Report

Sparse approximate inverse smoothers for geometric and algebraic multigrid

Author(s):
Bröker, Oliver; Grote, Marcus J.

Publication Date:
2000

Permanent Link:

https://doi.org/10.3929/ethz-a-006654205 →

Sparse Approximate Inverse Smoothers For Geometric and Algebraic Multigrid

Oliver Bröker [a] Marcus J. Grote [b]

[a] *Departement Informatik, ETH Zürich, CH-8092 Zürich, Switzerland*
[b] *Departement Mathematik, ETH Zürich, CH-8092 Zürich, Switzerland*

Dedicated to Rüdiger Weiss

Abstract

Sparse approximate inverses are considered as smoothers for geometric and algebraic multigrid methods. They are based on the SPAI-Algorithm (Grote and Huckle, 1997), which constructs a sparse approximate inverse M of a matrix A, by minimizing $I - MA$ in the Frobenius norm. This leads to a new hierarchy of inherently parallel smoothers: SPAI-0, SPAI-1, and SPAI(ε). For geometric multigrid, the performance of SPAI-1 is usually comparable to that of Gauss-Seidel smoothing. In more difficult situations, where neither Gauss-Seidel nor the simpler SPAI-0 or SPAI-1 smoothers are adequate, further reduction of ε automatically improves the SPAI(ε) smoother where needed. When combined with an algebraic coarsening strategy (Ruge and Stüben, 1987), the resulting method yields a robust, parallel, and algebraic multigrid iteration, easily adjusted even by the non-expert. Numerical examples demonstrate the usefulness of SPAI smoothers, both in a sequential and a parallel environment.

Essential advantages of the SPAI-smoothers are: improved robustness, inherent parallelism, ordering independence, and possible local adaptivity.

Key words: approximate inverses, robust smoothing, algebraic multigrid, parallel multigrid
PACS: 02.60.Dc, 02.60.Lj

Email address: broeker@inf.ethz.ch (Oliver Bröker).

1 Introduction

Multigrid methods rely on the subtle interplay of smoothing and coarse grid correction. Only their careful combination yields an efficient multigrid solver for large linear systems, resulting from the discretization of partial differential equations [7,14,15,27]. Standard smoothers for multigrid usually consist of a few steps of a basic iterative method. Here we shall consider smoothers that are based on sparse approximate inverses. Hence, starting from the linear system

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

we let M denote a sparse approximation of A^{-1}. The corresponding basic iterative method is

$$x^{(k+1)} = x^{(k)} - M(Ax^{(k)} - b). \tag{2}$$

Since the approximate inverse M is known explicitly, each iteration step requires only one additional $M \times v$ matrix-vector multiply; therefore, it is easy to parallelize and cheap to evaluate, because M is sparse.

Recently, various algorithms have been proposed, all of which attempt to compute directly a sparse approximate inverse of A [5,9,17]. For a comparative study of various approximate inverse preconditioners we refer to Benzi and Tuma [6]. Approximate inverse techniques are also gaining in importance as smoothers for multigrid methods. First introduced by Benson and Frederickson [3,4], they were shown to be effective on various difficult elliptic problems on unstructured grids by Tang and Wan [25]. Advantages of sparse approximate inverse smoothers over classical smoothers, such as damped Jacobi, Gauss-Seidel, or ILU, are inherent parallelism, possible local adaptivity, and improved robustness.

Here we shall consider sparse approximate inverse (SPAI) smoothers based on the SPAI-Algorithm by Grote and Huckle [13]. The SPAI-Algorithm computes an approximate inverse M explicitly by minimizing $I - MA$ in the Frobenius norm. Both the computation of M and its application as a smoother are inherently parallel. Since an effective sparsity pattern of M is in general unknown a priori, the SPAI-Algorithm attempts to determine the most promising entries dynamically. This strategy has proved effective in generating preconditioners for many difficult and ill-conditioned problems [1,13,24]. Moreover, it provides the means for adjusting the smoother locally and automatically, if necessary. Nevertheless, by choosing an a priori sparsity pattern for M, the computational cost can be greatly reduced. Possible choices include powers of A or $A^\top A$, as suggested by Huckle [16] or Chow [10]. Hence we shall investigate the following hierarchy of sparse approximate inverse smoothers: SPAI-0, SPAI-1, and SPAI(ε). For SPAI-0 and SPAI-1 the sparsity pattern of M is fixed: M is

diagonal for SPAI-0, whereas for SPAI-1 the sparsity pattern of M is that of A. For SPAI(ε) the sparsity pattern of M is determined automatically by the SPAI-Algorithm ([13]); the parameter ε controls the accuracy and the amount of fill-in of M.

As structured geometric grids are difficult to use with complex geometries, application code designers often turn to very large unstructured grids. Yet the lack of a natural grid hierarchy prevents the use of standard geometric multigrid. In this context, algebraic multigrid (AMG) is often seen as the most promising method for solving large-scale problems. The original AMG algorithm, first introduced in the 1980's by Ruge and Stüben [22], uses the (simple) Gauss-Seidel iteration as a smoother, but determines the coarse "grid" space in a sophisticated way to improve robustness of the method. But if the iteration fails to converge, there is no automatic way to improve on the smoother. As an alternative, we investigate the usefulness of smoothers based on SParse Approximate Inverses (SPAI). Not only inherently parallel, their performance can also easily be adjusted, even by a non-expert. Thus we aim for a more general and inherently parallel algebraic multigrid method.

In Section 2 we briefly review the SPAI-Algorithm and show how sparse approximate inverses are used as smoothers within a multigrid iteration. A heuristic Green's function interpretation underpins their effectiveness as smoothers. Rigorous results on the smoothing property of approximate inverses were proved in [8]; they are summarized in Section 2.4. Next, we present in Section 3 a detailed description of the algebraic coarsening strategy used ([22]), together with key components of the algorithm for efficient implementation. Finally, in Section 4, we compare the performance of SPAI smoothing to that of Gauss-Seidel smoothing on various test problems, either within a geometric or an algebraic multigrid setting.

2 Sparse approximate inverse smoothing

2.1 Classical smoothers

Consider a sequence of nested grids, $\mathcal{T}_0 \subset \ldots \subset \mathcal{T}_{\ell-1} \subset \mathcal{T}_\ell$. On the finest mesh, $\mathcal{T}_\ell$, we wish to solve the $n \times n$ linear system

$$A_\ell \, x_\ell = b_\ell \tag{3}$$

by a multigrid method – for further details on multigrid see Hackbusch [14] and ([15], Section 10) or Wesseling [27]. A multigrid iteration results from the recursive application of a two-grid method. A two-grid method consists of ν_1

pre-smoothing steps on level ℓ, a coarse grid correction on level $\ell-1$, and ν_2 post-smoothing steps again on level ℓ. This leads to the iteration

$$e_\ell^{(m+1)} = [S_\ell^{\nu_2}(I - pA_{\ell-1}^{-1}rA_\ell)S_\ell^{\nu_1}]\, e_\ell^{(m)},$$

for the error $e_\ell^{(m)} = x_\ell^{(m)} - x_\ell$. Here p and r are prolongation and restriction operators, respectively, between $\mathcal{T}_\ell$ and $\mathcal{T}_{\ell-1}$, while S_ℓ denotes the smoother. If nested finite element spaces with Galerkin discretization are used, the Galerkin product representation holds:

$$A_{\ell-1} = rA_\ell p, \quad \ell \geq 1. \tag{4}$$

Otherwise, one can still use (4) to define the coarse-grid problem for given r and p.

We shall always use $x_\ell^{(0)} = 0$ as our initial guess. The multigrid (V-cycle) iteration proceeds until the relative residual drops below a prescribed tolerance,

$$\frac{\|b - A_\ell\, x_\ell^{(m)}\|}{\|b\|} < \text{tol}. \tag{5}$$

Then we calculate the average rate of convergence

$$q = \left(\frac{\|b - A_\ell\, x_\ell^{(m)}\|}{\|b\|}\right)^{1/m}. \tag{6}$$

The expected *multigrid convergence behavior* is achieved if the number of multigrid iterations, m, necessary to achieve a fixed tolerance, is essentially independent of the number of grid levels ℓ.

Typically, the smoother has the form

$$S_\ell = I - W_\ell^{-1}A_\ell, \tag{7}$$

where W_ℓ approximates A_ℓ and is cheap to invert — of course, W_ℓ^{-1} is never computed explicitly. Let $A = D + L + U$, with D the diagonal, L the lower triangular part, and U the upper triangular part of A. Then, damped Jacobi smoothing corresponds to

$$S_\omega = I - \omega D^{-1}A, \tag{8}$$

whereas Gauss-Seidel smoothing corresponds to

$$S_{G-S} = I - (D + L)^{-1}A. \tag{9}$$

In (8) the parameter ω is chosen to maximize the reduction of the high frequency components of the error. The optimal value, ω^*, is problem dependent

and usually unknown a priori. Although Gauss-Seidel typically leads to faster convergence, it is more difficult to implement in parallel, because each smoothing step in (9) requires the solution of a lower triangular system. If neither Jacobi nor Gauss-Seidel smoothing lead to satisfactory convergence, one can either resort to more sophisticated (matrix dependent) prolongation and restriction operators ([29]) or to more robust smoothers, based on incomplete LU (ILU) factorizations of A_ℓ ([28]). Unfortunately, ILU-smoothing is inherently sequential and therefore difficult to implement in parallel. It is also difficult to improve locally, say near the boundary or a singularity, without affecting the fill-in everywhere in the LU factors.

2.2 SPAI-smoothers

As an alternative to inverting W_ℓ in (7), we propose to compute explicitly a sparse approximate inverse M of A, and to use it for smoothing – we drop the grid index ℓ to simplify the notation. This yields the SPAI-smoother,

$$S_{SPAI} = I - MA, \tag{10}$$

where M is computed by minimizing $\|I - MA\|$ in the Frobenius norm for a given sparsity pattern. In contrast to W_ℓ^{-1} in (7), the matrix M in (10) is computed explicitly. Therefore, the application of S_{SPAI} requires only matrix-vector multiplications, which are easy to parallelize; it does not require the solution of any upper or lower triangular systems. Moreover, the Frobenius norm naturally leads to inherent parallelism because the rows $m_k^\top$ of M can be computed independently of one another. Indeed, since

$$\|I - MA\|_F^2 = \sum_{k=1}^{n} \|e_k^\top - m_k^\top A\|_2^2 = \sum_{k=1}^{n} \|A^\top m_k - e_k\|_2^2, \tag{11}$$

the solution of (11) separates into the n *independent least-squares problems* for the $m_k^\top$:

$$\min_{m_k} \|A^\top m_k - e_k\|_2, \qquad k = 1, \ldots, n. \tag{12}$$

Here e_k denotes the k-th unit vector. Because A and M are sparse, these least-squares problems are small.

Since an effective sparsity pattern of M is unknown a priori, the original SPAI-Algorithm in [13] begins with a diagonal pattern. It then augments progressively the sparsity pattern of M to further reduce each residual $r_k = e_k - A^\top m_k$. Each additional reduction of the 2-norm of r_k involves two steps. First, the algorithm identifies a set of potential new candidates, based on the sparsity of A and the current (sparse) residual r_k. Second, the algorithm selects the most

profitable entries, usually less than five entries, by computing for each candidate a cheap upper bound on the reduction in $\|r_k\|_2$. Once the new entries have been selected and added to m_k, the (small) least-squares problem (12) is solved again with the augmented set of indices. The algorithm proceeds until each row $m_k^\top$ of M satisfies

$$\|e_k^\top - m_k A\|_2 < \varepsilon \tag{13}$$

where ε is a tolerance set by the user; it controls the fill-in and the quality of the preconditioner M. A larger value of ε leads to a sparser and less expensive approximate inverse, but also to a less effective smoother with a higher number of multigrid cycles. A lower value of ε usually reduces the number of cycles, but the cost of computing $M = \mathrm{SPAI}\,(\varepsilon)$ may become prohibitive; moreover, a denser M results in a higher cost per smoothing step. The optimal value of ε minimizes the total time; it depends on the problem, the discretization, the desired accuracy, and the computer architecture. Further details about the original SPAI-Algorithm can be found in [13].

In addition to SPAI(ε), we shall also consider the following two greatly simplified SPAI-smoothers with fixed sparsity patterns: SPAI-0, where M is diagonal, and SPAI-1, where the sparsity pattern of M is that of A. Both solve the least-squares problem (12), and thus minimize $\|I - MA\|$ in the Frobenius norm for the sparsity pattern chosen a priori. This eliminates the search for an effective sparsity pattern of M, and thus greatly reduces the cost of computing the approximate inverse. The SPAI-1 smoother coincides with the SAI(0,1) smoother of Tang and Wan [25].

For SPAI-0, $M = \mathrm{diag}(m_{kk})$ is diagonal and can be calculated directly. It is simply given by

$$m_{kk} = \frac{a_{kk}}{\|a_k\|_2^2}, \qquad 1 \le k \le n, \tag{14}$$

where $a_k^\top$ is the k-th row of A – note that M is always well-defined if A is nonsingular. In contrast to damped Jacobi, SPAI-0 is *parameter-free*.

To summarize, we shall consider the following hierarchy of SPAI-smoothers, which all minimize $\|I - MA\|$ in the Frobenius norm for a certain sparsity pattern of M.

SPAI-0: $M = \mathrm{diag}(m_{kk})$ is diagonal and given by (14).
SPAI-1: The sparsity pattern of M is that of A.
SPAI(ε): The sparsity pattern of M is determined automatically via the SPAI-Algorithm [13]. Each row $m_k^\top$ of M satisfies (13) for a given ε.

Any of these approximate inverses leads to the smoothing step

$$x^{(k+1)} = x^{(k)} - M\,(Ax^{(k)} - b). \tag{15}$$

We have found that in many situations, SPAI-0 and SPAI-1 yield ample smoothing. However, the added flexibility in providing an automatic criterion for improving the smoother via the SPAI-Algorithm remains very useful. Indeed, both SPAI-0 and SPAI-1 can be used as initial guess for SPAI(ε), and thus be locally improved upon where needed by reducing ε. For matrices with inherent (small) block structure, typical from the discretization of systems of partial differential equations, the Block-SPAI-Algorithm ([2]) greatly reduces the cost of computing M.

2.3 Green's function interpretation

Why do approximate inverses yield effective smoothers for problems which come from partial differential equations? As the mesh parameter h tends to zero, the solution of the linear system,

$$A^h u^h = f^h, \tag{16}$$

tends to the solution of the underlying differential equation,

$$\mathcal{L}\,u(x) = f(x), \tag{17}$$

with appropriate boundary conditions. Here the matrix A^h corresponds to a discrete version of the differential operator $\mathcal{L}$. Let y_k^h denote the k-th row of $(A^h)^{-1}$; it solves the linear system

$$(A^h)^\top y_k^h = e_k^h, \tag{18}$$

with e_k^h the k-th unit vector. As $h \to 0$, y_k^h tends to the Green's function $G(x; x_k)$, which solves

$$\mathcal{L}^* G(x; x_k) = \delta(x - x_k). \tag{19}$$

Here $\mathcal{L}^*$ denotes the adjoint differential operator and $\delta(x - x_k)$ the "delta-function" centered about x_k. To exhibit the correspondence between y_k^h and $G(x; x_k)$, we recall that $\mathcal{L}^*$ is formally defined by the identity

$$(\mathcal{L}u, v) = (u, \mathcal{L}^* v), \tag{20}$$

for all u, v in appropriate function spaces. Equation (20) is the continuous counterpart to the relation

$$(A^h u^h, v^h) = (u^h, (A^h)^\top v^h). \tag{21}$$

From (17), (19) and (20) we conclude that

$$u(x_k) = (\delta(x - x_k), u(x)) = (\mathcal{L}^* G(x; x_k), u(x)) \\ = (G(x; x_k), \mathcal{L}u(x)) = (G(x; x_k), f(x)). \qquad (22)$$

Similarly, the combination of (16), (18), and (21) leads to the discrete counterpart of (22),

$$u_k^h = (e_k^h, u^h) = ((A^h)^\top y_k^h, u^h) = (y_k^h, A^h u^h) = (y_k^h, f^h). \qquad (23)$$

Comparison of (22) and (23) shows that y_k^h corresponds to to $G(x, x_k)$ as $h \to 0$.

The k-th row, $(m_k^h)^\top$, of the approximate inverse, M^h, solves (12), or equivalently

$$\min_{m_k^h} \|(A^h)^\top m_k^h - e_k^h\|_2. \qquad (24)$$

Hence m_k^h approximates the k-th column of $A^{-\top}$, that is y_k^h in (18), in the (discrete) 2-norm for a fixed sparsity pattern of m_k^h. The nonzero entries of m_k^h usually lie in a neighborhood of x_k: they correspond to mesh points x_j close to x_k. Therefore, after an appropriate scaling in inverse powers of h, we see that m_k^h approximates $G(x; x_k)$ locally in the (continuous) $L^2(\Omega)$-norm. For (partial) differential operators, $G(x; x_k)$ typically is singular at x_k and decays smoothly, but not necessarily rapidly, with increasing distance $|x - x_k|$. Clearly the slower the decay, the denser M^h must be to approximate well A_h^{-1}; this deficiency of sparse approximate inverse preconditioners was also pointed out by Tang [24]. At the same time, however, it suggests that sparse approximate inverses, obtained by the minimization of $\|I - MA\|$ in the Frobenius norm, naturally yield smoothers for multigrid. Indeed to be effective, a preconditioner must approximate $(A^h)^{-1}$ uniformly over the entire spectrum of $\mathcal{L}$. In contrast, an effective smoother only needs to capture the high-frequency behavior of $(A^h)^{-1}$. Yet this high-frequency behavior corresponds to the singular, local behavior of $G(x; x_k)$, precisely that which is approximated by m_k^h.

To illustrate this fact, we consider the standard five-point stencil of the discrete Laplacian on a 15×15 grid. In Figure 1 on the following page we compare A^{-1} with the Gauss-Seidel approximate inverse, $(L + D)^{-1}$, and two explicit approximate inverses, SPAI-1 and SPAI(0.2). We recall that Gauss-Seidel, a poor preconditioner for this problem, remains an excellent smoother, because it captures the high-frequency behavior of A^{-1}. Similarly, SPAI-1 and SPAI(ε) yield local operators with, as we shall see, good smoothing property. Despite the resemblance between the Gauss-Seidel and the SPAI approximate inverses, we note the one-sidedness of the former, in contrast to the symmetry of the latter.

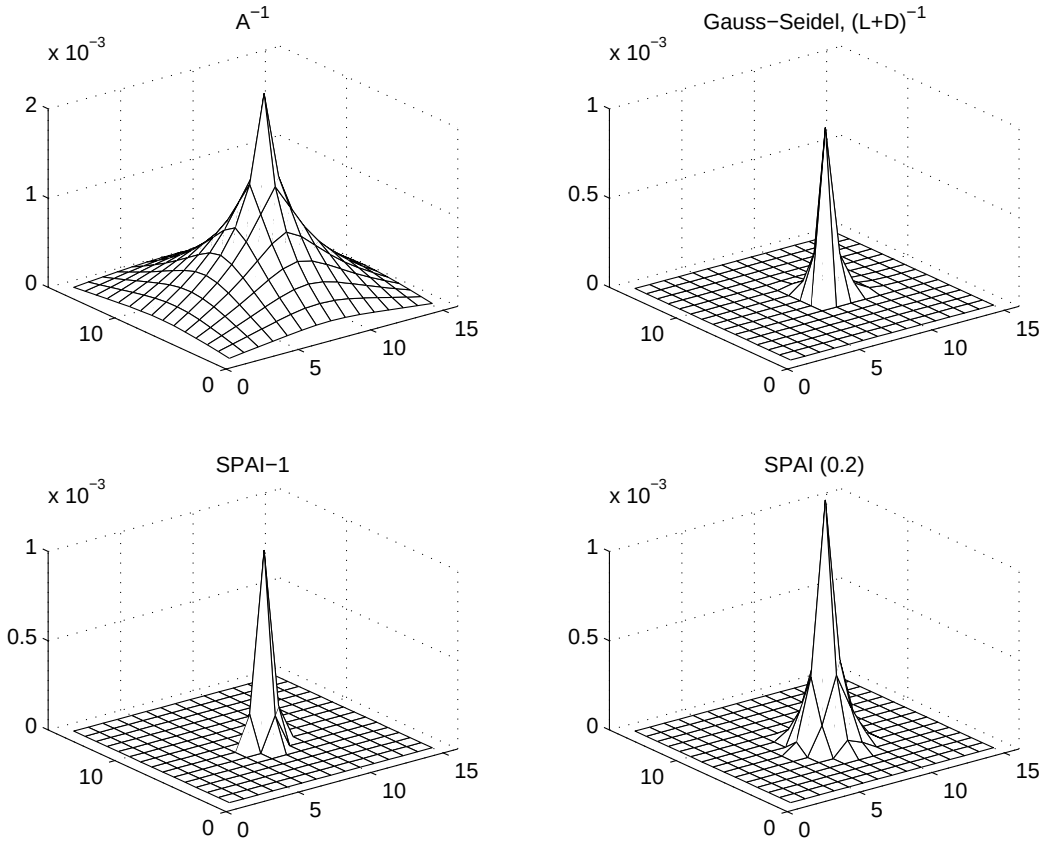


Fig. 1. Row 112 of the following operators: A^{-1} (top left), the Gauss-Seidel inverse $(L+D)^{-1}$ (top right), M computed with SPAI-1 (bottom left), and M computed with SPAI(0.2) (bottom right).

2.4 Theoretical Properties

In contrast to the heuristic interpretation of the previous section, we shall now summarize some rigorous results ([8]) on the smoothing property of the simplest smoother: SPAI-0.

Multigrid convergence theory rests on two fundamental conditions: the smoothing property ([15], Definition 10.6.3):

$$\|A_\ell S_\ell^\nu\|_2 \leq \eta(\nu)\|A_\ell\|_2, \qquad \text{for all} \quad 0 \leq \nu < \infty, \quad \ell \geq 1, \tag{25}$$

$$\eta(\nu) \text{ any function with } \lim_{\nu\to\infty} \eta(\nu) = 0,$$

and the approximation property ([15], Section 10.6.3). In general, the smoothing and approximation properties together imply convergence of the two-grid method and of the multigrid W-cycle, with a contraction number independent of the level number ℓ. Moreover, for symmetric positive definite problems, both conditions also imply multigrid V-cycle convergence independent

of ℓ – see Hackbusch ([15], Sect. 10.6) for details. The approximation property is independent of the smoother, S_ℓ; it depends only on the discretization $(A_\ell,\ A_{\ell-1})$, the prolongation operator p, and the restriction operator r. In [15] the approximation property is shown to hold for a large class of discrete elliptic boundary value problems. For symmetric positive definite problems the smoothing property usually holds for classical smoothers, such as damped Jacobi, (symmetric) Gauss-Seidel, and incomplete Cholesky.

In [8] the smoothing property (25) was shown to hold for the SPAI-0 smoother under reasonable assumptions on the matrix A. More precisely, for A symmetric and positive definite, the SPAI-0 smoother satisfies the smoothing property, either if A is weakly diagonally dominant, or if A has at most seven nonzero off-diagonal entries per row.

Furthermore, the two diagonal smoothers SPAI-0 and damped Jacobi, with optimal relaxation parameter ω^*, lead to identical smoothers for the discrete Laplacian with periodic boundary conditions in any space dimension [8]. In this special situation, the parameter-free SPAI-0 smoother automatically yields a scaling of $\mathrm{diag}(A)$, which minimizes the smoothing factor; in that sense it is optimal. In more general situations, however, both smoothers differ because of boundary conditions, even with constant coefficients on an equispaced mesh. Comparison of these two diagonal smoothers via numerical experiments showed that SPAI-0 is an attractive alternative to damped Jacobi [8]. Indeed, SPAI-0 is parameter-free and typically leads to slightly better convergence rates than damped Jacobi.

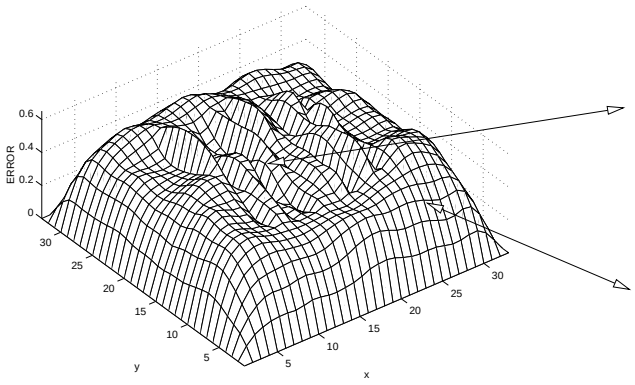
3 Algebraic Multigrid

Multigrid (MG) methods are sensitive to the subtle interplay between smoothing and coarse-grid correction. When a standard geometric multigrid method is applied to difficult problems, say with strong anisotropy, this interplay is disturbed because the error is no longer smoothed equally well in all directions. Although manual intervention and selection of coarse grids can sometimes overcome this difficulty, it remains cumbersome to apply in practice to unstructured grids and complex geometry. In contrast, an algebraic multigrid (AMG) approach compensates for the deficient smoothing by a sophisticated choice of the coarser grids and the interpolation operators, which is only based on the matrix A_ℓ. Many AMG variants exist, which differ in the coarsening strategy or the interpolation used – an introduction to various AMG methods can be found in ([26]).

Following Ruge and Stüben [22], we now describe the algebraic coarsening strategy and interpolation operators, which we shall combine with the SPAI

smoothers from Section 2.2 and use for our numerical experiments.

3.1 Coarsening strategy



Anisotropic stencil:

$$\frac{1}{h^2}\begin{bmatrix} & -\nu & \\ -1 & 2+2\nu & -1 \\ & -\nu & \end{bmatrix}$$

Isotropic stencil:

$$\frac{1}{h^2}\begin{bmatrix} & -1 & \\ -1 & 4 & -1 \\ & -1 & \end{bmatrix}$$

Fig. 2. The error after five Gauss-Seidel smoothing steps for the problem described in Section 4.2 on a with $\nu = 0.01$. The smooth error component is aligned with the anisotropy, which can be read from the stencils.

The fundamental principle underlying the coarsening strategy is based on the observation that interpolation should only be performed along smooth error components. For symmetric M-matrices, the error is smoothed well along large (negative) off-diagonal entries in the matrix A ([23]). Therefore, at each grid point p, we may identify among neighboring points q good candidates for interpolation, by comparing the magnitude of the corresponding entries a_{pq}. This leads to the following relations between the point p and its neighbors q in the connectivity graph of the matrix A:

Condition	Notation	Interpretation
$-a_{pq} \geq \tau \max_{a_{pr}<0} \lvert a_{pr} \rvert$ and $a_{pq} \neq 0$	$p \Leftarrow q$	p (strongly) depends on q *or* q (strongly) influences p
$-a_{pq} < \tau \max_{a_{pr}<0} \lvert a_{pr} \rvert$ and $a_{pq} \neq 0$	$p \leftarrow q$	p weakly depends on q *or* q weakly influences p

The parameter τ controls the threshold, which discriminates between strong and weak connections; typically $\tau = 0.25$. With this definition, all positive

off-diagonal entries are necessarily weak. The relations $p \Leftarrow q$ and $p \leftarrow q$ are symmetric only if A is symmetric.

Next, we define the set of *dependencies* of a point p as

$$D_p = \{q | p \Leftarrow q\},$$

the set of *influences* of a point p as

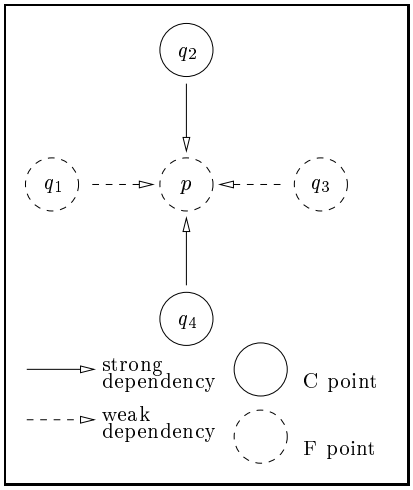
$$I_p = \{q | p \Rightarrow q\},$$

and the set of *weak dependencies* of a point p as

$$d_p = \{q | p \leftarrow q\}.$$

On every level, the coarsening strategy must divide P, the set of all points on that level, into two disjoint sets: C, the "coarse points", also present on the coarser level, and F, the "fine points", which are absent on the coarser level. The choice of C and F induces the *C/F–splitting* of $P = C \cup F$, with $C \cap F = \emptyset$.

Coarse grid correction heavily depends on accurate interpolation. Accurate interpolation is guaranteed if every F point is surrounded by sufficiently many strongly dependent C points. A typical configuration is shown in Figure 3.



Strong dependencies are indicated with solid arrows, while weak dependencies are represented by dashed arrows. C points are represented by solid circles, whereas F points are represented by dashed circles. Hence all the strong dependencies of point p (q_2 and q_4) are C points; therefore q_2 and q_4 are good candidates for interpolating p.

Fig. 3. Ideal coarsening configuration for interpolation

The coarsening algorithm attempts to determine a C/F–splitting, which maximizes the F–to–C dependency for all F points (Coarsening Goal 1), with a minimal set C (Coarsening Goal 2). It is important to strike a good balance between these two conflicting goals, as the overall computational effort depends not only on the convergence rate, but also on the amount of work per multigrid cycle. Clearly the optimal C/F-splitting minimizes total execution time. However, since the convergence rate is generally unpredictable, the

coarsening algorithm merely attempts to meet Coarsening Goals 1 and 2 in a heuristic fashion. In doing so, its complexity must not exceed $O(n \log n)$ to retain the overall complexity of the multigrid iteration.

3.2 Coarse grid selection: a greedy heuristic

To split P into C and F, every step of a greedy heuristic moves the most promising candidate from P into C, while forcing neighboring points into F. This procedure is repeated until all points are distributed. If every step requires at most $O(\log n)$ operations, and the complexity of all other computations does not exceed $O(n \log n)$, the desired overall complexity of $O(n \log n)$ is reached.

The greedy heuristic described in [23] is based on the following two principles, which correspond to Coarsening Goals 1 and 2:

(1) The most promising candidate, p, for becoming a C point, is that with the highest number of influences $|I_p|$. Then all influences of p are added to F. This choice supports Coarsening Goal 1 because all F points will eventually have at least one strong C dependency.
(2) To keep the number of C points low (Coarsening Goal 2), the algorithm should prefer C points near recently chosen F points; hence, these influences are given a higher priority.

Starting with all points as "undecided points", that is $U = P$, the algorithm proceeds by selecting from U the most promising C point with highest priority. The priority of any point p is defined by

$$\text{Priority}(p) = |I_p \cap U| + 2 \cdot |I_p \cap F| \tag{26}$$

Equation (26) reflects the preference in choosing the next C point for a point $p \in U$, which influences many previously selected F points. The key advantage of (26) is the possibility to update the priority locally and in $O(1)$ time, which results in the desired overall complexity of $O(n \log n)$. We now summarize the Coarse Grid Selection algorithm:

Algorithm 1 *Coarse Grid Selection*

procedure *C = CoarseGridSelection(P)*
  for all $p \in P$
    set Priority$(p) := |I_p|$
  end for all
  sort Priority
  $U := P, F := \emptyset, C := \emptyset$

```
    while U ≠ ∅
(1)   select p ∈ U with maximal Priority(p)
      C = C ∪ {p}
      for all q ∈ D_p (all dependencies of p)
(2)     Priority(q) := Priority(q) − 1
      end for all
      for all q ∈ I_p (all influences of p)
        F = F ∪ {q}
        for all r ∈ D_q (all dependencies of q)
(3)       Priority(r) := Priority(r) + 1
        end for all
      end for all
    end while
end procedure
```

To implement steps *(1)*, *(2)*, and *(3)* efficiently in $O(1)$ time, we maintain a list $\mathcal{Q}$ of all points sorted by priority, together with the list $\mathcal{I}$ of point indices of $\mathcal{Q}$. Moreover, a list of boundaries $\mathcal{B}$ of all priorities occurring in $\mathcal{Q}$ enables the immediate update of the sorted list $\mathcal{Q}$. Figure 4 shows a possible segment of the lists $\mathcal{Q}$, $\mathcal{B}$, and $\mathcal{I}$.

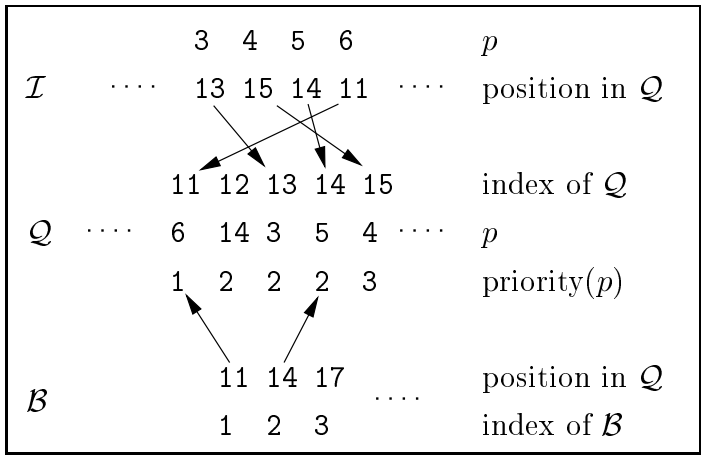


Fig. 4. The three lists $\mathcal{Q}$, $\mathcal{I}$, and $\mathcal{B}$ enable the efficient implementation of the coarsening algorithm.

During the set–up phase of the coarsening algorithm, $\mathcal{B}$ is computed and sorted by priority. Step *(1)* simply chooses the last element of $\mathcal{Q}$. Steps *(2)* and *(3)* are implemented by exchanging a point, whose priority must be either incremented or decremented, with its left– or rightmost neighbor in $\mathcal{Q}$ with that same priority; then its priority is adjusted, while $\mathcal{Q}$ remains sorted. Both $\mathcal{B}$ and $\mathcal{I}$ are updated accordingly. Following a suggestion of K. Stüben, we shall skip the second pass of the original coarsening algorithm in [22], which enforces even stronger F–to–C dependency, because of the high computational cost involved.

The grid function u_h, defined on the finer grid, is interpolated from the grid function u_H, defined on the coarser grid C, as follows:

$$u_h(p) = \begin{cases} u_H(p), & \text{if } p \in C \\ \sum_{q \in C} w_{pq} u_H(q), & \text{if } p \in F. \end{cases} \tag{27}$$

Hence, values at C points are simply transfered from the coarser level, whereas values at F points are interpolated from C neighbors. The four different dependencies possible between any F point and its neighbors are shown in Figure 5. For "standard interpolation" (see [23]), the choice of the weights, w_{pq}, for interpolating p, is based on the equation

$$a_{pp} e_p + \sum_{q \neq p} a_{pq}\, e_q = 0. \tag{28}$$

Indeed, if $Ae \simeq 0$, the smoothing effect is minimal, and the error e is declared "algebraically smooth" – see [23] for details. Clearly, we cannot interpolate p from surrounding F points, whereas weakly dependent C points are not included either, because of the rough nature of the error in that direction. Thus weak connections (q_1 and q_2 in Figure 5) are always ignored in the interpolation and the corresponding interpolation weights set to zero ($w_{p,q_1} = w_{p,q_2} = 0$). After cancellation of the weak dependencies, the neglected entries of the weakly dependent neighbors are added to the diagonal. Hence equation (28) becomes

$$\tilde{a}_{pp} e_p + \sum_{q \in D_p} a_{pq}\, e_q = 0, \quad \text{with} \quad \tilde{a}_{pp} = a_{pp} + \sum_{q \in d_p} a_{pq}. \tag{29}$$

Strong C dependencies, such as q_3 in Figure 5, cause no difficulty because the value of u_H is available at that coarse grid location. Division of (29) by $\tilde{a}_{pp}$ yields the weight

$$w_{pq} = -\frac{a_{pq}}{\tilde{a}_{pp}}. \tag{30}$$

However, strong F dependencies, such as q_4 in Figure 5, are not available for interpolation and must first be interpolated from C points, on which they strongly depend. To do so, we replace a_{qq} by $\tilde{a}_{qq}$ for every $q \in D_q \cap F$, with

$$\tilde{a}_{qq} = a_{qq} + \sum_{r \in d_p} a_{qr}. \tag{31}$$

For every point $r \in D_q$ this yields the weight

$$w_{pr} = \frac{a_{pq}}{\tilde{a}_{pp}} \cdot \frac{a_{qr}}{\tilde{a}_{qq}}. \tag{32}$$

If a a point q is both a direct and an indirect neighbor of p, so that both (30) and (32) apply, the two weights are calculated separately and then added to each other.

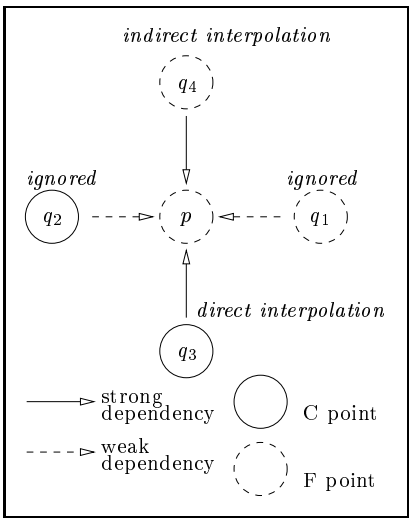


Fig. 5. The four different dependencies possible between p and its neighbors.

The algorithm described above determines the coarse grid levels only on the basis of A, and not on that of the approximate inverse M. In fact, the information contained in M can be used to determine coarse grid levels and interpolation weights, as suggested by Meurant [19,20].

3.4 Measuring computational costs and memory requirements

When comparing the performance of various smoothers, we cannot limit ourselves to comparing the number of multigrid iterations, but also need to estimate the additional amount of work due to the smoother. To do so, we calculate the total density ratio, ρ_M, of nonzero entries in M to those in A on all grid levels, $1 \leq i \leq \ell$, where smoothing is applied:

$$\rho_M = \frac{\sum_{i=1}^{\ell} \mathrm{nnz}(M_i)}{\sum_{i=1}^{\ell} \mathrm{nnz}(A_i)}. \tag{33}$$

The additional amount of work due to the smoother is proportional to ρ_M. While rapidly reducing the number of points from one level to the next, the

matrices A_i must also remain reasonably sparse, as measured by

$$\rho_A = \frac{\sum_{i=1}^{\ell} \mathrm{nnz}(A_i)}{\mathrm{nnz}(A_\ell)}. \tag{34}$$

For instance, as Galerkin coarse grid approximation enlarges the standard five–point stencil on the finest grid to nine–point stencils on subsequent levels, the resulting value of ρ_A for geometric multigrid is about 1.6. If semi–coarsening together with one-dimensional interpolation is used, ρ_A increases up to two.

All the results presented in the following section were computed with a MATLAB implementation. We shall evaluate the efficiency of the various approaches by comparing their respective values for ρ_M and ρ_A.

4 Numerical results

To illustrate the usefulness and versatility of SPAI smoothing, we shall now consider various standard test problems. In all cases, the differential equation considered is discretized on the finest level with standard finite differences on an equispaced mesh. For geometric multigrid, we use a regularly refined sequence of equispaced grids, with a single unknown remaining at the center of the domain. For algebraic multigrid, the coarser levels are obtained by the Coarse Grid Selection Algorithm described in Section 3.2. With $\tau = 0.25$ in the definition of strong dependency from section 3.1, the algorithm proceeds until the number of grid points drops below twenty. The coarse grid operators are obtained via the Galerkin product formula (4), with $r = p^\top$. For geometric multigrid, p correspond to standard linear interpolation, whereas for AMG p is obtained as described in Section 3.3. We use a multigrid V-cycle iteration, with two pre- and two post-smoothing steps ($\nu_1 = \nu_2 = 2$). The multigrid iteration proceeds until the relative residual satisfies the prescribed tolerance in (5), with $\mathrm{tol} = 10^{-8}$.

4.1 Rotating flow problem

We first consider the convection–diffusion problem,

$$-\nu \Delta u(x,y) + \begin{pmatrix} y - \frac{1}{2} \\ \frac{1}{2} - x \end{pmatrix} \cdot \nabla u(x,y) = 1 \tag{35}$$

in $(0,1)\times(0,1)$, with $u(x,y) = 0$ on the boundary. Here u represents any scalar quantity advected by the rotating flow field. For convection dominated flow,

$\nu << h$, the linear systems cease to be symmetric and positive definite, so that these problems lie outside of classical multigrid theory. We use centered second-order finite differences for the diffusion, but discretize the convection with first-order upwinding to ensure numerical stability.

Table 1
Geometric MG convergence rates for the rotating flow problem on a 128×128 grid, for different values of ν. The symbol † indicates that the multigrid iteration diverges.

	Smoother									
	Gauss-Seidel		SPAI-0		SPAI-1		SPAI(0.3)		SPAI(0.2)	
ν	q	ρ_M	q	ρ_M	q	ρ_M	q	ρ_M	q	ρ_M
10^0	0.05	—	0.09	(0.1)	0.04	(1.0)	0.07	(0.9)	0.03	(2.9)
10^{-3}	†		†		0.60	(1.0)	0.20	(1.4)	0.07	(3.6)
10^{-6}	†		†		†		†		0.13	(4.8)

Table 1 displays typical convergence rates q obtained with standard MG. All smoothers yield acceptable convergence rates in the diffusion dominated case, with $\nu = 10^0$. For $\nu = 10^{-3}$, however, the multigrid iteration diverges with Gauss–Seidel or SPAI-0 smoothing. In contrast, the SPAI-1 smoother still yields a convergent method. The use of SPAI(0.3) smoothing accelerates convergence even further, while ρ_M increases up to 1.4 only.

As we reduce the diffusion even further down to $\nu = 10^{-6}$, only the SPAI(0.2) smoother yields a convergent iteration. Although the resulting value of ρ_M is quite high, the construction of SPAI(0.2) remains parallel and fully automatic. We remark that symmetric Gauss-Seidel smoothing ([27]) leads to a convergent multigrid iteration, yet this approach does not generalize easily to unstructured grids.

Parallel results
Since the SPAI-1 smoother is inherently parallel, it is straightforward to apply within a parallel version of geometric MG. The data is distributed among processors via domain decomposition, which is well–known to work efficiently for a number of multigrid applications ([18]). The platform we shall use is the ETH–Beowulf cluster, which consists of 192 dual CPU Pentium III (500 MHz, 1 GB RAM, SuSe Linux 6.2) processors. All nodes are connected via a 100 MBit/s and 1 GB/s switched network, while communication is done with MPI.

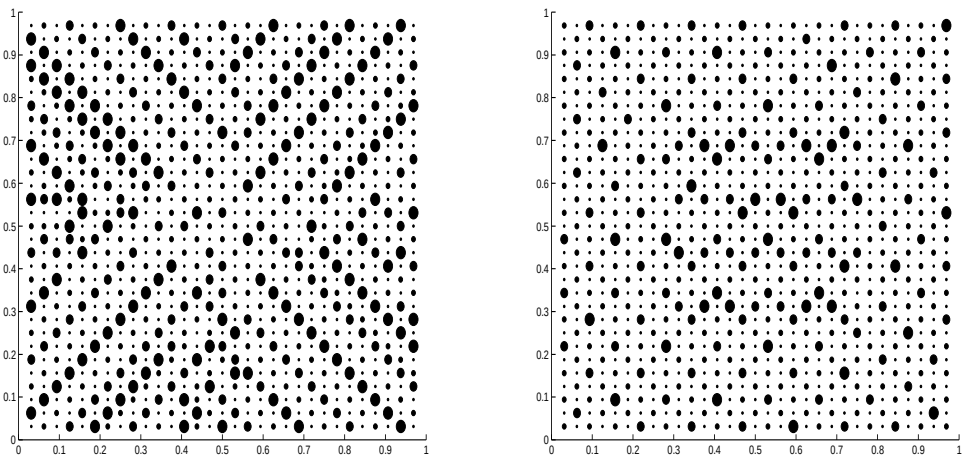
We now apply our parallel multigrid implementation to the rotating Flow Problem (35) with $\nu = 10^{-6}$. On 128 nodes, the total execution–time is 156 seconds on the 4096×4096 grid. The time includes the set–up for the construction of the SPAI-1 smoother, which requires the solution of about sixteen million small (25×9) and independent least–squares problems. As shown

in table 1, the use of a coarsest level, which consists only of a single mesh point, leads to a divergent multigrid iteration for $\nu = 10^{-6}$. By increasing the resolution of the coarsest level up to 32×32 mesh points, one obtains a convergent multigrid iteration.

Table 2
Scalability of parallel MG using SPAI-1. The problem size and the number of processors is increased by a factor of 4, while total time increases by 30% only.

Gridsize	512×512	× 4	1023×1023
Number of processors	4	⟶	16
Total time (sec)	20		26

To obtain good speed–up with a parallel MG code, it is important to perform *coarse grid agglomeration* (see [21]) because of the loss of efficiency on coarser grid levels. Although we have not implemented such an agglomeration strategy, our computations scale reasonably well as long as the problem size matches the size of the parallel architecture – see Table 2.



(a) *Rotating flow problem: algebraic coarsening is clearly aligned with the flow direction. Larger dots correspond to C points on coarser levels.*

(b) *Locally anisotropic diffusion: semi-coarsening is apparent in the center of the domain.*

Fig. 6. Examples of algebraic coarsening for the two model problems considered.

AMG results

None of these approaches, however, is entirely satisfactory for vanishing viscosity. To overcome the lack of robustness for small ν, we now apply the algebraic coarsening strategy described in Section 3. Figure 6(a) displays the coarse levels selected by the algorithm. In Table 3 both SPAI-0 and SPAI-1 yield convergence without any particular tuning. With $\varepsilon = 0.5$, the SPAI(ε)

Table 3
AMG convergence results for the rotating flow problem for varying ν on a 128×128 grid.

ν	10^0		10^{-3}		10^{-6}	
ρ_A	2.8		3.4		4.2	
Smoother	q	ρ_M	q	ρ_M	q	ρ_M
Gauss–Seidel	0.14	—	0.38	—	0.81	—
SPAI-0	0.26	(0.1)	0.35	(0.1)	0.36	(0.1)
SPAI-1	0.07	(1.0)	0.24	(1.0)	0.21	(1.0)
SPAI(0.5)	0.24	(0.4)	0.28	(0.4)	0.25	(0.8)

Table 4
AMG convergence rates for the rotating flow problem with $\nu = 10^{-6}$.

		Smoother							
		Gauss–Seidel		SPAI-0		SPAI-1		SPAI(0.5)	
Gridsize	ρ_A	q	ρ_M	q	ρ_M	q	ρ_M	q	ρ_M
64×64	4.0	0.68	—	0.32	(0.1)	0.18	(1.0)	0.19	(0.4)
128×128	4.2	0.81	—	0.36	(0.1)	0.21	(1.0)	0.27	(0.4)
256×256	4.3	0.96	—	0.38	(0.1)	0.24	(1.0)	0.34	(0.4)

smoother yields a compromise between the SPAI-0 and SPAI-1 smoothers: both the storage requirement and the convergence rate lie between those obtained with the fixed sparsity patterns of SPAI-0 and SPAI-1. Lower values of ε reduce the convergence rate even further. The poor convergence rates obtained with Gauss-Seidel could probably be improved, either by smoothing C points before F points ([23]) or by using symmetric Gauss–Seidel.

The results in Table 4 demonstrate the robustness of SPAI smoothing. Indeed, as $\nu \to 0$ all convergence rates obtained with the combined SPAI-AMG approach remain bounded.

4.2 Locally anisotropic diffusion

In this section we consider the locally anisotropic problem,

$$-\left(\nu(x,y)\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2}\right) = 1 \quad \text{in} \quad (0,1)\times(0,1), \tag{36}$$

with $u(x,y) = 0$ on the boundary. The diffusion coefficient $\nu(x,y) = 1$ everywhere except inside the square $[1/4, 3/4] \times [1/4, 3/4]$, where $\nu(x,y) \equiv \nu$ is

constant. In Table 5 we observe that geometric multigrid has difficulties for small values ν. Because of the unidirectional smoothing of the error, aligned with the strong anisotropy, standard (isotropic) interpolation fails.

Table 5
Locally anisotropic diffusion: geometric MG convergence rates q for varying ν on a 128×128 grid.

ν	10^0		10^{-3}		10^{-6}	
Smoother	q	ρ_M	q	ρ_M	q	ρ_M
Gauss–Seidel	0.05	—	0.97	—	0.98	—
SPAI-0	0.09	(0.1)	0.98	(0.1)	0.98	(0.1)
SPAI-1	0.04	(1.0)	0.97	(1.0)	0.98	(1.0)
SPAI(0.4)	0.12	(0.7)	0.95	(0.8)	0.97	(0.8)
SPAI(0.25)	0.04	(1.5)	0.75	(1.9)	0.87	(1.9)

AMG results

AMG overcomes these difficulties by performing automatic semi–coarsening and operator dependent interpolation only in the direction of strong couplings, which correspond to smooth error components. It is well–known (e.g. [23]) that AMG solves such problems with little difficulty. The results in Table 6 verify this fact for $\nu = 10^{-6}$, with acceptable densities ρ_A and ρ_M. Both densities could be lowered even further by dropping the smallest entries in the interpolation operators ([23]); we do not consider such truncated grid transfer operators here.

Table 6
AMG convergence results for locally anisotropic diffusion on a 128×128 grid. Note that q, ρ_A, and ρ_M remain bounded as $\nu \to 0$.

ν	10^0		10^{-3}		10^{-6}	
ρ_A	2.84		2.94		2.94	
Smoother	q	ρ_M	q	ρ_M	q	ρ_M
Gauss–Seidel	0.14	—	0.18	—	0.18	—
SPAI-0	0.26	(0.1)	0.32	(0.1)	0.32	(0.1)
SPAI-1	0.07	(1.0)	0.13	(1.0)	0.14	(1.0)
SPAI(0.5)	0.26	(0.1)	0.29	(0.2)	0.28	(0.2)

The convergence rates obtained with Gauss-Seidel and SPAI-1 are comparable and both below 0.2, while SPAI-0 results in slightly slower convergence. Overall the SPAI-1 smoother is the most efficient smoother for this problem. Although further reduction of ε results in even faster convergence, the approximate inverses become too dense and thus too expensive. Again, the results in

Tables 6 and 7 demonstrate robust multigrid behavior, either as $h \to 0$, or as $\nu \to 0$.

Table 7
AMG convergence rates q for locally anisotropic diffusion, with $\nu = 10^{-6}$. Note that q, ρ_A, and ρ_M remain bounded as $(h \to 0)$.

Gridsize	64×64		128×128		256×256	
ρ_A	2.89		2.94		2.95	
	q	ρ_M	q	ρ_M	q	ρ_M
Gauss-Seidel	0.12	—	0.18	—	0.22	—
SPAI-0	0.24	(0.1)	0.32	(0.1)	0.36	(0.1)
SPAI-1	0.08	(1.0)	0.14	(1.0)	0.18	(1.0)
SPAI(0.5)	0.23	(0.2)	0.28	(0.2)	0.32	(0.2)

5 Concluding remarks

Our results show that sparse approximate inverses, based on the minimization of the Frobenius norm, provide an attractive alternative to classical Jacobi or Gauss-Seidel smoothing. The simpler smoothers, SPAI-0 and SPAI-1, often provide ample smoothing, comparable to damped Jacobi or Gauss-Seidel. Nevertheless, situations such as convection dominated rotating flow, where SPAI-1 leads to a convergent multigrid iteration, unlike Gauss-Seidel, demonstrate the improved robustness. Our implementation of geometric multigrid combined with SPAI-1 smoothing enables the fast solution of very large convection-diffusion problems on massively parallel architectures. By incorporating the SPAI smoothers into AMG ([22]), we obtain a flexible, parallel, and algebraic multigrid method, easily adjusted to the underlying problem and computer architecture, even by the non-expert.

It is very interesting to incorporate information available from the approximate inverses into the coarsening strategy and grid transfer operators, as suggested in [19,20]. The expected benefit would include improved robustness and local adaptivity for these multigrid components as well. The authors are currently pursuing these issues and will report on them elsewhere in the near future.

Acknowledgment

We thank Klaus Stüben for useful comments and suggestions.

References

[1] S. T. Barnard, L. M. Bernardo, and H. D. Simon, *An MPI implementation of the SPAI preconditioner on the T3E*, Intl. J. High Perf. Comput. Appl. 13, 1999, pp. 107–128.

[2] S. T. Barnard and M. J. Grote, *A block version of the SPAI preconditioner*, in Proc. of the 9th SIAM conference on Parallel Processing for Scientific Computing, held in San Antonio, TX, March 1999.

[3] M. W. Benson and P. O. Frederickson, *Iterative solution of large sparse linear systems arising in certain multidimensional approximation problems*, Utilitas Mathematica 22, 1982, pp. 127–140.

[4] M. W. Benson, *Frequency domain behavior of a set of parallel multigrid smoothing operators*, Intern. J. Computer Math., 36 (1990), pp. 77–88.

[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), pp. 1135–1149.

[6] M. Benzi and M. Tuma, *A comparative study of sparse approximate inverse preconditioners*, Appl. Num. Math. 30, 1999, pp. 305–340.

[7] A. Brandt, *Multi-level adaptive solution to boundary-value problems*, Math. Comp. 31, pp. 333–390, 1977.

[8] O. Bröker, M. J. Grote, C. Mayer, A. Reusken, *Robust parallel smoothing for multigrid via sparse approximate inverses*, SIAM J. Sc. Comp., submitted.

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

[10] E. Chow, *A priori sparsity patterns for parallel sparse approximate inverse preconditioners*, SIAM J. Sc. Comput. 21, 2000, pp. 1804–1822.

[11] A. Cleary, et. al., *Robustness and scalability of algebraic multigrid*, SIAM J. Scientific Comp. 21, No. 5, 2000, pp. 1886–1908.

[12] A. Cleary, et. al., *Coarse-grid selection for parallel algebraic multigrid*, Proceedings of the Fifth International Symposium on Solving Irregularly Structured Problems in Parallel, Springer-Verlag Lecture Notes in Computer Science, New York, August 1998.

[13] M. J. Grote and T. Huckle, *Parallel preconditioning with sparse approximate inverses*, SIAM J. Sci. Comput., 18, 1997, pp. 838–853.

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

[15] W. Hackbusch, *Iterative Solution of Large Sparse Systems of Equations*, Springer-Verlag, New York, 1994.

[16] T. Huckle, *Approximate sparsity patterns for the inverse of a matrix and preconditioning*, Appl. Numer. Math. 30, pp. 291–303, 1999.

[17] L. Y. Kolotilina and A. Y. Yeremin, *Factorized sparse approximate inverse preconditionings: I. Theory*, SIAM J. Matrix Anal. Appl. 14, 1993, pp. 45–58.

[18] O. A. McBryan, P. O. Frederickson, J. Linden, A. Schüller, K. Solchenbach, K. Stüben, C. A. Thole and U. Trottenberg, *Multigrid methods on parallel computers a survey of recent developments*, Impact Comput. Sci. Eng., 3, 1991, pp. 1–75

[19] G. Meurant, *Numerical experiments with algebraic multilevel preconditioners*, 2000, submitted to ETNA

[20] G. Meurant, *A multilevel AINV preconditioner*, August 2000, submitted to Numerical Algorithms

[21] K. Oosterlee and U. Trottenberg, *Parallel adaptive multigrid — an elementary introduction*, Arbeitspapiere der GMD 1026, October 1996

[22] J. W. Ruge and K. Stüben, *Algebraic multigrid*, in Multigrid Methods, S. F. McCormick, ed., SIAM, Philadelphia, PA, 1987, pp. 73–130.

[23] K. Stüben, *Algebraic multigrid (AMG): An introduction with applications*, guest appendix in the book "*Multigrid Methods*" by U. Trottenberg, C.W. Oosterlee and A. Schüller, Academic Press, 2000. Also available as GMD Report 70, November 1999.

[24] W.-P. Tang, *Toward an effective approximate inverse preconditioner*, SIAM J. Matrix Anal. Appl. 20, 1999, pp. 970–986.

[25] W.-P. Tang and W. L. Wan, *Sparse approximate inverse smoother for multigrid*, SIAM J. Matrix Anal. Appl. 21, 2000, pp. 1236–1252.

[26] C. Wagner, *Introduction to algebraic multigrid*, Course Notes, University of Heidelberg, 1998/99 available at: `http://www.iwr.uni-heidelberg.de/~Christian.Wagner/`

[27] P. Wesseling, *An Introduction to Multigrid Methods*, John Wiley, Chichester, UK, 1992.

[28] G. Wittum, *On the robustness of ILU-smoothing*, SIAM J. Sci. Stat. Comput. 10, 1989, pp. 699–717.

[29] de Zeeuw, P. M., *Matrix prolongations and restrictions in a black-box multigrid solver*, J. Comp. and Appl. Math. 33, 1990, pp. 1–27.