INTERNATIONAL JOURNAL OF
NUMERICAL ANALYSIS AND MODELING, *SERIES B*
Volume 5, Number 1-2, Pages 21–30

A SPARSE APPROXIMATE INVERSE PRECONDITIONER FOR NONSYMMETRIC LINEAR SYSTEMS

YINGZHE FAN AND ZHANGXIN CHEN

Abstract. Motivated by the paper [16], where the authors proposed a method to solve a symmetric positive definite (SPD) system $Ax = b$ via a sparse-sparse iterative-based projection method, we extend this method to nonsymmetric linear systems and propose a modified method to construct a sparse approximate inverse preconditioner by using the Frobenius norm minimization technique in this paper. Numerical experiments indicate that this new preconditioner appears more robust and takes less time of constructing than the popular parallel sparse approximate inverse preconditioner (PSM) proposed in [6]

Key words. nonsymmetric linear systems, preconditioning, projection method, Krylov subspace methods, PSM, sparse approximate inverse

1. Introduction

Consider the solution of a sparse nonsymmetric linear system of algebraic equations:

$$Ax = b, \tag{1}$$

where $A \in R^{n\times n}$ is a nonsingular matrix, $x \in R^n$ is an unknown vector, and $b \in R^n$ is a given vector. This system arises in many areas of scientific computing, such as in fluid mechanics [9], solid mechanics [4], and fluid flow in porous media [5]. When A is large and sparse, direct solvers such as Gauss elimination may bring 'fill-in' phenomenon and require huge amount of work and memory storage. Another approach to solve this system uses Krylov subspace iterative methods such as the generalized minimal residual method (GMRES) and the biconjugate gradient stabilized method (BiCGSTAB) [15, 18]. These methods require less storage but the rate of convergence depends strongly on the spectral distribution of matrix A. Usually, the more clustered the eigenvalues of A are, the faster these methods converge. Toward this end, we may apply a preconditioning technique; i.e., we may transform system (1) into the following system:

$$AMy = b, \quad x = My \quad \text{or} \quad MAx = Mb, \tag{2}$$

where M is a nonsingular matrix and is required to be cheaply constructed, called a preconditioner. If $M \approx A^{-1}$, the coefficient matrix AM of system (2) always has a 'good' spectral distribution, and then using the Krylov subspace iterative methods for solving (2), we can achieve much faster convergence.

Recently, many preconditioning techniques have been developed; see, e.g., [2, 15] for a review. In this paper, we focus on a sparse approximate inverse technique based on minimizing the Frobenius norm [6, 7, 11, 12, 13, 14]. Because of the inherent parallel feature of this technique, it has attracted much attention. Its basic idea is to construct a sparse nonsingular matrix by the constrained minimization

Received by the editors January 1, 2014 and, in revised form, March 10, 2014.
2000 *Mathematics Subject Classification.* 65F10, 65F15.

problem:

$$\min_{M\in\wp} \|AM - I\|_F, \tag{3}$$

where $\wp$ is a set of sparsity pattern of matrices, $\|.\|_F$ denotes the Frobenius norm of a matrix, and I denotes the identity matrix. The minimization problem (3) can be decoupled into n independent linear least squares problems:

$$\min_{M\in\wp} \|AM - I\|_F = \min_{M\in\wp} \sum_{j=1}^{n} \|Am_j - e_j\|_2, \tag{4}$$

where m_j and e_j denote the jth column of M and I, respectively. Thus we can construct the preconditioner M by solving n independent linear least squares problems. However, how to choose a 'good' sparsity pattern of M that can be effectively constructed is still challenging. The aim of this paper is to construct a desired preconditioner M for the sparse nonsymmetric system (2), and we will discuss the construction process in the following sections in detail. Numerical experiments indicate that this new preconditioner appears more robust and takes less time of constructing than the popular parallel sparse approximate inverse preconditioner proposed in [6].

The paper is organized as follows. In Section 2, we briefly describe three basic algorithms for computing approximate solutions of nonsymmetric systems. Then, in Section 3, we develop a new method to construct the preconditioner we are proposing. Finally, numerical experiments to check this preconditioner's effectiveness are presented in Section 4.

2. Approximate Solutions of Nonsymmetric Systems

In this section, we extend the method which was used in [16] for solving SPD linear systems to general nonsymmetric systems and then modify this method so that it can be more flexible and effective.

Let $A \in R^{n\times n}$ be a general nonsingular matrix. Also, let K and L be two m-dimensional subspaces of R^n, and $x_0 \in R^n$ be an initial guess of the solution of system (1). A projection method is a process which finds an approximate solution $x \in R^n$ of (1) as follows:

$$\text{Find } x \in x_0 + K \text{ such that } b - Ax \perp L. \tag{5}$$

Now, let $K = \text{span}\{e_{i_1}, e_{i_2}, ..., e_{i_m}\}$ and $L = AK$, where e_{i_j} is the i_jth column of the identity matrix and m is a small integer. Then problem (5) can be transformed into the following form:

$$\begin{aligned} &\text{Find } x \in x_0 + Ey \text{ such that } r_0 - AEy \perp L, \\ &\text{i.e., } (E^TA^TAE)y = E^TA^Tr_0, \end{aligned} \tag{6}$$

where $E = [e_{i_1}, e_{i_2}, ..., e_{i_m}]$, $r_0 = b - Ax_0$, and $y \in R^m$.

If we loop (6) and use it to solve the linear systems:

$$Am_j = e_j, \tag{7}$$

then we define a new approach for solving the systems in (7) as follows:

Algorithm 1 (sparse approximate solution to the system $Am_j = e_j$):

1. Choose an initial guess m_j and compute $r = e_j - Am_j$;
2. For $i = 1 : n_p$,

3. select $J = \{i_1, i_2, ..., i_m\} \in \{1, 2, ..., n\}$ and set $E = [e_{i_1}, e_{i_2}, ..., e_{i_m}]$,
4. solve

$$(E^T A^T AE)y = E^T A^T r \text{ for } y, \tag{8}$$

5. compute $m_j = m_j + Ey$,
6. compute $r = r - AEy$;
7. Enddo

Remarks: a) In step 4 of the above algorithm, the coefficient matrix $E^T A^T AE \in R^{m\times m}$ is of very small scale since m is very small. Thus solving system (8) is not very expensive. b) Note that solving system (8) is equivalent to the solution of the least squares problem:

$$\min_{y\in R^m} \|r - AEy\|_2. \tag{9}$$

Therefore, in practice, in order to obtain high accuracy, we can solve (9) instead of (8) with the QR factorization method. c) Furthermore, it follows from [15] that solving (9) is equivalent to the solution of the minimization problem: $\min_{m_j} \|e_j - Am_j\|_2$, where $m_j = \text{span}\{e_{i_1}, e_{i_2}, ..., e_{i_m}\}$.

The following theorem on the projection method indicates that this algorithm is convergent:

Theorem 2.1. *Assume that $A \in R^{n\times n}$ is a real nonsingular matrix, and let $r_{k+1} = r_k - AEy_k$ after the kth loop of* **Algorithm 1**. *Then we have the following result:*

$$\|r_{k+1}\|_2^2 = \|r_k\|_2^2 - \|AEy_k\|_2^2. \tag{10}$$

Proof. We start with the relation $(E^T A^T AE)y_k = E^T A^T r_k$ in step 4 of **Algorithm 1**, which is equivalent to $E^T A^T(r_k - AEy_k) = 0$ so $(r_k - AEy_k, AEy_k) = 0$; i.e., $(r_k, AEy_k) = (AEy_k, AEy_k)$. From step 6 of the same algorithm, we see that

$$\begin{aligned}\|r_{k+1}\|_2^2 &= (r_{k+1}, r_{k+1}) = (r_k - AEy_k, r_k - AEy_k)\\ &= (r_k, r_k) - 2(r_k, AEy_k) + (AEy_k, AEy_k)\\ &= (r_k, r_k) - (AEy_k, AEy_k)\\ &= \|r_k\|_2^2 - \|AEy_k\|_2^2,\end{aligned}$$

which is the desired relation (10). □

Note that from relation (9), set $L = \{j \mid A(i, j) \neq 0 \text{ and } r(i) \neq 0\}$. Then the potential indices of J are contained in L. If A has no zero element in the diagonal position, in order to reduce the value of the residual r as greatly as possible, we can select the indices in J to be the indices of the m components with the largest absolute values in the current residual r. Then a new modified algorithm can be defined as follows:

Algorithm 2 (sparse approximate solution to the system $Am_j = e_j$):

1. Choose an initial guess $m_j = 0$ and set $r = e_j$;
2. For $i = 1 : n_p$,
3. select the indices in J to be the indices of the m components with the largest absolute
   values in the current residual r; i.e., $J = \{i_1, i_2, ..., i_m\} \in \{1, 2, ..., n\}$
   and set $E = [e_{i_1}, e_{i_2}, ..., e_{i_m}]$,
4. solve $(E^T A^T AE)y = E^T A^T r$ for y,
5. compute $m_j = m_j + Ey$,
6. compute $r = r - AEy$;
7. Enddo

Remarks: a) The stop criterion in step 2 of the above algorithm can also be set as $\|r\|_2 > eps$ and $nnz(m_j) < lfil$, where eps is usually a very small number, nnz indicates the number of nonzero entries in its argument matrix, and $lfil$ is a prescribed number of nonzero entries. Since m is usually chosen to be very small, such as m=2, 3 or 4, the sparsity of m_j is preserved only by m. b) The drawback is that in each loop, we need to search for the m indices of the components with the largest absolute values in the current residual r, which may be considerably expensive.

To overcome this disadvantage, the natural idea is to select the index vector $J = \{i_1, i_2, ..., i_m\}$, where $r(i_j) \neq 0$. Unfortunately, the number of nonzero entries in the residual vector r increases dramatically along with the loop. However, it is interesting to note that relation (10) shows that $\|r_{k+1}\|_2^2 \leq \|r_k\|_2^2 \leq \|r_0\|_2^2 = 1$. Thus the components of r in absolute value are all less than one. Therefore, after computing the residual vector r, we can drop the components whose absolute value is less than a threshold η ($0 \leq \eta < 1$). Now, another modified algorithm can be defined as follows:

Algorithm 3 (sparse approximate solution to the system $Am_j = e_j$):
1. Choose an initial guess $m_j = 0$ and set $r = e_j$;
2. For $i = 1 : n_p$,
3. select the index vector $J = \{i_1, i_2, ..., i_m\} \in \{1, 2, ..., n\}$, where $|r(i_j)| \geq \eta$, and set $E = [e_{i_1}, e_{i_2}, ..., e_{i_m}]$,
4. solve $(E^T A^T AE)y = E^T A^T r$ for y,
5. compute $m_j = m_j + Ey$,
6. compute $r = r - AEy$;
7. Enddo

Remarks: a) In practice, we can select $\eta = 10^{-1}$ or $\eta = 10^{-2}$ and $n_p = 1$, 2 or 3 often leads to good numerical results. b) Furthermore, if m_j is not sparse, we can also apply numerical dropping to m_j with threshold ϵ, but how to select ϵ is a problem since we may not be sure about the distribution of the values in m_j.

3. Construction of Preconditioners

In this section, we shall first review some strategies for determining the sparsity pattern of a preconditioner. In general, there are two ways to construct a preconditioner. The first one is the adaptive approach [8, 11, 12] and the other is the static approach [6, 13].

The SPAI (sparse parallel approximate inverse) method which was proposed by Grote and Huckle in [12] is the most successful approach based on the adaptive method. The main idea is to augment the sparsity pattern successively with a given initial simple pattern of M until a criterion of $\|Am_j - e_j\|_2 < \varepsilon$ is satisfied for a given tolerance ε or a maximum number of nonzeros in m_j has been reached. Unfortunately, the setup time of this approach is often high even if implemented in parallel; data must be transferred from one processor to other processors on a distributed memory computer when the augment process is being executed.

On the other hand, the static approach seems more attractive in terms of implementation because the sparsity pattern of M is prescribed (a-priori) before the computation starts; during the process of calculation, communication is easy. A particularly useful and effective strategy is to use the sparsity pattern of A^k (k is a positive integer) which is based on the Hamilton-Cayley Theorem that A^{-1} can be represented as a matrix polynomial of A. However, the number of nonzero

entries of A^k will become too big to compute in practice. This motivates the sparsified method proposed by Tang [17] and Chow [6]. Here 'sparsified' means that the entries below a prescribed threshold will be removed. This method takes the sparsified pattern of A as the sparsity pattern of the preconditioner M. Then, to achieve higher accuracy, the sparsity pattern of $A^2, A^3, ...$, may be used.

However, the sparsified threshold and the power of A are not easy to determine a-priori. If k is small, the method may not converge; while k is too big, it requires more time to set up. Table 3.1 shows test data using Parasails with different levels of sparsity patterns from papers [7, 20].

Table 1: Test results for the SPD matrix with the order = 12,205; time is in seconds.

Sparsity pattern	Density	Iteration	Setup time	Solution time
A	0.25	754	2.0	39.3
A^2	0.47	539	40.0	33.7
A^3	0.80	243	491.2	20.4

We can see that using a high level sparsity pattern such as A^2 and A^3 can lead to a better preconditioner as it reduces the number of iterations and the time of solution, but the setup time for constructing the preconditioner increases dramatically. The reduction in the solution time does not compensate for the huge increase in the setup time. Hence, as it was pointed out in [7] and [20], we are in a situation where the higher accurate sparsity pattern will be used in later computations and the initial high cost of extracting a high accurate sparsity pattern may be avoided.

In order to overcome the above drawbacks, we can combine the new method presented in Section 2 with the static method. Suppose that an approximate inverse preconditioner M with the sparsified pattern of A has been computed; if we find that M is not effective, we can use the new method in Section 2 to correct (modify) it until it is sufficiently satisfactory, instead of taking the sparsified pattern of A^2 or A^3 as the new sparsity pattern of M, which needs to be recomputed. One of the reasons is that if we take the sparsified pattern of A^2 or A^3 as the new sparsity pattern of M to recompute, we will lose the information that has been already obtained with the sparsified pattern of A. By making use of the new method, we can utilize the information such as in m_j and the residual r to get a better result because the new method can guarantee that $\|r_{k+1}\|_2 < \|r_k\|_2$.

The new algorithm for constructing a sparse approximate inverse preconditioner can be stated an follows:

Algorithm 4 (construction of a sparse approximate inverse preconditioner):

1. Let M have the sparsified pattern of A with threshold ε and set $m_j = Me_j$;
2. For each column $j = 1 : n$,
3. select the index vector $J = \{i_1, i_2, ..., i_m\} \in \{1, 2, ..., n\}$, where $|m_j(i_k)| \neq 0$, $k = 1, 2, ..., m$, and set $E = [e_{i_1}, e_{i_2}, ..., e_{i_m}]$,
4. solve $(E^T A^T AE)y = E^T A^T r$ for y,
5. compute $m_j = m_j + Ey$,
6. compute $r = r - AEy$;
7. For $i = 1 : n_p$,
8. select the index vector $J = \{i_1, i_2, ..., i_m\} \in \{1, 2, ..., n\}$, where $|r(i_j)| \geq \eta$, and set $E = [e_{i_1}, e_{i_2}, ..., e_{i_m}]$,
9. solve $(E^T A^T AE)y = E^T A^T r$ for y,

10. compute $m_j = m_j + Ey$,
11. compute $r = r - AEy$;
12. Enddo
13. Enddo

Remarks: The method with the sparsified pattern of A as the initial pattern always produces a good preconditioner with a small number of n_p. In addition, this method needs very little communication between the processors because of the small number of n_p. Of course, if the initial system is difficult to solve, we can also take the sparsified pattern of A^2 as the initial pattern. The numerical experiments carried out in the next section show that this method can produce a very robust and sparse preconditioner.

4. Numerical Experiments

In this section, we present numerical examples with matrices chosen from the Harwell-Boeing sparse matrix collection which are used typically in [6, 8, 12] for comparison. A brief description of these matrices is given in Table 1. In this table, n and nnz represent the size of the matrix and the number of nonzero entries, respectively. In this paper, we are not concerned with the implementation of the parallel algorithm so the numerical experiments are performed in the sequential environment. We emphasize that the algorithms presented here are inherently parallel. All the numerical experiments presented in this section are computed in double precision using MATLAB.

Table 2: The information of the test matrices

Matrix	n	nnz	Description
SHERMAN1	1000	3750	Oil reservoir simulation $10 \times 10 \times 10$ grid
SHERMAN3	5005	20033	Oil reservoir simulation $35 \times 11 \times 13$ grid
SHERMAN5	3312	20793	Oil reservoir simulation $16 \times 23 \times 3$ grid
ORSIRR1	1030	6858	Oil reservoir simulation $21 \times 21 \times 5$ grid
PORES2	1224	9613	Reservoir simulation
SAYLR4	3564	22316	3D reservoir simulation
FIDAP008	3096	106302	Finite element modeling
FIDAP009	3363	99397	Finite element modeling
FIDAP010	2410	54816	Finite element modeling

In all tables, ε and η denote the corresponding ε and η in **Algorithm 4**, and m represents the mean of m in this algorithm. ALG4(i) denotes **Algorithm 4** with $n_p = i$. To solve the linear equation, we use the BiCGSTAB method and the stopping criterion is 10^{-7}. The right preconditioning technique is used. P-iter is the number of BiCGSTAB iterations for convergence, S-time is the time for solving the linear system with the BiCGSTAB method, P-time is the time for constructing the preconditioner, and T-time is the sum of S-time and P-time. Time is in seconds. The symbol $-$ indicates that the corresponding method is not convergent within 1,000 steps.

In Table 2, the numerical results for matrix ORSIRR1 with different values of steps, i.e., n_p and threshold η are given. This table shows the effect of an increment

Table 3: ORSIRR1 ($\varepsilon = 0.5$)

n_p	η	m	Density	P-iter	S-time	P-time	T-time
0	0.1	0	0.21	287	0.06	0.14	0.20
1		2.6	0.39	183	0.05	0.28	0.33
2		3.3	0.57	60	0.02	0.41	0.43
3		3.5	0.67	46	0.02	0.55	0.57
4		2.5	0.79	44	0.02	0.67	0.69
5		2.3	0.80	45	0.02	0.79	0.81
1	0.01	3.0	0.50	186	0.05	0.28	0.33
2		4.8	0.74	59	0.02	0.44	0.46
3		8.4	1.31	33	0.01	0.66	0.67
4		15.6	2.49	18	0.01	1.09	1.10
5		23.8	4.20	15	0.01	1.78	1.79

in steps on the reduction of the number of the iterations for convergence, also along with the computational cost and the memory storage increase when $\eta = 0.01$. For these reasons, we usually choose n_p to be 1 or 2 for compromise. On the other hand, we can see that the ratio of the sparsity is stable along with the increment of steps when $\eta = 0.1$. The reason for this observation is that the number of the residual elements in r which are greater than 0.1 is getting fewer step-by-step. Therefore, to solve difficult matrices, a good compromise is to set $\eta = 0.1$ first and then change it to $\eta = 0.01$.

From Tables 3 and 7, we can see that more entries are dropped during the preprocessing phase along with the increment of ε. This way degrades the convergence rate of the preconditioners through increasing the powers of the sparsity of A, while the new method proposed in this paper still seems working well.

The other tables show that ALG4(1) always has a small sparse ratio than PSM(A^2) and also requires less time in constructing; ALG4(2) has similar features compared to PSM(A^3). Moreover, the threshold η leads to a good result when it is chosen to be 0.01 or 0.1.

Table 4: Matrix SHER-MAN1

Matrix	ε	m	η	Pattern	Density	P-iter	S-time	P-time	T-time
SHER-MAN1	0.5			PSM(A)	0.10	85	0.02	0.13	0.15
				PSM(A^2)	0.11	88	0.02	0.28	0.30
				PSM(A^3)	0.11	88	0.02	0.43	0.45
		2.5	0.1	ALG4(1)	0.20	44	0.01	0.24	0.25
		2.8	0.1	ALG4(2)	0.30	39	0.01	0.35	0.36
		3.3	0.01	ALG4(1)	0.26	49	0.01	0.25	0.26
		7.3	0.01	ALG4(2)	0.56	27	0.01	0.40	0.41

Table 5: Matrix SHER-MAN3

Matrix	ε	m	η	Pattern	Density	P-iter	S-time	P-time	T-time
SHER-MAN3	0.1			PSM(A)	0.69	243	0.15	1.47	1.62
				PSM(A^2)	1.57	178	0.17	3.32	3.49
				PSM(A^3)	3.18	138	0.26	0.43	6.06
		1.5	0.1	ALG4(1)	0.90	189	0.14	2.24	2.38
		1.0	0.1	ALG4(2)	0.99	173	0.13	2.92	3.05
		7.8	0.01	ALG4(1)	2.07	159	0.19	2.84	3.03

Table 6: Matrix SHER-MAN5

Matrix	ε	m	η	Pattern	Density	P-iter	S-time	P-time	T-time
SHER-MAN5	0.1			PSM(A)	0.48	60	0.04	0.81	0.85
				PSM(A^2)	1.38	258	0.21	2.21	2.42
				PSM(A^3)	3.45	25	0.04	0.43	5.65
		0.8	0.1	ALG4(1)	0.48	59	0.03	1.21	1.24
		2.0	0.01	ALG4(1)	0.65	56	0.04	1.34	1.38
		5.2	0.001	ALG4(1)	1.11	50	0.04	1.60	1.64

Table 7: Matrix SAY-LR4

Matrix	ε	m	η	Pattern	Density	P-iter	S-time	P-time	T-time
SAY-LR4	0.01			PSM(A)	0.43	–	0.54	0.92	1.46
				PSM(A^2)	0.67	695	0.47	2.46	2.93
				PSM(A^3)	0.86	465	0.40	4.00	4.40
		4.2	0.1	ALG4(1)	0.70	682	0.48	1.55	2.03
		5.2	0.1	ALG4(2)	0.92	467	0.40	2.17	2.57
	0.001			PSM(A)	0.46	–	0.57	0.93	1.50
				PSM(A^2)	0.75	647	0.47	2.52	3.00
				PSM(A^3)	1.08	501	0.42	4.14	4.56
		4.3	0.1	ALG4(1)	0.72	795	0.56	1.56	2.12
		5.0	0.1	ALG4(2)	0.93	482	0.42	2.18	2.60

5. Conclusions

In this paper, we have first introduced a new approach for solving nonsymmetric linear systems. Then, combining it with the method of patterns of a sparsified matrix, we have developed a new method for constructing sparse approximate inverse preconditioners for nonsymmetric matrices. This new method is nonsensitive to the threshold ε which is used for dropping the matrix entries during the preprocessing

Table 8: Matrix POR-ES2

Matrix	ε	m	η	Pattern	Density	P-iter	S-time	P-time	T-time
POR-ES2	0.1			PSM(A)	0.47	–	0.26	0.21	0.47
				PSM(A^2)	0.76	–	0.29	0.72	1.01
				PSM(A^3)	0.99	–	0.31	1.30	1.61
		15.4	0	ALG4(1)	1.96	506	0.21	0.72	0.93
	0.01			PSM(A)	0.69	–	0.30	0.24	0.54
				PSM(A^2)	1.79	687	0.27	0.95	1.22
				PSM(A^3)	3.79	486	0.26	2.04	2.30
		20	0	ALG4(1)	2.62	952	0.46	1.01	1.47

Table 9: Matrices FIDA-P008, FIDA-P009 and FIDA-P010

Matrix	ε	m	η	Pattern	Density	P-iter	S-time	P-time	T-time
FIDA-P008	0.1			PSM(A)	0.41	–	2.49	3.43	5.92
				PSM(A^2)	2.01	–	1.23	0.72	19.21
				PSM(A^3)	4.35	177	1.09	1.30	56.98
		90.0	0	ALG4(1)	2.64	250	1.23	32.52	33.75
FIDA-P009	0.1			PSM(A)	0.62	–	1.09	4.67	5.76
				PSM(A^2)	3.37	73	0.41	29.31	29.72
		42.5	0.01	ALG4(1)	1.51	217	0.85	19.64	20.49
FIDA-P010	0.1			PSM(A)	0.56	–	1.22	1.84	3.06
				PSM(A^2)	2.65	294	0.93	10.13	11.06
				PSM(A^3)	5.43	93	0.52	27.52	28.04
		31.7	0.01	ALG4(1)	1.45	259	0.68	6.51	7.19
		69.0	0	ALG4(1)	3.04	267	0.90	12.36	13.26

phase. The other threshold η which is used for choosing the indices always leads to a good result when it is chosen to be 0.1 or 0.01.

The numerical results have also shown favorable convergence rates and computational efficiency of the new method compared to the PSM method. This method is inherently parallel since all calculations are performed independently. Hence our future work is to focus on the parallel implementation of this method.

References

[1] O. Axelsson, Iterative Solution Methods, Cambridge University Press, Cambridge, 1996.
[2] M. Benzi, Preconditioning Techniques for Large Linear Systems: A survey, J. Comput. Phys., 182(2002) 418–477.
[3] M. Benzi and M. Tuma, A Sparse Approximate Inverse Preconditioner for Nonsymmetric Linear Systems, SIAM J. Sci. Comput., 19(1998) 968–994.
[4] Z. Chen, Finite Element Methods and Their Applications, Springer-Verlag, Heidelberg and New York, 2005.

[5] Z. Chen, G. Huan, and Y. Ma, Computational Methods for Multiphase Flows in Porous Media, in the Computational Science and Engineering Series, Vol. 2, SIAM, Philadelphia, 2006.
[6] E. Chow, A Priori Sparsity Patterns for Parallel Sparse Approximate Inverse Preconditioners, SIAM J. Sci. Comput., 21(2000) 1804–1822.
[7] E. Chow, Parallel Implementation and Pratical Use of Sparse Approximate Inverse Preconditioners with a Priori Sparsity Patterns, Int. J. High Perf. Comput. Appl., 15 (2001) 56–74.
[8] E. Chow and Y. Saad, Approximate Inverse Preconditioners Via Sparse-Sparse Iterations, SIAM J. Sci.Comput., 19(1998) 995–1023.
[9] R. Glowinski, Handbook of Numerical Analysis: Numerical Methods for Fluids, Elsevier Science Publishing Company, 2003.
[10] G.H. Golub and C.F. Van Loan, Matrix Computations, third ed., The Johns Hopkins University Press, Baltimore and London, 1996.
[11] N.I.M. Gould and J.A. Scott, Sparse Approximate-Inverse Preconditioners Using Normminimization Techniques, SIAM J. Sci. Comput., 19(1998) 605–625.
[12] M. Grote and T. Huckle, Parallel Preconditioning and Approximate Inverses, SIAM, J.Sci.Comput., 18(1997) 838–853.
[13] T. Huckle, Approximate Sparsity Patterns for the Inverse of a Matrix and Preconditioning. Appl. Numer. Math., 30(1999) 291–303.
[14] Z.X. Jia and B.C. Chen, A Power Sparse Approximate Inverse Preconditioning Procedure for Large Sparse Linear Systems, Numer. Linear Algebra Appl., 16(2009) 259–299.
[15] Y. Saad, Iterative Methods for Sparse Linear Systems, PWS Press, New York, 1995.
[16] D.K. Salkuyeh and F. Toutounian, A Sparse-Sparse Iteration for Computing a Sparse Incomplete Factorization of the Inverse of an SPD Matrix, J. Appl. Numer. Math., 59(2009) 1265–1273.
[17] W.P. Tang, Towards an Effective Sparse Approximate Inverse Preconditioner, SIAM J. Matrix Anal. Appl., 20(1999) 970–986.
[18] H.A. van der Vorst, Bi-CGSTAB: A Fast and Smoothly Converging Variant of Bi-CG for the Solution of Nonsymmetric Linear Systems, SIAM J. Sci. Stat. Comput., 12(1992) 631–644.
[19] K. Wang, S. Kim, and J. Zhang, A Comparative Study on Dynamic and Static Sparsity Patterns in Parallel Sparse Approximate Inverse Preconditioning, J. Math. Model., 2(2003) 203–215.
[20] K. Wang and J. Zhang, MSP: A Class of Parallel Multistep Successive Sparse Approximate Inverse Preconditioning Strategies, SIAM J. Sci. Comput., 24(2003) 1141–1156.

Center for Computational Geosciences, College of Mathematics and Statistics, Xi'an Jiaotong University Xi'an, 710049, Shaanxi Province, P. R. China. Department of Chemical and Petroleum Engineering, University of Calgary, Calgary, AB, Canada, T2N 1N4.

E-mail: fanyinzhe@163.com, zhachen@ucalgary.ca

URL: http://schulich.ucalgary.ca/chemical/JohnChen