, \tilde{\boldsymbol{r}}_0) = \text{span}\left\{\tilde{\boldsymbol{r}}_0, BA\tilde{\boldsymbol{r}}_0, \ldots, (BA)^{j-1}\tilde{\boldsymbol{r}}_0\right\}$, where $\tilde{\boldsymbol{r}}_0 = B\boldsymbol{r}_0$. The algorithm is given as follows.

Algorithm 2.2 BA-GMRES(l) method using inner iterations

1. *Let $\boldsymbol{x}_0$ be the initial approximate solution.*
2. *Roughly solve $A^TA\boldsymbol{z} = A^T\boldsymbol{r}_0 = A^T(\boldsymbol{b} - A\boldsymbol{x}_0)$ to obtain $\boldsymbol{z} \simeq \tilde{\boldsymbol{r}}_0 = B\boldsymbol{r}_0$ by using an inner iteration.*
3. *Compute $\beta = \|\tilde{\boldsymbol{r}}_0\|_2$, $\boldsymbol{v}_1 = \tilde{\boldsymbol{r}}_0/\beta$*

4. For $j = 1, 2, \ldots, l$, *Do*
5. Roughly solve $A^T A\boldsymbol{z} = A^T A\boldsymbol{v}_j$ *to obtain* $\boldsymbol{z} \simeq \boldsymbol{w}_j = BA\boldsymbol{v}_j$ *by using an inner iteration.*
6. For $i = 1, 2, \ldots, j$, *Do*
7. $h_{i,j} = (\boldsymbol{w}_j, \boldsymbol{v}_i)$
8. $\boldsymbol{w}_j = \boldsymbol{w}_j - h_{i,j}\boldsymbol{v}_i$
9. EndDo
10. $h_{j+1,j} = \|\boldsymbol{w}_j\|_2$
11. $\boldsymbol{v}_{j+1} = \boldsymbol{w}_j / h_{j+1,j}$
12. Find $\boldsymbol{y} \in \mathbf{R}^j$ *that minimizes* $\|\beta\boldsymbol{e}_1 - \bar{H}_j\boldsymbol{y}\|_2 = \|B\boldsymbol{r}_j\|_2$.
13. $\boldsymbol{x}_j = \boldsymbol{x}_0 + [\boldsymbol{v}_1, \boldsymbol{v}_2, \ldots, \boldsymbol{v}_j]\,\boldsymbol{y}_j$
14. If $\|A^T(\boldsymbol{b} - A\boldsymbol{x}_j)\|_2 < \varepsilon\|A^T\boldsymbol{r}_0\|_2$ *, then stop.*
15. EndDo
16. $\boldsymbol{x}_0 = \boldsymbol{x}_l$ *and go to 2*

The idea behind line 2 is as follows. First consider the problem

$$\min_{\boldsymbol{z}\in\mathbf{R}^n} \|\boldsymbol{r}_0 - A\boldsymbol{z}\|_2, \tag{2.6}$$

where $\boldsymbol{r}_0$ is given. This problem is equivalent to

$$A^T A\boldsymbol{z} = A^T\boldsymbol{r}_0, \quad \text{or} \quad A^T(\boldsymbol{r}_0 - A\boldsymbol{z}) = \boldsymbol{0}, \quad \text{or} \quad \boldsymbol{r}_0 - A\boldsymbol{z} \in \mathcal{N}(A^T) = \mathcal{R}(A)^\perp, \tag{2.7}$$

which, in turn, is equivalent to

$$BA\boldsymbol{z} = B\boldsymbol{r}_0, \quad \text{or} \quad B(\boldsymbol{r}_0 - A\boldsymbol{z}) = \boldsymbol{0}, \quad \text{or} \quad \boldsymbol{r}_0 - A\boldsymbol{z} \in \mathcal{N}(B) = \mathcal{R}(B^T)^\perp, \tag{2.8}$$

if $\mathcal{R}(A) = \mathcal{R}(B^T)$. Hence, if we roughly solve (2.6) to obtain $A^T A\tilde{\boldsymbol{z}} \simeq A^T\boldsymbol{r}_0$ and set $\tilde{\boldsymbol{z}} := B\boldsymbol{r}_0$, (2.6) gives

$$\|\boldsymbol{r}_0 - AB\boldsymbol{r}_0\|_2 = \|\boldsymbol{r}_0 - A\tilde{\boldsymbol{r}}_0\|_2 \simeq \min_{\boldsymbol{z}\in\mathbf{R}^n} \|\boldsymbol{r}_0 - A\boldsymbol{z}\|_2$$

so that B will serve as a preconditioner for solving (2.6)–(2.8).

The idea behind line 5 can be explained similarly by replacing $\boldsymbol{r}_0$ by $A\boldsymbol{v}_j$, where $\boldsymbol{v}_j$ is given.

When B is fixed for all outer iterations, the method minimizes $\|B\boldsymbol{r}_j\|_2$ over $\boldsymbol{x} \in \boldsymbol{x}_0 + \mathcal{K}_j(BA, \tilde{\boldsymbol{r}}_0)$, and is guaranteed to give a least squares solution if the conditions $\mathcal{R}(A) = \mathcal{R}(B^T)$ and $\mathcal{R}(A^T) = \mathcal{R}(B)$ of Theorem 2.4 are satisfied.

On the other hand, when B is not fixed for each outer iteration, the method tries to minimize $\|B_j\boldsymbol{r}_j\|_2$, where B_j is different for each outer iteration, and the approximate solution $\boldsymbol{x}$ is sought in

$$\boldsymbol{x}_0 + \operatorname{span}\{B_0\boldsymbol{r}_0, (B_1A)B_0\boldsymbol{r}_0, (B_2A)(B_1A)B_0\boldsymbol{r}_0, \ldots, (B_jA)(B_{j-1}A)\cdots B_0\boldsymbol{r}_0\},$$

which is no longer a Krylov subspace. In fact, numerical experiments showed that the method failed to converge when B was changed for each iteration by changing the number of inner iterations for each outer iteration.

2.3. Right-preconditioned CGNE method. Similarly, we can apply inner-iteration preconditioners to the CGNE method [28] as follows.

Algorithm 2.3 PCGNE method using inner iterations
1. Let $\boldsymbol{x}_0$ *be an initial vector, and compute* $\boldsymbol{r}_0 = \boldsymbol{b} - A\boldsymbol{x}_0$.

2. *Roughly solve* $AA^T\boldsymbol{z}_0 = \boldsymbol{r}_0$ *to obtain* $\tilde{\boldsymbol{z}}_0$ *by using an inner iteration.*
3. $\boldsymbol{q}_0 = \tilde{\boldsymbol{z}}_0, \gamma_0 = (\boldsymbol{r}_0, \tilde{\boldsymbol{z}}_0)$
4. *For* $j = 0, 1, 2, \dots$ *, Do*
5. $\quad \boldsymbol{s}_j = A^T\boldsymbol{q}_j$
6. $\quad \alpha_j = \gamma_j/(\boldsymbol{s}_j, \boldsymbol{s}_j)$
7. $\quad \boldsymbol{x}_{j+1} = \boldsymbol{x}_j + \alpha_j\boldsymbol{s}_j$
8. $\quad$ *If* $\|A^T(\boldsymbol{b} - A\boldsymbol{x}_{j+1})\|_2 < \varepsilon\|A^T\boldsymbol{r}_0\|_2$ *, then stop.*
9. $\quad \boldsymbol{r}_{j+1} = \boldsymbol{r}_j - \alpha_j A\boldsymbol{s}_j$
10. $\quad$ *Roughly solve* $AA^T\boldsymbol{z}_{j+1} = \boldsymbol{r}_{j+1}$ *to obtain* $\tilde{\boldsymbol{z}}_{j+1}$ *by using an inner iteration.*
11. $\quad \gamma_{j+1} = (\boldsymbol{r}_{j+1}, \tilde{\boldsymbol{z}}_{j+1})$
12. $\quad \beta_k = \gamma_{k+1}/\gamma_k$
13. $\quad \boldsymbol{q}_{j+1} = \tilde{\boldsymbol{z}}_{j+1} + \beta_j\boldsymbol{q}_j$
14. *EndDo*

In lines 2 and 10, the normal equations may be solved using an iterative method.

2.4. Left-preconditioned CGLS (CGNR) method. We also apply inner-iteration preconditioners to the CGLS method [28] as follows.

Algorithm 2.4 PCGLS method using inner iterations

1. *Let* $\boldsymbol{x}_0$ *be the initial solution, and compute* $\boldsymbol{r}_0 = \boldsymbol{b} - A\boldsymbol{x}_0$.
2. *Roughly solve* $A^TA\boldsymbol{z} = A^T\boldsymbol{r}_0$ *to obtain* $\boldsymbol{z} \simeq \boldsymbol{z}_0 = B\boldsymbol{r}_0$ *by using an inner iteration.*
3. $\boldsymbol{p}_0 = \boldsymbol{z}_0, \boldsymbol{s}_0 = A^T\boldsymbol{r}_0, \gamma_0 = (\boldsymbol{s}_0, \boldsymbol{z}_0)$
4. *For* $j = 0, 1, \dots$ *, Do*
5. $\quad \boldsymbol{q}_j = A\boldsymbol{p}_j$
6. $\quad \alpha_j = \gamma_j/(\boldsymbol{q}_j, \boldsymbol{q}_j)$
7. $\quad \boldsymbol{x}_{j+1} = \boldsymbol{x}_j + \alpha_j\boldsymbol{p}_j$
8. $\quad$ *If* $\|A^T(\boldsymbol{b} - A\boldsymbol{x}_{j+1})\|_2 < \varepsilon\|A^T\boldsymbol{r}_0\|_2$ *, then stop.*
9. $\quad \boldsymbol{r}_{j+1} = \boldsymbol{r}_j - \alpha_j\boldsymbol{q}_j$
10. $\quad \boldsymbol{s}_{j+1} = A^T\boldsymbol{r}_{j+1}$
11. $\quad$ *Roughly solve* $A^TA\boldsymbol{z} = A^T\boldsymbol{r}_{j+1}$ *to obtain* $\boldsymbol{z} \simeq \tilde{\boldsymbol{z}}_{j+1} = B\boldsymbol{r}_{j+1}$ *by using an inner-iteration.*
12. $\quad \gamma_{j+1} = (\boldsymbol{s}_{j+1}, \tilde{\boldsymbol{z}}_{j+1})$
13. $\quad \beta_j = \gamma_{j+1}/\gamma_j$
14. $\quad \boldsymbol{p}_{j+1} = \tilde{\boldsymbol{z}}_{j+1} + \beta_j\boldsymbol{p}_j$
15. *EndDo*

In lines 2 and 11, the normal equations may be solved roughly using an iterative method.

3. Inner-iteration preconditioners. In this section, we introduce several stationary iterative methods that can be used to perform the inner-iteration preconditioning for the outer solvers discussed in Section 2. In the following, we assume that A has no zero rows, or zero columns.

3.1. Jacobi-type iterations. Cimmino-NE and -NR methods are described here. They may be considered as kinds of Jacobi iterations.

3.1.1. Cimmino-NE method. The Cimmino-NE method [10], [28] for least squares problems solves the normal equation (1.3) iteratively. Let $\boldsymbol{\alpha}_i$ denote the ith column of the matrix A^T for $i = 1, 2, \dots, m$. At the kth step, the solution is updated by $\boldsymbol{u}^{(k+1)} = \boldsymbol{u}^{(k)} + \delta_i^{(k)}\boldsymbol{e}_i$, where $\boldsymbol{e}_i$ is the unit vector whose ith component is 1.

The scalar $\delta_i^{(k)}$ is determined as follows so that the ith component of the residual

becomes zero:

$$r_i^{(k)} = 0 \quad \Longleftrightarrow \quad \left(\boldsymbol{b} - AA^T\left(\boldsymbol{u}^{(k)} + \delta_i^{(k)}\boldsymbol{e}_i\right), \boldsymbol{e}_i\right) = 0 \quad \Longrightarrow \quad \delta_i^{(k)} = \frac{b_i - \left(\boldsymbol{\alpha}_i, \boldsymbol{x}^{(k)}\right)}{\|\boldsymbol{\alpha}_i\|_2{}^2},$$

when $\boldsymbol{b} \in \mathcal{R}(A)$, where $\boldsymbol{r}^{(k)} = \boldsymbol{b} - AA^T\boldsymbol{u}^{(k)}$ and $\boldsymbol{x}^{(k)} = A^T\boldsymbol{u}^{(k)}$. Suppose $\boldsymbol{\alpha}_i \neq \boldsymbol{0}$, $i = 1, 2, \ldots, m$. Thus, we have the following algorithm.

Algorithm 3.1 Cimmino-NE method

1. *Let* $\boldsymbol{x}^{(0)} \in \mathbf{R}^n$ *be the initial approximate solution.*
2. *For* $k = 0, 1, \ldots,$ *Do*
3. *For* $i = 1, 2, \ldots, m,$ *Do*
4. $\delta_i^{(k)} = \lambda(b_i - (\boldsymbol{\alpha}_i, \boldsymbol{x}^{(k)}))/\|\boldsymbol{\alpha}_i\|_2{}^2$
5. *EndDo*
6. $\boldsymbol{x}^{(k+1)} = \boldsymbol{x}^{(k)} + A^T\boldsymbol{\delta}^{(k)} \quad (\boldsymbol{u}^{(k+1)} = \boldsymbol{u}^{(k)} + \boldsymbol{\delta}^{(k)})$
7. *EndDo*

Here, λ is an acceleration parameter.

In Algorithm 3.1, we need only to store the vector $\boldsymbol{\delta} \in \mathbf{R}^m$ and the scalar $\lambda \in \mathbf{R}$ besides A, $\boldsymbol{x}$ and $\boldsymbol{b}$. One may also store $1/\|\boldsymbol{\alpha}_i\|_2{}^2$, $i = 1, 2, \ldots, m$, to reduce the computational cost.

Algorithm 3.1 with $\lambda = 1$ is equivalent to applying the Jacobi method to the second kind normal equation (1.3).

We analyse this method from the inner-iteration preconditioning point of view. The computations in lines 3–5 corresponds to the following:

$$\begin{aligned}\boldsymbol{\delta}^{(k)} &= \lambda\left[\frac{b_1 - (\boldsymbol{\alpha}_1, \boldsymbol{x}^{(k)})}{\|\boldsymbol{\alpha}_1\|_2{}^2}, \frac{b_2 - (\boldsymbol{\alpha}_2, \boldsymbol{x}^{(k)})}{\|\boldsymbol{\alpha}_2\|_2{}^2}, \ldots, \frac{b_m - (\boldsymbol{\alpha}_m, \boldsymbol{x}^{(k)})}{\|\boldsymbol{\alpha}_m\|_2{}^2}\right]^T \\ &= \lambda\begin{bmatrix} \frac{1}{\|\boldsymbol{\alpha}_1\|_2{}^2} & & & \\ & \frac{1}{\|\boldsymbol{\alpha}_2\|_2{}^2} & & \\ & & \ddots & \\ & & & \frac{1}{\|\boldsymbol{\alpha}_m\|_2{}^2} \end{bmatrix}\begin{bmatrix} b_1 - (\boldsymbol{\alpha}_1, \boldsymbol{x}^{(k)}) \\ b_2 - (\boldsymbol{\alpha}_2, \boldsymbol{x}^{(k)}) \\ \vdots \\ b_m - (\boldsymbol{\alpha}_m, \boldsymbol{x}^{(k)}) \end{bmatrix} = \lambda\Delta_m(\boldsymbol{b} - A\boldsymbol{x}^{(k)}),\end{aligned}$$

where $\Delta_m \equiv \operatorname{diag}\left(1/\|\boldsymbol{\alpha}_1\|_2{}^2, 1/\|\boldsymbol{\alpha}_2\|_2{}^2, \ldots, 1/\|\boldsymbol{\alpha}_m\|_2{}^2\right)$. By repeatedly using the relations $\boldsymbol{x}^{(k)} = \boldsymbol{x}^{(k-1)} + A^T\boldsymbol{\delta}^{(k-1)}$ and $\boldsymbol{\delta}^{(k)} = \lambda\Delta_m(\boldsymbol{b} - A\boldsymbol{x}^{(k)})$, we obtain

$$\boldsymbol{\delta}^{(k)} = (I_m - \lambda\Delta_m AA^T)\boldsymbol{\delta}^{(k-1)} = (I_m - \lambda\Delta_m AA^T)^k\boldsymbol{\delta}^{(0)},$$

where $I_m \in \mathbf{R}^{m\times m}$ is the identity matrix. Assume $\boldsymbol{x}^{(0)} = \boldsymbol{0}$. Then, we have $\boldsymbol{r}^{(0)} = \boldsymbol{b}$, which gives

$$\boldsymbol{\delta}^{(k)} = \lambda\Delta_m\left(I_m - \lambda AA^T\Delta_m\right)^k\boldsymbol{b}.$$

Finally, $\boldsymbol{x}^{(k+1)}$ can be expressed as

$$\begin{aligned}\boldsymbol{x}^{(k+1)} &= \boldsymbol{x}^{(k)} + A^T\boldsymbol{\delta}^{(k)} \\ &= \boldsymbol{x}^{(0)} + A^T\sum_{i=0}^{k}\boldsymbol{\delta}^{(i)} = B^{(k)}\boldsymbol{b},\end{aligned}$$

where

$$B^{(k)} \equiv \lambda A^T \Delta_m \sum_{i=0}^{k} \left(I_m - \lambda A A^T \Delta_m\right)^i, \tag{3.1}$$

which can be interpreted as the Cimmino-NE preconditioner for k iterations. Hence, this $B^{(k)}$ can act as B_j in the AB-GMRES(l) method of Algorithm 2.1.

Here, let

$$C^{(k)} \equiv \lambda \Delta_m \sum_{i=0}^{k} \left(I_m - \lambda A A^T \Delta_m\right)^i \tag{3.2}$$

so that (3.1) gives $B^{(k)} = A^T C^{(k)}$. Note that Δ_m is nonsingular. We omit k of $C^{(k)}$ for simplicity below.

Denote $\hat{A} = \Delta_m^{\frac{1}{2}} A$ and let $\hat{A} = \hat{U}\hat{\Sigma}\hat{V}^T$ be the singular value decomposition of $\hat{A}$. Then, we have

$$\begin{aligned} C &= \lambda \Delta_m^{\frac{1}{2}} \left[\sum_{i=0}^{k} \left(I_m - \lambda \Delta_m^{\frac{1}{2}} A A^T \Delta_m^{\frac{1}{2}}\right)^i\right] \Delta_m^{\frac{1}{2}} \\ &= \lambda \Delta_m^{\frac{1}{2}} \hat{U} \left[\sum_{i=0}^{k} \left(I_m - \lambda \hat{\Sigma}\hat{\Sigma}^T\right)^i\right] \hat{U}^T \Delta_m^{\frac{1}{2}} \\ &= \Delta_m^{\frac{1}{2}} \hat{U} \operatorname{diag}\left(\tau_{1,k}, \tau_{2,k}, \ldots, \tau_{r,k}, \lambda(k+1), \ldots, \lambda(k+1)\right) (\Delta_m^{\frac{1}{2}} \hat{U})^T, \end{aligned}$$

where

$$\tau_{l,k} \equiv \frac{1 - (1 - \lambda \hat{\sigma}_l^2)^{k+1}}{\hat{\sigma}_l^2},$$

$\hat{\sigma}_1 \geq \hat{\sigma}_2 \geq \cdots \geq \hat{\sigma}_r > 0$ are the singular values of $\hat{A}$, and $r = \operatorname{rank} A$. Hence, C is symmetric and congruent to $\operatorname{diag}(\tau_{1,k}, \tau_{2,k}, \ldots, \tau_{r,k}, \lambda(k+1), \ldots, \lambda(k+1))$. Hence, we have the following.

THEOREM 3.1.

$$\begin{aligned} \lambda < 0 &\implies C \text{ is negative definite}, \\ \lambda = 0 &\implies C = 0, \\ 0 < \lambda < \frac{2}{\hat{\sigma}_1^2} &\implies C \text{ is positive definite}, \\ \frac{2}{\hat{\sigma}_1^2} \leq \lambda &\implies \begin{cases} C \text{ is positive definite if } k \text{ is even}, \\ C \text{ is not definite if } k \text{ is odd}. \end{cases} \end{aligned}$$

THEOREM 3.2. *C is singular if and only if $\lambda = 0$, or k is odd and $\lambda = \frac{2}{\hat{\sigma}_i^2}$ for some i, $1 \leq i \leq r$.*

THEOREM 3.3. *Assume C is nonsingular and that the initial guess $\boldsymbol{x}^{(0)}$ for the Cimmino-NE inner-iteration preconditioning method is set to be zero. Then $\min_{\boldsymbol{x} \in \mathbf{R}^n} \|\boldsymbol{b} - A\boldsymbol{x}\|_2 = \min_{\boldsymbol{z} \in \mathbf{R}^m} \|\boldsymbol{b} - AB^{(k)}\boldsymbol{z}\|_2$ holds for all $\boldsymbol{b} \in \mathbf{R}^m$.*

Proof. $B^{(k)} = A^T C$ implies $\mathcal{R}(B^{(k)}) = \mathcal{R}(A^T)$. Since $\mathcal{R}(AA^T) = \mathcal{R}(A)$, $\mathcal{R}(A) = \mathcal{R}(AB^{(k)})$. From Theorem 2.1, the theorem holds. □

THEOREM 3.4. *Assume C is nonsingular,* $\operatorname{rank} A = m$*, and that the initial guess $\boldsymbol{x}^{(0)}$ for the Cimmino-NE inner-iteration preconditioning method is set to be zero. Then, with this inner-iteration preconditioning, AB-GMRES determines a least squares solution of $\min_{\boldsymbol{x}\in\mathbf{R}^n} \|\boldsymbol{b} - A\boldsymbol{x}\|_2$ for all $\boldsymbol{b} \in \mathbf{R}^m$ and for all $\boldsymbol{x}_0 \in \mathbf{R}^n$ without breakdown.*

Proof. $B^{(k)} = A^T C$ gives $\mathcal{R}(B^{(k)}) = \mathcal{R}(A^T)$. $m = \operatorname{rank} A = \operatorname{rank} A^T = \operatorname{rank} B^{(k)} = \operatorname{rank}\left(B^{(k)}\right)^T$ gives $\mathcal{R}\left(\left(B^{(k)}\right)^T\right) = \mathcal{R}(A) = \mathbf{R}^m$. Hence, from Theorem 2.3, the theorem holds. □

Note also that since C is symmetric, we may use it to precondition (1.3) for the CGNE method if C is also definite.

3.1.2. Cimmino-NR method. The Cimmino-NR method [28] solves the normal equation (1.2) iteratively. Let d_i be the difference between the ith component of the current solution and the previous solution. The kth Jacobi iteration applied to (1.2) gives the following relation for the ith component

$$(A^T\boldsymbol{b} - A^T A(\boldsymbol{x}^{(k)} + d_i^{(k)}\boldsymbol{e}_i), \boldsymbol{e}_i) = 0 \quad \Longrightarrow \quad d_i^{(k)} = \frac{(\boldsymbol{r}^{(k)}, \boldsymbol{a}_i)}{\|\boldsymbol{a}_i\|_2^{\,2}},$$

where $\boldsymbol{x}^{(k+1)} = \boldsymbol{x}^{(k)} + d_i^{(k)}\boldsymbol{e}_i$ and $\boldsymbol{a}_i$ denotes the ith column of A. Suppose $\boldsymbol{a}_i \neq \boldsymbol{0}, i = 1, 2, \ldots, n$.

Adding an acceleration parameter λ, we obtain the following algorithm [28].

Algorithm 3.2 Cimmino-NR method

1. *Let $\boldsymbol{x}^{(0)} \in \mathbf{R}^n$ be the initial approximate solution.*
2. *Compute $\boldsymbol{r}^{(0)} = \boldsymbol{b} - A\boldsymbol{x}^{(0)}$*
3. *For $k = 0, 1, \ldots,$ Do*
4. *For $i = 1, 2, \ldots, n,$ Do*
5. $d_i^{(k)} = \lambda(\boldsymbol{r}^{(k)}, \boldsymbol{a}_i)/\|\boldsymbol{a}_i\|_2^{\,2}$
6. *EndDo*
7. $\boldsymbol{x}^{(k+1)} = \boldsymbol{x}^{(k)} + \boldsymbol{d}^{(k)}$
8. $\boldsymbol{r}^{(k+1)} = \boldsymbol{r}^{(k)} - A\boldsymbol{d}^{(k)}$
9. *EndDo*

Here, λ is an acceleration parameter. At the last loop for k, $\boldsymbol{r}$ does not have to be updated, since it does not affect the final solution. The memory requirement for the right-hand side vector $\boldsymbol{b}$ can be replaced by the residual $\boldsymbol{r}$. Only the vector $\boldsymbol{d} \in \mathbf{R}^n$ needs to be stored besides A, $\boldsymbol{x}$ and $\boldsymbol{b}$.

Algorithm 3.2 with $\lambda = 1$ is equivalent to applying the Jacobi method to the normal equation (1.2).

The preconditioning matrix of the Cimmino-NR method can be derived in a similar way to that of the Cimmino-NE method. The computations in lines 4-6 at the kth step in Algorithm 3.2 correspond to the following:

$$d_i^{(k)} = \lambda\frac{(\boldsymbol{r}^{(k)}, \boldsymbol{a}_i)}{\|\boldsymbol{a}_i\|_2^{\,2}} \text{ for } i = 1, 2, \ldots, n$$

$$\Longleftrightarrow \quad \boldsymbol{d}^{(k)} = \lambda\left[\frac{(\boldsymbol{r}^{(k)}, \boldsymbol{a}_1)}{\|\boldsymbol{a}_1\|_2^{\,2}}, \frac{(\boldsymbol{r}^{(k)}, \boldsymbol{a}_2)}{\|\boldsymbol{a}_2\|_2^{\,2}}, \ldots, \frac{(\boldsymbol{r}^{(k)}, \boldsymbol{a}_n)}{\|\boldsymbol{a}_n\|_2^{\,2}}\right]^T.$$

Define $D_n \equiv \operatorname{diag}\left(1/\|\boldsymbol{a}_1\|_2^{\,2}, 1/\|\boldsymbol{a}_2\|_2^{\,2}, \ldots, 1/\|\boldsymbol{a}_n\|_2^{\,2}\right)$. Then, $\boldsymbol{d}^{(k)} = \lambda D_n A^T \boldsymbol{r}^{(k)}$. Let $\boldsymbol{x}^{(0)} = \boldsymbol{0}$. Applying the relations $\boldsymbol{r}^{(k)} = \boldsymbol{r}^{(k-1)} - A\boldsymbol{d}^{(k-1)}$ and $\boldsymbol{d}^{(k)} = \lambda D_n A^T \boldsymbol{r}^{(k)}$

repeatedly, we obtain

$$\begin{aligned}\boldsymbol{d}^{(k)} &= \lambda D_n A^T (I_m - \lambda A D_n A^T)^1 \boldsymbol{r}^{(k-1)} \\ &= \lambda D_n A^T (I_m - \lambda A D_n A^T)^k \boldsymbol{r}^{(0)} \\ &= \lambda (I_n - \lambda D_n A^T A)^k D_n A^T \boldsymbol{b},\end{aligned}$$

where $I_n \in \mathbf{R}^{n\times n}$ is the identity matrix. Finally, $\boldsymbol{x}^{(k+1)}$ is represented by using A and $\boldsymbol{b}$ as

$$\boldsymbol{x}^{(k+1)} = \boldsymbol{x}^{(k)} + \boldsymbol{d}^{(k)} = \boldsymbol{x}^{(0)} + \sum_{i=0}^{k} \boldsymbol{d}^{(i)}.$$

Let $B^{(k)}$ be

$$B^{(k)} \equiv \lambda \left[\sum_{i=0}^{k} (I_n - \lambda D_n A^T A)^i \right] D_n A^T. \tag{3.3}$$

Then $\boldsymbol{x}^{(k+1)} = B^{(k)}\boldsymbol{b}$ for $\boldsymbol{x}^{(0)} = \mathbf{0}$, and $B^{(k)}$ can be considered as the Cimmino-NR preconditioner for k inner iterations. Hence, this $B^{(k)}$ can act as B in the BA-GMRES(l) method of Algorithm 2.2.

Here, let

$$C^{(k)} \equiv \lambda \left[\sum_{i=0}^{k} (I_n - \lambda D_n A^T A)^i \right] D_n \tag{3.4}$$

so that (3.3) gives $B^{(k)} = C^{(k)} A^T$. Note that D_n is nonsingular. We omit k of $C^{(k)}$ below.

Denote $\check{A} = A D_n{}^{\frac{1}{2}}$ and let $\check{A} = \check{U}\check{\Sigma}\check{V}^T$ be the singular value decomposition of $\check{A}$. Then, we have

$$\begin{aligned}C &= \lambda D_n{}^{\frac{1}{2}} \left[\sum_{i=0}^{k} (I_n - \lambda D_n{}^{\frac{1}{2}} A^T A D_n{}^{\frac{1}{2}})^i \right] D_n{}^{\frac{1}{2}} \\ &= \lambda D_n{}^{\frac{1}{2}} \check{V} \left[\sum_{i=0}^{k} (I_n - \lambda \check{\Sigma}^T \check{\Sigma})^i \right] \check{V}^T D_n{}^{\frac{1}{2}} \\ &= D_n{}^{\frac{1}{2}} \check{V} \operatorname{diag}\left(\rho_{1,k}, \rho_{2,k}, \ldots, \rho_{r,k}, \lambda(k+1), \ldots, \lambda(k+1)\right) (D_n{}^{\frac{1}{2}} \check{V})^T,\end{aligned}$$

where

$$\rho_{l,k} \equiv \frac{1 - (1 - \lambda \check{\sigma}_l^2)^{k+1}}{\check{\sigma}_l^2},$$

$\check{\sigma}_1 \geq \check{\sigma}_2 \geq \cdots \geq \check{\sigma}_r > 0$ are the singular values of $\check{A}$, and $r = \operatorname{rank} A$. Hence, C is symmetric and congruent to $\operatorname{diag}(\rho_{1,k}, \rho_{2,k}, \ldots, \rho_{r,k}, \lambda(k+1), \ldots, \lambda(k+1))$. Hence, we have the following.

THEOREM 3.5.

$$\begin{aligned}
\lambda < 0 &\implies C \text{ is negative definite,}\\
\lambda = 0 &\implies C = 0,\\
0 < \lambda < \frac{2}{\check{\sigma}_1^2} &\implies C \text{ is positive definite,}\\
\frac{2}{\check{\sigma}_1^2} \le \lambda &\implies \begin{cases} C \text{ is positive definite if } k \text{ is even,}\\ C \text{ is not definite if } k \text{ is odd.}\end{cases}
\end{aligned}$$

THEOREM 3.6. *C is singular if and only if $\lambda = 0$, or k is odd and $\lambda = \frac{2}{\check{\sigma}_i^2}$ for some i, $1 \le i \le r$.*

THEOREM 3.7. *Assume C is nonsingular and that the initial guess $\boldsymbol{x}^{(0)}$ for the Cimmino-NR inner-iteration preconditioning method is set to be zero. Then $\min_{\boldsymbol{x}\in\mathbf{R}^n} \|\boldsymbol{b} - A\boldsymbol{x}\|_2$ and $\min_{\boldsymbol{x}\in\mathbf{R}^n} \|B^{(k)}\boldsymbol{b} - B^{(k)}A\boldsymbol{x}\|_2$ are equivalent for all $\boldsymbol{b} \in \mathbf{R}^m$.*

Proof. $B^{(k)} = CA^T$ implies $\mathcal{R}\left(\left(B^{(k)}\right)^T\right) = \mathcal{R}(A)$. Since $\mathcal{R}\left(\left(B^{(k)}\right)^T\right) = \mathcal{R}(A) \Longrightarrow \mathcal{R}(A) = \mathcal{R}\left(\left(B^{(k)}\right)^T B^{(k)}A\right)$ [18] and from Theorem 2.2, the theorem holds. □

THEOREM 3.8. *Assume C is nonsingular,* $\operatorname{rank} A = n$, *and that the initial guess $\boldsymbol{x}^{(0)}$ for the Cimmino-NR inner-iteration preconditioning method is set to be zero. Then with this inner-iteration preconditioning, BA-GMRES determines a least squares solution of $\min_{\boldsymbol{x}\in\mathbf{R}^n} \|\boldsymbol{b} - A\boldsymbol{x}\|_2$ for all $\boldsymbol{b} \in \mathbf{R}^m$ and for all $\boldsymbol{x}_0 \in \mathbf{R}^n$ without breakdown.*

Proof. $B^{(k)} = CA^T$ gives $\mathcal{R}\left(\left(B^{(k)}\right)^T\right) = \mathcal{R}(A)$. $n = \operatorname{rank} A = \operatorname{rank} A^T = \operatorname{rank}\left(B^{(k)}\right)^T = \operatorname{rank} B^{(k)}$ gives $\mathcal{R}(B^{(k)}) = \mathcal{R}(A^T) = \mathbf{R}^n$. Hence, from Theorem 2.4, the theorem holds. □

Note also that since C is symmetric, we may use it to precondition (1.2) for the CGLS method if C is also definite.

One advantage of the Jacobi iterations is that the vectors $\boldsymbol{d}^{(k)}$, $\boldsymbol{r}^{(k)}$, $\boldsymbol{x}^{(k)}$, and, $\boldsymbol{\delta}^{(k)}$ can be computed in parallel.

3.2. SOR-type iterations. This subsection provides an overview of the normal error SOR and normal residual SOR methods.

3.2.1. Normal error SOR method. The NE-SOR method [28] solves (1.3) iteratively. Let $\boldsymbol{\alpha}_i$ be the ith column of matrix A^T, $i = 1, 2, \ldots, m$. Suppose $\boldsymbol{\alpha}_i \neq \boldsymbol{0}$, $i = 1, 2, \ldots, m$.

Algorithm 3.3 NE-SOR method

1. Let $\boldsymbol{x}^{(0)} \in \mathbf{R}^n$ be the the initial approximate solution.
2. For $k = 0, 1, \ldots,$ Do
3. For $i = 1, 2, \ldots, m$, Do
4. $\delta_i^{(k)} = \omega\left[b_i - (\boldsymbol{\alpha}_i, \boldsymbol{x}^{(k)})\right] / \|\boldsymbol{\alpha}_i\|_2^{\,2}$
5. $\boldsymbol{x}^{(k)} = \boldsymbol{x}^{(k)} + \delta_i^{(k)}\boldsymbol{\alpha}_i \qquad (u_i^{(k+1)} = u_i^{(k)} + \delta_i^{(k)})$
6. EndDo
7. $\boldsymbol{x}^{(k+1)} = \boldsymbol{x}^{(k)}$
8. EndDo

Here, ω is an acceleration parameter.

Similar to Algorithm 3.1, Algorithm 3.3 can be derived by solving for $\boldsymbol{u}$ in (1.3) by performing the SOR-type iteration, supposing $\boldsymbol{b} \in \mathcal{R}(A)$. The iterative solution $\boldsymbol{x}^{(k)}$ is updated component-wise. This method is also known as the Kaczmarz's method [22], [32] or the row-action method [9].

In the following, we derive the preconditioning matrix corresponding to this method, similarly to [32]. Line 5 in Algorithm 3.3 can be expressed as

$$\boldsymbol{x} = P_i \boldsymbol{x} + \omega \frac{b_i}{{\|\boldsymbol{\alpha}_i\|_2}^2} \boldsymbol{\alpha}_i, \quad \text{where} \quad P_i \equiv I_n - \omega \frac{1}{{\|\boldsymbol{\alpha}_i\|_2}^2} \boldsymbol{\alpha}_i {\boldsymbol{\alpha}_i}^T, \quad i = 1, 2, \ldots, m.$$

Then, the kth step is given by

$$\boldsymbol{x}^{(k+1)} = P_m \left(\cdots \left(P_2 \left(P_1 \boldsymbol{x}^{(k)} + \omega \frac{b_1 \boldsymbol{\alpha}_1}{{\|\boldsymbol{\alpha}_1\|_2}^2} \right) + \omega \frac{b_2 \boldsymbol{\alpha}_2}{{\|\boldsymbol{\alpha}_2\|_2}^2} \right) \cdots \right) + \omega \frac{b_m \boldsymbol{\alpha}_m}{{\|\boldsymbol{\alpha}_m\|_2}^2}.$$

Let Q_i be

$$Q_i \equiv \begin{cases} P_m P_{m-1} \cdots P_i, & \text{if } i = 1, 2, \ldots, m, \\ I_n, & \text{if } i = m+1. \end{cases}$$

Define $R \equiv [Q_2 \boldsymbol{\alpha}_1, Q_3 \boldsymbol{\alpha}_2, \ldots, Q_{m+1} \boldsymbol{\alpha}_m]$,

$$\Delta_m \equiv \operatorname{diag} \left(\frac{1}{{\|\boldsymbol{\alpha}_1\|_2}^2}, \frac{1}{{\|\boldsymbol{\alpha}_2\|_2}^2}, \ldots, \frac{1}{{\|\boldsymbol{\alpha}_m\|_2}^2} \right),$$

$\bar{R} \equiv R \Delta_m$, and $Q \equiv Q_1$. Then, if the initial solution $\boldsymbol{x}^{(0)} = \boldsymbol{0}$, we have

$$\begin{aligned} \boldsymbol{x}^{(k+1)} &= Q \boldsymbol{x}^{(k)} + \sum_{i=1}^{m} \omega \frac{b_i}{{\|\boldsymbol{\alpha}_i\|_2}^2} Q_{i+1} \boldsymbol{\alpha}_i = Q \boldsymbol{x}^{(k)} + \omega \bar{R} \boldsymbol{b} \\ &= Q^{k+1} \boldsymbol{x}^{(0)} + \omega \left(\sum_{l=0}^{k} Q^l \right) \bar{R} \boldsymbol{b} = \omega \left(\sum_{l=0}^{k} Q^l \right) \bar{R} \boldsymbol{b}. \end{aligned}$$

Let $B^{(k)}$ be

$$B^{(k)} \equiv \omega \left(\sum_{l=0}^{k} Q^l \right) \bar{R},$$

which can be considered as the preconditioning matrix corresponding to Algorithm 3.3 with k iterations. Hence, this $B^{(k)}$ can act as B_j in the AB-GMRES(l) method of Algorithm 2.1. Here,

$$\bar{R} = \left[c_{11} \boldsymbol{\alpha}_1 + c_{12} \boldsymbol{\alpha}_2 + \cdots + c_{1m} \boldsymbol{\alpha}_m, c_{22} \boldsymbol{\alpha}_2 + c_{23} \boldsymbol{\alpha}_3 + \cdots + c_{2m} \boldsymbol{\alpha}_m, \ldots, c_{mm} \boldsymbol{\alpha}_m \right].$$

Noting that

$$\begin{aligned} Q \boldsymbol{\alpha}_i &= P_m P_{m-1} \cdots P_1 \boldsymbol{\alpha}_i \\ &= c'_1 \boldsymbol{\alpha}_1 + c'_2 \boldsymbol{\alpha}_2 + \cdots + c'_m \boldsymbol{\alpha}_m, \end{aligned}$$

generically

$$\mathcal{R}(B^{(k)}) = \operatorname{span} \{ \boldsymbol{\alpha}_1, \boldsymbol{\alpha}_2, \ldots, \boldsymbol{\alpha}_m \} = \mathcal{R}(A^T),$$

holds, provided $\omega \neq 0$. Hence, from Theorem 2.1, $\min_{\boldsymbol{x}\in\mathbf{R}^n} \|\boldsymbol{b} - A\boldsymbol{x}\|_2 = \min_{\boldsymbol{z}\in\mathbf{R}^m} \|\boldsymbol{b} - AB^{(k)}\boldsymbol{z}\|_2$ generically holds for $B^{(k)}$ of NR-SOR and $\boldsymbol{x}^{(0)} = \boldsymbol{0}$.

On the other hand,

$$\left(B^{(k)}\right)^T = \omega \begin{bmatrix} \frac{1}{\|\boldsymbol{\alpha}_1\|_2}{\boldsymbol{\alpha}_1}^T {Q_2}^T \\ \frac{1}{\|\boldsymbol{\alpha}_2\|_2}{\boldsymbol{\alpha}_2}^T {Q_3}^T \\ \vdots \\ \frac{1}{\|\boldsymbol{\alpha}_m\|_2}{\boldsymbol{\alpha}_m}^T {Q_m}^T \end{bmatrix} \sum_{l=0}^{k} \left(Q^T\right)^l,$$

and, if Q does not have an eigenvalue equal to 1, $\sum_{l=0}^{k} {Q^T}^l = (I-Q)^{-1}(I-Q^{k+1})$. Hence, if $\operatorname{rank} A = m$, and if Q^{k+1} does not have an eigenvalue equal to 1, we may expect that

$$\mathcal{R}\left(\left(B^{(k)}\right)^T\right) = \mathcal{R}(A) = \mathbf{R}^m$$

holds, generically. Hence, from Theorem 2.3, AB-GMRES with inner iteration of NE-SOR with $\boldsymbol{x}^{(0)} = \boldsymbol{0}$ and $\omega \neq 0$ can determine a least squares solution of $\min_{\boldsymbol{x}\in\mathbf{R}^n} \|\boldsymbol{b} - A\boldsymbol{x}\|_2$ for all $\boldsymbol{b} \in \mathbf{R}^m$ and for all $\boldsymbol{x}_0 \in \mathbf{R}^n$ without breakdown for underdetermined full rank problems.

3.2.2. Normal residual SOR method. The NR-SOR method [28] solves (1.2) iteratively. Let $\boldsymbol{a}_i$ be the ith column of matrix A, $i = 1, 2, \ldots, n$. Suppose $\boldsymbol{a}_i \neq \boldsymbol{0}, i = 1, 2, \ldots, n$.

Algorithm 3.4 NR-SOR method

1. Let $\boldsymbol{x}^{(0)} \in \mathbf{R}^n$ *be the initial approximate solution.*
2. Compute $\boldsymbol{r} = \boldsymbol{b} - A\boldsymbol{x}^{(0)}$
3. For $k = 0, 1, \ldots,$ *Do*
4. For $i = 1, 2, \ldots, n,$ *Do*
5. $\delta_i^{(k)} = \omega(\boldsymbol{r}, \boldsymbol{a}_i)/{\|\boldsymbol{a}_i\|_2}^2$
6. $x_i^{(k+1)} = x_i^{(k)} + \delta_i^{(k)}$
7. $\boldsymbol{r} = \boldsymbol{r} - \delta_i^{(k)}\boldsymbol{a}_i$
8. EndDo
9. EndDo

Here, ω is an acceleration parameter.

The right-hand side vector $\boldsymbol{b}$ can be overwritten by the residual $\boldsymbol{r}$. Hence, the storage requirement is only for the scalar δ_i, where δ_i can also be overwritten for each i.

Next, we derive the preconditioning matrix corresponding to this iterative method, similarly to [32]. The residual vector is updated for $i = 1, 2, \ldots, n$ as follows:

$$\boldsymbol{r} = S_i\boldsymbol{r}, \quad \text{where} \quad S_i = I_m - \omega\frac{1}{{\|\boldsymbol{a}_i\|_2}^2}\boldsymbol{a}_i{\boldsymbol{a}_i}^T.$$

Let T_i be

$$T_i \equiv \begin{cases} S_iS_{i-1}\cdots S_1 & \text{if } i = 1, 2, \ldots, n, \\ I_m & \text{if } i = 0. \end{cases}$$

Define $W \equiv \left[{T_0}^T \boldsymbol{a}_1, {T_1}^T \boldsymbol{a}_2, \ldots, {T_{n-1}}^T \boldsymbol{a}_n\right]^T$,

$$D_n \equiv \mathrm{diag}\left(\frac{1}{{\|\boldsymbol{a}_1\|_2}^2}, \frac{1}{{\|\boldsymbol{a}_2\|_2}^2}, \ldots, \frac{1}{{\|\boldsymbol{a}_n\|_2}^2}\right),$$

$\bar{W} \equiv D_n W$, and $T \equiv T_n$. Then, we have

$$\begin{aligned}\boldsymbol{x}^{(k+1)} &= \boldsymbol{x}^{(k)} + \omega \bar{W} \boldsymbol{r}^{(k)} \\ &= \boldsymbol{x}^{(0)} + \omega \bar{W} \sum_{l=0}^{k} \boldsymbol{r}^{(l)} \\ &= \boldsymbol{x}^{(0)} + \omega \bar{W} \left(\sum_{l=0}^{k} T^l\right) \boldsymbol{b}.\end{aligned}$$

Thus, if $\boldsymbol{x}^{(0)} = \boldsymbol{0}$, the preconditioning matrix is given by

$$B^{(k)} \equiv \omega \bar{W} \sum_{l=0}^{k} T^l,$$

which can be considered as a preconditioner for BA-GMRES. Hence,

$$\left(B^{(k)}\right)^T = \omega \left(\sum_{l=0}^{k} \left(T^T\right)^l\right) \bar{W}^T,$$

where

$$\begin{aligned}\bar{W}^T &= \left[\frac{1}{{\|\boldsymbol{a}_1\|_2}^2} {T_0}^T \boldsymbol{a}_1, \frac{1}{{\|\boldsymbol{a}_2\|_2}^2} {T_1}^T \boldsymbol{a}_2, \ldots, \frac{1}{{\|\boldsymbol{a}_n\|_2}^2} {T_{n-1}}^T \boldsymbol{a}_n\right] \\ &= \left[c_{11}\boldsymbol{a}_1, c_{21}\boldsymbol{a}_1 + c_{22}\boldsymbol{a}_2, \ldots, c_{n1}\boldsymbol{a}_1 + c_{n2}\boldsymbol{a}_2 + \cdots + c_{nn}\boldsymbol{a}_n\right],\end{aligned}$$

since

$$S_i = I_m + c_i \boldsymbol{a}_i {\boldsymbol{a}_i}^T, \quad {S_i}^T = S_i, \quad \text{and} \quad {T_i}^T = S_1 S_2 \cdots S_i.$$

Hence,

$$\left(B^{(k)}\right)^T = \omega \left(\sum_{l=0}^{k} \left(T^T\right)^l\right) \left[c_{11}\boldsymbol{a}_1, c_{21}\boldsymbol{a}_1 + c_{22}\boldsymbol{a}_2, \ldots, c_{n1}\boldsymbol{a}_1 + c_{n2}\boldsymbol{a}_2 + \cdots + c_{nn}\boldsymbol{a}_n\right].$$

Noting that

$$T^T \boldsymbol{a}_j = S_1 S_2 \cdots S_n \boldsymbol{a}_j = {c_1}' \boldsymbol{a}_1 + {c_2}' \boldsymbol{a}_2 + \cdots + {c_n}' \boldsymbol{a}_n,$$

generically

$$\mathcal{R}\left(\left(B^{(k)}\right)^T\right) = \mathrm{span}\left\{\boldsymbol{a}_1, \boldsymbol{a}_2, \ldots, \boldsymbol{a}_n\right\} = \mathcal{R}(A)$$

holds, provided $\omega \neq 0$. Hence, from Theorem 2.2, $\min_{\boldsymbol{x} \in \mathbf{R}^n} \|\boldsymbol{b} - A\boldsymbol{x}\|_2$ and $\min_{\boldsymbol{x} \in \mathbf{R}^n} \|B^{(k)}\boldsymbol{b} - B^{(k)}A\boldsymbol{x}\|_2$ are generically equivalent for $B^{(k)}$ of NR-SOR and $\boldsymbol{x}^{(0)} = \boldsymbol{0}$.

On the other hand,

$$B^{(k)} = \omega \begin{bmatrix} \frac{1}{\|\boldsymbol{a}_1\|_2}\boldsymbol{a}_1{}^T T_0 \\ \frac{1}{\|\boldsymbol{a}_2\|_2}\boldsymbol{a}_2{}^T T_1 \\ \vdots \\ \frac{1}{\|\boldsymbol{a}_n\|_2}\boldsymbol{a}_n{}^T T_{n-1} \end{bmatrix} \sum_{l=0}^{k} T^l,$$

and, if T does not have an eigenvalue equal to 1, $\sum_{l=0}^{k} T^l = (I-T)^{-1}\left(I - T^{k+1}\right)$. Hence, if $rankA = n$, and if T^{k+1} does not have an eigenvalue equal to 1, we may expect that

$$\mathcal{R}(B^{(k)}) = \mathcal{R}(A^T) = \mathbf{R}^n$$

holds generically. Hence, from Theorem 2.4, BA-GMRES with inner iteration of NR-SOR with $\boldsymbol{x}^{(0)} = \boldsymbol{0}$ and $\omega \neq 0$ can determine a least squares solution of $\min_{\boldsymbol{x}\in\mathbf{R}^n} \|\boldsymbol{b} - A\boldsymbol{x}\|_2$ for all $\boldsymbol{b} \in \mathbf{R}^m$ and for all $\boldsymbol{x}_0 \in \mathbf{R}^n$ without breakdown for overdetermined full rank problems.

3.2.3. Normal error and normal residual SSOR method. In order to precondition the CGNE and CGLS methods, a symmetric preconditioner is required. If lines 3–6 in Algorithm 3.3 and lines 4–8 in Algorithm 3.4 are updated in the forward order, $i = 1, 2, \ldots, m(n)$, and then in the reverse order, $i = m(n), m-1(n-1), \ldots, 1$, we obtain symmetric variants of NE- and NR-SOR, i.e., the NE-SSOR and NR-SSOR methods, respectively. These variants, when applied to the CGNE and CGNR methods with only one inner iteration, are discussed in [8], [28, Chapter 10].

Algorithm 3.5 gives the detail of the NR-SSOR method. Let $\boldsymbol{a}_i$ be the ith column of matrix A, $i = 1, 2, \ldots, n$. Suppose $\boldsymbol{a}_i \neq 0$, $i = 1, 2, \ldots, n$.

Algorithm 3.5 NR-SSOR method

1. *Let* $\boldsymbol{x}^{(0)} \in \mathbf{R}^n$ *be the initial approximate solution.*
2. *Compute* $\boldsymbol{r} = \boldsymbol{b} - A\boldsymbol{x}^{(0)}$
3. *For* $k = 0, 1, 2, \ldots,$ *Do*
4. *For* $i = 1, 2, \ldots, n,$ *Do*
5. $\delta_i^{(k)} = \omega(\boldsymbol{r}, \boldsymbol{a}_i)/\|\boldsymbol{a}_i\|_2{}^2$
6. $x_i^{(k+\frac{1}{2})} = x_i^{(k)} + \delta_i^{(k)}$
7. $\boldsymbol{r} = \boldsymbol{r} - \delta_i^{(k)}\boldsymbol{a}_i$
8. *EndDo*
9. *For* $i = n, n-1, \ldots, 1,$ *Do*
10. $\delta_i^{(k+\frac{1}{2})} = \omega(\boldsymbol{r}, \boldsymbol{a}_i)/\|\boldsymbol{a}_i\|_2{}^2$
11. $x_i^{(k+1)} = x_i^{(k+\frac{1}{2})} + \delta_i^{(k+\frac{1}{2})}$
12. $\boldsymbol{r} = \boldsymbol{r} - \delta_i^{(k+\frac{1}{2})}\boldsymbol{a}_i$
13. *EndDo*
14. *EndDo*

Here, ω is an acceleration parameter.

The algorithm for the NE-SSOR method can be given similarly.

The computational work for one iteration and memory requirement, for Cimmino-NE/NR, and NE/NR-SOR, and NE/NR-SSOR, are summarized in Tables 3.1 and 3.2, respectively. "MV" in Table 3.1 denotes the computational cost required for

TABLE 3.1
Number of operations required for one inner iteration.

	Cimmino-NE	Cimmino-NR	NE-SOR	NR-SOR	NE-SSOR	NR-SSOR
Number of op.	2MV+$3m+n$	2MV+$m+3n$	2MV+$3m$	2MV+$3n$	4MV+$6m$	4MV+$6n$

TABLE 3.2
Storage requirement in addition to A, $\boldsymbol{b}$, and $\boldsymbol{x}$.

	Cimmino-NE	Cimmino-NR	NE-SOR	NR-SOR	NE-SSOR	NR-SSOR
Storage requirement	m	$m+n$	1	m	1	m

one matrix-vector multiplication. Note that we assume that $\|\boldsymbol{\alpha}_i\|_2{}^2$ and $\|\boldsymbol{a}_j\|_2{}^2$ are computed beforehand and stored.

Table 3.1 shows that the computational work for one inner iteration is roughly proportional to the cost of the matrix-vector multiplications with A, and the work for NE/NR-SSOR is twice that for NE/NR-SOR. Table 3.2 shows that the inner iterations can be carried out by storing a few vectors only. Note that matrix decomposition-type preconditioners for iterative methods for solving least squares problems requires memory comparable to the size of A, e.g., [4], [5].

4. Numerical experiments. We tested the preconditioning methods that were discussed in Sections 2 and 3. We will apply our proposed methods to CG- and GMRES-based methods and compare them with the diagonal scaling and the RIF preconditioners.[1] Note that, theoretically, the RIF preconditioner may break down for rank-deficient problems.

The stopping criteria for the ith outer iteration was

$$\left\|A^T(\boldsymbol{b}-A\boldsymbol{x}_i)\right\|_2 < \varepsilon \left\|A^T\boldsymbol{r}_0\right\|_2 . \tag{4.1}$$

The left-hand side of (4.1) can converge to zero because of the equivalence between (1.1) and (1.2). This means that we explicitly compute the relative residual. In the numerical experiments, the CPU time for checking (4.1) was excluded from the total CPU time. The initial solutions for the inner iterations and the outer iterations were always set to $\boldsymbol{x}_0 := \boldsymbol{0}$. No restarts were used for GMRES.

In this paper, overdetermined problems ($m > n$) are tested. All zero columns and zero rows of the test matrices were deleted in advance. The elements of $\boldsymbol{b}$ were randomly generated using the Fortran built-in subroutine `random_number`. Therefore, the test problems were not necessarily consistent, i.e., $\boldsymbol{b}$ may not be in $\mathcal{R}(A)$. For overdetermined problems, BA-GMRES is computationally more efficient compared to AB-GMRES since the former works in a smaller dimensional ($n < m$) space [18]. Similarly, PCGLS is more efficient compared to PCGNE.

All computations were done on a PC workstation with an Intel Xeon X5492 3.4 GHz CPU, 16 GB RAM, and Scientific Linux 5.4. The experiments were done in double precision floating-point arithmetic. All programs for the iterative methods used for our tests were coded in Fortran 95 and compiled by Intel Fortran Version 11.1. For the direct methods, we used MATLAB 2010a.

4.1. Effect of condition number. Test matrices RANDLn, $n = 1, 2, \ldots, 7$, were randomly generated using the MATLAB function `sprandn`, as in [18]. Table 4.1 shows the condition numbers $\kappa(A)$ of the matrices. They all have a nonzero density of

[1] The RIF code developed by Professors Michele Benzi and Miroslav Tůma, available online at `http://www2.cs.cas.cz/~tuma/sparslab.html`, was implemented.

TABLE 4.1
Information on the random test matrices.

Name	$\kappa(A)$	Time	$\|A^T\boldsymbol{r}\|_2/\|A^T\boldsymbol{b}\|_2$
RANDL1	1.9×10	13.44	5.59×10^{-15}
RANDL2	1.6×10^2	13.79	5.75×10^{-15}
RANDL3	1.3×10^3	13.70	1.09×10^{-14}
RANDL4	2.0×10^4	13.41	6.10×10^{-14}
RANDL5	1.3×10^5	13.79	4.87×10^{-13}
RANDL6	1.3×10^6	13.95	2.85×10^{-13}
RANDL7	1.3×10^7	14.42	7.73×10^{-11}

0.1%, $m := 30{,}000$, and $n := 3{,}000$. The third column gives the CPU time in seconds taken by a direct method, namely the " backslash \ " solver in MATLAB. The fourth column gives the resulting relative residual $\|A^T\boldsymbol{r}\|_2/\|A^T\boldsymbol{b}\|_2$. The required accuracy for the relative residual $\varepsilon = 10^{-8}$ is satisfied in all cases. Observe that as the condition number increases, the accuracy decreases.

Table 4.2 gives the shortest CPU time for the iterative methods to achieve a relative residual $\|A^T\boldsymbol{r}_j\|_2/\|A^T\boldsymbol{b}\|_2$ less than $\varepsilon = 10^{-8}$ as in (4.1) for each problem. Diag., RIF, Cimm.-NR, NR-SOR, and NR-SSOR stand for the diagonal scaling, RIF

TABLE 4.2
Best results for artificial random problems.

Solver	CGLS				BA-GMRES			
Precon.	diag.	RIF	Cimm.-NR	NR-SSOR	diag.	RIF	Cimm.-NR	NR-SOR
RANDL1	68	62 (0.4)	35 (2, 0.8)	32 (1, 1.0)	68	56 (0.3)	35 (2, 0.8)	9 (5, 1.5)
	0.04	0.10 (0.07)	0.05	0.04	0.05	0.11 (0.06)	0.04	* 0.02
RANDL2	192	110 (0.1)	103 (2, 0.7)	88 (1, 1.0)	191	130 (0.2)	105 (2, 0.7)	15 (9, 1.7)
	0.12	0.20 (0.13)	0.15	0.12	0.21	0.20 (0.16)	0.14	* 0.06
RANDL3	624	259 (0.05)	328 (2, 0.7)	276 (1, 1.0)	595	144 (0.04)	234 (4, 0.7)	41 (10, 1.8)
	0.38	0.32 (0.15)	0.46	0.38	1.62	0.32 (0.18)	0.69	* 0.16
RANDL4	2,863	489 (0.01)	1,594 (2, 0.6)	1,253 (1, 1.0)	1,479	334 (0.009)	508 (8, 0.7)	126 (15, 1.9)
	1.74	0.79 (0.57)	2.25	1.70	8.73	1.09 (0.50)	3.36	* 0.73
RANDL5	5,482	1,033 (0.007)	3,019 (2, 0.6)	2,392 (1, 1.0)	1,613	569 (0.005)	560 (8, 0.7)	339 (6, 1.4)
	3.33	* 1.14 (0.43)	4.24	3.25	10.28	2.17 (0.65)	3.78	1.19
RANDL6	13,286	3,918 (0.009)	7,310 (2, 0.6)	5,648 (1, 1.0)	1,822	787 (0.002)	619 (10, 0.7)	422 (6, 1.3)
	8.08	2.87 (0.20)	8.72	7.68	12.83	3.93 (1.18)	5.05	* 1.62
RANDL7	74,093	7,429 (0.001)	41,310 (2, 0.6)	32,605 (1, 1.0)	3,023	1,125 (0.002)	784 (8, 0.7)	499 (6, 1.2)
	44.98	7.55 (2.31)	58.09	44.31	33.90	6.20 (0.98)	5.97	* 2.11

First row: Number of (outer) iterations (number of inner iterations, preconditioning parameter)
Second row: Total CPU time [seconds] (preconditioning time)
Relative residual $< 10^{-8}$

[6], Cimmino-NR, NR-SOR, and NR-SSOR preconditioners, respectively. The reason that we use NR-SSOR for CGLS is to obtain a symmetric preconditioner. The first row in each cell gives the number of (outer) iterations outside the brackets, and the number of inner iterations and the best parameter value for each method in brackets. The second row gives the total CPU time including the preconditioning time in seconds outside the brackets, and the time to set up the preconditioning matrix of RIF in brackets.

The optimal number of inner iterations and the optimal parameters λ and ω were experimentally determined so that it realizes the shortest CPU time. The parameters λ and ω tested for Cimmino-NE and NR-(S)SOR, respectively, were changed in the interval $[0.1, 1.9]$ with step size 0.1. We used $k \times 10^{-l}$ for the dropping tolerance for

RIF, where $k = 1, 2, \ldots, 9$ and $l = 1, 2, \ldots, 10$.

The $*$ indicates the fastest method, which was BA-GMRES with NR-SOR except for RANDL5. The superiority of the method becomes more pronounced as $\kappa(A)$ becomes large. For $\varepsilon = 10^{-8}$, the method was also faster than the direct method, even for the ill-conditioned problems. CGLS with reorthogonalization [18] was also tested combined with these preconditioners, but it was slow to converge.

Figures 4.1–4.3 show the relative residual defined in (4.1) vs. the number (outer) iterations and the relative residual defined in (4.1) vs. the CPU time for RANDL4 and RANDL7 for each method. The convergence behaviors of CGLS with diagonal scaling

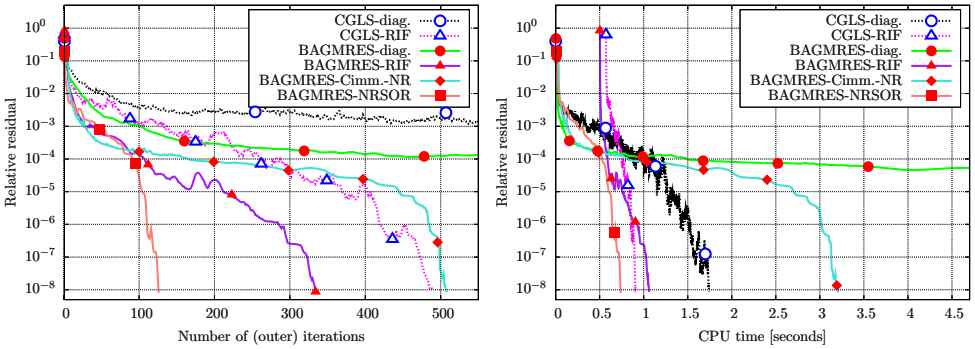

FIG. 4.1. *Relative residual* $\|A^T \boldsymbol{r}_j\|_2/\|A^T \boldsymbol{b}\|_2$ *vs. number of (outer) iterations and relative residual* $\|A^T \boldsymbol{r}_j\|_2/\|A^T \boldsymbol{b}\|_2$ *vs. CPU time for for RANDL4.*

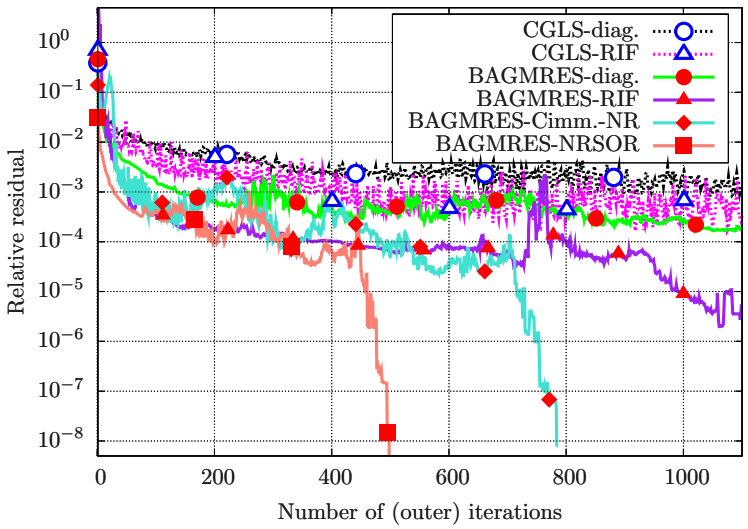

FIG. 4.2. *Relative residual* $\|A^T \boldsymbol{r}_j\|_2/\|A^T \boldsymbol{b}\|_2$ *vs. number of (outer) iterations for RANDL7.*

and RIF, and BA-GMRES with diagonal scaling, RIF, Cimmino-NR and NR-SOR are indicated by ○, △, ●, ▲, ♦, and ■, respectively. The convergence curve of the relative residual vs. CPU time for the methods with RIF is shifted to the right by the time required to construct the preconditioner. To observe the ordinary behavior of the relative residual for the methods, RANDL4 is suitable since it is fairly ill-conditioned.

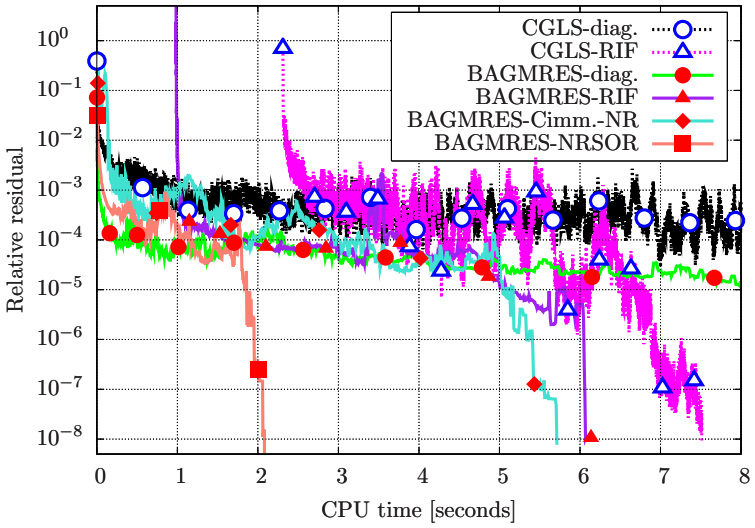

FIG. 4.3. *Relative residual* $\|A^T \boldsymbol{r}_j\|_2/\|A^T \boldsymbol{b}\|_2$ *vs. CPU time for RANDL7.*

The convergence curves for all the CGLS-based methods are quite oscillatory. BA-GMRES-based methods yield smoother curves. The convergence of BA-GMRES with NR-SOR is the fastest. The results for RANDL7 are representative of the superiority of BA-GMRES with NR-SOR, since RANDL7 is the most ill-conditioned problem in this experiment.

Experiments were done also for the consistent case $\boldsymbol{b} \in \mathcal{R}(A)$. Figure 4.4 shows the relative error vs. number of (outer) iterations for RANDL6 with $\boldsymbol{b} := A\,(1, 1, \ldots, 1)^T$.

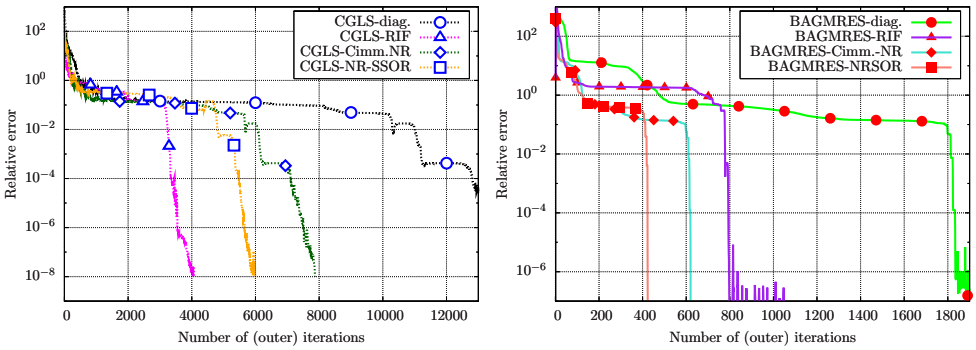

FIG. 4.4. *Relative error* $\|\boldsymbol{x}_j - \boldsymbol{x}_*\|_2/\|\boldsymbol{x}_*\|_2$ *vs. number of (outer) iterations for RANDL6.*

$\Diamond$ and $\square$ represent CGLS with Cimmino-NR and NR-SSOR, respectively. The other symbols are the same as in Figure 4.1. The figure on the left shows the results of the CGLS-based methods and the figure on the right shows the results of the BA-GMRES-based methods. The vertical axis gives the relative error $\|\boldsymbol{x}_j - \boldsymbol{x}_*\|_2/\|\boldsymbol{x}_*\|_2$, where $\boldsymbol{x}_*$ is the true solution $(1, 1, \ldots, 1)^T$. All the methods attain relative error less than 10^{-8}. Again, the convergence of BA-GMRES with NR-SOR is the fastest. Figure 4.4 shows that the error for the CGLS methods tend to decrease gradually,

whereas those of the BA-GMRES methods suddenly decrease at certain steps.

As shown in [18], preconditioned CGLS (PCGLS) and BA-GMRES for overdetermined problems minimize different quantities. PCGLS minimizes $\|\boldsymbol{r}_k|_{\mathcal{R}(A)}\|_2$, which is the 2-norm of the $\mathcal{R}(A)$ component of $\boldsymbol{r}_k$, whereas BA-GMRES minimizes $\|B\boldsymbol{r}_k\|_2$. Note that the convergence was monitored by $\|A^T\boldsymbol{r}_k\|_2$.

Figure 4.5 plots the number of outer iterations (left) and the CPU time (right) required to achieve relative residual less than $\varepsilon = 10^{-8}$ vs. the acceleration parameter ω for NR-SOR with BA-GMRES for RANDL7. k denotes the number of inner

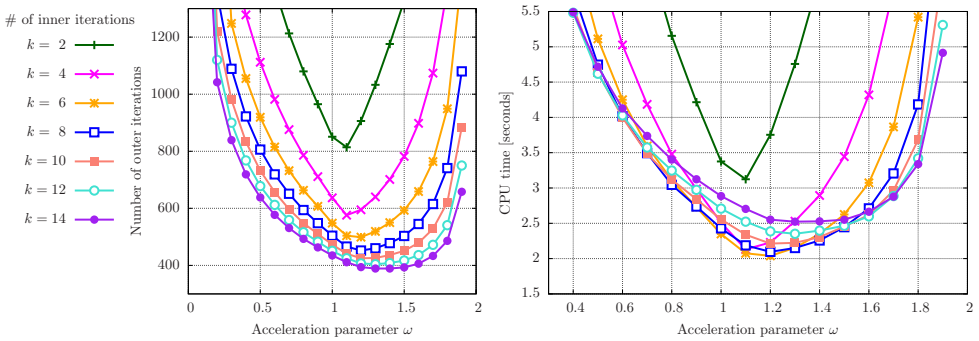

FIG. 4.5. *Number of outer iterations and CPU time to achieve relative residual* $< 10^{-8}$ *vs.* ω *for BA-GMRES with NR-SOR (RANDL7).*

iterations. The optimum value for the parameter ω with respect to the number of outer iterations and CPU time was between 1.1 and 1.3.

4.2. Experiments with practical problems. Next, we tested the methods for more practical problems.

4.2.1. Full rank problems. We tested the same methods as in Section 4.1. In Table 4.3, information on the test matrices, which were used in [6], are given, including the number of rows m, the number of columns n, the number of nonzero elements "nnz", and the density of the nonzero elements "dens". The first four matrices are

TABLE 4.3
Information on the practical test matrices.

Problem	m	n	nnz	dens. [%]	$\kappa(A)$	Time	$\|A^T\boldsymbol{r}_j\|_2/A^T\boldsymbol{b}\|_2$
well1033	1,033	320	4,732	1.43	1.66×10^2	0.001	8.50×10^{-16}
illc1033	1,033	320	4,732	1.43	1.89×10^4	0.001	1.08×10^{-13}
well1850	1,850	712	8,758	0.66	1.11×10^2	0.003	8.25×10^{-16}
illc1850	1,850	712	8,758	0.66	1.40×10^3	0.003	1.11×10^{-14}
SMALL	3,140	1,980	8,510	1.37×10^{-1}	1.06×10^2	0.01	4.78×10^{-10}
MEDIUM	9,397	6,119	25,013	4.35×10^{-2}	1.39×10^2	0.03	6.28×10^{-11}
LARGE	28,524	17,264	75,018	1.52×10^{-2}	1.93×10^2	0.12	6.21×10^{-11}
VERYL	174,193	105,882	463,303	2.51×10^{-3}	–	1.33	1.29×10^{-10}
HIRLAM	1,385,270	452,200	2,713,200	4.33×10^{-4}	–	128.97	1.23×10^{-15}

from [25], and the next four are from [19]. The last one, HIRLAM, is from [20].[2] The condition numbers were computed using the MATLAB functions `spnrank` [15] and `svd`. (The condition numbers of VERYL and HIRLAM could not be computed

[2]We would like to thank Professor Miroslav Tůma for providing the test data.

on our computers due to insufficient memory.) The CPU time and relative residual for the direct method are also given similar to Table 4.1.

Table 4.4 gives the results with $\varepsilon := 10^{-8}$, similar to Table 4.2. For convenience,

TABLE 4.4
Best results for practical full-rank problems.

Method	CGLS				BA-GMRES			
Precon.	diag.	RIF	Cimm.-NR	NR-SSOR	diag.	RIF	Cimm.-NR	NR-SOR
well1033	180	155 (0.5)	106 (2, 0.6)	84 (1, 1.0)	98	98 (0.8)	98 (1, 1.0)	65 (1, 1.0)
0.001	0.003	0.014 (0.010)	0.005	0.004	0.003	0.009 (0.006)	0.003	* 0.002
illc1033	3,748	421 (0.06)	2,371 (2, 0.4)	1,545 (1, 1.0)	256	98 (0.8)	256 (1, 1.0)	152 (1, 1.0)
0.001	0.071	0.019 (0.011)	0.107	0.077	0.013	0.009 (0.006)	0.014	* 0.007
well1850	449	287 (0.3)	252 (2, 0.6)	186 (1, 1.0)	399	151 (0.2)	170 (4, 0.7)	62 (5, 1.8)
0.003	0.016	0.027 (0.014)	0.022	0.019	0.061	0.034 (0.020)	0.033	* 0.012
illc1850	2,161	544 (0.2)	1,268 (2, 0.4)	928 (1, 0.9)	697	285 (0.2)	400 (6, 0.7)	245 (4, 1.4)
0.003	0.079	* 0.045 (0.018)	0.112	0.096	0.174	0.051 (0.014)	0.132	0.055
SMALL	171	90 (0.5)	90 (2, 0.7)	78 (1, 1.0)	170	56 (0.4)	63 (4, 0.7)	24 (6, 1.5)
0.01	0.008	0.015 (0.007)	0.010	0.010	0.031	0.020 (0.013)	0.014	* 0.007
MEDIUM	188	94 (0.5)	97 (2, 0.7)	82 (1, 1.0)	184	63 (0.4)	67 (4, 0.7)	25 (6, 1.4)
0.03	0.027	0.037 (0.013)	0.034	0.033	0.235	0.067 (0.034)	0.057	* 0.024
LARGE	190	99 (0.5)	97 (2, 0.7)	88 (1, 0.8)	185	100 (0.5)	55 (6, 0.7)	20 (8, 1.6)
0.12	0.079	0.116 (0.041)	0.104	0.102	0.770	0.312 (0.041)	0.190	* 0.065
VERYL	254	228 (0.7)	128 (2, 0.7)	114 (1, 0.9)	249	208 (0.5)	73 (4, 0.9)	18 (15, 1.7)
1.33	1.267	1.836 (0.263)	1.437	1.251	8.531	7.107 (0.485)	2.459	* 0.813
HIRLAM	180	73 (0.2)	95 (2, 0.9)	27 (1, 1.8)	170	53 (0.1)	63 (4, 0.9)	17 (9, 1.7)
128.97	8.965	5.541 (1.149)	10.334	* 3.013	41.163	8.155 (2.153)	14.675	4.568

First row: Number of (outer) iterations (number of inner-iterations, preconditioning parameter)
Second row: Total CPU time [seconds] (preconditioning time)
Relative residual: 10^{-8}

the CPU time for the direct method is given below the name of each problem. BA-GMRES with the NR-SOR inner iteration gave the shortest CPU time except for illc1850 and HIRLAM. For HIRLAM, CGLS with the NR-SSOR inner iterations gave the best result.

4.2.2. Rank-deficient problems. Next, experiments were done for rank-deficient problems given in [13]. The information on the matrices are given in Table 4.5, similar to Table 4.3. Maragal_8 was transposed so that $m > n$. The rank and the

TABLE 4.5
Information of the matrices.

Problem	m	n	nnz	dens. [%]	rank	$\kappa(A)$	Time	$\|A^T\boldsymbol{r}\|_2/\|A^T\boldsymbol{b}\|_2$
Maragal_3	1,682	858	18,391	1.27	613	1.10×10^3	0.07	1.63×10^{-13}
Maragal_4	1,964	1,027	26,719	1.32	801	9.33×10^6	0.14	4.15×10^{-9}
Maragal_5	4,654	3,296	93,091	0.61	2,147	1.19×10^5	1.75	5.60×10^{-2}
Maragal_6	21,251	10,144	537,694	0.25	8,331	2.91×10^6	381.15	1.37×10^{-1}
Maragal_7	46,845	26,525	1,200,537	0.10	20,843	8.98×10^6	483.39	7.95×10^{-2}
Maragal_8	60,845	33,093	1,308,415	0.06	–	–	74.84	1.12×10^{-9}

condition numbers were computed using the MATLAB functions `spnrank` [15] and `svd`. The last two columns give results for the direct method (backslash solver of MATLAB) similarly to Table 4.3. The direct solver attains relative residual less than $\varepsilon = 10^{-8}$ for Maragal_3, _4 and _8. For Maragal_5, _6, and _7, the solutions are inaccurate.

Table 4.6 gives the results for the iterative methods with $\varepsilon := 10^{-8}$, similar to Table 4.2 (The † in the first column indicates that the direct method did not

achieve relative residual less than 10^{-8}). For all the problems, BA-GMRES with

TABLE 4.6
Best results for practical rank-deficient problems.

Solver	CGLS			BA-GMRES		
Precon.	diag.	Cimm.-NR	NR-SSOR	diag.	Cimm.-NR	NR-SOR
Maragal_3	506	504 (1, 0.7)	127 (1, 1.1)	293	170 (4, 0.5)	88 (3, 1.2)
0.07	0.03	0.05	0.03	0.05	0.05	* 0.02
Maragal_4	1,230	1,192 (1, 1.1)	329 (1, 0.9)	316	149 (4, 0.5)	141 (2, 1.1)
0.14	0.11	0.15	0.08	0.07	0.06	* 0.04
Maragal_5	1,872	1,868 (1, 1.9)	377 (1, 1.1)	798	409 (5, 0.4)	151 (5, 1.5)
† 1.75	0.54	0.81	0.32	2.75	1.24	* 0.29
Maragal_6	64,824	64,093 (1, 0.8)	13,112 (1, 1.1)	2,708	1,445 (4, 0.4)	430 (7, 1.4)
† 381.15	187.06	264.15	86.90	92.16	40.21	* 7.89
Maragal_7	15,668	15,418 (1, 0.9)	2,138 (1, 1.1)	2,491	1,490 (3, 0.3)	334 (7, 1.5)
†483.39	103.51	155.97	39.93	211.65	99.47	* 14.83
Maragal_8	$> 10^6$	$> 10^6$	$> 10^6$	6,834	5,250 (2, 0.3)	924 (11, 1.3)
74.84	7,998.67			3,674.50	2,176.86	* 90.18

First row: Number of (outer) iterations (number of inner iterations, preconditioning parameter)
Second row: Total CPU time [seconds] (preconditioning time)
Relative residual $< 10^{-8}$

NR-SOR inner iterations gave best results. The method was particularly efficient for ill-conditioned problems like Maragal_8. The method was also faster than the direct method for $\varepsilon = 10^{-8}$, except for Maragal_8. The optimum number of inner iterations k was between 2 and 11. The CGLS methods did not converge for Maragal_8 for all parameters. The RIF preconditioner was also tested with various values for the threshold parameter, but it was slow to converge for large values and it broke down for small values.

Figure 4.6 shows the relative residual $\|A^T \boldsymbol{r}_j\|_2 / \|A^T \boldsymbol{b}\|_2$ vs. number of (outer) iterations for Maragal_6. The convergence graphs for the CGLS methods are oscillatory

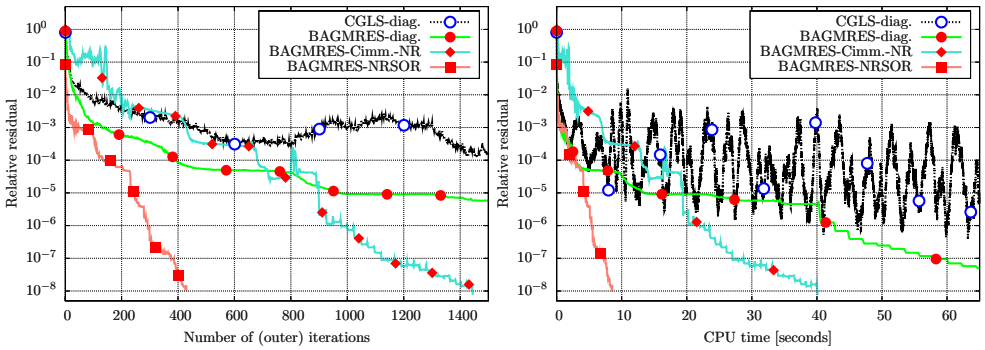

FIG. 4.6. *Relative residual* $\|A^T \boldsymbol{r}_j\|_2 / \|A^T \boldsymbol{b}\|_2$ *vs. number of (outer) iterations and relative residual* $\|A^T \boldsymbol{r}_j\|_2 / \|A^T \boldsymbol{b}\|_2$ *vs. CPU time for Maragal_6.*

and slow to converge. BA-GMRES with NR-SOR is shown to converge quickly.

Figure 4.7 plots the number of outer iterations (left) and the CPU time (right) for Maragal_7 similarly to Figure 4.5. k denotes the number of inner iterations. The optimum value for the parameter ω with respect to the number of outer iterations and CPU time was between 1.3 and 1.5.

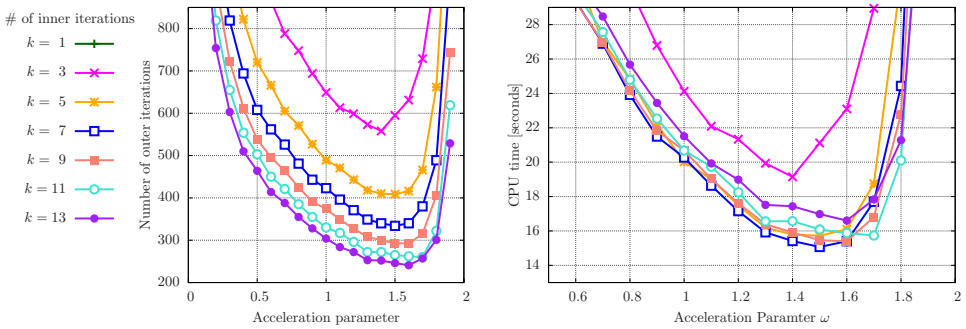

FIG. 4.7. *Number of outer iterations and CPU time to achieve relative residual* $< 10^{-8}$ *vs.* ω *for BA-GMRES with NR-SOR (Maragal_7).*

4.3. Automatic parameter tuning for NR-SOR. The NR-SOR method requires two parameters, the number of inner iterations k and the acceleration parameter ω. As seen in Figures 4.5 and 4.7, the number of outer iterations and CPU time for the NR-SOR inner-iteration BA-GMRES method vary with the values of these parameters. Hence, it is desirable to determine the parameters automatically for any given problem. The theoretical determination of the optimum relaxation parameter for some stationary iterative methods for some kinds of square matrices is described in [38], [34]. However, techniques for the determination for general matrices including rectangular matrices seem scarce. We proposed the following procedure, which should be performed before starting the main algorithm. This tunes the parameters numerically. The idea is to perform some test runs of the NR-SOR iterations alone beforehand in order to determine the near optimal n_{in} and ω.

1. Set ω, e.g., $\omega := 1$.
2. Starting from $n_{\text{in}} := 0$, find the minimum n_{in} which satisfies
$$\|\boldsymbol{x}^{(n_{\text{in}})} - \boldsymbol{x}^{(n_{\text{in}}+1)}\|_\infty \leq \eta \|\boldsymbol{x}^{(n_{\text{in}}+1)}\|_\infty.$$
3. Find ω_{opt} which minimizes $\|\boldsymbol{r}^{(n_{\text{in}})}\|_2$.

Here, ω_{opt} is the optimum acceleration parameter for NR-SOR. Let ω_i's be candidates for ω_{opt}. Since ω_{opt} is often in the range $[1, 2)$ and the quantity $\|\boldsymbol{r}^{(n_{\text{in}})}\|_2$ is usually a convex function of ω with a minimum at $\omega_{\text{opt}} \lesssim 2$, it is efficient to search for ω_{opt} from $\omega_i = i \times 10^{-1}$ for $i = 19, 18, \ldots, 1$ in this order.

The resulting n_{in} and ω_{opt} would not be absolutely optimum but would be nearly optimum. Fortunately, $\boldsymbol{x}^{(k)}$ and $\boldsymbol{r}^{(k)}$ appear in the algorithm for the NR-SOR iteration, so that the cost for this automatic tuning is marginal.

Table 4.7 gives the numerical experiment results with parameters automatically tuned by the above procedure for the problems presented in Section 4.2.2 with $\varepsilon = 10^{-8}$. The first row in each cell gives the number of outer iterations outside the brackets, and the automatically tuned number of inner iterations and acceleration parameter in brackets. The second row gives the total CPU time including the tuning time in seconds outside the brackets, and the parameter tuning time in brackets. Different values for the η in the tuning procedure from 10^{-2} to $10^{-0.5}$ were tested. The $*$ indicates the fastest case for each problem.

The CPU times for BA-GMRES with NR-SOR with automatically tuned parameters is close to those with optimum parameters given in Table 4.6. Moreover, the

TABLE 4.7
Results with automatically tuned parameters for the practical rank-deficient problems.

Method	BA-GMRES with NR-SOR				
η	10^{-2}	$10^{-1.5}$	10^{-1}	$10^{-0.5}$	Optimum
Maragal_3	18 (36, 1.8)	25 (20, 1.7)	51 (7, 1.5)	88 (3, 1.2)	88 (3, 1.2)
	0.046 (0.009)	0.036 (0.006)	0.026 (0.003)	* 0.023 (0.002)	0.02
Maragal_4	25 (73, 1.7)	47 (15, 1.7)	68 (7, 1.5)	118 (3, 1.2)	141 (2, 1.1)
	0.174 (0.030)	0.067 (0.006)	0.046 (0.004)	* 0.045 (0.002)	0.04
Maragal_5	39 (52, 1.8)	72 (17, 1.7)	112 (8, 1.5)	177 (4, 1.3)	151 (5, 1.5)
	0.619 (0.059)	0.391 (0.024)	* 0.307 (0.016)	0.332 (0.011)	0.29
Maragal_6	178 (105, 1.9)	268 (50, 1.9)	406 (9, 1.6)	515 (5, 1.4)	430 (7, 1.4)
	31.96 (0.53)	24.34 (0.26)	8.80 (0.09)	* 8.29 (0.07)	7.89
Maragal_7	185 (46, 1.9)	257 (13, 1.7)	334 (7, 1.5)	477 (4, 1.3)	334 (7, 1.5)
	39.68 (0.63)	18.32 (0.29)	* 15.06 (0.22)	17.67 (0.16)	14.83
Maragal_8	487 (140, 1.6)	844 (20, 1.6)	1,543 (4, 1.3)	2,294 (2, 1.1)	924 (11, 1.3)
	374.14 (4.51)	* 122.59 (0.65)	131.41 (0.19)	237.36 (0.12)	90.18

First row: Number of outer iterations (number of inner iterations, acceleration parameter)
Second row: Total CPU time [seconds] (parameter tuning time)

CPU time required for tuning the parameter was marginal compared to the total CPU time. The value $\eta = 10^{-1.5} \sim 10^{-0.5}$ for the tuning gave good results.

4.4. Estimation of convergence. In the numerical experiments so far, we used $\|A^T \boldsymbol{r}_j\|_2$ to check the convergence. In the CGLS (cf. Section 2.4), $\|A^T \boldsymbol{r}_j\|_2 = \|\boldsymbol{s}_j\|_2$ can be obtained directly from the algorithm. On the other hand, in the BA-GMRES algorithm, $\|A^T \boldsymbol{r}_j\|_2 = \|A^T (\boldsymbol{b} - A\boldsymbol{x}_j)\|_2$ is not directly available, so one has to compute it at each outer iteration, which may be expensive. However, in Algorithm 2.2 for BA-GMRES, $\|B\boldsymbol{r}_j\|_2$, $j = 1, 2, \ldots, l$, is given by the scalar

$$|\gamma_j| = \begin{cases} \beta, & j = 0, \\ |(\boldsymbol{e}_j, Q_j(\beta \boldsymbol{e}_1))|, & j = 1, 2, \ldots, l, \end{cases}$$

where Q_j is the product of Givens rotations that transforms $\bar{H}_j$ into upper triangular form. If B satisfies $\mathcal{R}(A) = \mathcal{R}(B^T)$, then the problem $\min_{\boldsymbol{x} \in \mathbf{R}^n} \|B(\boldsymbol{b} - A\boldsymbol{x})\|_2$ is consistent, and $\|B(\boldsymbol{b} - A\boldsymbol{x}_j)\|_2$ converges to zero, as $\boldsymbol{x}_j$ approaches the least squares solution of $\min_{\boldsymbol{x} \in \mathbf{R}^n} \|\boldsymbol{b} - A\boldsymbol{x}\|_2$. Hence, instead of $\|A^T \boldsymbol{r}_j\|_2$, γ_j may be used for monitoring the convergence of BA-GMRES.

Thus, we compare the convergence curve of $\|A^T \boldsymbol{r}_j\|_2$ with that of γ_j. Figures 4.8 and 4.9 show the relative residual $\|A^T \boldsymbol{r}_j\|_2 / \|A^T \boldsymbol{r}_0\|_2$ and γ_j/γ_0 vs. the number of outer iterations for BA-GMRES with NR-SOR. The quantities $\|A^T \boldsymbol{r}_j\|_2 / \|A^T \boldsymbol{r}_0\|_2$ and γ_j/γ_0 are plotted by the dotted curve and the solid curve, respectively. In Figure 4.8, the figure on the left shows the result for RANDL3 and the figure on the right shows the result for RANDL6. Similar results are shown for HIRLAM (left) and Maragal_6 (right) in Figure 4.9. The parameters for NR-SOR were the ones giving the shortest CPU time. The difference between $\|A^T \boldsymbol{r}_j\|_2 / \|A^T \boldsymbol{r}_0\|_2$ and γ_j/γ_0 for RANDL6 is more pronounced than that for RANDL3. Among RANDLn, $n = 1, 2, \ldots, 7$, γ_j/γ_0 tends to be different from $\|A^T \boldsymbol{r}_j\|_2 / \|A^T \boldsymbol{r}_0\|_2$ for the ill-conditioned problems. For HIRLAM and Maragal_6, γ_j/γ_0 gives a good estimate of $\|A^T \boldsymbol{r}_j\|_2 / \|A^T \boldsymbol{r}_0\|_2$.

In practice, if one wants to monitor $\|A^T \boldsymbol{r}_j\|_2$ in BA-GMRES, the cost for computing $\|A^T \boldsymbol{r}_j\|_2$ can be reduced by checking $\|A^T \boldsymbol{r}_j\|_2$ only after γ_j becomes sufficiently small, and then computing $\|A^T \boldsymbol{r}_j\|_2$ only every several iterations.

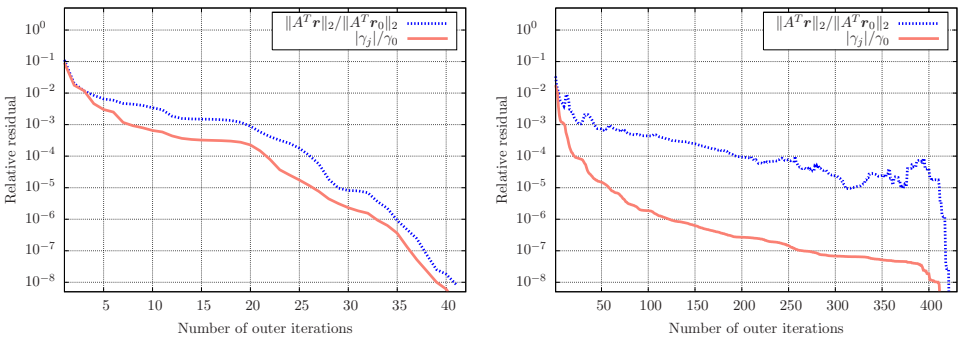

FIG. 4.8. *Relative residual vs. number of outer iterations for RANDL3 (left) and RANDL6 (right).*

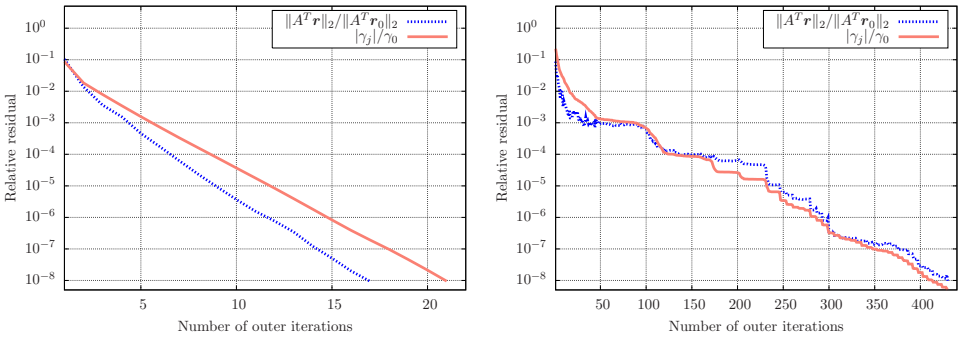

FIG. 4.9. *Relative residual vs. number of outer iterations for HIRLAM (left) and Maragal_6 (right).*

5. Conclusions. We proposed applying stationary inner iterations, as preconditioners, to Krylov subspace methods for solving least squares problems. For the inner iterations, Cimmino- (Jacobi-), SOR-, and SSOR-type iterations for least squares problems were employed. For the outer iterations, conjugate gradient type (CGNR and CGNE) methods, and GMRES methods were used.

The inner iterations are efficient in terms of CPU time and memory, and they also serve as powerful preconditioners effective also for ill-conditioned and rank-deficient least squares problems. Theoretical justifications for using the inner iterations as preconditioners were also presented.

Numerical experiments for overdetermined least squares problems, including ill-conditioned, rank-deficient, and practical problems, showed that the NR-SOR inner iterations combined with the left-preconditioned (BA) GMRES method is the most effective method, which outperforms previous methods. A strategy for choosing the parameters for the NR-SOR inner iterations was also proposed and shown to be effective.

Acknowledgments. We would like to thank Professors Zhong-Zhi Bai, Rafael Bru, Michael Eiermann, and José Mas, and Dr. Xiaoke Cui for discussions and valuable advice.

REFERENCES

[1] K. Abe and S.-Z. Zhang, *A variable preconditioning using the SOR method for GCR-like methods*, Int. J. Numer. Anal. Model., 2 (2005), pp. 147–161.

[2] D. Aoto, E. Ishiwata, and K. Abe, *A variable preconditioned GCR(m) method using the GSOR method for singular and rectangular linear systems*, J. Comput. Appl. Math., 234 (2010), pp. 703–712.

[3] O. Axelsson and P. S. Vassilevski, *A black box generalized conjugate gradient solver with inner iterations and variable-step preconditioning*, SIAM J. Matrix Anal. Appl., 12 (1991), pp. 625–644.

[4] Z.-Z. Bai, I. S. Duff, and A. J. Wathen, *A class of incomplete orthogonal factorization methods. I: Methods and theories*, BIT, 41 (2001), pp. 53–70.

[5] M. Benzi and M. Tůma, *A robust incomplete factorization preconditioner for positive definite matrices*, Numer. Linear Algebra Appl., 10 (2003), pp. 385–400.

[6] ———, *A robust preconditioner with low memory requirements for large sparse least squares problems*, SIAM J. Sci. Comp., 25 (2003), pp. 499–512.

[7] Å. Björck, *Numerical Methods for Least Squares Problems*, SIAM, Philadelphia, 1996.

[8] Å. Björck and T. Elfving, *Accelerated projection methods for computing pseudoinverse solutions of systems of linear equations*, BIT, 19 (1979), pp. 145–163.

[9] Y. Censor, *Row-action methods for huge and sparse systems and their applications*, SIAM Review, 23 (1981), pp. 444–466.

[10] G. Cimmino, *Calcolo approssimato per le soluzioni dei sistemi di equazioni lineari*, La Ricerca Scientiica, 2 (1938), pp. 326–333.

[11] X. Cui and K. Hayami, *Generalized approximate inverse preconditioners for least squares problems*, Japan J. Indust. Appl. Math., 26 (2009), pp. 1–14.

[12] X. Cui, K. Hayami, and J.-F. Yin, *Greville's method for preconditioning least squares problems*, Adv. Comput. Math., accepted.

[13] T. Davis, *The University of Florida Sparse Matrix Collection*, available online at `http://www.cise.ufl.edu/research/sparse/matrices/`, The University of Florida.

[14] E. De Sturler, *Nested Krylov methods based on GCR*, J. Comput. Appl. Math., 67 (1996), pp. 15–41.

[15] L. Foster, *San Jose State University Singular Matrix Database*, available online at `http://www.math.sjsu.edu/singular/matrices/`, San Jose State University.

[16] V. Frayssé, L. Giraud, and S. Gratton, *Algorithm 881: A set of flexible GMRES routines for real and complex arithmetics on high-performance computers*, ACM Trans. Math. Software, 35 (2008), pp. 13:1–12.

[17] G. H. Golub and Q. Ye, *Inexact preconditioned conjugate gradient method with inner-outer iteration*, SIAM J. Sci. Comput., 21 (1999), pp. 1305–1320.

[18] K. Hayami, J.-F. Yin, and T. Ito, *GMRES methods for least squares problems*, SIAM J. Matrix Anal. Appl., 31 (2010), pp. 2400–2430.

[19] M. Hegland, *Description and use of animal breeding data for large least squares problems*, Technical report TR/PA/93/50, CERFACS, (1993).

[20] A. Holstad and I. Lie, *On the computation of mass conservative wind and vertical velocity fields*, Technical report 141, The Norweigian Meteorological Institute, (2002).

[21] A. Jennings and M. A. Ajiz, *Incomplete methods for solving $A^T Ax = b$*, SIAM J. Sci. Stat. Comput., 5 (1984), pp. 978–987.

[22] M. S. Kaczmarz, *Angenäherte auflösung von systemen linearer bleichungen*, Bulletin international de l'Academie polonaise des Sciences et Lettres, 3A (1937), pp. 355–357.

[23] Y. Notay, *Flexible conjugate gradients*, SIAM J. Sci. Comput., 22 (2000), pp. 1444–1460.

[24] X.-W. Ping, R.-S. Chen, K.-F. Tsang, and E.-K.-N. Yung, *The SSOR-preconditioned inner outer flexible GMRES method for the FEM analysis of EM problems*, Microwave Optical Technology Letters, 48 (2006), pp. 1708–1711.

[25] National Institute of Standards and Technology, *Matrix Market*, available online at `http://gams.nist.gov/MatrixMarket/`.

[26] Y. Saad, *Preconditioning techniques for nonsymmetric and indefinite linear systems*, J. Comput. Appl. Math., 24 (1988), pp. 89–105.

[27] ———, *A flexible inner-outer preconditioned GMRES algorithm*, SIAM J. Sci. Comput., 14 (1993), pp. 461–469.

[28] ———, *Iterative Methods for Sparse Linear Systems*, 2nd ed., SIAM, Philadelphia, 2003.

[29] Y. Saad and M. H. Schultz, *GMRES: A generalized minimal residual algorithm for solving nonsymmetric linear systems*, SIAM J. Sci. Stat. Comput., 7 (1986), pp. 856–869.

[30] V. Simoncini and D. B. Szyld, *Flexible inner-outer Krylov subspace methods*, SIAM J. Numer.

Anal., 40 (2003), pp. 2219–2239.
[31] D. B. Szyld and J. A. Vogel, *FQMR: A flexible quasi-minimal residual method with inexact preconditioning*, SIAM J. Sci. Comput., 23 (2001), pp. 363–380.
[32] K. Tanabe, *Projection method for solving a singular system of linear equations and its applications*, Numer. Math., 17 (1971), pp. 203–214.
[33] H. A. van der Vorst and C. Vuik, *GMRESR: a family of nested GMRES methods*, Numer. Linear Algebra Appl., 1 (1994), pp. 369–386.
[34] R. S. Varga, *Matrix iterative analysis*, Springer Verlag, 2nd ed. ed., 2000.
[35] J. A. Vogel, *Flexible BiCG and flexible Bi-CGSTAB for nonsymmetric linear systems*, Appl. Math. Comp., 188 (2007), pp. 226–233.
[36] X. Wang, K. A. Gallivan, and R. Bramley, *CIMGS: An incomplete orthogonal factorization preconditioner*, SIAM J. Sci. Comp., 18 (1997), pp. 516–536.
[37] J.-F. Yin and K. Hayami, *Preconditioned GMRES methods with incomplete Givens orthogonalization method for large sparse least-squares problems*, J. Comput. Appl. Math., 226 (2009), pp. 177–186.
[38] D. M. Young, *Iterative solution of large linear systems*, Academic Press, 1971.