

# A two-phase preconditioning strategy of sparse approximate inverse for indefinite matrices \*

Eun-Joo Lee and Jun Zhang<sup>†</sup>

E-mail: elee3@csr.uky.edu, jzhang@cs.uky.edu

Laboratory for High Performance Scientific Computing & Computer Simulation,  
Department of Computer Science, University of Kentucky,  
773 Anderson Hall, Lexington, KY 40506-0046, USA

March 16, 2007

## Abstract

A two-phase preconditioning strategy based on a factored sparse approximate inverse is proposed for solving sparse indefinite matrices. In each phase, the strategy first makes the original matrix diagonally dominant to enhance the stability by a shifting method, and constructs an inverse approximation of the shifted matrix by utilizing a factored sparse approximate inverse preconditioner. The two approximate inverse matrices produced from each phase are then combined to be used as a preconditioner for the original matrix. Experimental results show that the presented strategy improves the accuracy and the stability of the preconditioner on solving indefinite sparse matrices. Furthermore, the strategy ensures that convergence rate of the preconditioned iterations of the two-phase preconditioning strategy is much better than that of the standard sparse approximate inverse ones for solving some indefinite matrices.

## 1 Introduction

Preconditioned Krylov subspace methods are generally considered as one of the most promising techniques [2, 4, 6, 16] for solving very large sparse linear systems of the form:

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

where  $A$  is a matrix of order  $n$ . Indeed, incomplete LU (ILU) preconditioning techniques have attracted much attention, because they have been successful in solving many symmetric and non-symmetric matrices. The techniques, however, may encounter difficulties in solving indefinite matrices for which the matrices have both positive and negative eigenvalues. In particular, there are at least two reasons which make the ILU techniques problematic. The first can be due to small or zero pivots in indefinite matrices, which may yield factorizations unstable and inaccurate [13]. In addition, small pivots are usually related to small or zero entries on the diagonal of a matrix, so an indefinite matrix with zero diagonal entries may have a higher possibility of encountering zero pivots if it is also nonsymmetric [15, 19]. Secondly, unstable triangular solutions can happen when the inverses of the triangular factors  $\|L^{-1}\|$  and  $\|U^{-1}\|$  are extremely large while the off-diagonal

---

\*Technical Report No. TR 476-07, Department of Computer Science, University of Kentucky, Lexington, KY, 2007. The authors' research work were supported in part by the U.S. National Science Foundation under grants CCR-0092532 and CCF-0527967, in part by the Kentucky Science and Engineering Foundation under grants KSEF-148-502-05-132 and KSEF-148-502-06-186, and in part by the Alzheimer's Association under a New Investigator Research Grant NIGR-06-25460.

<sup>†</sup>URL: <http://www.cs.uky.edu/~jzhang>.

elements of  $L$  and  $U$  are reasonably bounded. Such problems are also usually resulted from very small pivots [6, 9, 19].

Small pivots are often the origin of stability problems in computing ILU factorization on indefinite matrices. Therefore, several researchers have focused on methods that replace the small pivots of indefinite matrices with some large values. In [2, 14, 18], a shifting strategy that adds a value to the diagonals of an indefinite matrix is proposed to make the resulting preconditioner well conditioned. However, determining the value to be added for small pivots is usually critical to the performance of the resulting preconditioner [13]. That is, selecting a large replacing value may result in a factorization that is stable but less accurate. On the other hand, selecting a small replacing value may result in a factorization that is accurate but unstable. Such a tradeoff of the shifting strategy has been well studied in [19].

In recent years, a few preconditioning techniques in the form of sparse approximate inverse have been developed [1, 3, 5, 7, 10, 11, 12]. Such techniques have some advantages over the conventional ILU factorizations. Specifically, the process of applying the sparse approximate inverse preconditioning techniques can be performed by the matrix-vector operations, in which the operations are relatively easier to parallelize than the triangular solutions associated with the ILU factorizations. The sparse inverse preconditioning techniques may succeed in solving certain problems where the ILU factorizations are difficult to handle [7]. In addition to that, factored sparse approximate inverse (FAPINV) which a sparse approximate inverse has a factored form, tends to perform better in convergence rate for the same amount of nonzeros and also requires less computational cost than a non-factored form does. However, the resulting approximate inverse could still break down for solving indefinite matrices due to zero or small pivots [1, 20].

As part of our continuous efforts on solving indefinite matrices, we propose to adopt the idea of a shifting strategy [14] to replace small or zero elements on the diagonal of the original matrix, and to reinforce with a two-phase preconditioning process to deal with the tradeoff between stability and accuracy of the resulting preconditioner. More specifically, the first phase of the process employs the shifting strategy to the original matrix so that a shifted matrix can be well conditioned, and then an approximate inverse,  $M_1$ , of the shifted matrix is obtained by utilizing FAPINV. In the second phase of the process, a temporary matrix which is a product of  $M_1$  and the shifted matrix, is considered to acquire a better approximate inverse of the original matrix than  $M_1$ . Applying the shifting strategy again to the temporary matrix produces a second shifted matrix, and FAPINV computes a second inverse approximation,  $M_2$ , of the second shifted matrix. The resulting sparse approximate inverse,  $M$ , has the form of  $M = M_2M_1$ , where  $M_1$  and  $M_2$  are computed in each phase.

This paper is organized as follows. In Section 2, a function for determining the shifting parameter  $\alpha$ , and a two-phase preconditioning of shifted matrices for computing sparse approximate inverses are proposed. In Section 3, numerical results are presented to demonstrate advantages and preconditioning performance of the proposed preconditioner over the standard FAPINV preconditioner. Concluding remarks are in Section 4.

## 2 Stabilized FAPINV (SFAPINV) Preconditioner

We now introduce the stabilized factored approximate inverse (SFAPINV) algorithm for solving indefinite matrices. The SFAPINV preconditioner is computed in a two-phase preconditioning of two shifted matrices. Each phase generates an approximation of the inverse of a shifted matrix. These two approximations have factored forms of  $M_1 = L_1D_1U_1 \approx A_1^{-1}$  and  $M_2 = L_2D_2U_2 \approx A_2^{-1}$ , where  $L_i$  is a lower triangular matrix,  $D_i$  is a diagonal matrix, and  $U_i$  is an upper triangular matrix,  $i = 1, 2$ . Finally, the resulting preconditioner becomes the form of  $M = (L_2D_2U_2)(L_1D_1U_1)$ .

## 2.1 Determining the shifting parameter

Indefinite matrices usually have many small or zero pivots that can be the reason of breakdown in constructing sparse approximate inverse as well as ILU-type preconditioners. In order to prevent the resulting sparse approximate inverse from being unstable, we employ a shifting strategy which factors a shifted matrix  $A' = A + \alpha I$ , where  $\alpha$  is a scalar so that  $A + \alpha I$  is well conditioned (e.g., diagonally dominant). As was mentioned in the introduction, the choice of  $\alpha$  is significant for good performance for such strategies. For example, if a matrix  $A$  is ill-conditioned, the inverse of  $A + \alpha I$  could be quite different from  $A^{-1}$  [1, 2, 20]. Indeed,  $\alpha$  should be large enough to ensure the existence of the sparse approximate inverse factorization, but also small enough so that  $A + \alpha I$  is close to  $A$ .

We present an efficient algorithm `FIND-ALPHA( $A, \alpha$ )` in Function 2.1, to determine a proper value for the shifting parameter  $\alpha$ , rather than a brute-force approach which is computationally prohibitive.

**FUNCTION 2.1** `FIND-ALPHA( $A, \alpha$ )`

```

1. Do  $i = 1, n$ 
2.    $c(i) = 0$ 
3. End Do
4.  $\alpha = 0$ 
5. Do  $i = 1, n$ 
6.    $temp = 0$ 
7.   Do  $j = 1, n$ 
8.     If ( $j \neq i$ ), then  $c(i) = c(i) + |a_{ij}|$ 
9.     else  $temp = |a_{ii}|$ 
10.  End Do
11.  If ( $c(i) \leq temp$ ), then  $c(i) = temp$ 
12. End Do
13. Do  $i = 1, n$ 
14.  If ( $\alpha \leq c(i)$ ), then  $\alpha = c(i)$ 
15. End Do
16. Return  $\alpha$ 

```

Note that  $n$  refers to the order of the original matrix  $A$ ,  $a_{ij}$  denotes a nonzero element in row  $i$  and column  $j$ , and  $c(i)$  represents an accumulator of the off-diagonals in row  $i$ , where  $i, j = 1, \dots, n$ . The function starts with initializing  $c(i)$  for all rows and the shifting parameter  $\alpha$ . In lines 5–12,  $c(i)$  is determined by selecting a larger value between the summation of the absolute value of the off-diagonals in row  $i$  and the diagonal of row  $i$ . In lines 13–16, the shifting value  $\alpha$  is decided by choosing the largest value among  $c(i)$ s. At the end, the function returns the largest accumulator as the computed shifting parameter that will make all rows diagonally dominant.

## 2.2 Two-phase FAPINV preconditioner (SFAPINV)

Given a sparse approximate inverse  $M_1$  computed by using FAPINV to the original matrix  $A$ , we assume that  $M_1$  is inefficient, in the sense that many Krylov iterations are needed to solve the preconditioned linear system

$$M_1 A x = M_1 b. \quad (2.2)$$

Another sparse approximate inverse  $M_2$  for the preconditioned linear system (2.2) could be considered to acquire a closer inverse of  $A$  than  $M_1$ . A product of the two preconditioners,  $M_2 M_1$ , is utilized as a sparse approximate inverse of  $A$ , and  $M_2 M_1$  becomes more accurate to the inverse

of  $A$  than  $M_1$  does. In fact, the product matrix  $M_2M_1$  may hold more information than a single matrix  $M$  can. If  $M_2M_1$  is not successful to solve the preconditioned system, another approximate inverses may be considered, and this procedure can be continued for a few times to obtain a good preconditioner (See [17] for details).

In the case that the original matrix is indefinite, a combined preconditioner  $M_2M_1$ , computed directly from  $A$ , however, may be unstable because of small or zero pivots. Furthermore, the shifting strategy makes the original matrix diagonally dominant, but the inverse of  $A + \alpha I$  could be quite different from  $A^{-1}$  if  $A$  is ill-conditioned [1, 2, 20]. Thus, we combine the two-phase preconditioning with the shifting strategy, called STABILIZED FAPINV PRECONDITIONER (SFAPINV) in Algorithm 2.1, to improve the accuracy and stability of the sparse approximate inverse factorization. Here, FAPINV [20] is applied as a local preconditioner in each phase.

**ALGORITHM 2.1** STABILIZED FAPINV PRECONDITIONER

1. **Call Find-Alpha**( $A, \alpha_1$ )
2. Construct a shifted matrix  $A_1 = A + \alpha_1 I$
3. **Call FAPINV**( $A_1, \tau_1, M_1$ )
4. Compute a temporary matrix  $W = M_1 A$
5. Drop small entries of  $W$  with respect to  $\tau$
6. **Call Find-Alpha**( $W, \alpha_2$ )
7. Construct a shifted matrix  $A_2 = W + \alpha_2 I$
8. **Call FAPINV**( $A_2, \tau_2, M_2$ )
9. **Return**  $M_2M_1$

Note that  $\tau_1$ ,  $\tau_2$ , and  $\tau$  represent dropping tolerances. The algorithm first determines the shifting parameter  $\alpha_1$  from the FIND-ALPHA function, and constructs a shifted matrix  $A_1$  which becomes diagonally dominant. In line 3, FAPINV( $A_1, \tau_1, M_1$ ) applies the FAPINV preconditioner to the matrix  $A_1$  with the dropping tolerance  $\tau_1$ , and returns an approximate matrix  $M_1$  which has a factored form of  $M_1 = L_1D_1U_1$  of the inverse of  $A_1$ . Then with the factored approximate inverse  $M_1$ , the preconditioned system (2.2) can be written as

$$(L_1D_1U_1)Ax = (L_1D_1U_1)b. \quad (2.3)$$

The preconditioned system (2.3), however, may need many iterations to converge because the factorization  $L_1D_1U_1$  could be inaccurate if the shifting parameter  $\alpha_1$  is too large. For this potential problem in inaccuracy, in line 4, a temporary matrix  $W = M_1A$  is considered to further precondition the system (2.3). In line 5, a dropping threshold parameter,  $\tau$ , is applied to control the sparsity rate of  $W$ , where  $W$  is usually denser than  $A$  except the diagonal entries. By doing this, the diagonal entries of the matrix  $W$  are not dropped regardless of their magnitude. In lines 6-8, the shifting strategy is re-employed to obtain a more accurate and stabilized preconditioner, and the second FAPINV( $A_2, \tau_2, M_2$ ) factorization computes an approximation  $M_2$  which has a factored form  $M_2 = L_2D_2U_2$  of the inverse of  $A_2$ . In line 9, the algorithm returns a combined approximation,  $M_2M_1$ , of the inverse of  $A$ . As a result, the final preconditioned system becomes

$$(L_2D_2U_2)(L_1D_1U_1)Ax = (L_2D_2U_2)(L_1D_1U_1)b. \quad (2.4)$$

Now, we point out some important claims behind our strategy.

- It has been well studied that for a given matrix, finding a suitable shifting parameter of the existing shifting strategies may be complicated [18]. More specifically, too small diagonal shift will not have the desired effect of stabilization, but too large will result in an inaccurate preconditioner. In this regard, the FIND-ALPHA( $A, \alpha$ ) function provides an explicit and efficient way in determining the shifting parameters compared to a commonly used brute-force approach.

- It is natural to see that the resulting preconditioner  $M_2M_1$  is nonsingular. This claim can be justified by the two following facts. The first is that  $M_1$  and  $M_2$  are produced by the FAPINV preconditioner in a factored form. The second is that the two factored sparse approximate inverses are not singular [20].
- In general, the second shifting parameter  $\alpha_2$  requires to be much smaller than the first parameter  $\alpha_1$ . It can be observed that a matrix which has more zeros on its diagonal needs a larger shifting parameter to stabilize the original matrix. In fact,  $W = M_1A$  tends to be closer to  $I$ , or more diagonally dominant than  $A_1$  does. Here, we propose two possible settings in choosing the two shifting parameters  $\alpha_1$  and  $\alpha_2$ . For the case that a matrix is ill-conditioned and has a low percentage of zeros on its diagonal, the setting with  $\alpha_1 = \text{FIND-ALPHA}(A, \alpha_1)$  and  $\alpha_2 = \text{FIND-ALPHA}(W, \alpha_2)$  can be a proper choice. On the other hand, if a matrix has a large number of zeros on the diagonal of the matrix, the setting with  $\alpha_1 = \text{FIND-ALPHA}(A, \alpha_1)$  and  $\alpha_2 = 0$  is recommended.
- The resulting preconditioner  $M_2M_1 = (L_2D_2U_2)(L_1D_1U_1)$  can be not only stable but also accurate on solving some highly indefinite matrices. This claim will be supported by experiments with the two-phase preconditioning to the two shifted matrices. In short, the shifting method first enhances the stability of the factorization [14], and then the two-phase preconditioning improves the accuracy of the preconditioner.

### 3 Numerical Experiments

We present numerical experiments of the SFAPINV (for stabilized factored approximate inverse) preconditioner on solving indefinite and nonsymmetric matrices. The description of the test matrices is given in Table 1. The test matrices<sup>1</sup> were solved as they were, that is, no scalings or permutations were applied. The SFAPINV preconditioner was used as a right preconditioner for experiments, but it can be used as a left preconditioner as well since there was not much difference in preconditioning effect. The preconditioned iterative solver employed was GMRES(50). For all linear systems, the right-hand side was generated by assuming that the solution is a vector of all ones. The initial guess was a zero vector. The iteration was terminated when the  $l_2$ -norm of the initial residual was reduced by at least eight orders of magnitude, or when the number of iterations reached 500. The programs of our approach were coded in standard Fortran 77 programming language in double precision with 64-bit arithmetic. The computations were carried out on a Sun-Blade-100 workstation with a 500 MHz UltraSPARC III CPU and 1 GB of RAM.

In all tables with numerical results, “iter,” “comp,” “solu,” “cond,” “ $\alpha_1$ ,” and “ $\alpha_2$ ” denote the number of GMRES iterations, the CPU time in seconds for computing the preconditioner, the CPU time for the solution phase (both GMRES and preconditioner), the  $\text{condest}^2$ , the first and second shifting parameters, respectively. “ $\tau_1$ ,” “ $\tau_2$ ,” and “ $\tau = \tau_1 * \tau_2$ ” are the dropping tolerances. The value “-1” and “-3” indicate the failure of convergence within the maximum number of the allowed iterations (500) and with the GMRES solver breakdown, respectively.

#### 3.1 Shifting parameters and dropping tolerances

We present results arisen from different settings of the shifting parameters and the dropping tolerances. In order to obtain a concrete convergence rate, two different settings of the shifting parameters are used in constructing the preconditioner. Note that the first and the second setting are de-

<sup>1</sup>All of these matrices are available online from the Matrix Market of the National Institute of Standards and Technology at <http://math.mist.gov/matrixMarket>.

<sup>2</sup>A statistic,  $\text{condest}$ , is introduced by Chow and Saad [6] to measure the stability of triangular solutions.

Matrix	Description	$n$	$nnz$	$nnzdiag$	$condition$
FIDAP007	Matrices generated by the FIDAP Package	1633	46570	1633	1.93E+11
FIDAP014	Matrices generated by the FIDAP Package	3251	65747	2351	2.62E+16
FIDAP033	Matrices generated by the FIDAP Package	1733	20315	1733	1.20E+13
GRE_512	Simulation of computer systems	512	1976	296	3.8E+02
IMPCOL_B	Chemical engineering plant models Cavett's process	59	271	17	2.7E+05
NNC1374	Nuclear reactor models	1374	8588	870	1.0E+02
NNC261	Nuclear reactor models	261	1500	150	1.2E+15
NNC666	Nuclear reactor models	666	4044	410	1.8E+11
PSMIGR_2	Inter-county migration US Inter-county migration 1965-1970	3140	540022	0	1.0E+02
RBS480_A	Forward Kinematics for the Stewart platform of Robotics	480	17088	42	1.3E+05
RBS480_B	Forward Kinematics for the Stewart platform of Robotics	480	17088	43	1.6E+05
RW136	Markov Chain Transition Matrix	136	479	0	1.4E+05
WEST0067	Chemical engineering plant models	67	294	2	3.0E+02

Table 1: Description of the test matrices ;  $n$ ,  $nnz$ ,  $nnzdiag$ , and  $condition$  denotes the order, the number of nonzero entries, the number of nonzero entries on the main diagonal, and the condition number of a matrix, respectively.

Matrix	$n$	$nnzdiag$	$condition$	$DD - row$	$\alpha_1$	$\alpha_2$
FIDAP007	1633	1633	1.93E+11	0.0245	Yes	Yes
FIDAP014	3251	2351	2.62E+16	0.2125	Yes	Yes
FIDAP033	1733	1733	1.20E+13	0.0848	Yes	Yes
GRE_512	512	296	3.8E+02	0.1328	Yes	No
IMPCOL_B	59	17	2.7E+05	0.1694	Yes	No
NNC1374	1374	870	1.0E+02	0.	Yes	No
NNC261	261	150	1.2E+15	0.	Yes	No
NNC666	666	410	1.8E+11	0.	Yes	No
PSMIGR_2	3140	0	1.0E+02	0.	Yes	No
RBS480_A	480	42	1.3E+05	0.	Yes	No
RBS480_B	480	43	1.6E+05	0.	Yes	No
RW136	136	0	1.4E+05	0.	Yes	No

Table 2: The relationship between the matrix properties and the shifting factors.

noted as  $\alpha_1 = \text{FIND-ALPHA}(A, \alpha_1)$  and  $\alpha_2 = \text{FIND-ALPHA}(W, \alpha_2)$ , and  $\alpha_1 = \text{FIND-ALPHA}(A, \alpha_1)$  and  $\alpha_2 = 0$ , respectively.

We recommend that  $\alpha_1$  and  $\alpha_2$  be chosen to improve the stability of the preconditioner, but for some matrices,  $\alpha_2$  be set to zero to enhance the accuracy. Table 2 informs a guideline on choosing the proper setting of the shifting parameters for each matrix to achieve good convergence. Determining the shifting parameters is related with the statistics of each matrix, such as the condition number, diagonally dominant row rate (DD-row), and the number of nonzeros on diagonal. Specifically, when DD-row of a matrix is zero, the second setting to the parameters ( $\alpha_2 = 0$ ) could be chosen due to the need for increased accuracy in the second phase of the preconditioning. As we can see in Table 2, for the matrices which have small condition numbers and a large number of zeros on their diagonals, we select the second setting to the parameters ( $\alpha_2 = 0$ ). On the contrary, when DD-row of a matrix is not zero, the two parameters with the first setting ( $\alpha_2 \neq 0$ ) are usually selected to improve stability in the second phase of the preconditioning. In this case, the matrices

usually have large condition numbers. For example, The FIDAP matrices, FIDAP007, FIDAP014, and FIDAP033, with the first setting have large condition numbers in the range of  $[1.93\text{E}+11, 2.62\text{E}+16]$  and low DD-rate of  $[0.0245, 0.2125]$ .

Shifting factors		Dropping tolerances			Convergence results			
$\alpha_1$	$\alpha_2$	$\tau_1$	$\tau_2$	$\tau$	comp	solu	iter	cond
1.14E+07	5.00E-01	1.0E-01	1.0E-02	1.0E-03	2.30E+00	2.28E+00	167	1.76E-07
1.14E+07	5.00E-01	1.0E-01	1.0E-03	1.0E-04	2.44E+00	2.53E+00	167	1.76E-07
1.14E+07	5.00E-01	1.0E-01	1.0E-04	1.0E-05	2.50E+00	2.81E+00	167	1.76E-07
1.14E+07	5.00E-01	1.0E-02	1.0E-01	1.0E-03	2.32E+00	2.32E+00	180	1.76E-07
1.14E+07	4.95E-01	1.0E-02	1.0E-02	1.0E-04	3.20E+00	2.34E+00	163	1.78E-07
1.14E+07	4.95E-01	1.0E-02	1.0E-03	1.0E-05	3.40E+00	2.55E+00	163	1.78E-07
1.14E+07	4.95E-01	1.0E-03	1.0E-01	1.0E-04	6.10E+00	2.47E+00	169	1.78E-07
1.14E+07	4.95E-01	1.0E-03	1.0E-02	1.0E-05	6.10E+00	2.50E+00	162	1.78E-07
1.14E+07	4.95E-01	1.0E-04	1.0E-01	1.0E-05	1.07E+01	2.72E+00	169	1.78E-07
1.14E+07	0	1.0E-03	1.0E-01	1.0E-04	6.17E+00	7.31E+00	460	8.80E+00
1.14E+07	0	1.0E-01	1.0E-03	1.0E-04	2.75E+00	9.09E+00	436	8.80E+00

Table 3: Test results for solving the FIDAP014 matrix with different values of the shifting factors and the dropping parameters.

Shifting factors		Dropping tolerances			Convergence results			
$\alpha_1$	$\alpha_2$	$\tau_1$	$\tau_2$	$\tau$	comp	solu	iter	cond
1.21E+11	6.07E-01	1.0E-01	1.0E-02	1.0E-03	6.69E-01	1.55E+00	289	1.37E-11
1.21E+11	6.09E-01	1.0E-01	1.0E-03	1.0E-04	7.00E-01	1.87E+00	300	1.37E-11
1.21E+11	6.10E-01	1.0E-01	1.0E-04	1.0E-05	7.09E-01	1.96E+00	300	1.37E-11
1.21E+11	0.75E+00	1.0E-02	1.0E-01	1.0E-03	6.59E-01	1.67E+00	320	1.09E-11
1.21E+11	5.93E-01	1.0E-02	1.0E-03	1.0E-05	6.69E-01	1.60E+00	296	1.49E-11
1.21E+11	5.94E-01	1.0E-02	1.0E-03	1.0E-05	7.20E+00	1.89E+00	300	1.49E-11
1.21E+11	5.91E-01	1.0E-03	1.0E-01	1.0E-04	1.57E+00	1.78E+00	288	1.39E-11
1.21E+11	5.92E-01	1.0E-03	1.0E-02	1.0E-05	1.56E+00	1.89E+00	297	1.50E-11
1.21E+11	5.92E-01	1.0E-04	1.0E-01	1.0E-05	1.65E+00	1.79E+00	288	1.39E-11

Table 4: Test results of SFAPINV for solving the FIDAP033 matrix with different values of the shifting factors and the dropping parameters.

Shifting factors		Dropping tolerances			Convergence results			
$\alpha_1$	$\alpha_2$	$\tau_1$	$\tau_2$	$\tau$	comp	solu	iter	cond
1.00E+00	0	1.0E-03	1.0E-02	1.0E-05	1.17E+01	3.69E-01	12	2.44E+00
1.00E+00	0	1.0E-02	1.0E-02	1.0E-04	8.17E+00	3.29E-01	13	1.79E+00
1.00E+00	0	1.95E-03	1.0E-02	5.35E-03	1.05E+01	4.40E-01	15	4.19E+00
1.00E+00	0	1.0E-02	1.0E-01	1.0E-03	9.69E+00	7.09E+00	289	1.03E+02
1.00E+00	0	1.0E-02	1.0E-03	1.0E-05	8.08E+00	3.00E-01	12	1.52E+00
1.00E+00	0	1.0E-04	1.0E-01	1.0E-05	1.31E+01	1.07E+01	353	3.09E+02

Table 5: Test results of SFAPINV for solving the GRE\_512 matrix with different values of the shifting factors and the dropping parameters.

The choice of the first setting is usually acceptable for the FIDAP matrices. Tables 3 and 4 show that the FIDAP matrices with the first setting increases the accuracy of the preconditioning, and as a result of that the number of GMRES iterations is decreased noticeably. The FIDAP014

matrix converges in around 167 iterations with the first setting, but with the second, it converges in 436 (460) iterations. In addition, the FIDAP033 matrix cannot converge in 500 iterations with the second setting while it converges with the first setting.

In the second setting, the number of GMRES iterations may be related with the second dropping tolerance. For example, Table 5 indicates that the GRE\_512 matrix converges only with the second setting ( $\alpha_2 = 0$ ). The number of iterations are varying from 12 to 353 when the dropping tolerances are ranging of [1.0E-05, 1.0E-01]. More specifically, when  $\tau_2 = 0.1$ , the number of iterations are 289 and 353 while when  $\tau_2 \neq 0.1$ , the iterations are changed from 12 to 15. We found that for the matrix GRE\_512, the number of iterations becomes large when the second dropping tolerance sets 0.1.

### 3.2 Comparison between SFAPINV and FAPINV

The comparisons of the SFAPINV (for stabilized factored approximate inverse) preconditioner with the FAPINV (for factored approximate inverse) [20] preconditioner are presented in Table 6. Under ‘‘N/A’’, we report that the preconditioner was not defined, due to zeros on the diagonal (zero pivot). In each testing, the dropping tolerances were carefully chosen to keep the memory cost (sparsity ratio) of these two preconditioners comparable. Table 7 is a list of the parameters used in both the SFAPINV and FAPINV preconditioners for the test matrices.

Matrix	SFAPINV				FAPINV			
	comp	solu	iter	cond	comp	solu	iter	cond
FIDAP007	2.56E+00	1.56E+00	153	8.96E-10	1.79E+01	6.78E+01	-1	3.71E+03
FIDAP014	6.10E+00	2.50E+00	162	1.78E-07	9.64E+01	3.30E+02	-1	7.17E+09
FIDAP033	1.57E+00	1.78E+00	288	1.39E-11	8.09E+00	4.59E+01	-1	6.83E+01
GRE_512	1.17E+01	3.69E-01	12	2.44E+00	N/A	N/A	-3	N/A
IMPCOL_B	9.99E-03	9.99E-03	27	7.57E+03	N/A	N/A	-3	N/A
PSMIGR_2	1.92E+03	1.69E+01	19	1.49E+03	N/A	N/A	-3	N/A
NNC1372	3.61E+00	2.63E+00	46	9.57E+04	3.71E+00	3.02E+01	-1	6.59E+35
NNC261	7.00E-02	8.99E-02	44	5.99E+04	5.00E-02	1.05E+00	-1	2.93E+22
NNC666	5.60E-01	2.99E-01	20	1.06.E+05	4.39E-01	6.62E+00	-1	1.23E+12
RBS480_A	5.41E+00	4.20E-01	22	2.93E+01	N/A	N/A	-3	N/A
RBS480_B	5.40E+00	3.19E-01	16	1.34E+00	4.26E+00	8.09E+00	-1	1.22E+03
RW136	1.60E-02	7.99E-03	20	2.24E+10	3.99E-03	1.24E-01	-1	2.69E+15
WEST0067	2.99E-02	0.00E+00	5	1.14E+01	N/A	N/A	-3	N/A

Table 6: Comparisons of SFAPINV and FAPINV.

As shown in Table 6, in most cases, the SFAPINV preconditioner performed better than the FAPINV preconditioner. The test matrices that were not solved by the FAPINV preconditioner may be difficult to solve by ILU preconditioners, such as ILU(0), ILUT [15]. The data in the table also demonstrates that the construction cost of the preconditioners is quite inexpensive. For the case of the FIDAP014 matrices, the construction cost of the SFAPINV preconditioner was 16 times cheaper than that of the FAPINV preconditioner. In each phase, a sparse matrix (approximation) could be computed with low memory cost. As a result, the total computational cost of these sparse matrices becomes cheaper compared with computing a single sparse matrix.

We note that the SFAPINV preconditioner does not converge if the first shifting parameter  $\alpha_1$  is set zero (result is not shown in this paper). Thus, a nonzero value for the first shifting parameter is necessary for the two-phase preconditioning of the shifting method to achieve good performance.

Matrix	SFAPINV					FAPINV
	Shifting factors		Dropping tolerances			Dropping tolerance
	$\alpha_1$	$\alpha_2$	$\tau_1$	$\tau_2$	$\tau$	
FIDAP007	2.00E+09	5.64E-01	1.0E-03	1.0E-02	1.0E-05	1.0E-03
FIDAP014	1.14E+07	4.95E-01	1.0E-03	1.0E-02	1.0E-05	1.0E-03
FIDAP033	1.21E+11	5.91E-01	1.0E-03	1.0E-01	1.0E-04	1.0E-03
GRE_512	1.00E+00	0	1.0E-03	1.0E-02	1.0E-05	1.0E-03
IMPCOL_B	1.27E+01	0	1.0E-04	1.0E-01	1.0E-05	1.0E-04
NNC1374	3.56E+03	0	1.0E-01	1.0E-04	1.0E-05	1.0E-01
NNC261	3.56E+03	0	1.0E-01	1.0E-04	1.0E-05	1.0E-01
NNC666	3.56E+03	0	1.0E-01	1.0E-04	1.0E-05	1.0E-01
PSMIGR_2	1.00E+00	0	3.18E-04	1.0E-02	2.79E-05	3.18E-04
RBS480_A	5.79E+03	0	1.0E-03	1.0E-02	1.0E-05	1.0E-03
RBS480_B	6.16E+03	0	1.0E-03	1.0E-02	1.0E-05	1.0E-03
RW136	1.57E+00	0	1.0E-03	1.0E-02	1.0E-05	1.0E-03
WEST0067	6.14E+00	0	1.0E-03	1.0E-02	1.0E-05	1.0E-03

Table 7: Parameters used in SFAPINV and FAPINV.

## 4 Concluding Remarks

We proposed an algorithm called SFAPINV (for stabilized factored approximate inverse) for solving highly indefinite matrices. SFAPINV consists of two preconditioning phases to two shifted matrices. Each phase is a factorization of a shifted matrix of the form  $A + \alpha I$ . The shifting method is employed to prevent unstable factorization due to the small or zero pivots of the original matrix, and the two-phase preconditioning is utilized to improve the accuracy of the preconditioner. FAPINV [20] has been utilized as a local preconditioner in the SFAPINV preconditioning because of its low construction cost and robustness.

Numerical experiments demonstrate the stability and accuracy of the SFAPINV preconditioner. Moreover, for the NNC1372 and FIDAP matrices, the construction cost of the preconditioner is cheaper than that of a single sparse matrix in the FAPINV preconditioner although the SFAPINV preconditioner is composed of two sparse matrices,  $M_1$  and  $M_2$ . In fact, the memory cost of each sparse matrix is small and each of the sparse approximate inverse matrices could be computed inexpensively. According to [17], computing a series of two matrices where each matrix needs small memory cost may be cheaper than that of a matrix that needs high memory cost. Thus, the SFAPINV preconditioner could be robust and fast on solving indefinite matrices.

We remark that the concept of the two-phase preconditioning of shifted matrices may be applied to other preconditioning techniques. For example, one can construct a hybrid two-phase preconditioning by using a different preconditioner in each phase. Also, the presented algorithm can be extended to a parallel version because FAPINV naturally possesses of great inherent parallelism.

## References

- [1] M. Benzi and M. Tuma, *A sparse approximate inverse preconditioner for nonsymmetric linear systems*, SIAM J. Sci. Comput., **19-3** (1998), 968–994 .
- [2] M. Benzi and D. Bertaccini, *Approximate inverse preconditioning for shifted linear systems*, BIT, **43** (2003), 231–244.

- [3] T.F. Chan, W.P. Tang, and W.L. Wan, *Wavelet sparse approximate inverse preconditioners*, BIT, **37-3** (1997), 644–660.
- [4] T.F. Chan, and H.A. van der Vorst, *Approximate and incomplete factorizations*, Parallel Numerical Algorithms, ICASE/LaRC Interdisciplinary Series in Science and Engineering **4** (1997), 91–118. (<http://www.math.uu.nl/people/vorst/publ.html#publ94>)
- [5] E. Chow and Y. Saad, *Approximate inverse preconditioners for general sparse matrices*, Technical Report UMSI **94/101** (1994), Minnesota Supercomputer Institute, University of Minnesota, Minneapolis, MN.
- [6] E. Chow and Y. Saad, *Experimental study of ILU preconditioners for indefinite matrices*, J. Comput. Appl. Math., **86** (1997), 387–414.
- [7] E. Chow and Y. Saad, *Approximate inverse preconditioners via sparse-sparse iterations*, SIAM J. Sci. Comput., **19** (1998), 995–1023.
- [8] A.C.N. van Duin, *Scalable parallel preconditioning with the sparse approximate inverse of triangular matrices*, SIAM J. Matrix Anal. Appl. **20** (1999), 987–1006.
- [9] H.C. Elman, *A stability analysis of incomplete LU factorization*, Math. Comput., **47** (1986), 191–217.
- [10] N.I.M. Gould and J.A. Scott, *Sparse approximate-inverse preconditioners using norm-minimization techniques*, SIAM J. Sci. Comput., **19-2** (1998), 605–625.
- [11] G.A. Gravvanis, *An approximate inverse matrix technique for arrowhead matrices*, Int. J. Comput. Math., **70** (1998), 35–45.
- [12] M.J. Grote and T. Huckle, *Parallel preconditioning with sparse approximate inverses*, SIAM J. Sci. Comput., **18** (1997), 838–853.
- [13] D.S. Kershaw, *On the problem of unstable pivots in the incomplete LU-conjugate gradient method*, J. Comput. Phys. **38** (1980), 114–123.
- [14] T.A. Manteuffel, *An incomplete factorization techniques for positive definite linear systems*, Math. Comput., **34** (1980), 473–497.
- [15] Y. Saad, *ILUT: a dual threshold incomplete LU factorization*, Numer. Linear Algebra Appl. **14** (1994), 387–402.
- [16] Y. Saad, *Iterative Methods for Sparse Linear Systems*, PWS, New York, 1996.
- [17] K. Wang and J. Zhang, *MSP: A class of parallel multistep successive sparse approximate inverse preconditioning strategies*, SIAM J. Sci. Comput., **24-4** (2003), 1141–1156.
- [18] L. Wang and J. Zhang, *A two step combined stable preconditioning strategy for incomplete LU factorization of CFD matrices*, Appl. Math. Comput., **144-1** (2003), 75–87.
- [19] J. Zhang, *A multilevel dual reordering strategy for robust incomplete LU factorization of indefinite matrices*, SIAM J. Matrix Anal. Appl., **22-3** (2001), 925–947.
- [20] J. Zhang, *A sparse approximate inverse technique for parallel preconditioning of general sparse matrices*, Appl. Math. Comput., **130-1** (2002), 63–85.