Parallel Smoothers Using Sparse Approximate Inverse[*]

Wei-Pai Tang[†] W. L. Wan[‡]

December 7, 1998

Abstract

Sparse approximate inverses' usefulness in a parallel environment has motivated much interest in recent years. However, the superior capability of an approximate inverse in eliminating the local error has not yet been fully exploited in multi-grid algorithms. We propose a new class of sparse approximate inverse smoothers in this paper and present their analytic smoothing factors for constant coefficient PDEs. In particular, by adjusting the quality of the approximate inverse, the smoothing factor can be improved accordingly. For hard problems, this a useful feature. Our theoretical and numerical results have demonstrated the effectiveness of this new technique.

1 Introduction

In this paper, we propose a new class of smoothers derived from sparse approximate inverse preconditioners. It has a smoothing efficiency similar to Gauss-Seidel and it is independent of ordering. Moreover, for hard problems, we can improve the smoothing efficiency by adaptively adjusting the quality of the approximate inverse, for instance, by adding more nonzeros. Consequently, this new technique is more robust than the commonly used Gauss-Seidel smoothers. Our numerical testing verifies this statement.

We remark that Huckle [22] independently experimented with a sparse approximate inverse smoother for multi-grid by modifying the standard Gauss-Seidel iteration. Specifically, instead of using the exact inverse of the lower triangular matrix, he used a sparse approximate inverse of it computed by the techniques described in [19]. In our approach, we replace the Gauss-Seidel smoother entirely by a sparse approximate inverse smoother. The resulting multi-grid is efficient, and we have more flexibility in improving the smoothing quality for hard problems. Benson and his colleagues[3, 4, 5] also proposed to use an early version of sparse approximate inveses in multigrid applications. However, the potentials of the approximate inverse for a smoother was not fully exploited.

In Section 2, we describe the construction of the sparse approximate inverse smoother and analyze the computational complexity of constructing and applying the SAI smoothers. In Section 3, we describe how one can improve the SAI smoothers for different problems such as PDEs with anisotropic coefficients. In Section 4, we present a classical Fourier and

[*]This work was supported by the Natural Sciences and Engineering Research Council of Canada, by the Information Technology Research Centre, which is funded by the Province of Ontario, by Lawrence Livermore National Laboratory under Contract W07405-Eng-48, by the U.S. Department of Energy under the Accelerated Strategic Computing Initiative (ASCI), and by the Alfred P. Sloan Foundation as a Doctoral Dissertation Fellow.

[†]Department of Computer Science, University of Waterloo, Waterloo, Ontario, Canada N2L 3G1.

[‡]Scientific Computing and Computational Mathematics Program, Stanford University, Stanford, CA 94305-9025, USA.

local mode analysis on the smoothing property of SAI smoothers for constant coefficient PDEs. A numerical analysis based on the spectral decomposition is used to verify the theoretical results. Finally, in Section 5, we show the effectiveness of the SAI smoother for multi-grid by a variety of problems including anisotropic problems, discontinuous coefficient problems and unstructured grid problems.

2 SAI smoothers

Various techniques have been proposed for an effective sparse approximate inverse preconditioner [2, 6, 11, 14, 19, 23, 26]. However, the goal of constructing an effective smoother is very different from finding a good preconditioner. For a powerful preconditioner, the capability of removing both the high and low frequency errors is essential. In contrast, a good smoother may just damp the high frequency errors effectively. In this respect, much of the weakness [26] of the SAI preconditioners becomes the strength of the SAI smoothers. Our new proposal is to explore them to construct a powerful smoother.

Most sparse approximate inverse approaches seek a sparse matrix M so that the error of the residual $\mathcal{E} = MA - I$ is minimized in some measure. The sparsity constraint often limits the effectiveness of M as a preconditioner due to the locality of the sparse approximation. The requirement for a good smoother, however, can take advantage of the superiority in removing local error of SAI.

The construction of our SAI smoothers is based on a Frobenius norm approach first described by Benson [2]. A (left) SAI M is defined as a solution to the minimization problem: $\min_M \|MA - I\|_F^2$, subject to some constraint on the number and position of the nonzero entries of M. This minimization problem can be simplified to n much smaller least squares problems:

$$\min_{m_j} \|A_j^T m_j - e_j\|_2, \qquad j = 1, \ldots, n, \tag{1}$$

where m_j are some prescribed (or can be adaptively obtained) sparsity pattern for the given row j. Our modified approach simplifies (1) further by selecting a subset of nonzero rows of A_j^T.

In the following, we describe a systematic procedure of constructing submatrices of A for defining the m_j's. In contrast to many other SAI approaches, this modified Frobenius norm approach has direct control of the number and location of the nonzeros of the m_j's as well as the complexity of the least squares problems *a priori*. Our a priori approach significantly improved the effectiveness of a SAI smoother. It also improves the quality of the smoother over the early proposed local q-operator[5].

A sparse matrix A can be represented by a digraph $\mathcal{G} = (\mathcal{O}, \mathcal{E})$ [16]. Define $\mathcal{L}_k(o_i)$, the k-level neighbor set of node o_i, the nodes which are a distance $k+1$ or less from o_i. The 0-level neighbor set $\mathcal{L}_0(o_i)$ contains all the nodes which directly connect to node o_i. For PDE problems with a second order finite difference/finite discretization, $\mathcal{L}_0(o_i)$ is just the set of stencil points.

The study of the decay of the elements in an inverse [25] motivates the choice of $\mathcal{L}_k(o_i)$ for the sparsity pattern to the SAI at node o_i. The computation of these locations is also inexpensive.

The rectangular submatrix[1]

$$A_{k,l} \equiv A(\mathcal{L}_k(o_i), \mathcal{L}_l(o_i)) \tag{2}$$

[1]The Matlab notation is adopted for extracting a submatrix from a given matrix A

is defined as the (k,l)-level local matrix of node o_i. This matrix takes rows corresponding to nodes $\mathcal{L}_k(o_i)$ and columns corresponding to nodes $\mathcal{L}_l(o_i)$ from A.

We introduce the (k,l)-*level local least squares approximation* of an inverse as follows. For each node o_i, the least squares solution[2] m_{o_i} of

$$A_{k,l}^T m_{o_i} = e_{o_i} \tag{3}$$

is taken for the nonzero values at the corresponding location $\mathcal{L}_k(o_i)$ of the sparse approximation of the discrete Green's function of node o_i. More precisely, we inject each element of the least squares solution m_{o_i} into a zero vector of size of the matrix A at the corresponding locations in $\mathcal{L}_k(o_i)$ and use this sparse row to approximate the row o_i of the inverse A^{-1}.

It is clear that the sparsity pattern of the approximate inverse is determined by the locations of $\mathcal{L}_k(o_i)$ and the approximation range by the locations in $\mathcal{L}_l(o_i)$. A higher level k implies a denser approximation while a higher level l provides an approximation which is good for more neighbors. A properly chosen (k,l)-level is crucial to the success of SAI smoothers.

The relaxation method: $x^{n+1} = x^n + M(b - Ax^n)$, resulting from the SAI M constructed by the (k,l)-level local least squares approximation is called the (k,l)-level SAI smoothing.

We illustrate the (0,1)-level local least squares approximation method by a model constant coefficient second order elliptic PDE:

$$w_1 u_{xx} + w_2 u_{yy} + s u_x + t u_y \quad = \quad f(x,y), \qquad (x,y) \in \Omega, \qquad \text{where} \qquad u|_\Gamma = g(x,y)$$

on a rectangular domain Ω. Suppose we use a $m \times n$ regular grid. The resulting difference equation can be genearlly written as:

$$b u_{i-1,j} + d u_{i,j-1} + a u_{i,j} + c u_{i+1,j} + e u_{i,j+1} = h^2 f_{i,j}, \qquad 1 \le i \le m, \quad 1 \le j \le n,$$

where a, b, c, d and e are constants. The $(0,1)$-level local least squares problem for an interior node is:

$$\begin{pmatrix} a & & & & c \\ & a & & & d \\ & & a & & b \\ & & & a & e \\ b & e & c & d & a \\ c & & & & \\ d & c & & & \\ & d & & & \\ & b & d & & \\ & & b & & \\ & & e & b & \\ & & & e & \\ e & & & c & \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 1 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{pmatrix} \tag{4}$$

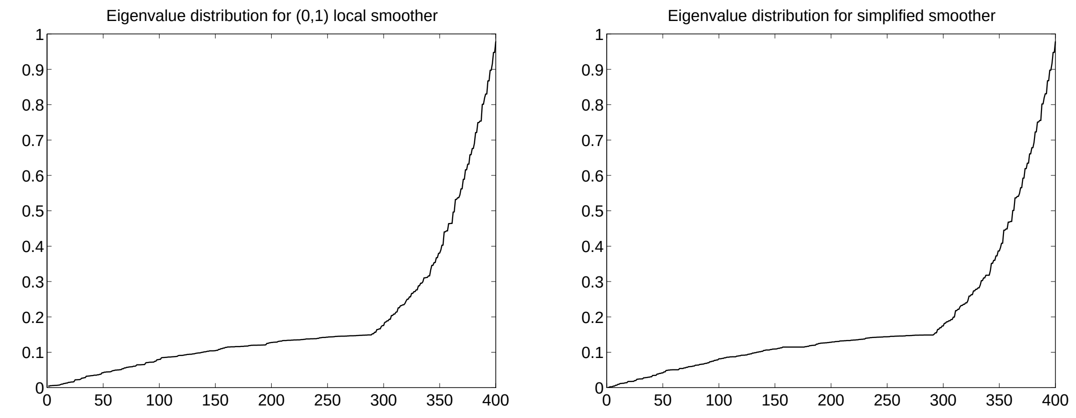
For any constant coefficient PDE on a regular mesh, we may further simplify the computations as follows. We observe that the local least squares problems, and hence the least squares solutions, corresponding to the interior points are identical (cf. (4)). However, they may be different near the boundary. We simply use the least squares solutions obtained

[2]In (3), e_{o_i} is a unit basis vector of size $|\mathcal{L}_l(o_i)|$, with one in the position corresponding to o_i and zeros in the rest of the locations in $\mathcal{L}_k(o_i)$. An example is given for (0,1)-level in (4).

from the interior points for the solutions at the boundary. Hence, we only need to solve one small least squares problem to obtain the entire approximate inverse. Our tests and analysis indicate that this simplified approach does not bring any noticeable penalty in performance.

As an illustration, consider solving the Poisson equation $\Delta u(x, y) = 1$, $(x, y) \in \Omega$. on a 20×20 retangular grid: In Fig. 1, we present the plots of the eigenvalues of the iterative matrices associated with the (0,1)-level SAI smoother and its simplified version. The eigenvalues in Fig 1(a) was calculated by Matlab using the *eig* function and those in Fig 1(b) by the analytic formula proved in Section 4. There is no any visible difference. Finally, we would like to mention a number of other techniques which can also be adopted to save the cost to compute the SAI[27].

Fig. 1. *Comparison of the eigenvalue distributions.*



(a) $(0, 1)$-level SAI smoother

(b) Simplified smoother

3 Adaptivity and anisotropic problems

In addition to simple construction and easy parallel implementation, SAI smoothers offer a capability to improve their quality by adjusting the number of level (i.e. the sparsity of the approximate inverse) used. Relaxation methods, for instance, Gauss-Seidel, do not have such feature which means the users have to look for other smoothers whenever the relaxation smoothers fail to give fast multi-grid convergence. For SAI smoothers, we may tune a parameter–the number of levels– to improve the resulting multi-grid convergence. Thus, SAI smoothers are more robust, flexible and can be applied to larger class of problems.

For anisotropic problems, they usually have many relatively small entries in both the matrix A and its inverse. We may significantly reduce the cost by discarding those small entries. For instace, a simple weighted distance technique may be used to prevent the fast growth of the size n_k corresponding to the sparsity pattern of the SAI. We compute the power of A^k and use the values as weights to determine the locations of the nonzeros in the sparse approximate inverse. When a problem is anisotropic, most of the weights are very small and therefore the corresponding neighbors can be discarded from the sparsity pattern of the SAI smoother. After the SAI is obtained, we may again discard the small entries in it. As shown in Section 5, Example 2, the quadratic growth in n_k is reduced to linear.

4 Smoothing factor analysis

We present an analysis of the smoothing factors for constant coefficient PDEs on two dimensional rectangular grids in this section. The special Poisson equation case has been analyzed by Tong [28]. For simplicity, we consider only the simplified (0,1)-level SAI smoothers. The smoothing factor analysis is based on the following theorem.

Theorem 4.1. *Given two tridiagonal matrices $B = tridiag(b, a', c)_{m\times m}$, and $C = tridiag(d, a'', e)_{n\times n}$, where b, c, d, e are non-positive. Then the eigenvalues of the matrix*

$$B \otimes I_{n\times n} + I_{m\times m} \otimes C \tag{5}$$

are

$$a - 2\sqrt{bc}\cos\theta_k - 2\sqrt{de}\cos\theta_j, \quad 1 \le k \le m; \quad 1 \le j \le n,$$

where $a = a' + a''$, $\theta_k = \frac{k\pi}{m+1}$, $\theta_j = \frac{j\pi}{n+1}$. The corresponding eigenvectors are

$$\left\{ \left(\frac{c}{b}\right)^{\frac{n-s}{2}} \left(\frac{e}{d}\right)^{\frac{n-t}{2}} \sin(s\theta_k)\sin(t\theta_j) \right\}, \quad 1 \le s \le m; \ 1 \le t \le n.$$

Unlike Fourier analysis for PDEs with periodic boundary condition, Theorem 4.1 shows that the sine functions $\{\sin(s\theta_k)\sin(t\theta_j)\}$ are not the eigenvectors for general constant coefficient PDEs with Dirichlet boundary condition; the eigenvectors may depend on the coefficients a, b, c, d and e. In the following analysis, we assume that A is symmetric. Thus, $b = c$ and $d = e$. Denote by M the simplified sparse approximate inverse of the matrix A. Then M can also be written in the form of (5).

Theorem 4.2. *The eigenvalues of the iterative matrix $I - MA$ are*

$$1 - ((x_5 + 2x_1\cos\theta_k + 2x_2\cos\theta_j)(a - 2b\cos\theta_k - 2d\cos\theta_j)),$$

where $1 \le k \le m$; $1 \le j \le n$.

Remark: This analysis can also be generalized to three dimensional problems.

4.1 Local mode analysis

We carry out the classical local mode analysis [7] to analyze the smoothing efficiency of the (0,1)-level SAI smoothers for Poisson equations. By a direct computation and Theorem 4.1, we have the following two lemmas.

Lemma 4.1. *Suppose A is the standard Laplacian operator. Then the least squares solution of (4) is:*

$$x_1 = x_2 = x_3 = x_4 = \frac{3}{61}, \qquad x_5 = \frac{17}{61}.$$

Lemma 4.2. *Let M be the (0,1)-level sparse approximate inverse of the standard Laplacian operator. Then the eigenvalues of M are:*

$$\frac{17}{61} + \frac{6}{61}\cos\theta_k + \frac{6}{61}\cos\theta_j.$$

Remark: Lemma 4.2 shows that $M > 0$, which implies that M is nonsingular.

As a consequence of Theorem 4.2, we have the following result.

Theorem 4.3. *The eigenvalues of the iterative matrix of the (0,1)-level smoother are:*

$$\lambda(\theta_k, \theta_j) = 1 - (\frac{17}{61} + \frac{6}{61}\cos\theta_k + \frac{6}{61}\cos\theta_j)(4 - 2\cos\theta_k - 2\cos\theta_j). \tag{6}$$

Now, we show the smoothing factor of our SAI smoother. Define the high frequencies in the standard way:

$$\Theta_H = \{(\theta_k, \theta_j) : \frac{m}{2} \le k \le m \ \text{ or } \ \frac{n}{2} \le j \le n\}.$$

and the smoothing factor as $\rho \equiv \max\{|\lambda(\theta_k, \theta_j)| : (\theta_k, \theta_j) \in \Theta_H\}$, where $\lambda(\theta_k, \theta_j)$ is given by Theorem 4.3. Then we have:

THEOREM 4.4.

$$\rho \le \frac{21}{61} \approx \frac{1}{3}.$$

The proof is presented in [27]. Thus, the SAI smoothers reduce the high frequencies by a factor of almost 1/3, and hence they are effective smoothers for smooth coefficient PDEs. As a comparison, similar calculations can be carried out for Gauss-Seidel with natural and red-black ordering. The smoothing factors are 1/2 and 1/4 respectively. In Section 5, we show that the SAI smoothers are also effective for tough PDEs such as anisotropic problems.

The spectral analysis of different smoothers can also be carried out. Out results again indicated the effectiveness of this new class of smoothers[27].

5 Numerical results

The first set of three PDE's illustrates that our SAI smoothers perform similarly as the GS smoothers, if the latter work. It shows that the additional easy parallel implementation feature of the SAI smoothers do not deteriorate convergence. Besides, for some problems, SAI smoothers may converge while the Gauss-Seidel smoothers do not.

In all the examples, Dirichlet boundary conditions are used, and the right-hand side function, $f(x) = 1$. In the multi-grid procedure, a V-cycle is used with two pre-smoothing and two post-smoothing. Linear interpolation is used for structured grid problems and a specialized energy-minimizing interpolation [29, 30] is used for unstructured grid problems. The number of multi-grid levels is such that the coarsest grid is 3×3 for structured square grid problems, and a total of four levels for unstructured grid problems. The iteration was terminated when the relative residual norm was less than 10^{-8}. We are using zeros as the initial guess in all cases. Actually, we may report better numbers of iterations if a random initial guess is used. The results are summarized in table form where the number of V-cycles and the average convergence rate of the last 10 iterations are shown.

The first variable coefficient problem in this set is:

$$((1 + x^2)u_x)_x + u_{yy} + \tan^2 y u_y = -100x^2.$$

The second helical spring problem [13] is:

$$u_{xx} + u_{yy} + \frac{3}{5 - y} u_x - 2G\lambda = 0,$$

where G and λ are some constants. The discontinuous coefficient PDE is:

$$(a(x, y)u_x)_x + (b(x, y)u_y)_y + u_x + u_y = \sin(\pi xy),$$

where the coefficients $a(x, y)$ and $b(x, y)$ are defined as:

$$a(x, y) = b(x, y) = \begin{cases} 10^{-3} & (x, y) \in [0, 0.5] \times [0.5, 1] \\ 10^3 & (x, y) \in [0.5, 1] \times [0, 0.5] \\ 1 & \text{otherwise.} \end{cases}$$

Table 1 shows the convergence results. In particular, we observe that the simplified (0,1)-level SAI performs essentially the same as its original version, while its setup cost only involves one least squares solve whose corresponding node is taken at the center of the domain. For the discontinuous coefficient problem, the two Gauss-Seidel smoothers diverge

Problems	Iteration				Conv. Rate			
	GS	GS(rb)	SAI	SSAI	GS	GS(rb)	SAI	SSAI
Poisson	9	7	9	9	0.11	0.07	0.11	0.11
Variable coeff.	13	10	12	17	0.22	0.15	0.19	0.35
Helical spring	12	9	12	12	0.20	0.12	0.19	0.20
Discont. coeff.	∞	∞	22	∞	∞	∞	0.40	∞

Table 1

Smooth and rough coefficient PDEs on a 33×33 *mesh. GS: Gauss-Seidel with natural ordering, GS(rb) Gauss-Seidel with red-black ordering, SAI: (0,1)-level SAI smoother, SSAI: simplified (0,1)-level SAI smoother.* ∞ *indicates divergence.*

while our (0,1)-level SAI smoother converges. In additional to the gain of parallel efficiency, our SAI smoothers have the capability of improving themselves by adjusting a parameter. When the relaxation smoothers do not work, we can do nothing to make them work. For the SAI smoothers, we may increase the number of levels and selectively adapt the sparsity pattern to improve their quality for hard problems.

In the second group of testing problems, we consider two model anisotropic coefficient PDEs.

Problem 1: The single direction anisotropic problem:

$$100u_{xx} + u_{yy} = 1, \qquad (x,y) \in \Omega, \qquad \text{where} \qquad u|_\Gamma = 0.$$

Problem 2: It has a more sophisticated anisotropy structure in both the x and y directions:

$$a(x,y)u_{xx} + b(x,y)u_{yy} = 1,$$

where the coefficients $a(x,y)$ and $b(x,y)$ are defined as:

$$a(x,y) = \begin{cases} 100 & (x,y) \in [0,0.5] \times [0,0.5] \text{ or } [0.5,1] \times [0.5,1] \\ 1 & \text{otherwise.} \end{cases}$$

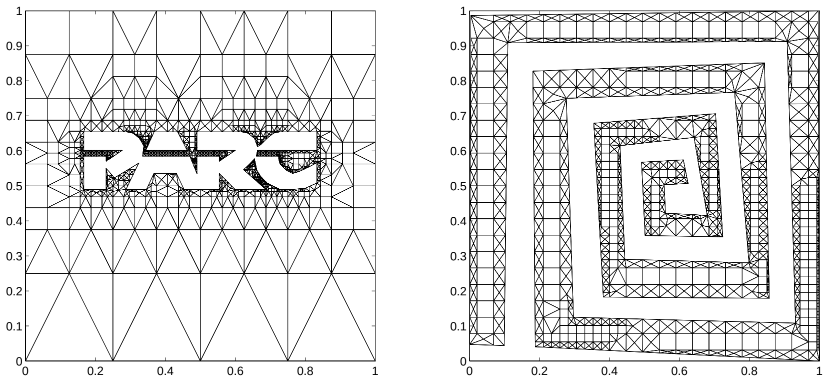
$$b(x,y) = \begin{cases} 100 & (x,y) \in [0,0.5] \times [0.5,1] \text{ or } [0.5,1] \times [0,0.5] \\ 1 & \text{otherwise.} \end{cases}$$

The results are shown in Table 2 and 3. The first three rows show the convergence results for problem 1 on 64×64, 128×128 and 256×256 grids, respectively. Similarly, row 4 to 6 show the results for problem 2. The last two rows show the results for problem 1 on two unstructured grids shown in Fig 2. As it is well-known, multi-grid with Gauss-Seidel smoother is not very effective for this kind of problems. Similar results are also observed for Gauss-Seidel with red-black ordering, and hence they are omitted. For problem 1 on regular grids, a standard technique[3] to improve the multi-grid convergence is to use line (block) relaxation methods. As indicated in the table, the performance of a block Jacobi smoother (BJ) is deteriorated when grid gets larger. Moreover, BJ smoother for problem 1 does not work for problem 2 due to two different orientations of the anisotropy. We

[3]Another technique is to use *semi-coarsening*. Since this approach uses a different coarsening rather than an improved smoother, we do not compare this method with our SAI smoother.

substitute the two line relaxation smoothings in the x direction by one line relaxation in the x direction and one line relaxation in the y direction. The results are still shown under the column BJ. For unstructured grid problems, blocks defined along the direction of the anisotropy are hard to determine, if possible. Thus we do not test the BJ smoother in this case. In any case, either the BJ smoother is not applicable, or it is slow.

Fig. 2. *The unstructured grids for Example 2.*



(a) parc (b) spiral

The previous (0,1)-level SAI smoother is not very effective in this case. We improve the performance by using higher level SAI smoothers (Section 2). We adopt the drop tolerance strategy described in Section 3 to maintain low computational complexity. Specifically, we drop entries in the matrix A whose absolute value is smaller than 2 before the sparse approximate inverse is computed. Then entries in the approximate inverse smaller than ϵ are also dropped. $SAI(k,\epsilon)$ denotes the $(k,k+1)$-level SAI smoother with ϵ=0.0008 for structured grid problems and ϵ=0.0004 for unstructured grids. Table 2 show that the higher

Grids	Iteration and Conv. Rate							
	GS		BJ		SAI(3,ϵ)		SAI(4,ϵ)	
problem 1 (64×64)	> 100	0.91	31	0.62	33	0.61	24	0.50
problem 1 (128×128)	> 100	0.92	> 100	0.88	37	0.64	27	0.53
problem 1 (256×256)	> 100	0.92	> 100	0.96	39	0.65	29	0.55
problem 2 (64×64)	> 100	0.89	29	0.57	28	0.55	22	0.47
problem 2 (128×128)	> 100	0.91	67	0.76	39	0.66	32	0.60
problem 2 (256×256)	> 100	0.92	> 100	0.92	48	0.70	40	0.66
parc	92	0.86	–	–	27	0.55	22	0.47
spiral	74	0.79	–	–	18	0.37	14	0.27

Table 2

Convergence results for anisotropic problems on different grids. GS: Gauss-Seidel with natural ordering, BJ: Block Jacobi. See text for the definitions of SAI(3,ϵ) and SAI(4,ϵ). ϵ= 0.0008 for the regular grid problems and 0.0004 for the unstructured grid problems.

level SAI smoothers outperform the Gauss-Seidel and the block Jacobi smoothers. More importantly, in contrast to line relaxation smoothers, the construction of the SAI smoothers does not require any information about the lines of different isotropy. Hence the exact same construction procedure can be applied to both problem 1 and 2. The higher level SAI allows us to capture the anisotropy easily by adjusting only one number–the level of fill-in, with no need to track geometrically the anisotropic directions which are often hard to determine. Furthermore, it does not matter whether it is a regular grid or irregular grid problem. For the two unstructured grid problems, the SAI smoothers converge much faster than the Gauss-Seidel smoother. Table 3 shows the costs of different smoothers relative to Gauss-

Grids	GS		BJ		SAI(3,ϵ)		SAI(4,ϵ)	
problem 1 (64×64)	1	> 100	2	62	1.75	58	1.79	43
problem 1 (128×128)	1	> 100	2	> 100	1.78	66	1.80	48
problem 1 (256×256)	1	> 100	2	> 100	1.79	70	1.80	52
problem 2 (64×64)	1	> 100	2	58	1.79	50	1.95	43
problem 2 (128×128)	1	> 100	2	> 100	1.80	70	1.88	60
problem 2 (256×256)	1	> 100	2	> 100	1.80	86	1.84	74
parc	1	92	–	–	2.62	73	3.06	67
spiral	1	74	–	–	2.09	38	2.32	32

Table 3

Cost of applying different smoothers relative to Gauss-Seidel. Under each smoother, the first column shows the cost per iteration relative to one GS iteration, and the second column shows the total cost=cost per iteration $\times$ total number of iterations.

Seidel. Under each smoother, the first column shows the cost per iteration of the smoother relative to one Gauss-Seidel iteration; the second column shows the total cost=cost per iteration $\times$ total number of iterations. The cost of each smoother is estimated as follows. The cost of one Gauss-Seidel iteration is estimated by the number of nonzeros in the matrix A. Thus, the relative cost of the SAI smoothers is estimated by the ratio of the number of nonzeros in the sparse approximate inverse to the number of nonzeros in the matrix A. The cost of inverting the diagonal block is about $8n$, assuming each block is a symmetric positive definite tridiagonal matrix [17]. Thus the cost of the block Jacobi smoother is about twice that of Gauss-Seidel.

By doubling the cost of applying a higher level SAI smoothers, we reduced more than half of the number of iterations. Moreover, the SAI smoothers also work well for unstructured grid problems where the standard multi-grid techniques for anisotropic problems on regular grids may not apply. Hence our SAI smoothers are more robust.

The setup cost of constructing the higher level SAI smoothers is kept low by the dropping strategy applied before and after computing the sparse approxiamte inverse. On the average, a size of 11×9 least sqaures problem is to be solved per row for the (3,4)-level SAI and a size of 13×11 for the (4,5)-level SAI. As a result, the quadratic growth in n_k is reduced to linear, which significantly reduce the overall computational cost.

References

[1] R. E. Bank, and C. C. Douglas, *Sharp Estimates for Multigrid Rates of Convergence with General Smoothing and Acceleration*, SIAM J. Num. Anal., 22:617–633, 1985.

[2] M. W. Benson, *Iterative solution of large scale linear systems*, M. Sc. Thesis, Lakehead University, Thunder Bay, Ontario, 1973.

[3] M. W. Benson, *Frequency Domain Behavor of a Set of Parallel Multigrid Smoothing Operators*, Intern. J. Computer Math., 36:77–88, 1990.

[4] M. W. Benson and R. N. Banerjee, *An Approximate Inverse Based Multigrid Approach to the Biharmonic Problem*, Intern. J. Computer Math., 40:201–210, 1991.

[5] M. W. Benson, *An Approximate Inverse Based Multigrid Approach to Thin Domain Problems*, Utilitas Mathematica, 45:39–51, 1994.

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

[7] A. Brandt, *Multi-Level Adaptive Solutions to Boundary-Value Problems*, Math. Comp., 31:333–390, 1977.

[8] R. Bridson, and W.-P. Tang, *Ordering, Anisotropy and Factored Sparse Approximate Inverses*, submitted to *SIAM J. Sci Comput.* .

[9] R. H. Chan, T. F. Chan and W. L. Wan, *Multigrid for Differential-Convolution Problems Arising from Image Processing*, CAM Report 97-20, Department of Mathematics, UCLA, 1997.

[10] T. Chan, W. -P. Tang and W. L. Wan, *Wavelet Sparse Approximate Inverse Preconditioners BIT,* 37, pp. 644–650, 1997.

[11] E. Chow and Y. Saad, *Approximate inverse techniques for block-partitioned matrices. SIAM J. Sci Comput.*, 18, pp. 1657–1675, 1997

[12] S. S. Clift, and W.-P. Tang, *Weighted graph based ordering techniques for preconditioned conjugate gradient methods*, BIT Vol. 35, No. 1, pp. 30-47, 1995 .

[13] L. Collatz, *The numerical treatment of differential equations*, Springer-Verlag, New York, 3rd ed., 1966.

[14] J. Cosgrove, J. Diaz, and A. Griewank, *Approximate inverse preconditionings for sparse linear systems*, Intern. J. Computer Math, 44 (1992), pp. 91–110.

[15] D. F. D'Azevedo, P. A. Forsyth and W.-P. Tang, *Towards a cost effective ILU preconditioner with high level fill*, BIT 32, pp. 442-463, 1992.

[16] A. George and J. Liu, *Computer solution of large sparse positive definite systems*, Prentice-Hall, Englewood Cliffs, NJ, 1981.

[17] G. H. Golub and C. F. Van Loan, *Matrix Computations*, Johns Hopkins, Baltimore and London, 1989.

[18] N. I. M. Gould and J. A. Scott, *On approximate-inverse preconditioners*, Tech. Rep. RAL-TR-95-026, June, 1995. Computing and Information System Dept., Atlas Center, Rutherford Appleton Lab., Osfordshire OX11 0QX, England.

[19] M. Grote and T. Huckle, *Parallel preconditioning with sparse approximate inverses.* SIAM J. Sci. Comput., 18, pp. 838-854, 1997.

[20] W. Hackbusch, *Multi-grid Methods and Applications.* Springer-Verlag, Berlin, 1985.

[21] T. Huckle and M. Grote, *A new approach to parallel preconditioning with sparse approximate inverses*, SCCM report, 1994.

[22] T. Huckle, *Sparse approximate inverses and multigrid methods*, Sixth SIAM Conference on Applied Linear Algebra, Snowbird, October29–November 1, 1997.

[23] L. Y. Kolotilina and A. Y. Yeremin, *Factorized Sparse Approximate Inverse Preconditionings I. Theory*, SIAM J. Matrix Anal. Appl., 14, 1993, pp. 45-58.

[24] J. Rice and R. Boisvert, *Solving Elliptic Problems Using ELLPACK*, Spring-Verlag, New York, 1985.

[25] W.-P. Tang, *Schwarz splitting and template operators*, PhD thesis, Stanford University, Computer Science Dept., Stanford, CA94305, 1987.

[26] W.-P. Tang, *Towards an effective approximate inverse preconditioner,* SIAM J. Sci. Comp., vol. 19, 1998.

[27] W.-P. Tang, and W. L. Wan , *Sparse Approximate Inverse Smoother for Multi-grid ,* Submitted to SIMAX, 1998.

[28] C. H. Tong, *Analysis of Some Approximate Inverse Preconditioners for Sparse Linear Systems*, unpublished manuscript, 1995.

[29] W. L. Wan, *An Energy-Minimizing Interpolation for Multigrid,* CAM Report 97-18, Department of Mathematics, UCLA, 1997.

[30] W. L. Wan, T. F. Chan and B. Smith, *An Energy-Minimizing Interpolation for Robust Multigrid*, CAM Report 98-6, Department of Mathematics, UCLA, 1998.