A Sparse-Sparse Iteration for Computing a Sparse Incomplete Factorization of the Inverse of an SPD Matrix

Davod Khojasteh Salkuyeh[1] **and Faezeh Toutounian**[2]

[1] Department of Mathematics, University of Mohaghegh Ardabili,
P. O. Box. 56199-11367, Ardabil, Iran
E-mail: khojaste@uma.ac.ir
[2] School of Mathematical Sciences, Ferdowsi University of Mashhad,
P. O. Box. 1159-91775, Mashhad, Iran
E-mail: toutouni@math.um.ac.ir

Abstract: In this paper, a method via sparse-sparse iteration for computing a sparse incomplete factorization of the inverse of a symmetric positive definite matrix is proposed. The resulting factorized sparse approximate inverse is used as a preconditioner for solving symmetric positive definite linear systems of equations by using the preconditioned conjugate gradient algorithm. Some numerical experiments on test matrices from the Harwell-Boeing collection for comparing the numerical performance of the presented method with one available well-known algorithm are also given.

AMS Subject Classification : 65F10, 65F50.
Keywords: Sparse matrices; Factorized sparse approximate inverse; Preconditioning; Krylov subspace methods; Symmetric positive definite; Preconditioned CG algorithm

1. Introduction

Consider the nonsingular linear system of equations

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

where the coefficient matrix $A \in \mathbb{R}^{n\times n}$ is large, sparse and $x, b \in \mathbb{R}^n$. It is well-known that the rate of convergence of iterative methods such as Krylov subspace methods for solving (1) is strongly influenced by the spectral properties of A.

Hence, iterative methods usually involve a second matrix that transforms the coefficient matrix into one with a more favorable spectrum. The transformation matrix is called a preconditioner. If M is a nonsingular matrix that approximates the inverse of A ($M \approx A^{-1}$), then the transformed linear system

$$AMy = b, \quad x = My, \tag{2}$$

will have the same solution as system (1), but the convergence rate of iterative methods applied to (2) may be higher. System (2) is preconditioned from the right, but left preconditioning is also possible, i.e., $MAx = Mb$. One can also define split-preconditioned systems. Let us assume that A has the LU factorization and

$$M = M_U M_L, \quad \text{where } M_U \approx U^{-1} \text{ and } M_L \approx L^{-1}, \tag{3}$$

where L and U are the lower and upper triangular factors of A. This type of preconditioning is known as factorized approximate inverses and M_U and M_L are called approximate inverse factors of A. Here, the transformed linear system can be considered as follows

$$M_U A M_L y = M_U b, \quad x = M_L y. \tag{4}$$

System (4) is called a split-preconditioned system.

In this paper we focus our attention on the computation of sparse approximate inverse factors of a matrix. There are different ways to compute sparse approximate inverse factors of a matrix and each of them has its own advantages and disadvantages. In [5, 7], the AINV method was proposed which is based on an algorithm which computes two sets of vectors $\{z_i\}_{i=1}^n$ and $\{w_i\}_{i=1}^n$ which are A-biconjugate, i.e., such that $w_i^T A z_j = 0$ if and only if $i \neq j$. Although the construction phase for the original AINV algorithm is sequential, its application is highly parallel, since it consists of matrix-vector products. A fully parallel AINV algorithm can be achieved by means of graph partitioning (see [2, 4, 8]). For symmetric positive definite (SPD) matrices, there exists a variant of the AINV method, denoted by SAINV (for Stabilized AINV), that is breakdown-free [3]. Another approach which was proposed by Kolotilina and Yeremin is the FSAI algorithm [11, 12]. They assume that A is SPD and then construct factorized sparse approximate inverse preconditioners which are also SPD. Each factor implicitly approximates the inverse of the lower triangular Cholesky factor of A. This method can be easily

extended to the nonsymmetric case. The FSAI algorithm is inherently parallel but its main disadvantage is the need to prescribe the sparsity of approximate inverse factors in advance.

In this paper, we first propose an iterative method for solving SPD linear systems of equations, and then, by exploiting this method, we develop an algorithm for computing an incomplete factorization of the inverse of an SPD matrix. The resulting factorized sparse approximate inverse is used as an explicit preconditioner for the solution of $Ax = b$ by the preconditioned conjugate gradient (PCG) method.

Throughout the paper $\|z\|_A$ stands for the A-norm of any vector z, i.e., $\|z\|_A = (Az, z)^{1/2}$. We will denote the largest and smallest eigenvalues of the matrix X by $\lambda_{max}(X)$ and $\lambda_{min}(X)$, respectively.

This paper is organized as follows. In section 2, we introduce an approach for computing a sparse approximate solution of an SPD linear system of equations. Section 3 is devoted to computing an incomplete factorization of the inverse of an SPD matrix. Numerical experiments are given in section 4. Finally, we give some concluding remarks in section 5.

2. Sparse approximate solution of an SPD linear system of equations

In this section, we first present an approach, based on the projection method, for solving an SPD linear system of equations. Then we develop an algorithm for computing a sparse approximate solution of an SPD linear system of equations.

Let $A = (a_{ij})$ be an SPD matrix and let us consider a projection method with $\mathcal{L} = \mathcal{K} = \text{span}\{e_{i_1}, e_{i_2}, \ldots, e_{i_m}\}$, where e_{i_j} is the i_j-th column of identity matrix and m is a small natural number. Given an initial guess x of the solution of (1) and the residual vector $r = b - Ax$, the new approximation takes the form

$$x_{new} = x + Ey, \tag{5}$$

for some $y \in \mathbb{R}^m$, and $E = [e_{i_1}, e_{i_2}, \ldots, e_{i_m}]$. The Petrov-Galerkin condition $r - AEy \perp \mathcal{L}$ yields

$$y = (E^T AE)^{-1} E^T r. \tag{6}$$

As known [15], this kind of update minimizes

$$\|x + E\widetilde{y} - x_{exact}\|_A$$

over all $\widetilde{y} \in \mathbb{R}^m$, where x_{exact} is the exact solution of $Ax = b$. It is obvious that the matrix $S = E^T AE$ is an SPD matrix of dimension m. Defining $\mathcal{J} = \{i_1, i_2, \ldots, i_m\}$, the matrix $E^T AE$ is the principal submatrix of A consisting of the rows and columns whose indices are in $\mathcal{J}$. This new approach for solving an SPD linear system of equations can be stated as follows.

Algorithm 1:

1. Choose an initial guess x and compute $r = b - Ax$
2. Until convergence, Do
3. Select $\mathcal{J} = \{i_1, i_2, \ldots, i_m\} \subseteq \{1, 2, \ldots, n\}$ and $E := [e_{i_1}, e_{i_2}, \ldots, e_{i_m}]$
4. Solve $(E^T AE)y = E^T r$ for y
5. Compute $x := x + Ey$
6. Compute $r := r - AEy$
7. EndDo

Step 6 of this algorithm can be written as

$$r := r - \sum_{k \in \mathcal{J}} y_k \; a_{:,k}, \tag{7}$$

where $a_{:,k}$ is the k-th column of A and $y = [y_1, y_2, \ldots, y_m]^T$. The relation (7) shows that, for updating r, we need m sparse SAXPY operations (a SAXPY operation is defined as $z := x + \alpha y$, where x and y are n-vectors and α is a scalar). The following theorem regarding the convergence rate of the Algorithm 1 can be stated.

Theorem 1. *Let A be a symmetric positive definite matrix. Assume that, at each projection step, the selected index set $\{i_1, i_2, \ldots, i_m\}$ contains the indices of the m components with largest absolute value in the current residual vector $r = b - Ax$. Then*

$$\|d\|_A^2 - \|d_{new}\|_A^2 \geq \frac{\sum_{k \in \mathcal{J}} r_k^2}{\sum_{k \in \mathcal{J}} a_{kk}}, \tag{8}$$

and

$$\|d_{new}\|_A \leq (1 - (\frac{\lambda_{min}(A)}{\sum_{k\in\mathcal{J}} a_{kk}})(\frac{\sum_{k\in\mathcal{J}} r_k^2}{\sum_{k=1}^{n} r_k^2}))^{1/2}\|d\|_A, \tag{9}$$

where $d_{new} = A^{-1}b - x_{new}$ *and* $d = A^{-1}b - x$. *Relation (9) shows that Algorithm 1 converges for any initial guess.*

Proof. We start by observing that $d_{new} = d - Ey$, $Ad = r$, and

$$(Ad_{new}, d_{new}) = (Ad, d) - (y, E^T r).$$

From (6) and using the Courant-Fisher min-max theorem [1, 15], we have

$$\begin{aligned}(y, E^T r) &= ((E^T AE)^{-1}E^T r, E^T r)\\ &\geq \frac{\|E^T r\|_2^2}{\lambda_{max}(E^T AE)}\\ &\geq \frac{\|E^T r\|_2^2}{\sum_{k\in\mathcal{J}} a_{kk}},\end{aligned}$$

and

$$(Ad, d) = (r, A^{-1}r) \leq \frac{\|r\|_2^2}{\lambda_{min}(A)}.$$

From these observations the desired results immediately follow. Relation (9) establishes the convergence of the method, since $\sum_{k\in\mathcal{J}} r_k^2 \leq \sum_{k=1}^{n} r_k^2$ and $\lambda_{min}(A) \leq \lambda_{min}(E^T AE) \leq \sum_{k\in\mathcal{J}} a_{kk}$ (see [1]). □

This theorem not only shows the convergence of the algorithm but also the rate of the reduction in the square of the A-norm of the error (Eq. (8)). In fact, the indices $i_j, j = 1, \ldots, m$ are chosen in such a way that the reduction in the square of the A-norm of the error is as large as possible. If A is a symmetric diagonally scaled matrix then $\sum_{k\in\mathcal{J}} a_{kk} = m$ and in the Eqs. (8) and (9), $\sum_{k\in\mathcal{J}} a_{kk}$ may be replaced by m.

Now, by using Algorithm 1, we propose an algorithm to compute a sparse approximate solution of an SPD linear system of equations. In this algorithm no dropping strategy is needed and sparsity of the solution is preserved only by specifying the maximum number of its nonzero entries, $lfil$, in advance. In each

iteration at most m ($m \ll n$) entries are added to the current approximate solution. This algorithm can be stated as follows.

Algorithm 2 : Sparse approximate solution to the SPD system $Ax = b$

1. Set $x := 0$ and $r := b$
2. While $\| r\| > eps$ and $nnz(x) < lfil$ Do
3. Select the indices of m components with largest absolute value in the current residual vector r, i.e., $\mathcal{J} = \{i_1, i_2, \ldots, i_m\} \subseteq \{1, 2, \ldots, n\}$ and set $E := [e_{i_1}, e_{i_2}, \ldots, e_{i_m}]$
4. Solve $(E^T AE)y = E^T r$ for y
5. Compute $x := x + Ey$
6. Compute $r := r - AEy$
7. EndDo

The vector x computed by Algorithm 2 has at most $lfil$ nonzero entries. In practical implementations of Algorithm 2 the number m is usually chosen to be too small, for example $m = 1, 2$ or 3. Throughout this paper we take $m = 2$. The parameter *eps* is used for stopping the process when the residual norm is small enough. As can be seen, in Algorithm 2 no dropping strategy is used and in each step of the algorithm, according to Theorem 1, the A-norm of the error is reduced.

3. Approximate inverse factors of a matrix via sparse-sparse iterations

In this section, for computing a sparse factorized approximate inverse of an SPD matrix, we combine Algorithm 2 of section 2 with the AIB (Approximate Inverse via Bordering) algorithm proposed by Saad in [15]. We first give a brief description of the AIB algorithm for symmetric matrices.

In the AIB algorithm, the sequence of matrices

$$A_{k+1} = \begin{pmatrix} A_k & v_k \\ v_k^T & \alpha_{k+1} \end{pmatrix}, \tag{10}$$

is made in which $A_n = A$. If the inverse factor U_k is available for A_k, i.e.,

$$U_k^T A_k U_k = D_k, \tag{11}$$

then the inverse factor U_{k+1} for A_{k+1} will be obtained by writing

$$\begin{pmatrix} U_k^T & 0 \\ -z_k^T & 1 \end{pmatrix} \begin{pmatrix} A_k & v_k \\ v_k^T & \alpha_{k+1} \end{pmatrix} \begin{pmatrix} U_k & -z_k \\ 0 & 1 \end{pmatrix} = \begin{pmatrix} D_k & 0 \\ 0 & \delta_{k+1} \end{pmatrix}, \tag{12}$$

in which

$$A_k z_k = v_k, \tag{13}$$

$$\delta_{k+1} = \alpha_{k+1} - z_k^T v_k. \tag{14}$$

Relation (14) can be exploited if the system (13) is solved exactly. Otherwise we should use

$$\begin{aligned} \delta_{k+1} &= \alpha_{k+1} - v_k^T z_k - z_k^T (v_k - A_k z_k) \\ &= \alpha_{k+1} - v_k^T z_k - z_k^T r_k \\ &= \alpha_{k+1} - z_k^T (v_k + r_k), \end{aligned} \tag{15}$$

instead of (14), where $r_k = v_k - A_k z_k$. Starting from $k = 1$, this procedure suggests an algorithm for computing the inverse factors of A. If a sparse approximate solution of (13) is computed, then an approximate factorization of A^{-1} is obtained. To do this, we use Algorithm 2 of section 2. This scheme can be summarized as follows.

Algorithm 3. AIB algorithm

1. Set $A_1 = [a_{11}]$, $U_1 = [1]$ and $\delta_1 = a_{11}$
2. For $k = 1, \ldots, n-1$ Do: (in parallel)
3. Compute a sparse approximate solution to $A_k z_k = v_k$,
   by using Algorithm 2, and the residual $r_k = v_k - A_k z_k$.
4. Compute $\delta_{k+1} = \alpha_{k+1} - z_k^T (v_k + r_k)$.
5. Form U_{k+1} and D_{k+1}
6. EndDo.
7. Set $U := U_n$ and $D := D_n$

This algorithm returns U and D such that $U^TAU \approx D$. The following theorem shows that δ_{k+1} is always positive, independently of the accuracy with which the system (13) is solved.

Theorem 2. *Let A be an SPD matrix. Then, the scalar δ_{k+1} computed in step 4 of Algorithm 3 is positive.*

Proof. Let r_k be the residual obtained in step 3 of the AIB algorithm, i.e.,

$$r_k = v_k - A_k z_k.$$

Hence, we have

$$z_k = A_k^{-1}(v_k - r_k).$$

By a little computation one can see that

$$\delta_{k+1} = \alpha_{k+1} - v_k^T A_k^{-1} v_k + r_k^T A_k^{-1} r_k = s + r_k^T A_k^{-1} r_k,$$

where $s = \alpha_{k+1} - v_k^T A_k^{-1} v_k \in \mathbb{R}$ is the Schur complement of A_{k+1} and is a positive real number (see Theorem 3.9 in [1]). So, the scalar δ_{k+1} is positive, since A is an SPD matrix and $r_k^T A_k^{-1} r_k > 0$ for $r_k \neq 0$. □

Hence the AIB algorithm is well-defined for SPD matrices.

4. Numerical examples

All the numerical experiments presented in this section were computed in double precision using Fortran PowerStation version 4.0 on a Pentium 4 PC, with a 3.06 GHz CPU and 1.00GB of RAM.

For the first set of the numerical experiments, we used nine SPD matrices (BCSSTK* and S*RMT3M*) from the Matrix-Market website [14] and three matrices (EX15, MSC04515 and KUU) from Tim Davis's collection [10]. These matrices with their generic properties are given in Table 1. For each matrix, the problem size n and the number of nonzero entries in the lower triangular part nnz are provided. In last two columns, the number of iterations (iters) and time required to solve the linear system using the conjugate gradient method without any scaling are given. The time was measured with the function `etime()` and given in seconds. The stopping criterion

$$\frac{\|b - Ax_i\|_2}{\|b\|_2} < 10^{-8},$$

Table 1: First set of test problems information.

matrix	n	nnz	time	iters
BCSSTK11	1473	17857	-	†
BCSSTK13	2003	42943	-	†
BCSSTK15	3948	60882	29.07	9219
BCSSTK21	3600	15100	15.32	7805
BCSSTK38	8032	181746	-	†
S1RMT3M1	5489	112505	24.13	4953
S2RMT3M1	5489	112505	-	†
S3RMT3M1	5489	112505	-	†
S3RMT3M3	5357	106526	-	†
MSC04515	4515	51111	15.12	4728
EX15	6867	52769	6.48	1506
KUU	7102	173651	3.84	550

was used and the initial guess was taken to be the zero vector. For all the examples, the right hand side of each system was taken such that the exact solution is a vector with random entries uniformly distributed in $(0, 1)$. No significant differences were observed for other choices of the right hand side vector. The maximum number of iterations was 10000. In all the tables a dagger (†) indicates no convergence of the iterative method.

We compare the numerical results of the new preconditioner with that of the SAINV preconditioner. The AINV and the SAINV algorithms have been widely compared with other preconditioning techniques, showing that they are the most effective algorithms for computing a sparse incomplete factorization of the inverse of a matrix [2, 3, 5, 6, 7]. For the SAINV algorithm we used the SAINV code of the SPARSLAB software provided by Tuma [1] with drop tolerance $\tau = 0.1$. This drop tolerance is very often the right one based on the numerical results reported in several papers. For Algorithm 2, we used the parameters $eps = 0.01$ and $lfil = 10$. We also used a parameter $lfil$ such that the number of nonzero entries in the incomplete U factor divided by the number of nonzero entries in the upper triangular part of A, ρ, is approximately equal to or less than that of

[1] http://www.cs.cas.cz/∼tuma/sparslab.html

Table 2: Setup time to compute sparse approximate inverse factors and results for the split-preconditioned CG algorithm.

	Algorithm 3						SAINV Algorithm				
matrix	$lfil$	ρ	P-Its	P-time	It-time	T-time	ρ	P-Its	P-time	It-time	T-time
BCSSTK11	13	0.58	628	0.23	1.20	1.43	0.58	2099	0.11	3.95	4.06
	10	0.45	650	0.17	1.19	1.36					
BCSSTK13	29	0.71	343	0.65	1.42	2.07	0.72	370	0.63	1.55	2.18
	10	0.26	550	0.10	1.80	1.90					
BCSSTK15	9	0.35	504	0.19	2.67	2.86	0.32	214	0.22	1.11	1.33
	10	0.37	491	0.20	2.73	2.93					
BCSSTK21	6	0.97	246	0.09	0.73	0.82	1.05	167	0.06	0.5	0.56
	10	1.48	164	0.13	0.52	0.65					
BCSSTK38	11	0.28	559	0.70	7.72	8.42	0.29	1131	0.81	15.67	16.48
	10	0.25	572	0.64	7.77	8.41					
S1RMT3M1	15	0.41	244	0.45	2.28	2.73	0.83	257	0.98	2.97	3.95
	10	0.29	318	0.32	2.77	3.09					
S2RMT3M1	15	0.41	526	0.58	4.97	5.55	1.29	539	1.53	7.5	9.03
	10	0.28	585	0.34	5.11	5.45					
S3RMT3M1	15	0.36	1298	0.65	11.89	12.54	2.81	5434	6.33	117.80	124.13
	10	0.26	1458	0.33	12.55	12.88					
S3RMT3M3	15	0.37	984	0.76	8.58	9.34	2.00	5047	3.83	84.40	88.23
	10	0.26	1087	0.31	8.95	9.26					
MSC04515	13	0.65	712	0.28	4.16	4.44	0.66	1020	0.20	5.93	6.13
	10	0.52	809	0.22	4.47	4.69					
EX15	20	1.11	601	0.81	4.88	5.69	1.83	1325	0.55	12.88	13.43
	10	0.65	511	0.34	3.53	3.87					
KUU	8	0.19	141	0.38	1.70	2.08	0.18	144	0.25	1.73	1.98
	10	0.23	131	0.42	1.67	2.09					

the SAINV preconditioner. The results of the split-preconditioned CG algorithm [15] in conjunction with the SAINV preconditioner and Algorithm 3 are given in Table 2. This table reports the density (ρ), the number of split-preconditioned CG iterations for convergence (P-Its), the setup time for the preconditioner (P-time), the time for the split-preconditioned iterations (It-time), and T-time which is equal to the sum of P-time and It-time. Numerical results presented in this table show that both algorithms are robust and the new method is better than the SAINV algorithm for 9 out of 12 problems, especially on the shell problems (S3RMT3M1 and S3RMT3M3). The results of this table also indicate that the parameters $eps = 0.01$ and $lfil = 10$ give good results.

In Table 3, the numerical results for matrices BCSSTK13 and BCSSTK27 with different values of $lfil$ are given. This table shows the effect of an increase in $lfil$ on the reduction of the number of the iterations for convergence. The results of this table also indicate that the choices $eps = 0.01$ and $lfil = 10$ lead to good results.

In [3], the numerical results of the SAINV preconditioner in conjunction with some preliminary transformations operated on the coefficient matrix such as symmetric diagonal scaling, reordering with the multiple minimum degree (MMD) al-

Table 3: Results for matrices BCSSTK13 and BCSSTK21 with different values of $lfil$

	BCSSTK13			BCSSTK21		
$lfil$	P-time	It-time	P-Its	P-time	It-time	P-Its
2	0.03	3.02	1039	0.06	0.81	316
4	0.05	2.77	917	0.08	0.75	270
6	0.06	2.48	793	0.09	0.72	246
8	0.08	2.19	685	0.10	0.61	198
10	0.11	1.81	550	0.11	0.53	164
12	0.14	1.69	514	0.14	0.50	148
14	0.17	1.84	529	0.17	0.50	142
16	0.22	1.78	502	0.20	0.47	125

gorithm [13] and diagonally compensated reduction of positive off-diagonal entries (DCR), were given. The authors have concluded that the SAINV preconditioner in conjunction with symmetric diagonal scaling and reordering with MMD (J-MMD-SAINV) is often the best choice between the variants of the preconditioners used in [3]. In continuation, we use 14 out of 16 matrices used in [3] for the numerical experiments. Most of these matrices can be extracted from the Matrix Market website. The exceptions are NASA2910 and NASA4704 which can be downloaded from Tim Davis's collection, and the SMT matrix, which was provided by R. Kouhia of the Helsinki University of Technology [2]. In Table 4, we give the numerical results of the new preconditioner in conjunction with symmetric diagonal scaling (J-N-M) and the J-MMD-SAINV preconditioner. Results of the J-MMD-SAINV preconditioner and the number of iterations of the conjugate gradient algorithm with Jacobi preconditioning (JCG-Its) were extracted from Table 7 and Table 1 in [3], respectively. It is necessary to mention that the parameter $lfil$ was chosen such that the parameter ρ of the new preconditioner is less than or approximately equal to that of the SAINV preconditioner. All assumptions and notations are as before.

Table 4 shows that the new preconditioner is better than the J-MMD-SAINV preconditioner for 9 out of 14 matrices presented in this table.

The last section of numerical experiments is devoted to some large matrices

[2] http://users.tkk.fi/∼kouhia/sparse.html

Table 4: Numerical results of the new method in conjunction with symmetric diagonal scaling and the J-MMD-SAINV preconditioner.

			J-N-M						J-MMD-SAINV		
matrix	n	nnz	$lfil$	ρ	P-Its	P-time	It-time	T-time	ρ	P-Its	JCG-Its
BCSSTK13	2003	42943	17	0.38	275	0.17	0.94	1.11	0.39	349	1406
BCSSTK14	1806	32630	9	0.28	83	0.06	0.23	0.29	0.27	73	409
BCSSTK15	3948	60882	11	0.32	176	0.17	0.95	1.12	0.33	167	518
BCSSTK16	4884	147631	6	0.12	95	0.20	0.89	1.09	0.12	98	191
BCSSTK17	10974	219812	16	0.40	653	1.19	12.03	13.22	0.40	711	2522
BCSSTK18	11948	80519	8	0.57	515	0.78	5.73	6.51	0.58	261	1120
BCSSTK21	3600	15100	10	1.46	179	0.11	0.58	0.69	1.51	191	559
BCSSTK25	15439	133840	10	0.59	1614	1.45	26.86	28.31	0.57	1512	†
S1RMQ4M1	5489	143300	8	0.19	247	0.27	2.41	2.68	0.20	248	692
S2RMQ4M1	5489	143300	10	0.23	403	0.33	3.98	4.31	0.25	528	1529
S3RMQ4M1	5489	143300	13	0.24	569	0.36	5.66	6.02	0.24	1140	6884
NASA2910	2910	88603	20	0.32	262	0.41	1.64	2.05	0.95	341	1350
NASA4704	4704	54730	20	0.85	567	0.47	3.77	4.24	0.91	1176	4866
SMT	25710	1889447	15	0.11	734	6.28	74.83	81.11	0.11	546	1984

Table 5: Numerical results of the new method in conjunction with symmetric diagonal scaling.

			New method with symmetric diagonal scaling						JCG	
matrix	n	nnz	$lfil$	ρ	P-Its	P-time	It-time	T-time	Its	Time
GRIDGENA	48962	280523	10	1.07	609	11.36	29.85	41.21	1720	48.61
			15	1.52	540	11.98	29.34	41.32		
			20	2.00	464	13.92	27.89	41.81		
CVXBQP1	50000	199984	10	1.14	1053	11.38	45.33	56.71	3330	90.95
			15	1.50	886	11.80	41.00	52.80		
			20	1.79	750	12.28	36.84	49.12		
APACHE1	80800	311492	10	1.43	325	29.44	24.20	53.64	1796	79.56
			15	1.80	245	30.36	19.48	49.84		
			20	2.22	218	31.44	18.52	49.96		
CF2D2	123440	1605669	10	0.45	1002	68.63	200.16	268.79	3870	368.91
			15	0.65	850	71.17	179.8	250.97		
			20	0.85	717	73.88	164.23	238.11		

extracted from Tim Davis's collection. Numerical results with different values of $lfil$ and with $eps = 0.01$ are given in Table 5. As can be seen, the new preconditioner in conjunction with symmetric diagonal scaling furnishes good results for large matrices.

We end this section by giving the numerical results for the matrix APACHE2 extracted from Tim Davis's collection. This is a large matrix of dimension $n = 715176$ with $nnz = 2776523$ nonzero entries in the lower part. The conjugate gradient method in conjunction with symmetric diagonal scaling converges in 2482 iterations. The conjugate gradient method in conjunction with the new preconditioner and symmetric diagonal scaling with $lfil = 5$ ($\rho = 0.98$) and $lfil = 10$ ($\rho = 1.52$) converges in 839 and 575 iterations, respectively. Convergence history of these

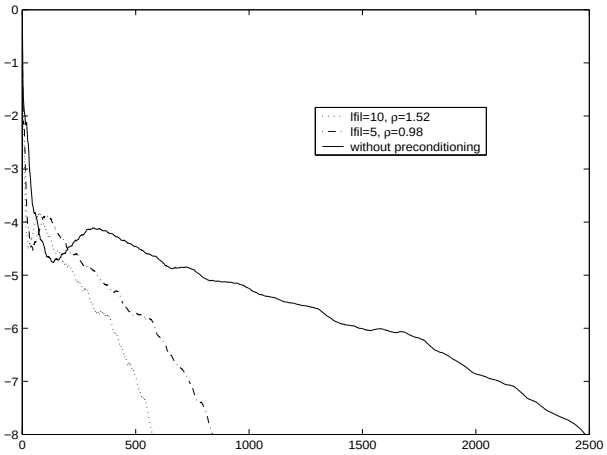

Figure 1: Convergence history of the new preconditioner for the matrix APACHE2.

methods is displayed in Figure 1.

Numerical results for the matrix APACHE2 and matrices in Table 5 (and previous tables) show that the new preconditioner reduces the number of iterations by about a factor of three. This is true for the J-MMD-SAINV preconditioner based upon a conclusion reported in ([3], page 1328).

5. Conclusion and future work

We have proposed an approach for computing a sparse incomplete factorization of the inverse of an SPD matrix. The resulting factorized sparse approximate inverse was used as a preconditioner for solving symmetric positive definite linear systems of equations by using the conjugate gradient algorithm. The new preconditioner does not need to specify the sparsity pattern of the inverse factor in advance. For preserving sparsity it is enough to specify two parameters *eps* and $lfil$. Numerical results show that $eps = 0.01$ and $lfil = 10$ usually give good results. Our numerical results also show that the proposed method in conjunction with the symmetric diagonal scaling is somewhat better than the J-MMD-SAINV preconditioner.

The new preconditioner is suitable for parallel computers. It can also be implemented for normal equations with a little revision.

Future work may focus on extending the proposed preconditioner to general matrices and studying the effect of different reordering techniques on the convergence rate.

Acknowledgments

The authors are gratefully indebted to Edmond Chow for carefully reading of an earlier draft of this paper and giving several valuable comments. The authors also are grateful to the anonymous referees and our editor Dr. Eric de Sturler for their comments which substantially improved the quality of this paper.

References

[1] O. Axelsson, *Iterative solution methods*, Cambridge University Press, Cambridge, 1996.

[2] M. Benzi, *Preconditioning techniques for large linear systems: A survey*, J. of Computational Physics, 182 (2002) 418-477.

[3] M. Benzi, J. K. Cullum, and M. Tuma, and C. D. Meyer, *Robust approximate inverse preconditioning for the conjugate gradient Method,* SIAM J. Sci. Comput., 22 (2000) 1318-1332.

[4] M. Benzi, J. Marin and M. Tuma *A Two-Level Parallel Preconditioner Based on Sparse Approximate Inverses*, in Iterative Methods in Scientific Computation IV, D. R. Kincaid and A. C. Elster, eds., IMACS Series in Computational and Applied Mathematics, Vol. 5, IMACS, New Brunswick, NJ (1999), pp. 167-178.

[5] M. Benzi, C. D. Meyer, and M. Tuma, *A sparse approximate inverse preconditioner for the conjugate gradient method,* SIAM J. Sci. Comput., 17 (1996) 1135-1149.

[6] M. Benzi, M. Tuma, *A comparative study of sparse approximate inverse preconditioners*, Applied Numerical Mathematics, 30 (1999)305-340.

[7] M. Benzi, M. Tuma, *A sparse approximate inverse preconditioner for nonsymmetric linear systems*, SIAM J. Sci. Comput., 19 (1998) 968-994.

[8] M. Benzi and M. Tuma, *A parallel solver for large-scale Markov chains*, Appl. Numer. Math., 41(2002) 305-340.

[9] B. N. Datta, *Numerical Linear Algebra and Applications*, Brooks Cole Publishing Company, 1995.

[10] T. Davis, *University of Florida sparse matrix collection*, NA Digest, 92(1994), http://www.cise.ufl.edu/research/sparse/matrices.

[11] L. Y. Kolotilina and A. Y. Yeremin, *Factorized sparse approximate inverse preconditioning I. Theory*, SIAM J. Matrix Anal. Appl., 14 (1993) 45-58.

[12] L. Y. Kolotilina and A. Y. Yeremin, *Factorized sparse approximate inverse preconditioning II: Solution of 3D FE systems on massively parallel computers*, Int. J. High Speed Comput., 7 (1995) 191-215.

[13] J. W. H. Liu, *Modification of the minimum degree algorithm by multiple elimination*, ACM Trans. Math. Software, 11(1985)141-153.

[14] Matrix Market, http://math.nist.gov/MatrixMarket (August 2005).

[15] Y. Saad, *Iterative Methods for Sparse linear Systems*, PWS press, New York, 1995.