Parallel Sparse Approximate Inverse Preconditioning with Neumann Expansion

Yuan Hong and Jun Zhang
Laboratory for High Performance Scientific Computing and Computer Simulation,
Department of Computer Science, University of Kentucky,
Lexington, KY 40508

January 10, 2004

Abstract

We investigate the use of Neumann expansion in generating sparsity patterns in computing parallel sparse approximate inverse (SAI) preconditioners. This is implemented in the framework of multistep successive sparse approximate inverse preconditioning (MSP) strategies. Numerical experiments with the new sparsity pattern selection strategy are compared with the original MSP strategies. Their advantages and disadvantages are discussed.

1 Introduction

The general idea to find a preconditioner for the linear system

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

is to find a nonsingular matrix M such that M is a good approximation to A^{-1} (or A) in some senses and that the system $MAx = Mb$ (or $M^{-1}Ax = M^{-1}b$) is much easier to solve by an iterative solver than Eq. (1). The standard incomplete LU (ILU) factorization preconditioners often fail for general sparse indefinite matrices, because they may give rise to unstable factors L and U by disallowing certain fill-ins. They also are notoriously difficult to implement on parallel architectures, especially for some unstructured general sparse matrices. In such cases, it may be attractive to approximate the inverse of the matrix directly. The driving motivation in search for efficient sparse approximate inverse (SAI) preconditioners has been their inherent parallelism and low computational cost based on sparse-sparse vector product [7].

Several techniques have been developed to construct SAI preconditioners. They roughly fall into three main categories: SAI based on the Frobenius norm minimization, SAI computed from an ILU factorization, and factored SAIs. Each of them has its own merits and limitations [2].

With the Frobenius norm minimization, the most important part of the SAI computation is to choose a good sparsity pattern for M. Both dynamic and static pattern selection strategies have been proposed and compared. In parallel implementations, it seems that the static pattern strategy is more cost effective [10]. But selecting good sparsity patterns for unstructured general sparse matrices is still a challenging problem [9, 10].

Recently, a multistep successive sparse approximate inverse preconditioning (MSP) strategy is proposed to compute robust SAI preconditioners [9]. In MSP, a sequence of low cost sparse matrices are computed to achieve the effect of a high accuracy preconditioner. The products of these sparse matrices is used to approximate the inverse of the original matrix. Lower storage cost and higher robustness and efficiency are obtained from the MSP strategy, compared with the standard SAI preconditioners. The key idea in the MSP strategy is to adjust the sparsity pattern step by step when a series of SAI matrices are computed.

In this paper, we follow the MSP strategy by using successive matrix approximation. But we propose a new strategy to choose the sparsity pattern for the next step SAI matrix. This new strategy comes from the Neumann expansion of the inverse of a matrix. With careful manipulations, the sparsity pattern comes from the error matrix of the previous SAI matrix and is obtained easily. We demonstrate the effectiveness of this sparsity pattern selection strategy by conducting a few numerical experiments involving several sparse matrices.

This paper is organized as follows. We discuss our ideas, motivations, and implementation for the new strategy in Section 2. Numerical experiments are conducted and results are discussed in Section 3. A brief concluding remark is given in Section 4.

2 MSP and Neumann Expansion

2.1 Multistep successive SAI preconditioning

In [9], Wang and Zhang present a recursive method to gradually approach the inverse of the matrix A. First, a simple and inexpensive SAI matrix M_0 is computed for the matrix A by using the sparsified sparsity pattern of A. If M_0 is not good enough, explicitly saying that $\|M_0A - I\|$ is not smaller than the threshold we set, another SAI matrix M_1 can be computed for the new preconditioned system

$$M_0 Ax = M_0 b, \tag{2}$$

which is equivalent to the original system (1). Assuming that M_0 is nonsingular the newly constructed sparse matrix M_1 is the approximate inverse to the matrix $A_1 = M_0 A$. The combined product matrix of $M_1 M_0$ is used to approximate the inverse of A. If $M_1 M_0$ is still not good enough, a third SAI matrix M_2 can be computed for the new system $M_1 M_0 Ax = M_1 M_0 b$. A sequence of sparse matrices $M_0, M_1, M_2, \ldots, M_k$ can be ultimately obtained such that $M_k M_{k-1} \cdots M_1 M0 \approx A^{-1}$. The MSP strategy is to some extent like the factored SAI preconditioners [3]. To avoid a dense preconditioner, Wang and Zhang [9] use the products of a series of sparse matrices, hoping that the product of these matrices can hold more information of A^{-1} than just a single matrix M_0.

2.2 Sparsity pattern selection in MSP

Sparsity pattern has a great influence on the efficiency and the performance of the SAI preconditioning techniques, which require a suitable strategy to identify the nonzero entries in the SAI matrix M, The idea is to keep the matrix M reasonably sparse while still maintaining the essential entries of the inverse.

There are two basic strategies to determine the sparsity pattern: dynamically or statically. The use of dynamic approach may construct the nonzero pattern of the preconditioner by monitoring the residual in the least-squares problems during the computing process [8, 5]. Although it is more accurate and effective, dynamic sparsity pattern is usually very expensive since its poor parallelism in the construction update phase leads to a large amount of data movements [10]. The static approach, on the other hand, requires an *a priori* nonzero pattern for the preconditioner [6]. Some research has demonstrated that it is also very expensive and costly to compute high accuracy SAI preconditioners solely based on a static sparsity pattern. This is partially due to the significant difficulty in finding a very good sparse pattern in advance, which can also be problem dependent [9].

According to the Green's function, the inverse of a matrix A may own a roughly similar sparsity structure to A itself if they both are sparsified to drop some small entries. Although the sparsity patterns of A^2 and A^3 do lead to higher accuracy SAI, their high construction costs usually overwhelm any speedups they provide to the iteration process. Thus the MSP strategy is attractive in balancing the accuracy of the computed preconditioner and the cost of its construction.

The MSP strategies of choosing sparsity pattern at each step are shown to be reasonable and effective in their numerical experiments [9]. However, deeper works can be done to further speedup the convergence and reduce the number of iterations for MSP. We currently focus on how to refine the sparsity pattern for the next step preconditioner computation, i.e., how to predict the sparsity pattern of the next step preconditioner M_k for the product matrix A_k more accurately.

At each step we have

$$A_{i+1} = M_i^{-1} - Ni,$$

in which M_i is the approximate inverse of A_{i+1} and N_i is the error matrix of the approximate inverse to the exact one. As always, we assume that all M_i's are nonsingular. Thus, from another view of point it is a splitting of the matrix A into two parts

$$A = P - Q. \tag{3}$$

This viewpoint may help us exploit polynomial preconditioning idea. Polynomial preconditioners are based on the Neumann expansion [1, 11]

$$A^{-1} = (I + G + G^2 + \cdots + G^{m-1})P^{-1}, \tag{4}$$

such that the spectral radius of $G = P^{-1}Q$ is less than 1. Here I is an identity matrix.

The Neumann expansion (4) can also be expressed in another form

$$A^{-1} = (I + G^{2^s})(I + G^{2^{s-1}}) \cdots (I + G^2)(I + G)P^{-1}. \tag{5}$$

Here s is some integer. From (5) we can find that the form of the inverse of A is similar to the product of the sequence of SAI matrices in the MSP strategy. A truncated form of series (5) can be used to approximate A^{-1}. Without loss of generality, we only consider the first step of the preconditioning computation. In the first step, we can find

$$A = M_0^{-1} - N_0. \tag{6}$$

According to (5),

$$A^{-1} = (I + G_0^{2^s})(I + G_0^{2^{s-1}}) \cdots (I + G_0^2)(I + G_0)M_0, \tag{7}$$

with $G_0 = M_0 N_0$ [4].

Before we go further, we should consider if the spectral radius of the matrix G_0 is less than 1. In fact, we do not need to directly compute G_0. From (6) we can get $G_0 = M_0 N_0 = I - M_0 A$ which is the residual of the least-squares problems in the Frobenius norm minimization approach. In [8] Grote and Huckel proved some essential properties of the matrix $M_0 A$. Our experiments have shown that the spectral radius of the matrix G_0 is always less than 1 if the norm of the expression $\|M_0 A - I\|$ is less than a certain threshold value. In fact if M_0 is an approximation to A, then $A_1 = M_0 A$ tends to be closer to I than A. Therefore we hope that $M_0 A - I$ is closer to the zero matrix which often has a small spectral radius.

According to the MSP definition we have

$$M_1 = (M_0 A)^{-1} = A^{-1} M_0^{-1}. \tag{8}$$

Combining with Eqs. (7) and (8) we find

$$M_1 = (I + G_0^{2^s})(I + G_0^{2^{s-1}}) \cdots (I + G_0^2)(I + G_0) \approx I + G_0. \tag{9}$$

Since the matrix G_0 trends to be a zero matrix and

$$\lim_{s \to +\infty} (I + G_0^{2^s}) \approx I,$$

we hope that the sparsity pattern of $I + G_0$, used for computing M_1, will be better than the original one $M_0 A$ in the original MSP strategy. We will give some test results to justify this argument. In MSP, we can recursively use the current computed preconditioner to predict the next good sparsity pattern.

The new algorithm for computing an enhanced MSP can be written as follows.

ALGORITHM **2.1** Enhanced Multistep Successive Preconditioning Algorithm.

1. Let l=stepNum, $A_0 = A$, $G_0 = A$
2. For ($i = 1$; i<l; i++)
3. Sparsify A_{i-1}
4. Set up sparsity pattern based on G_{i-1}
5. Compute an SAI $M_{i-1} \approx A_{i-1}^{-1}$
6. Sparsify M_{i-1} by dropping small entries
7. Compute $A_i = M_{i-1} A_{i-1}$
8. Compute $G_{i-1} = I - A_i$
9. Compute $G_{i-1} = I + G_{i-1}$
10. End;
11. Sparsify A_l
12. Compute an SAI $M_{i-1} \approx A_l^{-1}$
13. Sparsify M_l by dropping small entries
14. Return $\prod_{i=0}^{l} M_i$

Our implementation is based on the MSP strategy and code used by Wang and Zhang in [9].

3 Numerical Experiments

Several numerical experiments are conducted to examine the proposed new strategy. The computations are executed on a subsystem of an HP parallel computer with 32 processors at the University of Kentucky. Each processor runs at 1.5 GHz and has 2 GB local memory. The following tests are run with 4 processors.

3.1 Test Problems

The right hand sides of system (1) are literally computed from the known coefficient matrix A by assuming that the solution is a vector with all components equal to one. The initial guess is a zero vector. There are three kinds of test problems:

Convection-diffusion problems include 5 point central difference discretization of the 2D equation

$$-u_{xx} - u_{yy} - 10(\sin x \cos \pi y u_x - \cos \pi x \sin y u_y) = 0, \tag{10}$$

and 7 point central difference discretizations of the 3D equation

$$u_{xx} + u_{yy} + u_{zz} + 1000(p\, u_x + q\, u_y + r\, u_z) = 0. \tag{11}$$

Here the convection coefficient functions for the 3D case are presented as

$$p = x(x-1)(1-3y)(1-2z), q = y(y-1)(1-2x)(1-2z), r = z(z-1)(1-2x)(1-2y).$$

For 2D, we assume a unit square domain. For 3D, a unit cubic domain is assumed. Dirichlet boundary condition is assumed for both equations. Numerical simulation of 3D convection-diffusion problems tends to be computationally intensive and the size of the resulting linear system is often very large and sparse.

General sparse matrices are generated from coupled finite element discretization of Navier-Stokes equations used to numerically simulate incompressible fluid flows. Some of them are listed in Table 1.

Matrix Name	Order	Nonzeros	2-norm	Frobenius norm	Cond. number
FIDAP008	3096	90799	4.4e+07	4.2e+08	1.08e+11
FIDAP028	2603	76903	92	5.8e+02	4.30e+05
FIDAP031	3909	115299	23	88	1.01e+06
RAEFSKY1	3242	294276	3.7095	64.3408	3.16e+04
RAEFSKY2	3242	293551	3.7248	69.5413	1.08+e04

Table 1: Some FIDAP matrices used in the numerical tests.

3.2 Comparison of different strategies

We compare the performance of three strategies: enhanced MSP, original MSP with 1 step and original MSP with 2 steps. The test results listed in Table 2 are from solving the 2D problem. The test results listed in Table 3 are from solving the 3D problem. The original MSP with 1 step uses the sparsity pattern of A to obtain a sparse matrix $M_0 \approx A^{-1}$, then solves the $M_0 A = M_0 b$ using the

SAI preconditioner with the sparsity pattern M_0A. For the original MSP with 2 steps, it continues applying the SAI technique to obtain the next preconditioner $M_1 \approx (M_0A)^{-1}$ and solves the system $M_1M_0A = M1M_0b$. The purpose of this comparison is to show that the efficiency and facilitation conveyed by enhanced MSP. The first column is the number of unknowns of the matrices. The column "sparsity" means the sparsity ratio which denotes the sum of the number of nonzero entries of each multistep preconditioners M_i divided by the number of nonzero entries of the original matrix A. The column "iter." stands for the number of iterations of GMRES(50) to reduce the residual norm by 8 orders of magnitude.

	MSP with 1 step		MSP with 2 steps		EMSP with 1 step	
n	sparsity	iter.	sparsity	iter.	sparsity	iter.
100^2	2.58	139	10.61	73	8.03	95
200^2	2.59	249	10.70	187	8.11	208
250^2	2.59	354	10.72	192	8.13	238
300^2	2.59	400	10.74	230	8.14	270
350^2	2.59	427	10.74	258	8.15	352
400^2	2.60	536	10.74	283	8.16	387
450^2	2.60	625	10.74	386	8.16	403

Table 2: Comparison of MSP and enhanced MSP for solving the 2D problems.

From Table 2 we can find that the number of iterations of the enhanced MSP strategy is less than that of MSP with 1 preconditioning step but higher than that of MSP with 2 steps. However, the enhanced MSP strategy has smaller sparsity ratio compared with that of the original MSP. In Table 3 we also can get roughly the same results. With the matrix size increases, the importance of sparsity generally begins to overweigh that of the number of iterations since it means the matrix is getting denser and denser, and it requires more space to store the matrix and more time to complete one iteration. The enhanced MSP strategy hopes to maintain a proper point that a good tradeoff can be obtained by adjusting the sparsification threshold parameters.

	MSP with 1 step		MSP with 2 steps		EMSP with 1 step	
n	sparsity	iter.	sparsity	iter.	sparsity	iter.
10^3	4.27	19	18.36	13	13.03	12
20^3	4.42	26	20.70	17	14.67	14
30^3	4.47	31	15.71	17	8.13	16
40^3	4.50	49	21.83	25	16.84	19

Table 3: Comparison of MSP and enhanced MSP for solving 3D problems.

Convergence comparison for solving the FIDAP031 matrix The FIDAP031 matrix is solved to compare the MSP and enhanced MSP with different number of steps. We choose the values of the pre- and post-filtering parameters ("thresh" and "filter") both to be 0.03 for all test cases. In Table 4 we can find that although a larger number of steps leads to a better convergence rate, it obviously increases both the computational cost and memory cost for both strategies. However, the enhanced MSP strategy gets a faster convergence rate than the original MSP strategy by comparing the "sparsity" and number of iterations. We get 291 iterations with enhanced MSP at step 3 while we only get convergence at 342 iterations with the original MSP at step 5.

Original MSP					
steps	sparsity	iter.	setup time	solve time	total time
1	1.01	1439	1.7	2.5	4.2
2	1.43	573	4.4	1.8	6.2
3	1.62	392	8.0	1.4	9.4
4	1.72	398	14.2	1.7	15.9
5	1.78	342	19.5	1.8	21.3
Enhanced MSP					
steps	sparsity	iter.	setup time	solve time	total time
1	1.89	846	3.2	2.1	5.3
2	2.20	322	6.4	1.2	7.6
3	2.60	291	9.2	1.3	10.5
4	2.82	172	16.6	1.2	17.8

Table 4: Comparison of MSP and enhanced MSP for solving the FIDAP031 matrix.

General FIDAP and RAEFSKY matrices In Table 5, we use MSP(1) denotes the original MSP strategy with 1 step while MSP(2) denotes the original MSP strategy with 2 steps. EMSP means the enhanced MSP strategy and the number in its following parentheses indicates the step(s). The columns of "threshold" (column 3) and "filter" (column 4) are the two parameters used to sparsify the matrices [6]. The "setup time" is the CPU time in seconds for building the series of preconditioners and the "solve time" is the CPU time in seconds for solving the preconditioned sparse linear system. "total time" is the sum of "setup time" and "solve time".

In Table (5), we find that the enhanced MSP usually takes less CPU time than the original MSP method if their sparsity ratios are roughly same.

Matrix Name	Description	threshold	filter	sparsity	iter.	setup time	solve time	total time
RAEFSKY1	MSP(1)	0.05	0.05	0.16	207	1.2	1.5	2.7
	MSP(2)	0.02	0.02	0.44	50	10.2	0.2	10.4
	EMSP(1)	0.02	0.02	0.46	53	1.9	0.6	2.5
	EMSP(2)	0.02	0.02	0.51	47	3.7	0.5	4.2
RAEFSKY2	MSP(1)	0.05	0.02	0.32	535	2.9	3.3	6.2
	MSP(2)	0.02	0.01	0.93	169	31.9	1.1	33.0
	EMSP(1)	0.02	0.02	0.44	124	4.1	1.3	5.4
	EMSP2	0.02	0.02	0.68	89	9.5	1.1	10.6
FIDAP028	MSP(1)	0.0041	0.004	2.97	255	13.4	2.9	16.2
	MSP(1)	0.005	0.005	2.75	330	10.7	3.3	14.0
	EMSP(1)	0.004	0.004	3.25	183	13.7	2.1	15.8
	EMSP(1)	0.005	0.005	3.30	211	12.3	3.2	15.5
FIDAPM08	MSP(2)	0.01	0.01	3.28	729	48.3	3.4	51.7
	MSP(3)	0.01	0.01	5.12	291	142.2	2.2	144.5
	EMSP(2)	0.01	0.01	3.88	588	55.0	2.2	57.2
	EMSP(3)	0.01	0.01	6.50	220	116.5	5.1	121.6

Table 5: Comparison of MSP and enhanced MSP for solving general sparse matrices.

4 Concluding Remarks

The class of multistep successive sparse approximate inverse preconditioning strategies is studied and discussed in [9]. But our proposed new enhanced sparsity pattern selection strategy can make specific improvements on the efficiency and performance of the original MSP method. In fact the two parameters, e.g., "filter" and "threshold", which are used in both ParaSail and MSP are crucial to the performance of our strategy, since some necessary sparsification work must be done to keep the interim matrices not too dense. Moreover, in different situations within different construction steps, different parameters may render different results. In our experimental implementations, we manually change these parameters to meet with our own requirements. It is possible that, in the future, these parameters can be adaptively chosen.

Finally, we remark that we use polynomial preconditioning idea to improve the multistep successive preconditioning. Our experimental results show that the enhanced MSP strategy has some beneficially advantages over the original one. It is also possible to combine other preconditioning strategies into each step in a multistep successive preconditioning framework.

References

[1] L. Adams and E. Ong. Additive polynomial preconditioners for parallel computers. *Paral. Comput.*. 9:333–345 (1988).

[2] M. Benzi and M. Tuma. A comparative study of sparse approximate inverse preconditioners. *Appl. Numer. Math.*, 30(2-3):305–340 (1999).

[3] M. Bollhöfer and Y. Saad. ILUs and factorized approximate inverse are strongly related. part I: overview of results. *Report UMSI-2001-67, Minnesota Supercomputer Institute, University of Minnesota, Minneapolis, MN, 2001.*

[4] R. Bru, C. Corral, A. Martinez and J. Mas. Multisplitting preconditioners based on incomplete Choleski factorization. *SIAM J. Mat. Anal.*, 16(4):1210-1222 (1995).

[5] B. Carpentieri, I.S. Duff and L. Giraud. Sparse pattern selection strategies for robust Frobenius norm minimization preconditioners in electromagnetism equations. *Numer. Linear Algebra Appli.*, 7(7-8):667-685 (2000).

[6] E. Chow. Parallel implementation and performance characteristics of least squares sparse approximate inverse preconditioners. Technical Report UCRL-JC-138883, Lawrence Livermore National laboratory, Livermore, CA, 2000.

[7] E. Chow and Y. Saad. Approximate inverse preconditioners via sparse-sparse iterations. *SIAM J. Sci. Comput.*, 19(3):995–1023 (1998).

[8] M. J. Grote and T. Huckle. Parallel preconditioning with sparse approximate inverses. *SIAM J. Sci. Comput.*, 18(3):838–853 (1997).

[9] K. Wang and J. Zhang. MSP: a class of parallel multistep successive sparse approximate inverse preconditioning strategies. *SIAM J. Sci. Comput.* , 24(4):1141–1156 (2003).

[10] K. Wang and J. Zhang. A comparative study on dynamic and static sparsity patterns in parallel sparse approximate inverse. preconditioning *J. Math. Modeling Alg.*, to appear.

[11] H. A. van Vorst. A vectorizable variant of some ICCC methods. *SIAM J. Sci. Stat. Comput.*, 3:350–356 (1982).