Proceedings of ALGORITMY 2009
pp. 440–448

GREVILLE'S METHOD FOR PRECONDITIONING LEAST SQUARES PROBLEMS

XIAOKE CUI*, KEN HAYAMI†, AND J-F YIN‡

Abstract. In this talk, we present a preconditioner for least squares problems $\min \|b - Ax\|_2$, where A can be matrices with any shape or rank. When A is rank deficient, our preconditioner will be rank deficient too. The preconditioner itself is a sparse approximation to the Moore-Penrose inverse of the coefficient matrix A. We will also discuss the similarity between this preconditioner and the Robust Incomplete Factorization preconditioner [1].

Key words. Least Squares Problem, rank deficient, Preconditioning, Moore-Penrose Inverse, Greville Algorithm, GMRES

AMS subject classifications. 15A09, 65F10, 65F20, 65F30, 65F50, 93E24.

1. Introduction.

Consider the least squares problem,

$$\min_{x \in R^n} \|b - Ax\|_2, \tag{1}$$

where $A \in R^{m \times n}$, $b \in R^m$.

When A is large and sparse, iterative methods are preferred for solving (1). In [8], Hayami et al. proposed using use GMRES [14] to solve least squares problems by using some preconditioners. If we have a preconditioner $B \in R^{n \times m}$ and we precondition (1) from the left, we can transform problem (1) to

$$\min_{x \in R^n} \|Bb - BAx\|_2. \tag{2}$$

On the other hand, we can also precondition problem (1) from the right and transform the problem (1) to

$$\min_{y \in R^m} \|b - ABy\|_2. \tag{3}$$

When A is a nonsingular matrix, one way to precondition (1) is to construct B to be an approximation to the inverse of A. Thus B is called Approximate Inverse(AINV) Preconditioners [13], which were originally developed for solving large sparse linear systems of the form

$$Ax = b, \tag{4}$$

where A is square and nonsingular. Since here we assume that A is rectangular and not necessarily full rank, we consider how to construct a preconditioner M, which is an approximation to the Moore-Penrose inverse[12] of A, and use M to precondition the least squares problem (1).

*Department of Informatics, School of Multidisciplinary Sciences The Graduate University for Advanced Studies (Sokendai), 2-1-2, Hitotsubashi, Chiyoda-ku, Tokyo, Japan, 101-8430 (`xkcui@nii.ac.jp`).
†Principles of Informatics Research Division, National Institute of Informatics, 2-1-2, Hitotsubashi, Chiyoda-ku, Tokyo, Japan, 101-8430.
‡Department of Mathematics, Tongji University 1239 Siping Road, Shanghai, P. R. China.

1.1. Greville's Method. Greville's method [7] is an old method for computing the Moore-Penrose inverse of a matrix A. It is based on the following idea. Given a rectangular matrix $A \in R^{m\times n}$, rank(A) = r $\le$ min{m, n} and A's Moore-Penrose inverse $A^\dagger$, how can we compute the Moore-Penrose inverse of

$$A + cd^T, \quad c \in R^m, \quad d \in R^n, \tag{5}$$

which is a rank-one update of A?

To compute the Moore-Penrose inverse of A, we write A in the following summation form,

$$A = \sum_{i=1}^{n} a_i e_i^T, \tag{6}$$

where a_i is the ith column of A, e_i is the ith column of an identity matrix of order m. Further let $A_i = [a_1, \ldots, a_i, 0, \ldots, 0]$. Hence we have

$$A_i = \sum_{k=1}^{i} a_i e_i^T, \quad i = 1, \ldots, n, \tag{7}$$

and if we denote $A_0 = 0_{m\times n}$, then $A_i = A_{i-1} + a_i e_i^T, \quad i = 1, \ldots, n$. Thus every A_i, $i = 1, \ldots, n$ is a rank-one update of A_{i-1}. Noticing that $A_0^\dagger = 0_{n\times m}$, we can use the following formula to compute the Moore-Penrose inverse of A_i, and in the end we obtain $A_n^\dagger$, which is $A^\dagger$.

$$A_i^\dagger = \begin{cases} A_{i-1}^\dagger + (e_i - A_{i-1}^\dagger a_i)((I - A_{i-1}A_{i-1}^\dagger)a_i)^\dagger & \text{if} \quad a_i \notin \mathcal{R}(A_{i-1}) \\ A_{i-1}^\dagger + \frac{1}{\sigma_i}(e_i - A_{i-1}^\dagger a_i)(A_{i-1}^\dagger a_i)^T A_{i-1}^\dagger & \text{if} \quad a_i \in \mathcal{R}(A_{i-1}) \end{cases}, \tag{8}$$

where $\sigma_i = 1 + \|A_{i-1}^\dagger a_i\|_2^2$. We can judge if $a_i \in \mathcal{R}(A_{i-1})$ or not by observing vector $u := (I - A_{i-1}A_{i-1}^\dagger)a_i$, since

$$a_i \notin \mathcal{R}(A_{i-1}) \Leftrightarrow u = (I - A_{i-1}A_{i-1}^\dagger)a_i \ne 0, \tag{9}$$

$$a_i \in \mathcal{R}(A_{i-1}) \Leftrightarrow u = (I - A_{i-1}A_{i-1}^\dagger)a_i = 0. \tag{10}$$

This method was proposed by Greville in the 1960s[7].

1.2. Matrix Factorization. From Greville's method, a factorization for the Moore-Penrose inverse of A can be obtained. If we define vectors k_i, f_i and v_i as

$$k_i = A_{i-1}^\dagger a_i, \tag{11}$$

$$u_i = a_i - A_{i-1}k_i = (I - A_{i-1}A_{i-1}^\dagger)a_i, \tag{12}$$

$$\sigma_i = 1 + \|k_i\|_2^2, \tag{13}$$

$$f_i = \begin{cases} \|u_i\|_2^2 & \text{if} \quad a_i \notin \mathcal{R}(A_{i-1}) \\ \sigma_i & \text{if} \quad a_i \in \mathcal{R}(A_{i-1}) \end{cases}, \tag{14}$$

$$v_i = \begin{cases} u_i & \text{if} \quad a_i \notin \mathcal{R}(A_{i-1}) \\ (A_{i-1}^\dagger)^T k_i & \text{if} \quad a_i \in \mathcal{R}(A_{i-1}) \end{cases}, \tag{15}$$

we can express $A_i^\dagger$ in a unified form for general matrices as $A_i^\dagger = A_{i-1}^\dagger + \frac{1}{f_i}(e_i - k_i)v_i^T$, hence

$$A^\dagger = \sum_{i=1}^{n} \frac{1}{f_i}(e_i - k_i)v_i^T. \tag{16}$$

If we denote

$$K = [k_1, \ldots, k_n], \tag{17}$$

$$V = [v_1, \ldots, v_n], \tag{18}$$

$$F = \mathrm{Diag}\left\{f_1, \cdots, f_n\right\}, \tag{19}$$

we obtain a matrix factorization of $A^\dagger$ as follows.

THEOREM 1.1. *Let $A \in R^{m\times n}$ and $\mathrm{rank}(A) \leq \min\{m, n\}$. Using the above notations, the Moore-Penrose inverse of A has the following factorization*

$$A^\dagger = (I - K)F^{-1}V^T. \tag{20}$$

Here I is the identity matrix of order n, K is a strict upper triangular matrix, F is a diagonal matrix, whose diagonal elements are all positive.

If A is full column rank, then

$$V = A(I - K) \tag{21}$$

$$A^\dagger = (I - K)F^{-1}(I - K)^T A^T. \tag{22}$$

2. Main results. In this paper, we perform an incomplete version of Greville's method, so that we can construct an approximate Moore-Penrose inverse of A, maintaining the sparsity of the preconditioner and saving computations. We call the following algorithm the **Matrix-wise** Greville Preconditioning algorithm, since it forms or updates the whole matrix at a time rather than column by column.

ALGORITHM 1. *Matrix-wise Greville Preconditioning algorithm*

set $M_0 = 0$
for $i = 1 : n$
    $k_i = M_{i-1}a_i$
    $u_i = a_i - A_{i-1}k_i$
    if $\|u_i\| \neq 0$
        $f_i = \|u_i\|_2^2$
        $v_i = u_i$
    else
        $f_i = 1 + \|k_i\|_2^2$
        $v_i = M_{i-1}^T k_i$
    end if
    $M_i = M_{i-1} + \frac{1}{f_i}(e_i - k_i)v_i^T$
    perform numerical droppings to $M_i^\dagger$
end for
Get $M_n \approx A^\dagger$.

Remark 1. *In Algorithm 1, we do not need to store vectors k_i, v_i, f_i, $i = 1, \ldots, n$, because we form the $M_i^\dagger$ explicitly.*

If we want to construct the matrix K, F and V without forming M_i explicitly, we can use a vector-wise version of the above algorithm. In Algorithm 1, the column vectors of K are constructed one column at a step as follows,

$$
\begin{aligned}
k_i &= M_{i-1}a_i \\
&= \sum_{p=1}^{i-1}(e_p - k_p)\frac{1}{f_p}v_p^T a_i \\
&= \sum_{p=1}^{i-2}(e_p - k_p)\frac{1}{f_p}v_p^T a_i + (e_{i-1} - k_{i-1})\frac{1}{f_{i-1}}v_{i-1}^T a_i \\
&= M_{i-2}a_i + (e_{i-1} - k_{i-1})\frac{1}{f_{i-1}}v_{i-1}^T a_i.
\end{aligned}
$$

To form the last column of K, the requirement relationship can be expended as follows,

$$
\begin{array}{ccccccc}
 & & & k_n = M_{n-1}a_n & & & \\
 & & \nearrow & & \nwarrow & & \\
 & M_{n-2}a_n & & & & k_{n-1} = M_{n-2}a_{n-1} & \\
 \nearrow & & \nwarrow & & \nearrow & & \nwarrow \\
M_{n-3}a_n & & k_{n-2} = M_{n-3}a_{n-2} & & M_{n-3}a_{n-1} & & k_{n-2} = M_{n-3}a_{n-2} \\
 & & & \cdots\cdots . & & &
\end{array}
$$

Hence, to update k_i, we need to compute every $M_i^\dagger a_k,\ i = 1,\dots n-1,\ k = i+1,\dots,n$.

Based on vectors $k_i,\ i = 1,\dots,n$, vectors $v_i,\ i = 1,\dots n$ and scalars $f_i,\ i = 1,\dots,n$ can be computed easily. In the following, we rewrite Algorithm 1 into a vector-wise form.

ALGORITHM 2. *Vector-wise Greville Preconditioning Algorithm*

set $K = 0_{n\times n}$
for $i = 1 : n$
    $u = a_i - A_{i-1}k_i$
    if $\|u\| \neq 0$
        $f_i = \|u\|_2^2$
        $v_i = u$
    else
        $f_i = \|k_i\|_2^2 + 1$
        $v_i = (M_{i-1})^T k_i = \sum_{p=1}^{i-1} \frac{1}{f_p} v_p(e_p - k_p)^T k_i$
    end if
    for $j = i+1,\dots,n$
        $k_j = k_j + \frac{v_i^T a_j}{f_i}(e_i - k_i)$
        perform numerical droppings on k_j
    end for
end for
$K = [k_1,\dots,k_n]$, $F = Diag\{f_1,\dots,f_n\}$, $V = [v_1,\dots,v_n]$.

Remark 2. *When numerical droppings are performed, we have the following relationships,*

$$
\begin{aligned}
A^\dagger \approx M &= (I - K)F^{-1}V^T \\
V &= A(I - K) \quad \text{when } A \text{ is full column rank.}
\end{aligned}
$$

2.1. Greville's Method and RIF preconditioner. In this section, we especially take a look at the full column rank case. When A is full column rank, Algorithm 2 can be simplified as follows.

ALGORITHM 3. *Vector-wise Greville Preconditioning Algorithm for Full Column Rank Matrices*

1. *set* $K = 0_{n\times n}$
2. *for* $i = 1 : n$
3. $\quad u_i = a_i - A_{i-1}k_i$
4. $\quad f_i = \|u_i\|_2^2$
5. $\quad$ *for* $j = i+1, \ldots, n$
6. $\qquad k_j = k_j + \frac{u_i^T a_j}{f_i}(e_i - k_i)$
7. $\qquad$ *perform numerical droppings on* k_j
8. $\quad$ *end for*
9. *end for*
10. $K = [k_1, \ldots, k_n]$, $F = \mathrm{Diag}\{f_1, \ldots, f_n\}$.

In Algorithm 3,

$$
\begin{aligned}
u &= a_i - A_{i-1}k_i \\
&= [a_1, \ldots, a_i, 0, \ldots, 0] \begin{bmatrix} -k_{i,1} \\ \vdots \\ -k_{i,i-1} \\ 1 \\ 0 \\ \vdots \\ 0 \end{bmatrix} \\
&= A_i(e_i - k_i) \\
&= A(e_i - k_i).
\end{aligned}
$$

If we denote $e_i - k_i$ as z_i, then $u_i = Az_i$.

The Line 6 in the Algorithm 3, can also be rewritten as

$$
\begin{aligned}
k_j &= k_j + \frac{u_i^T a_j}{\|u_i\|_2^2}(e_i - k_i) \\
e_j - k_j &= e_j - k_j - \frac{u_i^T a_j}{\|u_i\|_2^2}(e_i - k_i) \\
z_j &= z_j - \frac{u_i^T a_j}{\|u_i\|_2^2} z_i.
\end{aligned}
$$

Denote $d_i = \|u_i\|_2^2$ and $\theta = \dfrac{u_i^T a_j}{d_i}$. Then combining all the new notations, we can rewrite the algorithm as follows.

Algorithm 4.

1. *set* $Z = I_{n\times n}$
2. *for* $i = 1 : n$
3. $\quad u_i = A_i z_i$
4. $\quad d_i = (u_i, u_i)$
5. $\quad$ *for* $j = i+1, \ldots, n$
6. $\qquad \theta = \frac{(u_i, a_j)}{d_i}$
7. $\qquad z_j = z_j - \theta z_i$
8. $\qquad$ *perform numerical droppings on* z_j
9. $\quad$ *end for*
10. *end for*
11. $Z = [z_1, \ldots, z_n]$, $D = Diag\{d_1, \ldots, d_n\}$.

Remark 3. *Since* $z_i = e_i - k_i$*, we have* $Z = I - K$*. In exact arithmetic, the factorization of* $A^\dagger$ *in Theorem 1.1 can be rewritten as*

$$A^\dagger = ZD^{-1}Z^TA^T. \tag{23}$$

Hence, we obtain that,

$$(A^TA)^{-1} = ZD^{-1}Z^T. \tag{24}$$

And if we define $Z^{-T} = L$*, the above equation equals*

$$A^TA = LDL^T, \tag{25}$$

which is a LDL^T *decomposition of* A^TA*.*

In Line 6, since $u_i = Az_i$*, and* $a_j = Ae_j$*, where* e_j *is the* j*th column of an identity matrix,*

$$\theta = \frac{(u_i, a_j)}{d_i} = \frac{(Az_i, Ae_j)}{(Az_i, Az_i)} = \frac{(z_i, e_j)_{A^TA}}{(z_i, z_i)_{A^TA}}. \tag{26}$$

Since when A *is full column rank,* A^TA *is SPD, which implies that* $(\cdot,\cdot)_{A^TA}$ *is a well defined inner product. If we do not perform numerical droppings in Algorithm 4, Algorithm 4 is nothing but a* **Gram-Schmidt process** *with respect to inner product* $(\cdot,\cdot)_{A^TA}$*. If* θ *in Line 6 is changed to*

$$\theta = \frac{(z_i, z_j)_{A^TA}}{(z_i, z_i)_{A^TA}}, \tag{27}$$

then Algorithm 4 is in corresponding to **Modified Gram-Schmidt process** *with respect to inner product* $(\cdot,\cdot)_{A^TA}$*.*

From the above discussion, for full column rank rectangular matrices A*, both Algorithm 4 and RIF which was proposed by Benzi and Tůma [1, section* 3*] perform a* A^TA*-orthogonalization. Hence when the same numerical droppings strategy is used, Algorithm 4 and RIF obtain the same* Z*.*

When A is rank deficient, Algorithm 4 and RIF which was proposed by Benzi and Tůma in [1] may breakdown due to a vector u_i which is very close to zero. On the other hand, the Greville's method tries to overcome the rank deficiency in A.

In [3], R. Bru, J. Martín, J. Mas and M. Tůma proposed **Balanced Incomplete Factorization**, which was based on inverse Sherman-Morrison formula. Suppose that the general nonsymmetric matrix A can be written as

$$A = A_0 + \sum_{k=1}^{n} x_k y_k^T, \tag{28}$$

where A_0 is a nonsingular matrix and $\{x_k\}_{k=1}^n$ and $\{y_k\}_{k=1}^n$ are two sets of vectors in R^n. The inverse of A when using the Sherman-Morrison formula [4] is given by

$$A_0^{-1} - A^{-1} = A_0^{-1} U_{A_0} D_{A_0}^{-1} V_{A_0}^T A_0^{-1}, \tag{29}$$

where U_{A_0} and V_{A_0} have the column vectors u_k and v_k given by,

$$u_k = x_k - \sum_{i=1}^{k} \frac{v_i^T A_0^{-1} x_k}{r_i} u_i \quad \text{and} \quad v_k = y_k - \sum_{i=1}^{k} \frac{y_k^T A_0^{-1} u_i}{r_i} v_i \tag{30}$$

respectively, and $D_{A_0} = \mathrm{diag}(r_1, \ldots, r_n)$, $r_k = 1 + y_k^T A_0^{-1} x_k$ for $k = 1, \ldots n$. Hence, if we define $A_k = A_0 + \sum_{i=1}^n x_i y_i^T$, and assume we know A_0^{-1}, we can compute A_k^{-1} by rank-one update from A_{k-1}^{-1}, and finally we can obtain A^{-1}. This method looks very similar to Greville's method, since both of them are based on rank-one update. However, if we let $A_0 = sI$, where s is a nonzero scalar, and I is an identity matrix, then let s go to zero, we can see that $A_0^{-1} \to \inf$. Notice that in Greville's method, we start from a zero matrix, hence, Greville's method is not a generalization to Sherman-Morrison formula, which implies that our preconditioning algorithm is not a generalization to BIF preconditioner.

2.2. Numerical Examples. In this subsection we present some numerical results to compare our Greville's method with the RIF preconditioner. More results will be shown in the future. All computations were run on a Dell Precision 690, where the CPU is 3 GHz and the memory is 16 GB, and the programming language and compiling environment was GNU C/C++ 3.4.3 in Redhat Linux.

We tested the matrices from University of Florida Sparse Matrix Collection. We use Greville's preconditioners and RIF preconditioners to precondition GMRES to solve a least squares problem (1), where b is A times a vector whose elements are all ones. And the stopping criterion is

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

The information of the matrix is listed below. The original matrix has some zero columns and rows, here we consider the matrix without zero columns and rows.

TABLE 1
Information on the matrix

Name	m	n	rank	density(%)
Maragal_2	536	260	171	2.24
lp_cycle	3371	1890	1875	0.3

In Algorithm 2, Line 4, we use $\|u\|_2$ to judge if the column a_i is linearly independent with previous columns or not. In practice, since we perform numerical droppings, Line 4 does not work well, hence we used the following inequality

$$\|u\|_2 < \tau_s * \|A_{i-1}\|_F * \|a_i\|_2, \tag{31}$$

where τ_s is a threshold. If the inequality (31) holds, we take a_i as a linearly dependent column. Denote the dropping tolerance as τ_d, we have the following results.

TABLE 2
Maragal_2, the number of rank deficient columns is 89

$\tau_d \setminus \tau_s$	10^{-1}	10^{-2}	10^{-3}	10^{-4}	10^{-5}	10^{-6}
10^{-1}	254 (0.02)	20 (0.02)	8 (0.02)	4 (0.02)	3 (0.03)	3 (0.02)
	×	170 (0.19)	169 (0.15)	170 (0.15)	170 (0.14)	170 (0.13)
	×	0.21	0.17	0.17	0.17	0.15
10^{-2}	254 (0.04)	39 (0.08)	4 (0.04)	0 (0.04)	0 (0.04)	0 (0.04)
	×	94 (0.10)	131 (0.12)	118 (0.1)	118 (0.1)	118 (0.1)
	×	0.18	0.16	0.14	0.14	0.14
10^{-3}	254 (0.12)	71 (0.15)	8 (0.06)	2 (0.05)	0 (0.06)	0 (0.06)
	×	56 (0.05)	71 (0.07)	83 (0.1)	106 (0.13)	106 (0.12)
	×	0.20	* 0.13	0.15	0.19	0.18
10^{-4}	254 (0.24)	95 (0.22)	22 (0.1)	4 (0.08)	2 (0.07)	1 (0.07)
	×	×	47 (0.06)	88 (0.15)	65 (0.09)	86 (0.1)
	×	×	0.16	0.23	0.16	0.17
10^{-5}	254 (0.33)	95 (0.24)	71 (0.19)	24 (0.1)	3 (0.08)	2 (0.09)
	×	×	15 (0.02)	30 (0.03)	70 (0.11)	101 (0.15)
	×	×	0.21	* 0.13	0.19	0.24
10^{-6}	254 (0.36)	95 (0.25)	77 (0.21)	59 (0.17)	16 (0.1)	3 (0.09)
	×	×	* 13 (0.02)	* 13 (0.02)	33 (0.05)	75 (0.09)
	×	×	0.22	0.19	0.16	0.18

In Table 2, in each cell, the first row is the number of columns that the method recognized as linearly dependent columns (preconditioning time), the second row is number of iterations (iteration time), and the third row is total cpu time. '×' means no convergence is achieved in 2000 steps. The best cpu time and number of iterations are indicated by $*$.

In Table 2, when no linearly dependent columns are detected, we end up with the RIF preconditioners. From the table, we can see that when too many columns (more than 89) are recognized as linearly dependent columns , GMRES does not converge, however if we can detect some linearly dependent columns, usually convergence is accelerated.

For this problem, the best cpu time and number of iterations are both achieved by Greville's method.

TABLE 3
lp_cycle, the number of rank deficient columns is 15, $\tau_s = 10^{-6}$

τ_d	10^{-1}	10^{-2}	10^{-3}	10^{-4}	10^{-5}	10^{-6}
Pre. T	12(1.11)	12(1.27)	12(1.80)	15(2.68)	17(3.59)	17(4.46)
Its. T	1668(90.74)	1264(56.38)	721(23.11)	280(6.57)	63(1.39)	38(0.90)
Tot. T	91.85	57.65	24.91	9.26	* 4.98	5.36

In Table 3, we have results for matrix lp_cycle. In "Pre. T" row, we list "number of linearly dependent columns we detected (preconditioning time)", in "Its. T", we list "number of iterations (iteration time)", and in "Tol. T", we gave out the total.

For this matrix, RIF preconditioning algorithm breaks down at the 182th column of A, which is the first linearly dependent column of A.

REFERENCES

[1] M. Benzi and M. Tůma, *A robust preconditioner with low memory requirements for large sparse least squares problems*, SIAM J. Comput., Vol. 25, pp. 499-512, 2003.
[2] P. N. Brown and H. F. Walker, *GMRES On (nearly) singular system*, SIAM J. Matrix Anal. Appl., Vol. 18, pp. 37-51, 1997.
[3] R. Bru, J. Marín , J. Mas and M. Tůma, *Balanced incomplete factorization*, SIAM J. Sci. Comput., Vol. 30, pp. 2302-2318, 2008.
[4] R. Bru, J.. Cerdán, J. Marín , J. Mas, *Preconditioning sparse nonsymmetric linear systems wit hthe Sherman-Morrison formula*, SIAM J. Sci. Comput., Vol. 25, pp. 701-715, 2003.
[5] I. S. Duff, R. G. Grimes and J. G. Lewis, *Sparse matrix test problems*, ACM Trans. Math. Software, Vol. 15, pp. 1-14, 1989.
[6] W. J. Duncan, *Some devices for the solution of large sets of simultaneous equations (with an appendix on the reciprocation of partitioned matrices)*, The London, Edinburgh and Dublin Philosophical Magazine and Journal of Science, Seventh Series, 35, pp. 660, 1944.
[7] T. N. E. Greville, *Some applications of the pseudoinverse of a matrix*, SIAM Review, Vol. 2, pp. 15-22, 1960.
[8] K. Hayami, J-F Yin, and T. Ito, *GMRES methods for least squares problem*, National Institute of Informatics Technical Report, NII-2007-09E, 2007.
[9] K. S. Riedel, *A Sherman Morrison Woodbury Identity for rank Augmenting Matrices With Application to Centering*, SIAM J. Mat. Anal., Vol. 12, No. 1, pp. 80-95, 1991.
[10] M. S. Bartlett, *An inverse matrix matrix adjustment arising in discriminant analysis*, Annals of Mathematical Statistics, Vol. 22, p107, 1951.
[11] J. A. Fill and D. E. Fishkind, *The Moore-Penrose Generalized Inverse for Sums of Matrices*, SIAM J. Matrix Anal. Appl., Vol. 21, pp. 629–635, 1999.
[12] S. L. Campbell and Carl. D. Meyer, Jr, *Generalized Inverses of Linear Transformations*, Pitamn Publishing Limited, 1979.
[13] Y. Saad, *Iterative Methods for Sparse Linear Systems (2nd edition)*, SIAM, Philadelphia, 2003.
[14] Y. Saad and M. H. Schultz, *GMRES: A generalized minimal residual method for solving nonsymmetric linear systems*, SIAM J. Sci. Statist. Comput., Vol. 7, pp. 856-869, 1986.
[15] P.-Å. Wedin, *Perturation Theory for Pseudo-inverse*, Bit, Vol. 13, pp. 217-232, 1973.