STATIONARY INNER-ITERATION PRECONDITIONED GMRES METHOD FOR LEAST SQUARES PROBLEMS *

KEIICHI MORIKUNI† AND KEN HAYAMI‡

Abstract. Stationary inner-iteration preconditioners are applied to Krylov subspace methods for solving least squares problems. Theoretical justifications for the preconditioners even for rank-deficient matrices are presented. The inner iterations are efficient in terms of computational work and memory. Numerical experiments for large sparse least squares problems including ill-conditioned and rank-deficient ones show that the proposed methods outperform previous methods.

Key words. least squares problems, iterative method, preconditioner, inner-outer iteration, GMRES method, CG method, Jacobi method, SOR 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 of full rank. The least squares problem (1.1) is equivalent to the following normal equation

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

Direct and iterative methods for solving the problem (1.1) are described in Bjorck [6]. The standard iterative methods are the (preconditioned) CGLS [20] and LSQR methods [25]. However, since $\kappa_2(A^\mathsf{T}A) = \kappa_2(A)^2$, iterative methods may be slow to converge. Here, $\kappa_2(A) = \sigma_{\max}/\sigma_{\min}$ is the condition number, where $\sigma_{\max}$ and $\sigma_{\min}$ are the largest and smallest positive singular values of A. Hence, when iterative methods are used, good preconditioners are necessary to achieve better convergence. For this purpose, there are preconditioners such as [22, 27, 36, 4, 5, 9, 37, 19, 10] 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 called inner-outer iteration methods [3, 28, 33, 13, 12, 14, 16, 24, 31, 1, 35, 15, 26]. General Krylov subspace inner-outer methods were analysed in [30].

As for least squares problems, the preconditioned generalized conjugate residual (GCR) method preconditioned by successive overrelaxation (SOR)-like inner iterations and applied to singular systems was proposed in [2]. However, this method is not theoretically guaranteed to work, and there are indeed examples for which the method does not converge.

*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).

In this paper, we propose using Jacobi overrelaxation and SOR-type iterative methods specifically designed for least squares problems, as inner-iteration preconditioners for CGLS and the generalized minimal residual method (GMRES). 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 that can be solved. Note that, previously, GMRES preconditioned by the robust incomplete factorization (RIF) [5] was comparable with, but not definitely superior to, reorthogonalized CGLS with RIF in terms of CPU time for ill-conditioned problems [19]. Moreover, RIF for least squares problems [5] will break down for rank-deficient matrices. We aim to improve on 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 stationary inner iterations. In Section 3, we present the stationary iterative methods used for the inner iteration preconditioning. In Section 4, we present numerical results comparing the new preconditioners with conventional methods. Section 5 concludes the paper.

Throughout this paper, we use bold letters for column vectors. $\boldsymbol{e}_j$ denotes the jth column of an identity matrix. We denote quantities for inner iterations with a superscript with brackets, e.g., $\boldsymbol{x}^{(k)}$, and for outer iterations with a subscript without brackets, e.g., $\boldsymbol{x}_k$.

2. Preconditioning framework for inner iteration. We present frameworks for applying inner stationary-iteration preconditioning to the Krylov subspace methods for least squares problems.

Without loss of generality, the initial approximate solution $\boldsymbol{x}_0$ for the Krylov subspace methods is set to zero in the remaining sections. The initial residual is therefore $\boldsymbol{r}_0 = \boldsymbol{b}$.

2.1. Stationary inner-iteration BA-GMRES method. The left-preconditioned GMRES method for least squares problems proposed in [19] is called the BA-GMRES method. It 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 BA-GMRES were given in [19]. Let $\mathcal{R}(A)$ denote the range space of A.

THEOREM 2.1. *$\min_{\boldsymbol{x}\in\mathbf{R}^n} \|\boldsymbol{b} - A\boldsymbol{x}\|_2$ 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 $\mathcal{R}(A) = \mathcal{R}(B^\mathsf{T}BA)$.*

THEOREM 2.2. *If $\mathcal{R}(A) = \mathcal{R}(B^\mathsf{T})$, 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 $\boldsymbol{x}_0 \in \mathbf{R}^n$ without breakdown if and only if $\mathcal{R}(A^\mathsf{T}) = R(B)$.*

Here, we give a new condition for the preconditioning matrix B for BA-GMRES for the consistent system from [18, Theorem 2.8].

THEOREM 2.3. *BA-GMRES determines a solution for all $B\boldsymbol{b} \in \mathcal{R}(BA)$, $\boldsymbol{x}_0 \in \mathbf{R}^n$ if and only if $\mathcal{R}(BA) \cap \mathcal{N}(BA) = \{\boldsymbol{0}\}$.*

Moreover, due to [19, Theorem 3.15], Theorem 2.3 gives the following.

THEOREM 2.4. *Suppose $\mathcal{R}(A) = R(B^\mathsf{T})$. Then, BA-GMRES determines a solution of $\min_{\boldsymbol{x}\in\mathbf{R}^n} \|\boldsymbol{b} - A\boldsymbol{x}\|_2$ without breakdown for all $\boldsymbol{b} \in \mathbf{R}^m$ and $\boldsymbol{x}_0 \in \mathbf{R}^n$ if and only if $\mathcal{R}(B) \cap \mathcal{N}(A) = \{0\}$.*

Note that $\mathcal{R}(A) = R(B^\mathsf{T})$ implies $B\boldsymbol{b} \in \mathcal{R}(BA)$, and also guarantees the equivalence between $\min_{\boldsymbol{x}\in\mathbf{R}^n} \|\boldsymbol{b} - A\boldsymbol{x}\|_2$ and $\min_{\boldsymbol{x}\in\mathbf{R}^n} \|B\boldsymbol{b} - BA\boldsymbol{x}\|_2$ for all $\boldsymbol{b} \in \mathbf{R}^m$ [19].

In [19], RIF [5] was used to construct 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.

Instead of forming an explicit preconditioning matrix B, we propose applying several steps of a certain stationary iterative method inside GMRES whenever B is needed in BA-GMRES. Therefore, our strategy can be considered as a preconditioned GMRES with an implicit B, i.e., B is not formed or stored explicitly.

2.1.1. Outer iteration. We consider applying stationary inner iterations to BA-GMRES [19]. Algorithm 2.1 shows the general framework for this approach, where p is the restart cycle. In the following and hereafter, $(\boldsymbol{a}, \boldsymbol{b})$ denotes the inner product $\boldsymbol{a}^\mathsf{T}\boldsymbol{b}$ between real vectors $\boldsymbol{a}$ and $\boldsymbol{b}$, and ε denotes the threshold for stopping the outer iteration.

Algorithm 2.1 Stationary inner-iteration BA-GMRES(p) method.

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

Here, $\bar{H}_k \equiv \{h_{pq}\} \in \mathbf{R}^{(k+1)\times k}$. If $\boldsymbol{x}_k$ is a least squares solution, the method terminates, and we do not have a breakdown. (In general, $\boldsymbol{x}_k$ may or may not be a least squares solution).

The idea behind line 1 is as follows. First consider the problem

$$\min_{\boldsymbol{z}\in\mathbf{R}^n} \|\boldsymbol{r}_0 - A\boldsymbol{z}\|_2, \tag{2.1}$$

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

$$A^\mathsf{T}A\boldsymbol{z} = A^\mathsf{T}\boldsymbol{r}_0, \quad \text{or} \quad A^\mathsf{T}(\boldsymbol{r}_0 - A\boldsymbol{z}) = \boldsymbol{0}, \quad \text{or} \quad \boldsymbol{r}_0 - A\boldsymbol{z} \in \mathcal{N}(A^\mathsf{T}) = \mathcal{R}(A)^\perp, \tag{2.2}$$

which, in turn, is equivalent to

$$BA\boldsymbol{z} = B\boldsymbol{r}_0, \quad \text{or} \quad B(\boldsymbol{r}_0 - A\boldsymbol{z}) = \boldsymbol{0}, \quad \text{or} \quad \boldsymbol{r}_0 - A\boldsymbol{z} \in \mathcal{N}(B) = \mathcal{R}(B^\mathsf{T})^\perp, \tag{2.3}$$

if $\mathcal{R}(A) = \mathcal{R}(B^\mathsf{T})$. Hence, if we roughly solve (2.1) to obtain $A^\mathsf{T}A\tilde{\boldsymbol{z}} \simeq A^\mathsf{T}\boldsymbol{r}_0$ and set $\tilde{\boldsymbol{z}} := B\boldsymbol{r}_0$, (2.1) 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 (2.1)–(2.3). The idea behind line 4 can be explained similarly by replacing $\boldsymbol{r}_0$ by $A\boldsymbol{v}_k$, where $\boldsymbol{v}_k$ is given.

When B is fixed for all outer iterations, the method determines an approximation $\boldsymbol{x}_k$ in $\mathcal{K}_k(BA, \tilde{\boldsymbol{r}}_0)$ minimizing $\|B\boldsymbol{r}_k\|_2$, and is guaranteed to give a least squares solution if the conditions $\mathcal{R}(A) = \mathcal{R}(B^{\mathsf{T}})$ and $\mathcal{R}(A^{\mathsf{T}}) = \mathcal{R}(B)$ of Theorem 2.2 are satisfied, where the Krylov subspace at the kth step is $\mathcal{K}_k(BA, \tilde{\boldsymbol{r}}_0) = \operatorname{span}\left\{\tilde{\boldsymbol{r}}_0, BA\tilde{\boldsymbol{r}}_0, \ldots, (BA)^{k-1}\tilde{\boldsymbol{r}}_0\right\}$, and $\tilde{\boldsymbol{r}}_0 = B\boldsymbol{r}_0$.

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

$$\boldsymbol{x}_0 + \operatorname{span}\left\{B_0\boldsymbol{r}_0, (B_1A)B_0\boldsymbol{r}_0, (B_2A)(B_1A)B_0\boldsymbol{r}_0, \ldots, (B_kA)(B_{k-1}A)\cdots B_0\boldsymbol{r}_0\right\},$$

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, cf. FGMRES [28] and GMRESR [33]. We keep B constant in the remaining sections.

2.1.2. Stationary inner iterations. For the rough solution in lines 1 and 4 in Algorithm 2.1.1, we consider using a stationary iterative method for solving the normal equation $A^{\mathsf{T}}A\boldsymbol{x} = A^{\mathsf{T}}\boldsymbol{b}$. Here, we denote $\boldsymbol{b}$ instead of $\boldsymbol{r}_0$ or $A^{\mathsf{T}}\boldsymbol{v}_k$ for convenience. Let $A^{\mathsf{T}}A = M - N$, where M is nonsingular. Then, we have a class of stationary iterative methods of the form

$$\boldsymbol{x}_{k+1} = M^{-1}N\boldsymbol{x}_k + M^{-1}A^{\mathsf{T}}\boldsymbol{b}, \quad k = 0, 1, \ldots. \tag{2.4}$$

Define the iteration matrix by $H = M^{-1}N$. Tanabe [32] defined the following.

DEFINITION 2.5. *H is convergent if $\lim_{l\to\infty} H^l$ exists, and divergent otherwise.*

Then, we obtain the following theorem for stationary inner-iteration BA-GMRES.

THEOREM 2.6. *Assume that H is convergent. Then, BA-GMRES with the inner-iteration preconditioning of the form* (2.1) *determines a least squares solution of $\min_{\boldsymbol{x}\in\mathbf{R}^n} \|\boldsymbol{b} - A\boldsymbol{x}\|_2$ without breakdown for all $\boldsymbol{b} \in \mathbf{R}^m$.*

In section 3, we will introduce concrete iterative methods satisfying the condition in Theorem 2.6.

2.1.3. Convergence bound for stationary inner-iteration BA-GMRES. The preconditioned matrix with $(l+1)$ stationary inner iterations is given by $B^{(l)}A = I - H^{l+1}$. Let ν be an eigenvalue of H, and let $\mu^{(l)} = 1 - \nu^{l+1}$. Assume that H is convergent and $r = \operatorname{rank} A$. Then, $B^{(l)}A$ has r eigenvalues such that $|1 - \mu^{(l)}| \leq \rho(H)^{l+1} < 1$, and $(n-r)$ zero eigenvalues. Hence, the r eigenvalues of $B^{(l)}A$ approach 1 as l increases.

If A has full-column rank, then the field of values of $B^{(l)}A$ is contained in a circle $\{z \in \mathbf{C} : |z-1| \leq \rho(H)^{l+1}\}$. Thus, from Greenbaum [17, p. 56], we have the following theorem.

THEOREM 2.7. *Assume that H is convergent. If A has full column rank, then the stationary inner-iteration BA-GMRES residual satisfies*

$$\|B^{(l)}\boldsymbol{r}_k\|_2 / \|B^{(l)}\boldsymbol{r}_0\|_2 \leq 2\rho(H)^{k\times(l+1)},$$

where k is the number of outer iterations, and l is that of inner iterations.

Theorem 2.7 implies that the convergence bound on the stationary inner-iteration BA-GMRES residual depends on the spectral radius of the iteration matrix. Moreover,

if the stopping criterion for the kth outer iteration is $\|B^{(l)}\boldsymbol{r}_k\|_2/\|B^{(l)}\boldsymbol{r}_0\|_2 < \varepsilon$, then the required number of outer iterations is at most $\min\{n, \lceil \log_{\rho(H)} \frac{\varepsilon}{2}/(l+1)\rceil\}$.

2.2. Stationary inner-iteration CGLS method. We can also apply symmetric inner-iteration preconditioners to CGLS [29] as follows.

Algorithm 2.2 Stationary inner-iteration PCGLS method.

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

In lines 1 and 10, the normal equations may be roughly solved using a symmetric iterative method. Here, the preconditioning matrix P is a symmetric positive definite matrix. The normal equation in lines 1 and 10 is solved symmetrically, if necessary, as follows

$$\begin{cases} \boldsymbol{x}^{(k+\frac{1}{2})} &= M^{-1}N\boldsymbol{x}^{(k)} + M^{-1}A^{\mathsf{T}}\boldsymbol{b}, \\ \boldsymbol{x}^{(k+1)} &= M^{-\mathsf{T}}N^{\mathsf{T}}\boldsymbol{x}^{(k+\frac{1}{2})} + M^{-\mathsf{T}}A^{\mathsf{T}}\boldsymbol{b}, \end{cases} \qquad k = 0, 1, \dots$$

3. Inner stationary-iteration preconditioning. In this section, we introduce iterative methods that can be used to perform the stationary inner-iteration preconditioning for the outer solvers discussed in Section 2.

In the following, assume that A has no zero columns, and the initial approximate solution for inner iterations $\boldsymbol{x}^{(0)}$ is set to zero. Let $\boldsymbol{a}_j$ denote the jth column of A. We denote a component of a vector by a non-bold letter with a subscript.

Kaczmarz [23] and Cimmino [8] designed stationary iterative methods for solving the normal equation $AA^{\mathsf{T}}\boldsymbol{u} = \boldsymbol{b}$, $\boldsymbol{x} = A^{\mathsf{T}}\boldsymbol{u}$ avoiding the explicit formation of AA^{T} in the case of a square matrix. Bjorck and Elfving [7] relates Kaczmarz's method to the SOR, and symmetric SOR methods for the normal equation $A^{\mathsf{T}}A\boldsymbol{x} = A^{\mathsf{T}}\boldsymbol{b}$ in the case of a rectangular matrix. Cimmino's method can be generalized to the Jacobi overrelaxation method also for the normal equation. We call the methods the NR-SOR, NR-SSOR, and Cimmino-NR methods, respectively, from Saad [29]. According to Dax [11], the above methods give a convergent iteration matrix. Consequently, letting $A^{\mathsf{T}}A = L + D + L^T$, where L is a strictly lower triangular matrix, and D is a diagonal matrix, we have the following theorems.

THEOREM 3.1. *Assume that the overrelaxation parameter ω for Cimmino-NR satisfies $0 < \omega < 2/\rho(D^{-1/2}A^{\mathsf{T}}AD^{-1/2})$. Then BA-GMRES with the inner-iteration preconditioning determines a least squares solution of $\min_{\boldsymbol{x}\in\mathbf{R}^n} \|\boldsymbol{b} - A\boldsymbol{x}\|_2$ without breakdown for all $\boldsymbol{b} \in \mathbf{R}^m$.*

THEOREM 3.2. *Assume that the overrelaxation parameter ω for NR-SOR satisfies $0 < \omega < 2$. Then BA-GMRES with the inner-iteration preconditioning determines a least squares solution of $\min_{\boldsymbol{x}\in\mathbf{R}^n} \|\boldsymbol{b} - A\boldsymbol{x}\|_2$ without breakdown for all $\boldsymbol{b} \in \mathbf{R}^m$.*

Note that we do not assume A to be full rank in Theorems 3.1 and 3.2.

4. Numerical experiments. Inner-iteration preconditioned BA-GMRES and CGLS with Cimmino-NR, NR-SOR, and NR-SSOR in sections 2 and 3 were compared with the methods preconditioned with the diagonal scaling (diag) and RIF preconditioners[1] in numerical experiments. Note that, theoretically, the RIF preconditioner may break down for rank-deficient problems. In addition, BiCGSTAB applied to the system $B^{(l)}A\boldsymbol{x} = B^{(l)}\boldsymbol{b}$ preconditioned with NR-SOR inner iterations was also compared.

The stopping criterion for the kth outer iteration was

$$\left\|A^{\mathsf{T}}(\boldsymbol{b} - A\boldsymbol{x}_k)\right\|_2 < 10^{-6}\left\|A^{\mathsf{T}}\boldsymbol{b}\right\|_2. \tag{4.1}$$

The CPU time for checking (4.1) was excluded from the total CPU time. The initial solution for the inner and outer iterations was set to zero. No restarts were used for BA-GMRES.

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 $\mathcal{R}(A)$.

All computations were done on a PC workstation with an Intel Xeon X5492 3.4 GHz CPU, 16 GB RAM, Scientific Linux 6.1, and double precision floating-point arithmetic. All programs for the iterative methods in our tests were coded in Fortran 95 and compiled by Intel Fortran Version 12.1.0. For the direct methods, we used MATLAB 2011b.

4.1. Comparisons of CPU time. We compare the CPU time for each method for test problems. Table 4.1 gives information on the test matrices. "nnz" is the number of nonzero elements, "dens" is the density of the nonzero elements. In the last two columns, D time and $\|A^{\mathsf{T}}\boldsymbol{r}\|_2/\|^{\mathsf{T}}\boldsymbol{b}\|_2$ are the CPU time and the resulting relative residual for the direct method. The rank and condition number were computed using the MATLAB functions `spnrank` and `svd`. "-" means the corresponding numerical values could not be computed due to insufficient memory. HIRLAM is from [21], and was used in [5]. Maragal_8 was transposed so that $m > n$.

Table 4.2 gives the least CPU time for the iterative methods to achieve the condition (4.1) for each problem. The first row in each cell gives the number of (outer)

TABLE 4.1
Information on the test matrices.

Matrix	m	n	nnz	dens. [%]	rank	$\kappa_2(A)$	D time	$\|A^{\mathsf{T}}\boldsymbol{r}\|_2/\|A^{\mathsf{T}}\boldsymbol{b}\|_2$
Maragal_6	21,251	10,144	537,694	0.25	8,331	2.91×10^6	355.46	1.13×10^{-1}
Maragal_7	46,845	26,525	1,200,537	0.10	20,843	8.98×10^6	474.88	6.38×10^{-2}
Maragal_8	33,093	60,845	1,308,415	0.06	–	–	67.60	1.34×10^{-9}
HIRLAM	1,385,270	452,200	2,713,200	0.0004	–	–	242.31	1.18×10^{-15}
LargeRegFile	2,111,154	801,374	4,944,201	0.00029	–	–	10.66	8.03×10^{-15}

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

TABLE 4.2
Comparisons of the CPU time.

Solver	CGLS			BA-GMRES				BiCGSTAB
Precond.	diag	RIF	NR-SSOR	diag	RIF	Cimmino-NR	NR-SOR	NR-SOR
Maragal_6	11,645	△	2,249 (1, 0.6)	1,817	△	720 (7, 0.4)	318 (6, 1.5)	▲
▼ 354.46	27.83		15.29	42.20		17.43	* 4.93	
Maragal_7	2,822	△	146 (2, 0.9)	652	△	395 (3, 0.2)	99 (4, 1.3)	71 (3, 1.2)
▼ 474.88	18.84		4.39	17.67		12.71	* 2.45	* 2.45
Maragal_8	$> 10^5$	△	38,497 (1, 0.7)	5,376	△	2781 (5, 0.4)	969 (6, 1.2)	▲
67.60	847.67		757.91	1,181		416.42	* 71.88	
HIRLAM	121	49 (0.2)	19 (1, 1.8)	121	35 (0.1)	58 (2, 0.9)	15 (9, 1.8)	7 (8, 1.6)
242.31	6.92	4.44 (1.34)	* 2.31	23.28	5.55 (2.21)	9.30	4.01	3.26
LargeRegFile	60	54 (0.8)	24 (1, 1.0)	60	46 (0.4)	33 (2, 0.7)	7 (5, 1.4)	9 (3, 1.3)
10.66	4.67	7.99 (3.66)	5.21	13.23	12.65 (3.76)	7.27	* 3.09	4.12

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

iterations outside 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 brackets, and the time in seconds to set up the preconditioning matrix of RIF in brackets.

The drop tolerance of RIF, number of inner iterations, and overrelaxation parameter ω were experimentally optimized to achieve the least CPU time. We used $k \times 10^{-l}$ for the drop tolerance for RIF, where $k = 1, 2, \ldots, 9$ and $l = 1, 2, 3$. We used $l = 0.1, 0.2, \ldots, 1.9$ for the overrelaxation parameter for Cimmino-NR and NR-SOR. For convenience, the CPU time for the direct method is given below the name of each matrix. The ▼ in the first column indicates that the direct method did not achieve the accuracy in (4.1). The △ indicates that the RIF preconditioner broke down with all the values for the tolerance we tested for Maragal_6–8. The ▲ indicates that BiCGSTAB with NR-SOR failed to converge with all the values of parameters. The $*$ indicates the fastest method for each problem.

CGLS with NR-SSOR gave the least CPU time for HIRLAM, BA-GMRES and BiCGSTAB with NR-SOR for Maragal_7, and BA-GMRES with NR-SOR for other problems. The CPU time for BA-GMRES with NR-SOR was less than that for the direct method except for Maragal_8. CGLS with diag did not converged within 10^5 iterations for Maragal_8. On the other hand, the reorthogonalized CGLS method [19] preconditioned with these methods was also tested but did not give less CPU time than the least CPU time for each problem.

Figure 4.1 shows the relative residual $\|A^\mathsf{T}\boldsymbol{r}_k\|_2/\|A^\mathsf{T}\boldsymbol{b}\|_2$ vs. CPU time (right) for LargeRegFile. BA-GMRES with NR-SOR is shown to converge quickly. Whereas the convergence graphs for CGLS-type methods are oscillatory and slow to converge. The convergence curve for CGLS with RIF is shifted to the right by the time required to construct the preconditioner.

Figure 4.2 plots the CPU time required to achieve relative residual (4.1) vs. the overrelaxation parameter ω for BA-GMRES with NR-SOR for LargeRegFile. l denotes the number of inner iterations. The optimum values for the parameter ω and the number of inner iterations l with respect to the CPU time were $\omega = 1.4$ and $l = 5$, respectively.

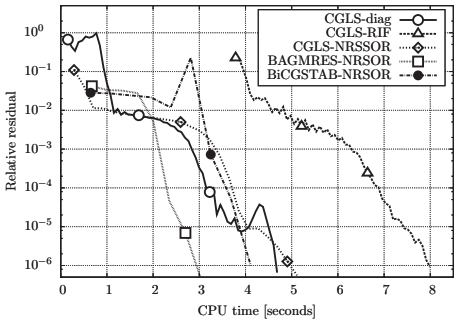

FIG. 4.1. *Relative residual* $\|A^{\mathsf{T}}\boldsymbol{r}_k\|_2/\|A^{\mathsf{T}}\boldsymbol{b}\|_2$ *vs. CPU time for LargeRegFile.*

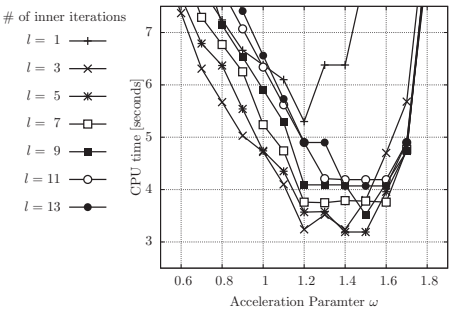

FIG. 4.2. *CPU time to achieve* (4.1) *vs.* ω *for BA-GMRES with NR-SOR for LargeRegFile.*

4.2. Automatic parameter tuning for NR-SOR. NR-SOR has two parameters, the number of inner iterations n_{in} and the overrelaxation parameter ω. As seen in Figures 4.2, the CPU time for BA-GMRES with NR-SOR varies 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 overrelaxation parameter for SOR 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 propose the following procedure, which should be performed before starting the outer iteration. The idea is to perform some test runs of NR-SOR alone beforehand in order to determine the near optimal n_{in} and ω.

1. Set $\omega := 1$.
2. Starting from $n_{\text{in}} := 0$, find the minimum n_{in} that 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} that minimizes $\|\boldsymbol{r}^{(n_{\text{in}})}\|_2$.

Here, η is an parameter, and ω_{opt} is the optimum overrelaxation parameter that minimizes the residual n_{in} Let ω_l's be candidates for ω_{opt}. In the following experiments, we used $\omega_{\text{opt}} = 10^{-l}$, where $l = 0.5, 1.0, 1.5, 2.0$. The tuned n_{in} and ω_{opt} would not be absolutely optimum but would be nearly optimum. Since $\boldsymbol{x}^{(k)}$ and $\boldsymbol{r}^{(k)}$ appear in NR-SOR, the cost for this automatic tuning is marginal.

Table 4.3 gives the numerical results for BA-GMRES with NR-SOR with parameters automatically tuned by the above procedure for the problems presented in Section 4.1. The first row in each cell gives the number of outer iterations outside brackets, and the automatically tuned number of inner iterations and overrelaxation parameter in brackets. The second row gives the total CPU time including the tuning time in seconds outside brackets, and the parameter tuning time in seconds in brackets. The $*$ indicates the fastest case for each problem.

The CPU time for BA-GMRES with NR-SOR with automatically tuned parameters was close to those with optimum parameters given in Table 4.2. In addition, the CPU time given for $\eta = 10^{-1.5}, 10^{-1}, 10^{-0.5}$ was less than CGLS and BA-GMRES with the diagonal scaling (See Table 4.2). Moreover, the CPU time required for tuning the parameter was marginal compared to the total CPU time. The values $\eta = 10^{-0.5}, 10^{-1}$ for the tuning gave (nearly) optimum parameters.

TABLE 4.3
Results with automatically tuned parameters for the test problems.

η	10^{-2}	$10^{-1.5}$	10^{-1}	$10^{-0.5}$	Optimum
Maragal_6	151 (105, 1.9)	222 (50, 1.9)	259 (9, 1.6)	355 (5, 1.4)	318 (6, 1.5)
	27.78 (0.55)	19.78 (0.26)	5.14 (0.09)	* 5.00 (0.07)	4.93
Maragal_7	102 (46, 1.9)	144 (13, 1.7)	165 (7, 1.5)	99 (4, 1.3)	99 (4, 1.3)
	21.84 (0.62)	9.68 (0.29)	6.71 (0.22)	* 2.61 (0.16)	2.45
Maragal_8	438 (140, 1.6)	721 (20, 1.6)	1,223 (4, 1.3)	1,781 (2, 1.1)	969 (6, 1.2)
	334.80 (4.46)	100.27 (0.64)	* 89.82 (0.19)	150.77 (0.12)	71.88

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

5. Conclusions. We studied stationary inner-iteration preconditioned Krylov subspace methods for solving least squares problems. For the inner iterations, Jacobi overrelaxation, SOR, and SSOR-type methods for least squares problems were employed. For the outer iterations, CG and GMRES-type methods were used.

The inner iterations are efficient in terms of 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. We showed the stationary inner-iteration BA-GMRES residual bound.

Numerical experiments including ill-conditioned, rank-deficient, and practical problems, showed that the NR-SOR inner iterations combined with the left-preconditioned (BA) GMRES method is, in terms of CPU time, the most effective method, which outperforms conventional methods.

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

[6] Å. BJÖRCK, *Numerical Methods for Least Squares Problems*, SIAM, Philadelphia, 1996.

[7] Å. BJÖRCK AND T. ELFVING, *Accelerated projection methods for computing pseudoinverse solutions of systems of linear equations*, BIT, 19 (1979), pp. 145–163.

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

[9] X. CUI AND K. HAYAMI, *Generalized approximate inverse preconditioners for least squares problems*, Japan J. Indust. Appl. Math., 26 (2009), pp. 1–14.

[10] X. CUI, K. HAYAMI, AND J.-F. YIN, *Greville's method for preconditioning least squares problems*, Adv. Comput. Math., 35 (2011), pp. 243–269.

[11] A. DAX, *The convergece of linear stationary iterative processes for solving singular unstructured systems of linear equations*, SIAM Review, 32 (1990), pp. 611–635.

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

[13] M. A. DELONG AND J. M. ORTEGA, *SOR as a preconditioner*, Appl. Numer. Math., 18 (1995), pp. 431–440.

[14] ———, *SOR as a preconditioner II*, Appl. Numer. Math., 26 (1998), pp. 465–481.
[15] 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. 1–12.
[16] G. H. Golub and Q. Ye, *Inexact preconditioned conjugate gradient method with inner-outer iteration*, SIAM J. Sci. Comput., 21 (1999), pp. 1305–1320.
[17] A. Greenbaum, *Iterative Methods for Solving Linear Systems*, SIAM, Philadelphia, 1997.
[18] K. Hayami and M. Sugihara, *A geometric view of Krylov subspace methods on singular systems*, Numer. Linear Algebra Appl., 18 (2011), pp. 449–469.
[19] K. Hayami, J.-F. Yin, and T. Ito, *GMRES methods for least squares problems*, SIAM J. Matrix Anal. Appl., 31 (2010), pp. 2400–2430.
[20] M. R. Hestenes and E. Stiefel, *Methods of conjugate gradients for solving linear systems*, J. Research Nat. Bur. Standards, 49 (1952), pp. 409–436.
[21] A. Holstad and I. Lie, *On the computation of mass conservative wind and vertical velocity fields*, Technical report 141, The Norweigian Meteorological Institute, (2002).
[22] A. Jennings and M. A. Ajiz, *Incomplete methods for solving $A^T Ax = b$*, SIAM J. Sci. Stat. Comput., 5 (1984), pp. 978–987.
[23] S. Kaczmarz, *Angenäherte auflösung von systemen linearer gleichungen*, Bull. Acad. Polon. Sciences et Lettres, A (1937), pp. 355–357.
[24] Y. Notay, *Flexible conjugate gradients*, SIAM J. Sci. Comput., 22 (2000), pp. 1444–1460.
[25] C. C. Paige and M. A. Saunders, *LSQR: An algorithm for sparse linear equations and sparse least squares*, ACM Trans. Math. Software, 8 (1982), pp. 43–71.
[26] 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.
[27] Y. Saad, *Preconditioning techniques for nonsymmetric and indefinite linear systems*, J. Comput. Appl. Math., 24 (1988), pp. 89–105.
[28] ———, *A flexible inner-outer preconditioned GMRES algorithm*, SIAM J. Sci. Comput., 14 (1993), pp. 461–469.
[29] ———, *Iterative Methods for Sparse Linear Systems*, 2nd ed., SIAM, Philadelphia, 2003.
[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, *Characterization of linear stationary iterative processes for solving a singular system of linear equations*, Numer. Math., 22 (1974), pp. 349–359.
[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., 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.