Algebraic Multigrid (AMG): An Introduction with Applications

K. Stüben

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

e-mail: stueben@gmd.de

November 10, 1999

GMD-Report 70, November 1999

Abstrakt. Seit den frühen neunziger Jahren besteht ein stark wachsender Bedarf an effizienteren Methoden zur Lösung großer, dünnbesetzter und *unstrukturierter* linearer Gleichungssysteme. Klassische Lösungsverfahren sind für praktisch relevante Problemgrößen an ihre Grenzen gestoßen und neue *hierarchische* Verfahren mußten entwickelt werden, um die numerische Effizienz zu steigern. Dieses Paper gibt eine elementare Einführung in die erste, *rein matrix-orientierte* Methode dieser Art, die *algebraische Mehrgittermethode* (AMG). In dieser Methode wird die klassische Mehrgitteridee der Kombination von Glättungs- mit Grobgitterkorrekturprozessen auf gewisse Klassen von Matrixproblemen übertragen. Neben ihrer Robustheit und Effizienz besteht ein großer Vorteil dieser Methode etwa in ihrer direkten Anwendbarkeit zur Lösung verschiedener Typen elliptischer partieller Differentialgleichungen auf unstrukturierten, zwei- oder dreidimensionalen Gittern. Weil AMG keine geometrische Information ausnutzt, kann es unmittelbar auch zur Lösung von Problemen eingesetzt werden, die keinen direkten geometrischen Hintergrund besitzen, vorausgesetzt, die zugrundeliegenden Matrizen erfüllen gewisse Voraussetzungen. Obwohl der AMG-Ansatz bereits zu Beginn der achtziger Jahre entwickelt wurde, stellt er immer noch einen der effizientesten Ansätze zur algebraischen Lösung entsprechender Probleme dar. Allerdings sind zwischenzeitlich einige Modifikationen und Erweiterungen hinzugekommen. Neben einer Einführung in die theoretischen Grundlagen von AMG beschreiben wir einen konkreten Algorithmus und demonstrieren seine Robustheit und Effizienz am Beispiel verschiedener Anwendungen.

Abstract. Since the early nineties, there has been a strongly increasing demand for more efficient methods to solve large sparse and *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. The purpose of this paper is to give an elementary introduction to the first hierarchical and *purely matrix-based* approach, *algebraic multigrid* (AMG). The main idea behind AMG is to extend the classical ideas of geometric multigrid (smoothing and coarse-grid correction) to certain classes of algebraic systems of equations. Besides its robustness and efficiency, the main practical advantage of AMG is that it can directly be applied, for instance, to 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 satisfies certain properties. Although the development of AMG goes back to the early eighties, it still provides one of the most efficient algebraic methods to solve corresponding problems. Compared to the original approach, however, several modifications and extensions have been introduced. In addition to the theoretical basics of AMG, we present a concrete algorithm in some detail and demonstrate its robustness and efficiency by means of a variety of typical applications.

To appear: Guest contribution (appendix) in the book "Multigrid" by U. Trottenberg, C.W. Oosterlee, A. Schüller; Academic Press, to appear 1999.

Contents

List of Figures

List of Tables

1 Introduction

In contrast to geometrically based multigrid, algebraic multigrid (AMG) does not require a given problem to be defined on a grid but rather operates directly on (linear sparse) algebraic equations

$$Au = f \quad \text{or} \quad \sum_{j=1}^{n} a_{ij} u_j = f_i \quad (i = 1, 2, ..., n) \; . \tag{1}$$

If one replaces the terms *grids, subgrids* and *grid points* by *sets of variables, subsets of variables* and *single variables*, respectively, one can describe AMG in formally the same way as a geometric multigrid method. In particular, coarse-grid discretizations used in geometric multigrid to reduce low-frequency error components now correspond to certain matrix equations of reduced dimension. However, no multigrid[1] hierarchy needs to be known a priori. In fact, the construction of a (problem-dependent) hierarchy – including the coarsening process itself, the transfer operators as well as the coarse-grid operators – is part of the AMG algorithm, based solely on algebraic information contained in the given system of equations

Although the central ideas behind AMG and its range of applicability are more general, in this introductory paper, the focus is on the solution of *scalar* elliptic partial differential equations of second order. Moreover, we mostly consider *symmetric, positive (semi-) definite* problems. This is because AMG is best developed for such problems. Various recent research activities aim at the application of AMG to *systems* of partial differential equations (such as Navier-Stokes equations or structural mechanics problems). However, although important progress has been achieved for different types of systems, major research is still ongoing and there is no well-settled approach yet.

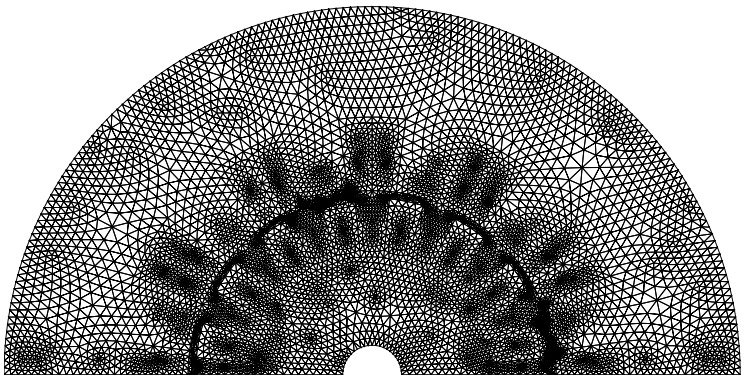


Figure 1: Unstructured finite element mesh

We will see that AMG provides very robust solution methods. However, the real practical advantage of AMG is that it can directly be applied to structured as well as unstructured grids (see Figure 1), in 2D as well as in 3D. In order to point out the similarities and differences of geometric and algebraic multigrid, we will first give a brief review of some major steps in the development of robust geometric approaches.

[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*.

1.1 Geometric multigrid

In the early days of multigrid, coarse-grid correction approaches were based on simple coarsening strategies (typically by doubling the mesh size in each spatial direction, that is, by $h \to 2h$ coarsening), straightforward geometric grid transfer operators (standard interpolation and restriction) and coarse-grid operators being natural analogs of the one given on the finest grid. Later on, it was realized that such simple "coarse-grid components" were not appropriate for various types of more complex problems such as diffusion equations with strongly varying or even discontinuous coefficients. The so-called *Galerkin operator* [32] was introduced as an alternative to the "natural" selection of the coarse-grid operators mentioned before. From a practical point of view, it is advantageous that this operator can be constructed purely algebraically. This makes it very convenient for the treatment of, for instance, differential operators with strongly varying coefficients. From a theoretical point of view, the major advantage of Galerkin-based coarse-grid correction processes is that they satisfy a variational principle (for symmetric and positive definite problems) which opened new perspectives for theoretical convergence investigations.

The introduction of *operator-dependent interpolation* [1, 85] – that is, interpolation which directly relies on the discretisation stencils – was equally important. Together with the Galerkin operator, this kind of interpolation allowed the treatment of larger classes of problems including problems with strongly discontinuous coefficients. The main trouble with such problems is that, after having applied a typical smoothing process (relaxation), the error is *not* geometrically smooth any longer: across discontinuities the smoothed error exhibits the same discontinuous behavior as the solution itself. Galerkin-based coarsening, however, requires interpolation which correctly operates on such error. While geometric interpolation (which can be accurately applied only to corrections with continuous first derivatives) does not correctly transfer such corrections to finer levels, the discretisation stencils themselves *do* reflect the discontinuities and, if used for interpolation, also correctly transfer the discontinuities. Galerkin-based coarse-grid correction processes with operator-dependent interpolation became increasingly popular since then.

All geometric multigrid approaches operate on pre-defined grid hierarchies. That is, the *coarsening process* itself is fixed and kept as simple as possible. Fixing the hierarchy, however, puts particular requirements on the smoothing properties of the smoother used in order to ensure an efficient interplay between smoothing and coarse-grid correction. Generally speaking, error components which cannot be corrected by appealing to a coarser-grid problem, must be effectively reduced by smoothing (and vice versa). For instance, assuming the coarser levels to be obtained by $h \to 2h$ coarsening, pointwise relaxation is very efficient for essentially isotropic problems. For anisotropic problems, however, pointwise relaxation exhibits good smoothing properties only "in the direction of strong couplings" (cf. Section 1.3). Consequently, more complex smoothers, such as alternating line-relaxation or ILU-type smoothers, are required in order to maintain fast multigrid convergence. Multigrid approaches for which the interplay between smoothing and coarse-grid correction works efficiently for large classes of problems are often called "robust".

While the implementation of efficient and robust smoothers was not difficult in 2D model situations, for 3D applications on complex meshes their realization tended to become rather cumbersome. For instance, the robust 3D analog of alternating line relaxation is alternating *plane* relaxation (realized by 2D multigrid within each plane) which, in complex

geometric situations, becomes very complicated, if possible at all. ILU smoothers, on the other hand, loose much of their smoothing property in general 3D situations.

It is therefore not surprising that a new trend arose which aimed at simplifying the smoother without sacrificing convergence. However, in order to maintain an efficient interplay between smoothing and coarse-grid correction, this required putting more effort into the coarse-grid correction process. More sophisticated coarsening techniques were developed, for example, employing more than one coarser grid on each level of the multigrid hierarchy such as the *multiple semi-coarsening* technique (semi-coarsening in multiple directions) [44, 82, 26, 43].

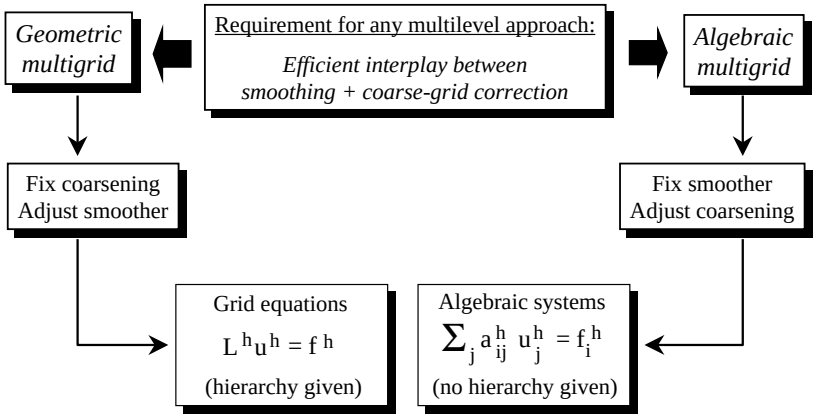


Figure 2: Geometric versus algebraic multigrid

1.2 Algebraic multigrid

Regarding the interplay between smoothing and coarse-grid correction, AMG can be regarded as the most radical attempt to maintain simple smoothers but still achieve robust convergence. Its development started in the early eighties [14, 15, 12] when Galerkin-based coarse-grid correction processes and, in particular, operator-dependent interpolation were introduced into geometric multigrid (see previous section). One of the motivations for AMG was the observation that reasonable operator-dependent interpolation and the Galerkin operator can be derived directly from the underlying matrices, without any reference to the grids. To some extent, this fact had already been exploited in the first "black-box" multigrid code [27]. However, regarding the selection of coarser levels, this code was still geometrically based. In a purely algebraic setting, the coarsening process itself also needs to be defined based only on information contained in the given matrix.

This leads to the most important conceptual difference between geometric and algebraic multigrid (cf. Figure 2). Geometric approaches employ fixed grid hierarchies and, therefore, an efficient interplay between smoothing and coarse-grid correction has to be ensured by selecting appropriate smoothing processes. In contrast to this, AMG fixes the smoother to some simple relaxation scheme such as plain point Gauss-Seidel relaxation, and enforces an efficient interplay with the coarse-grid correction by chosing the coarser levels and interpolation appropriately. Geometrically speaking, AMG attempts to coarsen

only in directions in which relaxation really smoothes the error for the problem at hand. However, since the relevant information is contained in the matrix itself (in terms of size and sign of coefficients), this process can be performed based only on matrix information, producing coarser levels which are *locally* adapted to the smoothing properties of the given smoother. The guiding principle in constructing the operator-dependent interpolation is to *force* its range to approximately contain those "functions" which are unaffected by relaxation. It will turn out that this is the crucial condition to obtain efficient coarse-grid correction processes.

The coarsening process is fully automatic. This automatism is the major reason for AMG's flexibility in adapting itself to specific requirements of the problem to be solved and is the main reason for its robustness in solving large classes of problems *despite using very simple point-wise smoothers*. There is no need for something like multiple semi-coarsened grids. Figure 3 visualises the hierarchy of grids created by AMG if applied to a diffusion equation discretised on the grid depicted in Figure 1. See Section 1.3 for an explanation of this type of picture and a more detailed example on AMG's coarsening strategy.

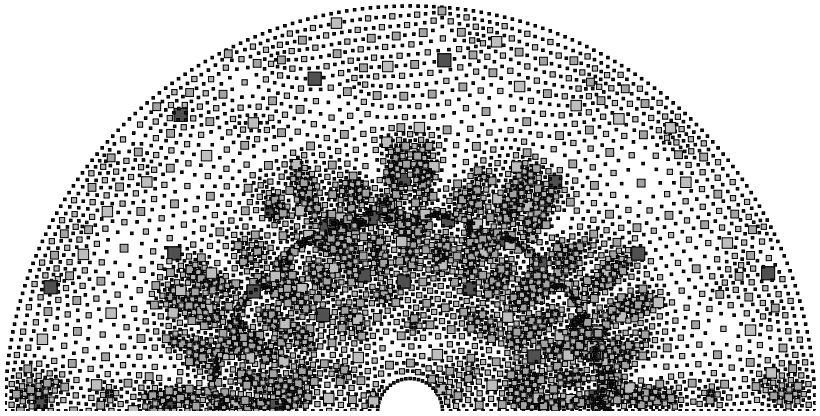


Figure 3: Standard AMG coarsening

The flexibility of AMG and its simplicity of use, of course, have a price: A *setup phase*, in which the given problem (1) is analysed, 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 will always be 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. 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 (as long as the given matrix satisfies certain conditions). That is, AMG provides an attractive multi-level variant whenever geometric multigrid is either too difficult to apply or cannot be used 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 see that AMG itself also provides a very efficient pre-conditioner. In fact, we will see that simplified AMG variants, used as pre-conditioner, are often better than more complex ones applied as stand-alone solver.

The first fairly general algebraic multigrid program is described and investigated in [62, 69, 63], see also [23]. Since the resulting code, AMG1R5, was made publically available in the mid eighties, there had been no substantial further research and development in algebraic multigrid for many years. However, since the early nineties, and even more since the mid nineties, there was a strong increase of interest in algebraically oriented multilevel methods. One reason for this is certainly the increasing geometrical complexity of applications which, technically, limits the immediate use of geometric multigrid. Another reason is the steadily increasing demand for efficient "plug-in" solvers. In particular, in commercial codes, this demand is driven by increasing problem sizes which clearly exhibit the limits of the classical one-level solvers which are still used in most packages. Millions of degrees of freedom in the underlying numerical models require hierarchical approaches for efficient solution and AMG provides a possibility to obtain such a solution without the need to completely re-structure existing software packages.

As a consequence of this development, there now exist various different algebraic approaches [70], all of which are hierarchical but some of which differ substantially from the original AMG ideas as outlined above. It is beyond the scope of this introduction to AMG to discuss all these approaches. For completeness, we will set pointers to the relevant literature in Section 10. This introduction stays close to the original AMG ideas as described in [63]. In particular, AMG as we understand it, is structurally completely analogous to standard multigrid methods in the sense that algorithmic components such as smoothing and coarse-grid correction play a role in AMG similar to the one they play in standard multigrid. Nevertheless, there is no unique AMG algorithm and one may think of various modifications and improvements in the concrete realization of AMG's coarsening strategy. We here refer to an approach which, by our experience, has turned out to be very flexible, robust and efficient in practice. It has been implemented in the code RAMG05[2], which is a successor of the original code AMG1R5. However, RAMG05 is completely new and, in particular, incorporates more efficient and more flexible interpolation and coarsening strategies.

1.3 An example

The flexibility of AMG in adjusting its coarsening process locally to the requirements of a given problem is demonstrated in Figure 4. The underlying problem is the differential equation

$$-(au_x)_x - (bu_y)_y + cu_{xy} = f(x,y) \tag{2}$$

defined on the unit square (with Dirichlet boundary conditions). We set $a = b = 1$ 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 $c = 2$.

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

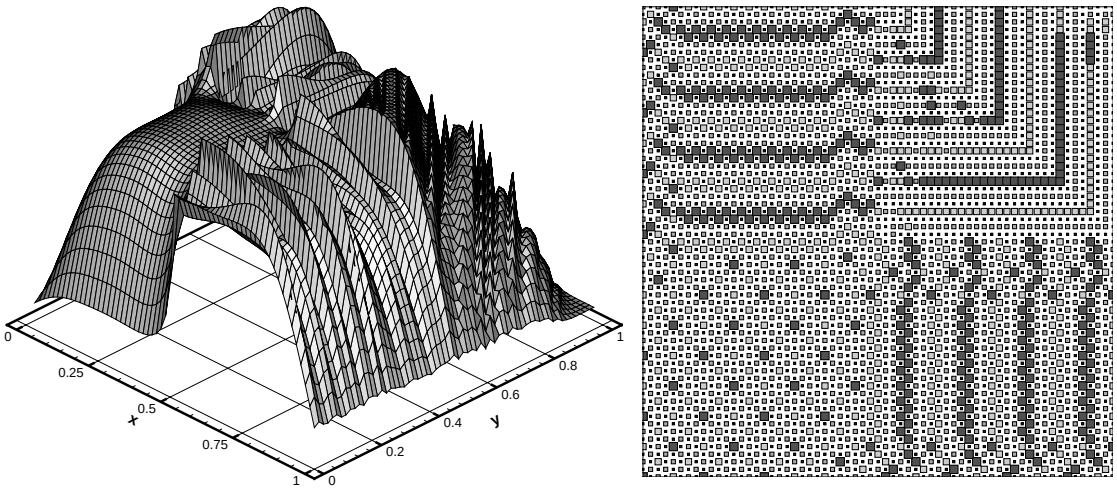


Figure 4: a) "Smooth" error in case of problem (2). b) The finest and three consecutive levels created by the standard AMG coarsening algorithm.

The diffusion part is discretized by the standard 5-point stencil and the mixed derivative by the (left-oriented) 7-point stencil (147). The resulting discrete system is isotropic in the lower left quarter of the unit square but strongly anisotropic in the remaining quarters. In the upper left and lower right quarters we have strong connections in the y- and x-directions, respectively. In the upper right quarter strong connectivity is in the diagonal direction. Figure 4a shows what a "smooth" error looks like on the finest level after having applied a few point relaxation steps to the homogeneous problem, 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 error can only be effectively reduced by means of a coarser grid if that grid is obtained by essentially coarsening in directions in which the error really changes smoothly in the geometric sense and if interpolation treats the discontinuities correctly. Indeed, as outlined before, 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.

Figure 4b depicts the finest and three consecutive grids created by using standard AMG coarsening and interpolation (cf. Section 7). 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 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 "coinci-

dence" 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.

1.4 Overview of the paper

The intention of this paper is to give an elementary, self-contained introduction to an algebraic multigrid approach which is suited, in particular, for the treatment of large classes of scalar elliptic differential equations and problems whose matrices have a similar structure. Although the theoretical considerations are in the framework of positive definite problems, the algorithm presented does not exploit symmetry and can, to some extent, also be applied to certain non-symmetric and indefinite problems.

We assume the reader to have some basic knowledge in standard (geometric) multigrid. In particular, he should be familiar with the basic principles – smoothing and coarse-grid correction – and with the recursive definition of multigrid cycles (such as V- or F-cycles). This is because, for simplicity, we limit all our descriptions to just two levels. Accordingly, whenever we talk about efficiency of a particular approach, we implicitly always assume the underlying two-level approach to be recursively extended to full cycles. (Clearly, a mere two-level method is hardly ever practical.)

Section 2 describes our notation and contains basic theoretical aspects. In particular, it summarizes well-known properties of Galerkin-based coarse-grid correction approaches and shows that AMG, in certain limit cases, degenerates to direct solvers (Section 2.3). Although, generally, these direct solvers are extremely inefficient in terms of computational work and memory requirement, they can be approximated by more realistic (iterative) approaches in various ways, indicating the formal generality of the approach.

The AMG method as efficiently used in practice, is largely heuristically motivated. However, under certain assumptions, in particular symmetry and positive definiteness, a two-level theory is available showing that convergence can be expected to be independent of the size of the problem and as fast (and expensive) as we wish. Actually, the *convergence* of AMG is not generally the problem (in fact, AMG can always be *forced* to converge rapidly) but rather the tradeoff between convergence and numerical work which is also directly related to the memory requirements. Note that this is, in a sense, just opposite to standard multigrid approaches where the numerical work per cycle is known and controlable but the convergence may not be satisfactory.

This paper covers both theoretical and practical aspects. While the theoretical investigations are contained in Sections 3-6, practical aspects (a concrete algorithm and a discussion of its performance for various types of problems) are presented in Sections 7-8. Although the theoretical results form the basis for the details of the algorithm presented, we have tried to keep the practically oriented sections as independent from the theoretical sections as possible. Readers not interested in the theory may thus decide to skip the corresponding sections. Whenever necessary, we will make reference to relevant theoretical aspects.

In Section 3, we first introduce the basic concept of algebraic smoothness [12]. This will be used in Section 4 to prove the convergence of two-level methods using *post-smoothing*. While the main approach is the same as in [12, 63], the definition of interpolation has been modified and extended. The case of *pre-smoothing* is considered in Section 5. In

both cases, it turns out that it is crucial to define coarsening and interpolation so that the "interpolation error", in some algebraic sense, is uniformly bounded. No realistic extension of the two-level theory to complete V-cycles is available yet (cf. Section 6). Moreover, while the underlying AMG code has successfully been applied also to various non-symmetric problems, there is no comparable theory yet for the non-symmetric case.

The algorithm used in the code RAMG05 mentioned above is described in some detail in Section 7. Although one can imagine several modifications and improvements, the approach presented has turned out to be very flexible, robust and efficient in practice. Various applications and a discussion of RAMG05's performance are presented in Section 8. We investigate both standard model cases as well as some industrially relevant cases, for instance, from computational fluid dynamics.

Section 9 outlines so-called "aggregation-based" AMG variants and points out their relation to the "standard" approach considered in the other parts of this paper. Finally, in Section 10, we summarize important further developments and draw some conclusions. Although we try to cover the most important references, the list is certainly not complete in this rapidly developing field of research.

Remark 1.1 Many considerations in the theoretical parts of this paper refer to a given matrix A. However, it should be clear that we are not really interested in, for instance, convergence estimates for one particular A only but rather in having *uniform* convergence if A ranges over some reasonable *class* of matrices, $\mathcal{A}$. A typical class is the class consisting of all M-matrices. However, a reasonable class may also consist of the discretization matrices of a particular elliptic differential equation discretized on a series of grids with mesh size $h \to 0$. Uniform convergence for $A \in \mathcal{A}$ then means that AMG convergence does not depend on the mesh size (a typical property of geometric multigrid methods). In this sense, we sometimes say that convergence does not depend on the *size of the matrix*. $\ll$

Remark 1.2 All results which refer to positive definite matrices carry over also to the semi-definite zero rowsum case. One just has to exclude the constant vectors from all considerations. Of course, besides treating the coarsest level properly, it is then crucial to transfer constants exactly between levels. Since this will be ensured by all interpolation processes discussed, we will not discuss the semi-definite case explicitly any further. $\ll$

Acknowledgements: I would like to thank A. Brandt, K. Witsch, R. Lorentz and A. Krechel for various discussions concerning specific aspects of this paper. I also would like to point out that significant parts of this paper rely on the early work [63] which was the result of a close cooperation with A. Brandt, S. McCormick, and J. Ruge. Finally, I would like to thank Computational Dynamics Ltd., London, for funding substantial parts of the work on AMG.

2 Theoretical basis and notation

As mentioned in the introduction, AMG is based on the Galerkin approach. The formal structure of AMG, with all its components, is described in Section 2.1; some additional notation is contained in Section 2.2. The remainder of the section summarizes fundamental aspects related to Galerkin-based coarse-grid correction processes. This includes the discussion of certain limit cases in Section 2.3 for which AMG degenerates to a direct solver. Since these (unrealistic) limit cases are presented mainly for reasons of motivation – in particular, to indicate the formal generality of the overall approach – this section may well be skipped in first reading. Section 2.4, which re-calls the variational principle of Galerkin-based coarse-grid correction processes for symmetric and positive definite matrices A, is more important for the concrete approaches investigated in this paper.

2.1 Formal AMG components

Since the recursive extension of any two-level process to a real multi-level process is formally straightforward, we describe the components of AMG only on the basis of two-level methods with indices h and H distinguishing the fine and coarse level, respectively. In particular, we re-write (1) as

$$A_h u^h = f^h \quad \text{or} \quad \sum_{j \in \Omega^h} a_{ij}^h u_j^h = f_i^h \quad (i \in \Omega^h) \tag{3}$$

with Ω^h denoting the index set $\{1, 2, ..., n\}$. We implicitly assume always that A_h corresponds to a *sparse* matrix. The particular indices, h and H, have been chosen to have a formal similarity to geometric two-grid descriptions. In general, they are not related to a discretization parameter.

In order to derive a coarse-level system from (3), we first need a splitting of Ω^h 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)*. Assuming such a splitting to be given and defining $\Omega^H = C^h$, coarse-level AMG systems,

$$A_H u^H = f^H \quad \text{or} \quad \sum_{l \in \Omega^H} a_{kl}^H u_l^H = f_k^H \quad (k \in \Omega^H) \ , \tag{4}$$

will be constructed based on the *Galerkin principle*, i.e. the matrix A_H is defined as the *Galerkin operator*

$$A_H := I_h^H \, A_h \, I_H^h \tag{5}$$

where I_H^h and I_h^H denote *interpolation* (or *prolongation*) and *restriction operators*, respectively, mapping coarse-level vectors into fine-level ones and vice-versa. We always assume both operators to have *full rank*.

Finally, as with any multigrid method, we need a *smoothing process* with a corresponding linear *smoothing operator* S_h. That is, one smoothing step is of the form

$$u^h \longrightarrow \overline{u}^h \quad \text{where} \quad \overline{u}^h = S_h u^h + (I_h - S_h) A_h^{-1} f^h \tag{6}$$

(I_h denotes the identity operator). Consequently, the error $e^h = u_\star^h - u^h$ ($u_\star^h$ denotes the exact solution of (3)) is transformed according to

$$e^h \longrightarrow \overline{e}^h \quad \text{where} \quad \overline{e}^h = S_h e^h \ . \tag{7}$$

Note that we normally use the letter u for *solution* quantities and the letter e for *correction* or *error* quantities.

As already mentioned before, AMG employs simple smoothing processes. In this introduction to AMG, we consider only plain *Gauss-Seidel relaxation* (i.e. $S_h = (I_h - Q_h^{-1} A_h)$ with Q_h being the lower triangular part of A_h, including the diagonal) or ω-*Jacobi-relaxation* (i.e. $S_h = I_h - \omega D_h^{-1} A_h$ with $D_h = diag(A_h)$). Clearly, unless A_h is positive definite (which we assume most of the time), the use of such variable-wise relaxation methods implicitly requires additional assumptions on A_h, in particular, its diagonal elements should be sufficiently large compared to the off-diagonal elements.

For completeness, we want to mention that, particularly in connection with theoretical investigations, we also consider *partial* relaxation steps, namely, Gauss-Seidel and Jacobi relaxation applied only to F-variables (with frozen values for the C-variables). Since such partial relaxation will then formally play the role of smoothing, we will refer to it as *F-smoothing*. Note, however, that F-smoothing by itself has no real smoothing properties in the usual sense.

Remark 2.1 The coarse-level system (4) formally plays the same role as coarse-grid correction equations in geometric multigrid. In particular, f^H and u^H actually correspond to *residuals* and *corrections*, respectively. More precisely, one two-level correction step is defined as

$$u_{new}^h = u_{old}^h + I_H^h e^H \quad \text{where} \quad A_H e^H = I_h^H (r_{old}^h) = I_h^H (f^h - A_h u_{old}^h) \ . \tag{8}$$

For the corresponding errors, this means

$$e_{new}^h = K_{h,H}\, e_{old}^h \quad \text{with} \quad K_{h,H} := I_h \ - \ I_H^h A_H^{-1} I_h^H A_h \ , \tag{9}$$

$K_{h,H}$ being the so-called *coarse-grid correction operator*. Consequently, error reduction by one complete two-grid iteration step – including ν_1 and ν_2 pre- and post-smoothing steps, respectively – is described by the *two-grid iteration operator* (cf. (7)):

$$e_{new}^h = M_{h,H}\, e_{old}^h \quad \text{with} \quad M_{h,H}(\nu_1, \nu_2) = S_h^{\nu_2}\, K_{h,H}\, S_h^{\nu_1} \ . \qquad \ll$$

Summarising, what needs to be *explicitly constructed* in order to formally set up a two-level (and by recursive application a multi-level) process, are the C/F-splitting and the transfer operators I_H^h and I_h^H. The construction of these components – which forms the major task of AMG's setup phase – are closely related processes and, whenever we talk about transfer operators, we implicitly always assume a suitable C/F-splitting to be given. These components need to be selected so that an efficient interplay between smoothing and coarse-grid correction – and consequently good convergence – is achieved. It is equally important that the splitting and the transfer operators are such that A_H is still reasonably sparse and much smaller than A_h. In Section 7, we will describe a practical algorithm.

From a more theoretical point of view, we consider AMG components in Sections 3-5.

Except for Section 2.3, the theoretical parts of this paper refer to *symmetric* and *positive definite* matrices A_h for which we always define the restriction as the transpose of interpolation,

$$I_h^H = (I_H^h)^T \ . \tag{10}$$

It then immediately follows that A_H is also symmetric and positive definite, independent of the concrete choice of I_H^h (as long as it has full rank):

$$(A_H u^H, u^H)_E = (I_h^H A_h I_H^h u^H, u^H)_E = (A_h I_H^h u^H, I_H^h u^H)_E = (u^H, A_H u^H)_E$$

where $(.,.)_E$ denotes the *Euclidian inner product*. (Unless explicitly stated otherwise, the terms symmetric and positive definite as well as the transpose of a matrix always refer to the Euclidian inner product.) Moreover, the coarse-grid correction operator $K_{h,H}$ turns out to satisfy a *variational principle* (cf. Section 2.4).

We finally want to mention that all interpolations $e^h = I_H^h e^H$ considered in this paper are of the form

$$e_i^h = (I_H^h e^H)_i = \begin{cases} e_i^H & \text{if } i \in C^h \\ \sum_{k \in P_i^h} w_{ik}^h e_k^H & \text{if } i \in F^h \end{cases} \tag{11}$$

where $P_i^h \subset C^h$ is called the set of *interpolatory variables*. Clearly, for reasons of efficiency, P_i should be a reasonably small subset of C-variables "near" i. Note that any such interpolation has *full rank*.

Remark 2.2 According to the above description, we regard the set of coarse-level variables as a subset of the fine-level ones. In particular, (11) expresses that the coarse-level correction e_k^H is used to directly correct the corresponding fine-level variable, u_k^h. Note that this is formally different from algebraic multigrid approaches based on "aggregation" (see, e.g., [73, 10, 19, 39]). However, we will see in Section 9 that aggregation-type approaches can be regarded as a special case of the approach considered here. $\ll$

2.2 Further notation

AMG is set up in an algebraic environment. However, rather than using vector-matrix terminology, it is often convenient to formally stick to the grid terminology by introducing *fictitious grids* with grid points being simply the nodes of the directed graph which can be associated with the given matrix. In this sense, we identify each $i \in \Omega^h$ with a *point* and define connections between points in the sense of the associated graph. That is, point $i \in \Omega^h$ is defined to be (directly) *coupled* (or *connected*) to point $j \in \Omega^h$ if $a_{ij}^h \neq 0$. Correspondingly, we define the (direct) *neighborhood* of a point i by

$$N_i^h = \{j \in \Omega^h : j \neq i,\ a_{ij}^h \neq 0\} \quad (i \in \Omega^h) \ . \tag{12}$$

Referring to a point $i \in \Omega^h$ means nothing else than referring to the variable u_i^h. Using grid terminology, we can formally interpret the equations $A_h u^h = f^h$ as *grid equations* on

the fictitious *grid* Ω^h. Analogously, coarser level equations $A_H u^H = f^H$ can be interpreted as grid equations on *subgrids* $\Omega^H \subset \Omega^h$.

In the course of this paper, we will use both grid and vector-matrix terminology whichever is more convenient for the given purpose. Moreover, we will usually omit the indices h and H, writing, for instance, A, e, C and K instead of A_h, e^h, C^h and $K_{h,H}$, respectively. We actually use these indices only if we explicitly need to distinguish two consecutive levels.

For theoretical investigations, it is often convenient to assume vectors and matrices to be re-ordered so that, w.r.t. a given C/F-splitting, the set of equations (3) 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 inter-grid transfer operators are then written as

$$I_H^h = \begin{pmatrix} I_{FC} \\ I_{CC} \end{pmatrix} , \quad I_h^H = (I_{CF}, I_{CC}) \tag{14}$$

with I_{CC} being the identity operator. Instead of $e^h = I_H^h e^H$ and (11) we simply write $e_F = I_{FC} e_C$ and

$$e_i = \sum_{k \in P_i} w_{ik} e_k \quad (i \in F) \ , \tag{15}$$

respectively. This is for simplicity and should not lead to any confusion.

We finally list some more specific notation. The *range* and *null space* of any operator Q are denoted by $\mathcal{R}(Q)$ and $\mathcal{N}(Q)$, respectively. For any square matrix Q, its *spectral radius* is denoted by $\rho(Q)$. At several places in this paper, we make use of the fact that, for any two matrices, Q_1 and Q_2, we have

$$\rho(Q_1 Q_2) = \rho(Q_2 Q_1) \ . \tag{16}$$

We write $A > 0$ if A is symmetric and positive definite. Correspondingly, $A > B$ stands for $A - B > 0$. For vectors, $u > 0$ and $u \geq 0$ mean that the corresponding inequalities hold componentwise.

If $A_h > 0$, we use the following three inner products in addition to the Euclidian one:

$$(u,v)_0 = (D_h u, v)_E , \quad (u,v)_1 = (A_h u, v)_E \quad \text{and} \quad (u,v)_2 = (D_h^{-1} A_h u, A_h v)_E , \tag{17}$$

along with their associated norms $\|.\|_i$ $(i = 0, 1, 2)$. Here, $D_h = diag(A_h)$. $(.,.)_1$ is the so-called *energy inner product* and $\|.\|_1$ the *energy norm*. Moreover, given any C/F-splitting, we will use the analogs of the first two inner products applied to A_{FF} (13) instead of A_h,

$$(u_F, v_F)_{0,F} = (D_{FF} u_F, v_F)_E \quad \text{and} \quad (u_F, v_F)_{1,F} = (A_{FF} u_F, v_F)_E \ , \tag{18}$$

and the associated norms $\|.\|_{i,F}$ $(i = 0, 1)$ where $D_{FF} = diag(A_{FF})$. (Note that A_{FF} is positive definite.)

Important parts of our theoretical discussion refer to the model class of *symmetric M-matrices*, where a symmetric matrix is defined to be an M-matrix if it is *positive definite*

and *off-diagonally non-positive*. Such matrices often arise from second order discretizations of scalar elliptic differential equations. If a matrix A contains both negative and positive off-diagonal entries, we use the notation

$$a_{ij}^{-} = \begin{cases} a_{ij} & (\text{if } a_{ij} < 0) \\ 0 & (\text{if } a_{ij} \geq 0) \end{cases} \quad \text{and} \quad a_{ij}^{+} = \begin{cases} 0 & (\text{if } a_{ij} \leq 0) \\ a_{ij} & (\text{if } a_{ij} > 0) \end{cases} \quad . \tag{19}$$

Correspondingly, we write

$$N_i^{-} = \{ j \in N_i : a_{ij}^h < 0 \} \quad \text{and} \quad N_i^{+} = \{ j \in N_i : a_{ij}^h > 0 \} \; . \tag{20}$$

2.3 Limit case of direct solvers

In this section, we will see that, for very specific (impractical) definitions of the smoothing and transfer operators, the two-level methods corresponding to pre- and post-smoothing degenerate to direct solvers, that is, we have either $K_{h,H} S_h = 0$ or $S_h K_{h,H} = 0$. This is true under the mere assumption that A_h is *non-singular*. In order to show this, let us first state some basic properties of the coarse-grid correction operator, $K_{h,H}$. The transfer operators I_H^h and I_h^H are required to have full rank but are not required to be the transpose of each other.

Lemma 2.1 *Let A_h be any non-singular matrix and assume the C/F-splitting and the transfer operators to be given such that A_H^{-1} exists. We then have*

$$K_{h,H} I_H^h \, e^H \equiv 0 \; , \quad K_{h,H}^2 = K_{h,H} \quad \text{and} \quad I_h^H A_h K_{h,H} \, e^h \equiv 0$$

which implies $\mathcal{N}(K_{h,H}) = \mathcal{R}(I_H^h)$ and $\mathcal{R}(K_{h,H}) = \mathcal{N}(I_h^H A_h)$. Consequently, given any smoothing operator S_h, the following holds:

$$K_{h,H} \, S_h = 0 \Longleftrightarrow \mathcal{R}(S_h) \subseteq \mathcal{R}(I_H^h) \quad \text{and} \quad S_h \, K_{h,H} = 0 \Longleftrightarrow \mathcal{N}(I_h^H A_h) \subseteq \mathcal{N}(S_h) \; .$$

Proof: All statements are immediate consequences of the fact that A_H is the Galerkin operator (5). For instance, the first identity holds because of

$$K_{h,H} I_H^h = I_H^h \; - \; I_H^h A_H^{-1} I_h^H A_h I_H^h = I_H^h - I_H^h = 0$$

which, in turn, implies $\mathcal{N}(K_{h,H}) \supseteq \mathcal{R}(I_H^h)$. The reverse relation, $\mathcal{N}(K_{h,H}) \subseteq \mathcal{R}(I_H^h)$, follows directly from the definition (9) of $K_{h,H}$. The proof of the remaining statements is similarly straightforward. $\triangle$

In the following, we use the notation (13) and (14) and, just for the purpose of this section, define a very specific "smoothing process" as follows:

$$u \longrightarrow \overline{u} \quad \text{where} \quad A_{FF} \, \overline{u}_F \, + \, A_{FC} \, u_C \, = f_F \, , \quad \overline{u}_C = u_C \; . \tag{21}$$

(Although this is not a practical smoothing process, we formally stick to the standard multigrid terminology.) In terms of the error, $e = u^\star - u$, this means

$$e \longrightarrow \overline{e} \quad \text{where} \quad A_{FF} \, \overline{e}_F \, + \, A_{FC} \, e_C \, = 0 \, , \quad \overline{e}_C = e_C \tag{22}$$

and the "smoothing operator" is seen to be

$$\widehat{S}_h = \begin{pmatrix} 0 & -A_{FF}^{-1}A_{FC} \\ 0 & I_{CC} \end{pmatrix} \tag{23}$$

which has the properties

$$\mathcal{R}(\widehat{S}_h) = \{e \,:\, e_F = -A_{FF}^{-1}A_{FC}\, e_C\} \quad \text{and} \quad \mathcal{N}(\widehat{S}_h) = \{e \,:\, e_C = 0\}\,. \tag{24}$$

In addition, we define very specific transfer operators by

$$\widehat{I}_{FC} = -A_{FF}^{-1}A_{FC} \quad \text{and} \quad \widehat{I}_{CF} = -A_{CF}A_{FF}^{-1}\,. \tag{25}$$

We then obtain the following theorem:

Theorem 2.1 [37] *Let A_h be non-singular and let a C/F-splitting be given such that A_{FF}^{-1} exists. Furthermore, use (21) as "smoothing process". Then the following statements hold:*

1. *For $I_{FC} = \widehat{I}_{FC}$ and arbitrary I_{CF}, A_H^{-1} exists and $K_{h,H}\widehat{S}_h = 0$.*
2. *For $I_{CF} = \widehat{I}_{CF}$ and arbitrary I_{FC}, A_H^{-1} exists and $\widehat{S}_h K_{h,H} = 0$.*
3. *In either of the two cases, the Galerkin operator (5) is just the Schur complement corresponding to (13), that is, $A_H = C_H$ where*

$$C_H := A_{CC} - A_{CF}A_{FF}^{-1}A_{FC}\,. \tag{26}$$

Proof: If $I_{FC} = \widehat{I}_{FC}$, a straightforward computation shows that, independent of I_{CF}, the Galerkin operator equals the Schur complement:

$$\begin{aligned} A_H = I_h^H A_h I_H^h &= \begin{pmatrix} I_{CF}\,, & I_{CC} \end{pmatrix} \begin{pmatrix} A_{FF} & A_{FC} \\ A_{CF} & A_{CC} \end{pmatrix} \begin{pmatrix} -A_{FF}^{-1}A_{FC} \\ I_{CC} \end{pmatrix} \\ &= \begin{pmatrix} I_{CF}\,, & I_{CC} \end{pmatrix} \begin{pmatrix} 0 \\ A_{CC} - A_{CF}A_{FF}^{-1}A_{FC} \end{pmatrix} = C_H\,. \end{aligned}$$

Since both A_h and A_{FF} are assumed to be non-singular, C_H is also non-singular. Hence, A_H^{-1} exists. By definition, we have $\mathcal{R}(I_H^h) = \mathcal{R}(\widehat{S}_h)$ which, according to Lemma 2.1, implies $K_{h,H}\widehat{S}_h = 0$. Regarding the second statement, one can see by analogous arguments as above that A_H equals the Schur complement also in this case. Because of

$$I_h^H A_h = \begin{pmatrix} -A_{CF}A_{FF}^{-1}\,, & I_{CC} \end{pmatrix} \begin{pmatrix} A_{FF} & A_{FC} \\ A_{CF} & A_{CC} \end{pmatrix} = \begin{pmatrix} 0\,, & A_H \end{pmatrix}$$

we have $I_h^H A_h e = A_H e_C$ for all e. Hence, $\mathcal{N}(I_h^H A_h) = \{e \,:\, e_C = 0\} = \mathcal{N}(\widehat{S}_h)$ which, according to Lemma 2.1, implies $\widehat{S}_h K_{h,H} = 0$. $\triangle$

According to the theorem, only one of the transfer operators has to be explicitly defined in order to obtain a direct method. For the two-level method to be a direct solver independent of whether pre- or post-smoothing is used, both operators have to be specified accordingly.

Remark 2.3 Note that interpolation $e_F = \widehat{I}_{FC} e_C$ is defined by exactly solving the homogeneous F-equations

$$A_{FF}\, e_F \;+\; A_{FC}\, e_C \;= 0\;. \tag{27}$$

That is, interpolation and "smoothing" (22) are based on the same set of equations. Note furthermore that, for *symmetric* matrices A_h, we have $\widehat{I}_{CF} = \widehat{I}_{FC}^T$, that is, the restriction is just the transpose of interpolation. In contrast to this, for *non-symmetric* matrices, $\widehat{I}_{CF} = \widetilde{I}_{FC}^T$ where $\widetilde{I}_{FC} = -(A_{FF}^{-1})^T A_{CF}^T$ which is related to solving

$$A_{FF}^T e_F + A_{CF}^T e_C = 0 \tag{28}$$

instead of (27). Thus, $\widehat{I}_{CF}$ is just the transpose of another interpolation, namely, the one corresponding to A_h^T. ≪

The specific two-level approaches defined above can be extended to full V-cycles in a straightforward way by recursively applying the same strategy to the coarse-level Galerkin problems (4). Assuming the coarsest level equations to be solved exactly, the resulting V-cycles then also converge in just one iteration step. However, such cycles are far from being practical, the obvious reason being that A_{FF}^{-1} is involved in computing both the smoothing and the transfer operators. Generally, the explicit computation of A_{FF}^{-1} is much too expensive and, moreover, a recursive application in a multi-level context would be prohibitive due to fill-in on coarser levels.

Of course, the complexity of the matrix A_{FF} strongly depends on the selected C/F-splitting, and by just choosing the splitting appropriately, one may *force* A_{FF} to become simple and easy to invert. For instance, on each level, the splitting can be selected so that all corresponding matrices A_{FF} simply become *diagonal* (assuming non-zero diagonal entries). In some exceptional situations, this indeed leads to an efficient method. For instance, if A_h corresponds to a tridiagonal matrix, the resulting V-cycle can easily be seen to coincide with the well-known method of *total reduction* [68]. Nevertheless, in general, the resulting method will still become extremely inefficient: although the selection of such special C/F-splittings often makes sense in constructing the *second* level, further coarsening rapidly becomes extremely slow causing the corresponding Galerkin matrices to become dense very quickly. This is illustrated in the following example.

Example 2.1 Consider any standard 5-point discretization on a rectangular mesh, for instance, the 5-point discretization of the 2D Poisson equation. Then, obviously, A_{FF} becomes diagonal if we select the C/F-splitting so that, for each $i \in F$, *all* of its neighbors are in C, that is, if we select *red-black coarsening*, yielding a grid coarsening ratio of 0.5. The coarse-grid operator on the second level (consisting of the black points, say) can be seen to correspond to 9-point stencils. That is, although the reduction of points is substantial, the overall size of the second-level matrix is still close to the finest-level one. Proceeding analogously in creating the third level, will now reduce the grid size only by a factor of 3/4. At the same time, the Galerkin operator grows further: the largest matrix rows on level 3 correspond to 21-point stencils. Clearly, continuing this process will lead to a completely impractical coarsening. For corresponding 3D problems, the situation is even more dramatic. ≪

The above V-cycles actually correspond to specific variants of Gauss elimination rather than real multigrid processes. Clearly, in order to obtain more practical *iterative* approaches, the explicit inversion of A_{FF} has to be avoided. From the multigrid point of view, it is most natural to approximate the operators $\widehat{I}_{FC}$ and $\widehat{S}_h$ by more realistic interpolation and smoothing operators, I_{FC} and S_h, respectively (and similarly $\widehat{I}_{CF}$ by some I_{CF} if A_h is non-symmetric). According to Remark 2.3, all $e \in \mathcal{R}(I_H^h)$ and all $e \in \mathcal{R}(S_h)$ should approximately satisfy equation (27). We do not want to quantify this here any further but rather refer to Sections 4 and 5.

2.4 The variational principle for positive definite problems

In the following, we summarize basic properties of Galerkin-based coarse-grid correction processes in case of symmetric, positive definite matrices A_h. For symmetric matrices, we always assume (10). We have already mentioned before that the Galerkin operator A_H (5) is then also symmetric and positive definite. Moreover, a variational principle for the coarse-grid correction operator $K_{h,H}$ (9) is implied (see the last statement of Corollary 2.1 below) which simplifies theoretical investigations substantially. This principle follows from well-known facts about orthogonal projectors which, for completeness, are summarized in the following theorem.

Theorem 2.2 *Let (.,.) be any inner product with corresponding norm $\|.\|$ and let the matrix Q be symmetric w.r.t. (.,.). Furthermore, let $Q^2 = Q$. Then Q is an orthogonal projector. That is, we have*

1. $\mathcal{R}(Q) \perp \mathcal{R}(I-Q)$
2. *For $u \in \mathcal{R}(Q)$ and $v \in \mathcal{R}(I-Q)$ we have $\|u+v\|^2 = \|u\|^2 + \|v\|^2$*
3. $\|Q\| = 1$
4. *for all u: $\|Qu\| = \min_{v \in \mathcal{R}(I-Q)} \|u - v\|$.*

Proof: The first statement follows immediately since Q is symmetric and $Q^2 = Q$:

$$(Qu, (I-Q)v) = (u, Q(I-Q)v) = (u, 0) = 0\,.$$

This, in turn, implies the second statement. Regarding the third statement, we obtain by decomposing $u = Qu + (I-Q)u$,

$$\|Q\|^2 = \sup_{u \neq 0} \frac{\|Qu\|^2}{\|u\|^2} = \sup_{u \neq 0} \frac{\|Qu\|^2}{\|Qu\|^2 + \|(I-Q)u\|^2} \leq 1$$

which shows $\|Q\| \leq 1$. Selecting any $u \in \mathcal{R}(Q)$, proves that $\|Q\| = 1$. Regarding the last statement, again by decomposing u as before, we obtain

$$\begin{aligned} \min_{v \in \mathcal{R}(I-Q)} \|u - v\|^2 &= \min_{\cdots} \|Qu + (I-Q)u - v\|^2 = \min_{\cdots} \|Qu - v\|^2 \\ &= \min_{\cdots} (\|Qu\|^2 + \|v\|^2) = \|Qu\|^2\,. \end{aligned}$$

△

To apply this theorem to $K_{h,H}$, we observe that $A_h K_{h,H}$ corresponds to a symmetric matrix, that is, $K_{h,H}$ itself is symmetric w.r.t. the *energy inner product* (17):

$$(K_{h,H}u^h, v^h)_1 = (A_h K_{h,H}u^h, v^h)_E = (u^h, A_h K_{h,H}v^h)_E = (u^h, K_{h,H}v^h)_1 \,.$$

Since we also have $K_{h,H}^2 = K_{h,H}$ (see Lemma 2.1), $K_{h,H}$ is an orthogonal projector. By finally observing that

$$\mathcal{R}(I_h - K_{h,H}) = \mathcal{R}(I_H^h) \,, \tag{29}$$

we obtain the following corollary:

Corollary 2.1 *Let $A_h > 0$ and let any C/F-splitting and any full rank interpolation I_H^h be given. Then the coarse-level correction operator $K_{h,H}$ is an orthogonal projector w.r.t. the energy inner product $(.,.)_1$. In particular, we have:*

1. $\mathcal{R}(K_{h,H}) \perp_1 \mathcal{R}(I_H^h)$, *i.e.,* $(A_h K_{h,H}u^h, I_H^h v^H)_E = 0$ *for all* u^h, v^H.
2. *For* $u^h \in \mathcal{R}(K_{h,H})$ *and* $v^h \in \mathcal{R}(I_H^h)$ *we have* $\|u^h + v^h\|_1^2 = \|u^h\|_1^2 + \|v^h\|_1^2$
3. $\|K_{h,H}\|_1 = 1$
4. *for all* e^h: $\|K_{h,H}e^h\|_1 = \min_{e^H} \|e^h - I_H^h e^H\|_1$.

The last statement of the corollary expresses the variational principle mentioned above: Galerkin-based coarse-grid corrections minimize the *energy norm* of the error w.r.t. all variations in $\mathcal{R}(I_H^h)$. As a trivial consequence, a two-level method can never diverge if the smoother satisfies $\|S_h\|_1 \leq 1$ (e.g., Gauss-Seidel relaxation or ω-Jacobi relaxation with a suitably selected under-relaxation parameter ω). That this holds also for complete V-cycles, assuming *any* hierarchy of C/F-splittings and (full rank) interpolation operators to be given, follows immediately by a recursive application (replacing exact coarse-grid corrections by V-cycle approximations with zero initial guess) of the following lemma:

Lemma 2.2 *Let the exact coarse-level correction e^H in (8) be replaced by any approximation $\widetilde{e}^H$ satisfying $\|e^H - \widetilde{e}^H\|_1 \leq \|e^H\|_1$ (where $\|\,.\,\|_1$ is taken w.r.t. A_H). Then the approximate two-level correction operator still satisfies $\|\widetilde{K}_{h,H}\|_1 \leq 1$.*

Proof: For the approximate two-level correction operator

$$\widetilde{K}_{h,H}\, e^h = e^h - I_H^h \widetilde{e}^H = K_{h,H}\, e^h + I_H^h(e^H - \widetilde{e}^H)$$

we obtain

$$\|\widetilde{K}_{h,H}\, e^h\|_1^2 = \|K_{h,H}\, e^h\|_1^2 + \|I_H^h(e^H - \widetilde{e}^H)\|_1^2 \,.$$

Since $\|I_H^h v^H\|_1 = \|v^H\|_1$ holds for all v^H, we have

$$\|I_H^h(e^H - \widetilde{e}^H)\|_1^2 = \|e^H - \widetilde{e}^H\|_1^2 \leq \|e^H\|_1^2 = \|I_H^h\, e^H\|_1^2 \,.$$

Hence,

$$\|\widetilde{K}_{h,H}\, e^h\|_1^2 \leq \|K_{h,H}\, e^h\|_1^2 + \|I_H^h\, e^H\|_1^2 = \|e^h\|_1^2$$

and, therefore, $\|\widetilde{K}_{h,H}\|_1 \leq 1$. $\triangle$

Although this does not say anything about the efficiency of a V-cycle, from a practical point of view, it ensures at least some kind of minimum robustness. Based on the properties in Corollary 2.1, one can easily formulate concrete conditions which imply V-cycle convergence at a rate which is independent of the size of A_h (see, for example, Theorem 3.1 in [63]). Since, unfortunately, these conditions are not suited for the explicit construction of realistic AMG processes, we here just refer to related discussions in [63].

Remark 2.4 In our final algorithm (see Section 7), we will employ certain truncation mechanisms in order to limit the growth of the Galerkin operators towards increasingly coarser levels. According to the variational principle and the above remarks, the truncation of *interpolation* (before computing the corresponding Galerkin operator) 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 themselves may be dangerous since this violates the validity of the variational principle and, if not applied with great care, may cause strong divergence in practice. $\ll$

3 Algebraic smoothing

In algebraic multigrid, smoothing and coarse-grid correction play formally the same role as in geometric multigrid. However, the meaning of the term "smooth" is different:

- In a geometric environment, the term "smooth" is normally used in a restrictive (viz. the "natural") way. Moreover, in the context of multigrid, an error is regarded as smooth only if it *can be* approximated on some pre-defined coarser level. That is, smoothness in geometric multigrid has always to be seen relative to a coarser level. For example, an error may be smooth with respect to a semi-coarsened grid but not with respect to a standard $h \to 2h$ coarsened grid. Correspondingly, the "smoothing property" of a given smoother always involves two consecutive levels.
- In contrast to this, in algebraic multigrid, there are no pre-defined grids and a smoothing property in the geometric sense becomes meaningless. Instead, we *define* an error e to be *algebraically smooth* if it is slow to converge with respect to S_h, that is, if $S_h e \approx e$. In other words, we call an error "smooth" if it *has to be* approximated by means of a coarser level (which then needs to be properly constructed) in order to speed up convergence. From an algebraic point of view, this is the important point in distinguishing smooth and non-smooth errors.

In this section, assuming A_h to be symmetric and positive definite ($A_h > 0$), we will consider algebraic smoothing by relaxation and introduce a concept [12] of how to characterize it. For typical types of matrices, we give some heuristic interpretation of algebraic smoothness which is helpful in finally constructing the coarsening and interpolation.

3.1 Basic norms and smooth eigenvectors

In investigating the smoothing properties of relaxation, we use the inner products and norms defined in (17). The following lemma (omitting the index h) summarizes some basic relations which will be needed later. Note that we can assume $\rho(D^{-1}A)$ to be uniformly bounded for all important classes $\mathcal{A}$ of matrices under consideration.

Lemma 3.1 *Let $A > 0$. Then the following inequalities hold for all e:*

$$\|e\|_1^2 \le \|e\|_0 \|e\|_2 , \quad \|e\|_2^2 \le \rho(D^{-1}A) \|e\|_1^2 , \quad \|e\|_1^2 \le \rho(D^{-1}A) \|e\|_0^2 . \tag{30}$$

Applying these norms to the eigenvectors of $D^{-1}A$, we have

$$D^{-1}A\,\phi = \lambda\phi \quad \Longrightarrow \quad \|\phi\|_2^2 = \lambda \|\phi\|_1^2 \quad \textit{and} \quad \|\phi\|_1^2 = \lambda \|\phi\|_0^2 . \tag{31}$$

Proof: The first inequality in (30) follows from Schwarz' inequality:

$$\|e\|_1^2 = (Ae, e)_E = (D^{-\frac{1}{2}}Ae, D^{\frac{1}{2}}e)_E \le \|e\|_2 \|e\|_0 .$$

The other inequalities follow from the equivalence

$$(B_1 e, e)_E \le c\,(B_2 e, e)_E \quad \Longleftrightarrow \quad \rho(B_2^{-1}B_1) \le c \tag{32}$$

which holds for all $B_1 > 0$ and $B_2 > 0$. The verification of (31) is straightforward. $\triangle$

Remark 3.1 The eigenvectors ϕ of $D^{-1}A$ play a special role. In particular, eigenvectors corresponding to the *smallest* eigenvalues λ are those which typically cause slowest convergence of relaxation and, therefore, correspond to what we defined as an algebraically smooth error. This can most easily be verified for ω-Jacobi relaxation (using proper under-relaxation) by observing that small λ's correspond to eigenvalues of the ω-Jacobi iteration operator $S = (I - \omega D^{-1}A)$ close to one. This is also true for related schemes such as Gauss-Seidel relaxation, but is not so easy to see.

Clearly, for all relevant applications, we can assume the smallest eigenvalues λ to approach zero (otherwise, standard relaxation methods converge rapidly on their own and no multi-level improvement is required). For instance, for standard elliptic problems of second order, discretized on a square grid with mesh size h, the smallest eigenvalues satisfy $\lambda = O(h^2)$ and, for isotropic (e.g. Poisson-like) problems, correspond to just those eigenfunctions ϕ which are very smooth geometrically in all spatial directions. On the other hand, the largest eigenvalues satisfy $\lambda = O(1)$ and correspond to geometrically non-smooth eigenvectors. For an illustration, see Example 3.1 below.

Generally, however, whether or not slow-to-converge error really corresponds to geometrically smooth error (assuming a geometric background to exist), depends on A (see Example 3.2 in the next section). $\ll$

Example 3.1 To illustrate the previous remark, consider the matrix A which corresponds to the Poisson operator, discretized on the unit square with mesh size $h = 1/N$,

$$\frac{1}{h^2}\begin{bmatrix} & -1 & \\ -1 & 4 & -1 \\ & -1 & \end{bmatrix}_h . \tag{33}$$

Assuming Dirichlet boundary conditions, the eigenvalues and eigenfunctions of $D^{-1}A = \frac{h^2}{4}A$ are known to be

$$\lambda_{n,m} = (2 - \cos n\pi h - \cos m\pi h)/2 \quad \text{and} \quad \phi_{n,m} = \sin(n\pi x)\sin(m\pi y) \tag{34}$$

where $n, m = 1, 2, ..., N-1$. Obviously, we have

$$\lambda_{min} = \lambda_{1,1} = 1 - \cos\pi h = O(h^2) \quad \text{and} \quad \lambda_{max} = \lambda_{N-1,N-1} = 1 + \cos\pi h = O(1) ,$$

corresponding to the lowest and highest frequency eigenfunctions, respectively,

$$\phi_{min} = \sin(\pi x)\sin(\pi y) \quad \text{and} \quad \phi_{max} = \sin((N-1)\pi x)\sin((N-1)\pi y) . \qquad \ll$$

The previous discussion on λ and the corresponding eigenvectors ϕ of $D^{-1}A$, together with the relations (31), motivate the significance of the above norms, in particular, in the context of algebraic smoothing: if applied to a slow-to-converge error $e = \phi$ (λ close to zero), all three norms are largely different in size,

$$\|\phi\|_2 \ll \|\phi\|_1 \quad \text{and} \quad \|\phi\|_1 \ll \|\phi\|_0 . \tag{35}$$

On the other hand, if applied to algebraically non-smooth error, all three norms are comparable in size. This different behavior makes it possible to identify slow-to-converge error by simply comparing different norms and gives rise to the characterization of algebraic smoothness in the next section.

3.2 Smoothing property of relaxation

We say that a relaxation operator S satisfies the *smoothing property* w.r.t. a matrix $A > 0$ if

$$\|Se\|_1^2 \leq \|e\|_1^2 - \sigma \|e\|_2^2 \quad (\sigma > 0) \tag{36}$$

holds with σ being independent of e. This implies that S is efficient in reducing the error e as long as $\|e\|_2$ is relatively large compared to $\|e\|_1$. However, it will generally become very inefficient if $\|e\|_2 \ll \|e\|_1$. In accordance with the motivations given before, such error is called *algebraically smooth*. We say that S satisfies the smoothing property w.r.t. a class $\mathcal{A}$ of matrices if (36) holds *uniformly* for all $A \in \mathcal{A}$, that is, with the same σ.

Below, we will show that Gauss-Seidel and ω-Jacobi relaxation satisfy (36) uniformly for all matrices which are of interest here. Before, however, we want to make some further remarks on algebraic smoothness.

Remark 3.2 Note that $\sigma \|e\|_2^2 \leq \|e\|_1^2$ is necessary for (36) to hold which, because of (32), is equivalent to $\rho(D^{-1}A) \leq 1/\sigma$. Consequently, a necessary condition for (36) to hold uniformly for all $A \in \mathcal{A}$ is the uniform boundedness of $\rho(D^{-1}A)$ in $\mathcal{A}$ which, as mentioned before, is satisfied for all important classes $\mathcal{A}$ under consideration. $\ll$

We have already indicated that the term "algebraically smooth" in the above sense is not necessarily related to what is called smooth in a geometric environment. In order to illustrate this, we give two examples.

Example 3.2 As an extreme case, consider the matrix $\widehat{A} > 0$ which corresponds to the (somewhat artificial) stencil

$$\frac{1}{h^2} \begin{bmatrix} & 1 & \\ 1 & 4 & 1 \\ & 1 & \end{bmatrix}_h \tag{37}$$

with Dirichlet boundary conditions and $h = 1/N$. That is, $\widehat{A}$ is similar to A in Example 3.1 (Poisson equation) except that the sign of all off-diagonal entries has changed from negative to positive. As a consequence of this, compared to the Poisson case, the role of geometrically smooth and non-smooth error is completely interchanged: algebraically smooth error is actually highly oscillatory geometrically and algebraically non-smooth error is very smooth geometrically. In order to see this more clearly, observe that

$$\widehat{A} = -(A - cI) \quad \text{with} \quad c = 8/h^2 \ .$$

That is, the eigenvalues and eigenfunctions of $\widehat{D}^{-1}\widehat{A} = \frac{h^2}{4}\widehat{A}$ are directly related to those of $D^{-1}A = \frac{h^2}{4}A$ (cf. Example 3.1), namely,

$$\widehat{\lambda}_{n,m} = -\lambda_{n,m} + 2 \quad \text{and} \quad \widehat{\phi}_{n,m} = \phi_{n,m}$$

where $n, m = 1, 2, ..., N-1$. A straightforward computation shows that the smallest and largest eigenvalues of $\widehat{D}^{-1}\widehat{A}$ and $D^{-1}A$ are the same,

$$\widehat{\lambda}_{min} = \widehat{\lambda}_{N-1,N-1} = -\lambda_{max} + 2 = \lambda_{min} \quad \text{and} \quad \widehat{\lambda}_{max} = \widehat{\lambda}_{1,1} = -\lambda_{min} + 2 = \lambda_{max} \ ,$$

but the corresponding eigenfunctions are interchanged. $\ll$

Example 3.3 For certain matrices, there is no algebraically smooth error at all. For instance, assume A to be strongly diagonally dominant, that is, $a_{ii} - \sum_{j \neq i} |a_{ij}| \geq \delta a_{ii}$ with $\delta > 0$. The latter immediately implies $\rho(A^{-1}D) \leq 1/\delta$ which, because of (32), is equivalent to $\|e\|_2^2 \geq \delta \|e\|_1^2$ for all e. That is, if δ is of significant size, there is no algebraically smooth error. Clearly, such cases are not really interesting here since they do not require any multi-level improvement. In fact, (36) implies rapid convergence for all e. In the following, we will tacitly exclude such cases. $\ll$

As seen from the above considerations, the term "smooth" is sometimes misleading and should better be replaced by, for instance, "slow-to-converge". However, for historical reasons, we stick to the term "smooth".

The following lemma is used for proving the subsequent theorems which refer to the smoothing properties of Gauss-Seidel and Jacobi relaxation, respectively.

Lemma 3.2 [63] *Let $A > 0$ and let the smoothing operator be of the form $S = I - Q^{-1}A$ with some non-singular matrix Q. Then the smoothing property (36) is equivalent to*

$$\sigma\, Q^T D^{-1} Q \leq Q + Q^T - A\,.$$

Proof: Using the particular form of S, a straightforward calculation shows

$$\|Se\|_1^2 = \|e\|_1^2 - ((Q + Q^T - A)Q^{-1}Ae, Q^{-1}Ae)_E\,.$$

Hence, (36) is equivalent to

$$\sigma\, \|e\|_2^2 \leq ((Q + Q^T - A)Q^{-1}Ae, Q^{-1}Ae)_E$$

which, in turn, is equivalent to

$$\sigma\, (D^{-1}Qe, Qe)_E \leq ((Q + Q^T - A)e, e)_E\,. \qquad \triangle$$

Theorem 3.1 [12, 63] *Let $A > 0$ and define, with any vector $w = (w_i) > 0$,*

$$\gamma_- = \max_i \Big\{ \frac{1}{w_i a_{ii}} \sum_{j<i} w_j |a_{ij}| \Big\}\,, \quad \gamma_+ = \max_i \Big\{ \frac{1}{w_i a_{ii}} \sum_{j>i} w_j |a_{ij}| \Big\}\,.$$

Then Gauss-Seidel relaxation satisfies (36) with $\sigma = 1/(1+\gamma_-)(1+\gamma_+)$.

Proof: Gauss-Seidel relaxation satisfies the assumptions of Lemma 3.2 with Q being the lower triangular part of A (including the diagonal) and we have $Q + Q^T - A = D$. Thus, (36) is equivalent to $\sigma\, (Q^T D^{-1} Qe, e)_E \leq (De, e)_E$ which, because of (32), is equivalent to $\sigma \leq 1/\rho(D^{-1}Q^T D^{-1}Q)$. A sufficient condition for the latter inequality is given by

$$\sigma \leq 1/|D^{-1}Q^T|\,|D^{-1}Q|$$

where $|\,.\,|$ stands for an arbitrary matrix norm which is induced by a vector norm (i.e., which is the corresponding operator norm). For the special choice

$$|L| = |L|_w = \max_i \Big\{ \frac{1}{w_i} \sum_j w_j |l_{ij}| \Big\} \tag{38}$$

we have $|D^{-1}Q| = 1 + \gamma_-$ and $|D^{-1}Q^T| = 1 + \gamma_+$ which proves the theorem. $\triangle$

From this theorem we conclude that Gauss-Seidel relaxation satisfies the smoothing property uniformly for all important classes $\mathcal{A}$ of matrices under consideration:

- For all *symmetric M-matrices*, the smoothing property is satisfied with $\sigma = 1/4$. This can be seen by observing that, for any such matrix, there exists a vector $z > 0$ with $Az > 0$ [67]. By chosing $w = z$ in Theorem 3.1, we obtain

$$\gamma_- = \max_i \Big\{ \frac{1}{z_i a_{ii}} \sum_{j<i} z_j |a_{ij}| \Big\} = \max_i \Big\{ 1 - \frac{1}{z_i a_{ii}} \sum_{j \le i} z_j a_{ij} \Big\} < 1 \,.$$

  Similarly, we obtain $\gamma_+ < 1$.

- The previous result, trivially, carries over to all $A > 0$ which are obtained from a symmetric M-matrix by symmetrically flipping some or all off-diagonal signs.

- For any $A > 0$ with $\le \ell$ non-vanishing entries per row, the smoothing property is satisfied with $\sigma = 1/\ell^2$. This can be seen by selecting $w_i = 1/\sqrt{a_{ii}}$. Because of $a_{ij}^2 < a_{ii} a_{jj}$ $(j \neq i)$, it follows that $\gamma_-, \gamma_+ < \ell - 1$.

- From a practical point of view, the previous result is far too pessimistic. We typically have $\sum_{j \neq i} |a_{ij}| \approx a_{ii}$, which means that, by selecting $w_i \equiv 1$, we can expect γ_- and γ_+ to be close to or even less than 1. That is, $\sigma \approx 1/4$ is typical for most applications we have in mind here.

Theorem 3.2 [12, 63] *Let $A > 0$ and $\eta \ge \rho(D^{-1}A)$. Then Jacobi relaxation with relaxation parameter $0 < \omega < 2/\eta$ satisfies (36) with $\sigma = \omega(2 - \omega\eta)$. In terms of η, the optimal parameter (which gives the largest value of σ) is $\omega^\star = 1/\eta$. For this optimal parameter, the smoothing property is satisfied with $\sigma = 1/\eta$.*

Proof: Jacobi relaxation satisfies the assumptions of Lemma 3.2 with $Q = \frac{1}{\omega} D$. Hence, (36) is equivalent to $(Ae, e)_E \le (2/\omega - \sigma/\omega^2)\,(De, e)_E$ which, because of (32), is equivalent to $\rho(D^{-1}A) \le 2/\omega - \sigma/\omega^2$. Replacing $\rho(D^{-1}A)$ by the upper bound η, leads to the sufficient condition $\eta \le 2/\omega - \sigma/\omega^2$, or, in terms of σ, $\sigma \le \omega(2 - \omega\eta)$. Obviously, σ is positive if $0 < \omega < 2/\eta$. This proves the theorem. $\triangle$

This theorem shows that Jacobi relaxation has smoothing properties similar to Gauss-Seidel relaxation. However, like in geometric multigrid, some relaxation parameter, ω, is required. Using $\eta = |D^{-1}A|_w$ as an upper bound for $\rho(D^{-1}A)$ (see (38)), one obtains, for instance, $\eta = 2$ for all symmetric M-matrices. More generally, for all typical scalar PDE applications satisfying $\sum_{j \neq i} |a_{ij}| \approx a_{ii}$ we have $\eta \approx 2$. That is, using the relaxation parameter $\omega = 1/2$, we have $\sigma \approx 1/2$.

We finally want to mention that Gauss-Seidel and ω-Jacobi relaxation also satisfy the following variant of the smoothing property (36),

$$\|Se\|_1^2 \leq \|e\|_1^2 - \widetilde{\sigma}\,\|Se\|_2^2 \quad (\widetilde{\sigma} > 0)\ . \tag{39}$$

Regarding the proof, we refer to [63]. Further discussions on smoothing properties of different relaxation schemes can be found in [12].

3.3 Interpretation of algebraically smooth error

We have seen in the previous section that, in the sense of (36), Gauss-Seidel and ω-Jacobi relaxation have smoothing properties for all matrices $A > 0$ under consideration. This smoothness needs to be exploited in order to finally construct reasonable C/F-splittings and interpolation (see Section 4.2). Therefore, in this section, we (heuristically) interprete algebraic smoothness for some typical cases.

Algebraically smooth error is characterized by $Se \approx e$ which, according to (36), implies $\|e\|_2 \ll \|e\|_1$ (see also (35)). In terms of the residual, $r = Ae$, this means

$$(D^{-1}r, r)_E \ll (e, r)_E$$

which indicates that, on the average, algebraically smooth error is characterized by (scaled) residuals which are much smaller than the error itself. This can also be seen directly. For instance, Gauss-Seidel relaxation, performed at point i, corresponds to replacing u_i by $\overline{u}_i$ where

$$\overline{u}_i = \frac{1}{a_{ii}}\,(f_i - \sum_{j\neq i} a_{ij}u_j) = \frac{1}{a_{ii}}\,(a_{ii}u_i + f_i - \sum_{j} a_{ij}u_j) = u_i + \frac{r_i}{a_{ii}}$$

or, in terms of the corresponding error,

$$\overline{e}_i = e_i - \frac{r_i}{a_{ii}}\,.$$

Here, r_i denotes the residual *before* relaxation at point i. From this we can heuristically conclude that, for algebraically smooth error (i.e. $\overline{e}_i \approx e_i$),

$$|r_i| \ll a_{ii}\,|e_i|\,.$$

That is, although the error may still be quite large globally, locally we can approximate e_i as a function of its neighboring error values e_j by evaluating

$$(r_i =)\ \ a_{ii}e_i + \sum_{j\in N_i} a_{ij}e_j = 0\,. \tag{40}$$

In this sense, algebraically smooth error provides some rough approximation to the solution of the basic equations (27).

The fact that (scaled) residuals are much smaller than the errors themselves, is, algebraically, the most important characteristic of smooth error. However, for some specific classes of matrices, we can give algebraic smoothness a more intuitive interpretation.

3.3.1 M-matrices

An algebraically smooth error e satisfies $\|e\|_2 \ll \|e\|_1$ which, because of the first inequality in (30), implies $\|e\|_1 \ll \|e\|_0$ (see also (35)) or, equivalently,

$$\frac{1}{2}\sum_{i,j}(-a_{ij})(e_i - e_j)^2 + \sum_i s_i e_i^2 \ll \sum_i a_{ii}\, e_i^2 \,. \tag{41}$$

This follows immediately from the equality

$$\|e\|_1^2 = (Ae,e)_E = \sum_{i,j} a_{ij}\, e_i\, e_j = \frac{1}{2}\sum_{i,j}(-a_{ij})(e_i - e_j)^2 + \sum_i s_i e_i^2 \tag{42}$$

which can easily be seen to hold for all symmetric matrices A. Here and in the sequel, $s_i = \sum_j a_{ij}$ denotes the i-th row sum of A.

For symmetric M-matrices (see Section 2.2), we have $a_{ij} \le 0$ $(j \ne i)$ and, in the most important case of $s_i \approx 0$, (41) means that, on the average for each i,

$$\sum_{j\ne i} \frac{|a_{ij}|}{a_{ii}} \frac{(e_i - e_j)^2}{e_i^2} \ll 1 \,. \tag{43}$$

That is, *algebraically smooth error varies slowly in the direction of large (negative) connections*, i.e., from e_i to e_j if $|a_{ij}|/a_{ii}$ is relatively large. In other words, relaxation schemes which satisfy the smoothing property (36), smooth the error along *strong (negative) connections*.

Example 3.4 The most typical example which illustrates the previous statement is given by matrices derived from the model operator $-\varepsilon u_{xx} - u_{yy}$, discretized on a uniform mesh. While, for $\varepsilon \approx 1$, algebraically smooth error changes slowly in both spatial directions, for $\varepsilon \ll 1$ (the anisotropic case) this is true only for the y-direction (cf. Figure 4 in Section 1.3 for an example with varying directions of anisotropies).

Another example is illustrated in Figure 5. Here A is derived by discretizing

$$-(\varepsilon u_x)_x - (\varepsilon u_y)_y = f(x,y) \tag{44}$$

on the unit square with mesh size h and using Dirichlet boundary conditions. The coefficient function ε is piecewise constant and defined as indicated in Figure 5a. Using standard 5-point differencing (for the definition, see Section 8.4.1), we obtain uniform stencils away from the interface of discontinuity and, consequently, in this area, algebraically smooth error changes smoothly in both coordinate directions. At the interface itself, however, the discretization stencil (depicted in Figure 5a) clearly shows that the inner subsquare is virtually decoupled from the rest of the domain: ε_{out} is negligible compared to ε_{in}. Consequently, the error inside the subsquare is unaffected by the error in the rest of the domain and we cannot expect an algebraically smooth error to change smoothly across the interface. In fact, it generally exhibits a sharp discontinuity. This is depicted in Figure 5b which shows a typical algebraically smooth error obtained after the application of a few Gauss-Seidel relaxation steps to the homogeneous equations (44). $\ll$

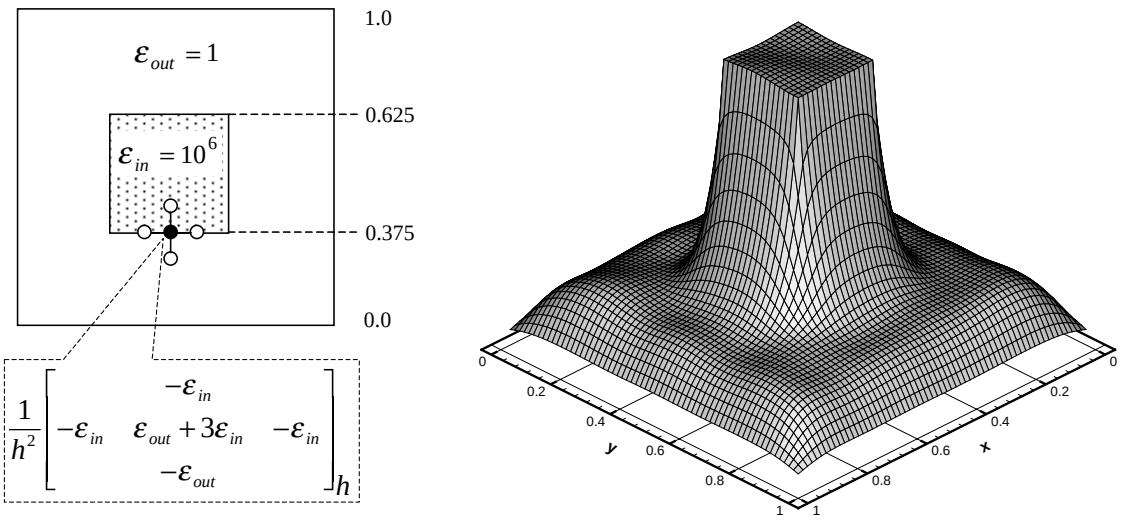


Figure 5: a) Coefficient ε for problem (44) and discretization stencil at the inner interface. b) Algebraically smooth error obtained after a few Gauss-Seidel relaxation steps.

3.3.2 Essentially positive-type matrices

A positive definite matrix is called of *essentially positive type* [12] if there exists a constant $c > 0$ such that, for all e,

$$\sum_{i,j}(-a_{ij})(e_i - e_j)^2 \geq c \sum_{i,j}(-a_{ij}^-)(e_i - e_j)^2 \,. \tag{45}$$

(Here and further below, we make use of the notation a_{ij}^- and a_{ij}^+ as defined in (19).) The main conclusion for M-matrices carries over to essentially positive type matrices. In particular, instead of (41), algebraically smooth error satisfies

$$\frac{c}{2}\sum_{i,j}(-a_{ij}^-)(e_i - e_j)^2 + \sum_i s_i e_i^2 \ll \sum_i a_{ii}\, e_i^2 \tag{46}$$

which still leads to the conclusion that an algebraically smooth error varies slowly in the direction of large (negative) connections.

Higher order difference approximations to second order elliptic problems or problems involving mixed derivatives often lead to essentially positive type matrices. Such and similar matrices have the property that, for each $a_{ij} > 0$, there exist paths of length two (or more) from i to j corresponding to relatively large *negative* connections (we call such paths "strong negative paths"). For instance, we may have $a_{ik} < 0$ and $a_{kj} < 0$ with $|a_{ik}|$, $|a_{kj}|$ being sufficiently large compared to a_{ij}. In such cases, (45) can explicitly be verified by using simple estimates like

$$\frac{\alpha\beta}{\alpha+\beta}(a+b)^2 \leq \alpha a^2 + \beta b^2 \quad (\alpha, \beta > 0) \,. \tag{47}$$

Example 3.5 Ignoring boundary conditions, the 4-th order discretization of $-\Delta u$ leads to the stencil

$$\frac{1}{12h^2}\begin{pmatrix} & & 1 & & \\ & & -16 & & \\ 1 & -16 & 60 & -16 & 1 \\ & & -16 & & \\ & & 1 & & \end{pmatrix}.$$

Using (47) with $\alpha = \beta = 1$, one can easily verify (45) with $c = 3/4$. Similarly, the 9-point discretization of $-\Delta u + u_{xy}$,

$$\frac{1}{h^2}\begin{pmatrix} -\frac{1}{4} & -1 & +\frac{1}{4} \\ -1 & 4 & -1 \\ +\frac{1}{4} & -1 & -\frac{1}{4} \end{pmatrix}, \tag{48}$$

satisfies (45) with $c = 1/2$. ≪

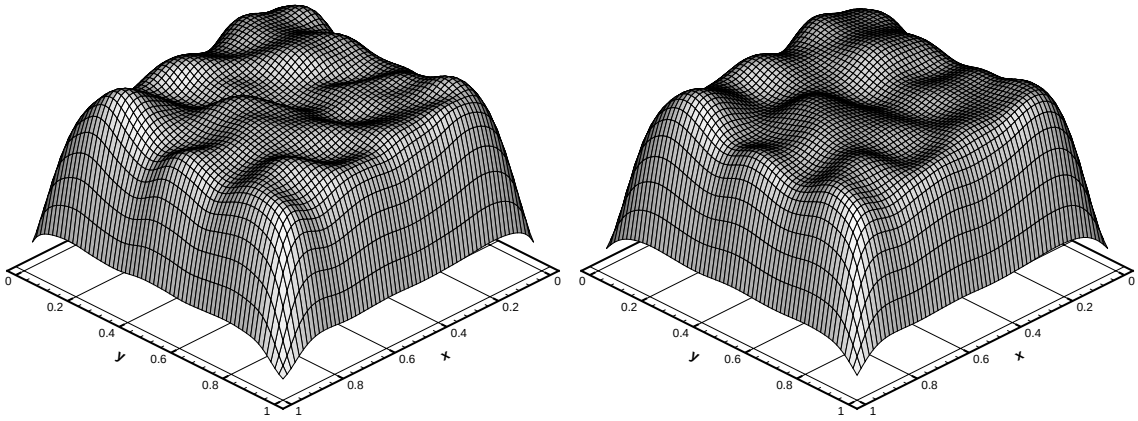


Figure 6: Algebraically smooth error in case of (48) and the standard 5-point Poisson stencil (33), respectively

Figure 6 compares algebraically smooth error corresponding to (48) with that corresponding to the standard 5-point Poisson stencil (33), obtained after the same number of Gauss-Seidel relaxation steps and starting with the same random function. The result is virtually the same, indicating that the positive matrix entries do not significantly influence the smoothing behavior of Gauss-Seidel.

Remark 3.3 Note that, for an essentially positive type matrix, each row containing off-diagonal elements has at least one negative off-diagonal entry. For the k-th row, this follows immediately by applying (45) to the special vector $e = (e_i)$ with $e_i = \delta_{ik}$ (Kronecker symbol). ≪

3.3.3 Large positive connections

For essentially positive type matrices, positive off-diagonal entries are relatively small. If there exist strong negative paths as in the previous examples, algebraically smooth error

still varies slowly *even in the direction of positive connections*. However, this cannot be expected to be true any more if positive connections exceed a certain size, in particular not, if $a_{ij} > 0$ and there exist *no* strong negative paths from i to j. We demonstrate this for matrices $A > 0$ which are close to being weakly diagonally dominant [33].

To characterize algebraically smooth error analogously as before, observe first that $s_i = t_i + 2\sum_{j\neq i} a_{ij}^+$ where $t_i := a_{ii} - \sum_{j\neq i} |a_{ij}|$. Using this, one can evaluate (42) further:

$$\begin{aligned}
(Ae,e)_E &= \frac{1}{2}\sum_{i,j} |a_{ij}^-|(e_i-e_j)^2 - \frac{1}{2}\sum_{i,j} a_{ij}^+(e_i-e_j)^2 + \sum_i s_i e_i^2 \\
&= \frac{1}{2}\sum_{i,j} |a_{ij}^-|(e_i-e_j)^2 + \sum_i \sum_{j\neq i} a_{ij}^+(2e_i^2 - (e_i-e_j)^2/2) + \sum_i t_i e_i^2 \\
&= \frac{1}{2}\sum_{i,j} |a_{ij}^-|(e_i-e_j)^2 + \frac{1}{2}\sum_i \sum_{j\neq i} a_{ij}^+(2e_i^2 + 2e_j^2 - (e_i-e_j)^2) + \sum_i t_i e_i^2 \\
&= \frac{1}{2}\sum_i \Big(\sum_{j\neq i} |a_{ij}^-|(e_i-e_j)^2 + \sum_{j\neq i} a_{ij}^+(e_i+e_j)^2 \Big) + \sum_i t_i e_i^2 \ . \qquad (49)
\end{aligned}$$

Assuming $t_i \approx 0$ (approximate weak diagonal dominance), $\|e\|_1 \ll \|e\|_0$ now leads to the conclusion that, on the average for each i, algebraically smooth error satisfies

$$\sum_{j\neq i} \frac{|a_{ij}^-|}{a_{ii}} \frac{(e_i-e_j)^2}{e_i^2} + \sum_{j\neq i} \frac{a_{ij}^+}{a_{ii}} \frac{(e_i+e_j)^2}{e_i^2} \ll 1 \qquad (50)$$

instead of (43).

Consequently, as before, algebraically smooth error can be expected to change slowly in the direction of strong *negative* directions. However, e_j tends to approximate $-e_i$ (relative to the size of e_i) if a_{ij} is *positive* and a_{ij}/a_{ii} is relatively large. In other words, algebraically smooth error tends to oscillate along strong positive connections.

Example 3.6 If A corresponds to the following stencil

$$\begin{bmatrix} & +1 & \\ -1 & 4 & -1 \\ & +1 & \end{bmatrix} , \qquad (51)$$

algebraically smooth error is geometrically smooth only in x-direction but strongly oscillatory in y-direction. This is depicted in Figure 7 (left picture). Note that the situation here is completely different from the anisotropic case $-u_{xx} - \varepsilon u_{yy}$ with $\varepsilon \ll 1$. While, in the latter case, the error between any two horizontal gridlines ($y \equiv const$) is virtually unrelated, it is strongly related in case of (51). According to the oscillatory behavior in y-direction, the error is actually rather smooth globally, if one considers only ever other horizontal gridline.

As an example which is more typical for differential problems, consider the standard discretization of Poisson equation with *anti-periodic* boundary conditions. Compared

to the corresponding *periodic* case (only negative off-diagonal entries), here certain off-diagonal entries have changed sign near the boundary. As a consequence, algebraically smooth error will generally exhibit a jump across the boundary. This is illustrated in Figure 7 (right picture). Note that AMG will not be able to detect the boundary: all equations with positive off-diagonals look like interior equations to AMG. ≪

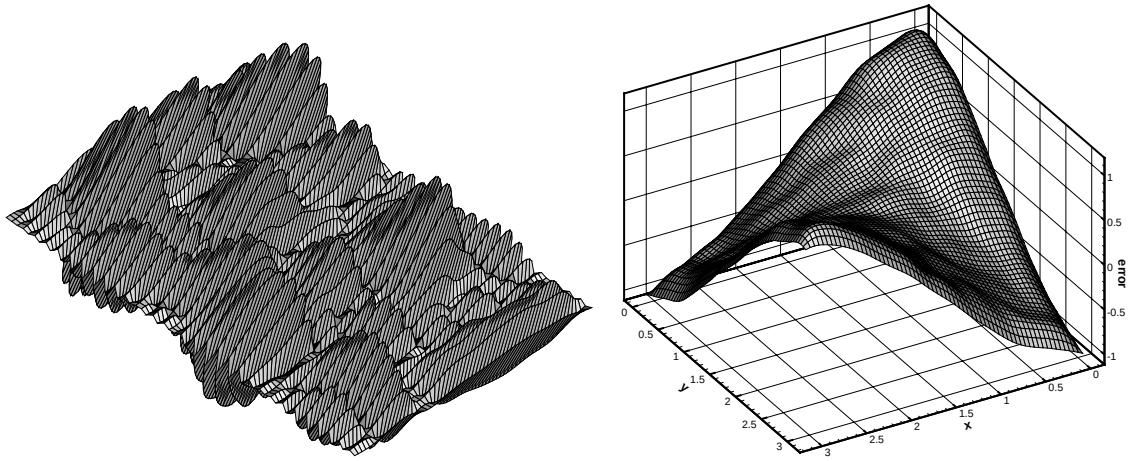


Figure 7: Algebraically smooth error in case of problem (51) and the 5-point Poisson operator with anti-periodic boundary conditions, respectively

4 Post-smoothing and two-level convergence

In this section, we investigate the two-level convergence for symmetric and positive definite problems. As mentioned before, using transfer operators satisfying (10), the corresponding Galerkin operators (5) are also symmetric and positive definite and the coarse-grid correction operators, $K = K_{h,H}$ (9), satisfy the variational principle described in Section 2.4.

We here consider the case of *post-smoothing*, adopting the theoretical approach introduced in [12], also see [63]. For simplicity, we assume that only one smoothing step is performed per cycle, that is, the two-grid operator to be considered is SK. In Section 4.1, we will derive a simple algebraic requirement on interpolation (which implicitly includes also a requirement on the C/F-splitting), in terms of a bound for its "error", which implies uniform two-level convergence w.r.t. the energy norm. In Sections 4.2 and 4.3, we will discuss concrete interpolation approaches satisfying this requirement for relevant classes of matrices. Compared to [63], interpolation has been modified and generalized.

Throughout this and the following Section 5 (which covers the case of pre-smoothing), we employ the inner products and norms defined in (17) and (18). Indices h and H will be used only if absolutely necessary.

4.1 Convergence estimate

For SKe to become small, it is important that the smoothing operator S efficiently reduces all vectors contained in $\mathcal{R}(K)$. Loosely speaking, the error after a coarse-grid correction step has to be "relaxable". Since, assuming property (36) to be satisfied, error reduction by smoothing becomes the less efficient the smaller $\|e\|_2$ is relative to $\|e\|_1$, the least we have to require is that, for all $e \in \mathcal{R}(K)$, $\|e\|_2$ is bounded *from below* by $\|e\|_1$. This leads to the following theorem.

Theorem 4.1 [63] *Let $A > 0$ and let S satisfy the smoothing property (36). Furthermore, assume the C/F-splitting and interpolation to be such that*

$$\|Ke\|_1^2 \leq \tau \, \|Ke\|_2^2 \tag{52}$$

with some $\tau > 0$ being independent of e. Then $\tau \geq \sigma$ and $\|SK\|_1 \leq \sqrt{1 - \sigma/\tau}$.

Proof: By combining (36) and (52) one immediately obtains

$$\|SKe\|_1^2 \leq \|Ke\|_1^2 - \sigma \, \|Ke\|_2^2 \leq (1 - \sigma/\tau) \, \|Ke\|_1^2 \;\leq\; (1 - \sigma/\tau) \, \|e\|_1^2$$

which proves the theorem. △

Condition (52) is not very practical. The following theorem gives a sufficient condition directly in terms of "accuracy" of interpolation.

Theorem 4.2 [63] *If the C/F-splitting and interpolation I_{FC} are such that, for all e,*

$$\|e_F - I_{FC} e_C\|_{0,F}^2 \leq \tau \, \|e\|_1^2 \tag{53}$$

with τ being independent of e, then (52) is satisfied.

Proof: Let any $e \in \mathcal{R}(K)$ be given. Then, for arbitrary e^H, the orthogonality properties of K (Corollary 2.1) imply

$$\|e\|_1^2 = (Ae, e - I_H^h e^H)_E$$

and a straightforward application of Schwarz' inequality yields

$$\|e\|_1^2 = (D^{-\frac{1}{2}} Ae, D^{\frac{1}{2}}(e - I_H^h e^H))_E \leq \|e\|_2 \, \|e - I_H^h e^H\|_0 \, . \tag{54}$$

By selecting e^H to be just the straight projection of e to the coarse level, we obtain

$$\|e - I_H^h e^H\|_0 = \|e_F - I_{FC} e_C\|_{0,F} \, .$$

Hence, assumption (53) implies $\|e\|_1^2 \leq \tau \, \|e\|_2^2$ for all $e \in \mathcal{R}(K)$ which proves (52). $\triangle$

We have already pointed out before (Remark 1.1) that we are not interested in convergence for one particular A only but rather in having *uniform* convergence if A ranges over some reasonable *class* of matrices, $\mathcal{A}$. Since, according to Section 3.2, standard relaxation schemes satisfy the smoothing property (36) uniformly for all problems under consideration, Theorem 4.1 actually implies uniform two-level convergence for $A \in \mathcal{A}$ if we can show that an (operator-dependent) interpolation can be constructed so that (53) holds uniformly for all such A (i.e. with the same τ). That this is possible for relevant classes $\mathcal{A}$, will be shown in the following section. Before, however, we want to make some general remarks on the requirement (53).

Remark 4.1 In the limit case of zero row sum matrices, the right hand side of (53) is zero for all constant vectors e. Hence, constants necessarily have to be interpolated exactly (cf. Remark 1.2). Note that this is not necessary for non-singular matrices. In fact, one may be able to satisfy (53) with smaller τ-values if one does *not* force constants to be interpolated exactly (see the related discussion in Section 4.2). $\ll$

Remark 4.2 Although (53) has to hold for all e, it implies a non-trivial condition only if e is algebraically smooth. This is most easily seen by writing (53) in terms of the eigenvectors ϕ of $D^{-1}A$ and using (31),

$$\|\phi_F - I_{FC} \phi_C\|_{0,F}^2 \leq \lambda \, \tau \, \|\phi\|_0^2 \, . \tag{55}$$

Requiring this to be uniformly satisfied within a relevant class $\mathcal{A}$ of matrices implies a non-trivial condition only for those $\phi = \phi_A$ which correspond to the small eigenvalues $\lambda = \lambda_A$, in particular, those which approach zero if A varies in $\mathcal{A}$. According to Remark 3.1, these are just the algebraically smooth eigenvectors. If $\mathcal{A}$ consists of the h-discretization matrices corresponding to a standard and isotropic elliptic problem, we know that algebraically smooth eigenvectors are also geometrically smooth and their eigenvalues satisfy $\lambda = O(h^2)$. In such cases, obviously, (53) is closely related to the requirement of *first order* interpolation. $\ll$

We finally want to note that the two-level approach considered here can be regarded as an approximation to the direct solver (post-smoothing variant) described in Section 2.3. In particular, the requirement that the error after a coarse-grid correction step has to be

"relaxable" (see beginning of this section) corresponds to the property of the direct solver that all vectors in $\mathcal{R}(K)$ are annihilated by smoothing (cf. Lemma 2.1).

4.2 Direct interpolation

In the following, we describe and discuss basic approaches to automatically construct (operator-dependent) interpolation which, for relevant classes $\mathcal{A}$, can be shown to uniformly satisfy (53). For ease of motivation, we start with the class of M-matrices in Section 4.2.1. Generalisations are given in Sections 4.2.2 and 4.2.3. Regarding a realisation in practice, we refer to Section 7.

In order to motivate the general approach in constructing interpolation, we recall that (53) is a non-trivial condition only for algebraically smooth error (see Remark 4.2). For such error, however, we have heuristically seen in Section 3.3 that the basic equations (27) are approximately satisfied (cf. (40)). Consequently, the definition of interpolation will be based on the same equations.

That is, given a C/F-splitting and sets $P_i \subseteq C$ $(i \in F)$ of interpolatory points, the goal is to define the interpolation weights w_{ik} in

$$e_i = \sum_{k \in P_i} w_{ik}\, e_k \quad (i \in F) \tag{56}$$

so that (56) yields a reasonable approximation for any algebraically smooth e which approximately satisfies

$$a_{ii} e_i + \sum_{j \in N_i} a_{ij} e_j = 0 \quad (i \in F)\ . \tag{57}$$

Of course, the actual construction of the C/F-splitting and the interpolation itself are closely related processes. Generally, the splitting has to be such that each F-point has a "sufficiently strong connection" to the set of C-points. Although this connectivity does not necessarily have to be via *direct* couplings, in the following sections we only consider "direct" interpolation, that is, we assume the sets of interpolatory points to satisfy $P_i \subseteq C \cap N_i$ where, as before, N_i denotes the direct neighborhood (12) of point i. This is for simplicity; some remarks on more general "indirect" interpolations are contained in Section 4.3.

Remark 4.3 Variables which are not coupled to any other variable (corresponding to matrix rows with all off-diagonal entries being zero) will always become F-variables which, however, do not require any interpolation. For simplicity, we exclude such trivial cases in the following. ≪

4.2.1 M-matrices

We have seen in Section 3.3.1 that, for symmetric M-matrices, algebraically smooth error varies slowly in the direction of *strong* couplings. That is, the error at a point i is essentially determined by a weighted average of the error at its strong neighbors. Consequently,

assuming $\emptyset \neq P_i \subseteq C \cap N_i$, the more strong connections of any F-variable i are contained in P_i, the better will

$$\frac{1}{\sum_{k\in P_i} a_{ik}} \sum_{k\in P_i} a_{ik}e_k \;\approx\; \frac{1}{\sum_{j\in N_i} a_{ij}} \sum_{j\in N_i} a_{ij}e_j \tag{58}$$

be satisfied for smooth error. This suggests approximating (57) by

$$a_{ii}e_i + \alpha_i \sum_{k\in P_i} a_{ik}e_k = 0 \quad \text{with} \quad \alpha_i = \frac{\sum_{j\in N_i} a_{ij}}{\sum_{k\in P_i} a_{ik}} \tag{59}$$

which leads to an interpolation formula (56) with matrix-dependent, positive weights:

$$w_{ik} = -\alpha_i a_{ik}/a_{ii} \quad (i \in F,\ k \in P_i)\ . \tag{60}$$

Note that the row sums of (57) and (59) are equal and we have

$$a_{ii}\Big(1 - \sum_{k\in P_i} w_{ik}\Big) = s_i := \sum_j a_{ij} \tag{61}$$

showing that $\sum_{k\in P_i} w_{ik} = 1$ if $s_i = 0$. Consequently, in the limit case of zero row sum matrices, constants are interpolated exactly (cf. Remark 4.1). For regular matrices, however, this is not the case. Instead, the weights are chosen so that $I_{FC}\,1_C$ is an approximation to $\widehat{I}_{FC}\,1_C$ (see Section 2.3). More precisely, $I_{FC}\,1_C$ equals the result of one Jacobi step applied to the equations (57) with the vector $e = 1$ as starting vector. (Here 1 denotes the vector with all components being ones.)

The above interpolation approach can formally be applied to any M-matrix and any C/F-splitting provided that $C \cap N_i \neq \emptyset$ for each $i \in F$. The following theorem shows that (53) can be satisfied uniformly within the class of weakly diagonally dominant M-matrices whenever the sets $C \cap N_i$ are reasonably large.

Theorem 4.3 *Let A be a symmetric M-matrix with $s_i = \sum_j a_{ij} \geq 0$. With fixed $\tau \geq 1$ select a C/F-splitting so that, for each $i \in F$, there is a set $P_i \subseteq C \cap N_i$ satisfying*

$$\sum_{k\in P_i} |a_{ik}| \geq \frac{1}{\tau} \sum_{j\in N_i} |a_{ij}|\ . \tag{62}$$

Then the interpolation (56) with weigths (60) satisfies (53).

Proof: We first note that, according to Remark 4.3, $P_i \neq \emptyset$ for all $i \in F$. Because of (42), we can estimate for all e

$$\|e\|_1^2 = (Ae, e)_E \geq \sum_{i\in F} \Big(\sum_{k\in P_i} (-a_{ik})(e_i - e_k)^2 + s_i e_i^2 \Big)\ . \tag{63}$$

On the other hand, employing Schwarz' inequality, we can estimate

$$\begin{aligned} \|e_F - I_{FC}e_C\|_{0,F}^2 &= \sum_{i\in F} a_{ii}\Big(e_i - \sum_{k\in P_i} w_{ik}e_k\Big)^2 \\ &= \sum_{i\in F} a_{ii}\Big(\sum_{k\in P_i} w_{ik}(e_i - e_k) + (1 - \sum_{k\in P_i} w_{ik})e_i \Big)^2 \qquad (64) \\ &\leq \sum_{i\in F} a_{ii}\Big(\sum_{k\in P_i} w_{ik}(e_i - e_k)^2 + (1 - \sum_{k\in P_i} w_{ik})e_i^2 \Big)\ . \qquad (65) \end{aligned}$$

Observing (61), the previous two estimates imply (53) if $a_{ii} w_{ik} \leq \tau |a_{ik}|$ holds for $i \in F$, $k \in P_i$. According to the definition of the interpolation weights (60), this is equivalent to $\alpha_i \leq \tau$ $(i \in F)$ which, in turn, is equivalent to assumption (62). $\triangle$

The requirement of weak diagonal dominance in the previous theorem is sufficient but not necessary. The following generalization applies to the class of M-matrices whose row sums are uniformly bounded from below and whose eigenvalues are uniformly bounded away from zero.

Theorem 4.4 *Let the symmetric M-matrix A satisfy $s_i = \sum_j a_{ij} \geq -c$ with some $c \geq 0$ and assume $(Ae,e)_E \geq \epsilon (e,e)_E$ for all e with some $\epsilon > 0$. With fixed $\tau \geq 1$, select a C/F-splitting as in Theorem 4.3. Then the interpolation (56) with weigths (60) satisfies (53) with τ replaced by some $\widetilde{\tau} = \widetilde{\tau}(\epsilon, c, \tau)$. As a function of ϵ and c, we have $\widetilde{\tau} \to \infty$ if either $c \to \infty$ or $\epsilon \to 0$.*

Proof: Let us assume that $s_i < 0$ for (at least) one i. Instead of (63) we employ the following estimate with $\widetilde{A} = A + cI$:

$$(\widetilde{A}e, e)_E \geq \sum_{i \in F} \Big(\sum_{k \in P_i} (-a_{ik})(e_i - e_k)^2 + (c + s_i) e_i^2 \Big) .$$

In order to estimate the interpolation error, we proceed as in the proof of the previous theorem. However, we need to modify the estimation of (64) for those $i \in F$ for which $s_i < 0$ (because $1 - \sum_{k \in P_i} w_{ik} < 0$, see (61)) by, for instance, inserting an additional estimate of the form $(a+b)^2 \leq 2(a^2 + b^2)$ for each such i. A straightforward computation, exploiting that $|s_i| \leq c$ and $a_{ii} \geq \epsilon$, then yields an estimate of the form

$$\| e_F - I_{FC} e_C \|_{0,F}^2 \leq \tau_1 (\widetilde{A}e, e)_E + \tau_2 (e,e)_E \leq \widetilde{\tau} \, \| e \|_1^2$$

with $\widetilde{\tau} = \widetilde{\tau}(\epsilon, c, \tau)$. The rest of the theorem follows from the explicit form of $\widetilde{\tau}$. $\triangle$

While the smoothing property (36) is uniformly satisfied in the class of *all* symmetric M-matrices, the previous theorem indicates that (53) cannot be expected to uniformly hold in this class. In particular, the smaller the first eigenvalue of A (i.e. the smaller ϵ) is, the higher is the required accuracy in interpolating the corresponding eigenvector. However, unless this eigenvector is constant (cf. Remark 1.2), this cannot be achieved by the above interpolation. The following example illustrates this situation.

Example 4.1 Consider the class of matrices A_c $(0 \leq c < \lambda_0)$ defined by discretizing the Helmholtz operator $-\Delta u - cu$ on the unit square with *fixed* mesh size h and Dirichlet boundary conditions. Here $\lambda_0 > 0$ denotes the smallest eigenvalue of the corresponding discrete Poisson operator, A_0. If we select $e = \phi_0$ as the corresponding eigenfunction (normalized so that $\| e \|_E = 1$), we have for each $A = A_c$ that $\| e \|_1^2 = \lambda_0 - c$. Thus, for (53) to hold independently of c,

$$\| e_F - I_{FC} e_C \|_{0,F}^2 \leq \tau \, (\lambda_0 - c)$$

is required which means that the first eigenfunction of the Poisson operator has to be approximated with increasing accuracy if $c \to \lambda_0$. However, this is generally not true

unless the interpolation formula is improved by special techniques (e.g. based on an approximate knowledge of ϕ_0 [63]). Regarding numerical results for this case, see Section 8.5.3. $\ll$

According to the above theorems, the C/F-splitting should be selected so that, for each $i \in F$, a *fixed fraction* of its total strength of connection is represented in C (and used for interpolation). However, given any τ, this leaves quite some freedom in realising a concrete splitting algorithm. Although specific realisations cannot be distinguished within the framework of this algebraic two-level theory, the finally resulting convergence may substantially depend on the details of this realisation. We want to make some basic remarks:

Remark 4.4 The concrete choice of τ is crucial. Clearly, the larger τ is, the weaker is the assumption (62). In particular, for large τ, (62) allows for rapid coarsening, but the two-level convergence will be very slow (see Section 4.1). On the other hand, the choice $\tau = 1$ gives best convergence, but will force *all* neighbors of $i \in F$ to be in C. In fact, since the latter means that the submatrix A_{FF} becomes diagonal, this results in a direct solver as described in Section 2.3 (if combined with F-relaxation for smoothing) and we have already seen that this approach, if applied recursively, will be extremely inefficient (cf. Example 2.1). A reasonable compromise is $\tau = 2$ which means that about 50% of the total strength of connections of every F-point has to be represented on the next coarser level. However, from a practical point of view, coarsening may still be too slow, in particular, for matrices which have many row entries of similar size. We come back to a practical algorithm in Section 7. $\ll$

Remark 4.5 To satisfy (62) with as few C-points as possible, one should arrange the splitting so that C-points are only chosen from the *strongest* connections of every F-point. This just means coarsening "in the direction of smoothness". $\ll$

Remark 4.6 Given an application with geometrical background, the algebraic condition (62) does not take the geometric locations of C-points relative to the F-points into account. In fact, this is the main reason for the limited accuracy of purely algebraically defined interpolation as mentioned in Remark 4.2. In practice, however, the accuracy of interpolation – and through this the resulting convergence – can often be substantially improved by just arranging the C/F-distribution carefully. As a rule, it has turned out to be beneficial to arrange the C/F-splitting so that the set of C-points builds (approximately) a maximal set with the property that the C-points are not strongly coupled among each other ("maximally independent set") and that the F-points are "surrounded" by their interpolatory C-points. This can be ensured to a large extent by merely exploiting the connectivity information contained in the matrix. For an illustration, see the following example. $\ll$

Example 4.2 A careful arrangement of the C/F-splitting in the sense of the previous remark is particularly beneficial if the matrix A corresponds to a geometrically posed problem with many off-diagonal entries being *of essentially the same size*. Consider, for

instance, the 9-point discretization of the Poisson operator,

$$\frac{1}{3h^2}\begin{bmatrix} -1 & -1 & -1 \\ -1 & 8 & -1 \\ -1 & -1 & -1 \end{bmatrix}_h . \tag{66}$$

Figure 8 illustrates some local C/F-arrangements all of which are admissible (in the sense of (62)) if we set $\tau = 4$, say. Clearly, interpolation given by the left-most arrangement is worst: it just gives first order accuracy, the best we can really *ensure* by purely algebraic means (cf. Remark 4.2). Following the rule mentioned in Remark 4.6, we would obtain a C/F-splitting for which the two right-most arrangements are characteristic. Both of them correspond to second order interpolation which, as we know from geometric multigrid, gives a much better performance for such balanced stencils as considered in this example. The second arrangement does not give second order, but is still better than the first one.

This illustrates that a proper arrangement of the splitting may substantially enhance convergence. Ignoring this, may, in the worst case, not only cause relatively slow convergence of the two-level method, but also an h-dependent convergence behavior of full V-cycles. For an extreme example of such a situation, see Example 6.1. Clearly, in general, there is no way to strictly ensure optimal interpolation and convergence by exploiting only algebraic information contained in the matrix. In practice, however, it is usually sufficient to avoid extreme one-sided interpolation (see the related discussion in Section 6 and also in Section 9). ≪

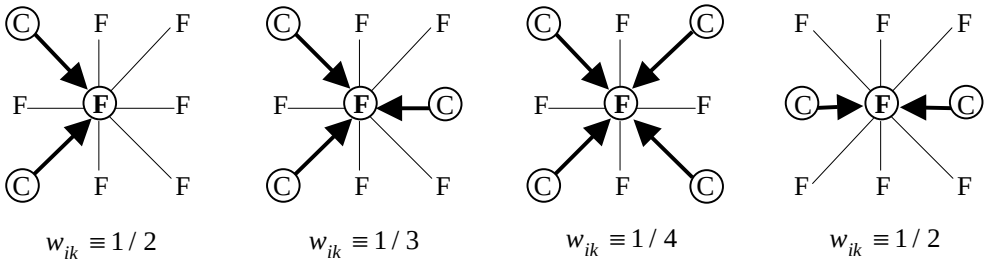


Figure 8: Different C/F-arrangements and corresponding interpolation formulas

For completeness, we want to briefly mention a few other typical approaches to define interpolation which are, however, not in the focus of this paper and may well be skipped in reading. In each case, we briefly discuss requirements which, instead of (62), ensure (53) for weakly diagonally dominant M-matrices. Note that any interpolation (56) with $P_i \subseteq C \cap N_i$ satisfies (53) if the following two inequalities hold:

$$0 \le a_{ii} w_{ik} \le \tau \, |a_{ik}| \, , \quad 0 \le a_{ii}(1 - \sum_{k \in P_i} w_{ik}) \le \tau \, s_i \, . \tag{67}$$

This follows immediately from the proof of Theorem 4.3 and will be used below.

Variant 1. One variant, considered in [63], starts from the assumption

$$\sum_{j \notin P_i} a_{ij} e_j \;\approx\; (\sum_{j \notin P_i} a_{ij}) e_i$$

instead of (58). This corresponds to adding all non-interpolatory connections to the diagonal, leading to the weights

$$w_{ik} = -a_{ik} / \sum_{j \notin P_i} a_{ij} \ . \tag{68}$$

A certain drawback of this approach is that the denominator in (68) may, in principle, become zero or even negative for matrices which contain rows with $s_i < 0$. Apart from this, however, this variant performs very comparably to the interpolation (60) (in fact, in the limit case of zero row sums, $s_i \equiv 0$, both interpolations are identical). For weakly diagonally dominant M-matrices, the requirement

$$\sum_{j \notin P_i} a_{ij} \geq \frac{1}{\tau} a_{ii} \quad (i \in F) \ . \tag{69}$$

can be seen to imply (67) and, hence, also (53).

Variant 2. The interpolation weights

$$w_{ik} = a_{ik} / \sum_{j \in P_i} a_{ij}$$

are constructed so that $\sum_{k \in P_i} w_{ik} \equiv 1$ which forces constants to be *always* interpolated exactly. Compared to (60), no essential difference in behavior is expected as long as $s_i \approx 0$. However, if this is strongly violated for some i (e.g. near boundaries), the approximation of algebraically smooth error (which is *not* constant) becomes less accurate which, in turn, may cause a certain reduction in convergence speed. According to (67), (53) is satisfied if

$$\sum_{k \in P_i} |a_{ik}| \geq \frac{1}{\tau} a_{ii} \quad (i \in F) \ . \tag{70}$$

This also indicates that, in cases of strongly diagonally dominant rows, (70) may unnecessarily slow down the coarsening process compared to (62), or, alternatively, lead to worse bounds for the interpolation error.

Variant 3. As an alternative to the previous interpolation, it is sometimes proposed to simply use equal weights $w_{ik} = 1/n_i$ where $n_i = |P_i|$ just denotes the number of neighbors used for interpolation. Then (53) is satisfied if

$$n_i \, |a_{ik}| \geq \frac{1}{\tau} a_{ii} \quad (i \in F, \, k \in P_i) \ . \tag{71}$$

Obviously, for this interpolation to be reasonable it is particularly crucial to interpolate *only* from strong connections (otherwise, n_i has to be too large). Unless all interpolatory connections are of approximately the same size, this interpolation will be substantially worse than the previous ones.

Variant 4. Aggregation-based AMG approaches (see Section 9) can be interpreted as to use interpolation to any F-point from *exactly one* C-point only (with weight 1), that is, $|P_i| = 1$. This interpolation allows for a rapid coarsening and, moreover, the computation of the coarse-level Galerkin operators becomes extremely simple. On the other hand, this interpolation is the crudest possible by definition, leading to rather bad bounds for the interpolation error. In fact, (53) is satisfied only if

$$|a_{ik}| \geq \frac{1}{\tau} a_{ii} \quad (i \in F,\, k \in P_i) \; . \tag{72}$$

Consequently, the smaller the (significant) off-diagonal entries are compared to the diagonal, the larger τ will be. In particular, (53) cannot be satisfied uniformly within the class of weakly diagonally dominant M-matrices any more. Generally, using this most simple interpolation, convergence will be rather poor (and h-dependent for complete V-cycles; see the related discussion in Section 6).

In spite of this, aggregation-based AMG approaches have become quite popular, one reason being their simplicity and ease in programming. However, to become practical, they require additional acceleration by means of various techniques, for instance, by smoothing of interpolation ("smoothed aggregation"). In addition, aggregation-based AMG is typically used as pre-conditioner rather than stand-alone. We will come back to such approaches in Section 9.

4.2.2 Essentially positive-type matrices

In the previous section we considered off-diagonally non-positive matrices. However, the essential estimates carry over to certain matrices with relatively small positive off-diagonal entries such as the *essentially positive-type matrices* considered in Section 3.3.2. The theorem below shows that, for such matrices, it is sufficient to base the splitting on the connectivity structure induced by the negative off-diagonal entries only. Accordingly, interpolation needs to be done only from neighbors with negative coefficients. In the following, we make use of the notation a_{ij}^+, a_{ij}^- and N_i^+, N_i^- as defined in Section 2.2.

Using interpolatory points $\emptyset \neq P_i \subseteq C \cap N_i^-$ and recalling the heuristic considerations on algebraically smooth error in Section 3.3.2, we might define interpolation exactly as before (59). However, for reasons explained in Remark 4.8 below, we rather extend the interpolation to the case of positive entries by adding all such entries to the diagonal (to preserve row sums). That is, we use

$$\widetilde{a}_{ii} e_i + \alpha_i \sum_{k \in P_i} a_{ik}^- e_k = 0 \quad \text{with} \quad \widetilde{a}_{ii} = a_{ii} + \sum_{j \in N_i} a_{ij}^+ \, , \;\; \alpha_i = \frac{\sum_{j \in N_i} a_{ij}^-}{\sum_{k \in P_i} a_{ik}^-} \tag{73}$$

instead of (59), yielding positive interpolation weights

$$w_{ik} = -\alpha_i a_{ik}^- / \widetilde{a}_{ii} \quad (i \in F,\, k \in P_i) \; . \tag{74}$$

Note that the row sums of (57) and (73) are equal and we have

$$\widetilde{a}_{ii} \Big(1 - \sum_{k \in P_i} w_{ik} \Big) = s_i \; . \tag{75}$$

Theorem 4.5 *Let $A > 0$ be essentially positive-type (45) with $s_i = \sum_j a_{ij} \geq 0$. With fixed $\tau \geq 1$, select a C/F-splitting such that, for each $i \in F$, there is a set $P_i \subseteq C \cap N_i^-$ satisfying*

$$\sum_{k \in P_i} |a_{ik}^-| \geq \frac{1}{\tau} \sum_{j \in N_i} |a_{ij}^-| \,. \tag{76}$$

Then the interpolation (56) with weights (74) satisfies (53) with τ/c rather than τ.

Proof: The proof runs analogously to the one of Theorem 4.3. We first note again that $P_i \neq \emptyset$ for all $i \in F$ (cf. Remark 3.3). Since $c \leq 1$ and $s_i \geq 0$, (42) and (45) imply

$$\begin{aligned} (Ae,e)_E &\geq \frac{c}{2} \sum_{i,j} (-a_{ij}^-)(e_i - e_j)^2 + \sum_i s_i e_i^2 \\ &\geq c \sum_{i \in F} \Big(\sum_{k \in P_i} (-a_{ik})(e_i - e_k)^2 + s_i e_i^2 \Big) \,. \end{aligned}$$

Using (65) and observing that $\widetilde{a}_{ii} \geq a_{ii}$, we see that (53) is satisfied with τ/c rather than τ if $\alpha_i \leq \tau$ $(i \in F)$ which is equivalent to (76). $\triangle$

Remark 4.7 There is a straightforward generalization of this theorem to the case of negative row sums which is analogous to Theorem 4.4. $\ll$

Remark 4.8 The reason for adding positive entries to the diagonal (rather than using the same formula as in the previous section) is mainly practical: In practice, we want to implement an interpolation which (at least formally) can also be employed to matrices which are not of strictly essentially positive type, for instance, which contain some particularly large positive entries. However, in such cases, $\sum_{j \in N_i} a_{ij}$ might become zero or even positive for certain $i \in F$ and, using (59), we would obtain zero or even negative interpolation weights. This is avoided by adding positive entries to the diagonal. Nevertheless, the approximation (73) is, obviously, only reasonable if we can assume that, for each $i \in F$, an algebraically smooth error satisfies

$$\sum_j a_{ij}^+ e_j \approx \sum_j a_{ij}^+ e_i \tag{77}$$

which, for $j \neq i$, either requires $e_j \approx e_i$ or a_{ij}^+ to be small (relative to a_{ii}). According to the heuristic considerations in Section 3.3.2, this can be assumed for essentially positive type matrices as considered here. However, we will see in the next section that (73) becomes less accurate (in the sense of (53)) if (77) is strongly violated. $\ll$

4.2.3 General case

Although the previous interpolation can, formally, always be used (provided each i has, at least, one negative connection), it is well suited only if (77) can be assumed to hold for an algebraically smooth error. To demonstrate this, let us consider a problem already mentioned in Section 3.3.3 (Example 3.6):

Example 4.3 Consider the matrix A which corresponds to the stencil

$$\begin{bmatrix} & +1 & \\ -1 & 4 & -1 \\ & +1 & \end{bmatrix} , \tag{78}$$

applied on an $N \times N$ grid with mesh size $h = 1/N$ and Dirichlet boundary conditions.

We select the particular error vector e with components defined by +1 (-1) along even (odd) horizontal grid lines. For this vector, we have $e_i^2 = 1$ for all i, $(e_i - e_j)^2 = 0$ if $a_{ij} < 0$ and $(e_i - e_j)^2 = 4$ if $a_{ij} > 0$. Using (42), we therefore see that

$$(Ae, e)_E = -2 \sum_{i,j} a_{ij}^+ + \sum_i s_i = \sum_i \Big(s_i - 2 \sum_j a_{ij}^+ \Big) = \sum_i t_i$$

with $t_i := a_{ii} - \sum_{j \neq i} |a_{ij}|$. Since $t_i = 0$ in the interior and $t_i = 1$ along the boundary, we obtain

$$(Ae, e)_E = O(N) . \tag{79}$$

Assume now any C/F-splitting and any interpolation I_{FC} with $P_i \subseteq C \cap N_i^-$ to be given. Using (64), we then see that

$$\| e_F - I_{FC} e_C \|_{0,F}^2 = \sum_{i \in F} a_{ii} \Big(1 - \sum_{k \in P_i} w_{ik} \Big)^2 . \tag{80}$$

If I_{FC} is the particular interpolation from the previous section, (75) implies that each term of the sum in (80) is non-zero and we obtain

$$\| e_F - I_{FC} e_C \|_{0,F}^2 = O(N^2) . \tag{81}$$

Because of this and (79), inequality (53) cannot hold independently of N. $\ll$

The problem seen in this example is that an algebraically smooth error is geometrically smooth only in x-direction but highly oscillatory in y-direction (cf. Section 3.3.3). (The particular error vector considered in the example is actually algebraically smooth.) That is, we have $e_j \approx -e_i$ if $a_{ij} > 0$ and $j \neq i$. Consequently, (77) is strongly violated which explains the high interpolation error observed above. A re-definition of $\widetilde{a}_{ii}$ in (73) by *subtracting* all positive connections from the diagonal (rather than *adding* them), is a way out for situations such as the one considered here (in fact, this would give $O(N)$ in (81) rather than $O(N^2)$) but it is just the wrong thing to do in other cases.

This indicates that the correct treatment of positive connections in interpolation is, in general, more critical than that of negative connections. We have seen in Section 3.3.2 that, assuming $a_{ij} > 0$, algebraically smooth error e can still be expected to change slowly from i to j for essentially positive type matrices, in particular, if there exist strong negative paths from i to j. On the other hand, for matrices which are approximately weakly diagonally dominant, we have to expect an oscillatory behavior (which is the stronger the larger a_{ij} is relative to a_{ii}, see (50)). In *both* cases, however, we may expect that those e_k which correspond to positive connections $a_{ik} > 0$, change slowly *among each other* (unless a_{ik}

is very small in which case we can ignore its influence). This gives rise to the following generalization of the interpolation approach (59) (M-matrix case) which is completely symmetric in the treatment of negative and positive connections.

Let us assume that some $i \in F$ has both negative and positive connections, that is, $N_i^- \neq \emptyset$ and $N_i^+ \neq \emptyset$. Then, assuming the C/F-splitting to be such that at least one connection of either sign is contained in C, we can select two sets of interpolation points, $\emptyset \neq P_i^- \subseteq C \cap N_i^-$ and $\emptyset \neq P_i^+ \subseteq C \cap N_i^+$. Setting $P_i = P_i^- \cup P_i^+$, we then use

$$a_{ii} e_i + \alpha_i \sum_{k \in P_i} a_{ik}^- e_k + \beta_i \sum_{k \in P_i} a_{ik}^+ e_k = 0 \tag{82}$$

instead of (59) with

$$\alpha_i = \frac{\sum_{j \in N_i} a_{ij}^-}{\sum_{k \in P_i} a_{ik}^-} \quad \text{and} \quad \beta_i = \frac{\sum_{j \in N_i} a_{ij}^+}{\sum_{k \in P_i} a_{ik}^+} \, .$$

At this point, we have assumed that, for an algebraically smooth error, approximations analogous to (58) hold *separately* for the negative and the positive connections. This leads to the following interpolation weights:

$$w_{ik} = \begin{cases} -\alpha_i \, a_{ik}/a_{ii} & (k \in P_i^-) \\ -\beta_i \, a_{ik}/a_{ii} & (k \in P_i^+) \end{cases} \, . \tag{83}$$

Note that $w_{ik} > 0$ ($k \in P_i^-$) and $w_{ik} < 0$ ($k \in P_i^+$). If either $N_i^+ = \emptyset$ or $N_i^- = \emptyset$, these definitions are to be modified in a straightforward way by setting $P_i^+ = \emptyset$, $\beta_i = 0$ and $P_i^- = \emptyset$, $\alpha_i = 0$, respectively. In particular, for M-matrices, the above interpolation is identical to the one in (59). In any case, the row sums of (57) and (82) are equal and we have

$$a_{ii} \Big(1 - \sum_{k \in P_i} w_{ik} \Big) = s_i \, . \tag{84}$$

Formally, this approach can always be applied even in cases where most, or even all, entries are positive. The following theorem – a direct generalization of Theorem 4.3 – shows that (53) can uniformly be satisfied in the class of weakly diagonally dominant matrices A under completely symmetric conditions for positive and negative entries. Note that the above example belongs to this class. A more realistic application arises, for instance, in connection with *anti-periodic* boundary conditions (see also Section 3.3.3).

Theorem 4.6 *Let $A > 0$ and $t_i = a_{ii} - \sum_{j \in N_i} |a_{ij}| \geq 0$. With fixed $\tau \geq 1$, select a C/F-splitting such that the following holds for each $i \in F$. If $N_i^- \neq \emptyset$, there is a set $P_i^- \subseteq C \cap N_i^-$ satisfying*

$$\sum_{k \in P_i^-} |a_{ik}| \geq \frac{1}{\tau} \sum_{j \in N_i^-} |a_{ij}| \tag{85}$$

and, if $N_i^+ \neq \emptyset$, there is a set $P_i^+ \subseteq C \cap N_i^+$ satisfying

$$\sum_{k \in P_i^+} a_{ik} \geq \frac{1}{\tau} \sum_{j \in N_i^+} a_{ij} \, . \tag{86}$$

Then the interpolation (56) with weights (83) satisfies (53).

Proof: Using (49), we can estimate

$$(Ae,e)_E \geq \sum_{i\in F}\Big(\sum_{k\in P_i^-}|a_{ik}|(e_i-e_k)^2+\sum_{k\in P_i^+}a_{ik}(e_i+e_k)^2+t_ie_i^2\Big)\,. \tag{87}$$

Regarding an estimate for $\|e_F - I_{FC}e_C\|^2_{0,F}$, we start from (64). We first note that

$$a_{ii}(1-\sum_{k\in P_i}w_{ik}) = s_i = 2\sum_{j\in N_i^+}a_{ij}+t_i = 2\beta_i\sum_{k\in P_i^+}a_{ik}+t_i = 2a_{ii}\sum_{k\in P_i^+}|w_{ik}|+t_i$$

which implies

$$\sum_{k\in P_i^+}w_{ik}(e_i-e_k)+(1-\sum_{k\in P_i}w_{ik})e_i = \sum_{k\in P_i^+}|w_{ik}|(e_i+e_k)+t_i/a_{ii}\,e_i\,.$$

The latter, inserted into (64), gives the following estimate by applying Schwarz' inequality

$$\begin{aligned}\|e_F - I_{FC}e_C\|^2_{0,F} &= \sum_{i\in F}a_{ii}\Big(\sum_{k\in P_i^-}w_{ik}(e_i-e_k)+\sum_{k\in P_i^+}|w_{ik}|(e_i+e_k)+t_i/a_{ii}\,e_i\Big)^2\\ &\leq \sum_{i\in F}a_{ii}\Big(\sum_{k\in P_i^-}w_{ik}(e_i-e_k)^2+\sum_{k\in P_i^+}|w_{ik}|(e_i+e_k)^2+t_i/a_{ii}\,e_i^2\Big)\,. \end{aligned}\tag{88}$$

Regarding this estimate, note that $\sum_{k\in P_i}|w_{ik}|+t_i/a_{ii}=1$. The estimates (87) and (88) imply (53) if $a_{ii}|w_{ik}|\leq\tau|a_{ik}|$ ($i\in F$, $k\in P_i$) which is equivalent to the assumptions of the theorem. △

Remark 4.9 We note that there is a straightforward extension of this theorem to the case $t_i\geq -c$ with some $c\geq 0$ which is analogous to Theorem 4.4. ≪

Although the above theorem applies only to weakly diagonally dominant matrices, it is heuristically clear that the approach (82) is also reasonable in other cases. In particular, if we can assume algebraically smooth error to vary slowly even along positive connections (as in the essentially positive type case), (82) is as good an approximation as (73). In fact, replacing e_k ($k\in P_i^+$) in (82) by e_i gives exactly (73).

The latter indicates that, in practice, there is no need to use the approach (82) for all $i\in F$ but rather only for those for which oscillatory behavior has to be expected. This simplifies the coarsening algorithm substantially in case of elliptic differential problems where the largest couplings will usually be negative. In such cases, we will use the full interpolation as described above only in case there really exist large positive couplings (comparable in size to the largest negative connection, say). Otherwise, we proceed as described in the previous section, that is, we do not interpolate from positive connections but rather add them to the diagonal (for more details, see Section 7).

Remark 4.10 A different interpolation has been considered in [33] (which is actually a direct generalization of the one used in [63]). Compared to (82), $\alpha_i=\beta_i=1$ has been selected and all non-interpolatory connections are used to modify the diagonal element:

$$a_{ii} \quad\longrightarrow\quad \widetilde{a}_{ii} = a_{ii} - \sum_{j\notin P_i,\,j\neq i}|a_{ij}|\,. \tag{89}$$

The resulting interpolation is limited to (and actually has been developed for) weakly diagonally dominant matrices where algebraically smooth error really oscillates along positive connections (note that positive matrix entries are *subtracted* from the diagonal element rather than *added*; see the related comment further above). It is not suited for other applications. In particular, (89) does not preserve row sums and, consequently, constants are not interpolated exactly if $s_i \equiv 0$. Also, the denominator in (89) may become zero or negative. $\ll$

We finally want to note that interpolation from positive connections is not always necessary even if an algebraically smooth error tends to oscillate in certain directions. We just have to ensure that the C/F-splitting is such that the C-variables can represent the oscillations sufficiently well and that interpolation along negative couplings is "accurate enough". We do not want to discuss this aspect any further but rather refer to the above Example 4.3 where the situation is particularly simple. If we perform no coarsening in y-direction (i.e. in the direction of strong positive connections), we may, for instance, use

$$w_{ik} = a_{ik} / \sum_{j \in P_i} a_{ij} \quad \text{with} \quad k \in P_i \subseteq C \cap N_i^-$$

for interpolation (cf. Variant 2 in Section 4.2.1).

4.3 Indirect interpolation

In the previous sections, we considered approaches to construct interpolation based on *direct* connections, that is, interpolation to an F-point i only from its direct neighbors. Correspondingly, the C/F-splittings had to be such that each $i \in F$ is sufficiently strongly connected to the set of C-points via direct connections.

Although a strong F-to-C connectivity is indeed crucial, it does not necessarily have to be via direct connections. In practice, this may limit the speed of coarsening. Too slow coarsening, however, may cause high memory requirements, often unacceptably high. Clearly, a faster coarsening will typically imply a slower convergence. However, the advantages in terms of less memory requirement and lower computational cost per cycle and for the setup, in many cases outweigh the disadvantage of slower convergence. Moreover, the use of AMG as a pre-conditioner (rather than stand-alone) is usually a very efficient means to bring the speed of convergence back up again. This will be seen in Section 8.

In order to permit more rapid coarsening, one has to allow that F-points may be interpolated via sufficiently many of their strong *F-neighbors*. For an illustration, consider the typical geometric scenario of (isotropic) 5-point discretizations on regular meshes. Interpolation based only on direct connections would not allow for the $h \to 2h$ coarsening which is typically used in geometric multigrid methods, the reason being that those F-points i located in the center of a coarse-grid cell have no direct connection to the C-points (see Figure 9, left picture). However, all their direct F-neighbors, j, *do have* strong connections to the C-points and, thus, can interpolate directly.

This simple scenario gives rise to a straightforward generalization of interpolation: First interpolate the j-variables and then, via the resulting interpolation formulas, the i-variable (see Figure 9, right picture). Since the details of a corresponding generalization of the theorems in the previous sections are elementary but rather involved, we here just

outline the main additional step required for the generalization of Theorem 4.3 (M-matrix case). We have a scenario in mind which is analogous to the one described above.

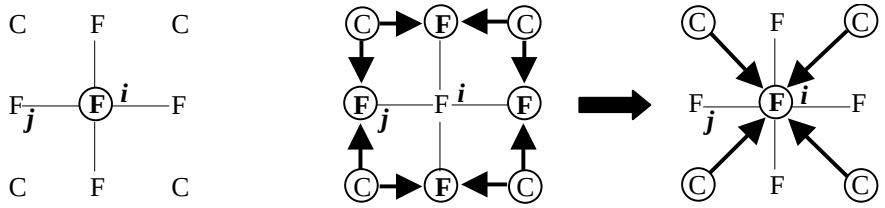


Figure 9: Illustration of indirect interpolation in case of 5-point stencils

Assuming a given C/F-splitting to be such that there is some $i \in F$ which has no (strong) connection to any C-point, we select a set of points $P_i^F \subseteq F \cap N_i$ satisfying (62) with P_i replaced by P_i^F. With the same arguments as at the beginning of Section 4.2.1 we approximate in an intermediate step

$$e_i = \sum_{j \in P_i^F} w_{ij}^F e_j \quad \text{with} \quad w_{ij}^F = -\alpha_i a_{ij}/a_{ii}\,, \quad \alpha_i \leq \tau\,. \tag{90}$$

Assume now that each of the neighboring points $j \in P_i^F$ *can be* interpolated, that is, we have

$$e_j = \sum_{k \in P_j} w_{jk} e_k \quad \text{with} \quad w_{jk} = -\alpha_j a_{jk}/a_{jj}\,, \quad \alpha_j \leq \tau\,. \tag{91}$$

Inserting (91) into (90), yields

$$e_i = \sum_{j \in P_i^F} \sum_{k \in P_j} w_{ij}^F w_{jk} e_k =: \sum_{k \in P_i} w_{ik} e_k \quad \text{with} \quad P_i = \bigcup P_j\,. \tag{92}$$

If, for simplicity, we finally assume that the pairwise intersection of all P_j's is empty (otherwise there will be several terms of the following form), we can estimate for $k \in P_i$

$$\begin{aligned} a_{ii} w_{ik} (e_i - e_k)^2 &= a_{ii} w_{ij}^F w_{jk} (e_i - e_k)^2 \\ &\leq 2 a_{ii} w_{ij}^F w_{jk} ((e_i - e_j)^2 + (e_j - e_k)^2) \\ &= 2 \alpha_i \alpha_j \frac{a_{ij} a_{jk}}{a_{jj}} ((e_i - e_j)^2 + (e_j - e_k)^2) \\ &\leq c_1 \, |a_{ij}| (e_i - e_j)^2 + c_2 \, |a_{jk}| (e_j - e_k)^2\,, \end{aligned}$$

where c_1 and c_2 depend only on τ but not on the given matrix.

By means of such simple estimates and by using the full representation (42) of $\|e\|_1^2$ in (63), one can immediately extend the proof of Theorem 4.3 showing that (53) holds with some $\widetilde{\tau}$ which is larger than τ but does not depend on A. Regarding practical aspects, we refer to Section 7.2.2 ("multi-pass interpolation").

5 Pre-smoothing and two-level convergence

In this section, we investigate two-level convergence if *pre-smoothing* is used rather than post-smoothing, that is, we consider the (energy) norm of the two-level operator KS^ν with some $\nu \geq 1$. Post- and pre-smoothing are treated separately because the *immediate* interaction between coarse-grid correction and smoothing is different for these cases. Clearly, regarding the *asymptotic* two-level convergence, it does not matter whether pre- or post-smoothing is used. In this sense, the conclusions for either of the two cases carry over to the other one. (In practice, we usually employ post- *and* pre-smoothing anyway.)

The theoretical approach used in this section is quite different from the one used in the previous section. In particular, it does not use the smoothing property (36) but is rather directly based on the following equivalence which is an immediate consequence of the variational principle (last statement in Corollary 2.1):

Corollary 5.1 *The two-level estimate* $\|KS^\nu\|_1 \leq \eta$ *holds if and only if, for all* e*, there exists an* e^H *such that*

$$\|S^\nu e - I_H^h e^H\|_1 \leq \eta \, \|e\|_1 \, . \tag{93}$$

The interpretation of (93) is that the speed of convergence depends solely on the efficient interplay between *smoothing* and *interpolation*. To this end, S does *not* necessarily have to satisfy the smoothing property as long as $S^\nu e$ is represented in $\mathcal{R}(I_H^h)$ sufficiently well. The better this is satisfied, the faster the convergence will be.

In fact, we will see in Section 5.1 that we can force η to be small (independently of $A \in \mathcal{A}$ for relevant classes $\mathcal{A}$) *by smoothing only at F-points*. Interpolation can be constructed as described in Section 4.2. Additional Jacobi relaxation steps, applied to the interpolation operator, can be used to speed up convergence further. In Section 5.2 we make some remarks on the use of complete (rather than just F-) relaxation steps for smoothing.

5.1 Convergence using mere F-smoothing

In order to motivate our theoretical approach, let us briefly recall the results on direct solvers (pre-smoothing variant) described in Section 2.3. The very specific smoothing and interpolation operators introduced there, $\widehat{S}$ and $\widehat{I}_H^h$ (see (23) and (25), respectively), have the property

$$\mathcal{R}(\widehat{S}) = \mathcal{R}(\widehat{I}_H^h) = \mathcal{E} \coloneqq \{e \, : \, e_F = -A_{FF}^{-1} A_{FC} \, e_C\} \tag{94}$$

which, obviously, implies that the left side of (93) is identically zero (for $\nu = 1$). This suggests trying to approximate $\widehat{S}$ and $\widehat{I}_H^h$ by some more realistic operators, S and I_H^h.

One can imagine various ways to do this. Recalling that $\widehat{S}$ and $\widehat{I}_H^h$ have been defined by exactly solving the F-equations (21) and (27), respectively, we will here define S and I_H^h by solving these equations only approximately. This becomes particularly easy, if one assumes the submatrix A_{FF} to be *strongly diagonally dominant*,

$$a_{ii} - \sum_{j \in F, j \neq i} |a_{ij}| \geq \delta a_{ii} \quad (i \in F) \tag{95}$$

with some fixed, pre-defined $\delta > 0$. This assumption is very natural for large classes of problems, in particular those considered in this paper. It essentially means that a strong F-to-C connectivity is required which, however, has to be ensured by a reasonable C/F-splitting anyway (cf. Section 4.2; see also Remark 5.3 further below).

All of the following is based on the assumption of strong diagonal dominance (95) and we consider the simplest means to approximate the solutions of the F-equations (21) and (27), namely, by straightforward relaxation involving only F-variables ("F-relaxation", either Gauss-Seidel or Jacobi).

Remark 5.1 Note that strong diagonal dominance (95) is assumed for simplicity. What we really require is that F-relaxation converges at a rate which is independent of the original matrix A. Strong diagonal dominance of A_{FF}, with pre-defined δ, is sufficient to ensure this, but it is not necessary. $\ll$

Before we derive the main convergence estimate in Section 5.1.4, we formally define the smoothing and interpolation operators in Sections 5.1.2 and 5.1.3. The following section contains an auxiliary result.

5.1.1 An auxiliary result

In all of the following, the subspace $\mathcal{E}$, defined in (94), will play an essential role. Given any $e = (e_F, e_C)^T$, we denote its projection onto $\mathcal{E}$ by $\widehat{e}$,

$$\widehat{e} := (\widehat{e}_F, e_C)^T \quad \text{where} \quad \widehat{e}_F := \widehat{I}_{FC} e_C = -A_{FF}^{-1} A_{FC} e_C \ . \tag{96}$$

The following lemma states some basic properties which will be needed below. In particular, it relates the energy norm of e to that of $\widehat{e}$.

Lemma 5.1 *Let $A > 0$ and any C/F-splitting be given. Then the Schur complement C_H (26) is also positive definite and satisfies $\rho(C_H^{-1}) \leq \rho(A^{-1})$. For all e, we have*

$$(Ae, e)_E = (A_{FF}(e_F - \widehat{e}_F), e_F - \widehat{e}_F)_E + (C_H e_C, e_C)_E = \|e - \widehat{e}\|_1^2 + \|\widehat{e}\|_1^2 \tag{97}$$

which immediately implies

$$\|\widehat{e}\|_1 \leq \|e\|_1 \quad \textit{and} \quad \|e_F - \widehat{e}_F\|_{1,F} = \|e - \widehat{e}\|_1 \leq \|e\|_1 \ . \tag{98}$$

Proof: The left equality in (97) follows by a straightforward computation. The positive definiteness of C_H then follows from $(C_H e_C, e_C)_E = (A\widehat{e}, \widehat{e})_E$. Denoting the smallest eigenvalue of A by $\epsilon > 0$, we have $(Ae, e)_E \geq \epsilon (e, e)_E$ for all e. We can then estimate for all e_C

$$(C_H e_C, e_C)_E = (A\widehat{e}, \widehat{e})_E \geq \epsilon (\widehat{e}, \widehat{e})_E \geq \epsilon (e_C, e_C)_E \ .$$

Hence, $\rho(C_H^{-1}) \leq 1/\epsilon = \rho(A^{-1})$. $\triangle$

5.1.2 F-smoothing

As motivated above, we define ν smoothing steps by applying ν F-relaxation steps to approximately solve (21), starting with the most recent approximation $u = (u_F, u_C)^T$ (keeping u_C fixed). We will refer to this process as *F-smoothing*.

> We here use the term "smoothing" although mere F-relaxation (like any other partial relaxation) has no real smoothing properties in the usual sense. In particular, the considerations in Section 3 on algebraically smooth error do not apply here. In fact, F-relaxation aims at approximately *solving* the F-equations (21) (with fixed u_C) rather than smoothing the error of the full system of equations.

Consequently, a relaxation parameter, ω, may be used to speed up the convergence. However, since we require A_{FF} to be strongly diagonally dominant, this is not really necessary and, for simplicity, we only consider the case $\omega = 1$. Having this in mind, one F-smoothing step consists of

$$u \longrightarrow \overline{u} \quad \text{where} \quad Q_{FF}\overline{u}_F + (A_{FF} - Q_{FF})u_F + A_{FC}\,u_C = f_F\,, \quad \overline{u}_C = u_C\ . \tag{99}$$

Q_{FF} is the lower triangular part of A_{FF} (including the diagonal) in case of *Gauss-Seidel relaxation* and $Q_{FF} = D_{FF}$ in case of *Jacobi relaxation*. (In practice, we only use Gauss-Seidel relaxation.) To explicitly compute the corresponding smoothing operator, we re-write this as

$$\overline{u}_F = S_{FF}u_F + (I_{FF} - S_{FF})A_{FF}^{-1}(f_F - A_{FC}\,u_C)\,, \quad \overline{u}_C = u_C \tag{100}$$

or, in terms of the error $e = u^\star - u$ (with $u^\star$ denoting the exact solution of $Au = f$),

$$\overline{e}_F = S_{FF}e_F - (I_{FF} - S_{FF})A_{FF}^{-1}A_{FC}e_C\ , \quad \overline{e}_C = e_C\ .$$

Here, $S_{FF} = I_{FF} - Q_{FF}^{-1}A_{FF}$ denotes the iteration matrix corresponding to the relaxation of the F-variables. Consequently, the relaxation operator reads

$$S_h = \begin{pmatrix} S_{FF} & (S_{FF} - I_{FF})A_{FF}^{-1}A_{FC} \\ 0 & I_{CC} \end{pmatrix} \tag{101}$$

and a simple calculation shows that

$$S_h^\nu e = \begin{pmatrix} S_{FF}^\nu(e_F - \widehat{e}_F) + \widehat{e}_F \\ e_C \end{pmatrix}\ . \tag{102}$$

Obviously, we have rapid convergence $S_h^\nu e \to \widehat{e}$ $(\nu \to \infty)$ for each e.

> **Remark 5.2** Note that F-smoothing does *not* satisfy the smoothing property (36). In fact, (102) shows that $Se = e$ for all $e \in \mathcal{E}$. Hence, (36) cannot hold. $\ll$

5.1.3 Jacobi-interpolation

Given any e_C, we define interpolation by applying μ F-relaxation steps to approximately solve the F-equations (27). However, in order to keep the resulting operator as "local" as possible, we only consider Jacobi-relaxation. That is, we iteratively define

$$e_F^{(\mu)} = P_{FF} e_F^{(\mu-1)} - D_{FF}^{-1} A_{FC} e_C \tag{103}$$

and set $I_{FC} e_C = I_{FC}^{(\mu)} e_C := e_F^{(\mu)}$. Here P_{FF} denotes the Jacobi iteration operator, $P_{FF} = I_{FF} - D_{FF}^{-1} A_{FF}$. In contrast to the F-smoothing process described in the previous section, however, here no "natural" first approximation, $e_F^{(0)}$, is available to start the relaxation process with. Using zero as the first guess is not sufficient as will be seen in Remark 5.6 further below. For now, we assume any "first guess interpolation", $I_{FC}^{(0)}$, to be given and use $e_F^{(0)} = I_{FC}^{(0)} e_C$ as the first guess. (In the following section, we will derive a requirement on $I_{FC}^{(0)}$.)

Since the interpolation operator needs to be known *explicitly* in order to compute the Galerkin operator, we re-write (103) in operator form,

$$I_{FC}^{(\mu)} = P_{FF} I_{FC}^{(\mu-1)} - D_{FF}^{-1} A_{FC} \tag{104}$$

starting with the first-guess interpolation operator, $I_{FC}^{(0)}$. (Regarding the practical computation of the interpolation, see Section 7.2.3.) By subtracting these equations from the equality $\widehat{I}_{FC} = P_{FF} \widehat{I}_{FC} - D_{FF}^{-1} A_{FC}$, we obtain

$$J_{FC}^{(\mu)} = I_{FC}^{(\mu)} - \widehat{I}_{FC} = P_{FF}^{\mu} (I_{FC}^{(0)} - \widehat{I}_{FC}) = P_{FF}^{\mu} J_{FC}^{(0)} \tag{105}$$

where $J_{FC} = I_{FC} - \widehat{I}_{FC}$ denotes the "interpolation error" of I_{FC} relative to $\widehat{I}_{FC}$.

We will later refer to this relaxation of interpolation as *Jacobi-interpolation*. Clearly, for any given $e = (e_F, e_C)^T$, we have rapid convergence $(I_H^h)^{(\mu)} e_C \to \widehat{e}$ $(\mu \to \infty)$.

5.1.4 Convergence estimate

The following theorem yields a requirement on the first guess interpolation $I_{FC}^{(0)}$ which is sufficient to imply uniform two-level convergence. Further below, in Lemma 5.2, we will see that this requirement is very much related to the corresponding one (53) in Section 4.1.

Theorem 5.1 *Let $A > 0$ and assume the C/F-splitting to be such that A_{FF} is strongly diagonally dominant (95) with fixed $\delta > 0$. Let smoothing be performed by $\nu \geq 1$ F-relaxation steps (99). Finally, let the interpolation be defined by $I_{FC} = I_{FC}^{(\mu)}$ with some $\mu \geq 0$ (see (104)) and assume that the first guess interpolation, $I_{FC}^{(0)}$, satisfies*

$$\| (\widehat{I}_{FC} - I_{FC}^{(0)})\, e_C \|_{1,F} \;\leq\; \tau \, \| e \|_1 \tag{106}$$

for all e with some $\tau \geq 0$ being independent of e. Then the following estimate holds:

$$\| K S^{\nu} e \|_1 \;\leq\; (\| S_{FF} \|_{1,F}^{\nu} + \tau \, \| P_{FF} \|_{1,F}^{\mu}) \, \| e \|_1 \; . \tag{107}$$

Proof: Because of the variational principle (Corollary 2.1) and exploiting the representation (102) of S^ν, we can estimate for all e:

$$\begin{aligned} \|KS^\nu e\|_1 &= \min_{e^H} \|S^\nu e - I_H^h e^H\|_1 \;\le\; \|S^\nu e - I_H^h e_C\|_1 \\ &= \|S_{FF}^\nu (e_F - \widehat{e}_F) + \widehat{e}_F - I_{FC} e_C\|_{1,F} \,. \end{aligned}$$

Recalling that $\widehat{e}_F = \widehat{I}_{FC} e_C$, the application of the triangular inequality gives

$$\|KS^\nu e\|_1 \le \|S_{FF}^\nu (e_F - \widehat{e}_F)\|_{1,F} + \|(I_{FC} - \widehat{I}_{FC}) e_C\|_{1,F} \,.$$

Finally, because of (105) and (98), assumption (106) implies

$$\begin{aligned} \|KS^\nu e\|_1 &\le \|S_{FF}\|_{1,F}^\nu \, \|e_F - \widehat{e}_F\|_{1,F} + \|P_{FF}\|_{1,F}^\mu \, \|(I_{FC}^{(0)} - \widehat{I}_{FC})\, e_C\|_{1,F} \\ &\le (\|S_{FF}\|_{1,F}^\nu + \tau \, \|P_{FF}\|_{1,F}^\mu) \, \|e\|_1 \,, \end{aligned}$$

which proves the theorem. △

Clearly, the norms of S_{FF} and P_{FF} in (107) are less than one and depend only on δ but not on A. In particular, the larger δ is, the smaller these norms are. Consequently, the previous theorem shows that, in principle, we can enforce arbitrarily fast two-level convergence by selecting ν and μ accordingly. Moreover, the convergence is *uniform* for $A \in \mathcal{A}$ if we can construct the first guess interpolation, $I_{FC}^{(0)}$, so that (106) is uniformly satisfied for all such A. We will see next that this can be achieved for the same classes of matrices for which the related condition (53) can be uniformly satisfied (which has been discussed in detail in Section 4.2). In fact, the following lemma shows that (106) and (53) are essentially equivalent.

Lemma 5.2 *Consider the two estimates*

$$\text{(a)} \;\; \|e_F - I_{FC} e_C\|_{0,F}^2 \le \tau_1 \, \|e\|_1^2 \,, \quad \text{(b)} \;\; \|(\widehat{I}_{FC} - I_{FC})\, e_C\|_{1,F}^2 \;\le\; \tau_2 \, \|e\|_1^2 \,. \tag{108}$$

If (a) holds for all e and if $\eta \ge \rho(D^{-1}A)$, then (b) holds for all e with $\tau_2 = \eta\tau_1$. If (b) holds for all e and if A_{FF} is strongly diagonally dominant (95), then (a) holds for all e with $\tau_1 = (1 + \sqrt{\tau_2})^2/\delta$.

Proof: We first note that $\rho(D_{FF}^{-1} A_{FF}) \le \rho(D^{-1}A)$:

$$\rho(D_{FF}^{-1} A_{FF}) = \sup_{e_F} \frac{(A_{FF} e_F, e_F)_E}{(D_{FF} e_F, e_F)_E} = \sup_{(e_F,0)} \frac{(Ae,e)_E}{(De,e)_E} \le \sup_{e} \frac{(Ae,e)_E}{(De,e)_E} = \rho(D^{-1}A) \,.$$

Using this and assuming (a) to hold for all e, we obtain because of (32)

$$\|e_F - I_{FC} e_C\|_{1,F}^2 \le \eta \, \|e_F - I_{FC} e_C\|_{0,F}^2 \le \eta\tau_1 \, \|e\|_1^2 \,.$$

Applying this to $\widehat{e}$ rather than e and using (98), gives

$$\|\widehat{e}_F - I_{FC} e_C\|_{1,F}^2 = \|(\widehat{I}_{FC} - I_{FC}) e_C\|_{1,F}^2 \le \eta\tau_1 \, \|\widehat{e}\|_1^2 \le \eta\tau_1 \, \|e\|_1^2$$

which proves the first statement. Regarding the second one, we first estimate for any e

$$\|e_F - I_{FC}\, e_C\|_{1,F} \le \|e_F - \widehat{e}_F\|_{1,F} + \|(\widehat{I}_{FC} - I_{FC})\, e_C\|_{1,F} \le (1+\sqrt{\tau_2})\|e\|_1 \ .$$

By observing that

$$\rho(A_{FF}^{-1}D_{FF}) = \ 1 \,/\, min\{\lambda : \lambda \text{ eigenvalue of } D_{FF}^{-1}A_{FF}\} \le 1/\delta \ ,$$

we conclude that

$$\|e_F - I_{FC} e_C\|_{0,F}^2 \le \rho(A_{FF}^{-1}D_{FF})\, \|e_F - I_{FC} e_C\|_{1,F}^2 \le \frac{1}{\delta}(1+\sqrt{\tau_2})^2\, \|e\|_1^2 \ .$$

This proves the lemma. $\triangle$

According to this lemma, we can use the same interpolation approaches as described in Section 4.2 to define $I_{FC}^{(0)}$. Regarding the practical realisation, we want to make the following remark:

Remark 5.3 The requirement of strong diagonal dominance (95) can most easily be satisfied. If the C/F-splitting and interpolation are constructed according to Theorems 4.3 and 4.6, strong diagonal dominance is even automatically satisfied, namely, with $\delta = 1/\tau$. For instance, the assumptions of Theorem 4.3 imply for all $i \in F$:

$$\begin{aligned} a_{ii} - \sum_{j\in F,\, j\neq i} |a_{ij}| &= s_i + \sum_{j\in P_i} |a_{ij}| + \sum_{j\in C\setminus P_i} |a_{ij}| \\ &\ge s_i + \frac{1}{\tau}\sum_{j\in N_i} |a_{ij}| = \frac{1}{\tau}a_{ii} + (1-\frac{1}{\tau})s_i \ge \frac{1}{\tau}a_{ii} \ . \end{aligned}$$

A second remark refers to the parameter μ. Although the two-level method converges for all $\mu \ge 0$, Theorem 5.1 states fast convergence only if μ is sufficiently large. In practice, however, $\mu > 2$ is hardly ever required (at least if δ is not too small). Nevertheless, each additional relaxation step increases the "radius" of interpolation (causing additional fill-in in the resulting Galerkin operator). Most of the new entries, however, will be relatively small and can be ignored without sacrificing convergence seriously. Consequently, in order to keep the resulting Galerkin operator as sparse as possible, relaxation of interpolation should *always* be combined with a reasonable truncation, performed before the Galerkin operator is computed (cf. Remark 2.4; see also Section 7.2.4). We also note that, in practice, 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). $\ll$

For completeness, the following remarks summarize some algebraic conditions which are equivalent to (106). They are not important for the remainder of the paper, though.

Remark 5.4 Requirement (106) is equivalent to

$$\|(\widehat{I}_{FC} - I_{FC}^{(0)})\, e_C\|_{1,F}^2 \ \le \ \tau^2\, (C_H e_C, e_C)_E \tag{109}$$

for all e_C. This follows immediately by applying (106) to $\widehat{e}$ rather than e and using Lemma 5.1. (109), in turn, is equivalent to

$$\rho(C_H^{-1} J_{CF}^{(0)} A_{FF} J_{FC}^{(0)}) \;=\; \|A_{FF}^{\frac{1}{2}} J_{FC}^{(0)} C_H^{-\frac{1}{2}}\|_E^2 \;\leq\; \tau^2 \tag{110}$$

where, as above, $J_{FC}^{(0)} = I_{FC}^{(0)} - \widehat{I}_{FC}$ and $J_{CF}^{(0)} = (J_{FC}^{(0)})^T$ denote the "errors" of the first guess interpolation and restriction, respectively. $\ll$

Remark 5.5 Denoting the Galerkin operator corresponding to the first-guess interpolation by $A_H^{(0)}$, the requirement that (106) holds uniformly for $A \in \mathcal{A}$ is equivalent to the *spectral equivalence* of C_H and $A_H^{(0)}$ for $A \in \mathcal{A}$,

$$(C_H e_C, e_C)_E \leq (A_H^{(0)} e_C, e_C)_E \leq (1+\tau^2)\,(C_H e_C, e_C)_E\;. \tag{111}$$

In order to see this, one first verifies by a straightforward computation that

$$A_H^{(0)} = C_H + J_{CF}^{(0)} A_{FF} J_{FC}^{(0)}\;.$$

This equality, together with (109), proves the statement. Note that, because of (32), (111) implies that the related (spectral) condition number is uniformly bounded,

$$\kappa\big((A_H^{(0)})^{-1}\, C_H\big) := \frac{\lambda_{max}\big((A_H^{(0)})^{-1}\, C_H\big)}{\lambda_{min}\big((A_H^{(0)})^{-1}\, C_H\big)} \leq 1+\tau^2\;. \qquad \ll$$

We conclude with a remark which stresses the importance of constructing the first guess interpolation reasonably.

Remark 5.6 If we select the first guess interpolation too simply, we can, generally, not expect uniform two-level convergence. For instance, if we just select $I_{FC}^{(0)} = 0$, (110) is equivalent to

$$\rho(C_H^{-1} A_{CF} A_{FF}^{-1} A_{FC}) \;=\; \|A_{FF}^{-\frac{1}{2}} A_{FC} C_H^{-\frac{1}{2}}\|_E^2 \;\leq\; \tau^2\;. \tag{112}$$

For h-discretized elliptic problems, we typically have $\|A_{FF}^{-1}\|_E = O(h^2)$ (provided A_{FF} is strongly diagonally dominant) and $\|A_{FC}\|_E = O(h^{-2})$. Hence, (106) cannot be expected to hold with τ being independent of $h \to 0$; we actually have $\tau = O(h^{-1})$. In order to still obtain uniform convergence, we would need to select $\mu = O(log(h^{-1}))$. $\ll$

5.2 Convergence using full smoothing

The approach discussed in the previous section is not really in the spirit of multigrid since smoothing in the usual sense is not exploited. Two-level convergence is actually obtained by purely algebraic means. In fact, as already pointed out before, the role of F-smoothing is merely to *force* $S^\nu e \approx \widehat{e}$ rather than to smooth the error of the full system. This,

together with Jacobi-interpolation, is a "brute force" approach to make $\|S^\nu e - I_H^h e_C\|_1$ small for all e.

Although this brute force approach helps convergence, in particular in case of "tough problems", we will see in Section 8 that the use of *full* relaxation steps for smoothing usually leads to cycles which are considerably more efficient if computational work is taken into regard. The heuristic reason is that, assuming S to satisfy the smoothing property (36), relatively simple interpolation of the type derived in Section 4.2 is usually sufficient to approximate algebraically smooth error. However, if mere F-smoothing is employed, approximations of the type (58) – as used in Section 4.2 – are too crude and, generally, additional effort needs to be invested to "improve" interpolation by Jacobi-relaxation in order to cope with all those error components which are not affected by mere F-smoothing. In particular, recall that an error $e \in \mathcal{E}$ is not reduced at all by F-smoothing (cf. Remark 5.2).

Unfortunately, Theorem 5.1 does not carry over to the use of general smoothing processes based on full relaxation steps. This is because the proof is based on an estimate of $\|S^\nu e - \widehat{e}\|_1$. The latter, however, can most easily be obtained for F-smoothing but not for smoothing by full relaxation steps, except if we perform relaxation in *CF-ordering*, that is, if we first relax all C-variables and afterwards all F-variables. In this case, Theorem 5.1 trivially carries over, at least for $\nu = 1$, by simply ignoring the C-part of the relaxation. For completeness, we give this result here although it is unrealistic in the sense that it cannot explain the better performance mentioned above.

Corollary 5.2 *Under the same assumptions as in Theorem 5.1, except that smoothing is replaced by one step of Gauss-Seidel relaxation in CF-order, we obtain the following estimate*

$$\|KSe\|_1 \;\leq\; (\|S_{FF}\|_{1,F} + \tau \, \|P_{FF}\|_{1,F}^{\mu}) \, \|e\|_1 \; . \tag{113}$$

Gauss-Seidel CF-relaxation has turned out to be a very efficient smoother in practice. In particular, for positive definite problems, it is usually more efficient than Gauss-Seidel relaxation without any specific order of variables. (Note that CF-relaxation is related to red-black relaxation in geometric multigrid.)

From a practical point of view, (113) is too pessimistic since it implies convergence only if μ is sufficiently large. However, since the *asymptotic* two-level convergence does not depend on whether pre- or post-smoothing is performed, we can conclude from the results in Section 4.1 that the above two-level cycle asymptotically converges *even for* $\mu = 0$ and convergence is uniform for $A \in \mathcal{A}$ provided (106) holds uniformly for all such A.

Compared to the results in Section 4.1, the relevance of Theorem 5.1 and Corollary 5.2 is due to the complementary information. In particular, the fact that Jacobi-interpolation with $\mu > 0$ provides a (purely algebraic) means to improve convergence and that (additional) F-smoothing steps can be used in case of "tough" problems. We will present examples in Section 8 demonstrating the effect of relaxation of interpolation. Various numerical experiments employing F-smoothing and Jacobi-interpolation can also be found in [37].

6 Limits of the theory

The two-level investigations of the previous sections are the basis for the definition of our final *multi-level* method in Section 7. Various results will be presented in Section 8, showing that AMG's V-cycle convergence is, to a large extent, independent of the size of the problem, at least for the geometrically posed problems as considered here.

Unfortunately, the latter cannot be proven in a purely algebraic setting. Since the main reason for this implies some important additional objective which should be taken into regard in AMG's coarsening algorithm – indicated already in Remark 4.6 –, we want to briefly discuss the limitations of the theoretical approach presented before.

First, uniform two-level convergence has strictly been proven only for certain "ideal" classes of positive definite matrices such as M-matrices, essentially positive type matrices, weakly diagonally dominant matrices and some perturbations thereof. Although it is plausible that we can still expect uniform convergence w.r.t. certain larger classes, the uniformity may get lost if the matrices considered are too far off. A special limit case has already been discussed in Example 4.1. It is also clear that we have to expect a degradation if the given problem is still symmetric and positive definite but corresponds to a *system* of PDEs (e.g. from structural mechanics) rather than a scalar PDE. In such cases, generally, the overall approach requires modification: the specific connectivity between different physical quantities needs to be taken into account (at least, all these quantities should be kept separate) in order to still obtain an efficient solution method. Since such problems are not in the focus of this paper, we will not discuss such modifications here but rather refer to the preliminary discussion in [63].

However, apart from such limit cases, AMG's performance in practice turns out to be fairly insensitive to deviations of the underlying matrices from the ideal types. This is important to know, in particular, in recursively extending two-level to real multi-level cycles: even if the given matrix A belongs to one of the ideal classes mentioned above, the recursively defined coarse-level Galerkin operators will generally not. It is often possible to avoid this by particular coarsening strategies. For instance, if A is a weakly diagonally dominant M-matrix, the corresponding Galerkin operator A_H will also be a weakly diagonally dominant M-matrix if coarsening is performed according to Theorem 4.3 with $\tau \leq 2$ (for a proof, we refer to [63]). A similar result can be shown for weakly diagonally dominant matrices $A > 0$ (cf. [33]). However, these results turned out to be not really relevant in practice since they put unnecessary restrictions on the coarsening strategy:

> From experience, allowing faster coarsening and accepting that the coarse-level matrices do not exactly remain in the respective classes, typically leads to much more efficient solution processes.

However, even if all coarse-level matrices belong to one of the ideal classes (for instance, if they are all weakly diagonally dominant M-matrices), the two-level theory does *not* carry over to a multi-level V-cycle theory. To demonstrate this, we consider the following very simple but characteristic counter-example.

Example 6.1 [12, 63] Let A_h be 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. (However, since the concrete boundary conditions are irrelevant for this example, we may ignore the boundary in the following.) One possibility of satisfying the assumptions of Theorem 4.3 with $\tau = 2$ is to assume $h \to 2h$ coarsening and define interpolation to each F-point *strictly one-sided* (with the interpolation weight being 1), see Figure 10.

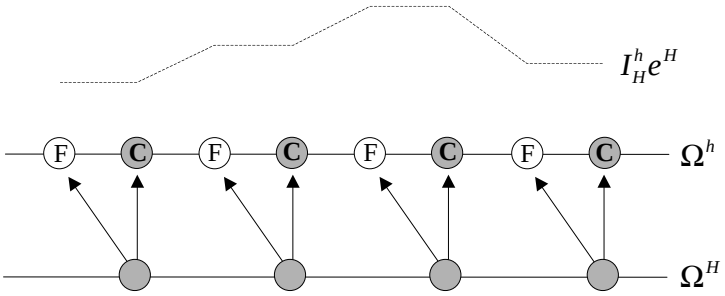


Figure 10: Strictly one-sided interpolation (piecewise constant)

The corresponding coarse-grid operator, A_H, is easily computed to correspond to the difference stencil

$$\frac{1}{(2h)^2}[-4 \quad 8 \quad -4]_{2h}$$

which, after proper scaling of the restriction operator by $1/2$, is seen to be off by a factor of 2 compared to the "natural" $2h$-discretization of $-u''$. Due to this, for a very smooth error frequency, $\sin(\pi x)$, say, we obtain $Ke \approx \frac{1}{2}e$. Consequently, as smoothing hardly effects this frequency (if h is very small), *we cannot expect the asymptotic two-level convergence factor to be better than* $1/2$.

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 operator is off by a factor of 2 compared to the previous one. A simple recursive argument – applied to the same error frequency as above – 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 h-dependent.* $\ll$

The major cause of the problem seen here is that the interpolation is only *piecewise constant* (first order) which, obviously, is insufficient to ensure h-independent V-cycle convergence. (We will consider piecewise constant interpolation again in the context of aggregation-based AMG in Section 9, where the basic problem with piecewise constant interpolation will become clear; see Section 9.1.) Note that one-sided interpolation has been artificially introduced in this particular example to demonstrate the consequences. In practice, each F-point should, of course, be interpolated from *both* its C-neighbors, leading to linear interpolation (which, in this 1D case, even gives a direct solver). In any case, Theorem 4.3 formally allows such interpolations, and h-dependent V-cycle convergence

always has to be expected whenever essentially one-sided interpolation is employed to solve analogous problems, in any dimension.

The main hurdle in extending the two-level to a V-cycle theory is due to the fact that the basic algebraic condition for interpolation, (53) (and hence also (106)), is too weak to imply uniform V-cycle convergence (cf. Remark 4.2). In [63], the stronger requirement

$$\|e_F - I_{FC} e_C\|_{0,F}^2 \leq \tau \|e\|_2^2 \tag{114}$$

has been discussed which is more suited for an algebraic V-cycle convergence theory. Indeed, following the same arguments as in Remark 4.2, one sees that interpolation based on (114) is related to *second order* interpolation (assuming an adequate geometric problem). Unfortunately, it hardly seems possible to construct interpolation so that (114) is satisfied exactly by using only algebraic information such as the matrix entries.

In practice, however, it turned out that potential problems with interpolation can most easily be avoided to a large extent, at least for all applications considered here. In fact, for geometrically posed applications, a sufficient improvement of the accuracy of interpolation (compared to the worst-case as considered above) is generally obtained, by just arranging the C/F-splittings as uniformly as possible (based on the connectivity information contained in the matrices) so that each F-point is reasonably surrounded by its interpolatory neighbors. The importance of this has already been stressed in Remark 4.6. Of course, if there is no geometrical background of a given problem or if the underlying connectivity structure is far away from being local, there is no a priori guarantee of highest efficiency.

The application of one Jacobi relaxation step to a given interpolation is another simple (but generally more costly) way of improvement. In the above example, for instance, this would immediately "overwrite" the given piecewise constant interpolation by linear interpolation. Although, in general, relaxation of interpolation will not be able to increase the *order* of interpolation, it tends to substantially improve it.

The most popular way to overcome h-dependent V-cycle convergence is to use "better" cycles like F- or W-cycles. However, apart from the fact that such cycles are more expensive (which may be considerable in AMG, depending on the actual coarsening), they will, at best, have the same convergence factor as the corresponding two-level method which, in turn and for the same reasons which may lead to h-dependent V-cycle convergence, might not be fully satisfactory. That is, although better cycles may well be a pragmatic way to overcome a convergence problem, they tend to hide the true reasons for such problems and should be only a possibility of second choice.

Although not needed for the type of applications considered here, we finally want to mention that one can imagine other strategies to improve interpolation. For instance, by exploiting a minimum amount of geometric information (e.g. point locations). More algebraically, local fitting of interpolation weights provides various possibilities to better approximate smooth error (cf. [63]), for instance, fitting based on some "test vector(s)" provided by the user upon calling AMG. In general, however, such "sophisticated" techniques are rather complicated and tend to be computationally expensive.

7 The AMG algorithm

The application of AMG to a given problem is a two part process. The first part, a fully automatic *setup phase*, consists of recursively choosing the coarser levels and defining the transfer and coarse-grid operators. The second part, the *solution phase*, just uses the resulting components in order to perform normal multigrid cycling until a desired level of tolerance is reached (usually involving Gauss-Seidel relaxation for smoothing). The solution phase is straightforward and requires no explicit description.

This section describes algorithmic components used in the setup phase of the code RAMG05 mentioned before. According to Section 2.1, only the C/F-splitting and the interpolation, I_{FC}, need to be explicitly defined. These definitions closely follow the approaches suggested by the analysis contained in Sections 4.2 and 4.3. Restriction is taken as the transpose of interpolation (10) and the computation of the coarse-level Galerkin operators (5) is straightforward. There is still much room for modifications and further enhancements, but the algorithmical components proposed here have been tested for a wide variety of problems and have been found to lead to robust and efficient solution processes. Typical results will be presented in Section 8.

We point out that the algorithm described below does not exploit symmetry, except that restriction is *always* taken as the transpose of interpolation (which is not necessarily the best for non-symmetric problems, see Section 2.3). Without any modification, the algorithm has been applied to various non-symmetric problems. Practical experience indicates that, generally, the non-symmetry by itself does not necessarily cause particular problems for AMG. Other properties of the underlying matrices, such as a strong violation of weak diagonal dominance, seem to influence the performance of AMG (as it stands) to a much larger extent.

In the following, we will first describe the splitting process (Section 7.1) and afterwards the interpolation (Section 7.2). The approach for constructing the splitting and the interpolation is the same for all levels of the AMG hierarchy. Therefore, the following description will be for any fixed level. Clearly, all of the following has to be repeated recursively for each level until the level reached contains sufficiently few variables (for a direct solve, say). All quantities occuring will actually depend on the level, but the index will be omitted for convenience.

To make the current section easier to read and more self-contained, we keep references to previous sections to a minimum. Instead, we repeat the most relevant aspects.

7.1 Coarsening

In order to achieve fast convergence, algebraically smooth error needs to be approximated well by the interpolation. As a rule, this can be achieved the better the stronger the F-to-C connectivity is. On the other hand, the size of the coarse-level operator (and the time to compute it) strongly depends on the number of C-variables. Since the overall efficiency is determined by both the speed of convergence and the amount of work needed per cycle (which is directly related also to the total memory requirement), it is absolutely imperative to limit the number of C-variables while still guaranteeing that all F-variables are sufficiently strongly connected to the C-variables.

However, the goal should not just be to minimize the total number of C-points, say.

According to Remark 4.6 and the related discussion in Section 6, an important objective is to create C/F-splittings which are as uniform as possible with F-variables being "surrounded" by C-variables to interpolate from. Although there is no algebraic proof, interpolation tends to be considerably better, resulting in much faster convergence, if this objective is taken into regard. A simple algorithm is described in Section 7.1.1.

Requiring strong F-to-C connectivity does not necessarily mean that all F-variables need to have strong *direct* connections to C-variables. In general, strong connectivity may be via strongly connected neighboring F-variables (cf. Section 4.3). This leads to "aggressive" coarsening strategies as described in Section 7.1.2. Such strategies allow for a drastic reduction of the setup and cycle cost, the complexity of the coarse-level operators as well as the memory requirement. Clearly, these benefits will be at the expense of a reduced convergence speed since smoothing becomes less efficient and since it becomes more difficult to "match the ranges" of the smoothing and the interpolation operators. In practice, however, it has turned out that this disadvantage is usually more than made up for by the benefits, in particular, if AMG is used as a pre-conditioner rather than stand-alone (cf. Section 7.3). We will present examples in Section 8.

7.1.1 Standard coarsening

In this section, we consider C/F-splittings based on *direct* couplings: each F-variable i is required to have a minimum number of those of its couplings $j \in N_i$ be represented in C which affect the error at i most, that is, for which $|a_{ij}|$ is largest in some sense ("strong connections").

For all applications we have in mind here, by far most of the strong couplings are *negative* and we first describe a fast procedure which generates a C/F-splitting taking only negative couplings into account (regarding positive couplings, see Section 7.1.3). That is, the resulting C/F-splitting will be such that all F-variables have a substantial (direct) *negative* connectivity to neighboring C-variables. In other words, we essentially coarsen in directions *in which algebraically smooth error changes slowly* (cf. Section 3.3).

To be more specific, let us define a variable i to be *strongly negatively coupled* (or *strongly n-coupled*) to another variable, j, if

$$-a_{ij} \geq \varepsilon_{str} \max_{a_{ik}<0} |a_{ik}| \quad \text{with fixed} \quad 0 < \varepsilon_{str} < 1 \tag{115}$$

and let us denote the set of all strong n-couplings of variable i by S_i,

$$S_i = \{j \in N_i : \ i \text{ strongly n-coupled to } j\} \ . \tag{116}$$

(Note that *all* positive connections are regarded as *weak* at this point.) According to practical experience, the concrete value of ε_{str} is not critical, $\varepsilon_{str} = 0.25$ being a reasonable default value. Since the relation of variables being strongly n-coupled is generally non-symmetric (even if A is symmetric), we introduce the set S_i^T of strong *transpose* n-couplings of i consisting of all variables j which are strongly n-coupled to i:

$$S_i^T = \{j \in \Omega : i \in S_j\} \ .$$

The proposed simple splitting algorithm corresponds to the "preliminary C-point choice" as described in [63]. Essentially, one starts with defining some first variable,

i, to become a C-variable. Then all variables, j, which are strongly n-coupled to i (i.e. all $j \in S_i^T$) become F-variables. Next, from the remaining undecided variables, another one is defined to become a C-variable and all variables which are strongly n-coupled *to it* (and which have not yet been decided upon before) become F-variables. This process is repeated until all variables have been taken care of.

The only problem is that – in order to avoid randomly distributed C/F-patches but rather obtain reasonably uniform distributions of C- and F-variables – we need to perform this process in a certain order. In order to ensure that there is a tendency to build the splitting starting from one variable and continuing "outwards" until all variables are covered, we introduce a "measure of importance", λ_i, of any undecided variable i to become the next C-variable. We define

$$\lambda_i = |S_i^T \cap U| + 2|S_i^T \cap F| \quad (i \in U)$$

where U, at any stage of the algorithm, denotes the current set of undecided variables. (For any set P, $|P|$ denotes the number of elements it contains.) λ_i acts as a measure of how valuable a variable $i \in U$ is as a C-variable, given the current status of C and F. Initially, variables with many others strongly n-coupled to them become C-variables, while later the tendency is to pick as C-variables those on which many *F-variables* strongly depend.

The complete algorithm is outlined in Figure 11. We point out that the measure λ_i has to be computed globally only once at the beginning of the algorithm. At later stages, it just needs to be updated locally. For isotropic 5-point and 9-point stencils, the first coarsening steps are illustrated in Figure 12.

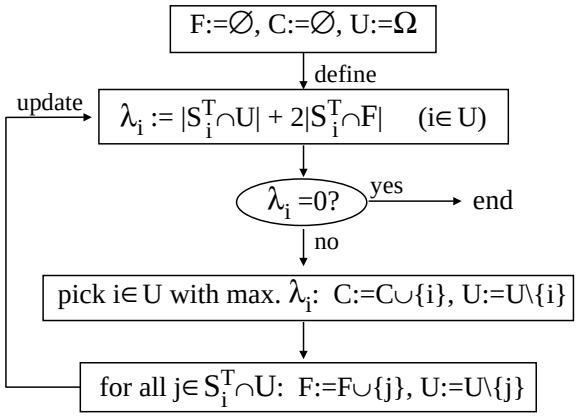


Figure 11: Standard coarsening algorithm [63]

Remark 7.1 Before the above algorithm starts, variables which have no connection at all (e.g., resulting from Dirichlet boundary points which have not been eliminated from the system) are filtered out and become F-variables. Trivially, such variables do not require interpolation. Similarly, variables which correspond to (very) strongly diagonally dominant rows of the matrix might be filtered out at this point. ≪

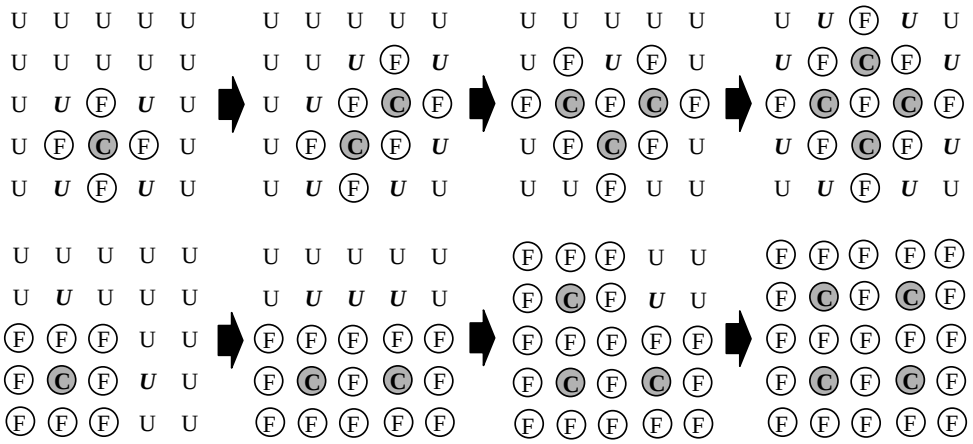


Figure 12: First steps of the standard coarsening process in case of isotropic 5-point (top) and 9-point stencils (bottom). At each stage, those undecided points with highest λ-value are shown in bold-italics.

Remark 7.2 After termination of the above algorithm, all F-variables have (at least) one strong n-coupling to a C-variable (except for the "trivial" ones taken out at the very beginning, see Remark 7.1). However, there may be a few U-variables left, in particular, in non-symmetric problems. Such particular variables have the property that they are not strongly n-coupled to any of the C-variables (otherwise they would have become F-variables earlier in the process). Moreover, such variables have no strong n-connection among each other nor is any F-variable strongly n-coupled to any of them (otherwise their measure λ_i would be non-zero). However, each of these U-variables is strongly n-coupled to (at least) one of the F-variables. We therefore re-define all potentially remaining U-variables to become F-variables. In interpolation, they will be interpolated *indirectly* via their strong F-couplings (see Section 7.2.1). $\ll$

Remark 7.3 None of the C-variables is strongly n-coupled to any of those C-variables created prior to itself in the coarsening process described above. However, since the relation of being strongly n-coupled is not necessarily symmetric, this may not fully be true the other way around. In any case, however, the resulting set of C-variables is close to a *maximal* set of variables which are not strongly n-connected among each other (see Remark 4.6). $\ll$

Remark 7.4 The theoretical investigation of special processes such as F-smoothing (Section 5.1.2) and relaxation of interpolation (Jacobi-interpolation, Section 5.1.3) was based on the assumption that the submatrices A_{FF} are strongly diagonally dominant. Clearly, if required, this condition can be exactly satisfied during the coarsening step or, most easily, by adding a few C-points afterwards (if necessary at all). However, for those applications considered in this paper, the above coarsening algorithm tends to ensure diagonal dominance to a sufficient extent without any modification. Therefore, by default, we do not explicitly check for strong diagonal dominance. $\ll$

7.1.2 Aggressive coarsening

In many PDE applications, we have to deal with small stencils. In such cases, the previous splitting algorithm, because it is based on direct connections, may cause a relatively high complexity (memory requirement due to the coarse-level Galerkin operators). For instance, isotropic 7-point stencils on regular 3D meshes, will cause the first coarser level to correspond to the black points of a red-black coarsened grid (as in the 2D case depicted in Figure 12, upper picture). One easily sees that the Galerkin operator on this level corresponds to a 19-point stencil. That is, the Galerkin *matrix* is larger than the original matrix by a factor of 1.36. Although subsequent coarsening will typically become faster (simply because the corresponding stencils are larger), the first coarsening step significantly contributes to the final complexity. Complexity can substantially be reduced by employing "aggressive coarsening".

In order to allow aggressive coarsening, we extend the definition of strong connectivity to also include variables which are not directly coupled. Following [63], we introduce the concept of *long-range strong n-connections*: A variable i is said to be strongly n-connected to a variable j *along a path of length* ℓ if there exists a sequence of variables $i_0, i_1, ..., i_\ell$ with $i = i_0$ and $j = i_\ell$ such that $i_{k+1} \in S_{i_k}$ for $k = 0, 1, 2, ..., \ell - 1$. With given values $p \geq 1$ and $\ell \geq 1$, we then define a variable i to be *strongly n-connected to a variable* j *w.r.t.* (p, ℓ) if at least p paths of length $\leq \ell$ exist such that i is strongly n-connected to j along each of these paths (in the above sense).

In principle, for any given p and ℓ, the splitting algorithm described in the previous section immediately carries over if we apply it to the set

$$S_i^{p,\ell} = \{j \in \Omega : \ i \text{ strongly n-connected to } j \text{ w.r.t. } (p, \ell)\} \tag{117}$$

rather than S_i (116). From a practical point of view, however, it generally does not pay to exploit strong n-connectivity in this generality. In fact, the cases $p = 2, \ell = 2$ and $p = 1, \ell = 2$ turn out to be the most useful. Moreover, it hardly ever pays to use aggressive coarsening on more than one level. (On all but the first level, standard coarsening is usually efficient enough.) We will refer to the coarsening strategies corresponding to $S_i^{2,2}$ and $S_i^{1,2}$ as *A2-* and *A1-coarsening*, respectively.

Remark 7.5 Applying the splitting algorithm directly to $S_i^{p,\ell}$ instead of S_i, requires the storage of the complete connectivity information (and the corresponding transpose information contained in $(S_i^{p,\ell})^T$) for each variable i. Even for the cases $S_i^{2,2}$ and $S_i^{1,2}$ considered here, this may be quite substantial but can be avoided to a large extent by *applying the standard coarsening algorithm twice*: In the first step, it is applied exactly as described in the previous section. Then, instead of (117), we define strong n-connectivity *only between the resulting C-variables* (via neighboring F-variables). That is, for each variable $i \in C$, we define

$$\widehat{S}_i^{p,\ell} = \{j \in C : \ i \text{ strongly n-connected to } j \text{ w.r.t. } (p, \ell)\} \ . \tag{118}$$

Using this definition, the standard coarsening algorithm is now applied a second time but restricted to the set of C-variables. The subset of "new" C-variables resulting from this second step will then be used as the next coarser level. $\ll$

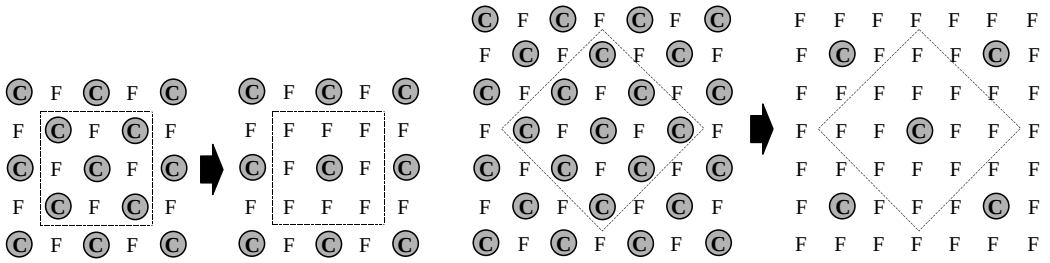


Figure 13: Results of aggressive A2 (left) and A1 coarsening (right) in case of isotropic 5-point stencils. The dashed boxes depict the range of strong connectivity in the sense of $\widehat{S}_i^{2,2}$ and $\widehat{S}_i^{1,2}$, respectively.

Clearly, A1- is faster than A2-coarsening. As an example, Figure 13 illustrates the second step of the two-step process described in the previous remark for isotropic 5-point stencils. Generally, while A2-coarsening is effective only in (at least) "plane-wise" isotropic problems, A1 is also effective in strongly anisotropic cases. In illustration, Figure 14 shows the result of using A2 and A1 coarsening in case of the example discussed in Section 1.3. In case of strategy A2 (left picture), the coarsening pattern is essentially the same as for standard coarsening (see Figure 4) except in the lower left quarter where the problem is isotropic and coarsening is faster. Strategy A1, on the other hand, also speeds coarsening up in the anisotropic areas. Indeed, the right picture in the figure shows that coarsening now is substantially faster everywhere: it is still essentially in the direction of strong couplings, coarse-level points are further apart than before though.

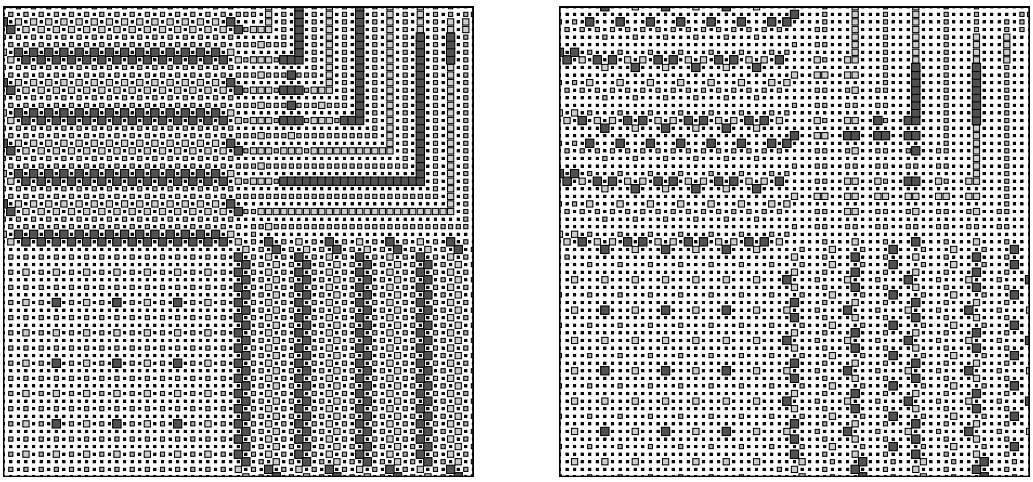


Figure 14: The finest and three consecutive AMG levels if aggressive A2 (left) and A1 coarsening (right) is applied (only) on the first level

Remark 7.6 If aggressive coarsening is used, strong diagonal dominance of the corresponding submatrices A_{FF} (theoretically required if F-smoothing or Jacobi-interpolation is to be employed in the sense of Section 5.1), can no longer be assumed in the strict sense, at least not for each row. $\ll$

7.1.3 Strong positive connections

The previous approaches to construct a C/F-splitting were based on negative couplings only. Provided that potentially existing *positive* couplings are relatively small, they can, indeed, be ignored in coarsening and interpolation (cf. the related discussion in Section 4.2). However, this cannot always be assumed and we have to allow for matrices which also contain some strong *positive* entries.

According to Theorem 4.6, a more general splitting process should ensure that, for all F-variables which have strong negative *and* positive couplings, a minimum number of *both* types of couplings is represented in C. However, to construct such a splitting within one step, turns out to be relatively complicated. Since, for all problems we have in mind here, far most strong connections are negative, we propose a very simple alternative:

After one of the coarsening processes described before has been applied, we test for all F-variables i whether or not there exist strong positive F-to-F couplings. For instance, we simply check whether

$$a_{ij} \geq \varepsilon_{str}^{+} \max_{k \neq i} |a_{ik}| \tag{119}$$

holds for some $j \neq i$. Here, ε_{str}^{+} is some reasonable tolerance, for instance, $\varepsilon_{str}^{+} = 0.5$. If such a j exists, *all* j's satisfying (119) will *a posteriori* be added to the set S_i (116) (this will affect the performance of the interpolation routines, see Section 7.2) and the variable which corresponds to the largest positive coupling, will be re-defined to become a C-variable. (Regarding an example demonstrating the effect of this process, we refer to Section 8.4.3.)

Clearly, this a posteriori update of the C/F-splitting is suitable only if there are not too many strong positive connections. Otherwise, one has to change the original splitting algorithm as mentioned above.

7.2 Interpolation

In defining interpolation to the currently finest level, we assume that a C/F-splitting has been constructed either by means of standard or aggressive coarsening. In the first case, interpolation is used as described in Section 7.2.1 (*direct* or *standard interpolation*). In the second case, interpolation is used as described in Section 7.2.2 (*multi-pass interpolation*). In both cases, interpolation can optionally be improved further by means of additional relaxation steps (Section 7.2.3, *Jacobi-interpolation*).

Some of the interpolation variants described in the following exploit strong *indirect* C-couplings which will increase the "radius" of interpolation. In all such cases, it is important to reasonably truncate the resulting interpolation operator before computing the Galerkin operator (see Section 7.2.4).

The following abbreviations will be needed below. Here, S_i is as defined in (116), possibly modified a posteriori as described in Section 7.1.3:

$$\begin{aligned} C_i &= C \cap N_i\,, \quad C_i^s = C \cap S_i\,, \\ F_i &= F \cap N_i\,, \quad F_i^s = F \cap S_i\,. \end{aligned}$$

7.2.1 Direct and standard interpolation

The following procedures apply in case the C/F-splitting has been constructed by means of standard coarsening.

Direct interpolation

In the simplest case, the definition of interpolation, as described in Section 4.2.3, is applied immediately. More precisely, for each $i \in F$, we define the set of interpolatory variables by $P_i = C_i^s$ and approximate

$$a_{ii}e_i + \sum_{j\in N_i} a_{ij}e_j = 0 \quad \Longrightarrow \quad a_{ii}e_i + \alpha_i \sum_{k\in P_i} a_{ik}^- e_k + \beta_i \sum_{k\in P_i} a_{ik}^+ e_k = 0 \tag{120}$$

with

$$\alpha_i = \frac{\sum_{j\in N_i} a_{ij}^-}{\sum_{k\in P_i} a_{ik}^-} \quad \text{and} \quad \beta_i = \frac{\sum_{j\in N_i} a_{ij}^+}{\sum_{k\in P_i} a_{ik}^+} \, .$$

This immediately leads to the interpolation formula

$$e_i = \sum_{k\in P_i} w_{ik}\, e_k \quad \text{with} \quad w_{ik} = \begin{cases} -\alpha_i\, a_{ik}/a_{ii} & (k \in P_i^-) \\ -\beta_i\, a_{ik}/a_{ii} & (k \in P_i^+) \end{cases} \, . \tag{121}$$

If $P_i^+ = \emptyset$, this formula is modified according to Section 4.2.2, that is, we set $\beta_i = 0$ and add all positive entries, if there are any, to the diagonal. Since this interpolation involves only direct connections of variable i, we will refer to it as *direct interpolation* later on.

Remark 7.7 The above procedure can be applied as long as $C_i^s \neq \emptyset$. However, this is ensured by the standard coarsening algorithm for all F-points i with the potential exception of just a few of them (see Remark 7.2). Such "exceptional" F-points, however, necessarily have at least one strong connection to a "regular" F-point, and will be interpolated indirectly as described next. $\ll$

Standard interpolation

The standard coarsening strategy ensures that there is a strong F-to-C connectivity. However, it does not strictly enforce what actually is required by the two-level theory, namely, that each F-variable should have a *fixed percentage* of its total connectivity be reflected in C (defined by τ, see Section 4.2). Although this is usually not a problem in practice (since the coarsening algorithm by itself usually ensures a sufficient F-to-C connectivity), we can make up for this in a simple way: We modify the previous direct interpolation so that, for each $i \in F$, its strong *F-connections* are also (indirectly) included in interpolation.

That is, instead of immediately approximating the i-th equation (left equation in (120)), we first (approximately) eliminate all e_j ($j \in F_i^s$) by means of the corresponding j-th equations. More specifically, for each $j \in F_i^s$, we replace

$$e_j \longrightarrow -\sum_{k\in N_j} a_{jk}e_k/a_{jj} \tag{122}$$

resulting in a new equation for e_i,

$$\widehat{a}_{ii}e_i + \sum_{j\in\widehat{N}_i} \widehat{a}_{ij}e_j = 0 \quad \text{with} \quad \widehat{N}_i = \{j \neq i : \widehat{a}_{ij} \neq 0\} \ . \tag{123}$$

By defining P_i as the union of C_i^s and all C_j^s ($j \in F_i^s$), we now define interpolation exactly as in (120)-(121) with all a's replaced by $\widehat{a}$'s and N_i replaced by $\widehat{N}_i$.

This modification usually enhances the quality of interpolation substantially (see Example 7.1 below), the main reason being that the type of approximation (120), if applied to the "extended" equation (123), introduces less error. Moreover, it further contributes to the objective of having F-variables largely be "surrounded" by interpolatory variables. This modified interpolation will be referred to as *standard interpolation* below.

Example 7.1 In illustration, consider the same case as in Example 6.1 except that the coarse grid is assumed to be created by $h \to 3h$ coarsening, see Figure 15. (This is just for ease of demonstration; the standard coarsening process described in Section 7.1.1 would not really create this coarsening.)

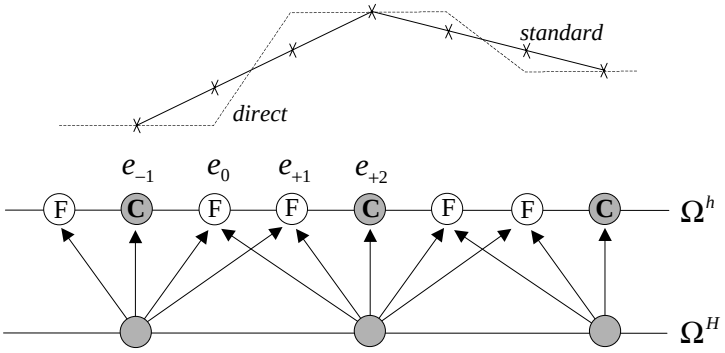


Figure 15: Direct versus standard interpolation

Direct interpolation in this situation would obviously give piecewise constant interpolation (dashed line in the figure) which, as we know from Example 6.1, is not quite satisfactory. (Compared to the case considered in Example 6.1, the Galerkin operator here is even off by a factor of 3 rather than 2.) Standard interpolation, on the other hand, can easily be seen to correspond to linear interpolation. For instance, the interpolated value for e_0 is computed from the equation $-e_{-1} + 2e_0 - e_1 = 0$ by substituting

$$e_1 \longrightarrow (e_0 + e_2)/2 \ ,$$

giving $e_0 = \frac{2}{3}e_{-1} + \frac{1}{3}e_2$. $\ll$

Of course, direct and standard interpolation processes may also be mixed in a straightforward way. That is, standard interpolation is used only for variables i for which, based on some reasonable criterion, the (direct) F-to-C connectivity appears to be too low. However, for simplicity, such mixed interpolation will not be considered here. Moreover, for critical F-variables i, one might be tempted to eliminate *all* e_j ($j \in F_i$) (rather than just

the *strong* F-neighbors) and to use the union of C_i and all C_j ($j \in F_i$) as P_i. However, taking computational work into account, this *extended interpolation* is rarely ever advantageous and will not be discussed further.

Remark 7.8 Apart from other minor differences, interpolation in AMG1R5 was a compromise between the direct interpolation and the standard interpolation described above. There, an attempt was made to replace e_j ($j \in F_i^s$) by averages *involving only variables in* C_i^s. That is, the goal was to improve the direct interpolation *without* increasing the set of interpolatory variables P_i. If it turned out that this was not possible, based on certain criteria, new C-variables were added to the splitting, *increasing* C_i^s *a posteriori*. Although this approach works quite well in many situations, it has two drawbacks.

First, an a posteriori introduction of additional C-variables may turn a fairly regular C/F-splitting (produced by the coarsening algorithm) into a quite disturbed one which, during subsequent coarsening steps, may lead to more irregular and more complex Galerkin operators. In fact, in complex 3D situations, many additional C-variables are typically introduced a posteriori often causing unacceptably high complexities (see Section 8 for examples). Second, the above-mentioned replacement of the e_j's by averaged values was motivated by geometric arguments. In fact, it works very well in case of matrices which are close to M-matrices and are related to regular geometric situations. However, it may substantially deteriorate in other cases.

In practice, the standard interpolation as described above – extending the interpolation pattern on a *fixed* set C followed by a truncation (see Section 7.2.4) – has turned out to be more robust and often considerably more efficient. $\ll$

7.2.2 Multi-pass interpolation

The following interpolation procedure applies in case the C/F-splitting has been constructed by means of aggressive coarsening. It proceeds in several passes, using direct interpolation whenever possible and, for the remaining variables, exploiting interpolation formulas at neighboring F-variables. (This corresponds to the approach described in Section 4.3.) The individual passes are as follows:

1. Use direct interpolation (Section 7.2.1) to derive formulas for all $i \in F$ for which $C_i^s \neq \emptyset$ and define the set $F^\star$ to contain all these variables. If $F^\star = F$ stop, otherwise proceed.

2. For all $i \in F \setminus F^\star$ for which $S_i \cap F^\star \neq \emptyset$ do the following: Take the i-th equation (left equation in (120)) and, for all $j \in S_i \cap F^\star$, replace

$$e_j \longrightarrow \sum_{k \in P_j} w_{jk} e_k$$

leading to a new equation (123) for e_i. Defining the set of interpolatory variables P_i as the union of all P_j for $j \in S_i \cap F^\star$, an interpolation formula then is computed exactly as in the case of standard interpolation. If all such variables i have been processed, update $F^\star$ to also include all variables which have obtained an interpolation formula during this pass.

3. If $F^\star = F$ stop. Otherwise go back to step 2.

Using the agressive (A1 or A2) coarsening strategy as described in Remark 7.5, this process can be shown to terminate after at most four passes. Note that the update of $F^\star$ is done in a Jacobi (not Gauss-Seidel) fashion. This is done to preserve the locality of interpolation. We will refer to this interpolation as *multi-pass interpolation*.

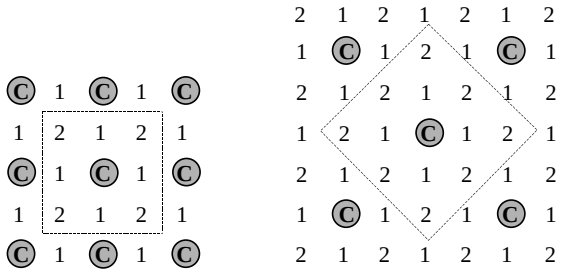


Figure 16: Multi-pass interpolation for isotropic 5-point problems (A2- and A1-coarsening)

Example 7.2 Figure 16 illustrates multi-pass interpolation in case of A2- and A1-coarsening, applied to the 5-point Poisson stencil (cf. Figure 13). F-points marked by "1" and "2" are those which are interpolated in the first and second pass, respectively. In case of A2-coarsening (left picture), the resulting interpolation can easily be seen to be linear. For A1-coarsening (right picture), we obtain constant interpolation at all points marked by "1", and linear interpolation at the remaining points (see also Example 7.3 below). $\ll$

7.2.3 Jacobi-interpolation

Given any of the previous interpolation formulas, we can optionally improve it by a posteriori applying Jacobi-relaxation as formally described in Section 5.1.3. More explicitly, one step of this Jacobi-relaxation – proceeding from iteration $\mu - 1$ to iteration μ (where $\mu = 0$ corresponds to the given interpolation) – proceeds as follows.

For all variables $i \in F$ in turn, take the i-th equation (left equation in (120)) and, for *all* $j \in F_i$, replace

$$e_j \longrightarrow \sum_{k \in P_j} w_{jk}^{(\mu-1)} e_k \tag{124}$$

which leads to a new equation (123) for e_i. Defining the set of interpolatory variables P_i as the union of C_i and all P_j for $j \in F_i$, the μ-th interpolation formula then is obtained exactly as in the case of standard interpolation.

Clearly, only one or (at most) two steps are practical. Depending on the situation, relaxation of interpolation may enhance convergence considerably. Often, however, the additional cost for computing this interpolation does not pay if total computational work is taken into account. Clearly, one may save a substantial amount of work by applying relaxation of interpolation only locally wherever it appears to be reasonable. This is not done in the following.

The interpolation as described above will be referred to as *fully relaxed Jacobi-interpolation*. In most cases, it will be sufficient to use the replacement (124) only for $j \in F_i^s$, that is, only for the *strongly* connected F-variables. Accordingly, the set P_i is then selected as the union of C_i^s and all P_j for $j \in F_i^s$. This will be referred to as *partially relaxed Jacobi-interpolation*.

Example 7.3 Consider the same case as in the previous Example 7.2. We have seen there that multi-pass interpolaton, applied to the A1-coarsened grid (right picture in Figure 16), gives constant interpolation at those points marked by "1", and linear interpolation at the remaining points. One easily sees that, applying Jacobi-relaxation just to the points marked by "1" yields linear interpolation everywhere. $\ll$

7.2.4 Truncation of interpolation

For both standard as well as Jacobi-interpolation, the sets P_i of interpolatory variables may become quite large. This is, in particular, true for the Jacobi-interpolation since each relaxation step introduces, roughly, a "new layer" of additional C-variables to be used for interpolation. Consequently, even if only one Jacobi step is applied at each AMG level, the resulting Galerkin operators will substantially increase towards coarser levels. This process, without reasonable truncation, will generally be much too costly.

However, interpolation weights corresponding to variables "far away" from variable i will usually be much smaller than the largest ones. Before computing the coarser-level Galerkin operator, we therefore always truncate the full interpolation operator by ignoring all interpolatory connections which are smaller (in absolute value) than the largest one by a factor of ε_{tr} and re-scale the remaining weights so that the total sum remains unchanged. In practice, a value of $\varepsilon_{tr} = 0.2$ is usually taken.

Remark 7.9 If interpolation contains substantial positive *and* negative weights, one should truncate and re-scale positive and negative weights *separately* (analogously to the definition of interpolation (120).) Otherwise, convergence may substantially degrade. $\ll$

Remark 7.10 One might be tempted to truncate the *Galerkin operator* rather than the interpolation operator. In fact, this would formally give more control on the growth of the coarse-level operators. However, we have already pointed out in Remark 2.4, that this may cause serious convergence problems if not applied with great care. $\ll$

7.3 AMG as pre-conditioner

In order to increase the robustness of standard 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 [72] or GMRES [64, 65]. In the simplest case, complete multigrid cycles are merely used as pre-conditioners [34, 49]; in more sophisticated approaches, acceleration is even used on the individual grids of the hierarchy [50, 17]. 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 optimise 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, efficiently operates on *all* error components, short-range as well as long-range. This has the implication that, instead of using AMG stand-alone, it is generally more efficient to put less effort into the (expensive) setup phase and use AMG as pre-conditioner, for example, by using aggressive coarsening strategies (cf. the applications in Section 8).

In this context, we also point out that, 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. And even then, there is no final guarantee that such situations can be avoided. (We note that this even happens with "robust" geometric multigrid methods, see, for instance, Remark 8.10.)

8 Applications

In this section we will demonstrate the efficiency and robustness of AMG in solving second order elliptic differential equations. All results presented have been obtained by the code RAMG05 described in the previous section. Although the strength of RAMG05 is its direct applicability to geometrically complex problems, we will often consider selected model problems on simple geometries in quite some detail. Such model problems certainly do not give the full picture, but they most easily allow the investigation of AMG's asymptotic behavior as well as its dependence on various specific aspects such as anisotropies, discontinuities, singular perturbations and the like. Practical experience has shown that AMG's performance in geometrically complex situations, in 2D as well as 3D, is very comparable to that in related model situations. We will present some typical examples demonstrating this.

We have already pointed out before that it is not at all sufficient to merely look at AMG's convergence behavior in order to judge its performance. Definitely, useful comparisons have to take both *computing times* and *memory requirements* into account. Having this in mind, we will compare the influence of different algorithmical components (such as type of interpolation and speed of coarsening) and solution approaches (stand-alone versus accelerated cycles) on the performance. Moreover, in order to quantify the overall efficiency, we will make some comparisons with well-known standard (one-level) solution methods such as ILU pre-conditioned conjugate gradient ("cg"). However, we want to point out that the purpose of these comparisons is merely to give a first indication, they are not suited to give a final picture:

First, many variants and improvements of ILU pre-conditioned cg methods are available and here we just focus on simple and best-known strategies. In particular, our comparisons are not meant to judge the performance of such classical methods in general. Second, RAMG05 is still under development and is continuously being enhanced and generalized. In particular, RAMG05 is far from being optimized. In fact, this code has not been designed for highest efficiency but rather for flexibility in testing and extending the method. In particular, the setup cost may substantially be reduced. Depending on the concrete approach, 50% savings or even more seem realistic. Moreover, our main interest here is to demonstrate typical trends in the influence of different algorithmical components. For simplicity, these components are implemented as "fixed" strategies, that is, they are not locally adjusted to particular requirements of the given problem. For example, if Jacobi-relaxation of interpolation is performed, it is always applied "globally". In many situations, however, a local application, controled by some reasonable measure (e.g. based on the total strength of C-connectivity found at an F-point), may give similar convergence improvements at much lower cost and memory. Similarly, if aggressive coarsening is performed, it is done everywhere. Thus, there is much room for quite substantial optimisations. Nevertheless, the results indicate that the code is very efficient even as it stands.

In the sequel, for brevity, we will refer to "AMG" rather than to "RAMG05". However, one should keep in mind that there is an ongoing rapid development of new AMG approaches and variants and that there is no unique and best AMG approach yet.

8.1 Default settings and notation

Various parameters have to be set to define AMG's setup and cycle (see Section 7). Unless explicitly stated otherwise, we use the following default settings and procedures:

- $\varepsilon_{str} = 0.25$ to define *strong connectivity* (Section 7.1.1).
- $\varepsilon_{tr} = 0.2$ to define *truncation of interpolation* (Section 7.2.4).
- Coarsening is terminated if the number of variables on the coarsest level drops below 40. The coarsest-level equations are solved by direct Gauss elimination.
- Smoothing is done by Gauss-Seidel relaxation, one pre- and one post-smoothing step being the default. Unless explicitly stated otherwise, the order of relaxation is "CF", that is, first all C-variables are relaxed and then all F-variables. (This corresponds to red-black relaxation in geometric multigrid.)

Other degrees of freedom in defining the concrete strategy will be varied in the experiments below and some notation is required to distinguish these cases:

- **Type of cycle and coarsening.** The abbreviations VS and VA are used to distinguish V-cycles based on *standard* and *aggressive coarsening*, respectively (Section 7.1). Aggressive coarsening is performed only in creating the second AMG level, and only the types A1 and A2 are used (see Section 7.1.2). Correspondingly, we distinguish VA1 and VA2 cycles. For F-cycles, the "V" is replaced by "F".
- **Type of smoothing.** As mentioned above, by default we use one Gauss-Seidel CF-relaxation step for pre- and post-smoothing. If this is not the case, we append the type of smoothing to the abbreviation of the cycle. For instance, VS-FF stands for a V-cycle using standard coarsening but employing two F-smoothing steps rather than one CF-step for pre- and post-smoothing (cf. Section 5.1.2). SGS stands for symmetric Gauss-Seidel relaxation.
- **Type of interpolation.** The type of interpolation used is appended in parentheses: The letters "D" and "S" stand for *direct* and *standard interpolation*, respectively (see Section 7.2.1). Our standard AMG cycle is VS(S). Note that, if aggressive coarsening is employed, interpolation to the finest level is *always multi-pass interpolation* (Section 7.2.2). For example, VA2(S) means that standard interpolation is performed on all *but the finest level.*

  If, in addition, Jacobi-relaxation is applied to improve interpolation, the letters "F" and "P" refer to *fully* and *partially* relaxed interpolation, respectively (Section 7.2.3). For instance, VS(S-2F) stands for a V-cycle using standard coarsening and standard interpolation improved by 2 full Jacobi-relaxations. As mentioned above, truncation with $\varepsilon_{tr} = 0.2$ is the default. Otherwise, the truncation parameter is also contained within the parentheses, for example, VS(S-1F,0.02).
- **Acceleration.** If a cycle is used as *pre-conditioner* rather than stand-alone, the type of accelerator is appended to the corresponding cycle abbreviation. For instance, VA1(D)/cg means that the VA1(D) cycle is used as pre-conditioner for cg. Note that,

if a cycle is used as pre-conditioner for cg, pre- and post-smoothing will always be done in a symmetric way. For instance, if pre-smoothing is done by CF-relaxation, post-smoothing will be CF-relaxation *with the order of points reversed*. For non-symmetric problems, we will usually use AMG as pre-conditioner for BI-CGSTAB.

The following sections contain results on *asymptotic convergence*, *memory requirement* as well as *computational work*. The *asymptotic convergence factor*, ρ, is always computed numerically by applying a v. Mises vector iteration to the homogeneous problem, usually starting with a random first approximation. Results on memory requirement will be given in terms of the *grid* and *operator complexity*, c_G and c_A,

$$c_G = \sum_\ell n_\ell / n_1 \quad \text{and} \quad c_A = \sum_\ell m_\ell / m_1 \ , \tag{125}$$

where n_ℓ and m_ℓ denote the number of variables and non-zero matrix entries, respectively, on level ℓ. Note that $\ell = 1$ corresponds to the finest level. Although the true memory requirement by AMG is not fully reflected by these quantities (some extra work space still needs to be allocated), they are closely related.

Unless explicitly stated otherwise, all timings given have been obtained on a Pentium II/300 PC using the Lahey F90 Compiler (version 4.0). We point out that timings for a particular machine always have to be judged with care. Comparisons typically change from machine to machine and even from compiler to compiler. For instance, the Pentium II is relatively fast in integer (compared to floating point) computations. This is advantageous for substantial parts of the AMG algorithm which essentially require integer computations (in particular, during the setup phase). Consequently, comparisons of the setup and solution costs may give a different picture on machines for which floating point computations are more efficient than integer computations (such as on IBM RS6000 workstations).

8.2 Poisson-like problems

In the following, we investigate the performance of AMG in some detail if applied to the diffusion equation

$$-((1+\sin(x+y))u_x)_x - (e^{x+y}u_y)_y = f(x,y) \ , \tag{126}$$

defined on the unit square with $f(x,y) \equiv 1$ and homogeneous Dirichlet boundary conditions. Discretization is on a uniform grid of mesh size $h = 1/N$ using standard 5-point stencils. Although this example is very simple, the resulting AMG behavior is typical for general "Poisson-like" problems and the relevant conclusions qualitatively carry over also to unstructurd meshes (cf. the examples in Section 8.3). We summarize our practical experience with AMG in the following remark.

Remark 8.1 Compared to problems on very regular meshes, a certain decrease of AMG convergence has to be expected in case of irregular meshes. This is essentially due to the fact that, on regular meshes, standard AMG interpolation tends to be close to geometrical interpolation (which is very good for Poisson-like problems as considered here). This cannot be expected to be satisfied to the same extent on irregular meshes. Similarly,

convergence has to be expected to be somewhat slower in 3D than in 2D situations. In 3D, we have the additional effect that smoothing by Gauss-Seidel relaxation is (slightly) less efficient than in 2D problems (just like in geometric multigrid). By how much convergence will finally be influenced by the irregularity of the grid and its dimension, depends somewhat on the concrete problem. By experience, however, the effects mentioned are very limited and the results presented in the following exhibit the typical AMG behavior as observed also in many other cases. $\ll$

8.2.1 Coarsening and complexity

Problem (126) has a slight anisotropy towards the upper right corner. Due to the setting $\varepsilon_{str} = 0.25$, however, AMG still treats all connections contained in the corresponding matrix as strong, at least on the finest level. Consequently, the first standard coarsening step of AMG corresponds to geometrical red-black coarsening. This is shown in Figure 17a. Subsequent coarsening then becomes faster (here, grid size ratio 1:4) simply because the Galerkin stencils become larger on coarser levels. For example, the Galerkin operator on level 2 corresponds to a 9-point stencil. There is a slight disturbance of the regular coarsening in the upper right corner where the anisotropy of the problem is largest. However, the coarsening pattern is still essentially the same.

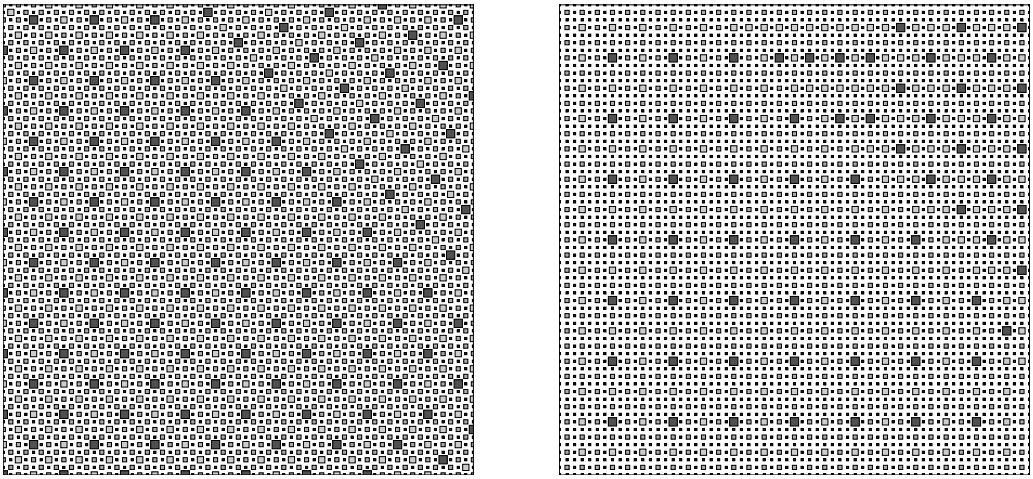


Figure 17: The finest and three consecutive AMG levels created by a) standard coarsening, b) aggressive A2-coarsening (applied only in the first coarsening step).

This type of coarsening is typical for 5-point stencils with all connections being strong, yielding grid complexities of $c_G \approx 1.7$. If interpolation were to be defined geometrically (i.e., linear interpolation), the Galerkin operators on all coarser levels would correspond to 9-point stencils and the "ideal" operator complexity would be $c_A \approx 2.2$. For AMG, however, the situation is more involved, since the final values of c_G and c_A are influenced by the AMG interpolation operator which tends to cover more points than geometric interpolation, especially towards coarser levels. As a consequence, the AMG Galerkin operators will tend to become somewhat larger towards coarser levels. This effect, however, is normally limited and is more than made up for by the decrease of grid points. In any

case, one has to expect the final operator complexity of AMG to be larger than the ideal one by a certain factor. In the above example, we obtain $c_A \approx 2.38$ if standard coarsening and interpolation are used (see Table 1).

Remark 8.2 Memory requirement is somewhat higher in corresponding 3D situations. For instance, if applied to 7-point stencils with all connections being strong, standard AMG coarsening again yields geometrical (3D) red-black coarsening in creating the first coarser level, with the Galerkin operator corresponding to a 19-point stencil. Subsequent coarsening, as before, will become faster. However, the "ideal" operator complexity (obtained if geometrical interpolation was used) is now $c_A \approx 2.8$. Consequently, the true AMG complexity, generally, has to be expected to be larger than 3.0. $\ll$

If memory requirement is an issue, aggressive coarsening may be used instead of standard coarsening. As mentioned earlier, it is usually sufficient to apply this type of coarsening only in the first coarsening step and maintain standard coarsening for all subsequent levels. Figure 17b shows the resulting first 4 levels if aggressive A2-coarsening is performed to create the first coarser level. The first AMG coarsening step now corresponds to geometrical $h \rightarrow 2h$ coarsening rather than red-black coarsening. Except for the upper right area of the domain, this also holds for the subsequent (standard) coarsening steps. Near the upper right corner, the situation is slightly different. Obviously, the Galerkin operator on level 2 is more anisotropic than it was on the finest level. AMG detects this and creates the third level by line-wise coarsening in y-direction. Since line-wise coarsening essentially removes the anisotropy, the next coarsening step is again in both directions.

Ignoring the special coarsening near the upper right corner, the grid complexity now is only $c_G \approx 1.33$ and the ideal operator complexity becomes $c_A \approx 1.6$ (assuming linear interpolation, all Galerkin operators correspond to 9-point stencils). As before, the true AMG operator complexity will be somewhat larger; in the above example we obtain $c_A \approx 1.77$ (see Table 1) which is very reasonable. In corresponding 3D situations, the gain in terms of memory reduction by means of aggressive coarsening is even higher. Memory usage can further be reduced either by using A2-coarsening also on the coarser levels (which usually does not pay, see above) or by using A1-coarsening to create the first coarser level (cf. the examples in Section 8.3).

8.2.2 Performance and comparisons

Figure 18a shows asymptotic convergence factors, ρ, for several AMG strategies and increasing N. We first observe that our standard cycle, the VS(S)-cycle, exhibits a very stable convergence behavior with $\rho < 0.15$ for increasing N. Investing more effort into the interpolation, by applying one Jacobi F-relaxation, improves convergence only marginally, indicating that the standard interpolation is fairly good in this case (cf. VS(S-1F,0.02)-cycle). In contrast to this, investing more work into the cycle itself, by applying an F- instead of a V-cycle, causes extremely fast convergence (cf. Remark 8.4 below).

The influence of aggressive coarsening is demonstrated by the low-memory VA2(S)- and VA1(S)-cycles. As expected, convergence becomes considerably slower but still approaches an upper limit for large N. As mentioned earlier, aggressive coarsening causes not only the smoothing to be less effective (we still use only one CF-relaxation step for pre- and post-smoothing) but also interpolation to be less accurate (multi-pass interpolation from

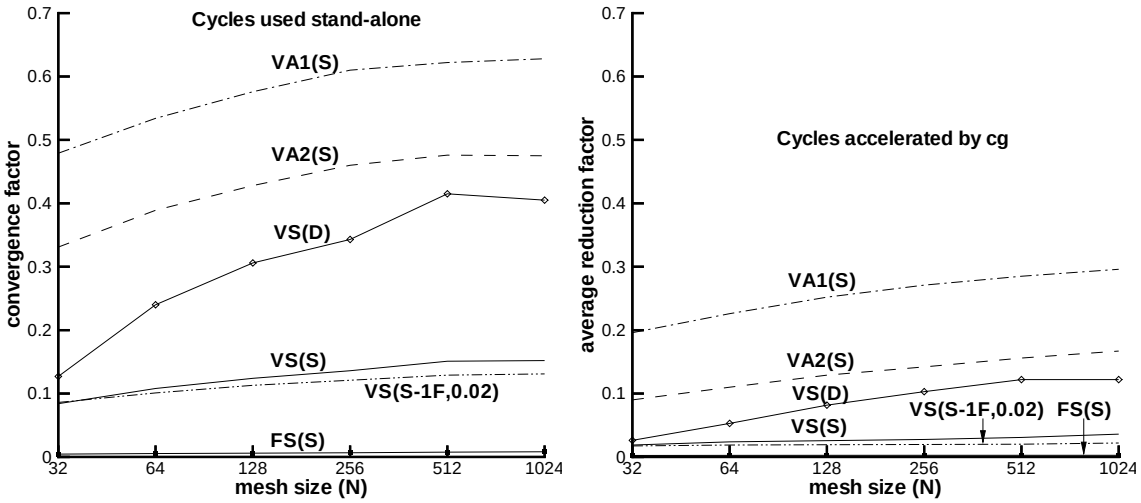


Figure 18: a) Convergence factors for cycles used stand-alone. b) Average reduction factors for accelerated cycles.

level 2 to level 1). Clearly, convergence can be improved, for instance, by using twice as many smoothing steps and using Jacobi-relaxation to improve interpolation to the finest level. However, this would substantially increase the cycle cost and, more importantly, tend to reduce the advantage of low operator complexity which was the major purpose for using aggressive coarsening to begin with.

A simpler and very effective way of improving the convergence of any cycle, in particular those using aggressive coarsening, is to use them as pre-conditioners rather than stand-alone. This is demonstrated in Figure 18b which shows the AMG convergence if used as pre-conditioner for cg. For the same cycles as before, the average residual reduction factors are shown, obtained over (at most) 50 iterations in solving the homogeneous system starting with a random first approximation. At relatively little extra cost, convergence is enhanced substantially.

Remark 8.3 Figure 18 also depicts the convergence of the VS(D)-cycle (where standard interpolation is replaced by the simpler direct interpolation). Although this cycle also gives good convergence in most cases, its convergence behavior is often not as stable as that of the VS(S)-cycle. This is why the VS(S)-cycle is our standard cycle. $\ll$

Remark 8.4 The purpose of using F-cycles rather than V-cycles is to solve the coarse-level correction equations more accurately and to reduce the accumulation of errors from the individual levels of the AMG hierarchy. Clearly, in general, F-cycle convergence cannot be better than the corresponding two-level convergence. The drastic improvement of convergence shown in Figure 18 (cf. VS(S)- and FS(S)-cycles) is due to a particular situation which cannot be expected in general: Since standard coarsening, applied to a 5-point stencil with all connections being strong, corresponds to red-black coarsening, there are no F-to-F connections on the finest level (i.e., A_{FF} is a diagonal matrix). Therefore, according to Section 2.3, the two-level method involving the first two levels corresponds to a direct solver. Consequently, any increase of accuracy in solving the correction equations on level 2, directly improves the AMG convergence by a corresponding amount. Since,

compared to the V-cycle, the F-cycle solves the second level correction equation approximately twice as accurately, this immediately causes the overall convergence to be twice as fast. (Note that we have a similar effect for isotropic 7-point stencils in 3D.) ≪

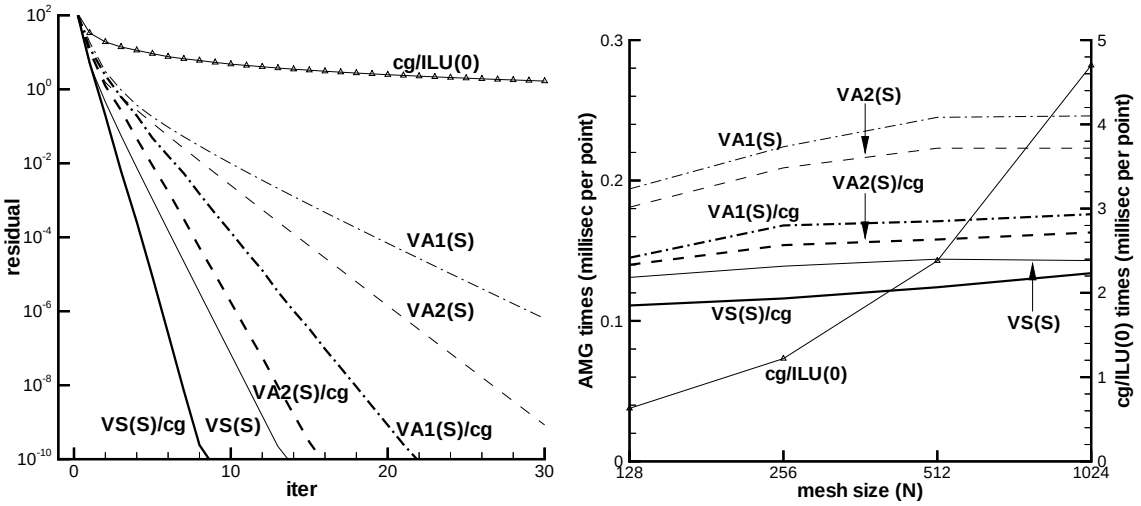


Figure 19: a) Convergence histories for $N = 512$. b) Total time in millisec per finest grid point to reduce the residual by 10 orders of magnitude.

Figure 19a shows the convergence histories of the VS-, VA2- and VA1-cycles, with and without acceleration by cg, for solving the given problem ($N = 512$ and using $u \equiv 1$ as first approximation)[3]. The figure also shows the convergence history of ILU(0) pre-conditioned cg for the first 30 iterations.

However, the computational time, and how it increases with increasing N, is more important than convergence histories. Figure 19b shows computational times to solve (126) for varying mesh sizes up to a residual reduction by 10 orders of magnitude. Times are given in milliseconds *per finest grid point* and include the setup cost. For all AMG variants discussed here, the total cost approaches an upper limit for increasing mesh sizes which demonstrates their computational optimality for solving problems of the kind at hand. The figure also depicts the corresponding increase of cost for ILU(0) pre-conditioned cg. Since the convergence speed of cg/ILU(0) depends on N (the number of cg iterations required increases proportionally with N), the advantage of AMG over cg/ILU(0) substantially increases with increasing problem size. For $N = 1024$, it can be seen that the accelerated standard cycle, VS(S)/cg, is about 37 times faster than cg/ILU(0).

Remark 8.5 The computational work of AMG is essentially determined by the operator complexity c_A and the convergence factor ρ. Only if both quantities are bounded as a function of h, do we have an asymptotically optimal performance. We here just remark that c_A is indeed virtually independent of N for all AMG variants shown. The slight increase of the total cost for medium sized meshes, seen in Figure 19b, is caused by the small increase of the convergence factors in this area (cf. Figure 18). ≪

[3]For ease of reading, we always use thin lines for stand-alone AMG cycles and corresponding fat lines for their accelerated analogs.

method	complexities		times (sec) / Pentium II, 300 MHz				
			setup	stand-alone		conjugate gradient	
	c_A	c_G	time	cycle	$\varepsilon_0 = 10^{-10}$	cycle	$\varepsilon_0 = 10^{-10}$
ILU(0)			0.87			1.07	628.6 (587)
AMG1R5	2.42	1.71	6.97	2.10	25.9 (9)		
VS(S)	2.38	1.67	11.8	2.32	37.4 (11)	2.93	32.6 (7)
FS(S)	"	"	"	3.87	31.2 (5)	4.46	29.7 (4)
VS(D)	2.20	1.67	8.51	2.22	48.5 (18)	2.83	39.7 (11)
VA2(S)	1.77	1.35	10.3	1.78	58.2 (27)	2.38	41.3 (13)
VA1(S)	1.50	1.19	8.07	1.47	65.4 (39)	2.07	45.3 (18)

Table 1: Complexities and computing times ($N = 512$)

Detailed measurements are given in Table 1 for $N = 512$ including the complexity values, c_G and c_A. Besides the computational times for the setup phase and single cycles, total execution times (including setup) are given for the reduction of the residual by a factor of $\varepsilon_0 = 10^{-10}$ (the values in parentheses indicate the number of iterations required). The table shows that the standard cycle, VS(S)/cg, is nearly 20 times faster than standard ILU(0) preconditioned cg. The lowest-memory cycle, VA1(S), reduces the memory overhead for storing the coarse-level matrices by approximately 64%. If used as a pre-conditioner, it is still approximately 14 times faster than cg/ILU(0).

Table 1 also shows complexity values and timings for the original AMG solver, AMG1R5, which should be compared with the VS(S)-cycle. Obviously, for the current problem, AMG1R5 converges somewhat faster (9 iterations instead of 11) and the total execution time is lower[4]. The faster convergence is due to the particularly simple geometrical situation which benefits the interpolation used in AMG1R5 (cf. Remark 7.8). We will see later that this advantage gets lost in more complex geometric situations or for more complicated problems.

Remark 8.6 The cost of "better" cycles such as F-cycles (and even more of W-cycles) is usually substantially higher than that of the simpler V-cycles, at least in connection with standard coarsening: While, in V-cycles, each level is visited just once, in F-cycles, level n is visited n times. This increase of cost is, generally, more critical in AMG than it is for comparable geometric multigrid cycles. This is mainly due to the more complex AMG coarse-level operators. Thus, although convergence of F-cycles may be faster than that of their V-cycle analog, the total computational time is usually higher. That, in Table 1, the total times for the FS(S)-cycle are (slightly) *lower* than that for the VS(S)-cycle is a consequence of the extremely fast F-cycle convergence which, in turn, is a consequence of the particular situation mentioned in Remark 8.4. $\ll$

8.2.3 F-smoothing and Jacobi-interpolation

According to the theoretical results of Section 5.1.4, it is possible to employ mere F-smoothing instead of full smoothing except that standard interpolation might then not be

[4]This lower computational time is, to some extent, due to the fact that AMG1R5 is a FORTRAN77 code using only static arrays while RAMG05 uses dynamic FORTRAN90 arrays (which decreases the performance in case of the Lahey compiler used here).

sufficient any more. Indeed, generally, additional work needs to be invested in improving interpolation by Jacobi F-relaxation in order to cope with all those error components which cannot efficiently be reduced by mere F-smoothing. That just one Jacobi-step is enough is demonstrated in Figure 20 which compares the convergence factors of the standard cycle, VS(S), with that of various cycles using mere F-smoothing[5].

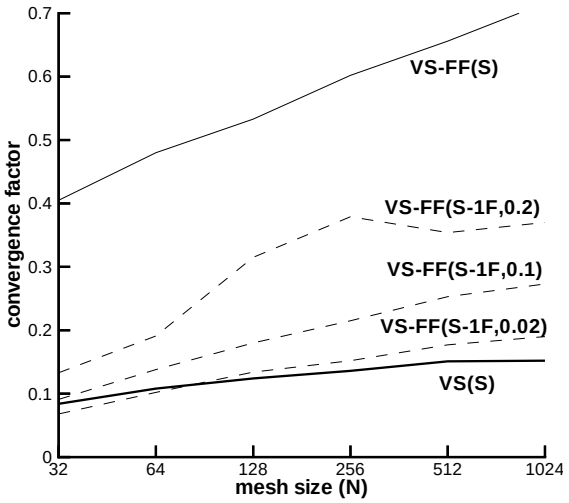


Figure 20: Convergence factors of cycles using F-smoothing

method	complexities		times (sec) / Pentium II, 300 MHz			ρ
	c_A	c_G	setup	cycle	$\varepsilon_0 = 10^{-10}$	
VS(S)	2.38	1.67	11.8	2.32	37.4 (11)	0.151
VS-FF(S)	2.45	1.68	13.1	2.65	113.6 (38)	0.656
VS-FF(S-1F,0.2)	2.82	1.67	22.0	2.91	65.6 (15)	0.354
VS-FF(S-1F,0.1)	2.95	1.67	24.7	3.02	60.9 (12)	0.253
VS-FF(S-1F,0.02)	3.24	1.67	39.6	3.20	71.9 (10)	0.177

Table 2: Complexities and computing times for cycles using F-smoothing ($N = 512$)

Obviously, the VS-FF(S) cycle (i.e., the VS(S)-cycle with each CF-relaxation step replaced by two F-relaxation steps), is substantially inferior to the VS(S)-cycle and h-dependent. However, one Jacobi F-relaxation step, applied to the interpolation, enhances convergence substantially: if truncation with $\varepsilon_{tr} \leq 0.1$ is used, convergence approaches that of the VS(S)-cycle. (Truncation with the default value, $\varepsilon_{tr} = 0.2$, is not quite sufficient.) Unfortunately, this is at the expense of an increase of the total solution cost (mainly because of a strong increase of the setup cost). In addition, operator complexities substantially increase. Detailed results are shown in Table 2 for the case $N = 512$. Although we have already mentioned ways of improvement, the trend indicated is typical: skipping relaxation of the C-equations may unnecessarily increase cost and memory requirement (cf. Section 5.2).

[5]We have forced strong diagonal dominance (95) here with $\delta = 0.75$.

8.3 Computational fluid dynamics

Industrial CFD applications involve very complicated flow problems. For instance, in the car industry, 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 unstructurd, 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. Figures 21 and 22 show two exemplary meshes used to model the flow through a down-shot coal furnace and the cooling jacket of a four-cylinder engine, respectively[6].

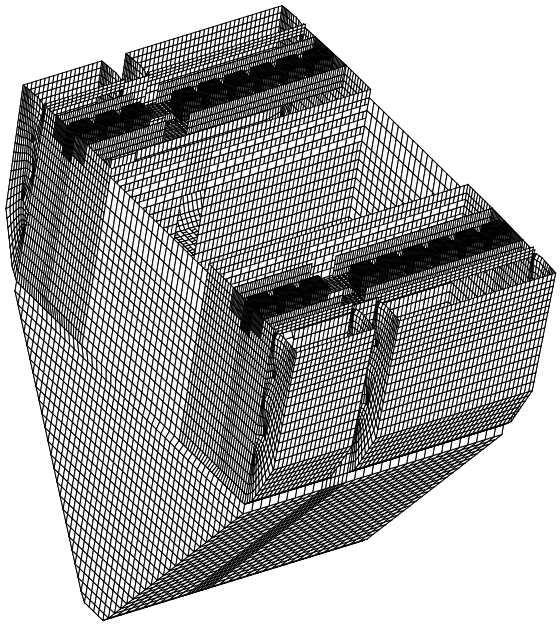


Figure 21: View into the interior of the bottom part of a coal furnace model (325,000 mesh cells; for simplicity, only the mesh surface is visualised)

The software industry is continuously improving the generality and the efficiency of their codes. The incorporation of multigrid methods would be one way to improve performance. Geometrically oriented approaches, however, can hardly cope with the complex geometries under consideration. Generally, there is no natural grid hierarchy which could easily be exploited. But even if there was such a hierarchy, the coarsest level would still be required to be fine enough to resolve the geometry to some extent. For industrially relevant configurations, such coarsest grids would still be much too fine for an efficient multi-level solution. AMG is of particular interest here since it can be used as a "plug-in solver" for existing codes.

In this section, we present some examples of AMG's performance if applied to industrial CFD applications based on segregated solution methods.

[6] All examples have been provided by Computational Dynamics Ltd.

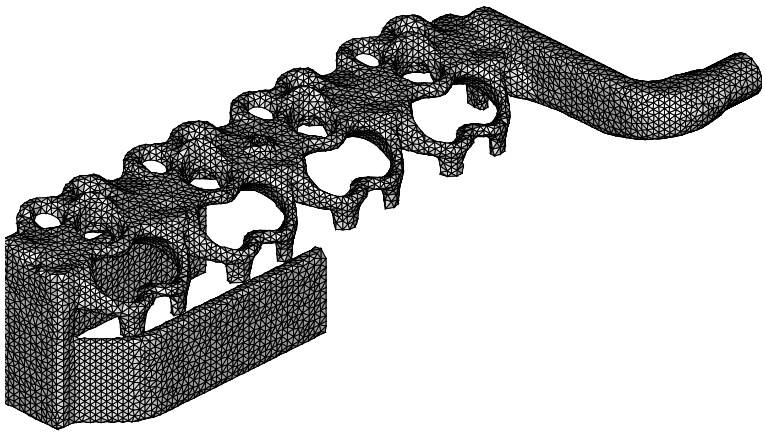


Figure 22: Cooling jacket of a four-cylinder engine (100,000 cells)

8.3.1 Segregated solution methods

The basic equations to be solved are the Navier-Stokes equations

$$\mathbf{u}_t - \frac{1}{Re}\Delta\mathbf{u} + \mathbf{u}\cdot\nabla\mathbf{u} + \nabla p = f \tag{127}$$

$$\nabla\cdot\mathbf{u} = 0 \tag{128}$$

where (127) are the momentum equations and (128) is the continuity equation. $\mathbf{u}$ denotes the velocity vector, p the pressure and f the body force. Re is the Reynolds number.

Segregated solution methods (or *pressure correction type* methods) to tackle the solution of the Navier-Stokes equations belong to the most established approaches used in general-purpose commercial CFD codes. Their major advantage is that, at each time step, the approximate solution of just a series of scalar equations is required rather than the solution of the coupled Navier-Stokes system.

Assuming an implicit time-stepping scheme (here backward Euler for simplicity) and a stable discretisation in space (indicated by the subscript h), equations of the following form have to be solved at the n-th time step:

$$\frac{1}{\delta t}(\mathbf{u}_h^{(n)} - \mathbf{u}_h^{(n-1)}) - \frac{1}{Re}\Delta_h\mathbf{u}_h^{(n)} + \mathbf{u}_h^{(n-1)}\cdot\nabla_h\mathbf{u}_h^{(n)} + \nabla_h p_h^{(n)} = f_h^{(n)} \tag{129}$$

$$\nabla_h\cdot\mathbf{u}_h^{(n)} = 0\,. \tag{130}$$

Here, the convective part has been linearised. (If required, the solution of the nonlinear equations can be computed iteratively in a straightforward way.) For ease of reading, we omit the subscript h in the following.

There are several variants of pressure correction type approaches all of which proceed in two steps: First, an intermediate velocity approximation, $\mathbf{u}^\star$, is computed by replacing $p^{(n)}$ in (129) by values from the previous time step:

$$\frac{1}{\delta t}(\mathbf{u}^\star - \mathbf{u}^{(n-1)}) - \frac{1}{Re}\Delta\mathbf{u}^\star + \mathbf{u}^{(n-1)}\cdot\nabla\mathbf{u}^\star + \nabla p^{(n-1)} = f^{(n)}\,. \tag{131}$$

Second, corrections

$$\mathbf{u}^{(n)} = \mathbf{u}^\star + \mathbf{u}' \quad \text{and} \quad p^{(n)} = p^{(n-1)} + p' \tag{132}$$

are computed such that $\mathbf{u}^{(n)}$ is an improved solution of (129) satisfying the continuity equation (130). To strictly satisfy (129), one would have to solve

$$\frac{1}{\delta t}\mathbf{u}' - \frac{1}{Re}\Delta\mathbf{u}' + \mathbf{u}^{(n-1)} \cdot \nabla\mathbf{u}' + \nabla p' = 0 \; . \tag{133}$$

However, since the correction $\mathbf{u}'$ is assumed to be relatively small and change little in space, the $\mathbf{u}'$-dependent part in (133) is approximated by $A(\mathbf{u}^{(n-1)})\mathbf{u}'$ with some simple (invertible) matrix A, depending on old velocity values. Usually A is assumed to be diagonal, in the simplest case just $A = \frac{1}{\delta t}I$. Well-known methods such as SIMPLE and SIMPLEC are based on such approximations. Consequently, the velocity correction $\mathbf{u}'$ is computed via

$$\mathbf{u}' = -[A(\mathbf{u}^{(n-1)})]^{-1}\nabla p' \tag{134}$$

where p' is the solution of the so-called *pressure correction equation*

$$\nabla \cdot [A(\mathbf{u}^{(n-1)})]^{-1}\nabla p' = \nabla \cdot \mathbf{u}^\star \; . \tag{135}$$

The latter follows immediately from (134) because of the requirement that the velocity, after its correction (132), has to satisfy the continuity equation (130). Typically, the pressure correction equation has to be solved several times at each time step.

In practice, segregated solution methods are used to solve time-dependent as well as steady-state problems with well-known approaches being PISO and SIMPLER, respectively (for more information on segregated solution methods, we refer to [51, 52, 53, 45, 36, 22]).

Summarising, two different types of scalar equations have to be solved within each step: A set of (de-coupled) convection-diffusion equations (131) and the pressure-correction equation (135), a Poisson-like equation with coefficients which, in general, depend on known velocity values. AMG can efficiently be used to solve both types of scalar equations. Regarding Poisson-like equations, we have demonstrated this in Section 8.2 by means of a simple model problem. Convection-diffusion problems will be considered in Section 8.5.2. However, the pressure-correction equation is generally by far the more expensive one to solve. We therefore put our focus on this equation. For a further discussion of segregated solution methods, in particular in the context of AMG, see also [30].

Remark 8.7 The fact that the pressure-correction equation is just one component within an outer iteration has two implications. First, there is no need to solve it too accurately, in particular not in steady-state computations. We will therefore consider the efficiency of AMG not only for obtaining high- but also low-accuracy approximations. Second, potential performance gains through the use of an efficient solver solely for solving the pressure-correction equations is limited by the cost of the remaining components. But still, since the solution of the pressure-correction equations typically makes up the largest part of the overall computation, potential benefits may be substantial. $\ll$

Remark 8.8 Currently, there is a trend in commercial code development towards solving the Navier-Stokes equations directly as a fully coupled system. However, this has several drawbacks, for instance, regarding overall memory requirements (which is a major concern for all commercial software providers). On the other hand, having efficient solvers available for directly solving coupled systems on complex geometries, would increase the overall performance substantially. AMG solvers can play a very important role here. Extensions of the AMG approach which can handle such coupled systems are under development. ≪

8.3.2 Industrial test cases

In order for any new solver to be interesting for industrial use, it has to be *fast*, *robust* and require only a *low amount of memory*. Whether or not this is satisfied, has to be judged by comparing with solvers typically used such as ILU pre-conditioned conjugate gradient.

Regarding performance, AMG is generally much more efficient than any one-level method, in particular, if the underlying meshes are large and complex, if there are thin substructures in different directions, or if coefficients are not smoothly varying. Also robustness has turned out to be extraordinarily high for all industrial problems solved by now. Regarding memory requirement, AMG can certainly not compete with a simple one-level method. In fact, any hierarchical solver – and hierarchical approaches are *necessary* to obtain fast solution – requires additional memory. Memory requirement, however, 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. In fact, most industrial users would prefer to wait longer for the results if they would otherwise not be able to solve what they really want to solve.

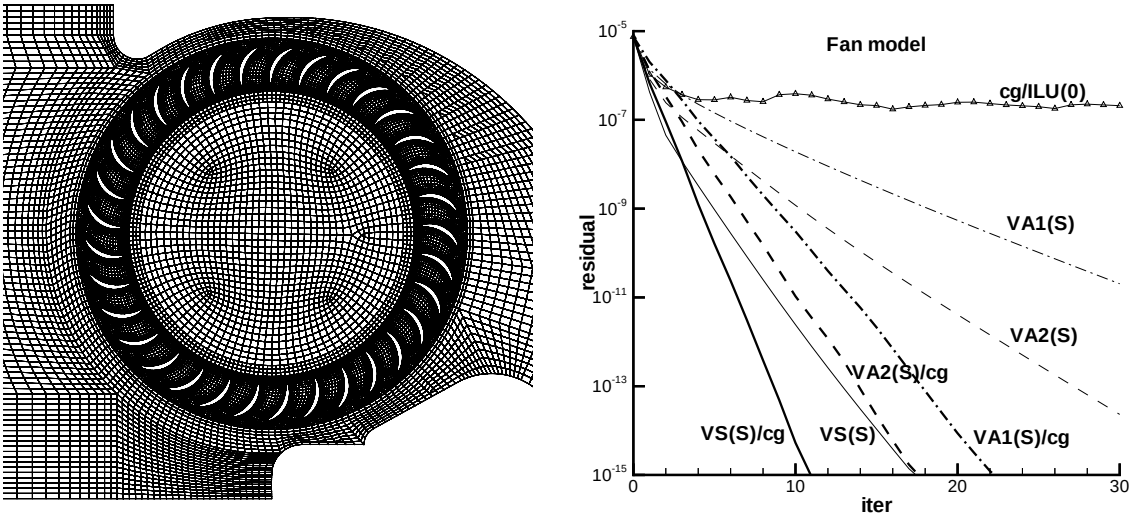


Figure 23: a) Core part of a fan model. b) Convergence histories.

For these reasons, low-memory AMG approaches are of particular interest, even if the reduced memory requirement causes an increase of the total computational time. A memory overhead of some tens of percents is certainly acceptable. In any case, however,

the operator complexity c_A must not be significantly larger than 2.0, say. We will see that the low-memory cycles, VA1(S) and VA2(S), will satisfy the industrial requirements in all cases considered.

In this section, our focus will be on solving pressure-correction equations to a *high accuracy*, namely, by reducing the residual by 10 orders of magnitude. (Regarding *low-accuracy* approximations, see Section 8.3.3.) We consider problems with different types of meshes. Discretisation is based on a standard finite-volume approach. In all cases, the concrete data used corresponds to one particular time step taken from a normal production run.

The first problem corresponds to the 2D simulation of the flow through a fan model. The core part of the corresponding mesh is outlined in Figure 23a. The mesh consists mostly of quadrilaterals and some triangles.

Figure 23b shows the convergence histories for the standard VS(S) cycle as well as for the low-memory variants VA1(S) and VA2(S), used with and without acceleration by conjugate gradient. We observe that the convergence behavior is very much comparable to that of the simple model equation (126) except that convergence is slightly slower here (cf. Figure 19a). As before, the VA-cycles are not supposed to be used stand-alone but rather as preconditioners.

	complexities		times (sec) / Pentium II, 300 MHz				
			setup	stand-alone		conjugate gradient	
method	c_A	c_G	time	cycle	$\varepsilon_0 = 10^{-10}$	cycle	$\varepsilon_0 = 10^{-10}$
ILU(0)			0.11			0.09	32.0 (354)
VS(S)	2.38	1.65	0.93	0.20	4.50 (18)	0.25	3.68 (11)
VA2(S)	1.87	1.43	0.83	0.16	6.70 (37)	0.21	4.56 (18)
VA1(S)	1.39	1.21	0.60	0.13	8.44 (63)	0.17	4.60 (23)

Table 3: Complexities and computing times (fan model)

The fastest cycle, VS(S)/cg, needs 11 steps to reduce the residual by 10 orders of magnitude. It also provides the best method in terms of computational time as can be seen from Table 3: It requires a total time of 3.68 sec which is about 8.5 times less than the time required by ILU(0)/cg. Memory overhead is reasonable for this 2D problem, however, relative to our above requirements, still somewhat too high. The VA2- and VA1-cycles are still much faster than ILU(0)/cg but require substantially less memory. In particular, the operator complexity of the VA1/cg-cycle is only $c_A = 1.39$. That is, its memory overhead is smaller than that of the standard cycle by 72% at the expense of some 25% increase in total execution time.

The following two examples correspond to 3D flow computations with largely different unstructured meshes, namely, the flows through the cooling jacket of a four-cylinder engine (Figure 22) and through a coal furnace (Figure 21), respectively. While the first mesh is a fairly uniform tetrahedral mesh, the second one consists mainly of hexahedra and a few thousand pentahedra, including many locally refined grid patches. According to this, the discretized problems employ mostly 5-point stencils in the first case and varying stencil sizes in the second case (ranging from 4- to 11-point stencils).

The convergence histories for both problems are depicted in Figure 24. Observe first that the difference in size and structure of the above meshes hardly influences the con-

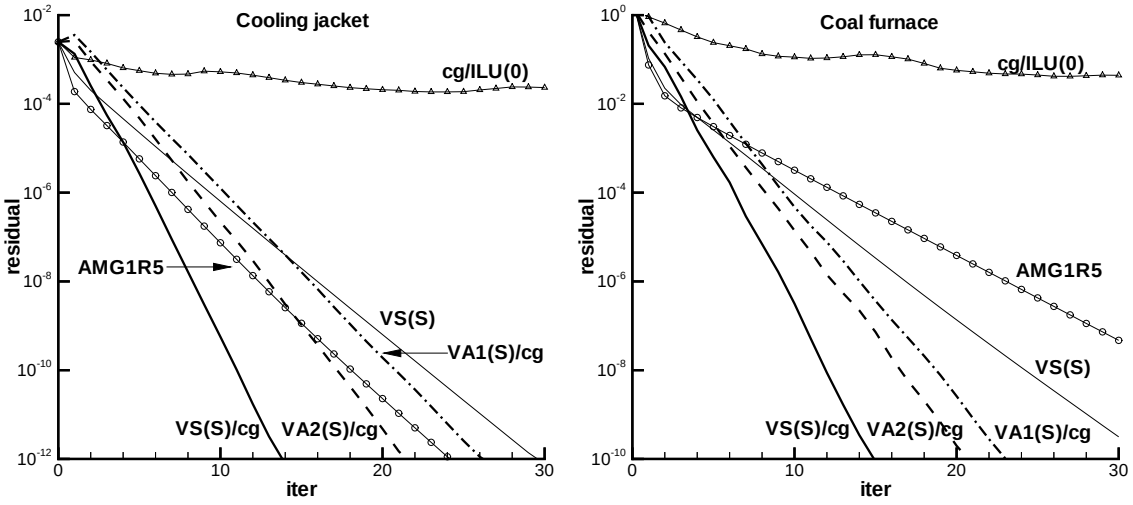


Figure 24: Convergence histories: a) cooling jacket, b) coal furnace.

vergence of AMG. Compared to the 2D problem (Figure 23b), however, convergence is somewhat slower here which is typical for 3D applications (cf. Remark 8.1). The only major difference between the 2D and 3D problems is that the standard VS(S)-cycle, used stand-alone, converges significantly slower in the 3D cases. As a consequence, the accelerated VA-cycles converge faster than the standard cycle. For both meshes, the accelerated standard cycle exhibits fastest convergence and requires 15 iterations to reduce the residual by 10 orders of magnitude.

Table 4 shows that, in terms of total execution time, the accelerated standard cycle is nearly 20 times faster than ILU(0)/cg for the cooling jacket, and around 6.5 times for the coal furnace. The lowest-memory cycle, VA1(S)/cg, is still over 17 times faster than cg/ILU(0) for the first case. For the second case, it is even cheaper than the accelerated standard cycle although it requires 8 additional iterations.

Generally, standard coarsening requires significantly more memory in 3D than it does in 2D. According to the table, the operator complexity of the VS(S)-cycle is given by $c_A = 2.77$ and $c_A = 3.39$ for the two problems (which is still practical but too high w.r.t our particular requirements). The reasons for this increase of complexity are similar to those pointed out already in Remark 8.2 for the model problem: the first standard coarsening step tends to be relatively slow (in terms of a reduction of grid points) while, at the same time, the size of the Galerkin stencils on the second level becomes substantially larger than on the finest one. As can be seen from Table 4, aggressive coarsening avoids this problem very efficiently in both cases. In particular, A1-coarsening reduces the memory overhead by around 80% in both test cases. Although this significantly increases the number of iterations required, the resulting method is still very efficient (even the most efficient in some cases).

Remark 8.9 For comparison, Figure 24 and Table 4 show also the performance of the original code AMG1R5. Note first that convergence of AMG1R5 is comparable to RAMG05 for the cooling jacket case but significantly slower for the coal furnace case. More importantly, however, the complexity values of AMG1R5 – $c_A = 5.35$ and $c_A = 7.06$, re-

method	complexities		times (sec) / Pentium II, 300 MHz				
			setup	stand-alone		conjugate gradient	
	c_A	c_G	time	cycle	$\varepsilon_0 = 10^{-10}$	cycle	$\varepsilon_0 = 10^{-10}$
Cooling jacket							
ILU(0)			0.39			0.40	434.1 (1084)
AMG1R5	5.35	1.98	7.25	1.29	42.1 (27)		
VS(S)	2.77	1.55	5.77	0.90	34.4 (32)	1.11	22.6 (15)
VA2(S)	2.25	1.30	4.94	0.73	55.6 (69)	0.96	27.2 (23)
VA1(S)	1.44	1.14	3.18	0.56	59.2 (100)	0.76	24.5 (28)
Coal furnace							
ILU(0)			1.86			1.70	743.8 (436)
AMG1R5	7.06	1.90	72.6	6.76	370.0 (44)		
VS(S)	3.39	1.59	33.9	4.36	174.2 (32)	5.27	113.5 (15)
VA2(S)	2.12	1.27	21.8	3.00	173.5 (51)	3.93	104.6 (21)
VA1(S)	1.47	1.14	14.4	2.33	170.2 (67)	3.27	89.5 (23)

Table 4: Complexities and computing times

spectively – indicate an unacceptably high memory requirement of AMG1R5 in both test cases. This confirms what has already been mentioned in Remark 7.8: the coarsening strategy used in AMG1R5 may become very inefficient in case of non-regular 3D meshes such as those considered here. Since c_A is directly related to the computational time, AMG1R5 is more expensive, in particular, in the coal furnace case. $\ll$

8.3.3 Low-accuracy approximations

Particularly in steady-state computations, the pressure-correction equation usually needs to be solved only with a low accuracy of one or two digits, say. One might expect that then the use of AMG solvers is an "overkill" (in particular, because of the high setup cost involved) and simple one-level methods would become more efficient.

Indeed, this seems to be true if one compares the total computing times needed by AMG and cg/ILU(0) to reduce the residual by only *one order of magnitude*. Table 5 shows corresponding timings (in the columns labeled "resid") for both the fan model and the cooling jacket. The results show that the execution time of AMG is still comparable to that of cg/ILU(0) in case of the cooling jacket but is considerably higher in case of the fan model where cg/ILU(0) appears to be up to 4 times faster. Note that, in this case, AMG's setup time alone is already 2 to 3 times higher than the total execution time of cg/ILU(0)! If, however, the residual is required to be reduced by *two orders of magnitude* instead, the execution time of cg/ILU(0) becomes again higher than that of AMG (accelerated VA1(S)-cycle) by factors of approximately 8 and 23 for the two problems.

Generally, however, one has to be very careful in drawing conclusions from small residual reductions to corresponding reductions in the error. Indeed, if one compares AMG with cg/ILU(0) on the basis of *error* rather than *residual* reductions, the picture looks completely different. To demonstrate this, Table 5 contains also total computing times on the basis of the true error reduction (in the columns labeled "error"). According to these results, AMG is *always* faster than cg/ILU(0). For instance, even if the requirement on

method	fan model					cooling jacket				
	setup	$\varepsilon_0 = 10^{-1}$		$\varepsilon_0 = 10^{-2}$		setup	$\varepsilon_0 = 10^{-1}$		$\varepsilon_0 = 10^{-2}$	
	cost	resid	error	resid	error	cost	resid	error	resid	error
cg/ILU(0)	0.11	0.28	9.98	11.1	12.2	0.39	7.64	114.8	202.8	177.4
VS(S)	0.93	1.15	1.32	1.32	1.69	5.77	7.62	8.50	10.2	11.2
VS(S)/cg	"	1.20	1.45	1.70	1.45	"	8.99	8.99	10.3	8.99
VA2(S)	0.83	1.10	1.23	1.41	1.54	4.94	8.84	9.50	13.2	13.9
VA2(S)/cg	"	1.20	1.20	1.65	1.20	"	8.95	7.00	10.8	8.95
VA1(S)	0.60	0.86	1.22	1.47	1.83	3.18	7.53	8.62	12.9	14.1
VA1(S)/cg	"	0.94	0.94	1.46	1.46	"	4.70	4.70	8.49	6.22

Table 5: Total computing times to reach a low *residual* and *error* reduction, respectively

the error reduction is merely one order of magnitude, the execution time of VA1(S)/cg is lower than that of cg/ILU(0) by factors of approximately 10 and 25 for the first and second test case, respectively. Note that these results depend, to some extent, on the used norm (here, we used the Euclidian norm). The tendency, however, will be similar also for other norms.

This advantageous behavior of AMG in terms of error reduction is related to its property to *globally* reduce errors much more effectively than a one-level method (such as ILU(0)). To illustrate this different behavior further, Figure 25 shows convergence histories separately for residuals and errors both for cg/ILU(0) and VA2(S)/cg. Obviously, during the first iterations, AMG reduces errors much more effectively than cg/ILU(0). This unsatisfactory behavior of cg/ILU(0) makes the use of termination criteria, based merely on the residual reduction, very unpractical: If the given tolerance is too large, cg/ILU(0) may stop after only a few iterations although the error may still be far too large. On the other hand, selecting a (slightly) smaller tolerance, may drastically increase computing cost.

Nevertheless, in computing low-accuracy approximations, AMG's setup cost becomes quite substantial. In fact, far most of the total computing time may be spent in the setup routines. Incidentally, however, in situations as described here, typically chains of (often many hundreds or thousands of) problems have to be solved for which the underlying matrices usually change only slowly from one step to the next. Consequently, AMG's setup phase needs to be performed only once in a while. For most of the problems, the complete setup can be "frozen" (or "updated" by fixing the interpolation and just re-computing the Galerkin operators). To control such an optimized use of AMG by an efficient and automatic strategy is straightforward. In this way, AMG's total setup overhead can be drastically reduced. For those examples considered here, for instance, new setups are, on the average, needed only after every 5th time step. Clearly, this substantially enhances the efficiency of AMG further.

8.4 Problems with discontinuous coefficients

In this section, we consider problems with strongly discontinuous coefficients, again beginning with the investigation of a typical model problem. We will see that, compared to

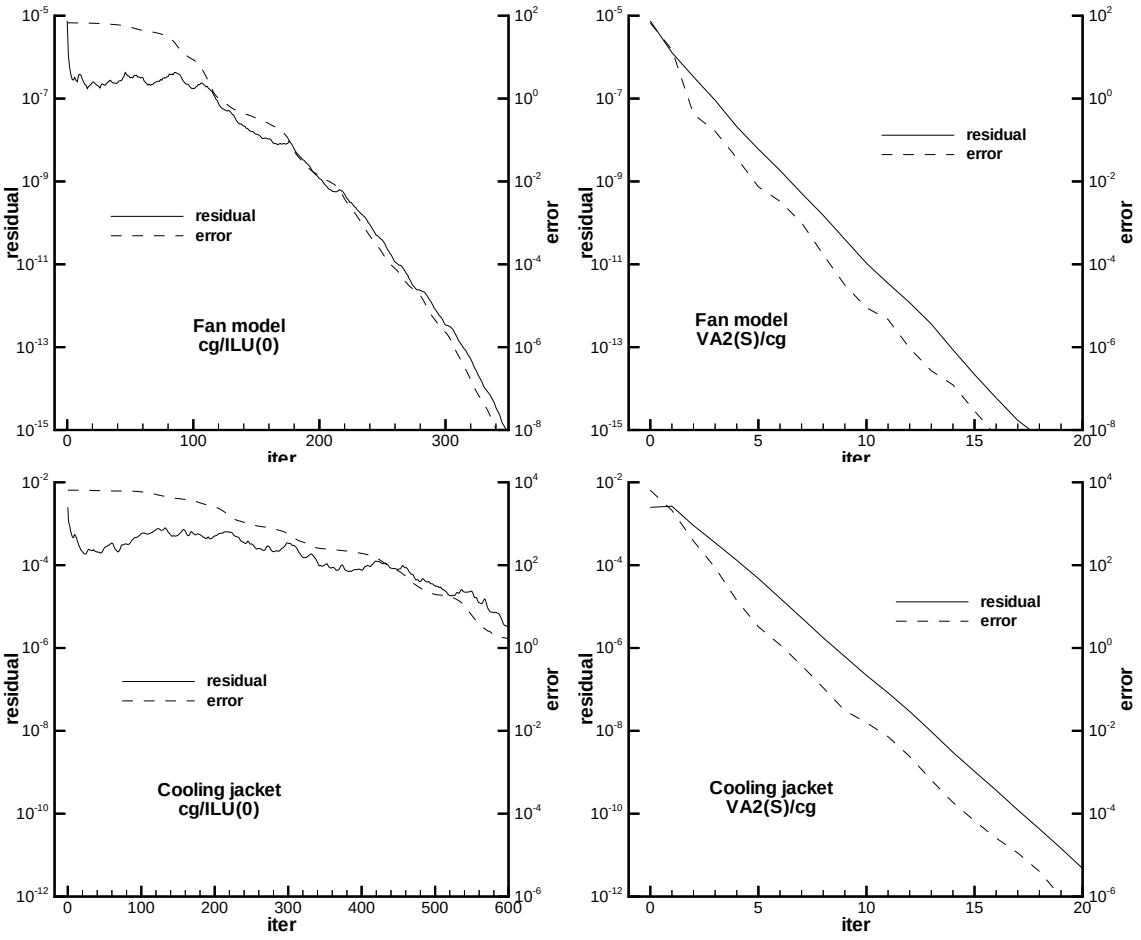


Figure 25: Convergence histories: residual vs. error (fan model and cooling jacket)

Poisson-like problems, the overall performance of AMG decreases to some extent. Qualitatively, however, AMG behaves as before. In particular, its performance for more complex problems is very similar to that observed in Section 8.3. We give typical results in Sections 8.4.2 and 8.4.3.

8.4.1 A model problem

We consider the diffusion problem [49]

$$-(a\,u_x)_x \;-\; (b\,u_y)_y \;=\; f(x,y) \tag{136}$$

on the unit square with discontinuous coefficients $a > 0$ and $b > 0$ being defined as indicated in Figure 26. $f(x,y)$ is defined to be 0 except for the points $(0.25, 0.25), (0.5, 0.5)$ and $(0.75, 0.75)$ where it is defined to be 10. Dirichlet boundary conditions are given as

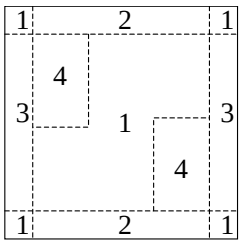
$$u = 1 \quad \text{for} \quad x \le 0.5,\, y = 0 \text{ and } x = 0,\, y \le 0.5\,; \quad \text{otherwise}: \; u = 0\,.$$

Discretisation is assumed to be done by the standard 5-point stencil on a regular grid of mesh size $h = 1/N$. For instance, the x-derivative $-(a u_x)_x$ at point x_0 is approximated

by

$$\frac{1}{h^2}\Big(-a(x_0-h/2)u(x_0-h) + c\ u(x_0) - a(x_0+h/2)u(x_0+h)\Big) \tag{137}$$

with $c = a(x_0-h/2) + a(x_0+h/2)$.



1: $a = 10^{-3}$, $b = 10^{-3}$
2: $a = 10^{+3}$, $b = 10^{-3}$
3: $a = 10^{-3}$, $b = 10^{+3}$
4: $a = 1$, $b = 1$

Figure 26: Distribution of coefficients

Besides discontinuous changes in the size of the coefficients by orders of magnitude, the resulting equations are strongly anisotropic near the boundary, with the strong connectivity being in the direction of the boundary. Regarding the treatment of such problems by geometric multigrid methods, both properties require special attention. In order to see the similarities between geometric and algebraic multigrid if applied to such problems, we briefly recall a typical geometric approach.

First, assuming usual $h \to 2h$ coarsening, smoothing needs to be done by "robust" smoothers such as alternating line relaxation. Second, geometric (linear) interpolation of corrections is not appropriate any more. This is because linear interpolation for u at a grid point x_0 requires the continuity of its first derivatives. However, in our example, u_x and u_y are not continuous but rather au_x and bu_y. Since corresponding corrections exhibit the same discontinuous behavior, proper interpolation has to approximate the continuity of au_x and bu_y rather than that of u_x and u_y. Consequently, we obtain a better interpolation, for instance in x-direction, if we start from the equation

$$(au_x)(x_0-h/2) = (au_x)(x_0+h/2)$$

and approximate this by

$$a(x_0-h/2)(u(x_0)-u(x_0-h)) = a(x_0+h/2)(u(x_0+h)-u(x_0))\ .$$

This yields the interpolation formula

$$c\ u(x_0) = a(x_0-h/2)u(x_0-h) + a(x_0+h/2)u(x_0+h) \tag{138}$$

for computing $u(x_0)$ from its neighbors.

The extension of such relations to both space dimensions forms the basis for the definition of "operator-dependent" interpolation in geometric multigrid. Clearly, by means of some additional approximations, the "interpolation pattern" has to be modified in order to match the coarse-grid points really available. Such an interpolation has first been investigated in [1]. In that paper, it was also shown that the use of operator-dependent interpolation gives most robust multigrid convergence if, in addition, Galerkin operators are used on coarser levels (rather than the coarse-level stencils corresponding to (136)).

The development of AMG can actually be regarded as the attempt to generalise the ideas contained in [1]. In fact, operator-dependent interpolation (138), motivated geometrically above, is nothing else but an approximation to the (homogeneous) difference equations (137). This is exactly the way AMG attempts to define interpolation. Clearly, in contrast to the geometric approach, line-relaxations are not required in AMG since anisotropies are "resolved" by coarsening essentially in the directions of strong connectivity (see Figure 28a).

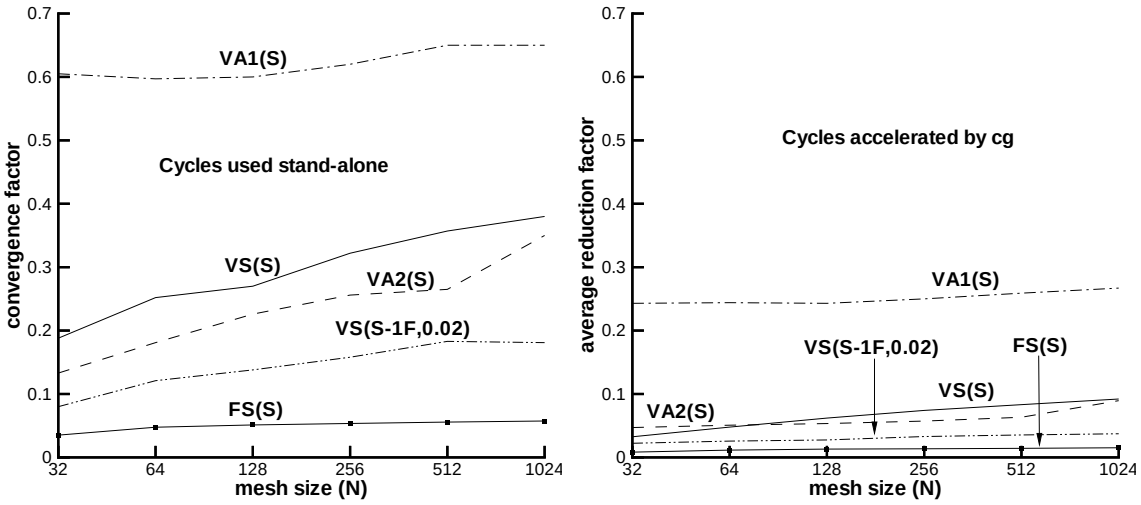
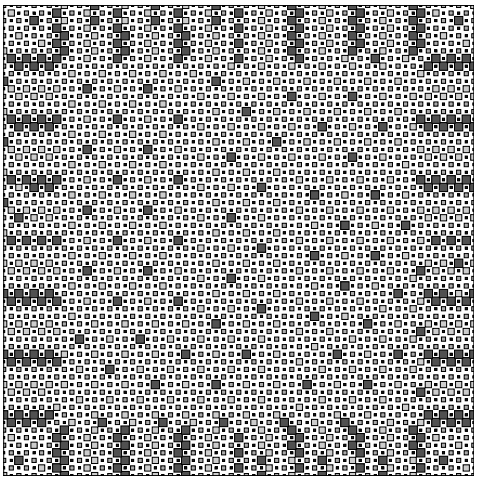


Figure 27: a) Convergence factors of cycles used stand-alone. b) Average reduction factors of accelerated cycles.

Figure 27a shows the convergence factors of the VS- and VA-cycles if applied to the above problem. The standard VS(S)-cycle converges somewhat slower than for the Poisson-like model problem and its rate exhibits a (slight) h-dependence. But even for the finest grid, it still converges at a rate of 0.38 per cycle. In contrast to the Poisson-like example, improving interpolation by one additional Jacobi-relaxation step (VS(S-1F)-cycle) substantially speeds up convergence. In fact, the improved interpolation restores the convergence observed for the standard interpolation in the Poisson-like case (cf. VS(S)-cycle in Figure 18a). The F-cycle convergence factor is around 0.05 and virtually constant for all grids considered. The VA1(S)-cycle converges asymptotically at about the same rate as in the Poisson case. Finally, the VA2-cycle converges even faster than the VS-cycle. However, this is unusual and cannot be expected in general.

As before, acceleration by conjugate gradient improves convergence substantially (s. Figure 27b). Moreover, convergence speed becomes virtually independent of the mesh size. How this improvement translates into number of iteration steps and computational time needed to solve (136) by 8 orders of magnitude (for $N = 512$ and starting with $u \equiv 0$ as first approximation), is shown in Figure 28b and Table 6. The accelerated VS-cycle requires just 8 iterations and is over 20 times faster than cg/ILU(0). The corresponding F-cycle even solves this problem in only 4 cycles and is approximately 23 times faster than cg/ILU(0). However, this extraordinarily rapid convergence is for similar reasons as already mentioned in Remark 8.4: Although the two-level method involving the first two

levels does no longer strictly correspond to a direct solver (due to the one-dimensional coarsening near the boundary, see Figure 28a), it is still very close since the F-points of the finest level are only (very) weakly connected.



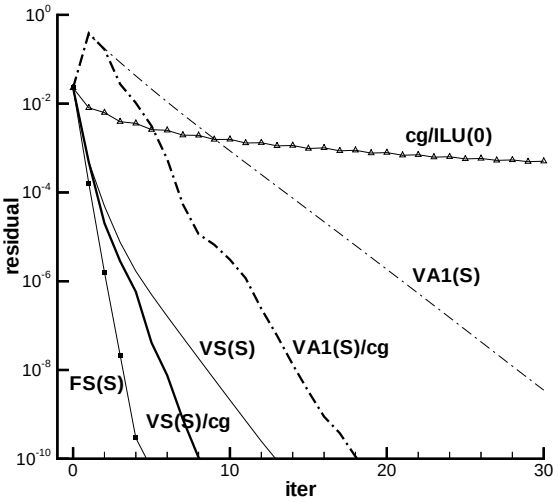


Figure 28: a) AMG standard coarsening. b) Convergence histories ($N = 512$).

	complexities		times (sec) / Pentium II, 300 MHz				
			setup	stand-alone		conjugate gradient	
method	c_A	c_G	time	cycle	$\varepsilon_0 = 10^{-8}$	cycle	$\varepsilon_0 = 10^{-8}$
ILU(0)			0.93			1.07	693.4 (647)
AMG1R5	2.58	1.81	6.87	2.19	41.9 (16)		
VS(S)	2.52	1.79	9.43	2.44	41.1 (13)	3.03	33.8 (8)
FS(S)	"	"	"	4.67	32.8 (5)	5.20	30.2 (4)
VA2(S)	2.14	1.58	8.78	2.08	29.6 (10)	2.68	30.3 (8)
VA1(S)	1.78	1.32	7.52	1.67	65.8 (35)	2.29	48.6 (18)
VS(S-1F,0.02)	3.20	1.79	25.9	2.79	48.2 (8)	3.40	46.3 (6)
VS(S-1P,0.02)	3.05	1.79	19.0	2.82	47.2 (10)	3.30	42.1 (7)

Table 6: Complexities and computing times ($N = 512$)

We note that the memory requirement for strongly anisotropic problems is typically higher than that for isotropic problems, the reason being that AMG will essentially perform one-dimensional coarsening (in the direction of strong connectivity). To a limited extent, this is also observed in Table 6. To reduce memory requirement in anisotropic areas, A1-coarsening needs to be employed and is quite effective. From the table we see that, compared to the VS-cycle, the VA1-cycle requires 50% less overhead memory at the expense of a 50% increase in execution time.

We have seen above that relaxation of interpolation improves convergence substantially. However, although the mere solution time of VS(S-1F,0.02) (*not* counting the setup) is significantly lower than that of VS(S), the table shows that this advantage is eaten up by a much higher setup cost. Unfortunately, this is also typical for more general situations.

Moreover, relaxation of interpolation naturally increases the memory requirement. As can be seen in Table 6, the VA1-cycle requires only one third of the memory overhead of the VS(S-1F,0.02)-cycle. As already mentioned at the beginning of Section 8, there is much room for optimizing the application of relaxation of interpolation, for instance, by applying it only locally where really needed and by locally optimizing truncation of interpolation.

Remark 8.10 The above test case was constructed as a worst-case example for the geometric multigrid code MG2 [49]. Although MG2 is normally very efficient in solving problems with discontinuous coefficients, for this very particular problem there exist eigenvalues of the MG2 iteration matrix which are very close to one. As a consequence, MG2 V-cycle convergence becomes extremely slow. Using MG2 as pre-conditioner substantially improves convergence. But still, on the order of 60 MG2 V-cycles have to be performed to reduce the residual by 8 orders of magnitude. Even the F-cycle still requires on the order of 40 cycles (see [49]). Compared to this, AMG does not show any particular problems for this test case and converges much faster. $\ll$

8.4.2 Oil reservoir simulation

In oil reservoir simulation, the basic task is to solve complex multiphase flows in porous media. For each phase, ℓ, of a multiphase problem, the governing equations are the *continuity equation*

$$-\nabla \cdot (\rho_\ell \, \mathbf{u}_\ell) = \frac{\partial}{\partial t}(\rho_\ell \, \phi \, S_\ell) + q_\ell \tag{139}$$

and *Darcy's law*

$$\mathbf{u}_\ell = -K \frac{k_{r\ell}}{\mu_\ell} (\nabla p_\ell - \rho_\ell \, g \, \nabla z) \; . \tag{140}$$

The continuity equation describes the mass conservation. For each phase, $\mathbf{u}_\ell$ denotes the velocity vector, S_ℓ the saturation, ρ_ℓ the density distribution (depending on p_ℓ) and q_ℓ represents injection or production wells. ϕ denotes the porosity of the medium. Darcy's law essentially describes the velocity-pressure dependence. Here, p_ℓ denotes the pressure, μ_ℓ the viscosity and $k_{r\ell}$ the relative permeability (depending on S_ℓ). g is the gravity acceleration constant (we here assume that gravity acts in the direction of the z-axis). Finally, K is a tensor (absolute permeability). The absolute permeability varies in space by, typically, several orders of magnitude in a strongly discontinuous manner. In Figure 29, the gray scale indicates the variation of the permeability as a function of space for a typical case.

By inserting (140) into (139), the phase velocities can be eliminated. For *incompressible flows* we can assume $\partial\phi/\partial t = 0$ and $\partial\rho_\ell/\partial t = 0$, and one obtains the following equations involving pressures and saturations

$$\nabla \cdot \Big(K \frac{k_{r\ell}}{\mu_\ell} (\nabla p_\ell - \rho_\ell \, g \, \nabla z) \Big) = \phi \, \frac{\partial}{\partial t} S_\ell + q_\ell / \rho_\ell \; . \tag{141}$$

From this set of equations, pressures and saturations can be computed if one takes into account that $\sum_\ell S_\ell \equiv 1$ and that the individual phase pressures are directly interrelated by

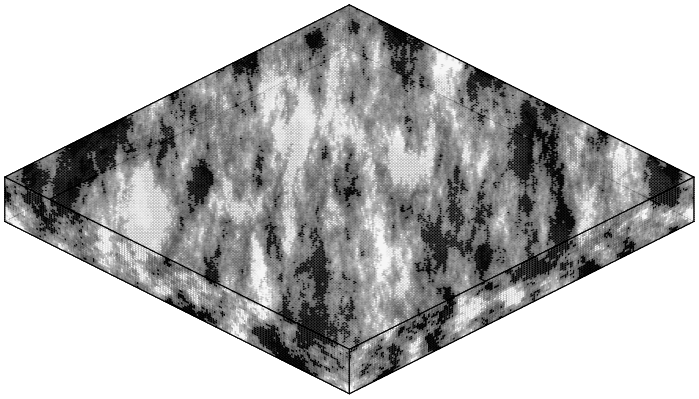


Figure 29: Distribution of permeability as a function of space (logarithmic gray scale)

means of simple non-PDE relations involving known (but saturation-dependent) capillary pressures.

Solving the finally resulting non-linear system *fully implicitly* is rather expensive and currently strongly limits the size of problems which can be handled. The more classical IMPES approach (implicit in pressure, explicit in saturation) treats (141) by an *explicit* time-stepping. Consequently, in each time step, the pressures need to be computed with all saturations being known from the previous time step. Exploiting the interrelation of the individual phase pressures mentioned above, only *one* pressure (for instance, the oil pressure) requires the solution of a partial differential equation of the form

$$-\nabla \cdot (T \nabla p) = Q \tag{142}$$

which is obtained by adding up the individual equations (141) (and using $\sum_\ell S_\ell \equiv 1$). The tensor T is directly related to K. According to the assumption of incompressibility, both T and Q depend only on the saturations and given quantities.

Clearly, as with any explicit time-stepping method, the major drawback of the IMPES approach is the serious restriction in the maximally permitted time step size (CFL condition). Since this restriction gets increasingly strong with decreasing spatial mesh size or increasing variation in the magnitude of T, the classical IMPES method also strongly limits the treatment of large problems in practice.

Recently, however, a new IMPES-type approach has become quite popular which eliminates the time step restriction due to the CFL condition. Rather than updating the saturations directly on the grid based on (141), a *streamline method* is used instead [71, 9]. By transporting fluids along periodically changing streamlines, the streamline approach is actually equivalent to a dynamically adapting grid that is decoupled from the underlying, static, grid used to describe the reservoir geology (and to compute the pressure). The 1D nature of a streamline allows decoupling the 3D problem into multiple 1D problems. The main advantage of this approach is that the CFL conditions are eliminated from the fluid transport, allowing global time step sizes that are independent of the underlying grid constraints.

Although this approach cannot (yet) be applied to all relevant situations occuring in oil-reservoir simulation, it is well suited for large heterogeneous multi-well problems that

are convectively dominated. This has been demonstrated in [9] for a problem consisting of one million mesh cells. Cases of this size could not be solved as easily and quickly by standard implicit methods. Clearly, an efficient solver for the pressure equation (142) then becomes highly important.

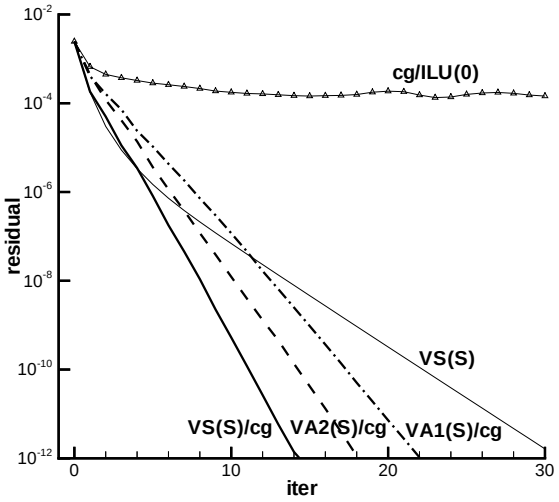


Figure 30: Convergence histories (one million cell case)

	complexities		times (sec) / IBM PowerPC, 333 MHz				
			setup	stand-alone		conjugate gradient	
method	c_A	c_G	time	cycle	$\varepsilon_0 = 10^{-10}$	cycle	$\varepsilon_0 = 10^{-10}$
ILU(0)			3.89			3.74	3376. (902)
AMG1R5	7.66	2.21	167.	22.7	758.6 (26)		
VS(S)	2.85	1.56	43.8	10.5	401.2 (34)	12.6	245.1 (16)
VA2(S)	2.56	1.39	41.6	9.20	492.2 (49)	11.3	267.3 (20)
VA1(S)	1.41	1.13	26.7	5.56	476.4 (81)	7.67	210.7 (24)

Table 7: Complexities and computing times (one million cell case)

The one million cell case previously mentioned, provided by StreamSim Technologies (CA, USA), has been used as a test case for AMG. The variation of the absolute permeability, which directly corresponds to a discontinuous variation of the coefficients in the resulting matrix by four orders of magnitude, is shown in Figure 29. Figure 30 shows the convergence histories of the typical AMG cycles for this case (starting with the zero first approximation). We see that all cycles presented show essentially the same convergence behavior as for the Poisson-like problems considered in Section 8.3.2 (cf. Figure 24). This demonstrates the robustness of AMG with respect to strong, discontinuous variations in the matrix coefficients.

Table 7 presents some detailed measurements. For all cycles considered, we see a substantial benefit by using them as pre-conditioners rather than stand-alone. The accelerated standard VS(S)-cycle takes 16 iterations to reduce the residual by 10 orders of magnitude. Although the lowest-memory cycle, VA1(S)/cg, converges more slowly (24

iterations), in terms of total computation time it is fastest and about 16 times faster than cg/ILU(0). Its complexity value $c_A = 1.41$ is very reasonable. Note, however, that the memory reduction by aggressive A2-coarsening is not very effective, the reason being the strong anisotropies in the problem. On the whole, the AMG performance is very much comparable to that shown in Table 4 for Poisson-like cases.

The performance of AMG1R5 shows that the original interpolation (cf. Remark 7.8) leads to unacceptably high memory requirements for this example: $c_A = 7.66$ as compared to $c_A = 2.85$ for VS(S). As a consequence, although AMG1R5 converges faster than VS(S), its efficiency is substantially lower.

8.4.3 Electromagnetic systems

In this section we consider a synchronous line-start motor excited with permanent magnets. The knowledge of the magnetic field inside such a motor, induced by the currents in the stator and the magnets in the rotor, allows its optimization w.r.t. functionality and efficiency. For an example, see Figure 31a.

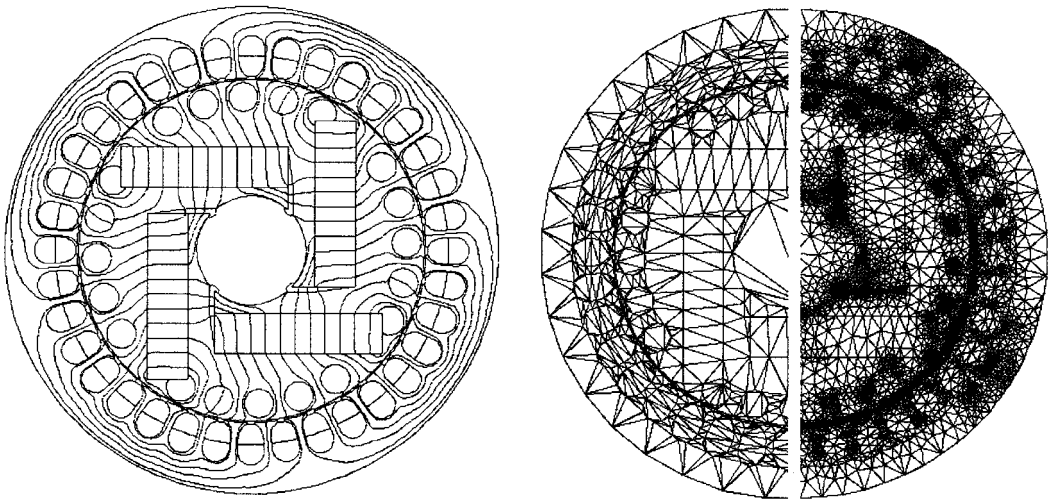


Figure 31: Synchronous line-start motor: a) magnetic field plot, b) initial and locally refined mesh [42].

The governing equation is the *magnetostatics Maxwell equation* (also known as *Ampere's law*)

$$\nabla \times H = J \tag{143}$$

where H denotes the *magnetic field intensity* and J the *electric current density*. According to Maxwell's equation for the *magnetic flux density*,

$$\nabla \cdot B = 0\,,$$

we know that there is a *magnetic vector potential*, A, such that $B = \nabla \times A$. Observing finally the constitutive relation $B = \mu H$ with μ being the *permeability*, (143) can be

re-written as

$$\nabla \times (\nu \nabla \times A) = J \tag{144}$$

where $\nu = 1/\mu$ is the *reluctivity*.

We here consider only 2D intersections (Cartesian coordinates) and, for reasons of symmetry, we can assume A and J to be of the special form

$$A = (0,\, 0,\, u(x,y)) \;, \quad J = (0,\, 0,\, f(x,y)) \;.$$

Hence, (144) can be seen to correspond to a scalar diffusion equation for the (z-component of the) magnetic potential, namely,

$$-\nabla \cdot (\nu \nabla u) = f \;. \tag{145}$$

For isotropic materials, the reluctivity ν is a scalar quantity. Normally, it is a function of u and (145) has to be solved by some outer linearization (for instance, Newton's method). More importantly, however, ν is strongly discontinuous and differs by three orders of magnitude between the steel and air areas inside the motor.

An accurate solution of (145) by finite elements requires local refinements near the critical areas (see Figure 31b). Instead of solving the discrete equations on the full circular domain (with Dirichlet boundary conditions), one may solve it more efficiently on half the domain using *periodic* boundary conditions or on a quarter of the domain using *anti-periodic* bounday conditions.

Figures 32a-b show the performance of AMG for both the periodic and the anti-periodic case. Except that the low-memory cycle converges somewhat slower here, the overall performance is very much comparable to the case considered in the previous section. The underlying mesh, provided by the Dept. of Computer Science of the Katholieke Universiteit Leuven, is shown in Figure 1 (half the domain). The coarser levels produced by AMG's standard coarsening are depicted in Figure 3.

We point out that *anti-periodic* boundary conditions cause strong *positive* connections to occur in the underlying matrix (in all equations which correspond to points near the anti-periodic boundary). As theoretically discussed in Section 4.2.3, interpolation should take such connections properly into account, for instance, in the way as described in Section 7.1.3. If this is ignored, that is, if interpolation is done only via strong *negative* couplings, the discontinuous behavior of corrections across the anti-periodic boundary is not properly reflected by the interpolation, leading to a substantial degradation of the method.

This is demonstrated in Figure 32c. We observe that the standard VS(S)-cycle, without acceleration, hardly converges. In fact, convergence speed is limited by the convergence of the smoother on the finest level. It is heuristically clear that this slow convergence is only caused by particular error components, namely, those which really exhibit the discontinuity. All other components are still reduced very effectively. Consequently, the use of accelerated cycles "cures" this problem to some extent as can be seen from the figure. However, it takes 10 "wasted" cycles before the critical error components are sufficiently reduced by conjugate gradient and the AMG performance becomes visible again. Although this demonstrates that the use of accelerators such as conjugate gradient helps

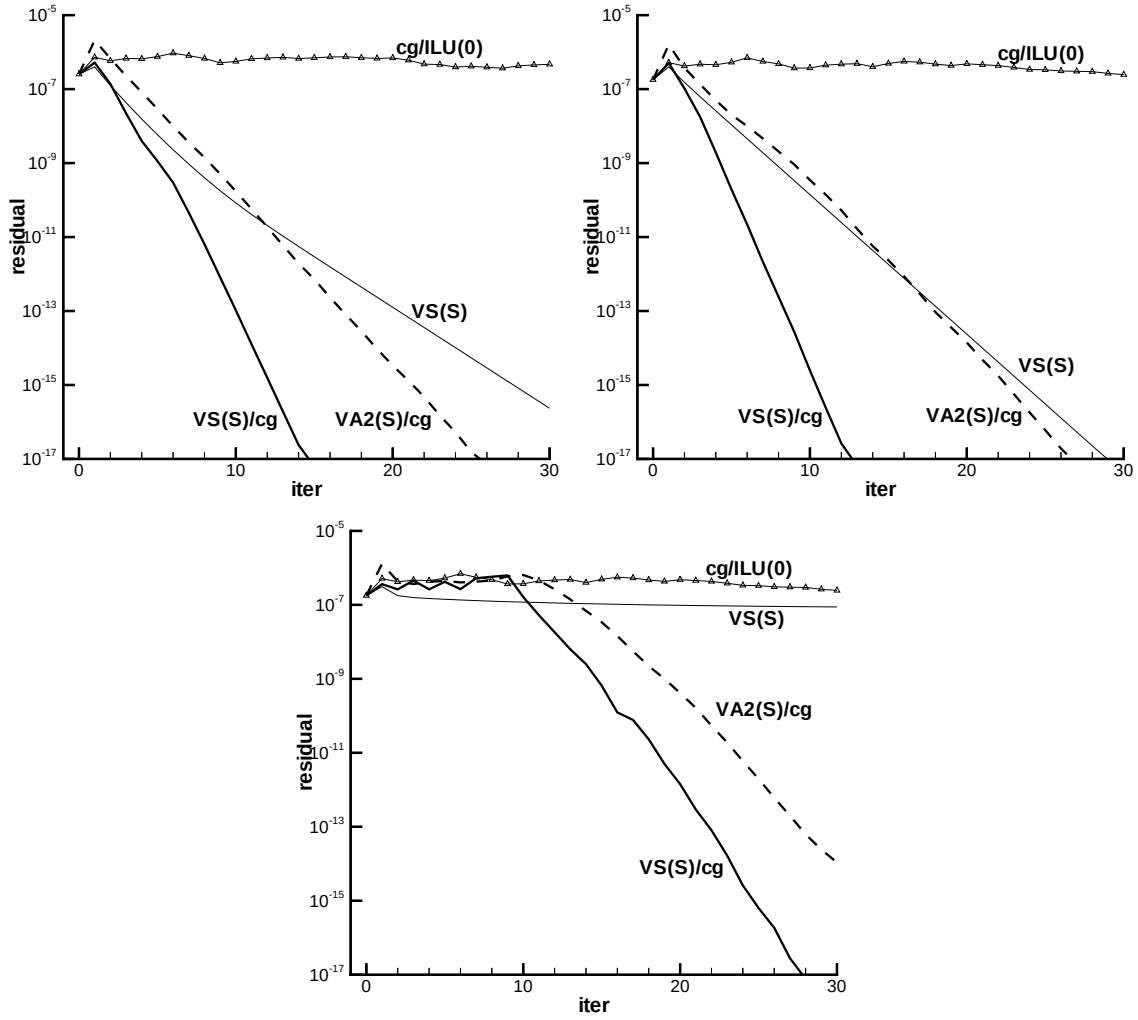


Figure 32: Convergence histories: a) periodic case, b) anti-periodic case, c) anti-periodic case (positive connections ignored)

to stabilize convergence, a situation like that shown here should clearly be avoided since it demonstrates that something is wrong conceptually. Moreover, the area of "stalling" convergence (here just the first 10 iterations) strongly depends on the problem and the size of the grid (more precisely, on the distribution of those eigenvalues of the AMG iteration matrix which are close to one).

8.5 Further model problems

8.5.1 Special anisotropic problems

We have seen before that AMG treats anisotropies by coarsening in the proper direction. This works perfectly if the anisotropies are essentially aligned with the grid. Moreover, since AMG adjusts its coarsening *locally*, diffusion equations (136) with strongly varying anisotropies are no problem for AMG.

However, one has to expect certain difficulties if strong anisotropies are *not* aligned with the grid. The following model problem is a well-known test case for such a situation:

$$-(c^2+\varepsilon s^2)u_{xx} + 2(1-\varepsilon)scu_{xy} - (s^2+\varepsilon c^2)u_{yy} = f(x,y) \tag{146}$$

with $s = \sin\alpha$ and $c = \cos\alpha$. We consider this differential operator on the unit square with $f(x,y) \equiv 1$, homogeneous Dirichlet boundary conditions, $\varepsilon = 10^{-3}$ and $0° \le \alpha \le 90°$. For such values of α, u_{xy} is most naturally discretized by the *left-oriented* 7-point stencil

$$\frac{1}{2h^2}\begin{bmatrix} -1 & 1 & \\ 1 & -2 & 1 \\ & 1 & -1 \end{bmatrix} \tag{147}$$

where $h = 1/N$.

The main difficulty with the differential operator (146) is that it corresponds to the operator $-u_{ss} - \varepsilon u_{tt}$ in an (s,t)-coordinate system obtained by rotating the (x,y)-system by an angle of α ("rotated anisotropic diffusion equation" [81]). That is, (146) is strongly anisotropic with the direction of strong connectivity given by the angle α (see Figure 33).

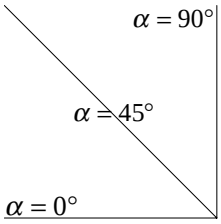


Figure 33: Direction of strong connectivity ($\varepsilon \ll 1$)

In particular, for $\alpha = 0°$ and $\alpha = 90°$, (146) becomes $-u_{xx} - \varepsilon u_{yy} = f$ and $-\varepsilon u_{xx} - u_{yy} = f$, respectively, and the anisotropies are just aligned with the axes. Geometric multigrid methods solve these equations very efficiently by employing $h \to 2h$ coarsening and using line relaxations (in the direction of strong connectivity) for smoothing. In contrast to this, AMG uses point relaxation for smoothing but coarsens in the direction of strong connectivity. The standard VS(S)-cycle, for instance, converges at a rate of 0.1 per cycle, independent of the grid size. This can be seen from Figure 34a. Using acceleration by conjugate gradient even gives a convergence factor better than 0.01 per cycle (see Figure 34b).

The figure shows that AMG performs similarly well for $\alpha = 45°$ also. In this case, the discretisation of (146) corresponds to the stencil

$$\frac{1}{h^2}\begin{bmatrix} -\frac{1-\varepsilon}{2} & -\varepsilon & \\ -\varepsilon & 1+3\varepsilon & -\varepsilon \\ & -\varepsilon & -\frac{1-\varepsilon}{2} \end{bmatrix}, \tag{148}$$

which exhibits a strong connectivity in the diagonal direction. It has only non-positive off-diagonal entries and essentially degenerates to a 3-point stencil for small ε. Since the anisotropy is still aligned with the grid, AMG can cope with this anisotropy as efficiently as in the previous cases by coarsening in diagonal direction. Nevertheless, solving this case

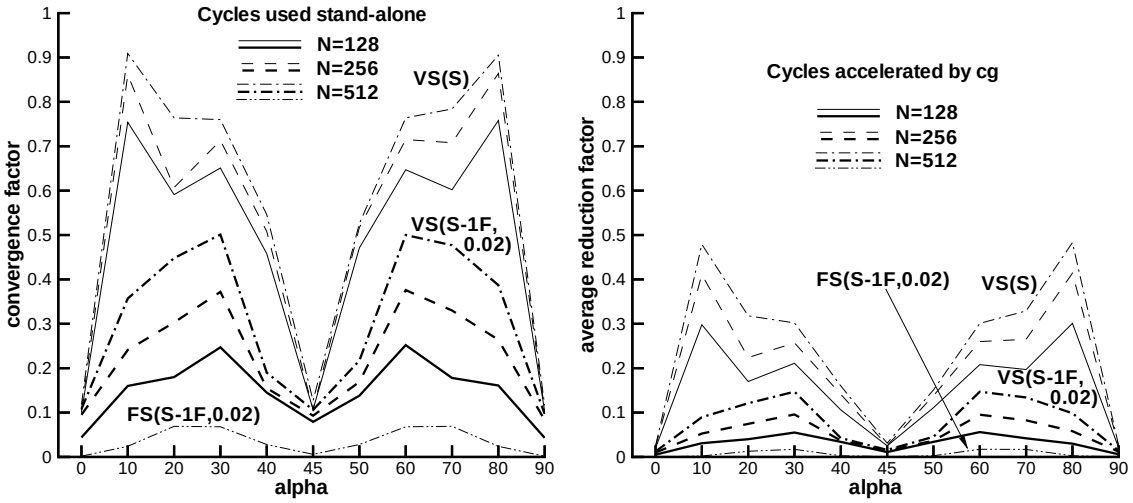


Figure 34: a) Convergence factors of cycles used stand-alone. b) Average reduction factors of accelerated cycles.

with geometric multigrid (using $h \to 2h$ coarsening), brings a problem: Even alternating line relaxation does not have good smoothing properties any more. In fact, it is no better than point relaxation (since the connections in both coordinate directions are very weak).

For other values of α, the strong anisotropies are no longer aligned with the grid. (Note also that, generally, the resulting discretization matrices are not M-matrices.) This causes particular difficulties for *any* multigrid method. In geometric multigrid, as above, neither point- nor line-relaxation schemes have good smoothing properties with respect to $h \to 2h$ grid coarsening. More importantly, however, the extent to which the anisotropy is captured by grid points, strongly depends on α and is different on different grid levels. This substantially reduces the effectiveness of coarse-grid correction processes and, through this, the overall cycle convergence.

Since AMG cycles also obtain their correction quantities from points which form subgrids of the given grid, the non-alignment influences the AMG performance, too. This is demonstrated in Figure 34a which, for $N = 128, 256, 512$ and different cycles, shows convergence factors as a function of α. One sees that, for certain values of α, the standard VS(S)-cycle (upper three curves in the figure) converges very slowly and the convergence factor depends on the mesh size h. For instance, for $\alpha = 10°$ and $N = 512$, the convergence factor is worse than 0.9. We note that the convergence of the corresponding F-cycle (not shown here) is faster but still shows a similar h-dependency for most values of α.

This confirms that the slow convergence is not (only) due to the mere accumulation of errors (introduced by the inaccurate solution of the coarse-level correction equations in a V-cycle) but that there are also convergence problems for the "intermediate" two-level methods. Consequently, an improvement of interpolation by relaxation should help. Although the corresponding VS(S-1F,0.02)-cycle indeed converges much better than the standard VS(S)-cycle, it still shows a significant h-dependency as can be seen from the figure. However, using the improved interpolation in conjunction with the F-cycle eliminates this problem: the FS(S-1F,0.02)-cycle converges at a rate which is the same for all

mesh sizes and better than 0.1 for all α considered.

Figure 34b shows average reduction factors obtained for the corresponding cycles if used as pre-conditioner for conjugate gradient. Although the shapes of the curves are similar to the previous ones, the accelerated cycles converge much faster. In particular, the accelerated F-cycle with improved interpolation converges at a rate ≤ 0.02 for all α and h considered.

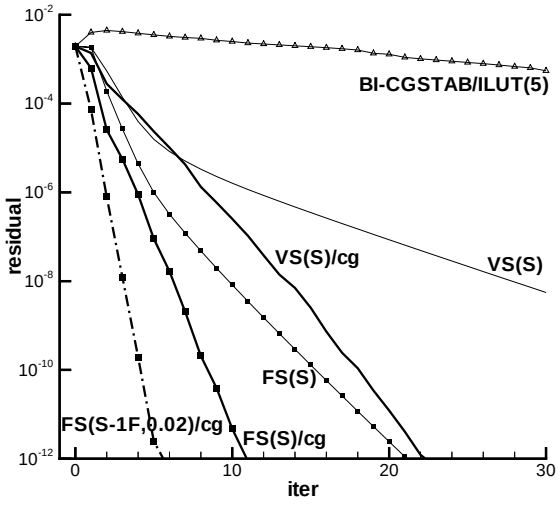


Figure 35: Convergence histories ($N = 512$, $\alpha = 20°$)

Finally, Figure 35 shows convergence histories for $N = 512$ and $\alpha = 20°$, starting with $u = 0$ as first approximation. One observes that the VS(S)-cycle converges rapidly for the first few cycles before convergence levels off and reaches its slow asymptotic value. This effect is virtually eliminated for the corresponding accelerated cycle. To a lesser extent, a similar effect occurs also for the FS(S)-cycle. The accelerated F-cycle with improved interpolation reduces the residual by 10 orders of magnitude in 6 iteration steps only.

We see that it is easy to obtain fast convergence even in this example which can be regarded as very difficult for geometric multigrid methods. However, as can be seen from Table 8, cost and memory requirement become rather high if relaxation of interpolation is employed: an operator complexity c_A of over 6 is unacceptably high for practical applications. Related to this is also a setup cost which is much higher than for the other cycles. The table shows that, although the cycles with standard interpolation converge much slower, in terms of total computational work they are still more efficient, the best one being the accelerated VA2-cycle. The following remark outlines the main reason for the particularly high memory requirement observed in this example:

Remark 8.11 We have already mentioned earlier that, compared to isotropic problems, memory requirement is generally higher for anisotropic problems. Memory requirement increases further for problems as discussed here where anisotropies are not aligned with the grid. In fact, Table 8 shows that, using standard coarsening and standard interpolation, the operator complexity is $c_A = 3.24$ while for the Poisson-like problem discussed earlier, it was only $c_A = 2.38$ (see Table 1). The major reason is that the non-alignment causes the strong connections to "fan out" so that each point is strongly connected to an increasing

method	complexities		times (sec) / Pentium II, 300 MHz				
			setup	stand-alone		conjugate gradient	
	c_A	c_G	time	cycle	$\varepsilon_0 = 10^{-10}$	cycle	$\varepsilon_0 = 10^{-10}$
AMG1R5	3.02	1.84	12.1	3.07	414. (131)		
VS(S)	3.24	1.84	18.6	3.55	263.6 (69)	4.22	119.7 (24)
FS(S)	"	"	"	6.87	183.4 (24)	7.49	108.4 (12)
VA2(S)	1.72	1.41	11.6	2.25	153.1 (63)	2.92	75.8 (22)
VS(S-1F,0.02)	6.05	1.84	91.9	5.65	199.2 (19)	6.45	162.9 (11)
FS(S-1F,0.02)	"	"	"	13.1	210.0 (9)	13.8	174.4 (6)
VS(D-1F,0.5)	2.54	1.88	15.4	3.13	125.1 (35)	3.82	76.5 (16)
FS(D-1F,0.5)	"	"	"	5.91	109.9 (16)	6.53	74.2 (9)

Table 8: Complexities and computing times ($N = 512$, $\alpha = 20°$)

number of points on both sides of the "true" anisotropic line. Thus interpolation stencils get larger on coarser levels and, as an immediate consequence, so do the Galerkin operators. Clearly, this effect is strongly amplified by relaxation of interpolation. ≪

The fan-out effect mentioned in this remark may be reduced by choosing a larger truncation value for interpolation, ε_{tr}. Indeed, this dramatically improves efficiency as seen from the last two rows in Table 8 where we have chosen $\varepsilon_{tr} = 0.5$: compared to the case $\varepsilon_{tr} = 0.02$, the operator complexity is reduced from 6.05 to 2.54 and the total execution time is reduced by more than a factor of two! (Note that we also have used the simpler direct interpolation instead of the standard one.)

In this context we want to recall that our main goal in this chapter on applications is to demonstrate how different AMG components may influence the overall performance. We have not tried to find optimal components or parameters but rather confined ourselves to a few typical ones. The above results clearly show that optimized parameter settings may, depending on the application, improve the performance substantially further.

Remark 8.12 If, instead of the 7-point discretisation (147), we use the standard 9-point discretization, the AMG convergence behavior is qualitatively the same (except that $\alpha = 45°$ does not play such a particular role any more). Generally, the performance of AMG suffers from the non-alignment of the anisotropies just as geometric multigrid does. However, in AMG, due to its higher flexibility in creating the coarser levels, this problem is much less severe and easy to cure in the cases considered here, at least in terms of robust convergence. Concerning the convergence behavior of geometric multigrid in such cases, see, for example, [49]. ≪

8.5.2 Convection-diffusion problems

So far, we have only considered symmetric problems. However, as mentioned earlier, RAMG05 does not make use of the symmetry and can formally also be applied to non-symmetric problems. Practical experience has shown that, generally, the non-symmetry by itself does not necessarily cause particular problems for AMG. Other properties of the

given system typically influence the performance of RAMG05 to a much larger extent, for instance, whether or not the underlying matrices are (approximately) weakly diagonally dominant. If this is strongly violated, there is no guarantee that the method converges.

We are not going to discuss non-symmetric problems in detail here but rather present results for one typical example from a class of non-symmetric problems for which AMG has turned out to yield robust and fast convergence. This is the class of convection-dominant equations

$$-\varepsilon\,\Delta u \;+\; a(x,y)\,u_x \;+\; b(x,y)\,u_y \;=\; f(x,y) \tag{149}$$

with some small $\varepsilon > 0$, discretised by standard first order upwind differences. Note that the resulting discretization matrices are off-diagonally negative.

Generally, for such equations, AMG converges very quickly, in particular, if the characteristics are straight lines. Heuristically, this is an immediate consequence of AMG's coarsening strategy. Since strong connections are only in the upstream direction, interpolation to any F-point i will typically use relatively many of them for interpolation and, consequently be rather accurate in the sense of AMG. Thus, the reason for fast convergence is essentially algebraic, and not a consequence of smoothing in the usual sense.

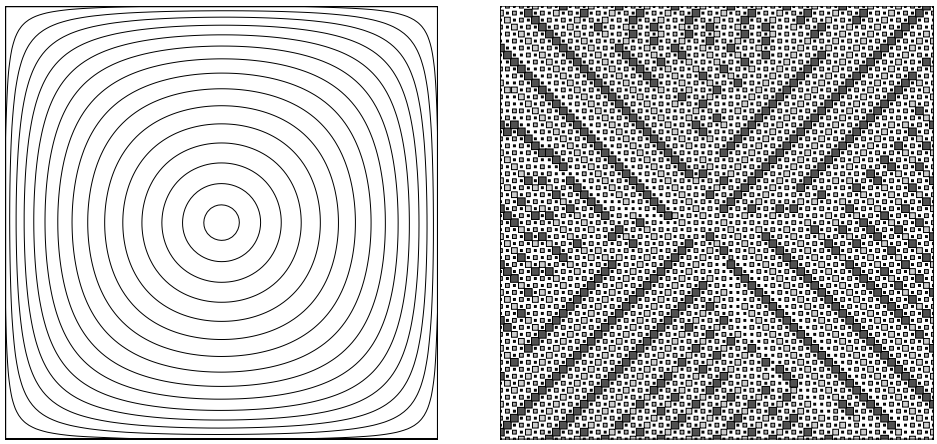


Figure 36: a) Solution contours, b) standard coarsening pattern.

When the characteristics change over the region, the simple directionality of the strong connections on coarser grids is lost, and AMG will exhibit a more typical convergence behavior. As an example, we here consider a worst-case problem given by selecting

$$a(x,y) = -\sin(\pi x)\cos(\pi y) \quad \text{and} \quad b(x,y) = \sin(\pi y)\cos(\pi x)\;, \tag{150}$$

$f(x,y) \equiv 1$ and $u = \sin(\pi x) + \sin(13\pi x) + \sin(\pi y) + \sin(13\pi y)$ on the boundary of the unit square. Finally, we set $\varepsilon = 10^{-5}$.

The major difficulty with this particular example is that a and b are chosen to yield *closed characteristics* and a stagnation point in the center of the domain. Consequently, (149) becomes more and more singular for $\varepsilon \longrightarrow 0$. For $\varepsilon = 0$, the continuous problem is no longer well defined: any function which is constant along the characteristic curves, solves the homogeneous equation. According to the results shown in [49], geometric multigrid

approaches have serious difficulties with this example: covergence becomes very slow and mesh dependent.

Figure 36 depicts the coarsening strategy performed by AMG. Since strong connectivity is in the circular direction only, AMG attempts to not coarsen in the radial direction (within the limits imposed by the grid).

Figure 37a shows average reduction factors of some AMG cycles if used as a preconditioner for BI-CGSTAB [72]. In all cases, instead of CF-relaxation, we employed symmetric Gauss-Seidel relaxation for smoothing (which usually gives faster convergence for convection dominant problems). The figure shows rapid convergence in all cases. In particular, the cycles using standard coarsening, on the average reduce the residual by approximately two orders of magnitude per BI-CGSTAB iteration. (Note that each BI-CGSTAB iteration involves the performance of *two* AMG cycles.) The VA2(S)-cycle still converges very fast (better than 0.1 reduction per iteration) but exhibits a relatively significant h-dependency. Figure 37b shows the convergence histories of cycles with and without acceleration.

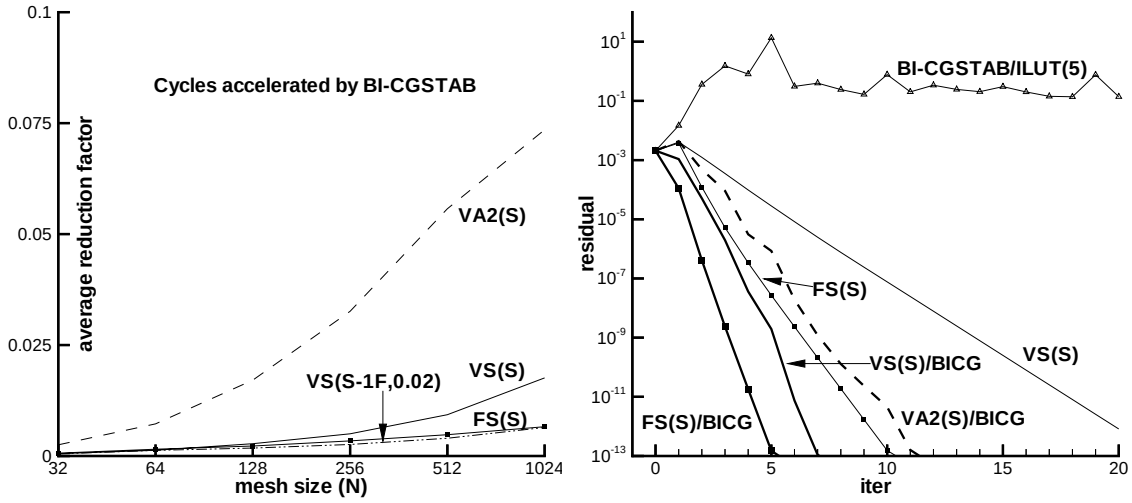


Figure 37: a) Average reduction factors, b) Convergence histories for $N = 512$.

	complexities		times (sec) / Pentium II, 300 MHz				
			setup	stand-alone		BI-CGSTAB	
method	c_A	c_G	time	cycle	$\varepsilon_0 = 10^{-10}$	cycle	$\varepsilon_0 = 10^{-10}$
AMG1R5	3.94	2.11	9.72	2.94	180.2 (58)		
VS(S)	3.33	1.92	12.0	3.48	88.6 (22)	7.52	64.8 (7)
FS(S)	"	"	"	8.34	95.4 (10)	17.3	98.4 (5)
VA2(S)	2.58	1.68	9.62	2.87	107.1 (34)	6.22	78.0 (11)
VS(S-1F,0.02)	5.14	1.94	89.3	4.93	138.6 (10)	10.4	141.2 (5)

Table 9: Complexities and computing times ($N = 512$)

Table 9 shows detailed performance measurements for $N = 512$. The accelerated VS(S) cycle requires 7 iterations to reduce the residual by 10 orders of magnitude, the accelerated

FS(S)- and VS(S-1F)-cycles require even only 5 cycles. In terms of total cost, however, the accelerated standard cycle is most efficient. Note that acceleration by BI-CGSTAB is not really effective here for those cycles which exhibit a fast stand-alone convergence (FS(S) and VS(S-1F)): although acceleration reduces the number of iterations by a factor of two, there is no gain in computational time since, as mentioned above, each BI-CGSTAB iteration requires the performance of two AMG cycles. However, for the other cycles, acceleration *is* beneficial. Regarding the relatively high memory requirement observed in the table, note that we have a similar "fan out" effect (in the upstream direction) as described in the previous section (cf. Remark 8.11).

8.5.3 Indefinite problems

We consider the Helmholtz equation (with constant c)

$$-\Delta u - cu = f(x,y) \quad (c \geq 0) \tag{151}$$

on the unit square with homogeneous Dirichlet boundary conditions. Discretization is on a regular mesh with *fixed* mesh size $h = 1/N$ using the standard 5-point stencil,

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

The corresponding discretization matrix, A_c, is non-singular as long as c does not equal any of the eigenvalues

$$\lambda_{n,m} = \frac{2}{h^2}(2 - \cos n\pi h - \cos m\pi h) \quad (n,m = 1,2,...,N-1) \tag{152}$$

of the corresponding discrete Poisson operator, A_0. If $c = \lambda_{n,m}$ $(= \lambda_{m,n})$, A_c is singular and its nullspace is spanned by the eigenfunctions

$$\phi_{n,m} = \sin(n\pi x)\sin(m\pi y) \quad \text{and} \quad \phi_{m,n} = \sin(m\pi x)\sin(n\pi y) \tag{153}$$

of A_0 corresponding to the eigenvalue $\lambda_{n,m}$.

A_c is positive definite as long as c is smaller than the first eigenvalue of A_0, that is, if $c < \lambda_{1,1}$. However, according to our comments in Section 4.2.1 (see Example 4.1), we have to expect a performance degradation of AMG if c approaches $\lambda_{1,1}$. This is demonstrated in Figure 38 where the convergence factor of the VS(S)-cycle (used stand-alone) is depicted as a function of c (for $h = 1/256$). Indeed, if c approaches $\lambda_{1,1} = 2\pi^2 + O(h^2)$, AMG's convergence factor tends to one. We have shown in Example 4.1 that, in order to avoid this, interpolation in AMG necessarily would have to approximate $\phi_{1,1}$ increasingly better if $c \to \lambda_{1,1}$.

If $c > \lambda_{1,1}$, A_c is no longer positive definite. Nevertheless, Figure 38 shows that AMG converges at a slightly reduced rate as long as c remains well between the first two eigenvalues (the second eigenvalue being $\lambda_{1,2} = \lambda_{2,1} \approx 49.4$). Increasing c further, we see that AMG converges as long as c remains well between consecutive eigenvalues, but that this convergence becomes poorer and poorer. By the time c approaches the 6th eigenvalue ($c \approx 150$), AMG diverges, even for c in the "valleys" between the eigenvalues. The reason

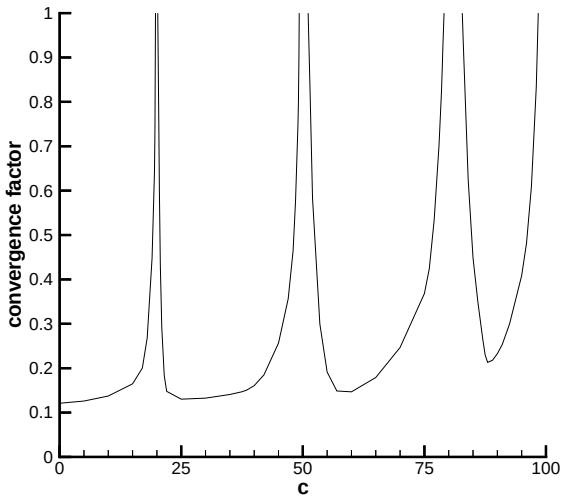


Figure 38: Convergence factor of stand-alone VS(S)-cycle as a function of c (h=1/256)

for this degradation is the fact that Gauss-Seidel relaxation, although it still has good smoothing properties (on the finer levels), diverges for all (smooth) eigenfrequencies $\phi_{n,m}$ with $\lambda_{n,m} < c$. Consequently, as in usual geometric multigrid, the overall method will still converge as long as the coarsest level used is fine enough to represent these smooth eigenfrequencies sufficiently well (and a direct solver is used on that coarsest level). That is, the size of the coarsest level limits the convergence of AMG when c gets larger: the more variables are represented on the coarsest level, the higher the value of c for which AMG converges. (In the above computations, we used five AMG levels resulting in a coarsest grid containing 500 variables.)

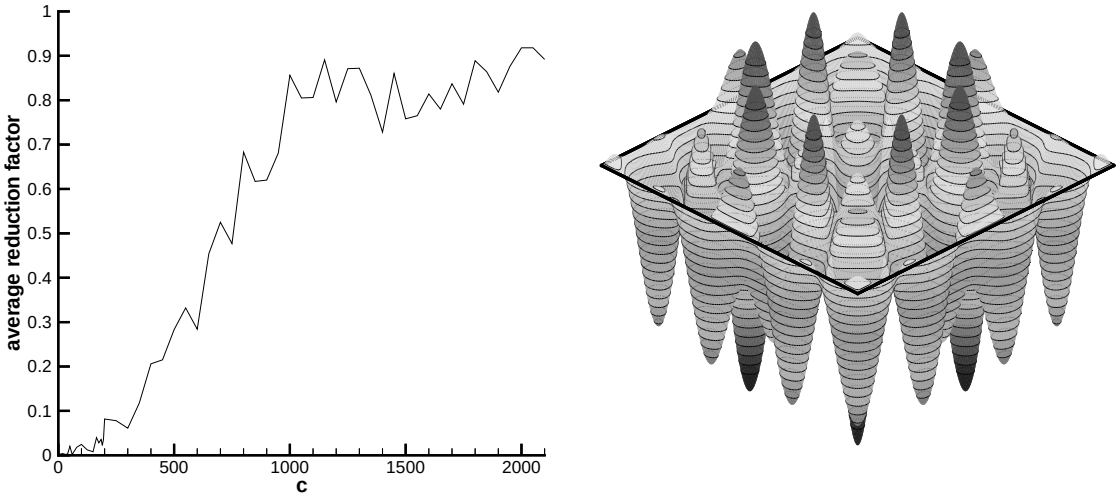


Figure 39: a) Average reduction factor of the VS(S)/BI-CGSTAB-cycle as a function of c (h=1/256). b) Solution of (151) for $f(x,y) \equiv 1$ and c=1000.

If we use the VS(S)-cycle as a pre-conditioner for BI-CGSTAB rather than stand-alone,

we obtain reasonable convergence for much larger values of c. This is demonstrated in Figure 39a which shows the average reduction factor of VS(S)/BI-CGSTAB as a function of c in solving the homogeneous problem with the first approximation being constant to one. (The values shown are the average reduction factors *per BI-CGSTAB iteration* – each of which requires two AMG iterations – observed in reducing the residual by 8 orders of magnitude.) The figure shows that acceptable convergence is achieved up to $c \approx 1000$ (close to the 40th eigenvalue of A_0). Figure 39b shows the solution of (151) for $f(x, y) \equiv 1$ and $c = 1000$.

For even larger values of c the average reduction factor per iteration is only $0.8 - 0.9$. Finally, for $c > 2100$, AMG breaks down since some diagonal entries in the coarsest level Galerkin operator become negative. As before, we used five levels for all computations. Decreasing the number of levels (i.e., increasing the number of variables on the coarsest level) gives convergence for even larger values of c. In any case, however, the size of c permitted will remain limited. We will not discuss this problem any further since it is well-known that the efficient solution of Helmholtz equation with very large values of c requires different algorithmical approaches [16].

9 Aggregation-based AMG

In this section, we consider a particularly simple limiting case of the AMG approach discussed in this paper, namely, the case that interpolation is defined such that each F-variable interpolates from *exactly* one C-variable only. That is, although each F-variable i may have more than one connection to the set of C-variables, the sets of interpolatory variables, P_i, are restricted to contain exactly one C-variable each. According to Section 4.2, the corresponding interpolation weight should equal one if the i-th rowsum of the given matrix is zero. To simplify interpolation further, let us *always* define this weight to be one even if the i-th rowsum of the given matrix is not zero.

Consequently, the total number of variables can be subdivided into "aggregates" I_k where $k \in C$ and I_k contains (apart from k itself) all indices i corresponding to F-variables which interpolate from variable k (see Figure 40). With this notation, the computation of the Galerkin coarse-level operator now becomes very simple. One easily sees that

$$I_h^H \, A_h \, I_H^h = \left(a_{kl}^H \right) \quad \text{where} \quad a_{kl}^H = \sum_{i \in I_k} \sum_{j \in I_l} a_{ij}^h \quad (k, l \in C) \; , \tag{154}$$

that is, the coefficient a_{kl}^H is just the sum of all cross-couplings between I_k and I_l.

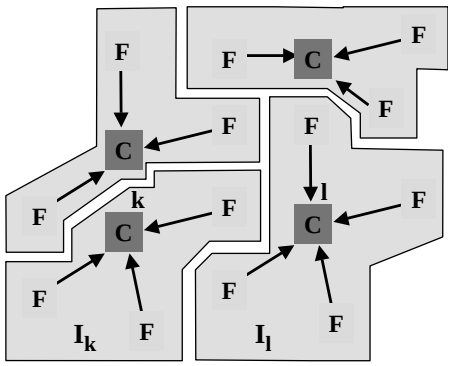


Figure 40: Subdivision of fine-level variables into aggregates. The arrows indicate which C-variable an F-variable interpolates from.

Obviously, regarding the coefficients a_{kl}^H, the particular role of the variables k and l (as being C-variables) is not distinguished from the other variables. In fact, 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 is nothing else than *piecewise constant interpolation* from these new coarse-level variables to the associated aggregates.

This leads to the so-called *aggregation-type* AMG approaches [73, 74, 10] which originally had 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 is defined to be piecewise constant. The above description just points out that the aggregation approach can be regarded as a limiting case of the approach considered in this paper (which started from the interpretation that the coarse-level variables form a subset of the fine-level ones; see Remark 2.2).

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. Since directionality of strength of connectivity plays no role in this approach, strength of connectivity is most naturally defined in a symmetric way. More precisely, one usually defines two variables i and j to be strongly connected to each other if $a_{ij}^2/a_{ii}a_{jj}$ exceeds a certain size.

Unfortunately, an immediate implementation of this simple coarsening and interpolation approach leads to rather inefficient AMG cycles, even if used as a pre-conditioner. Convergence will be very slow and not at all robust (this has already been pointed out in Section 4.2.1, see "Variant 4"). In particular, V-cycle convergence will exhibit a *strong h-dependency* if applied to differential problems. In fact, if regarded as a limiting case of the approach considered in this paper, the aggregation approach just *forces* worst-case situations as discussed in Section 6 (see Example 6.1). By the definition of the approach, remedies to avoid such worst-case situations in practice (by a proper distribution of the C- and F-variables) as discussed in Section 6, cannot be realised here. Finally, piecewise constant interpolation cannot account for any potential oscillatory behavior of the error and, thus, is not suitable if there are strong positive couplings.

Consequently, the basic idea of aggregation-based AMG needs certain improvements in order to become practical. In the following sections, we sketch two possibilities introduced in [10] and [73, 74], respectively. Since we just want to highlight the main ideas, we restrict our motivation to simple but characteristic (Poisson-like) problems.

At this point, for completeness, we also want to mention [21] where some concept of aggregation has been introduced for the first time.

9.1 Re-scaling of the Galerkin operator

In [10] it is demonstrated that the coarse-grid correction of smooth error, and by this the overall convergence, can often be substantially improved by using "over-interpolation", that is, by multiplying the actual correction (corresponding to piecewise constant interpolation) by some factor $\alpha > 1$. Equivalently, this means that the coarse-level Galerkin operator is re-scaled by $1/\alpha$,

$$I_h^H A_h I_H^h \quad \longrightarrow \quad \frac{1}{\alpha} I_h^H A_h I_H^h \ .$$

To motivate this approach, 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 us assume any error, e^h, to be given which satisfies the homogeneous boundary conditions. If no re-scaling is done ($\alpha = 1$), the variational principle (last statement in Corollary 2.1) tells us that the two-level correction, $I_H^h e^H$, is optimal in the sense that it minimizes $\|e^h - I_H^h e^H\|_1$ w.r.t. all possible corrections in

$\mathcal{R}(I_H^h)$. Because of (42) this means that $I_H^h e^H$ minimizes

$$\|v^h\|_1^2 = (A_h v^h, v^h)_E = \frac{1}{2h^2}\sum_{i,j}{}'(v_i^h - v_j^h)^2 + \sum_i s_i (v_i^h)^2 \tag{155}$$

where $v^h = e^h - I_H^h e^H$. (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 $s_i = 0$), 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.

The result of this minimization is illustrated in Figure 41 (see also [10, 12]), assuming the aggregates to be built by joining pairs of neighboring variables (marked by dashed boxes). 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.

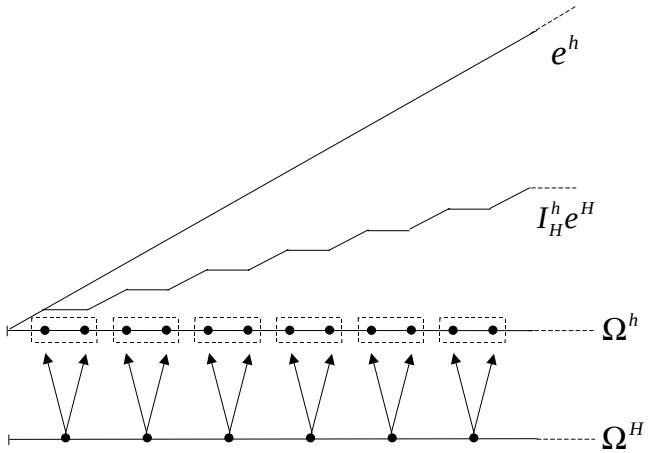


Figure 41: Optimal approximation $I_H^h e^H$ of e^h w.r.t. the energy norm

This simple argument illustrates that the optimal approximation of e^h by elements in $\mathcal{R}(I_H^h)$ w.r.t. the *energy norm* is quite bad in the sense of approximating the actual *values* of e^h. In fact, multiplying the resulting approximation by a factor of $\alpha = 2$ gives a much more effective correction in this sense. Note that subsequent smoothing smooths out the "wiggles", but does not improve the quality of the correction.

Remark 9.1 Note that the aggregation approach considered above for the model problem $-u''$ coincides with the approach considered in Example 6.1, where we have shown that the Galerkin operator is off by a factor of 2. In fact, we could have used this result immediately to motivate that a re-scaling of the Galerkin operator by $\alpha \approx 2$ makes sense. However, the above considerations show the *origin* of the problem more clearly, namely, the inability of piecewise constant interpolation to approximate the *values* of smooth error if approximation is based on the energy norm. Piecewise *linear* (second order) interpolation would not exhibit this problem (see the next section). $\ll$

The main argument carries 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 [10] that a slightly reduced value of $\alpha = 1.8$ (in order to avoid "overshooting") yields substantially improved V-cycle convergence for various types of problems, at least if the cycle is used as a preconditioner and if the number of coarser levels is kept fixed (in [10] four levels are always used). Smoothing is done by symmetric Gauss-Seidel relaxation sweeps.

Clearly, the robustness and efficiency of this (very simple and easy to program) approach are somewhat limited since a good value of α depends on various aspects such as the concrete problem, the type of mesh and, in particular, the 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 above show that the best weight would be $\alpha \approx 3$ in case of Poisson's equation. If the size of the aggregates varies over the domain, it becomes difficult to define a good value for α.

9.2 Smoothed aggregation

Another approach to accelerate aggregation-based AMG is developed and analyzed in [73, 74, 75]. Here, piecewise constant interpolation is only considered as a first-guess interpolation which is improved by some *smoothing process* ("smoothed aggregation") before the Galerkin operator is computed. In [73, 74], 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 $\widetilde{I}_H^h$. Then the final interpolation operator used is defined by

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

where $D_h = diag(A_h^f)$ 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 first approximation $\widetilde{I}_H^h e^H$. (Note that this process will increase the "radius" of interpolation and, hence, destroy the simplicity of the basic approach. Note also that Jacobi relaxation here serves a quite different purpose than Jacobi F-relaxation as considered in Section 5.1.3.)

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 [73, 74] in each spatial direction). Note first that, since all connections are strong, we have $A_h^f = A_h$. Figure 42 depicts both the piecewise constant interpolation (dashed line) and the smoothed interpolation obtained after the application of one Jacobi-step with $\omega = 2/3$ (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.

In order to see this explicitly, we use the notation as introduced in the figure and note first that the result of piecewise constant interpolation is

$$\widetilde{e}_{i,k-1} = e_{k-1}, \quad \widetilde{e}_{i,k} = e_k \quad \text{and} \quad \widetilde{e}_{i,k+1} = e_{k+1} \qquad (i = -1, 0, 1)\,. \tag{156}$$

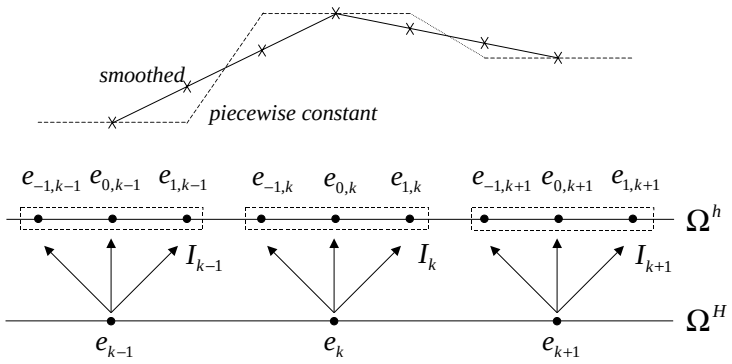


Figure 42: Piecewise constant versus smoothed interpolation

The application of one ω-Jacobi step as formally described above, using $\widetilde{e}$ as first approximation, means that the final interpolated values within the aggregate I_k (and analogously in all other aggregates) are computed as follows:

$$e_{-1,k} = \widetilde{e}_{-1,k} + \omega(\overline{e}_{-1,k} - \widetilde{e}_{-1,k}) \qquad \text{where} \qquad \overline{e}_{-1,k} = \frac{1}{2}(\widetilde{e}_{1,k-1} + \widetilde{e}_{0,k}) \ ,$$

$$e_{1,k} = \widetilde{e}_{1,k} + \omega(\overline{e}_{1,k} - \widetilde{e}_{1,k}) \qquad \text{where} \qquad \overline{e}_{1,k} = \frac{1}{2}(\widetilde{e}_{0,k} + \widetilde{e}_{-1,k+1}) \ ,$$

$$e_{0,k} = \widetilde{e}_{0,k} + \omega(\overline{e}_{0,k} - \widetilde{e}_{0,k}) \qquad \text{where} \qquad \overline{e}_{0,k} = \frac{1}{2}(\widetilde{e}_{-1,k} + \widetilde{e}_{1,k}) \ .$$

Inserting (156), this gives

$$e_{-1,k} = \frac{\omega}{2}e_{k-1} + (1 - \frac{\omega}{2})e_k \ , \quad e_{1,k} = (1 - \frac{\omega}{2})e_k + \frac{\omega}{2}e_{k+1} \quad \text{and} \quad e_{0,k} = e_k \ .$$

The special choice $\omega = 2/3$ indeed leads to linear interpolation as pointed out above,

$$e_{-1,k} = \frac{1}{3}e_{k-1} + \frac{2}{3}e_k \ , \quad e_{1,k} = \frac{2}{3}e_k + \frac{1}{3}e_{k+1} \quad \text{and} \quad e_{0,k} = e_k \ .$$

Remark 9.2 Linear interpolation does not exhibit a scaling problem as described in the previous section for piecewise constant interpolation. In fact, for the above model case, one easily computes the Galerkin operator to be

$$\frac{1}{(3h)^2}[-3 \quad 6 \quad -3]_{3h}$$

which, after proper scaling of the restriction operator by $1/3$, is seen to exactly correspond to the "natural" $3h$-discretization of $-u''$. $\ll$

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 $\omega = 2/3$ is kept fixed, interpolation will typically be much better than for the piecewise constant one. This is demonstrated in [74] by means of various 2D and 3D examples. (Smoothing is done by a mixture of Gauss-Seidel and

SOR sweeps.) 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. In [73], the tendency is to compose aggregates of three neighboring variables in each spatial direction. If, instead, only *two* neighbors would be aggregated in each spatial direction (as in the previous section), one easily sees by similar arguments as above that $\omega \approx 0.5$ should be chosen.

In the following, we compare the performance of RAMG05 with that of the aggregation-based AMG code distributed by Mandel/Vanek [73]. We want to stress that this is *not* meant to be any kind of judgement of the underlying two approaches in general; too much can still be improved in either approach. We just want to point out some differing behavior of the interpolations as currently used in these two codes.

	complexities		times (sec) / IBM PowerPC, 333 MHz				
			setup	stand-alone		conjugate gradient	
method	c_A	c_G	time	cycle	$\varepsilon_0 = 10^{-10}$	cycle	$\varepsilon_0 = 10^{-10}$
Example from Section 8.4.1, $h = 1/512$							
aggregat. AMG	1.56	1.31	10.7	2.04	245.4 (115)	2.48	57.90 (19)
VS(S)	2.52	1.79	4.86	1.74	27.49 (13)	2.11	21.71 (8)
FS(S)	2.52	1.79	4.86	3.30	21.42 (5)	3.64	19.52 (4)
VA2(S)	2.14	1.58	4.76	1.45	19.28 (10)	1.81	19.25 (8)
Example from Section 8.4.2 (one million cell case)							
aggregat. AMG	2.64	1.29	108.	17.0	4634. (266)	19.1	414.4 (16)
VS(S)	2.85	1.56	43.8	10.5	401.2 (34)	12.6	245.1 (16)
FS(S)	2.85	1.56	43.8	17.5	288.6 (14)	19.5	258.2 (11)
VA1(S)	1.41	1.13	26.7	5.56	476.4 (81)	7.67	210.7 (24)
Example from Section 8.5.1, $h = 1/512$, $\alpha = 20°$							
aggregat. AMG	1.22	1.13	11.2	2.10	1610. (762)	2.37	191.5 (76)
VS(S)	3.24	1.84	8.82	2.57	186.2 (69)	3.03	81.66 (24)
FS(S)	3.24	1.84	8.82	5.03	129.6 (24)	5.43	74.12 (12)
VA2(S)	1.72	1.41	6.09	1.54	103.3 (63)	1.99	49.86 (22)

Table 10: Complexities and computing times

In general, both codes perform comparably if applied to relatively smooth (Poisson-like) problems. Sometimes RAMG05 is slightly faster and sometimes the aggregation-based code. A major advantage of aggregation-type AMG is that it, typically, needs still less memory than RAMG05 (due to its very fast coarsening which causes a lower operator complexity c_A). On the other hand, the aggregation-based code seems to *require* acceleration by conjugate-gradient to maintain its efficiency and robustness in more complex situations. Since RAMG05 puts more effort into the construction of interpolation and performs a slower coarsening, its performance seems to depend on aspects such as strong discontinuities only to a lesser extent. This is demonstrated in Table 10 for three examples all of which exhibit strong anisotropies or discontinuites. The results clearly show that, at least for the kind of problems considered here, aggregation-based AMG behaves critically if used stand-alone (with total computing times being higher than those of RAMG05 by factors of 8.5 to 16). However, if used as a pre-conditioner, efficiency substantially increases.

10 Further developments and conclusions

The AMG approach described in this introductory paper has been seen to provide very robust and efficient methods for solving certain types of matrix equations such as those typically arising in the numerical solution of (scalar) elliptic PDEs. This has been demonstrated by a variety of applications of different type, on structured as well as unstructured grids, in 2D and 3D. Although all applications were geometrically based and many of them were even defined on very simple grids, AMG did not make use of any information other than that contained in the given matrix. The only reason for also considering certain model problems on simple geometries was that they most easily allow the investigation of AMG's asymptotic behavior as well as its dependency on various specific aspects such as anisotropies, discontinuities, singular perturbations and the like. AMG's performance in geometrically complex situations is very much comparable as demonstrated by some selected examples.

From a practical point of view, this is the main strength of AMG: its applicability to complex geometric situations, independently of the spatial dimension, 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 (as long as the underlying matrix satisfies certain conditions). That is, AMG provides an attractive multi-level variant whenever geometric multigrid is either too difficult to apply or cannot be used at all.

Clearly, AMG should not be regarded as a competitor for geometric multigrid: whenever geometric multigrid *can be* applied efficiently, it will usually be superior. Instead, AMG should be regarded as an efficient alternative to standard numerical methods such as conjugate gradient accelerated by typical (one-level) pre-conditioners: AMG not only converges much faster but its convergence is, to a large extent, also independent of the size of the given problem. Although it is designed to be used stand-alone, practical experience has clearly shown that one often can increase efficiency further by using AMG as a pre-conditioner. We have seen that cheaper (e.g. low-memory) AMG variants, used as a pre-conditioner, are often better than more sophisticated ones applied stand-alone.

Further developments and applications which are close to the original AMG ideas are, for example, contained in [20, 29, 30, 33, 37, 41, 42, 55, 83, 84]. Related approaches, but with a focus on different coarsening and interpolation strategies, are, for example, found in [28, 35]. AMG methods based on smoothed aggregation (see Section 9.2) are an efficient alternative to standard AMG, at least if employed as a pre-conditioner rather than stand-alone. Applications of the (non-smoothed) aggregation-type approach in computational fluid dynamics are found in [39, 54, 61, 80].

Many aspects have not been addressed in this introductory paper, for instance, the further improvement of interpolation. The expressed focus of this introduction was on purely matrix-based approaches. However, as long as interpolation is defined merely on the basis of the algebraic information contained in the given matrix, its "quality" (assuming an adequate geometric problem to be given) is somewhat limited. Although, for all type of problems considered here, this could be compensated for by a proper arrangement of the algorithm, in general this limitation is the major reason for the fact that the two-level theory presented does not carry over to a V-cycle theory (proving V-cycle convergence factors which are independent of the size of the given problem). Generally, the more

effort is put into the construction of interpolation, 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.

Moreover, 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., [10, 18, 40, 63, 74]), 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 symmetric positive definite, 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 [56]. 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 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 [63].

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, [18]) 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 eigenvalues. Instead of defining interpolation by directly exploiting (40), the goal in [18] is to define interpolation so that eigenvectors are interpolated the better the smaller the associated eigenvalue is. 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 [18] 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 [40, 79]. 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 [79] concentrates on scalar applications, an extension to systems of PDEs (from linear elasticity) is one major aspect in [40]. 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, see Remark 4.1.) It is interesting that the approach in [40] can be regarded as an extension of the earlier work [74] 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 [40] 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 [19] and the references given therein.

In the approach of [77], A is not assumed to be symmetric, 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, $\|.\|_E$. More precisely, the attempt is to make

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

where e^H denotes the straight injection of $S_h e^h$ to the coarse level, as small as possible (cf. (93)). In [77], 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 standard 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 [31].

An important aspect which has not been addressed in this paper is the parallelisation of AMG. An efficient parallelisation of AMG is rather complicated and requires certain algorithmical modifications in order to limit the communication cost without sacrificing convergence significantly. Most parallelisation approaches investigated up to now either

refer to simple aggregation-based variants (e.g. [61]) or use straightforward domain decomposition techniques (such as Schwarz' alternating method) for parallelisation. A parallelisation strategy which stays very close to the standard AMG approach is presented in [38]. 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 [77] is also available in parallel. There are several further ongoing parallelisation activities, for instance, at the University of Bonn and the National Laboratories LLNL [24] and LANL, but no results have been published by now.

It is beyond the scope of this introduction 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, 3, 4, 5, 25, 46, 47, 48, 57, 76]. Multi-level structures have also been introduced into ILU type pre-conditioners, for example, in [66]. Recently, some hybrid methods have been developed which use ideas both from ILU and from multigrid [6, 7, 8, 58, 59, 60]. For a further discussion, see also [78].

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] Alcouffe, R.E.; Brandt, A.; Dendy, J.E.; Painter, J.W.: *The multi-grid method for the diffusion equation with strongly discontinuous coefficients*, SIAM J. Sci. Stat. Comput. 2, pp 430-454, 1981.

[2] Axelsson, O.; Vassilevski, P.S.: *Algebraic multilevel preconditioning methods I*, Num. Math. 56, pp 157-177, 1989.

[3] Axelsson, O.; Vassilevski, P.S.: *Algebraic multilevel preconditioning methods II*, SIAM Numer. Anal. 27, pp 1569-1590, 1990.

[4] Axelsson, O.: *The method of diagonal compensation of reduced matrix entries and multilevel iteration*, J. Computat. Appl. Math. 38, pp 31-43, 1991.

[5] Axelsson, O.; Neytcheva, M.: *Algebraic Multilevel Iteration Method for Stieltjes Matrices*, Numerical Linear Algebra with Applications, Vol 1(3), pp 213-236, 1994.

[6] Bank, R.E.; Smith, R.K.: *The incomplete factorization multigraph algorithm*, SIAM J. Sci. Comput., to appear.

[7] Bank, R.E.; Smith, R.K.: *The hierarchical basis multigraph algorithm*, SIAM J. Sci. Comput., submitted.

[8] Bank, R.E.; Wagner, Ch.: *Multilevel ILU Decomposition*, Numer. Math., to appear.

[9] Batycky, R.; Blunt, M.; Thiele, M.: *A 3D Field-Scale Streamline-Based Reservoir Simulator*, SPE Reservoir Engineering, November 1997 issue.

[10] Braess, D.: *Towards Algebraic Multigrid for Elliptic Problems of Second Order*, Computing 55, pp 379-393, 1995.

[11] Brandt, A.: *Multigrid Techniques: 1984 Guide with Applications to Fluid Dynamics*, Arbeitspapiere der GMD 85, 1984.

[12] Brandt, A.: *Algebraic multigrid theory: the symmetric case*, Appl. Math. Comp. 19, pp 23-56, 1986.

[13] Brandt, A: *General highly accurate algebraic coarsening schemes*, Proceedings of the Ninth Copper Mountain Conference on Multigrid Methods, Copper Mountain, April 11-16, 1999.

[14] Brandt, A.; McCormick, S.F.; Ruge, J.: *Algebraic multigrid (AMG) for automatic multigrid solution with application to geodetic computations*, Institute for Computational Studies, POB 1852, Fort Collins, Colorado, 1982.

[15] Brandt, A.; McCormick, S.F.; Ruge, J.: *Algebraic multigrid (AMG) for sparse matrix equations*, in "Sparsity and its Applications", D.J. Evans (ed.), Cambridge University Press, pp 257-284, Cambridge, 1984.

[16] Brandt, A.; Livshits, I.: *Wave-ray multigrid method for the standing wave equation*, Electr. Trans. on Num. Analysis 6, 162-181, 1997.

[17] Brandt, A.; Mikulinsky, V.: *On recombining iterants in multigrid algorithms and problems with small islands*, SIAM J. Sci. Comput. 16, pp 20-28, 1995.

[18] Brezina, M.; Cleary, A.J.; Falgout, R.D.; Henson, V.E.; Jones, J.E.; Manteuffel, T.A.; McCormick, S.F.; Ruge, J.W.: *Algebraic Multigrid Based on Element Interpolation (AMGe)*, LLNL technical teport UCRL-JC-131752, submitted to SIAM Journal on Scientific Computing.

[19] Chan, T.; Zikatanov, L.; Xu, J.: *An agglomeration multigrid method for unstructured grids*, Proceedings of the 10-th International Conference on Domain Decomposition Methods, 1998.

[20] Chang, Q.; Wong, Y.S.; Fu, H.: *On the algebraic multigrid method*, J. Comp. Phys. 125, 279-292, 1996.

[21] Chatelin, F.; Miranker, W.L.: *Acceleration by aggregation of successive approximation methods*, LAA 43, 17-47, 1982.

[22] Chen, L.; Armfield, S.: *A simplified marker and cell method for unsteady flows on non-staggered grids*, Int. J. Num. Meth. Fluids, 21, pp 15-34, 1995.

[23] Cleary, A.J.; Falgout, R.D.; Henson, V.E.; Jones, J.E.; Manteuffel, T.A.; McCormick, S.F.; Miranda, G.N.; Ruge, J.W.: *Robustness and scalability of algebraic multigrid*, SIAM Journal on Scientific Computing, special issue on the "Fifth Copper Mountain Conference on Iterative Methods", 1998.

[24] Cleary, A.J.; Falgout, R.D.; Henson, V.E.; Jones, J.E.: *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 1457, Springer-Verlag, New York, pp. 104-115, 1998.

[25] Dahmen, W.; Elsner, L.: *Algebraic multigrid methods and the Schur complement*, Notes on Numerical Fluid Mechanics 23, Vieweg Verlag, Braunschweig, 1988.

[26] Dendy, J.E.; McCormick, S.F.; Ruge, J.; Russell, T.; Schaffer, S.: *Multigrid methods for the three-dimensional petroleum reservoir simulation*, Procs. 10th SPE Symposium on Reservoir Simulation, Feb 6-8, 1989.

[27] Dendy, J.E. (Jr.): *Black box multigrid*, J. Comp. Physics 48, pp. 366-386, 1982.

[28] Fuhrmann, J.: *A modular algebraic multilevel method*, Tech. Report Preprint 203, Weierstrass-Institut für Angewandte Analysis und Stochastik, Berlin, 1995.

[29] Grauschopf, T.; Griebel, M.; Regler, H.: *Additive multilevel-preconditioners based on bilinear interpolation, matrix dependent geometric coarsening and algebraic multigrid coarsening for second order elliptic PDEs*, Appl. Numer. Math. 23, pp 63-96, 1997.

[30] Griebel, M.; Neunhoeffer, T.; Regler, H.: *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.

[31] Guillard, H.; Vanek, P.: *An aggregation multigrid solver for convection-diffusion problems on unstructured meshes*, Center for Computational Mathematics, University of Denver, Report 130, 1998.

[32] Hemker, P.W.: *A note on defect correction processes with an approximate inverse of deficient rank*, J. Comp. Appl. Math. 8, pp 137-139, 1982.

[33] Huang, W.Z.: *Convergence of algebraic multigrid methods for symmetric positive definite matrices with weak diagonal dominance*, Appl. Math. Comp. 46, pp 145-164, 1991.

[34] Kettler, R.: *Analysis and comparison of relaxation schemes in robust multigrid and preconditioned conjugate gradient methods*, Lecture Notes in Mathematics 960, pp 1-176, Springer, 1982.

[35] Kickinger, F.: *Algebraic multi-grid for discrete elliptic second order problems*, Institutsbericht 513, Universität Linz, Institut für Mathematik, 1997.

[36] Kim, Y.; Chung, T.: *Finite-element analysis of turbulent diffusion flames*, AIAA Journal 27, pp 330-339, 1988.

[37] Krechel, A.; Stüben, K.: *Operator dependent interpolation in algebraic multigrid*, Proceedings of the Fifth European Multigrid Conference, Stuttgart, Oct. 1-4, 1996; Lecture Notes in Computational Science and Engineering 3, Springer, 1998.

[38] Krechel, A.; Stüben, K.: *Parallel algebraic multigrid based on subdomain blocking*, GMD-Report 71, 1999. Submitted to Parallel Computing.

[39] Lonsdale, R.D.: *An algebraic multigrid solver for the Navier-Stokes equations on unstructured meshes*, Int. J. Num. Meth. Heat Fluid Flow 3, 3-14, 1993.

[40] Mandel, J.; Brezina, M.; Vanek, P.: *Energy optimization of algebraic multigrid bases*, UCD/CCM Report 125, 1998.

[41] McCormick, S.; Ruge, J.: *Algebraic multigrid methods applied to problems in computational structural mechanics*, in: "State-of-the-Art Surveys on Computational Mechanics", pp 237-270, ASME, New York, 1989.

[42] Mertens, R.; De Gersem, H.; Belmans, R.; Hameyer, K.; Lahaye, D.; Vandewalle, S.; Roose, D.: *An Algebraic Multigrid Method for Solving Very Large Electromagnetic Systems*, to appear in: IEEE Trans. Magn. 34, 1998.

[43] Mulder, W.A.: *A new multigrid approach to convection problems*, J. Comp. Phys. 83, 303-323, 1989.

[44] Naik, N.H.; van Rosendale, J.: *The improved robustness of multigrid elliptic solvers based on multiple semicoarsened grids*, SIAM Num. Anal. 30, 215-229, 1993.

[45] Nonino, C.; del Guidice, S.: *An improved procedure for finite-element methods in laminar and turbulent flow*, in: "Numerical Methods in Laminar and Turbulent Flow, Part I" (Taylor, C., ed.), Pineridge Press, pp 597-608, 1985.

[46] Notay, Y.: *Optimal V-cycle algebraic multilevel preconditioning*, Num. Lin. Alg. Appl., to appear.

[47] Notay, Y.: *Using approximate inverses in algebraic multilevel methods*, Numer. Math. 80, pp 397-417, 1998.

[48] Notay, Y.: *An efficient algebraic multilevel preconditioner robust with respect to anisotropies*, in: "Algebraic Multilevel Iteration Methods with Applications" (Axelsson, O.; Polman, B. eds.), Department of Mathematics, University of Nijmegen, pp 111-228, 1996.

[49] Oosterlee, C.W.; Washio, T.: *An evaluation of parallel multigrid as a solver and a preconditioner for singularly perturbed problems*, SIAM J. Sci. Comput. 19, pp 87-110, 1998.

[50] Oosterlee, C.W.; Washio, T.: *Krylov subspace acceleration of nonlinear multigrid with application to recirculating flow*, to appear in: SIAM J. Sci. Comput., 1999.

[51] Patankar, S.; Spalding, D.: *A calculation procedure for heat, mass and momentum transfer in three-dimensional parabolic flows*, Int. J. Heat Mass Transfer, 15, pp 1787-1806, 1972.

[52] Patankar, S.: *Numerical Heat Transfer and Fluid Flow*, McGraw-Hill, 1980.

[53] Patankar, S.: *A calculation procedure for two-dimensional elliptic situations*, Num. Heat Transfer, 4, pp 409-425, 1981.

[54] Raw, M.: *A coupled algebraic multigrid method for the 3D Navier-Stokes equations*, report: Advanced Scientific Computing Ltd., 554 Parkside Drive, Waterloo, Ontario N2L 5Z4, Canada.

[55] Regler, H.: *Anwendungen von AMG auf das Plazierungsproblem beim Layoutentwurf und auf die numerische Simulation von Strömungen*, PhD thesis, TU München, 1997.

[56] Reitzinger, S.: *Algebraic Multigrid and Element Preconditioning I*, SFB-Report 98-15, University Linz, Austria, Dec 1998.

[57] Reusken, A.A.: *Multigrid with matrix-dependent transfer operators for a singular perturbation problem*, Computing 50 (3), pp. 199-211, 1993.

[58] Reusken, A.A.: *A multigrid method based on incomplete Gaussian elimination*, Eindhoven University of Technology, Report RANA 95-13, ISSN 0926-4507, 1995.

[59] Reusken, A.A.: *Approximate Cyclic Reduction Preconditioning*, in: Multigrid Methods 5, (Hackbusch, W.; Wittum, G.; eds.), Lecture Notes in Computational Science and Engineering, Vol 3, pp 243-259, Springer Berlin, 1998.

[60] Reusken, A.A.: *On the Approximate Cyclic Reduction Preconditioner*, Report 144, Institut für Geometrie und Praktische Mathematik, RWTH Aachen, 1997. (To appear in SIAM J. Sci. Comput.)

[61] Robinson, G.: *Parallel computational fluid dynamics on unstructured meshes using algebraic multigrid*, Parallel Computational Fluid Dynamics 92 (Pelz, R.B.; Ecer, A.; Häuser, J. eds.), Elsevier Science Publishers B.V., 1993.

[62] Ruge, J.W.; Stüben, K.: *Efficient solution of finite difference and finite element equations by algebraic multigrid (AMG)*, Multigrid Methods for Integral and Differential Equations (Paddon, D.J.; Holstein H.; eds.), The Institute of Mathematics and its Applications Conference Series, New Series Number 3, pp. 169-212, Clarenden Press, Oxford, 1985.

[63] Ruge, J.W.; Stüben, K.: *Algebraic Multigrid (AMG)*, In "Multigrid Methods" (McCormick, S.F., ed.), SIAM, Frontiers in Applied Mathematics, Vol 5, Philadelphia, 1986.

[64] Saad, Y.: *Iterative Methods for Sparse Linear Systems*, PWS Publishing, Boston, USA, 1996.

[65] Saad, Y.; Schultz, M.H.: *GMRES: A generalized minimal residual algorithm for solving nonsymmetric linear systems*, SIAM J. Sci. Comput. 7, pp 856-869, 1986.

[66] Saad, Y.: *ILUM: a multi-elimination ILU preconditioner for general sparse matrices*, SIAM J. Sci. Comput. 17, pp 830-847, 1996.

[67] Schröder, J.: *Operator inequalities*, Mathematics in Science and Engineering, Vol. 147, Academic Press, 1980.

[68] Schröder, J.; Trottenberg, U.: *Reduktionsverfahren für Differenzengleichungen bei Randwertaufgaben I*, Numer. Math. 22, pp. 37-68, 1973.

[69] Stüben, K.: *Algebraic multigrid (AMG): Experiences and comparisons*, Appl. Math. Comp. 13, pp. 419-452, 1983.

[70] Stüben, K.: *A Review of Algebraic Multigrid*, GMD-Report 69, 1999. To appear in Journal of Computational and Applied Mathematics, 2000.

[71] Thiele, M.; Batycky, R.; Blunt, M.: *A Streamline-Based 3D Field-Scale Compositional Reservoir Simulator*, paper SPE 38889 presented at the 1997 SPE Annual Technical Conference and Exhibition, San Antonio, Texas, 5-8 October.

[72] Van der Vorst, H.: *BICGSTAB: A fast and smoothly converging variant of Bi-CG for the solution of non-symmetric linear systems*, SIAM J. Sci. Comput. 13, pp 631-644, 1992.

[73] Vanek, P.; Mandel, J.; Brezina, M.: *Algebraic multigrid on unstructured meshes*, University of Colorado at Denver, UCD/CCM Report No 34, 1994.

[74] Vanek, P.; Mandel, J.; Brezina, M.: *Algebraic multigrid by smoothed aggregation for second and fourth order elliptic problems*, Computing 56, pp 179-196, 1996.

[75] Vanek, P.; Brezina, M.; Mandel, J.: *Convergence of algebraic multigrid based on smoothed aggregation*, UCD/CCM Report 126, 1998. Submitted to Numer. Math.

[76] Wagner, C.; Kinzelbach, W.; Wittum, G.: *Schur-complement multigrid - a robust method for groundwater flow and transport problems*, Numer. Math. 75, pp 523-545, 1997.

[77] Wagner, C.: *On the algebraic construction of multilevel transfer operators*, IWR-Report, Universität Heidelberg, submitted to Computing, 1999.

[78] Wagner, C.: *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.

[79] Wan, W.L.; Chan, T.F.; Smith, B.: *An energy minimization interpolation for robust multigrid methods*, Department of Mathematics, UCLA, UCLA CAM Report 98-6, 1998.

[80] Webster, R.: *An algebraic multigrid solver for Navier-Stokes problems in the discrete second order approximation*, Int. J. Num. Meth. in Fluids 22, 1103-1123, 1996.

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

[82] Washio, T.; Oosterlee, C.W.: *Flexible multiple semicoarsening for 3D singularly perturbed problems*, SIAM J. Sci. Comput. 19, pp 1646-1666, 1998.

[83] Zaslavsky, L.: *An adaptive algebraic multigrid for multigroup neutron diffusion reactor core calculations*, Appl. Math. Comp. 53, pp 13-26, 1993.

[84] Zaslavsky, L.: *An adaptive algebraic multigrid for reactor critically calculations*, SIAM J. Sci. Comput. 16, pp 840-847, 1995.

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