Journal of Computational and Applied Mathematics 128 (2001) 281–309

JOURNAL OF
COMPUTATIONAL AND
APPLIED MATHEMATICS

www.elsevier.nl/locate/cam

A review of algebraic multigrid

K. Stüben

German National Research Center for Information Technology (GMD), Institute for Algorithms and Scientific Computing (SCAI), Schloss Birlinghoven, D-53754 St. Augustin, Germany

Received 3 September 1999; received in revised form 9 November 1999

Abstract

Since the early 1990s, there has been a strongly increasing demand for more efficient methods to solve large sparse, *unstructured* linear systems of equations. For practically relevant problem sizes, classical *one-level* methods had already reached their limits and new *hierarchical* algorithms had to be developed in order to allow an efficient solution of even larger problems. This paper gives a review of the first hierarchical and *purely matrix-based* approach, *algebraic multigrid* (AMG). AMG can directly be applied, for instance, to efficiently solve various types of elliptic partial differential equations discretized on unstructured meshes, both in 2D and 3D. Since AMG does not make use of any geometric information, it is a "plug-in" solver which can even be applied to problems without any geometric background, provided that the underlying matrix has certain properties. © 2001 Elsevier Science B.V. All rights reserved.

Keywords: Algebraic multigrid

1. Introduction

The efficient numerical solution of large systems of discretized elliptic partial differential equations (PDEs) requires hierarchical algorithms which ensure a rapid reduction of both short- and long-range error components. A break-through, and certainly one of the most important advances during the last three decades, was due to the multigrid principle. Any corresponding method operates on a hierarchy of grids, defined a priori by coarsening the given discretization grid in a geometrically natural way ("geometric" multigrid). Clearly, this is straightforward for logically regular grids. However, the definition of a natural hierarchy may become very complicated for highly complex, unstructured meshes, if possible at all.

The first attempt to automate the coarsening process took place in the early 1980s [10,12,13], at the time when the so-called *Galerkin-principle* and *operator-dependent interpolation* were combined in geometric multigrid to increase its robustness (aiming at the efficient solution of diffusion

E-mail address: stueben@gmd.de (K. Stüben).

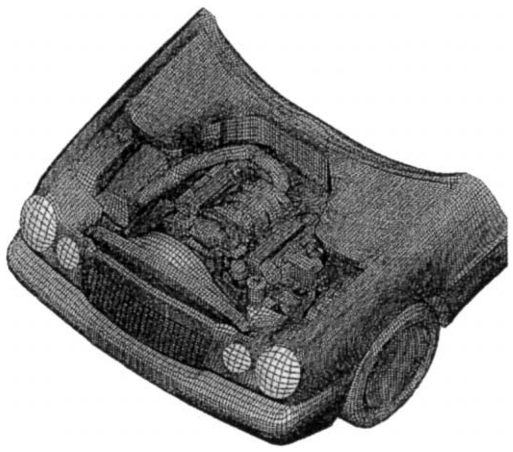

Fig. 1. Mesh for computing the underhood flow of a Mercedes-Benz E-Class.

problems with jumping coefficients [1,20]). This attempt was motivated by the observation that reasonable operator-dependent interpolation and the Galerkin operator can often be derived directly from the underlying matrices, without any reference to the grids. The result was a multigrid-like approach which did not merely allow an automatic coarsening process, but could be directly applied to (line sparse) algebraic equations of certain types, without any pre-defined hierarchy ("algebraic" multigrid,[1] AMG).

The first fairly general AMG program was described and investigated in [47,48,50]. Since this code was made publically available in the mid-1980s (AMG1R5), there had been no substantial further research and development in AMG for many years. However, since the early 1990s, and even more since the mid-1990s, there was a strong increase of interest in algebraically oriented multilevel methods. One reason for this was certainly the increasing geometrical complexity of applications which, technically, limited the immediate use of geometric multigrid. Another reason was the steadily increasing demand for efficient "plug-in" solvers. In particular, in commercial codes, this demand was driven by increasing problem sizes which made clear the limits of the classical one-level solvers still used in most packages.

For instance, CFD applications in the car industry involve very complicated flow regions. Flows through heating and cooling systems, complete vehicle underhood flows, or flows within passenger compartments are computed on a regular basis. Large complex meshes, normally unstructured, are used to model such situations. Requirements on the achievable accuracy are ever increasing, leading to finer and finer meshes. Locally refined grid patches are introduced to increase the accuracy with as few additional mesh points as possible. Fig. 1 shows an example.

In the recent past, several ways to realize concrete AMG algorithms have been investigated and there is still an ongoing rapid development of new AMG and AMG-like approaches and variants. Consequently, there is no unique and best approach yet. Whenever we talk about AMG in the

[1] We should actually use the term multi*level* rather than multi*grid*. It is just for historical reasons that we use the term multi*grid*.

context of concrete numerical results, we actually refer to the code RAMG05[2] (described in detail in [51]), which is a successor of the original code AMG1R5 mentioned above. However, RAMG05 is completely new and, in particular, incorporates more efficient and more robust interpolation and coarsening strategies.

This paper gives a survey of the classical AMG idea [48], certain improvements and extensions thereof, and various new approaches. The focus in Sections 2–6 is on fundamental ideas and aspects, targeting classes of problems for which AMG is best-developed, namely, *symmetric positive-definite* (s.p.d.) problems of the type as they arise, for instance, from the discretization of *scalar* elliptic PDEs of second order. We want to point out, however, that the potential range of applicability is much larger. In particular, AMG has successfully been applied to various nonsymmetric (e.g. convection–diffusion) and certain indefinite problems. Moreover, important progress has been achieved in the numerical treatment of *systems* of PDEs (mainly Navier–Stokes and structural mechanics applications). However, major research is still ongoing and much remains to be done to obtain an efficiency and robustness comparable to the case of scalar applications. In particular in Section 7, we will set pointers to the relevant literature where one can find further information or more recent AMG approaches. Although we try to cover the most important references, the list is certainly not complete in this rapidly developing field of research.

2. Algebraic versus geometric multigrid

Throughout this paper, we assume the reader to have some basic knowledge of geometric multigrid. In particular, he should be familar with the fundamental principles (smoothing and coarse-grid correction) and with the recursive definition of multigrid cycles. This is because, for simplicity, we limit our main considerations to just two levels. Accordingly, whenever we talk about the efficiency of a particular approach, we implicitly always assume the underlying two-level method to be recursively extended to a real multi-level method, involving only a small number of variables (20–40, say) on the coarsest level. Regarding more detailed information on geometric multigrid, we refer to [61] and the extensive list of references given therein.

A two-level AMG cycle to solve (sparse) s.p.d. systems of equations

$$A_h u^h = f^h \quad \text{or} \quad \sum_{j\in\Omega^h} a_{ij}^h u_j^h = f_i^h \quad (i\in\Omega^h) \tag{1}$$

is formally defined in the same way as a Galerkin-based geometric two-grid cycle. The only difference is that, in the context of AMG, Ω^h is just an *index set* while it corresponds to a *grid* in geometric multigrid. Accordingly, a coarser level, Ω^H, just corresponds to a (much smaller) index set.

If we know how to map H-vectors into h-vectors by some (full rank) "interpolation" operator I_H^h, the (s.p.d.) coarse-level operator A_H is defined via

$$A_H := I_h^H A_h I_H^h \quad \text{with } I_h^H = (I_H^h)^{\mathrm{T}}.$$

One two-level correction step then runs as usual, that is

$$u_{\text{new}}^h = u_{\text{old}}^h + I_H^h e^H, \tag{2}$$

[2] The development of RAMG05 has partly been funded by Computational Dynamics Ltd., London.

where the correction e^H is the exact solution of

$$A_H e^H = r^H \quad \text{or} \quad \sum_{j \in \Omega^H} a_{ij}^H e_j^H = r_i^H \quad (i \in \Omega^H)$$

with $r^H = I_h^H(r_{\text{old}}^h)$ and $r_{\text{old}}^h = f^h - A_h u_{\text{old}}^h$. (Note that we normally use the letter u for *solution* quantities and the letter e for *correction* or *error* quantities.) For the corresponding errors $e^h = u_\star^h - u^h$ ($u_\star^h$ denotes the exact solution of (1)), this means

$$e_{\text{new}}^h = K_{h,H} e_{\text{old}}^h \quad \text{with } K_{h,H} := I_h - I_H^h A_H^{-1} I_h^H A_h, \tag{3}$$

being the *coarse-grid correction operator* (I_h denotes the identity).

We finally recall that – given any relaxation operator, S_h, for smoothing – the convergence of Galerkin-based approaches can most easily be investigated w.r.t. the *energy norm*, $\|e^h\|_{A_h} = (A_h e^h, e^h)^{1/2}$. Assuming v relaxation steps to be performed for (pre-) smoothing, the following well-known variational principle holds (see, for instance [51]),

$$\|K_{h,H} S_h^v e^h\|_{A_h} = \min_{e^H} \|S_h^v e^h - I_H^h e^H\|_{A_h}. \tag{4}$$

As a trivial consequence, convergence of two-level cycles and, if recursively extended to any number of levels, the convergence of complete V-cycles is always ensured as soon as the relaxation method converges. This is true for any sequence of coarser levels and interpolation operators. More importantly, (4) indicates that the *speed* of convergence strongly depends on the efficient interplay between *relaxation* and *interpolation*. Based on (4), we want to outline the basic conceptual difference between geometric and algebraic multigrid.

2.1. Geometric multigrid

In geometric multigrid, fixed coarsening strategies are employed and interpolation is usually defined geometrically, typically by linear interpolation. Depending on the given problem, this necessarily imposes strong requirements on the smoothing properties of S_h (in order for the right-hand side in (4) to become small), namely, that the error after relaxation varies in a geometrically smooth way from the fine-level grid points to the neighboring coarse-level ones. In other words, the error after relaxation *has to be geometrically smooth*, relative to the coarse grid.

As an illustration, let us assume the coarser levels to be defined by standard geometric $h \to 2h$ coarsening in each spatial direction. It is well known that pointwise relaxation geometrically smooths the error in each direction only if the given problem is essentially isotropic. In case of anisotropic problems, however, smoothing is only "in the direction of strong couplings". In such cases, more complex smoothers, such as alternating line-relaxation or ILU-type schemes, are required in order to still achieve a good interplay between smoothing and interpolation and, thus, fast multigrid convergence.

While the construction of "robust smoothers" is not difficult in 2D model situations, for 3D applications on complex meshes their realization tends to become rather cumbersome. For instance, the robust 3D analog of alternating line relaxation is alternating *plane* relaxation (e.g., realized by 2D multigrid within each plane) which, in complex geometric situations, becomes very complicated to implement, if possible at all. ILU smoothers, on the other hand, loose much of their smoothing property in general 3D situations. The only way to loosen the requirements on the smoothing properties of the relaxation and still maintain an efficient interplay relaxation and interpolation is to use more sophisticated coarsening techniques. In geometric multigrid, steps in this direction have been done by, for example, employing more than one coarser grid on each multigrid level ("multiple semi-coarsening" [35,59,21,34]).

2.2. Algebraic multigrid

While geometric multigrid essentially relies on the availability of robust smoothers, AMG takes the opposite point of view. It assumes a simple relaxation process to be given (typically plain Gauss–Seidel relaxation) and then attempts to construct a suitable operator-dependent interpolation I_H^h (including the coarser level itself). According to (4), this construction has to be such that error of the form $S_h^\nu e^h$ is sufficiently well represented in the *range of interpolation*, $\mathcal{R}(I_H^h)$. The better this is satisfied, the faster the convergence can be. Note that it is *not* important here whether relaxation smooths the error in any geometric sense. What *is* important, though, is that the error after relaxation can be characterized algebraically to a degree which makes it possible to automatically construct coarser levels and define interpolations which are *locally* adapted to the properties of the given relaxation. This local adaptation is the main reason for AMG's flexibility in adjusting itself to the problem at hand and its robustness in solving large classes of problems *despite using very simple point-wise smoothers.*

3. The classical AMG approach

In classical AMG, we regard the coarse-level variables as a subset of the fine-level ones. That is, we assume the set of fine-level variables to be split into two disjoint subsets, $\Omega^h = C^h \cup F^h$, with C^h representing those variables which are to be contained in the coarse level (*C-variables*) and F^h being the complementary set (*F-variables*). Given such a C/F-splitting, we define $\Omega^H = C^h$ and consider (full rank) interpolations $e^h = I_H^h e^H$ of the form

$$e_i^h = (I_H^h e^H)_i = \begin{cases} e_i^H & \text{if } i \in C^h, \\ \sum_{k \in P_i^h} w_{ik}^h e_k^H & \text{if } i \in F^h, \end{cases} \tag{5}$$

where $P_i^h \subset C^h$ is called the set of *interpolatory variables.* (For reasons of sparsity of A_H, P_i^h should be a reasonably small subset of C-variables "near" i.) Clearly, $\mathcal{R}(I_H^h)$ strongly depends on both the concrete selection of the C-variables *and* the definition of the interpolation. In a given situation, one can easily imagine "bad" C/F-splittings which just do not allow any interpolation which is suitable in the sense that was outlined in the previous section. That is, the construction of concrete C/F-splittings and the definition of interpolation are closely related processes.

Concrete algorithms used in practice are largely heuristically motivated. In Section 3.1, we mainly summarize the basic ideas as described in [48] and some modifications introduced in [51]. In Section 3.2, we take a closer look at some theoretical and practical aspects in case that A_h contains only nonpositive off-diagonal entries (M-matrix). To simplify notation, we usually omit the index h in the following, for instance, we write S, A, K and e instead of S_h, A_h, $K_{h,H}$ and e^h. Moreover, instead of (5), we simply write

$$e_i = \sum_{k \in P_i} w_{ik} e_k \quad (i \in F). \tag{6}$$

3.1. The basic ideas

Classical AMG uses plain Gauss–Seidel relaxation for smoothing. From some heuristic arguments, one can see that the error e, obtained after a few relaxation steps, is characterized by the fact that the (scaled) residual is, on the average for each i, much smaller than the error itself, $|r_i| \ll a_{ii}|e_i|$. This implies that e_i can *locally* be well approximated by

$$e_i \approx -\left(\sum_{j\in N_i} a_{ij}e_j\right) \Bigg/ a_{ii} \quad (i\in\Omega), \tag{7}$$

where $N_i = \{j\in\Omega\colon j\neq i,\ a_{ij}\neq 0\}$ denotes the *neighborhood* of any $i\in\Omega$. Such an error is called *algebraically smooth*. According to the remarks at the end of Section 2, it is this kind of error which has to be well represented in $\mathscr{R}(I_H^h)$. That is, the general goal is to construct C/F-splittings and define sets of interpolatory variables $P_i\subset C$ $(i\in F)$ along with corresponding weights w_{ik} such that (6) yields a reasonable approximation for each algebraically smooth vector e.

Obviously, a very "accurate" interpolation in this sense is obtained by directly using (7), that is, by choosing $P_i = N_i$ and $w_{ik} = -a_{ik}/a_{ii}$. However, this would require selecting a C/F-splitting so that, for each $i\in F$, *all of its neighbors* are contained in C. Although any such selection can even be seen to yield a *direct* solver, this approach is of no real practical relevance since, in terms of computational work and memory requirement, the resulting method will generally be extremely inefficient if recursively extended to a hierarchy of levels [51].

In practice, we want to achieve rapid convergence with as small sets of interpolatory variables P_i as possible (in order to allow for a rapid coarsening and to obtain reasonably sparse Galerkin operators). Various approaches have been tested in practice which cannot be described in detail here. In the following, we just give an outline of some typical approaches.

3.1.1. Direct interpolation

We talk about *direct interpolation* if $P_i\subseteq N_i$. Such an interpolation can immediately be derived from (7) if we know how to approximate the "noninterpolatory part" (i.e. that part of the sum in (7) which refers to $j\in N_i\setminus P_i$) for an algebraically smooth error. This approximation is the most critical step in defining interpolation.

For M-matrices A, for instance, such an approximation can be obtained by observing that an algebraically smooth error varies slowly in the direction of *strong* (large) couplings. In particular, the more strong couplings of any variable i are contained in P_i, the better an algebraically smooth error satisfies

$$\frac{1}{\sum_{k\in P_i} a_{ik}} \sum_{k\in P_i} a_{ik}e_k \approx \frac{1}{\sum_{j\in N_i} a_{ij}} \sum_{j\in N_i} a_{ij}e_j \quad (i\in\Omega).$$

Inserting this into (7), we obtain an interpolation (6) with positive weights

$$w_{ik} = -\alpha_i a_{ik}/a_{ii} \quad \text{where } \alpha_i = \frac{\sum_{j\in N_i} a_{ij}}{\sum_{\ell\in P_i} a_{i\ell}}. \tag{8}$$

Practically, this means that we have to construct C/F-splittings so that each $i\in F$ has a sufficiently large number of *strongly* coupled C-neighbors which are then taken as the set of interpolatory variables P_i. (See Section 3.2 regarding some important additional aspects.)

In the case of (scalar) elliptic PDEs, the largest off-diagonal entries are usually negative. If there are also *positive* off-diagonal entries, a similar process as before can be applied as long as such entries are relatively small: Small positive couplings can simply be ignored by just considering them as *weak*. However, the situation becomes less obvious, if A contains *large* positive off-diagonal entries. In many such cases, an algebraically smooth error can be assumed to *oscillate* along such couplings (e.g. in case of *weakly diagonally dominant* s.p.d. matrices A [26,51]). This can be used to generalize the above approach by, for instance, a suitable separation of positive and negative couplings, leading to interpolation formulas containing both positive and negative weights. A corresponding approach has been proposed in [51] which can formally be applied to arbitrary s.p.d. matrices. However, the resulting interpolation is heuristically justified only as long as, for any i, those error components e_k which correspond to large positive couplings $a_{ik} > 0$, change slowly *among each other* (unless a_{ik} is very small in which case its influence can be ignored).

In practice, these simple algebraic approaches to construct an interpolation cover a large class of applications. However, there is no best way yet to automatically construct an interpolation which is good for *arbitrary* s.p.d. matrices, at least not by merely considering the size and sign of coefficients. For instance, in case of particular higher-order finite-element discretizations or discretizations by bilinear elements on quadrilateral meshes with large aspect ratios, the resulting matrices typically contain significant positive off-diagonal entries and are far from being weakly diagonally dominant. In such cases, the performance of AMG may be only suboptimal. If this happens, it often helps to exploit some information about the origin of these positive connections rather than to rely only on information directly contained in the matrix. For instance, one could try to structurally simplify the given matrix before applying AMG (see, e.g. [41]). Alternative approaches, defining the coarsening and interpolation are outlined in Section 7.

3.1.2. More complex interpolations

There are several ways to improve the direct interpolation of the previous section. To outline some possibilities, let us assume a C/F-splitting and, for each $i \in F$, a set of (strongly coupled) interpolatory variables $P_i \subseteq N_i \cap C$ to be given. Rather than to immediately approximate the noninterpolatory part of the ith equation (7) as done above, one may first (approximately) eliminate all strongly coupled $e_j\,(j \notin P_i)$ by means of the corresponding jth equation. The ideas outlined in the previous section can then analogously be applied to the resulting extended equation for e_i, leading to an interpolation with an increased set of interpolatory variables. The corresponding interpolation (called *standard interpolation* in [51]) is, in general, considerably more robust in practice. Alternatively, one may obtain an improved interpolation by applying one F-relaxation step (for more details, see "Jacobi-interpolation" in Section 5) to either the direct or the standard interpolation.

In both approaches, compared to the direct interpolation, the "radius" of interpolation increases which, in turn, will reduce the sparsity of the resulting Galerkin operator. However, interpolation weights corresponding to variables "far away" from variable i are typically much smaller than the largest ones. Before computing the Galerkin operator, one should therefore truncate the interpolation operator by ignoring all small entries (and re-scale the remaining weights so that the total sum remains unchanged). Note that, because of the variational principle, the truncation of interpolation is a "safe process"; in the worst case, overall convergence may slow down, but no divergence can occur. On the other hand, a truncation of the *Galerkin operators* can be dangerous since this destroys the validity of the variational principle and, if not applied with great care, may even cause strong divergence in practice.

Apart from other minor differences, the original AMG interpolation proposed in [48] (and realized in the code AMG1R5) can be regarded as a compromise between the direct interpolation and the standard interpolation described before. There, an attempt was made to replace all strongly coupled $e_j\,(j \notin P_i)$ in (7) by averages *involving only variables in* P_i. However, for this to be reasonable, based on certain criteria, new C-variables had to be added to a given splitting a posteriori ("final C-point choice" in [48]). Although this approach worked quite well in those cases treated in [48], typically too many additional C-variables are required in geometrically complex 3D situations, causing unacceptably high fill-in towards coarser levels (see Section 4.2 for examples). In practice, the standard interpolation outlined above (in combination with a reasonable truncation) has turned out to be more robust and often considerably more efficient.

The above improvements of interpolation generally lead to faster convergence but also increase the computational work per cycle and the required memory to some extent. Whether or not this finally pays, depends on the given application. If memory is an issue (as it is often in commercial environments), one may, instead, wish to *simplify* interpolation at the expense of a reduced convergence speed. One way to achieve this is to generally allow interpolation from variables which are *not* in the direct neighborhood. Such "long-range" interpolation [48] generally allows a much faster coarsening and drastically increases the sparsity on coarser levels. For details of a simple approach which has been tested in practice, see [51] ("aggressive coarsening" and "multi-pass interpolation").

3.2. The M-matrix case

In practice, it turns out that AMG V-cycle convergence is, to a large extent, independent of the problem size. Unfortunately, this cannot be proved based merely on algebraic arguments. Nevertheless, some important aspects can be investigated theoretically, in particular, matrix-independent two-level convergence can be proved for various classes of matrices if interpolation is defined properly. We consider here the class of M-matrices. Generalizations to other classes and the corresponding proofs can be found in [51].

3.2.1. Two-level considerations

The following theorem shows that direct interpolation based on (8) ensures matrix-independent two-level convergence if, for each $i \in F$, the connectivity represented in P_i is a *fixed fraction* of the total connectivity.

Theorem 1. *Let A be a symmetric, weakly diagonally dominant M-matrix. With fixed* $0 < \tau \leqslant 1$ *select a C/F-splitting so that, for each $i \in F$, there is a set $P_i \subseteq C \cap N_i$ satisfying*

$$\sum_{k \in P_i} |a_{ik}| \geqslant \tau \sum_{j \in N_i} |a_{ij}| \tag{9}$$

and define interpolation according to (8). *Then the two-level method, using one Gauss–Seidel relaxation step for* (*post*-) *smoothing, converges at a rate which depends only on τ but not on the*

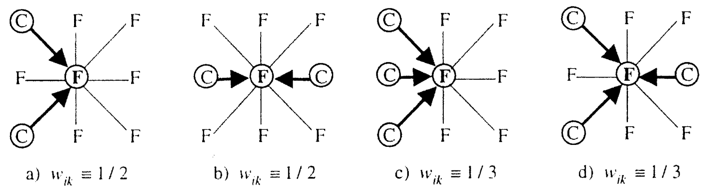

Fig. 2. Different C/F-arrangements and corresponding interpolation formulas.

given matrix,

$$\|SK\|_A \leqslant \sqrt{1-\tau/4}.$$

The above theorem confirms that it is the *strong* couplings which are important to interpolate from, while the use of *weak* couplings would increase the computational work but hardly affect the convergence. The more strong connections are used for interpolation, the better the convergence can be. Note that this implicitly means that coarsening will be "in the direction of smoothness" which is the main reason for the fact that AMG's convergence does not sensitively depend on anisotropies. Moreover, AMG's interpolation can be regarded as an algebraic generalization of the operator-dependent interpolation introduced in [1,20], which explains why the performance of AMG does not sensitively depend on large, discontinuous variations in the coefficients of the given system of equations. For an illustration, see Section 4.1.

From a practical point of view, the above convergence estimate is a worst-case estimate, at least if the given problem has some kind of geometric background (which it typically does). The reason is that the algebraic requirement (9) does not take the location of the interpolatory C-points, relative to the F-points, into account. For an illustration, consider the 9-point discretization of the Poisson operator

$$\frac{1}{3h^2}\begin{bmatrix} -1 & -1 & -1 \\ -1 & 8 & -1 \\ -1 & -1 & -1 \end{bmatrix}_h . \tag{10}$$

From geometric multigrid, we know that linear interpolation yields fast convergence. The algebraic interpolation, however, cannot distinguish between geometrically "good" and "bad" C/F-splittings. For instance, in Fig. 2a and b we use the same total weight for interpolation but the second arrangement will clearly result in much better convergence. Similarly, the arrangement in Fig. 2d, although it does not give exactly second order, will be much better than the one in Fig. 2c.

This illustrates that the concrete arrangement of a C/F-splitting will have a substantial influence on the quality of interpolation, and, through this, on the final convergence. In order to *strictly ensure* an optimal interpolation, we would have to exploit the geometric location of (strongly coupled) points among each other. In practice, however, it turns out to be sufficient to base the construction of a C/F-splitting on the following additional objective. As a rule, one should arrange the C/F-splitting so that the set of C-variables builds (approximately) a *maximal* set with the property that the C-variables are not strongly coupled among each other ("maximally independent set") and that the F-variables are "surrounded" by their interpolatory C-variables. This can be ensured to a sufficient extent by merely exploiting the connectivity information contained in the matrix (for an algorithm, see [48,51]). Note that strong connectivity does not necessarily have to be via direct couplings.

Observing this objective will, in practice, substantially enhance convergence even if only small sets of interpolatory variables are used.

3.2.2. *Extension to multi-level cycles*

Unfortunately, the assumptions on interpolation in Theorem 1 are sufficient for uniform two-level, but not for uniform V-cycle convergence. Although, by choosing $\tau \geqslant \frac{1}{2}$, one can ensure that all recursively defined Galerkin operators remain weakly diagonally dominant M-matrices and, hence, the formal extension to complete V-cycles is straightforward, A-independent V-cycle convergence cannot be proved. The reason is the limited accuracy of purely algebraically defined interpolation as discussed in the previous section. We will return to this problem in Section 6 where we consider a worst-case scenario in the context of "aggregation-type" AMG approaches.

In practice, however, one can observe V-cycle convergence which is, to a large extent, independent of the problem size if we take the additional objective of the previous section into account. Furthermore, it turns out that it is not important to force the coarse-level matrices to exactly remain M-matrices. On the contrary, such a requirement puts too many restrictions on the coarsening process, in particular on lower levels, where the size of the Galerkin operators then may grow substantially.

In this context, we want to emphasize again that, for an efficient overall performance, convergence speed is only one aspect. An equally important aspect is the complexity (sparsity) of the coarser level matrices produced by AMG (which directly influences both the run time and the overall memory requirement). Only if both the convergence speed and the *operator complexity*

$$c_A = \sum_{\ell} |m_\ell| / |m_1|, \tag{11}$$

(m_ℓ denotes the number of nonzero entries contained in the matrix on level ℓ) are bounded independent of the size of A, do we have an asymptotically optimal performance. The typical AMG performance in case of some complex problems is given in Section 4.2.

3.3. *AMG as a pre-conditioner*

In order to increase the robustness of geometric multigrid approaches, it has become very popular during the last years, to use multigrid not as a stand-alone solver but rather combine it with acceleration methods such as conjugate gradient. BI-CGSTAB or GMRES. This development was driven by the observation that it is often not only simpler but also more efficient to use accelerated multigrid approaches rather than to try to optimize the interplay between the various multigrid components in order to improve the convergence of stand-alone multigrid cycles.

This has turned out to be similar for AMG which, originally, was designed to be used stand-alone. Practical experience has clearly shown that AMG is also a very good pre-conditioner, much better than standard (one-level) ILU-type pre-conditioners, say. Heuristically, the major reason is due to the fact that AMG, in contrast to any one-level pre-conditioner, aims at the efficient reduction of *all* error components, short range as well as long range. However, although AMG tries to capture all relevant influences by proper coarsening and interpolation, its interpolation will hardly ever be optimal. It may well happen that error reduction is significantly less efficient for some very specific error components. This may cause a few eigenvalues of the AMG iteration matrix to be considerably closer to 1 than all the rest. If this happens, AMG's convergence factor is limited by the slow convergence of just a few exceptional error components while the majority of the error components is reduced very quickly. Acceleration by, for instance, conjugate gradient typically eliminates these particular frequencies very efficiently. The alternative, namely, to try to prevent such situations by putting more effort into the construction of interpolation, will generally be much more expensive. Even then, there is no final guarantee that such situations can be avoided. (We note that this even happens with "robust" geometric multigrid methods.)

4. Applications and performance

The flexibility of AMG and its simplicity of use, of course, have a price: A *setup phase*, in which the given problem is analyzed, the coarse levels are constructed and all operators are assembled, has to be concluded before the actual *solution phase* can start. This extra overhead is one reason for the fact that AMG is usually less efficient than geometric multigrid approaches (if applied to problems for which geometric multigrid *can* be applied efficiently). Another reason is that AMG's components can, generally, not be expected to be "optimal". They are always constructed on the basis of compromises between numerical work and overall efficiency. Nevertheless, if applied to standard elliptic test problems, the computational cost of AMG's solution phase (ignoring the setup cost) is typically comparable to the solution cost of a *robust* geometric multigrid solver [47].

However, AMG should not be regarded as a competitor of geometric multigrid. The strengths of AMG are its robustness, its applicability in complex geometric situations and its applicability to even solve certain problems which are out of the reach of geometric multigrid, in particular, problems with no geometric or continuous background at all. In such cases, AMG should be regarded as an efficient alternative to standard numerical methods such as conjugate gradient accelerated by typical (one-level) pre-conditioners. We will show some concrete performance comparisons in Section 4.2. Before, however, we want to illustrate the flexibility of AMG in adjusting its coarsening process locally to the smoothing properties of relaxation by means of a simple but characteristic model equation.

4.1. A model problem for illustration

We consider the model equation

$$-(au_x)_x-(bu_y)_y+cu_{xy}=f(x,y), \tag{12}$$

defined on the unit square with Dirichlet boundary conditions. We assume $a=b=1$ everywhere except in the upper left quarter of the unit square (where $b=10^3$) and in the lower right quarter (where $a=10^3$). The coefficient c is zero except for the upper right quarter where we set $c=2$. The resulting discrete system is isotropic in the lower left quarter of the unit square but strongly anisotropic in the remaining quarters. Fig. 3a shows what a "smooth" error looks like on the finest level after having applied a few Gauss–Seidel point relaxation steps to the homogeneous problem,

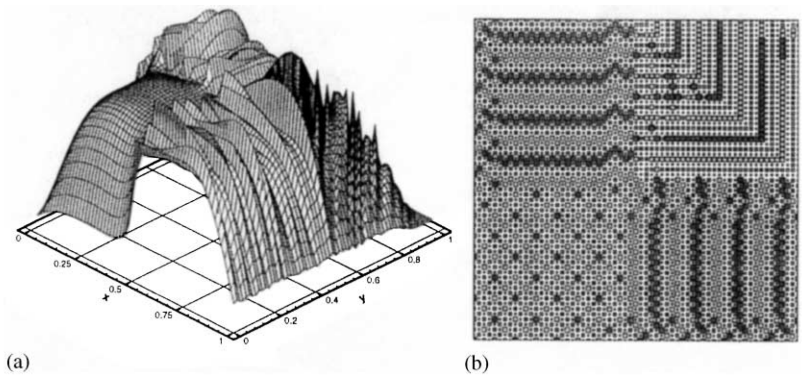

Fig. 3. (a) "Smooth" error in case of problem (12). (b) The finest and three consecutive levels created by the standard AMG coarsening algorithm.

starting with a random function. The different anisotropies as well as the discontinuities across the interface lines are clearly reflected in the picture.

It is heuristically clear that such an error can only be effectively reduced by means of a coarser grid if that grid is obtained by essentially coarsening in the directions in which the error really changes smoothly in the geometric sense and if interpolation treats the discontinuities correctly. Indeed, see Section 3.2, this is exactly what AMG does. First, the operator-based interpolation ensures the correct treatment of the discontinuities. Second, AMG coarsening is in the direction of strong connectivity, that is, in the direction of smoothness.

Fig. 3b depicts the finest and three consecutive grids created by using standard AMG coarsening and interpolation. The smallest dots mark grid points which are contained *only* on the finest grid, the squares mark those points which are also contained on the coarser levels (the bigger the square, the longer the corresponding grid point stays in the coarsening process). The picture shows that coarsening is uniform in the lower left quarter where the problem is isotropic. In the other quarters, AMG adjusts itself to the different anisotropies by locally coarsening in the proper direction. For instance, in the lower right quarter, coarsening is in the x-direction only. Since AMG takes only *strong* connections in coarsening into account and since all connections in the y-direction are *weak*, the individual lines are coarsened *independently of each other*. Consequently, the coarsening of neighboring x-lines is not "synchronized"; it is actually a matter of "coincidence" where coarsening starts within each line. This has to be observed in interpreting the coarsening pattern in the upper right quarter: within each diagonal line, coarsening is essentially in the direction of this line.

4.2. Complex applications

For a demonstration of AMG's efficiency, we consider some complex problems of the type typically solved in two commercial codes designed for *oil reservoir simulation* and for *computational fluid dynamics*, respectively. In both codes, the numerical kernel requires the fast solution of scalar elliptic equations. While, in oil reservoir simulation, geometries are typically fairly simple but the underlying problems have strongly anisotropic and discontinuous coefficients (jumps by several orders of magnitude in a nearly random way), in computational fluid dynamics these problems are

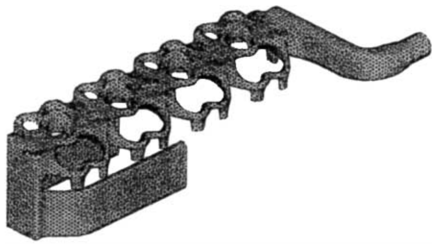

Fig. 4. Cooling jacket of a four-cylinder engine.

Poisson-like but defined on very complex, unstructured grids. For more details on these codes, see [51].

The following test cases are considered: [3]

(1) *Reservoir*. The underlying reservoir corresponds to a simple domain discretized by a mesh consisting of 1.16 million cells. The variation of absolute permeabilities results in a discontinuous variation of the coefficients by four orders of magnitude.
(2) *Cooling jacket*. Computation of the flow through the cooling jacket of a four-cylinder engine. The underlying mesh consists of 100 000 tetrahedra cells (see Fig. 4).
(3) *Coal furnace*. Computation of the flow inside the model of a coal furnace. The underlying mesh consists of 330 000 hexahedra and a few thousand pentahedra, including many locally refined grid patches.
(4) *Underhood*. Computation of the underhood flow of a Mercedes-Benz E-class model. The mesh is highly complex and consists of 910 000 cells (see Fig. 1).
(5) *E-Class*. Computation of the exterior flow over a Mercedes-Benz E-class model (see Fig. 5). The original mesh consists of 10 million cells. Due to memory restrictions, our test runs to two reduced mesh sizes consisting of 2.23 and 2.82 million cells, respectively. (Note that the underlying mesh also includes all modeling details of the previous underhood case.)

Memory requirement is a major concern for any commercial software provider. Industrial users of commercial codes always drive their simulations to the limits of their computers, shortage of memory being a serious one. For these reasons, in a commercial environment, low-memory AMG approaches are of primary interest, even if the reduction of the memory requirement is at the expense of a (limited) increase of the total computational time. Compared to standard one-level solvers, a memory overhead of some tens of percents is certainly acceptable. In any case, however, the operator complexity c_A (11) must not be larger than 2.0 say. Therefore, in the following test runs, we employ an aggressive coarsening strategy (cf. Section 3.1) and, in order to make up for the resulting reduced convergence speed, use AMG as a pre-conditioner rather than stand-alone.

Fig. 6 shows AMG's V-cycle convergence histories for each of the above cases, based on the code RAMG05 mentioned in the introduction. The results reflect the general experience that the convergence of AMG depends, to a limited extent, on the type of elements used as well as on the type of problem, but hardly on the problem size. In particular, the three Mercedes meshes are

[3] The first case has been provided by StreamSim Technologies Inc., the other ones by Computational Dynamics Ltd.

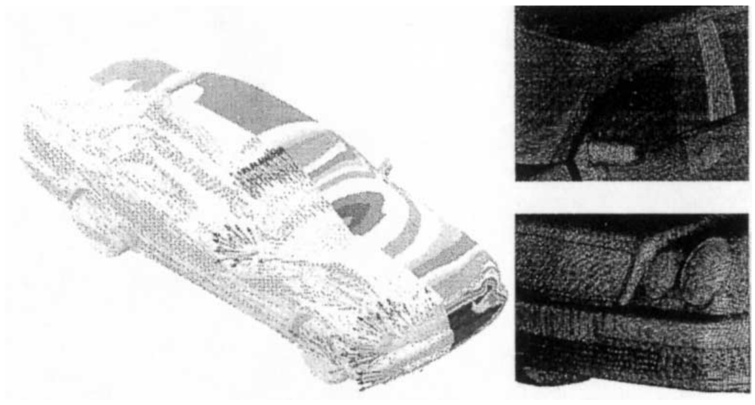

Fig. 5. Model of a Mercedes-Benz E-Class.

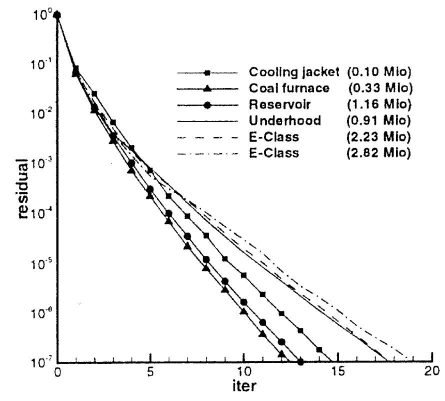

Fig. 6. RAMG05 convergence histories for various problems.

	RAMG05/cg			ILU(0)/cg		AMG1R5
problem	time	iter	c_A	time	iter	c_A
jacket	12.3	21	1.44	218.2	926	5.35
furnace	45.7	18	1.47	292.8	286	7.06
reservoir	165.0	18	1.41	2707.0	720	7.66
underhood	172.9	25	1.43	1364.0	461	5.64
eclass (2.23)	438.8	25	1.46	8282.0	1151	6.24

Fig. 7. Performance of RAMG05 vs. ILU(0)/cg.

comparable in their type but their size varies by more than a factor of three. Convergence, obviously, is influenced only marginally.

Fig. 7 compares the RAMG05 performance with that of ILU(0) pre-conditioned conjugated gradient. For both methods and for each of the above problems, the number of iterations as well as total run times (in s), required to reduce the residuals by nine digits, are shown. Compared to ILU(0)/cg.

AMG reduces the number of iterations by up to a factor of 46. In terms of run time, AMG is up to 19 times faster. The figure also shows that the industrial requirements in terms of memory, mentioned before, are fully met. In fact, the A-complexity (11) is very satisfactory for all cases, namely $c_A \approx 1.45$.

For a comparison, the last column in the figure shows the unacceptably high complexity values of RAMG05's forerunner, AMG1R5. As already mentioned in Section 3.1, AMG1R5 typically performs quite well in the case of 2D problems. In complex 3D cases as considered here, however, the results clearly demonstrate one of the advantages of the different coarsening and interpolation approaches used in RAMG05 (Fig. 7). (For more information on the differences in the two codes we refer to [51].)

5. AMG based on mere *F*-relaxation

In this section, we consider a very different approach [28,51] which can be used to *force* the right-hand side of (4) to become small. For a description, we assume vectors and matrices to be re-ordered so that, w.r.t a given C/F-splitting, the set of equations (1) can be written in block form

$$A_h u = \begin{pmatrix} A_{FF} & A_{FC} \\ A_{CF} & A_{CC} \end{pmatrix} \begin{pmatrix} u_F \\ u_C \end{pmatrix} = \begin{pmatrix} f_F \\ f_C \end{pmatrix} = f. \tag{13}$$

Correspondingly, the interpolation operator is re-written as $I_H^h = (I_{FC}, I_{CC})^{\mathrm{T}}$ with I_{CC} being the identity operator. Instead of $e^h = I_H^h e^H$, we simply write $e_F = I_{FC} e_C$.

5.1. The basic idea

The approach mentioned above is based on the fact that the sub-matrix A_{FF} is very well conditioned if we just select the C/F-splitting accordingly. For instance, for all problems we have in mind here, we can easily force A_{FF} to be *strongly diagonally dominant*,

$$a_{ii} - \sum_{j \in F, j \neq i} |a_{ij}| \geqslant \delta a_{ii} \quad (i \in F) \tag{14}$$

with some fixed $\delta > 0$. Assuming this to hold in the following, we can efficiently approximate the solution of the F-equations (with frozen e_C),

$$A_{FF} e_F + A_{FC} e_C = 0 \tag{15}$$

for instance, by relaxation (in the following called *F-relaxation*). Using this as the basis for both the definition of smoothing *and* interpolation, we can force the right-hand side of (4) to become as small as we wish.

To be more specific, given any e_C, interpolation is defined by applying μF-relaxation steps to approximately solve (15). In order to keep the resulting operator as "local" as possible, we only consider Jacobi-relaxation (below, we refer to this as *Jacobi-interpolation*). That is, we iteratively define a sequence of operators,

$$I_{FC}^{(\mu)} = P_{FF} I_{FC}^{(\mu-1)} - D_{FF}^{-1} A_{FC} \quad \text{where } P_{FF} = I_{FF} - D_{FF}^{-1} A_{FF}, \tag{16}$$

starting with some reasonable first-guess interpolation operator, $I_{FC}^{(0)}$. Because of (14), we have rapid convergence $(I_H^h)^{(\mu)} e_C \to \hat{e}\,(\mu \to \infty)$ at a rate which depends only on δ. Here $\hat{e} := (\hat{e}_F, e_C)^T$ where $\hat{e}_F := -A_{FF}^{-1} A_{FC} e_C$ denotes the solution of (15).

Similarly, we also use F-relaxation for smoothing (referred to as *F-smoothing* below). That is, we define one smoothing step by $u \to \bar{u}$ where

$$Q_{FF}\bar{u}_F + (A_{FF} - Q_{FF})u_F + A_{FC}u_C = f_F, \quad \bar{u}_C = u_C. \tag{17}$$

In contrast to the interpolation, we here normally use Gauss–Seidel relaxation, i.e., Q_{FF} is the lower triangular part of A_{FF} (including the diagonal). The corresponding smoothing operator is easily seen to be

$$S_h^{\nu} e = \begin{pmatrix} S_{FF}^{\nu}(e_F - \hat{e}_F) + \hat{e}_F \\ e_C \end{pmatrix} \quad \text{where } S_{FF} = I_{FF} - Q_{FF}^{-1} A_{FF}. \tag{18}$$

As with the interpolation, for any given $e = (e_F, e_C)^T$, we have rapid convergence $S_h^{\nu} e \to \hat{e}\ (\nu \to \infty)$.

5.2. Two-level convergence

For various classes of matrices A one can show that F-smoothing and Jacobi-interpolation can be used to obtain *matrix-independent* two-level convergence if the first-guess interpolation, $I_{FC}^{(0)}$, is selected properly. Moreover, two-level convergence becomes arbitrarily fast if ν, μ are chosen sufficiently large. As an example, we again consider the class of M-matrices (cf. Theorem 1).

Theorem 2. *Let A be a symmetric, weakly diagonally M-matrix. Define the interpolation by applying $\mu \geqslant 0$ Jacobi F-relaxation steps, using an interpolation as defined in Theorem* 1 *as the first-guess* (*with fixed* $0 < \tau \leqslant 1$). *Then, if $\nu \geqslant 1$ Gauss–Seidel F-relaxation steps are used for* (*pre-*)*smoothing, the following two-level convergence estimate holds,*

$$\|KS^{\nu}\|_A \leqslant \|S_{FF}\|_{A_{FF}}^{\nu} + \tilde{\tau}\|P_{FF}\|_{A_{FF}}^{\mu},$$

where $\|S_{FF}\|_{A_{FF}} < 1$ and $\|P_{FF}\|_{A_{FF}} < 1$, and both norms as well as $\tilde{\tau}$ depend only on τ but not on the matrix A.

In this theorem, we have used the interpolation from Theorem 1 as a first guess. In particular, we assume the C/F-splitting to satisfy (9) which can easily be seen to ensure strong diagonal dominance (14) with $\delta = \tau$. Although one may think of various other ways to define the first-guess interpolation, we want to point out that a proper selection of the first-guess interpolation is important for obtaining matrix-independent two-level convergence (it is, for instance, not sufficient to simply select $I_{FC}^{(0)} = 0$). Generally, the first-guess interpolation has to be such that the Galerkin operator which corresponds to it, $A_H^{(0)}$, is spectrally equivalent to the Schur complement, $A_{CC} - A_{CF}A_{FF}^{-1}A_{FC}$, w.r.t. all matrices in the class under consideration. For more details and generalizations of the above theorem as well as the proofs, see [51].

Note that the AMG approach discussed here is not really in the spirit of standard multigrid since smoothing in the usual sense is not exploited. In fact, the role of F-smoothing is merely to *force* $S^{\nu}e \approx \hat{e}$ rather than to smooth the error of the full system. This, together with Jacobi-interpolation, is a "brute force" approach to make $\|S^{\nu}e - I_H^h e_C\|_A$ small for all $e = (e_F, e_C)^T$.

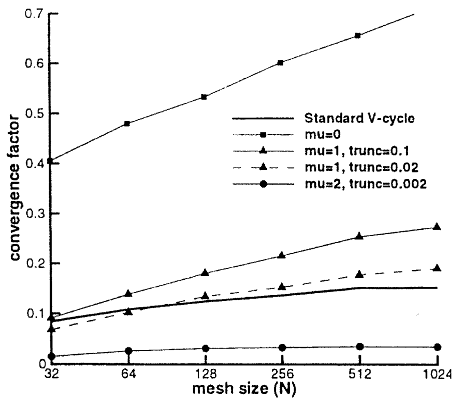

Fig. 8. Convergence factors of AMG based on F-relaxation.

5.3. Practical remarks

The mere fact that AMG can be *forced* to converge as fast as we wish, is only of little relevance in practice. Each F-relaxation step applied to the interpolation increases its "radius" by one additional layer of couplings, causing increased fill-in in the Galerkin operator. The resulting gain in convergence speed is, generally, more than eaten up for by a corresponding increase of matrix complexities towards coarser levels. Consequently, the main problem is the tradeoff between convergence and numerical work (which is directly related to the memory requirements). Note that this is, in a sense, just opposite to geometric multigrid where the numerical work per cycle is known and controllable but the convergence may not be satisfactory.

For a practical realization of Jacobi-interpolation, several things are important to observe. First, most of the new entries introduced by each additional relaxation step will be relatively small and can be truncated (before computing the Galerkin operator) without sacrificing convergence seriously (cf. also Section 3.1). Second, it is usually not necessary to perform F-relaxation with the complete matrix A_{FF}. Instead, one may well ignore all those entries of A_{FF} which are relatively small (and add them to the diagonal, say, in order to preserve the row sums of interpolation). Finally, we want to remark that, although Theorem 2 states fast convergence only if μ is sufficiently large, in practice, $\mu > 2$ is hardly ever required (at least if δ is not too small).

Fig. 8 shows some V-cycle convergence factors as a function of the mesh size for model equation

$$-((1+\sin(x+y))u_x)_x - (e^{x+y}u_y)_y = f(x,y), \tag{19}$$

discretized on the unit square with uniform mesh size $h = 1/N$. We first observe the rapid h-independent convergence of the "standard" AMG V-cycle (corresponding to the approach outlined in Section 3, using one full Gauss–Seidel relaxation step both for pre- and post-smoothing). Convergence drastically drops, and becomes strongly h-dependent, if we just replace each full smoothing step by two F-smoothing steps and leave interpolation unchanged (case $\mu = 0$). This has to be expected since the definition of interpolation in classical AMG is based on the assumption that the error after relaxation is *algebraically smooth* (cf. Section 3.1). This is, clearly, not true if only *partial* relaxation, such as F-relaxation, is performed. However, if we use just one Jacobi F-relaxation step to improve interpolation ($\mu = 1$), convergence becomes comparable to that of the standard cycle. Results are shown using two different truncation parameters, 0.1 and 0.02, respectively. Finally, the case $\mu = 2$ (and four partial relaxation steps for smoothing rather than two) gives a convergence which is about twice as fast as that of the standard cycle.

We note that, if computational time and memory requirement is taken into account in this example, the standard V-cycle is more efficient than the others. In particular, the cycle employing $\mu = 2$ is substantially inferior, mainly due to the considerably higher setup cost. This seems typical for applications for which algebraically smooth errors, in the sense of Section 3.1, can be characterized sufficiently well. The heuristic reason is that then, using *full* smoothing steps, relatively simple interpolations of the type outlined in Section 3.1 are usually sufficient to approximate algebraically smooth errors and obtain fast convergence. This is no longer true if mere F-smoothing is employed and, generally, additional effort needs to be invested to "improve" interpolation by F-relaxation in order to cope with all those error components which are not affected by mere F-smoothing. (In particular, note that all error components of the form $\hat{e}$ are not reduced at all by F-smoothing.)

In general, however, when the characterization of algebraically smooth errors is less straightforward, the use of F-relaxation provides a means to enhance convergence. Further numerical experiments employing F-smoothing and Jacobi-interpolation can be found in [28,51]. F-relaxation is a special case of a "compatible relaxation" which, in a more general context, is considered in [11].

6. Aggregation-type AMG

In the previous sections, we have considered increasingly complex interpolation approaches. In this section, we go back and consider the most simple case that each F-variable interpolates from *exactly* one C-variable only. We have already pointed out in Section 3.2 that the use of such "one-sided" interpolations is not recommendable. In fact, one important goal of the additional objective introduced in Section 3.2 was just *to avoid* such extreme interpolations. On the other hand, the resulting method is so easy to implement that it, nevertheless, has drawn some attention. We will outline the fundamental problems with this approach in Section 6.2 and summarize three possibilities of improvement in Sections 6.3–6.5. Since we just want to highlight the main ideas, we restrict our motivations to very simple but characteristic (Poisson-like) problems.

6.1. The basic approach

Let us consider C/F-splittings and interpolations (6) where, for each $i \in F$, $w_{ik} = 1$ for just one particular $k \in C$ and zero otherwise. Consequently, the total number of variables can be subdivided into "aggregates" I_k ($k \in C$) where I_k contains (apart from k itself) all indices i corresponding to F-variables which interpolate from variable k (see Fig. 9).

With this notation, the computation of the Galerkin operator becomes very simple. One easily sees that

$$I_h^H A_h I_H^h = (a_{kl}^H) \quad \text{where } a_{kl}^H = \sum_{i \in I_k} \sum_{j \in I_l} a_{ij}^h \quad (k, l \in C), \tag{20}$$

that is, the coefficient a_{kl}^H is just the sum of all cross-couplings between I_k and I_l. Obviously, regarding the coefficients a_{kl}^H, the particular role of the variables k and l (as being C-variables) is

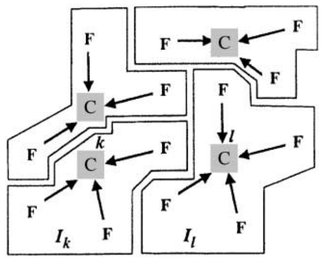

Fig. 9. Subdivision of fine-level variables into aggregates. The arrows indicate which C-variable an F-variable interpolates from.

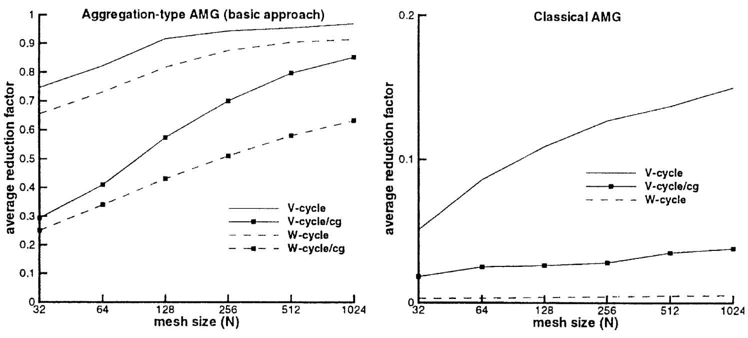

Fig. 10. Convergence of (a) aggregation-type AMG, (b) classical AMG.

not distinguished from the other variables; the Galerkin operator merely depends *on the definition of the aggregates*. Consequently, we might as well associate each aggregate I_k with some "new" coarse-level variable which has no direct relation to the C-variable k. The above interpolation then is nothing else than *piecewise constant interpolation* from these new coarse-level variables to the associated aggregates.

Originally, such *aggregation-type* AMG approaches [52,53,9] have been developed the other way around: Coarsening is defined by building aggregates (rather than constructing C/F-splittings), a new coarse-level variable is associated with each aggregate and interpolation I_H^h is defined to be piecewise constant. (That is, the set of coarse-level variables is generally *not* considered as a subset of the fine-level ones.) Clearly, for a given subdivision into aggregates to be reasonable, all variables in the same aggregate should strongly depend on each other. Otherwise, piecewise constant interpolation makes no real sense.

As expected, an immediate implementation of this simple coarsening and interpolation approach leads to very inefficient solvers, even if used only as a pre-conditioner. Fig. 10a shows the typical convergence of both the V- and the W-cycle, used as stand-alone and as pre-conditioner, in solving the model equation (19). Convergence is indeed very slow and exhibits a *strong h-dependency*. For a comparison, the much better performance of classical AMG is depicted in Fig. 10b.

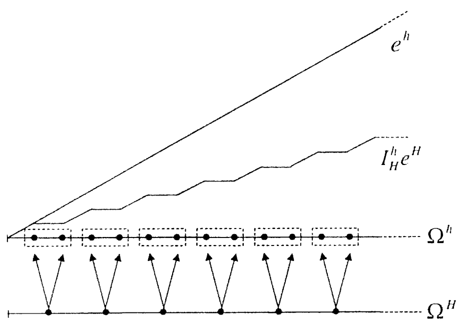

Fig. 11. Optimal approximation $I_H^h e^H$ of e^h w.r.t. the energy norm.

6.2. *The reason for slow convergence*

The main reason for this unsatisfactory convergence is that piecewise constant interpolation is not able to approximate the *values* of smooth error if approximation is based on the *energy norm* (cf. (4)). In fact, the approximation order becomes zero.

To illustrate this, let us consider the most simple case that A_h is derived from discretizing $-u''$ on the unit interval with meshsize h, i.e., the rows of A_h correspond to the difference stencil

$$\frac{1}{h^2}[-1 \quad 2 \quad -1]_h$$

with Dirichlet boundary conditions. Let e^h be an error satisfying the homogeneous boundary conditions. According to the variational principle, the corresponding two-level correction, $I_H^h e^H$, is optimal in the sense that it minimizes $\|e^h - I_H^h e^H\|_{A_h}$ w.r.t. all possible corrections in $\mathcal{R}(I_H^h)$. (At this point, the concrete choice of I_H^h is not relevant.) This implies that $I_H^h e^H$ minimizes

$$\|v^h\|_{A_h}^2 = (A_h v^h \cdot v^h) = \frac{1}{2h^2} \sum_{i,j}{}' (v_i^h - v_j^h)^2 + \sum_i s_i (v_i^h)^2, \tag{21}$$

where $v^h = e^h - I_H^h e^H$ and $s_i = \sum_j a_{ij}^h$. (The prime indicates that summation is only over neighboring variables i and j.) This, in turn, means that, away from the boundary (where we have $s_i \equiv 0$), the Euclidian norm of the *slope* of v^h is minimal. At the boundary itself we have $s_i \neq 0$, and v^h equals zero.

Assuming now the aggregates to be built by joining the pairs of neighboring variables, the result of this minimization is illustrated in Fig. 11 (see also [9,10]). We here consider a smooth error e^h in the neighborhood of the left boundary of the unit interval. On each aggregate, interpolation is constant and the slope of $I_H^h e^H$ necessarily vanishes. On the remaining intervals, the Euclidian norm of the slope of v^h becomes minimal if the slope of $I_H^h e^H$ equals that of e^h. Consequently, $I_H^h e^H$ has, on the average, *only half the slope of* e^h (independent of h). That is, the resulting approximation is off by a factor of approximately 0.5 if compared to the best approximation in the Euclidian sense. (Note that subsequent smoothing smooths out the "wiggles", but does not improve the quality of the correction.) Accordingly, the Galerkin operator, which can easily be computed, turns out to be too large by a factor of two compared to the "natural" $2h$-discretization of $-u''$.

If the same strategy is now used recursively to introduce coarser and coarser levels, the above arguments carry over to each of the intermediate levels and, in particular, each coarser-grid Galerkin operator is off by a factor of 2 compared to the previous one. A simple recursive argument shows that errors are accumulated from grid to grid and the asymptotic V-cycle convergence factor cannot be expected to be better than $1-2^{-m}$, where m denotes the number of coarser levels. That is, *the V-cycle convergence is strongly h-dependent.*

6.3. Improvement by re-scaling the Galerkin operators

The fact that piecewise constant interpolation produces badly scaled AMG components, was the basis for an improvement introduced in [9]. In that paper, it is demonstrated that convergence can be substantially improved by just multiplying corrections $I_H^h e^H$ by some suitable factor $\alpha > 1$ ("over-correction"). This is equivalent to re-scaling the Galerkin operator by $1/\alpha$

$$I_h^H A_h I_H^h \to \frac{1}{\alpha} I_h^H A_h I_H^h$$

and leaving everything else unchanged.

In case of the simple model equation $-u''$ considered in the previous section, $\alpha = 2$ would be the optimal choice. However, the main arguments carry over to the Poisson equation in 2D and 3D, assuming a uniform grid and the aggregates to be built by 2×2 and $2 \times 2 \times 2$ blocks of neighboring variables, respectively. In case of more general problems and/or different grids, the optimal weight is no longer $\alpha = 2$. Nevertheless, it has been demonstrated in [9] that a slightly reduced value of $\alpha = 1.8$ (in order to reduce the risk of "overshooting") yields substantially improved V-cycle convergence for various types of problems, *if the cycle is used as a pre-conditioner* and if the number of coarser levels is kept fixed (in [9] four levels are always used). Smoothing is done by symmetric Gauss–Seidel relaxation sweeps.

A comparison of Figs. 10a and 12a shows the convergence improvement if re-scaling by $\alpha = 1.8$ is applied to the model equation (19). (In contrast to [9], we here have not restricted the number of coarser levels.) Fig. 12a shows that there is indeed a risk of "overshooting": For larger meshes, the V-cycle starts to diverge. (Note that the above re-scaling destroys the validity of the variational principle and the iteration process may well diverge.) Using the V-cycle as a pre-conditioner, eliminates the problem.

We want to point out that the above comparison shows only the tendency of improvements due to re-scaling, the concrete gain depends on how the aggregates are chosen precisely (which is not optimized here and can certainly be improved to some extent). In any case, the gain in convergence, robustness and efficiency of this (very simple and easily programmable) approach are somewhat limited, one reason being that a good value of α depends on various aspects such as the concrete problem, the type of mesh and, in particular, the type and size of the aggregates. For instance, if the aggregates are composed of three neighboring variables (rather than two) in each spatial direction, the same arguments as in the previous section show that the best weight would be $\alpha \approx 3$ in case of Poisson's equation. If the size of the aggregates strongly varies over the domain, it becomes difficult to define a good value for α.

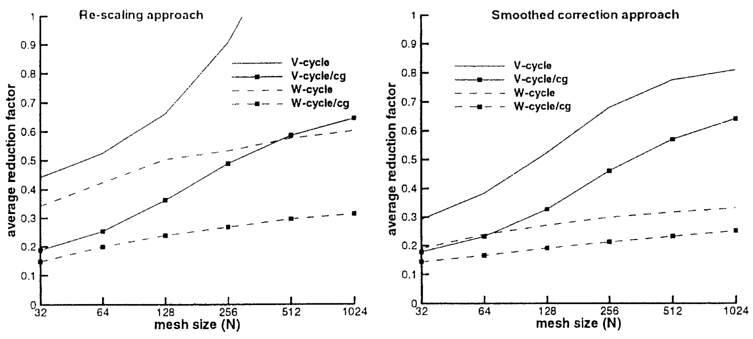

Fig. 12. (a) Re-scaling approach ($\alpha = 1.8$), (b) smoothed correction approach.

6.4. *Improvement by smoothing corrections*

Rather than explicitly prescribing a scaling factor α as before, a reasonable scaling can also be performed automatically. The idea is to modify the coarse-level correction step (2) by replacing the true (piece-wise constant) correction $e^h = I_H^h e^H$ by some approximation, e_0^h, and then compute u_{new}^h by

$$u_{\text{new}}^h = u_{\text{old}}^h + \alpha e_0^h \quad \text{with } \alpha = \frac{(f^h - A_h u_{\text{old}}^h, e_0^h)}{(A_h e_0^h, e_0^h)}, \tag{22}$$

instead of (2). Note that α is defined so that the energy norm of the error of u_{new}^h becomes minimal.

Clearly, for this minimization to be meaningful, the selection of e_0^h is crucial. Most importantly, e_0^h should be some sufficiently *smooth* approximation to e^h. (The choice $e_0^h = e^h$ would not give any gain: The variational principle would just imply $\alpha = 1$.) One possible selection is

$$e_0^h = S_h^\nu e^h, \tag{23}$$

which requires the application of ν smoothing steps to the *homogeneous* equations (starting with e^h). Note that, loosely speaking, this process will leave the "smooth part" of e^h essentially unchanged; only its "high-frequency part" will be reduced. Consequently, the regular smoothing steps, applied to u_{new}^h after the coarse-grid correction, will effectively correct this.

The effect of this process of *smoothing corrections*, is demonstrated in Fig. 12b (using $\nu = 2$). Apart from the fact that, compared to the re-scaling approach (see Fig. 12a), convergence is slightly better here, there is no risk of "overshooting" as before since the process "controls itself". On the other hand, the additional smoothing steps increase the overall computing time.

Although smoothing of corrections is a simple means to automatically correct wrong scalings to some extent, its possibilities are limited. In any case, the resulting overall performance is generally worse than that of classical AMG.

6.5. *Improvement by smoothing the interpolation*

A more sophisticated (but also more costly) way to accelerate the basic aggregation-type AMG approach is developed and analyzed in [52–54]. Here, piecewise constant interpolation is only considered as a first-guess interpolation which is improved by some *smoothing process* ("smoothed

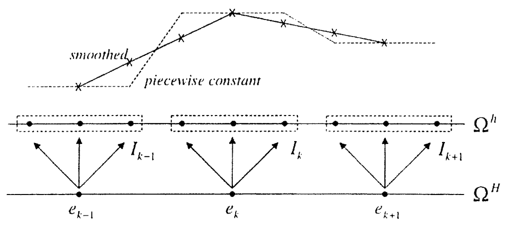

Fig. 13. Piecewise constant versus smoothed interpolation.

aggregation") before the Galerkin operator is computed. In [52,53], this smoothing is proposed to be done by applying one ω-Jacobi-relaxation step.

To be more specific, denote the operator corresponding to piecewise constant interpolation by $\tilde{I}_H^h$. Then the interpolation actually used is defined by

$$I_H^h = (I_h - \omega D_h^{-1} A_h^f)\tilde{I}_H^h,$$

where $D_h = \mathrm{diag}(A_h^f)$ and A_h^f is derived from the original matrix A_h by adding all weak connections to the diagonal ("filtered matrix"). That is, given some coarse-level vector e^H, $e^h = I_H^h e^H$ is defined by applying one ω-Jacobi relaxation step to the homogeneous equations $A_h^f v^h = 0$, starting with the piecewise constant vector $\tilde{I}_H^h e^H$. (Note that this process will increase the "radius" of interpolation and, hence, destroy the simplicity of the basic approach. Moreover, interpolation will generally *not* be of the special form (5) any more. Note also that here Jacobi relaxation serves a quite different purpose than Jacobi-F-relaxation as considered in Section 5. In particular, Jacobi-relaxation is here used as a smoother, applied to the full system of equations, which requires the use of an under-relaxation parameter, ω.)

To illustrate this process, we again consider the 1D case of $-u''$ and assume the aggregates to consist of three neighboring variables (corresponding to the typical size of aggregates used in [52,53] in each spatial direction). Note first that, since all connections are strong, we have $A_h^f = A_h$. Fig. 13 depicts both the piecewise constant interpolation (dashed line) and the smoothed interpolation obtained after the application of one Jacobi-step with $\omega = \frac{2}{3}$ (solid line). Obviously, the smoothed interpolation just corresponds to *linear* interpolation if the coarse-level variables are regarded as the fine-level analogs of those variables sitting in the center of the aggregates. Linear interpolation, however, does not exhibit a scaling problem as described in Section 6.2 for piecewise constant interpolation.

Of course, in more general situations, relaxation of piecewise constant interpolation will not give exact linear interpolation any more and a good choice of ω depends on the situation. Nevertheless, even if $\omega = \frac{2}{3}$ is kept fixed, smoothed interpolation will typically be much better than the piecewise constant one. (Actually, the real advantage of smoothed compared to piecewise constant interpolation is that errors, obtained after interpolation from the coarser level, have a much lower "energy"; see also Section 7.) This is demonstrated in [53] by means of various examples using a mixture of Gauss–Seidel and SOR sweeps for error smoothing. The tendency is to compose aggregates of three neighboring variables in each spatial direction. Note that a good value for ω depends not only on the problem and the underlying mesh, but also on the size of the aggregates. If, instead, only *two* neighbors would be aggregated in each spatial direction, one easily confirms that $\omega \approx 0.5$ should be chosen.

In general, classical AMG and AMG based on smoothed aggregation perform comparably if applied to relatively smooth (Poisson-like) problems. A certain advantage of aggregation-type AMG is that it, often, requires less memory than classical AMG (due to its very fast coarsening which causes a particularly low operator complexity c_A). On the other hand, this is payed for by a reduced robustness: The aggregation-type code seems to *require* acceleration by conjugate-gradient to maintain an acceptable efficiency and robustness in more complex situations. Since classical AMG puts more effort into the construction of interpolation and performs a slower coarsening, its performance generally depends on aspects such as strong discontinuities only to a lesser extent (for some examples, see [51]).

7. Further developments and conclusions

Algebraic multigrid provides a general approach to developing robust and efficient methods for solving large classes of matrix equations such as those typically arising in the numerical solution of elliptic PDEs, on structured as well as unstructured grids, in 2D and 3D. The construction of suitable interpolation operators (including the coarsening process) is crucial for obtaining fast and (nearly) h-independent convergence. Generally, the more effort is put into this construction, the faster the convergence can be, but, unfortunately, the required numerical work may increase even faster. That is, the main problem in designing efficient AMG algorithms is the tradeoff between convergence and numerical work, and keeping the balance between the two is the ultimate goal of any practical algorithm.

We have summarized and discussed various possibilities to realize concrete AMG algorithms. For most applications of the type discussed in this paper, methods based on the classical AMG approach still belong to the most efficient ones. An extensive list of experiments, based on the original code AMG1R5, can be found in [18]. Robustness and efficiency can substantially be improved, in particular in case of complex 3D meshes, by employing modifications as mentioned in Sections 3.1 and 5 (for more details, see [51]). AMG methods based on smoothed aggregation (see Section 6.5) are an efficient alternative to classical AMG, at least if employed as a pre-conditioner rather than stand-alone. Further developments and applications which are close to the classical AMG ideas are, for example, contained in [16,23,24,26,32,33,40,62,63]. Related approaches, but with a focus on different coarsening and interpolation strategies, are, for example, found in [22,27]. Applications of the aggregation-type approach in computational fluid dynamics are found in [30,39,46,60].

However, there are still many applications for which algebraically defined interpolation, and hence the resulting AMG performance, are not yet satisfactory. For instance, one of the major current research activities in AMG aims at its generalization to efficiently treat *systems* of PDEs such as linear elasticity problems. Although AMG has successfully been applied to various cases (see, e.g., [9,14,31,48,53]), its development has not yet reached a state where a particular approach is well settled. However, even for scalar applications, there are still questions about best ways to define coarsening and interpolation, for instance, if the given matrix is s.p.d., contains relatively large positive off-diagonal entries, and is far from being weakly diagonally dominant. In such cases, the performance of classical AMG may be only suboptimal.

It is often possible to avoid such situations by simplifying the given matrix before applying AMG [41]. One can also imagine situations where it would be advantageous (and easy) to provide AMG with some additional information on the problem at hand. For instance, information on the geometry (in terms of point locations) or more concrete descriptions on what an "algebraically smooth" error looks like (e.g. in the form of some user-provided "test-vector(s)"). This additional information can be used to fit AMG's interpolation in order to approximate certain types of error components particularly well. Straightforward possibilities have already been pointed out in [48].

In the following, we briefly summarize a few more recent approaches to define interpolation which aim at increasing the robustness in cases such as those mentioned above.

A new way to construct interpolation (AMGe, [14]) starts from the fact that an algebraically smooth error is nothing else but an error which is slow-to-converge w.r.t. the relaxation process. Hence, an algebraically smooth error, generally, corresponds to the eigenvectors of A belonging to the smallest eigen-values. Instead of defining interpolation by directly approximating (7), the goal in [14] is to define interpolation so that the smaller the associated eigenvalue is the better the eigenvectors are interpolated. To satisfy this by explicitly computing eigenvectors is, of course, much too expensive. However, in the case of finite element methods – assuming the element stiffness matrices to be known – one can derive measures (related to measures used in classical multigrid theory) whose minimization allows the determination of *local* representations of algebraically smooth error components in the above sense. The added robustness has been demonstrated in [14] by means of certain model applications. However, the approach is still in its infancy. In particular, significant development work still has to be invested to link the processes of coarsening and interpolation definition in order to obtain an optimal algorithm. In any case, it is an interesting new approach which has the potential of leading to more generally applicable AMG approaches.

Other algebraic approaches, designed for the solution of equations derived from finite-element discretizations, have been considered in [31,58]. Both approaches are aggregation based and the coarse space basis functions are defined so that their energy is minimized in some sense. (In the finite-element context it is natural to define interpolation implicitly by constructing the coarse space basis functions.) This does not require the element stiffness matrices to be known, but leads to a *global* (constraint) minimization problem the exact solution of which would be very expensive. However, iterative solution processes are proposed in both papers to obtain approximate solutions, indicating that the extra work (invested in the setup phase) is acceptable. While Wan et al. [58] concentrate on scalar applications, an extension to systems of PDEs (from linear elasticity) is one major aspect in [31]. Special attention is paid to the correct treatment of zero-energy modes (e.g. rigid-body modes in case of linear elasticity): such modes should be contained in the span of the coarse space basis functions, at least away from Dirichlet boundaries. (Note that, for typical scalar problems, this corresponds to the requirement that constants should be interpolated exactly away from Dirichlet boundaries, cf. (8).) It is interesting that the approach in [31] can be regarded as an extension of the earlier work [53] on smoothed aggregation: if only one iteration step is performed to approximately solve the energy minimization problem, the resulting method coincides with the smoothed aggregation approach. In contrast to the latter, however, further iterations will *not* increase the support of the basis functions (i.e., the radius of interpolation). Some test examples in [31] indicate the advantages of this new interpolation in terms of convergence speed. Unfortunately, however, this benefit is essentially offset by the expense of the minimization steps.

There are various other papers with focus on the development of multigrid methods to solve finite-element problems on unstructured grids. Although some of them are also based on algorithmical components which are, more or less, algebraically defined, most of them are not meant to be algebraic multigrid solvers in the sense as considered in this paper. We therefore do not want to discuss such approaches here further but rather refer, for example, to [15] and the references given therein.

In the approach of [56], A is not assumed to be asymmetric, and interpolation and restriction are constructed separately. Interpolation, for instance, is constructed so that a smooth error, $S_h e^h$, is interpolated particularly well w.r.t. the *Euclidian* norm, $\|\cdot\|_2$. More precisely, the attempt is to make

$$\|S_h e^h - I_H^h e^H\|_2,$$

where e^H denotes the straight injection of $S_h e^h$ to the coarse level, as small as possible (cf. (4)). In [56], this leads to certain local minimizations which are used to find, for each variable, pairs of neighboring variables which would allow a good interpolation in the above sense, and, at the same time, compute the corresponding weights (of both the interpolation and the restriction). Based on this information, a C/F-splitting is constructed which allows each F-variable to interpolate from one of the pairs found before. A heuristic algorithm is used to minimize the total number of C-variables.

In this context, we want to point out that, although classical AMG has been developed in the variational framework, it has successfully been applied to a large number of non-symmetric problems without any modification. This can be explained heuristically but no theoretical justification is available at this time. In the context of smoothed aggregation-based AMG, a theoretical analysis can be found in [25].

An important aspect which has not been addressed in this paper is the parallelization of AMG. An efficient parallelization of classical AMG is rather complicated and requires certain algorithmical modifications in order to limit the communication cost without sacrificing convergence significantly. Most parallelization approaches investigated up to now either refer to simple aggregation-based variants (e.g. [46]) or use straightforward domain decomposition techniques (such as Schwarz' alternating method) for parallelization. A parallelization strategy which stays very close to the classical AMG approach is presented in [29]. Results for complex 3D problems demonstrate that this approach scales reasonably well on distributed memory computers as long as the number of unknowns per processor is not too small. The method discussed in [56] is also available in parallel. There are several further ongoing parallelization activities, for instance, at the University of Bonn and the National Laboratories LLNL [17] and LANL, but no results have been published by now.

It is beyond the scope of this paper to also discuss the variety of hierarchical algebraic approaches which are not really related to the multigrid idea in the sense that these approaches are not based on the fundamental multigrid principles, smoothing and coarse-level correction. There is actually a rapid and very interesting ongoing development of such approaches. For completeness, however, we include some selected references. Various approaches based on approximate block Gauss elimination ("Schur-complement" methods) are found in [2–5,19,36–38,42,57]. Multi-level structures have also been introduced into ILU-type pre-conditioners, for example, in [49]. Very recently, some hybrid methods have been developed which use ideas from ILU and from multigrid [6–8,43–45]. For a further discussion, see also [55].

Summarizing, the development of hierarchically operating algebraic methods to efficiently tackle the solution of large sparse, unstructured systems of equations, currently belongs to one of the most active fields of research in numerical analysis. Many different methods have been investigated but, by now, none of them is really able to efficiently deal with *all* practically relevant problems. All methods seem to have their range of applicability but all of them may fail to be efficient in certain other applications. Hence, the development in this exciting area of research has to be expected to continue for the next years.

References

[1] R.E. Alcouffe, A. Brandt, J.E. Dendy, J.W. Painter, The multi-grid method for the diffusion equation with strongly discontinuous coefficients, SIAM J. Sci. Statist. Comput. 2 (1981) 430–454.
[2] O. Axelsson, The method of diagonal compensation of reduced matrix entries and multilevel iteration, J. Comput. Appl. Math. 38 (1991) 31–43.
[3] O. Axelsson, M. Neytcheva, Algebraic multilevel iteration method for Stieltjes matrices, Numer. Linear Algebra Appl. 1 (3) (1994) 213–236.
[4] O. Axelsson, P.S. Vassilevski, Algebraic multilevel preconditioning methods I, Numer. Math. 56 (1989) 157–177.
[5] O. Axelsson, P.S. Vassilevski, Algebraic multilevel preconditioning methods II, SIAM Numer. Anal. 27 (1990) 1569–1590.
[6] R.E. Bank, R.K. Smith, The incomplete factorization multigraph algorithm, SIAM J. Sci. Comput., to appear.
[7] R.E. Bank, R.K. Smith, The hierarchical basis multigraph algorithm, SIAM J. Sci. Comput., submitted.
[8] R.E. Bank, Ch. Wagner, Multilevel ILU decomposition, Numer. Math. 82 (1999) 543–576.
[9] D. Braess, Towards algebraic multigrid for elliptic problems of second order, Computing 55 (1995) 379–393.
[10] A. Brandt, Algebraic multigrid theory: the symmetric case, Appl. Math. Comput. 19 (1986) 23–56.
[11] A. Brandt, General highly accurate algebraic coarsening schemes, Proceedings of the Ninth Copper Mountain Conference on Multigrid Methods, Copper Mountain, April 11–16, 1999.
[12] A. Brandt, S.F. McCormick, J. Ruge, Algebraic multigrid (AMG) for automatic multigrid solution with application to geodetic computations, Institute for Computational Studies, POB 1852, Fort Collins, Colorado, 1982.
[13] A. Brandt, S.F. McCormick, J. Ruge, Algebraic multigrid (AMG) for sparse matrix equations, in: D.J. Evans (Ed.), Sparsity and its Applications, Cambridge University Press, Cambridge, 1984, pp. 257–284.
[14] M. Brezina, A.J. Cleary, R.D. Falgout, V.E. Henson, J.E. Jones, T.A. Manteuffel, S.F. McCormick, J.W. Ruge, Algebraic multigrid based on element interpolation (AMGe), LLNL Technical Report UCRL-JC-131752, SIAM J. Sci. Comput., to appear.
[15] T. Chan, J. Xu, L. Zikatanov, An agglomeration multigrid method for unstructured grids, Proceedings of the 10th International Conference on Domain Decomposition Methods, 1988.
[16] Q. Chang, Y.S. Wong, H. Fu, On the algebraic multigrid method, J. Comput. Phys. 125 (1996) 279–292.
[17] A.J. Cleary, R.D. Falgout, V.E. Henson, J.E. Jones, Coarse-grid selection for parallel algebraic multigrid, Proceedings of the "Fifth International Symposium on Solving Irregularly Structured Problems in Parallel", Lecture Notes in Computer Science, vol. 1457, Springer, New York, 1998, pp. 104–115.
[18] A.J. Cleary, R.D. Falgout, V.E. Henson, J.E. Jones, T.A. Manteuffel, S.F. McCormick, G.N. Miranda, J.W. Ruge, Robustness and scalability of algebraic multigrid, SIAM J. Sci. Comput., special issue on the "Fifth Copper Mountain Conference on Iterative Methods", 1998.
[19] W. Dahmen, L. Elsner, Algebraic Multigrid Methods and the Schur Complement, Notes on Numerical Fluid Mechanics, 23, Vieweg Verlag, Braunschweig, 1988.
[20] J.E. Dendy Jr., Black box multigrid, J. Comput. Phys. 48 (1982) 366–386.
[21] J.E. Dendy, S.F. McCormick, J. Ruge, T. Russell, S. Schaffer, Multigrid methods for three-dimensional petroleum reservoir simulation, Proceedings of 10th SPE Symposium on Reservoir Simulation, February 6–8, 1989.
[22] J. Fuhrmann, A modular algebraic multilevel method, Technical Report Preprint 203, Weierstrass-Institut für Angewandte Analysis und Stochastik, Berlin, 1995.
[23] T. Grauschopf, M. Griebel, H. Regler, Additive multilevel-preconditioners based on bilinear interpolation, matrix dependent geometric coarsening and algebraic multigrid coarsening for second order elliptic PDEs, Appl. Numer. Math. 23 (1997) 63–96.

[24] M. Griebel, T. Neunhoeffer, H. Regler, Algebraic multigrid methods for the solution of the Navier–Stokes equations in complicated geometries, SFB-Bericht Nr. 342/01/96 A, Institut für Informatik, Technische Universität München, 1996.
[25] H. Guillard, P. Vanek, An aggregation multigrid solver for convection-diffusion problems on unstructured meshes, Center for Computational Mathematics, University of Denver, Report 130, 1998.
[26] W.Z. Huang, Convergence of algebraic multigrid methods for symmetric positive definite matrices with weak diagonal dominance, Appl. Math. Comput. 46 (1991) 145–164.
[27] F. Kickinger, Algebraic multi-grid for discrete elliptic second order problems, Institutsbericht 513, Universität Linz, Institut für Mathematik, 1997.
[28] A. Krechel, K. Stüben, Operator dependent interpolation in algebraic multigrid, Proceedings of the Fifth European Multigrid Conference, Stuttgart, October 1–4, 1996; Lecture Notes in Computational Science and Engineering, vol. 3, Springer, Berlin, 1998.
[29] A. Krechel, K. Stüben, Parallel algebraic multigrid based on subdomain blocking, Parallel Comput., to appear.
[30] R.D. Lonsdale, An algebraic multigrid solver for the Navier–Stokes equations on unstructured meshes, Int. J. Numer. Methods Heat Fluid Flow 3 (1993) 3–14.
[31] J. Mandel, M. Brezina, P. Vanek, Energy optimization of algebraic multigrid bases, UCD/CCM Report 125, 1998.
[32] S. McCormick, J. Ruge, Algebraic multigrid methods applied to problems in computational structural mechanics, in: State-of-the-Art Surveys on Computational Mechanics, ASME, New York, 1989, pp. 237–270.
[33] R. Mertens, H. De Gersem, R. Belmans, K. Hameyer, D. Lahaye, S. Vandewalle, D. Roose, An algebraic multigrid method for solving very large electromagnetic systems, IEEE Trans. Magn. 34 (1998), to appear.
[34] W.A. Mulder, A new multigrid approach to convection problems, J. Comput. Phys. 83 (1989) 303–323.
[35] N.H. Naik, J. van Rosendale, The improved robustness of multigrid elliptic solvers based on multiple semicoarsened grids, SIAM Numer. Anal. 30 (1993) 215–229.
[36] Y. Notay, An efficient algebraic multilevel preconditioner robust with respect to anisotropies, in: O. Axelsson, B. Polman (Eds.), Algebraic Multilevel Iteration Methods with Applications, Department of Mathematics, University of Nijmegen, 1996, pp. 111–228.
[37] Y. Notay, Using approximate inverses in algebraic multilevel methods, Numer. Math. 80 (1998) 397–417.
[38] Y. Notay, Optimal V-cycle algebraic multilevel preconditioning, Numer. Linear Algebra Appl., to appear.
[39] M. Raw, A coupled algebraic multigrid method for the 3D Navier–Stokes equations, Report: Advanced Scientific Computing Ltd., 554 Parkside Drive, Waterloo, Ontario N2L 5Z4, Canada.
[40] H. Regler, Anwendungen von AMG auf das Plazierungsproblem beim Layoutentwurf und auf die numerische Simulation von Strömungen, Ph.D. thesis, TU München, 1997.
[41] S. Reitzinger, Algebraic multigrid and element preconditioning I, SFB-Report 98-15, University Linz, Austria, December 1998.
[42] A.A. Reusken, Multigrid with matrix-dependent transfer operators for a singular perturbation problem, Computing 50 (3) (1993) 199–211.
[43] A.A. Reusken, A multigrid method based on incomplete Gaussian elimination, Eindhoven University of Technology, Report RANA 95-13, ISSN 0926-4507, 1995.
[44] A.A. Reusken, On the approximate cyclic reduction preconditioner, Report 144, Institut für Geometrie und Praktische Mathematik, RWTH Aachen, 1997. SIAM J. Sci. Comput., to appear.
[45] A.A. Reusken, Approximate cyclic reduction preconditioning, in: W. Hackbusch, G. Wittum (Eds.), Multigrid Methods, Vol. 5, Lecture Notes in Computational Science and Engineering, Vol. 3, Springer, Berlin, 1998, pp. 243–259.
[46] G. Robinson, Parallel computational fluid dynamics on unstructured meshes using algebraic multigrid, in: R.B. Pelz, A. Ecer, J. Häuser (Eds.), Parallel Computational Fluid Dynamics, Vol. 92, Elsevier, Amsterdam, 1993.
[47] J.W. Ruge, K. Stüben, Efficient solution of finite difference and finite element equations by algebraic multigrid (AMG), in: D.J. Paddon, H. Holstein (Eds.), Multigrid Methods for Integral and Differential Equations, The Institute of Mathematics and its Applications Conference Series, New Series vol. 3, Clarendon Press, Oxford, 1985, pp. 169–212.
[48] J.W. Ruge, K. Stüben, Algebraic multigrid (AMG), in: S.F. McCormick (Ed.), Multigrid Methods, Frontiers in Applied Mathematics, Vol. 5, SIAM, Philadelphia, 1986.

[49] Y. Saad, ILUM: a multi-elimination ILU preconditioner for general sparse matrices, SIAM J. Sci. Comput. 17 (1996) 830–847.
[50] K. Stüben, Algebraic multigrid (AMG): Experiences and comparisons, Appl. Math. Comput. 13 (1983) 419–452.
[51] K. Stüben, Algebraic multigrid (AMG): an introduction with applications, in: U. Trottenberg, C.W. Oosterlee, A. Schüller (Eds.), Multigrid, Academic Press, New York, 2000. Also GMD Report 53, March 1999.
[52] P. Vanek, J. Mandel, M. Brezina, Algebraic multigrid on unstructured meshes, University of Colorado at Denver, UCD/CCM Report No. 34, 1994.
[53] P. Vanek, J. Mandel, M. Brezina, Algebraic multigrid by smoothed aggregation for second and fourth order elliptic problems, Computing 56 (1996) 179–196.
[54] P. Vanek, M. Brezina, J. Mandel, Convergence of algebraic multigrid based on smoothed aggregation, UCD/CCM Report 126, 1998. Numer. Math., submitted.
[55] C. Wagner, Introduction to algebraic multigrid, Course notes of an algebraic multigrid course at the University of Heidelberg in the Wintersemester 1998/99, http://www.iwr.uni-heidelberg.de/ ~ Christian.Wagner, 1999.
[56] C. Wagner, On the algebraic construction of multilevel transfer operators, IWR-Report, Universität Heidelberg, Computing (2000).
[57] C. Wagner, W. Kinzelbach, G. Wittum, Schur-complement multigrid – a robust method for groundwater flow and transport problems, Numer. Math. 75 (1997) 523–545.
[58] W.L. Wan, T.F. Chan, B. Smith, An energy minimization interpolation for robust multigrid methods, Department of Mathematics, UCLA, UCLA CAM Report 98-6, 1998.
[59] T. Washio, C.W. Oosterlee, Flexible multiple semicoarsening for 3D singularly perturbed problems, SIAM J. Sci. Comput. 19 (1998) 1646–1666.
[60] R. Webster, An algebraic multigrid solver for Navier–Stokes problems in the discrete second order approximation, Int. J. Numer. Methods Fluids 22 (1996) 1103–1123.
[61] P. Wesseling, C.W. Oosterlee, Geometric multigrid with applications to computational fluid dynamics, this volume, Comput. Appl. Math. 128 (2001) 311–334.
[62] L. Zaslavsky, An adaptive algebraic multigrid for multigroup neutron diffusion reactor core calculations, Appl. Math. Comput. 53 (1993) 13–26.
[63] L. Zaslavsky, An adaptive algebraic multigrid for reactor critically calculations, SIAM J. Sci. Comput. 16 (1995) 840–847.