Generalized Approximate Inverse Preconditioners for Least Squares Problems

Xiaoke Cui[*] and Ken Hayami[†]

[*] *Department of Informatics, School of Multidisciplinary Sciences*
The Graduate University for Advanced Studies (Sokendai)
E-mail: xkcui@nii.ac.jp
[†] *Principles of Informatics Research Division*
National Institute of Informatics
E-mail: hayami@nii.ac.jp

Received January 29, 2008
Revised August 4, 2008

This paper is concerned with a new approach for preconditioning large sparse least squares problems. Based on the idea of the approximate inverse preconditioner, which was originally developed for square matrices, we construct a generalized approximate inverse (GAINV) M which approximately minimizes $\|I - MA\|_{\mathrm{F}}$ or $\|I - AM\|_{\mathrm{F}}$. Then, we also discuss the theoretical issues such as the equivalence between the original least squares problem and the preconditioned problem. Finally, numerical experiments on problems from Matrix Market collection and random matrices show that although the preconditioning is expensive, it pays off in certain cases.

Key words: approximate inverse, least squares problem, preconditioning, rectangular matrix

1. Introduction

Approximate inverse (AINV) preconditioners [16] were originally developed for solving large sparse linear systems of the form,

$$Ax = b, \tag{1.1}$$

where $A \in \mathbb{R}^{n\times n}$ is a nonsingular matrix and $b \in \mathbb{R}^n$ is a right-hand-side vector. As is well known, the rate of convergence of iterative methods for solving (1.1) is strongly influenced by the spectral properties of A. It is therefore natural to transform the original system into an equivalent system with more favorable spectral properties, and this transformation is called preconditioning. There are many approaches for preconditioning [5]. Some algorithms focus on constructing an approximation to the coefficient matrix A. One of the most popular algorithms in this category is the incomplete LU factorization. On the other hand, there are also algorithms which construct a direct approximation to the inverse of A. One way to accomplish this construction is to find a matrix M which minimizes the following Frobenius norm

$$\min_{M\in\mathcal{S}}\|I - MA\|_{\mathrm{F}} \quad \text{or} \quad \min_{M\in\mathcal{S}}\|I - AM\|_{\mathrm{F}} \tag{1.2}$$

over all $n \times n$ matrices with a certain sparsity pattern $\mathcal{S}$, where I is the $n \times n$ identity matrix. Hence, MA and AM are approximations to an identity matrix, which implies that M is an approximation to the inverse of A. This idea of constructing M by minimizing the Frobenius norm $\|I - AM\|_{\mathrm{F}}$ was first proposed by Benson in his master's thesis [3]. See also Benson and Frederickson [4].

However, it is very difficult to choose a suitable sparsity pattern for M. Hence several authors developed adaptive methods which start from a simple initial nonzero pattern and gradually refine it until $\|I - MA\|_{\mathrm{F}} < \epsilon$ is achieved, where ϵ is a threshold [6, 10, 12, 13]. The most successful of these methods is the one proposed by Grote and Huckle [13], which is called SPAI. Unfortunately, the setup time for adaptive SPAI is often high [1, 2, 6]. Thus, Chow and Saad developed the minimal residual (MR) [9] method so that no nonzero pattern needs to be prescribed in advance. For the left preconditioning case, the algorithm can be written in the following form.

Algorithm 1.1. MR algorithm [16]

```
1. Set $M_0 = \alpha_0 A^{\mathrm{T}}$, $\alpha_0 = \frac{\|A\|_{\mathrm{F}}^2}{\|A^{\mathrm{T}}A\|_{\mathrm{F}}^2}$.
2. **for** $k = 1, 2, \ldots$ **until convergence, do**
3.    Compute $R_{k-1} = I - M_{k-1}A$, and $G_{k-1}$.
4.    Compute $\alpha_k = \mathrm{trace}(R_{k-1}^{\mathrm{T}} G_{k-1} A) / \|G_{k-1}A\|_{\mathrm{F}}^2$.
5.    Compute $M_k := M_{k-1} + \alpha_k G_{k-1}$.
6.    Apply numerical dropping to $M_k$.
7. **end do**
```

In the above algorithm, α_0 is chosen to minimize $\|I - \alpha A^{\mathrm{T}}A\|_{\mathrm{F}}$, and α_k, $k \geq 1$ are chosen to minimize

$$\|I - (M_{k-1} + \alpha G_{k-1})A\|_{\mathrm{F}}. \tag{1.3}$$

One choice for G_k is the residual matrix $R_k = I - M_k A$, and another popular choice is $G_k = (I - M_k A)A^{\mathrm{T}}$, which is the direction of steepest descent.

There is another way to minimize (1.2). Instead of minimizing globally as a function of matrix M, it can be minimized column by column (row by row), as follows:

$$\begin{aligned} \min\|I - MA\|_{\mathrm{F}} &\iff \min\|e_i - m_i A\|_2, \quad i = 1, \ldots, n, \\ \min\|I - AM\|_{\mathrm{F}} &\iff \min\|e^i - Am^i\|_2, \quad i = 1, \ldots, m, \end{aligned}$$

where e_i and m_i are rows of the identity matrix I and M, e^i and m^i are columns of the identity matrix I and M, respectively. The advantage of performing the minimization column by column or row by row is that it can be easily parallelized. For the left preconditioning, the row-oriented algorithm [16] is as follows.

Algorithm 1.2.

```
1.  Set $M_0 = \alpha_0 A^{\mathrm{T}}$, $\alpha_0 = \frac{\|A\|_{\mathrm{F}}^2}{\|A^{\mathrm{T}}A\|_{\mathrm{F}}^2}$.
2.  for $j = 1, \ldots, n$ do
3.     Define $m_j = e_j M$.
4.     for $k = 1, \ldots, n_k$ do
5.        $r_j = e_j - m_j A$
6.        $\alpha_j = \frac{\langle r_j A, r_j \rangle}{\|r_j A\|_2^2}$
7.        $m_j = m_j + \alpha_j r_j$
8.        Apply numerical dropping to $m_j$.
9.     end do
10. end do
```

In this paper, we consider applying Saad's MR algorithm to the least squares problems

$$\min_{x \in \mathbb{R}^n} \|b - Ax\|_2, \tag{1.4}$$

where $A \in \mathcal{R}^{m \times n}$, $\mathrm{rank}(A) = r$, $b \in \mathcal{R}^m$, which to the authors' knowledge, is new. Thus, we aim to construct a preconditioner $M \in \mathbb{R}^{n \times m}$ which minimizes

$$\|I - MA\|_{\mathrm{F}} \quad \text{or} \quad \|I - AM\|_{\mathrm{F}}, \tag{1.5}$$

where I is an $n \times n$ or $m \times m$ identity matrix, respectively. Since now matrices A and M_k are rectangular, we cannot choose the correction matrix G_k as $R_k = I - M_k A$. Hence, we let $G_k = (I - M_k A)A^{\mathrm{T}}$. We will also give mathematical justifications for applying the method to least squares problems.

The rest of the paper is organized as follows. In Section 2, we discuss for the left preconditioning, under what condition we can use this generalized approximate inverse to precondition a least squares problem. Similar theoretical results for the right preconditioning are developed in Section 3. Finally, we present numerical results in Section 4. The results suggest that although the preconditioning is expensive, good convergence behavior can be achieved, so that it becomes a competitive option for solving least squares problems with multiple right-hand-sides.

2. Left preconditioning

Consider solving the least squares problem (1.4) by transforming it into a left preconditioned form,

$$\min_{x \in \mathbb{R}^n} \|Mb - MAx\|_2, \tag{2.1}$$

where $A \in \mathbb{R}^{m \times n}$, $M \in \mathbb{R}^{n \times m}$, and b is a right-hand-side vector $b \in \mathbb{R}^m$.

For preconditioning, one important issue is whether the solution of the preconditioned problem is the solution of the original problem. For square nonsingular linear systems, the condition for this equivalence is that the preconditioner M should be nonsingular. Since we are dealing with rectangular problems, we need some other conditions to ensure that the preconditioned problem (2.1) is equivalent to the original least squares problem (1.4).

First note the following [14], where $\mathcal{R}(X)$ denotes the range space of matrix X.

LEMMA 2.1.

$$\|b - Ax^*\|_2 = \min_{x\in\mathbb{R}^n} \|b - Ax\|_2$$

and

$$\|Mb - MAx^*\|_2 = \min_{x\in\mathbb{R}^n} \|Mb - MAx\|_2$$

are equivalent for all $b \in \mathbb{R}^m$, *if and only if* $\mathcal{R}(A) = \mathcal{R}(M^{\mathrm{T}}MA)$.

Hence, in order that the preconditioned problem (2.1) is equivalent with the original problem (1.4), the matrix M of Algorithm 1.1 should satisfy the condition in Lemma 2.1.

In order to analyze this condition, we rewrite Algorithm 1.1 for left preconditioning on the rectangular matrix A as follows.

Algorithm 2.1.

```
1. Set $M_0 = \alpha_0 A^{\mathrm{T}}$, $\alpha_0 = \frac{\|A\|_{\mathrm{F}}^2}{\|A^{\mathrm{T}}A\|_{\mathrm{F}}^2}$.
2. for $k = 1, 2, \ldots$ until convergence, do
3.    Compute $R_{k-1} = I - M_{k-1}A$.
4.    Compute $G_{k-1} = R_{k-1}A^{\mathrm{T}}$.
5.    Compute $\alpha_k = \|G_{k-1}\|_{\mathrm{F}}^2 / \|G_{k-1}A\|_{\mathrm{F}}^2$.
6.    Compute $M_k := M_{k-1} + \alpha_k G_{k-1}$.
7.    Apply numerical dropping to $M_k$.
8. end do
```

In the above Algorithm 2.1, $M_0 = \alpha_0 A^{\mathrm{T}}$, where α_0 minimizes $\|I - \alpha A^{\mathrm{T}}A\|_{\mathrm{F}}$ over all real scalar α. Hence, we have

$$\begin{aligned} M_1 &= M_0 + \alpha_1 G_0 \\ &= M_0 + \alpha_1 (I - M_0 A) A^{\mathrm{T}} \\ &= (\alpha_0 + \alpha_1 - \alpha_0\alpha_1 A^{\mathrm{T}}A) A^{\mathrm{T}} \\ &= p_1(A^{\mathrm{T}}A) A^{\mathrm{T}}, \end{aligned}$$

where $p_k(\,\cdot\,)$ is a polynomial of degree k. Similarly, if we assume $M_{k-1} = p_{k-1}(A^{\mathrm{T}}A)A^{\mathrm{T}}$, then for M_k, we have

$$\begin{aligned} M_k &= M_{k-1} + \alpha_k G_{k-1} \\ &= M_{k-1} + \alpha_k (I - M_{k-1}A)A^{\mathrm{T}} \\ &= M_{k-1}(I - \alpha_k AA^{\mathrm{T}}) + \alpha_k A^{\mathrm{T}} \\ &= (p_{k-1}(A^{\mathrm{T}}A)(I - \alpha_k A^{\mathrm{T}}A) + \alpha_k)A^{\mathrm{T}} \\ &\equiv p_k(A^{\mathrm{T}}A)A^{\mathrm{T}}. \end{aligned}$$

Combining all the above argument, we have the following.

THEOREM 2.1. *If no numerical droppings are performed,* M_k *in* Algorithm 2.1 *can be expressed as* $M_k = p_k(A^{\mathrm{T}}A)A^{\mathrm{T}}$*, where* $p_k(\,\cdot\,)$ *is a polynomial of degree* k*,* p_0 *is the scalar* α_0 *defined in* Algorithm 2.1*.*

By expressing M_k in the form $M_k = p_k(A^{\mathrm{T}}A)A^{\mathrm{T}}$, we can easily deduce a condition for the equivalence between the preconditioned problem (2.1) and the original problem (1.4) as follows.

THEOREM 2.2. *The preconditioned problem* (2.1) *is equivalent to the original problem* (1.4) *if and only if* $p_k(\sigma_i^2) \neq 0$ *for all singular values* $\sigma_i > 0$ *of* A*.*

Proof. By Lemma 2.1, we only need to prove $\mathcal{R}(A) = \mathcal{R}(M_k^{\mathrm{T}}M_kA)$. Since

$$\begin{aligned} \mathcal{R}(M_k^{\mathrm{T}}M_kA) &= \mathcal{R}(Ap_k(A^{\mathrm{T}}A)p_k(A^{\mathrm{T}}A)A^{\mathrm{T}}) \\ &\subseteq \mathcal{R}(A), \end{aligned}$$

$\mathcal{R}(A) = \mathcal{R}(M_k^{\mathrm{T}}M_kA)$ is equivalent to

$$\operatorname{rank}(A) = \operatorname{rank}(M_k^{\mathrm{T}}M_kA). \tag{2.2}$$

Assume that the SVD of A is $A = U\Sigma V^{\mathrm{T}}$, where U is an $m \times m$ orthogonal matrix, V is an $n \times n$ orthogonal matrix, and $\Sigma = \operatorname{diag}\{\sigma_1, \ldots, \sigma_r, 0, \ldots, 0\}_{m\times n}$, $\sigma_i > 0$, $i = 1, \ldots, r$. Then,

$$\begin{aligned} M_k^{\mathrm{T}}M_kA &= Ap_k(A^{\mathrm{T}}A)p_k(A^{\mathrm{T}}A)A^{\mathrm{T}}A \\ &= U\Sigma p_k^2(\Sigma^{\mathrm{T}}\Sigma)\Sigma^{\mathrm{T}}\Sigma V^{\mathrm{T}} \\ &= U \operatorname{diag}\{\sigma_1^3 p_k^2(\sigma_1^2), \ldots, \sigma_r^3 p_k^2(\sigma_r^2), 0, \ldots, 0\}_{m\times n} V^{\mathrm{T}}. \end{aligned}$$

Hence, (2.2) is equivalent to $p_k(\sigma_i^2) \neq 0$ for $\sigma_i > 0$, $i = 1, \ldots, r$. □

It is difficult to prove that $p_k(\sigma_i^2) \neq 0$ for $\sigma_i > 0$, $i = 1, \ldots, r$. However, we can assume that it holds generically, i.e., the probability of $p_k(\sigma_i^2) = 0$ for any $1 \leq i \leq r$ is zero. Also in our numerical experiments, we never observed $p_k(\sigma_i^2) = 0$ to happen.

Besides the above equivalence theorem, we are also concerned whether any breakdown may occur when we solve the preconditioned problem (2.1) using Krylov subspace methods. Brown and Walker [8] gave a necessary and sufficient condition such that the GMRES method [17] does not break down. Let $\mathcal{N}(X)$ denote the null space of X.

Lemma 2.2. *The GMRES method determines a least squares solution of* $Ax = b$, $A \in R^{n\times n}$, *without breakdown for arbitrary* $b \in R^n$ *and initial guess* $x_0 \in R^n$ *if and only if* $\mathcal{N}(A) = \mathcal{N}(A^{\mathrm{T}})$.

Hence, for the GMRES method, we have the following.

Theorem 2.3. *For* M_k *in* Algorithm 2.1*,* M_kA *is symmetric, so that the GMRES method determines a least squares solution of the preconditioned problem* $\min_{x\in\mathcal{R}^n}\|M_kb - M_kAx\|_2$ *without breakdown for arbitrary* $b \in R^m$ *and initial guess* $x_0 \in R^n$.

Proof. The proof follows directly from Theorem 2.1 and Lemma 2.2. Since $M_k = p_k(A^{\mathrm{T}}A)A^{\mathrm{T}}$, M_kA is symmetric, which implies $\mathcal{N}(M_kA) = \mathcal{N}((M_kA)^{\mathrm{T}})$. □

From Theorem 2.2 and Theorem 3.3, the GMRES method can be used to solve the preconditioned least squares problem (2.1) with the preconditioner M_k from Algorithm 2.1, to obtain a least squares solution to the original least squares problem (1.4) without breakdown. Moreover, since M_kA is symmetric, the MINRES method [15], which is equivalent to the GMRES method for symmetric matrices and uses short recurrences, can be used instead to save computation time and memory.

Remark 2.1. In Theorem 2.1 we assume that there is no numerical droppings performed so that we have Theorem 2.2 and Theorem 3.3. When the numerical dropping strategy is used, M_k cannot be written in the polynomial form $p_k(A^{\mathrm{T}}A)A^{\mathrm{T}}$ as we show in the following.

In Algorithm 2.1, $M_0 = \alpha_0A^{\mathrm{T}}$, where α_0 minimizes $\|I - \alpha A^{\mathrm{T}}A\|_{\mathrm{F}}$ over all real scalars α. When A is sparse, we do not need to do numerical droppings for M_0. Denote the dropped part in the process of computing M_i as E_i, Hence, we have

$$\begin{aligned}
M_1 &= M_0 + \alpha_1(I - M_0A)A^{\mathrm{T}} - E_1\\
&= p_1(A^{\mathrm{T}}A)A^{\mathrm{T}} - E_1,\\
M_2 &= M_1 + \alpha_2(I - M_1A)A^{\mathrm{T}} - E_2\\
&= p_1(A^{\mathrm{T}}A)A^{\mathrm{T}} - E_1 + \alpha_2(I - p_1(A^{\mathrm{T}}A)A^{\mathrm{T}}A - E_1A)A^{\mathrm{T}} - E_2\\
&= p_1(A^{\mathrm{T}}A)A^{\mathrm{T}} + \alpha_2(I - p_1(A^{\mathrm{T}}A)A^{\mathrm{T}}A)A^{\mathrm{T}} - E_1 - \alpha_2E_1AA^{\mathrm{T}} - E_2\\
&= p_2(A^{\mathrm{T}}A)A^{\mathrm{T}} - E_1 - E_2 - \alpha_2E_1AA^{\mathrm{T}},\\
&\cdots
\end{aligned}$$

where $p_k(\,\cdot\,)$ is a polynomial of degree k.

According to the above discussion, we cannot ensure the equivalence and the breakdown free theorems when numerical droppings are used. However, in our numerical experiments, when the dropping threshold is not too large, we did not encounter breakdown of GMRES.

The row-oriented Algorithm 1.2 can also be modified to be applied to rectangular matrices.

Algorithm 2.2.

```
1. Set $M_0 = \alpha_0 A^{\mathrm{T}}$, $\alpha_0 = \frac{\|A\|_{\mathrm{F}}^2}{\|A^{\mathrm{T}}A\|_{\mathrm{F}}^2}$.
2. for $j = 1, \ldots, n$ do
3.    Define $m_j = e_j M$.
4.    for $i = 1, \ldots, n_i$ do
5.       $r_j = e_j - m_j A$
6.       $g_j = r_j A^{\mathrm{T}}$
7.       $\alpha_j = \frac{\|g_j\|_2^2}{\|r_j A^{\mathrm{T}} A\|_2^2}$
8.       $m_j = m_j + \alpha_j g_j$
9.       Apply numerical dropping to $m_j$.
10.   end do
11. end do
```

However, it is difficult to show equivalence theorems for this row-oriented method.

3. Right preconditioning

So far we have discussed left preconditioning. For over-determined problems, i.e., $A \in \mathbb{R}^{m\times n}$, $m > n$, left preconditioning is more favorable, since the size of the preconditioned matrix $M_k A$ is $n \times n$. However, if we are considering an under-determined problem, i.e., $A \in \mathbb{R}^{m\times n}$, $m < n$, right preconditioning is more suitable. Results analogous to the left preconditioning hold for the right preconditioning case.

When we precondition the original least squares problem (1.4) from the right-hand-side, we have,

$$\min_{y\in\mathbb{R}^m} \|b - AMy\|_2. \tag{3.1}$$

First note the following [14].

Lemma 3.1. *$\min_{x\in\mathbb{R}^n} \|b - Ax\|_2 = \min_{z\in\mathbb{R}^m} \|b - AMz\|_2$ holds for all $b \in \mathbb{R}^m$ if and only if $\mathcal{R}(A) = \mathcal{R}(AM)$.*

Next, we rewrite Algorithm 2.1 for right preconditioning as follows.

Algorithm 3.1.

```
1. Set $M_0 = \alpha_0 A^{\mathrm{T}}$, $\alpha_0 = \frac{\|A\|_{\mathrm{F}}^2}{\|AA^{\mathrm{T}}\|_{\mathrm{F}}^2}$.
2. for $k = 1, 2, \ldots$ until convergence, do
3.    Compute $R_{k-1} = I - AM_{k-1}$.
4.    Compute $G_{k-1} = A^{\mathrm{T}} R_{k-1}$.
5.    Compute $\alpha_k = \|G_{k-1}\|_{\mathrm{F}}^2 / \|AG_{k-1}\|_{\mathrm{F}}^2$.
6.    Compute $M_k := M_{k-1} + \alpha_k G_{k-1}$.
7.    Apply numerical dropping to $M_k$.
8. end do
```

Similar to the left preconditioning case, M_k from Algorithm 3.1 also has a polynomial form.

Theorem 3.1. *M_k in* Algorithm 3.1 *can be expressed as $M_k = A^{\mathrm{T}} p_k(AA^{\mathrm{T}})$, where $p_k(\,\cdot\,)$ is a polynomial of degree k, p_0 is a scalar α_0 defined in* Algorithm 3.1.

Proof. According to Algorithm 3.1,

$$\begin{aligned} M_0 &= \alpha_0 A^{\mathrm{T}} \\ &\equiv A^{\mathrm{T}} p_0(AA^{\mathrm{T}}), \\ M_1 &= M_0 + \alpha_1 G_0 \\ &= \alpha_0 A^{\mathrm{T}} + \alpha_1 A^{\mathrm{T}}(I - \alpha_0 AA^{\mathrm{T}}) \\ &= A^{\mathrm{T}}(\alpha_0 + \alpha_1 - \alpha_0\alpha_1 AA^{\mathrm{T}}) \\ &\equiv A^{\mathrm{T}} p_1(A^{\mathrm{T}}A). \end{aligned}$$

Thus assume M_{k-1} can be expressed as $M_{k-1} = A^{\mathrm{T}} p_{k-1}(AA^{\mathrm{T}})$. Then,

$$\begin{aligned} M_k &= M_{k-1} + \alpha_k G_{k-1} \\ &= A^{\mathrm{T}} p_{k-1}(AA^{\mathrm{T}}) + \alpha_k A^{\mathrm{T}}(I - AA^{\mathrm{T}} p_{k-1}(AA^{\mathrm{T}})) \\ &= A^{\mathrm{T}}((I - \alpha_k AA^{\mathrm{T}}) p_{k-1}(AA^{\mathrm{T}}) + \alpha_k) \\ &\equiv A^{\mathrm{T}} p_k(AA^{\mathrm{T}}). \end{aligned}$$ □

Combining this Theorem 3.1 and Lemma 3.1, we get the following equivalence theorem for right preconditioning.

Theorem 3.2. *The preconditioned problem* (3.1) *is equivalent to the original problem* (1.4) *if and only if $p_k(\sigma_i^2) \neq 0$ for all singular values $\sigma_i > 0$ of A.*

Proof. By Lemma 3.1, we only need to prove $\mathcal{R}(A) = \mathcal{R}(AM_k)$. Since

$$\begin{aligned} \mathcal{R}(AM_k) &= \mathcal{R}(AA^{\mathrm{T}} p_k(AA^{\mathrm{T}})) \\ &\subseteq \mathcal{R}(A), \end{aligned}$$

$\mathcal{R}(A) = \mathcal{R}(AM_k)$ is equivalent to

$$\operatorname{rank}(A) = \operatorname{rank}(AM_k). \tag{3.2}$$

Let the SVD of A be $A = U\Sigma V^{\mathrm{T}}$, where U is an $m \times m$ orthogonal matrix, V is an $n \times n$ orthogonal matrix, and $\Sigma = \mathrm{diag}\{\sigma_1, \ldots, \sigma_r, 0, \ldots, 0\}_{m\times n}$, $\sigma_i > 0$, $i = 1, \ldots, r$. Then,

$$\begin{aligned} AM_k &= AA^{\mathrm{T}} p_k(AA^{\mathrm{T}}) \\ &= U\Sigma\Sigma^{\mathrm{T}} p_k(\Sigma\Sigma^{\mathrm{T}})U^{\mathrm{T}} \\ &= U\,\mathrm{diag}\{\sigma_1^2 p_k(\sigma_1^2), \ldots, \sigma_r^2 p_k(\sigma_r^2), 0, \ldots, 0\}_{m\times m} U^{\mathrm{T}}. \end{aligned}$$

Now, it is easy to see that the equation (3.2) holds if and only if $p_k(\sigma_i^2) \neq 0$ for $\sigma_i > 0$, $i = 1, \ldots, r$. $\square$

Again, we may expect that $p_k(\sigma_i^2) \neq 0$ for $\sigma_i > 0$, $i = 1, \ldots, r$ holds generically.

Combining Lemma 2.2 and Theorem 3.1, we also obtain a breakdown free theorem for the right preconditioning case.

THEOREM 3.3. *For M_k in* Algorithm 3.1, *AM_k is symmetric, so that the GMRES method determines a least squares solution of the preconditioned problem $\min_{x\in\mathcal{R}^m} \|b - AM_k x\|_2$ without breakdown for arbitrary $b \in R^m$ and initial guess $x_0 \in R^n$.*

Proof. The proof follows directly from Theorem 3.1 and Lemma 2.2. Since $M_k = A^{\mathrm{T}} p_k(AA^{\mathrm{T}})$, AM_k is symmetric, which implies $\mathcal{N}(AM_k) = \mathcal{N}((AM_k)^{\mathrm{T}})$. $\square$

4. Numerical Results

In this section, we compare our preconditioning method with well known methods, i.e., CGLS and the diagonally scaled CGLS method [7]. Table 1 provides some basic information about the test matrices. In the table, m is the number of the rows, n is the number of columns, *nnz* is the total number of nonzeros, *density* is the density of the nonzeros in the matrices, *cond* is the condition number of the matrices. The first matrix illc1850 was taken from the Matrix Market [11], and the matrix sprandn8L and sprand8S are random matrices generated by the MATLAB command: `sprandn`.

Table 1. Test matrices

	Origin	m	n	*nnz*	*density*	*cond*
illc1850	Matrix Market	1850	712	8636	0.007	10^3
sprandn8L	Random matrix	10000	1000	487816	0.0488	10^8
sprandn8S	Random matrix	2000	500	48788	0.0488	10^8

All computations were done on the IBM Thinkpad T60 (CPU 2 GHz, 997 MHz 1 GB RAM) with MATLAB 7.5. All the computation times in the tables are results of averaging a hundred runs.

For the first matrix, we use a random right-hand-side vector, which is generated by MATLAB, and the problem is inconsistent. The initial guess was set to $x_0 = 0$. The convergence criterion we used for this problem is

$$\|A^{\mathrm{T}}(b - Ax_k)\|_2 < 10^{-8}\|A^{\mathrm{T}}b\|_2.$$

The time to compute $\|A^{\mathrm{T}}(b - Ax_k)\|_2$ was neglected in all the iteration times. The numerical results are given in Table 2. In Table 2, no numerical dropping was performed, i.e., no nonzero elements were neglected in M_k. Hence, M_kA is symmetric, and we can use the MINRES method instead of the GMRES method to solve the preconditioned problems (2.1).

Table 2. Results for illc1850

Method	k	ITS	Pre. T	Its. T	Tot. T
Global Pre(k) + GMRES	0	700	1.6e − 2	4.25	4.27
	1	661	9.40e − 2	4.59	4.69
	2	573	2.81e − 1	3.86	4.14
	3	516	6.10e − 1	3.50	4.11
	4	476	8.75e − 1	3.04	3.92
	5	445	1.14	2.82	3.96
	10	354	2.44	1.94	4.38
Global Pre(k) + MINRES	0	1904	1.6e − 2	1.84e − 1	2.00e − 1
	1	1317	9.40e − 2	9.40e − 2	1.88e − 1*
	2	1066	2.81e − 1	6.24e − 2	3.43e − 1
	3	802	6.10e − 1	3.42e − 2	6.44e − 1
	4	796	8.75e − 1	3.76e − 2	9.13e − 1
	5	663	1.14	3.12e − 2	1.17
Row-oriented(k) + GMRES	1	658	7.5e − 1	4.54	5.29
	2	553	1.35	3.65	5.00
	3	485	2.03	3.15	5.17
	4	449	2.21	2.78	4.99
	5	418	2.29	2.44	4.73
	10	333	2.72	1.75	4.48
CGLS		2083	0.00	3.26e − 1	3.26e − 1
Diag-CGLS		2081	5.31e − 3	3.66e − 1	3.72e − 1

In Table 2, we choose different k in Algorithm 2.1 to precondition GMRES and MINRES. In the table, 'ITS' is the number of iterations for the algorithm to reach convergence, 'Pre. T' is the preconditioning time, 'Its. T' is the iteration time, and 'Tol. T' is the total time, in seconds, respectively. The asterisk $*$ indicates the shortest total time in the table. According to the table, we observe that as k increases, the number of iterations decrease significantly for both MINRES and GMRES. All the preconditioners M_k can achieve better iteration numbers than CGLS and diagonal scaled CGLS. The MINRES preconditioned by Algorithm 2.1

with $k = 1$ was the fastest in computation time. Comparing MINRES and GMRES, the GMRES required less number of iterations, but the iteration time was longer than MINRES. The reason is that GMRES is more robust against rounding error, but at the cost of the more expensive Gram–Schmidt process. Comparing the global preconditioner and the row-oriented preconditioner, the row-oriented preconditioner required less iterations. However, it requires more preconditioning time than global preconditioners do. When $k = 0$, the global preconditioner and row-oriented preconditioner give the same M_0, thus we did not list the result of $k = 0$ for row-oriented preconditioner.

In Table 3 , we gave results for an over-determined problem sprandn8L, which is larger, much more ill-conditioned and denser. We chose b as $A[1, \dots, 1]^{\mathrm{T}}$ as the right-hand-side vector, so that the problem is consistent. The convergence criterion was chosen as

$$\|b - Ax_k\|_2 < 10^{-6}\|b\|_2.$$

Table 3. Results for sprandn8L

Method	k	ITS	Pre. T	Its. T	Tot. T
Global Pre(k) + GMRES, $\tau_{\text{global}} = 10^{-5}$	0	840	4.80e − 1	1.17e + 1	1.22e + 1*
	1	847	7.18	1.19e + 1	1.91e + 1
	2	857	1.53e + 1	1.23e + 1	2.75e + 1
	3	865	2.36e + 1	1.25e + 1	3.61e + 1
Row-oriented(k) + GMRES, $\tau_{\text{row-oriented}} = 10^{-4}$	1	890	2.83e + 1	1.34e + 1	4.17e + 1
	2	882	4.09e + 1	1.42e + 1	5.51e + 1
	3	876	4.85e + 1	1.44e + 1	6.28e + 1
CGLS		50000+	0.00	2.54e + 2	2.54e + 2
Diag-CGLS		21548	1.60e − 2	1.10e + 2	1.10e + 2

Since this problem is larger than the first problem, the numerical dropping strategy was used, which implies that $M_k A$ is not symmetric and the MINRES method cannot be employed. For the global preconditioner we dropped the (i, j) element in $G = (I - MA)A^{\mathrm{T}}$ when

$$|G(i, j)| < \tau_{\text{global}}\|G\|_1$$

holds. For the row-oriented preconditioner, we dropped the i-th element in $g_j = (e_j - m_j A)A^{\mathrm{T}}$ when

$$|g_j(i)| < \tau_{\text{row-oriented}}\|g_j\|_2$$

holds. The notations are the same as the ones used in Algorithm 2.2. The numerical results are given in Table 3. Compared to the global preconditioner, the row-oriented preconditioner usually gives a denser M_k. Thus, the preconditioning time is much longer than that of the global preconditioner. In the table, we set τ_{global}

to 10^{-5}, and $\tau_{\text{row-oriented}}$ to 10^{-4}, since they are nearly optimal and give M_k with approximately the same density.

Table 3 shows that increasing k does not necessarily result in improved convergence for the global method with droppings. On the other hand, for the row-oriented preconditioner the convergence improves as k is increased. However, the numbers of iterations are more than the global preconditioner. For this extremely ill-conditioned matrix A, the convergence criterion is stricter than $\|A^{\mathrm{T}}(b-Ax_k)\|_2 < 10^{-8}\|A^{\mathrm{T}}b\|_2$. The criterion $\|b - Ax_k\|_2 < 10^{-6}\|b\|_2$ is approximately equivalent to $\|A^{\mathrm{T}}(b - Ax_k)\|_2 < 10^{-12}\|A^{\mathrm{T}}b\|_2$, and the convergence behavior becomes somewhat irregular when a very accurate solution is required.

From Table 2 to Table 3, we observe that the proposed preconditioning is time-consuming compared to the iteration time especially for large k. Improvement in the iteration time cannot compensate the expense for preconditioning. However, when dealing with multi-right-hand-side problems, the CPU time spent on preconditioning pays off. In Table 4, we solved a multiple right-hand-side problem. The coefficient matrix A is the matrix sprand8S of Table 1. The right-hand-side vectors were given by A times a series of random vectors which were also generated by MATLAB.

Table 4. Results for multiple right-hand-side problem (sprandn8S)

Number of subproblems	$k=0$	$k=1$	$k=2$	$k=6$	$k=7$	Diag-CGLS
1	1.453	2.798	3.734	6.704	7.357	7.190e − 1*
25	4.014e + 1	2.946e + 1	2.725e + 1	2.312e + 1	2.174e + 1	1.998e + 1*
50	7.969e + 1	5.670e + 1	5.232e + 1	4.057e + 1	3.720e + 1*	3.981e + 1
75	1.189e + 2	8.448e + 1	7.803e + 1	5.802e + 1	5.262e + 1*	5.903e + 1
100	1.583e + 2	1.128e + 2	1.020e + 2	7.460e + 1	6.738e + 1*	7.867e + 1

In Table 4, we give the total computation time in seconds for solving the least squares problems with different number of right-hand-side vectors b. The global left preconditioning with the MINRES method was used. We can observe that as the number of the right-hand-side vectors increases, the preconditioners become more and more competitive. This is also shown in Fig. 1.

From Fig. 1, we see that as k increases, the preconditioning time becomes larger, but the iteration time per problem decreases. When k keeps increasing to 8, the iteration time per problem starts to increase. Hence, there is an optimal k which minimizes the total CPU time. For this problem, the optimal k is 7.

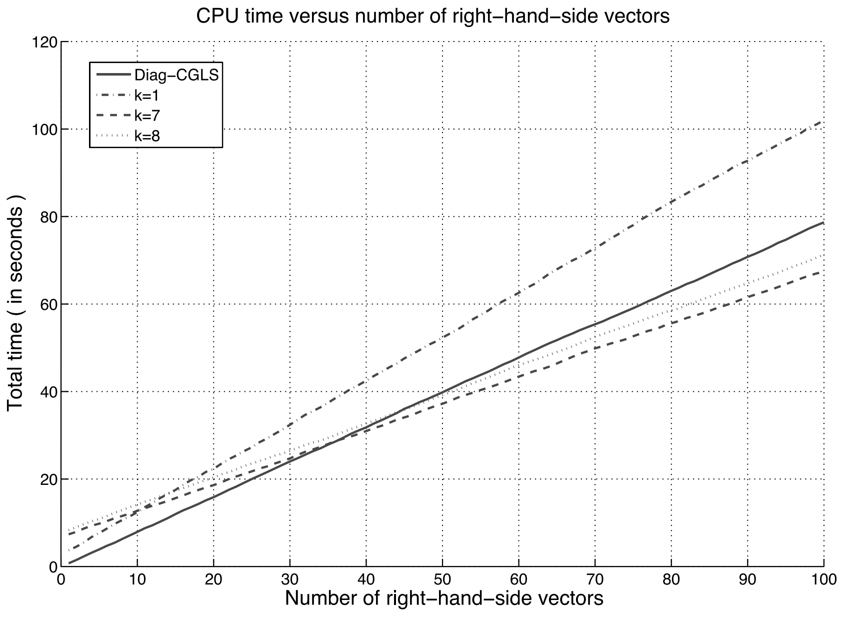

Fig. 1. Multiple right-hand-side problem

5. Conclusion

We applied the approximate inverse preconditioner to least squares problem. Based on the preconditioner M_k from the MR algorithm, we gave equivalence theorems and breakdown free theorems for both the left preconditioning case and the right preconditioning case.

Numerical experiments showed that with the above preconditioning, the MINRES method achieves a faster convergence for solving least squares problems, although the preconditioning is time-consuming. However, for multiple right-hand-side problems, the CPU time spent on preconditioning pays off.

References

[1] R.E. Bank and C. Wagner, Multilevel ILU decomposition. Numer. Math., **82** (1999), 543–574.

[2] S.T. Barnard, L.M. Bernardo and H.D. Simon, An MPI implementation of the SPAI preconditioner on the T3E. Int. J. High Perform. Comput. Appl., **13** (1999), 107–128.

[3] M.W. Benson, Iterative Solution of Large Scale Linear Systems. Master's Thesis, Lakehead Universeity, Thunder Bay, ON, Canada, 1973.

[4] M.W. Benson and P.O. Frederickson, Iterative solution of large sparse linear systems arising in certain multidimensional approximataion problems. Utilitas Math., **22** (1982), 127–140.

[5] M. Benzi, Preconditioning techniques for large linear systems: a survey. J. Comp. Phy., **182** (2002), 418–477.

[6] M. Benzi and M. Tůma, A comparative study of sparse approximate inverse preconditioners. Appl. Numer. Math., **30** (1999), 305–340.

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

[8] P.N. Brown and H.F. Walker, GMRES on (nearly) singular system. SIAM J. Matrix Anal. Appl., **18** (1997), 37–51.

[9] E. Chow and Y. Saad, Approximate inverse preconditioners via sparse-sparse iterations. SIAM J. Sci. Comput, **19** (1998), 995–1023.

[10] J.D.F. Cosgrove, J.C. Ďaz and A. Griewank, Approximate inverse preconditioning for sparse linear systems. Int. J. Comput. Math., **44** (1992), 91–110.

[11] I.S. Duff, R.G. Grimes and J.G. Lewis, Sparse matrix test problems. ACM Trans. Math. Software, **15** (1989), 1–14.

[12] N.I.M. Gould and J.A. Scott, Sparse approximate-inverse preconditioners using norm-minimization techniques. SIAM J. Sci. Comput., **19** (1998), 605–625.

[13] M. Grote and T. Huckle, Parallel preconditioning with sparse approximate inverses. SIAM J. Sci. Comput., **18** (1997), 838–853.

[14] K. Hayami, J.-F. Yin and T. Ito, GMRES methods for least squares problems. NII Technical Report, NII-2007-009E, July 2007, 1–28.

[15] C.C. Paige and M.A. Saunders, Solution of sparse indefinite systems of linear equations. SIAM J. Numer. Anal., **12** (1975), 617–629.

[16] Y. Saad, Iterative Methods for Sparse Linear Systems (2nd edition). SIAM, Philadelphia, 2003.

[17] Y. Saad and M.H. Schultz, GMRES: a generalized minimal residual method for solving nonsymmetric linear systems. SIAM J. Sci. Statist. Comput., **7** (1986), 856–869.