9th International Forum on Reservoir Simulation

December 9-13, 2007, Abu Dhabi, United Arab Emirates

Solving Reservoir Simulation Equations

Klaus Stüben

Fraunhofer Institute SCAI
Sankt Augustin, Germany

Reservoir simulation is a necessary tool for making decisions. It is through numerical simulation that one can obtain knowledge pertaining to the processes occurring in the interior of an oil reservoir and hence, to enable an analysis of the various recovery strategies in order to guarantee optimal exploitation. History matching, optimization, and what-if scenarios require the simulation of large numbers of complex field-scale models.

Deep inside any reservoir simulator, large linear systems of equations need to be solved numerically over and over again. During the last years, reservoir models have been growing in complexity (regarding both the geometry and the physical model), heterogeneity, and size, causing these linear systems to get increasingly large and difficult to solve by classical numerical solvers. In fact, the computational time required to solve these systems of equations is today's major bottleneck in the practicability of numerical simulation. Hence, advanced reservoir simulators necessarily require particularly efficient linear solver modules as well as parallel computers to yield answers in an economical time frame.

The so-called *algebraic* multigrid (AMG) technique - the algebraic analog of the *geometric* multigrid (GMG) technique - provides a state-of-the-art means to construct numerical solvers suitable to tackle large applications with highest efficiency, usually much faster than any classical solver. As a consequence, AMG-based approaches are becoming increasingly popular as numerical kernel in many industrial simulation codes. AMG's major advantages - numerical efficiency, robustness, scalability and ease-of-use - have become the driving forces behind its growing success in industrial use. A state-of-the-art software package, SAMG, developed at the Fraunhofer Institute SCAI, is currently being used in many industrial areas. In particular, in the oil industry, SAMG has become a well-established tool for various software providers as well as major oil companies.

This paper gives a short introduction to the multigrid technique in general, focusing on the key aspects that make it scalable. We will then outline the idea of AMG, how it relates to GMG, and how it can be employed to dramatically reduce computational times in reservoir simulation. For demonstration we will present various examples.

Contents

1 Introduction

The use and the impact of numerical simulation – e.g. for virtual product development, for the understanding of product properties or the understanding and optimization of environmental processes - are continuously growing. For the exploitation of oil fields, it is only through numerical simulation that knowledge pertaining to the processes occurring in the interior of an oil reservoir can be obtained and that an analysis of the various recovery strategies in order to guarantee optimal exploitation can be made. A reduction of product cycle times, for example in the automotive industry, is not possible without increasing the use of numerical simulation.

It is still the creativity and experience of the engineer which ultimately determines the quality of a product or the optimality of a process. However, he will be able to enhance his abilities to the degree which

- simulation software is more strongly integrated into the engineering process;
- the precision of the model involved is increased;
- simulations can be carried out interactively.

Here, however, numerical simulation often reaches limits in that standard solver technology is insufficient to account for the increasing complexity arising in industrial simulation. In reservoir simulation, for instance, a primary challenge for a new generation of reservoir simulators is the accurate description of multiphase flow in highly heterogeneous media and very complex geometries. The implementation of robust and efficient solvers for such simulations is one of the main challenges that most simulator developers currently face in the oil industry. Simulation run times are often much too long to be practicable.

The general situation is the same in many other fields of numerical simulation such as in computational fluid dynamics, structural mechanics, oil reservoir and ground water simulation, casting and molding, process and device simulation in solid state physics, electro-chemistry and many others. All of these areas are characterized by (usually time-dependent) partial differential equations (PDEs).

For such problems to be treatable by a computer, they must first be *discretized.* That is, the continuum is replaced by a *grid* (or a *mesh*) and each continuum function is replaced by a *grid (*or *mesh) function,* represented by a set of discrete variables, each attached to a certain point of the grid. After having discretized the PDEs by means of finite differences or finite elements one finally obtains systems of linear or nonlinear algebraic equations. Hence, the core of the computation at each time (and may be linearization) step is governed by the successive solution of linear systems of equations, formally denoted as

$$Au = f \; . \tag{1}$$

We will use the letter N to denote the total number of variables. Solving (1) may become very difficult and extremely time consuming for several reasons:

1. First, it is the sheer size of the linear systems (1) which causes an enormous computational complexity. In fact, to properly approximate the continuum and resolve the interesting phenomena, the discretization grid must be sufficiently fine. The number N of discrete variables thus tends to be huge. The number of algebraic

relations between the variables must, of course, be equally large. Today's large-scale simulations operate on grids of the order of up to many millions of grid points. For instance, in the automotive industry, flow simulations around complete cars bodies are done on grids involving 10 to 100 million points (cf. Figure 1 and Figure 2).

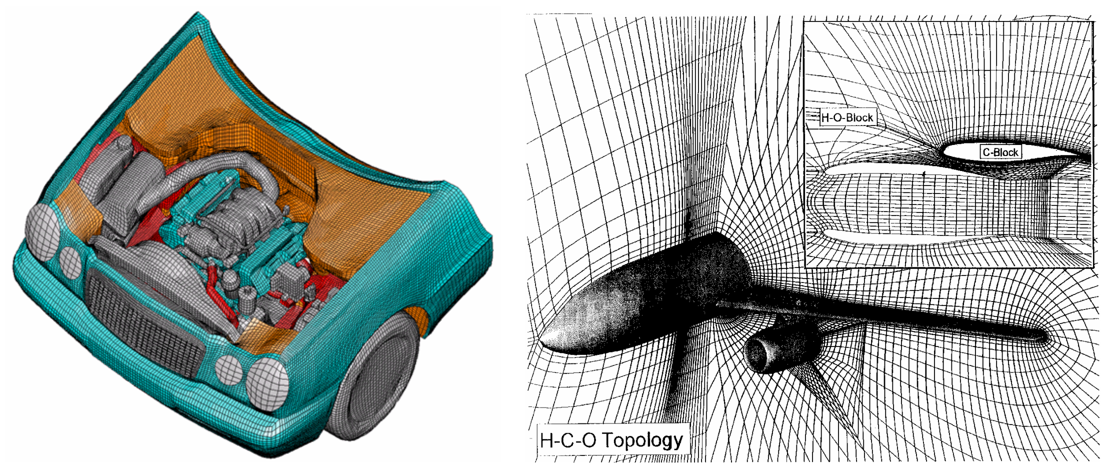


Figure 1: Grids for computing the underhood flow (left; courtesy of Computational Dynamics and Daimler-Benz) and the flow around an aircraft (right).

2. The problem to design *efficient* solvers becomes particularly difficult due to the fact that we are not dealing with *arbitrary* matrices A. One of the characteristics of PDEs is the *locality* of the basic physical laws. When discretized, the resulting algebraic systems will be local too, that is, each algebraic relation will involve only a small number of neighboring variables. As a consequence, the matrices A will be very sparsely populated, typically only $O(N)$ matrix entries are nonzero. At best, of course, the computational time to solve such linear systems is also just $O(N)$. Unfortunately, standard solvers have a much higher numerical complexity than that (see below), the reason being that they do not sufficiently exploit the origin and nature of the discrete linear systems.

3. Special characteristics may cause a given problem to be particularly difficult to solve. For instance, extremely unstructured or highly locally refined grids, not optimally shaped elements, extreme parameters contrasts (see Figures 2,4,14,19). In reservoir engineering, generally, the systems are highly nonsymmetric and indefinite. Moreover, the condition number and degree of coupling may be subject to dramatic changes due to abrupt flow variations induced by the high-heterogeneity and complex well operations during the simulation process.

4. A final source of increased complexity is the need to repeat the calculations many times over. The solution of linear systems of equations is routinely repeated over and over, for a variety of reasons – to identify "characteristic parameters" on which the system depends, to follow the time evolution of a problem, to resolve a non-linearity, to solve an inverse problem, to follow a bifurcation diagram, etc. History matching, optimization, and what-if scenarios require the simulation of large numbers of complex field-scale models.

In principle, many numerical solvers are available to solve linear systems of equations. Highest robustness is ensured by *direct* solvers (based on Gaussian elimination in one way or the other). Unfortunately, except for relatively small N, the application of direct solvers to sparse matrices is not feasible. Both memory requirement and computational complexity grow dramatically with increasing N. Typically, the computational time grows like $O(N^2)$ or even faster. In contrast to direct solvers, the memory requirement of classical *iterative* solvers is much lower, usually only $O(N)$. From this point of view, iterative methods are often the only choice in practice[1]. The computational work of classical iterative solvers typically grows like $W(N)=O(N^{\alpha})$ with some $\alpha>1$ depending on the concrete numerical method. Unfortunately, any method with $\alpha>1$ is not practicable if only the problem becomes large enough (cf. Figure 3). Instead, robust numerical solvers are requested for which the numerical work grows only linearly with the number of variables, $W(N)=O(N)$. Such methods are called "optimal" or "scalable".

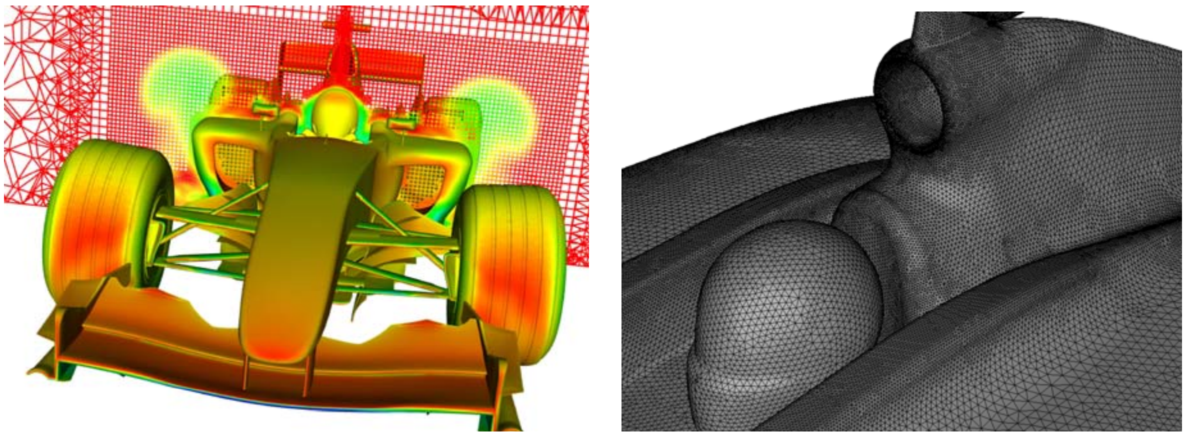


Figure 2: (left) Cross section of a highly complex mesh for the simulation of the exterior flow around a racing car. (right) Corresponding surface mesh near the driver. (Courtesy of Fluent and Sauber-Petronas)

In order to construct scalable solvers, one needs to exploit the physical background of the problems, namely, their spatial origin. Any problem with that origin can be discretized and treated not only on one grid, but on a hierarchy of increasingly fine grids. Based on such a hierarchy of grids (or "levels") it is possible to design numerical procedures that greatly benefit from intergrid (interscale) interactions. Roughly speaking, each level of discretization within the hierarchy is supposed to contribute only those "components" to the wanted solution which really require the grain size of that level. Everything else is computed on coarser levels. It is common opinion that, in order to really ensure scalability of a solver, such kind of hierarchical approach is *necessary*.

A breakthrough, and certainly one of the most important advances in the development of numerical solvers, was due to the invention of the geometric multigrid (GMG) principle (see [50] and the references given therein). This principle was the first to allow the construction of truly scalable numerical methods for solving elliptic PDEs. Unfortunately, the impact of GMG on industrial software development was disappointingly limited. One reason for this is certainly that GMG-based solvers cannot simply be integrated into an existing simulator. In fact, GMG is not just a solver but it rather provides a technology which requires the complete simulation environment to be "multigridded". The development of GMG-based simulators "from scratch", however, is a very demanding task and, under certain industrial requirements, may technically even be too complicated, if possible at all.

[1] Very often incomplete LU factorization methods – for instance, ILUT or ILU(k) – are combined with standard accelerators such as conjugate gradient, BiCGstab, GMRes or ORTHOMIN (see [37,38,39,40,52,53]).

Algebraic multigrid (AMG) solvers [36,43,18] attempt to combine the advantage of GMG, namely, its efficiency and scalability, with those of easy-to-use "plug-in solvers" as commonly used in most simulators. In contrast to GMG, which explicitly requires and exploits grid structures, AMG operates directly on the linear system of equations (1). To ensure scalability, AMG also requires a hierarchy (of linear systems of equations rather than grids) following the same basic principles as GMG. In contrast to GMG, however, a reasonable hierarchy is not only constructed fully automatically, it is also directly based on the entries in the discretization matrix *A* without exploiting any geometric information. Hence, AMG-based solvers are easy to integrate into existing simulators and they are independent of the spatial dimension as well as the structure of the underlying mesh – major reasons for their industrial success. Therefore, the interest in incorporating AMG methods as basic linear solvers in industrial simulation codes - in particular, in reservoir simulation - has been steadily increasing.

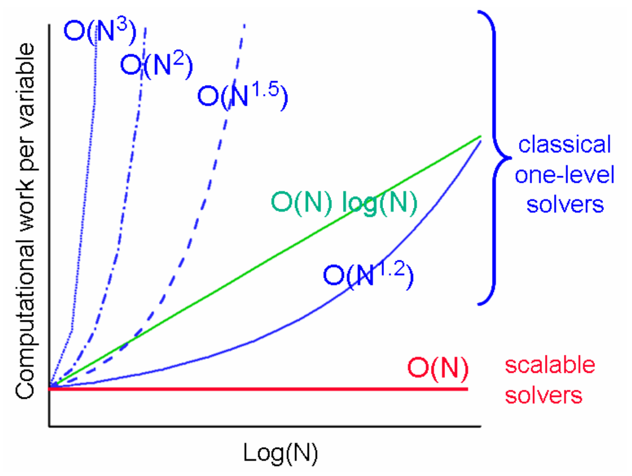


Figure 3: Computational work (scaled by the number of variables) for different types of solvers.

Moreover, AMG-based solvers have also efficiently been parallelized. This is important since parallel computers provide an important means to improve performance further, enhancing the possibilities to perform much larger simulations. One might even be tempted to believe that, because of the rapid improvement of parallel hardware, the numerical scalability of solvers is not really that important any more. However, the opposite is true: the larger the computer (in particular, its memory), the larger the problems to be simulated and the larger the gain through numerical scalability will be! Remember that the gain through numerical scalability is not simply a factor (as in the case of hardware), the gain increases with increasing *N* (cf. Figure 3)!

Hence, from a practical point of view, it is most important to combine numerical scalability with the scalability of parallel hardware. Only then can one achieve real breakthroughs in terms of overall performance. In this paper, we will not discuss the parallelization aspects further. However, we point out that a parallelization of AMG is not straightforward, and a lot of research has been done in this respect. The program package employed for all numerical computations in this paper, SAMG (see further below), is also available in parallel [30].

In this paper we discuss possibilities how to efficiently solve linear systems (1) obtained by discretizing (generally elliptic) PDEs, with special focus on reservoir engineering applications. The paper is organized as follows. Section 2 outlines the main idea of GMG, highlighting the key aspects that make it scalable. We focus on aspects which are relevant for the development of its algebraic analog, AMG. Section 3 presents the idea of "classical" AMG originally designed for solving *scalar* elliptic PDEs, outlining how it relates to GMG. Although ideas how to extend "classical" AMG to *coupled systems* of PDEs have already been considered in the early literature [36], a systematic application of AMG to industrial PDE systems - such as those arising in reservoir simulation - has only been done in recent years [18,19,46]. Possible ways of how to extend AMG to solve coupled systems of PDEs are summarized in Section 4.

Section 5 describes how AMG is currently employed in reservoir engineering. We will present examples of reservoir simulations based on different AMG approaches. All results presented in this paper have been obtained by the AMG package SAMG ("Systems AMG" [47]), developed at Fraunhofer Institute SCAI. For many years, SAMG has systematically been developed and extended with industrial applications in mind.

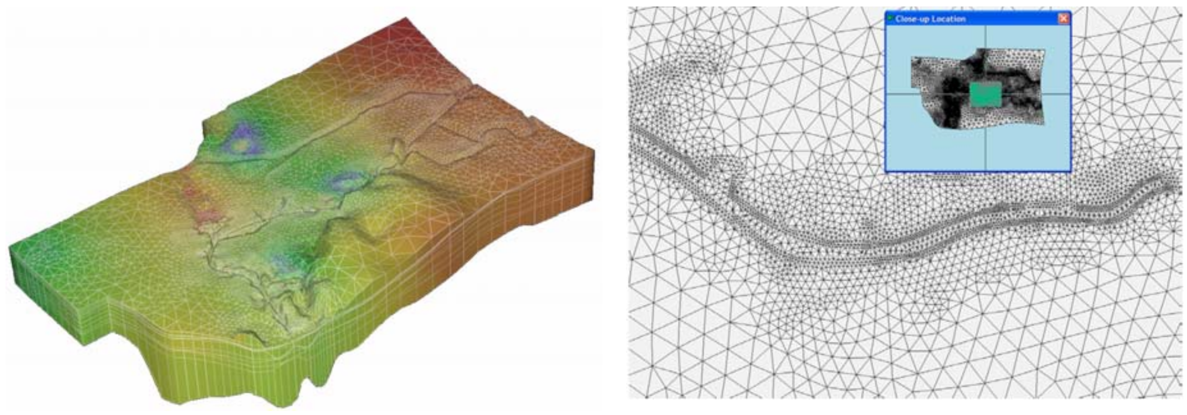


Figure 4: Large-scale 3D flow model with unstructured mesh. This can be considered as a typical transient finite-element groundwater model used in practical water resources simulation. (Courtesy of Wasy GmbH)

Acknowledgements

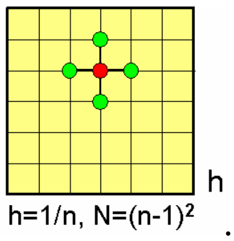
We point out that significant parts of this work relies on [46] which was the result of a close cooperation with Mary F. Wheeler and her group. We are particularly thankful to Hector Klie. The results on adaptive implicit modeling have been derived by my colleague Tanja Clees [19] in close cooperation with Leonard Ganzer, SMT Alps[2]. We thank the Austin group headed by Mary F. Wheeler, the Stanford group headed by Hamdi Tchelepi as well as Martin Blunt and Stephan Matthai from the Imperial College for their support of our work. Our very particular thanks are directed to Marco Thiele from StreamSim Technologies for a long-lasting cooperation, his never ending help and personal support.

2 Geometric Multigrid (GMG)

For ease of motivation let us consider Poisson's equation $-\Delta u = f$, defined on the unit square $\Omega = [0,1]^2$ with Dirichlet boundary conditions, discretized by the standard 5-point stencil on a grid Ω_h with mesh size *h*,

[2] now University of Leoben, Austria.

(2)
$$-\Delta_h u_h(x,y) = f_h(x,y)$$
$$-\Delta_h = \frac{1}{h^2}\begin{bmatrix} & -1 & \\ -1 & 4 & -1 \\ & -1 & \end{bmatrix}_h$$



.

The resulting linear system of *N* variables will be denoted as

(3) $$A_h u_h = f_h \,.$$

Some other model problems will be discussed in Section 2.5. Here and in the following we will usually omit the index *h*. Only if we explicitly need to distinguish different levels of discretizations will we use grid indices for clarity.

2.1 Relaxation as a solver

Thinking of u as a grid function, $u_{i,j}=u(x_i,y_j)=u(ih,jh)$, rather than a vector, the rows of (3) (except near boundaries) read as

$$4u_{i,j} - u_{i-1,j} - u_{i+1,j} - u_{i,j-1} - u_{i,j+1} = h^2 f_{i,j} \,.$$

A classical numerical process to solve such systems consists of satisfying one equation at a time by changing the associated variable, that is,

(4) $$u_{i,j} \leftarrow (u_{i-1,j} + u_{i+1,j} + u_{i,j-1} + u_{i,j+1} + h^2 f_{i,j})/4 \,.$$

Passing with that kind of local processing in some order over the entire grid - corresponding to one iteration step - is called a Gauss-Seidel (GS) *relaxation sweep*.

The original purpose of iterating such sweeps is to drive the system towards a solution. Unfortunately, the convergence will be very slow which is a direct consequence of the locality of relaxation and the ellipticity of the mathematical model: While the ellipticity implies that the solution at any grid point is determined by the boundary values *all along* the boundary[3], the local processing means that information "travels" very slowly across the grid. In fact, in each iteration step, information at any grid point is essentially transported only to its immediate neighbors whereas the influence on its further away neighbors decays exponentially with distance. That is, it takes a large number of iterations for information to travel from one side of the grid to the opposite side, say. Obviously, the finer the grid, the more iterations are needed.

This heuristically explains not only why a local process such as GS relaxation converges extremely slowly, it also explains why its convergence gets increasingly slow with decreasing mesh size. To be more specific, one can prove that *O*(*N*) relaxation sweeps are needed to reduce the error by a fixed quantity. Since the numerical cost of a single sweep is also *O*(*N*),

[3] This is in contrast to hyperbolic problems for which the transport of information follows certain characteristic directions.

the total work to reduce the error by a fixed quantity is $O(N^2)$. That is, while the memory requirement is minimal, the total computational time is comparable to that of a direct solver! Hence, GS relaxation in its classical form is of virtually no use for practice (cf. Table 1).

2.2 Relaxation as a smoother

In order to analyze GS relaxation a bit further, we consider the characteristic behavior of the *residual* and the *error*,

(5) $$r = f - Au \quad \text{and} \quad e = u^* - u\,,$$

respectively, in some more detail. Here u^* denotes the exact solution of $Au=f$, and u its current approximation.

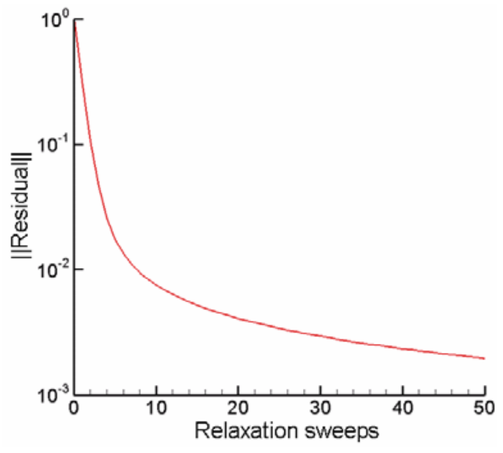


Figure 5: Typical convergence history of GS relaxation to solve (2)

Figure 5 illustrates the typical convergence history in terms of the residual, measured in some norm. While the residual drops rapidly during the very first relaxation steps, convergence drastically slows down afterwards and gets increasingly slow with increasing number of iterations. In order to understand the reason for this behavior, we need to have a closer look at the corresponding behavior of the error. By subtracting (4) from the fixed point equation,

$$u^*_{i,j} \equiv (u^*_{i-1,j} + u^*_{i+1,j} + u^*_{i,j-1} + u^*_{i,j+1} + h^2 f_{i,j})/4\,,$$

we immediately see that the error e changes according to

(6) $$e_{i,j} \leftarrow (e_{i-1,j} + e_{i+1,j} + e_{i,j-1} + e_{i,j+1})/4\,.$$

Obviously, this is just an averaging process for the error, causing the error very quickly to become smooth. Figure 6 displays the shape of the error for a few consecutive relaxation sweeps, starting with a random first guess: *While the error gets smooth very quickly, its size hardly changes!*

Writing the residual (5) in terms of the error, $r = f - Au = A(u^* - u) = Ae$, or explicitly,

$$r_{i,j} = (4e_{i,j} - e_{i-1,j} - e_{i+1,j} - e_{i,j-1} - e_{i,j+1}) / h^2 ,$$

we see that *the slow-down of residual reduction observed in Figure 5 is directly related to the error getting smooth*. In fact, the randomness of the error at the beginning corresponds to relatively large and random residual values $r_{i,j}$. The increasing smoothness of the error during the very first relaxation sweeps causes the residual to drop substantially. Once the error becomes smooth, the residual is small (relative to the size of the error itself) and it is plausible that a further reduction gets increasingly slow with increasing smoothness.

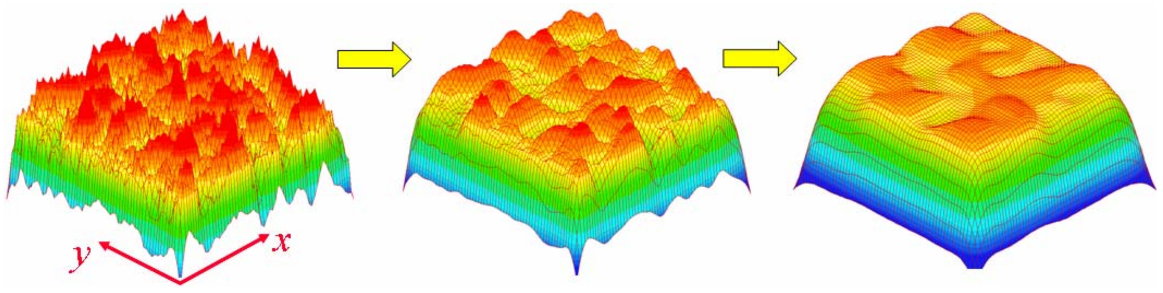


Figure 6: The shape of the error for the very first relaxation sweeps, starting from a random first guess.

Obviously, relaxation reduces non-smooth error "components" much faster than smooth ones. In fact, a closer analysis shows that the non-smooth components are reduced at a rate which is even independent of the mesh size *h*. This can be analyzed quantitatively by considering Fourier decompositions of the error and distinguishing "low" and "high" frequencies: By low frequencies we denote those which *can* be approximated ("are visible") also on a coarser level, whereas the high frequencies are those which *cannot* be approximated ("are invisible")[4] on a coarser grid. For the case of "standard coarsening" (i.e., the coarser level is obtained by doubling the mesh size), Figure 7 illustrates high and low frequencies in one dimension. From the figure we see that high frequencies are those with a wavelength less than 4*h*. Frequencies with larger wavelengths correspond to the low frequencies. This definition can directly be extended to higher dimensions: Using standard *h*→2*h* coarsening, a Fourier component is called high frequency if its wavelength is less than 4*h* in at least one spatial direction. Otherwise, it is called low frequency.

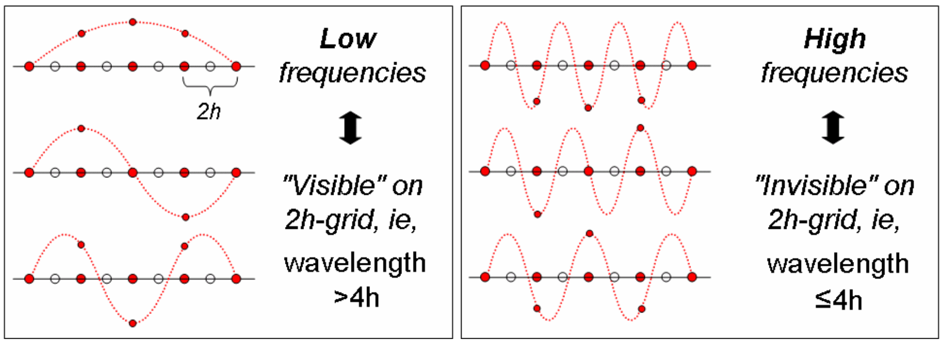

2h
Low frequencies
"Visible" on 2h-grid, ie, wavelength >4h
High frequencies
"Invisible" on 2h-grid, ie, wavelength ≤4h

Figure 7: Separation of Fourier components into low (smooth) and high (non-smooth) frequencies, relative to a coarser grid of mesh size 2h (red dots).

[4] More precisely, high frequency oscillations are not representable correctly any more on the next coarser level. In fact, if restricted to the coarser level, high frequencies coincide with certain low frequencies.

The separation of the error into low and high frequency Fourier components is the theoretical basis for a very general smoothing analysis (see, for example, [56] and the references given therein). In particular, one can define the so-called "smoothing factor" which is the minimum factor by which *all* high frequency error components are reduced per relaxation sweep. The main property of the smoothing factor is that it is *independent of the mesh size*. For example, for the 2D Poisson equation, the smoothing factor of GS relaxation is 0.5. Ultimately, the scalabilty of multigrid methods results from exploiting the *h*-independent smoothing behavior not just on a single grid but rather on a hierarchy of grids.

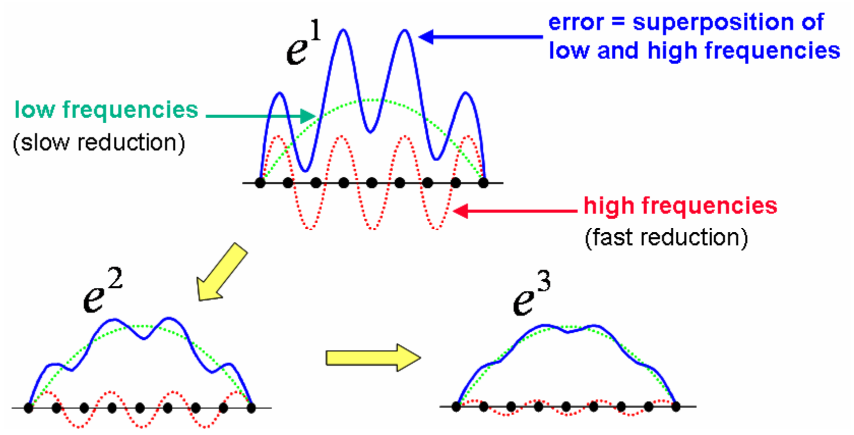


Figure 8: Illustration of how relaxation acts on some error which is the superposition of a low and a high frequency component.

2.3 Two-grid method

According to the previous section, it is only the low frequency error components which are reduced slowly by relaxation, whereas the high frequency components are reduced very rapidly and, most importantly, at a rate which is independent of the mesh size *h*. This gives rise to the idea to combine the smoothing property of relaxation with a correction step using a coarser grid ("coarse-grid correction step"). That is, we use relaxation only to smooth the error; once the error is smooth, we approximate it by means of a coarse grid[5].

Let us consider the case of just two consecutive grids based on "standard coarsening", Ω_h and Ω_{2h}. The following process describes the general frame of a two-grid method; various components still need to be specified. In particular, the restriction I_h^{2h} and interpolation I_{2h}^h - mapping grid functions on Ω_h to those on Ω_{2h} and vice versa - need to be chosen. Postponing this for a moment, the mathematical description of one iteration step (cycle), computing a new approximation, u_h^{m+1}, from a previous one, u_h^m, is as follows (cf. Figure 9 for a graphical representation of the same process):

1. Apply ν_1 (typically 1-2) smoothing steps starting with u_h^m. For ease of notation denote the resulting intermediate grid function again by u_h^m;
2. Compute the residual: $r_h^m = f_h - A_h u_h^m$;

[5] Note that only smooth quantities can be approximated by the coarse level. Consequently, it is not the original equation (3) which is transferred to the coarse grid, but rather the (equivalent) residual equation, $A_h e_h = r_h$.

3. Restrict the residual to the coarse grid: $r_{2h}^m = I_h^{2h} r_h^m$;
4. Solve the coarse-grid correction equations[5]: $A_{2h} e_{2h} = r_{2h}^m$;
5. Interpolate e_{2h} to the fine grid, and correct the old approximation: $u_h^m + I_{2h}^h e_{2h}$;
6. Apply ν_2 (typically 1-2) smoothing steps giving the new iterate u_h^{m+1}.

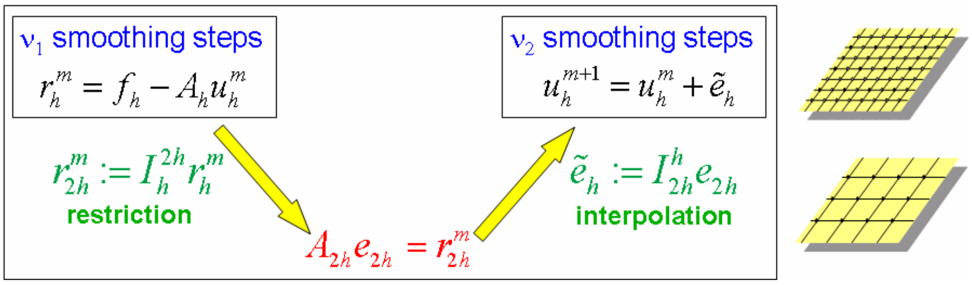

ν_1 smoothing steps
$r_h^m = f_h - A_h u_h^m$
ν_2 smoothing steps
$u_h^{m+1} = u_h^m + \tilde{e}_h$
$r_{2h}^m := I_h^{2h} r_h^m$
restriction
$\tilde{e}_h := I_{2h}^h e_{2h}$
interpolation
$A_{2h} e_{2h} = r_{2h}^m$

Figure 9: Mathematical description of a two-grid method

Typically, restriction is some local averaging, whereas interpolation is bi-linear. A important class of applications for which this is not appropriate is given in Section 2.5.1. While A_{2h} is often chosen to be the analog of A_h applied on the coarse level, there are also other possibilities (e.g., the Galerkin operator (10)). We mention that, generally, also the processes of smoothing and coarsening are components which can be selected differently. Reasonable choices specifically for the Poisson problem are summarized in Figure 10.

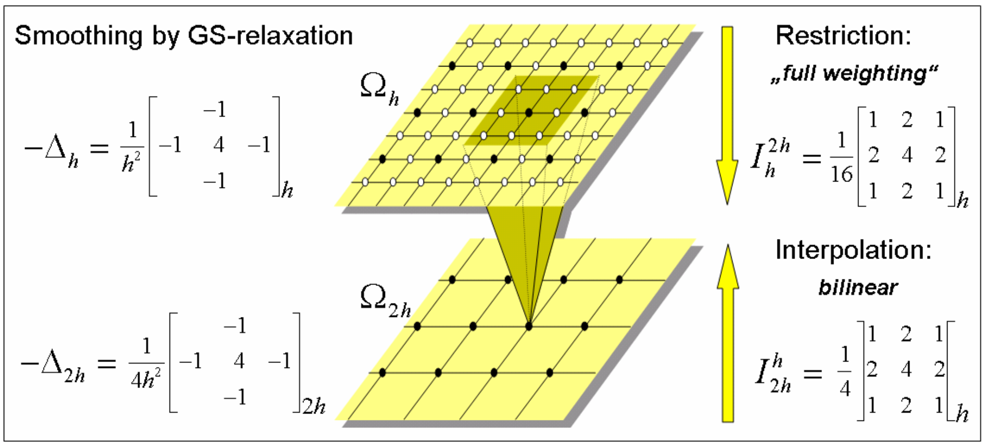

Smoothing by GS-relaxation
$-\Delta_h = \frac{1}{h^2}\begin{bmatrix} & -1 & \\ -1 & 4 & -1 \\ & -1 & \end{bmatrix}_h$
$-\Delta_{2h} = \frac{1}{4h^2}\begin{bmatrix} & -1 & \\ -1 & 4 & -1 \\ & -1 & \end{bmatrix}_{2h}$
Ω_h
Ω_{2h}
Restriction:
„full weighting“
$I_h^{2h} = \frac{1}{16}\begin{bmatrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{bmatrix}_h$
Interpolation:
bilinear
$I_{2h}^h = \frac{1}{4}\left]\begin{matrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{matrix}\right[_h$

Figure 10: Reasonable components defining a two-grid method for solving Poisson equation. The special fine-to-coarse averaging is usually called "full weighting"; the coarse-to-fine mapping is just bi-linear interpolation.

2.4 Multigrid and scalability

Mathematically, the two-grid method is sufficient to describe the multigrid principle. However, the coarse grid Ω_{2h} will generally still be much too fine to allow for an efficient solution. Hence, to obtain a numerically efficient process, the two-grid method has to be recursively applied in a straightforward way to increasingly coarse grids down to a coarsest grid for which the cost for a solution is negligible. Apart from the data transfer between grids (and the direct solver on the very coarsest grid), we see that smoothing is the only process performed on each grid level (right picture in Figure 11). Let us give some heuristic arguments why this process leads to a scalable method, meaning that its convergence rate is independent of *h*.

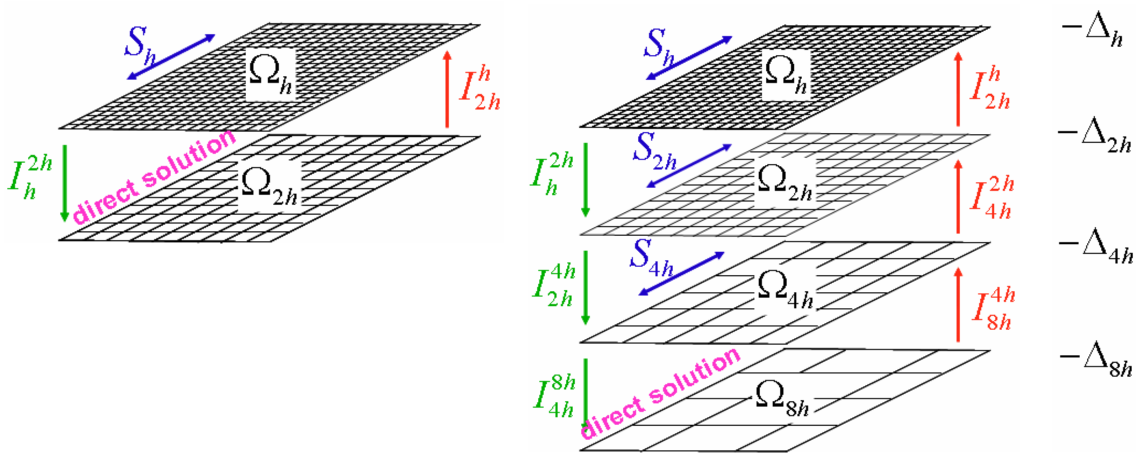


Figure 11: (left) Schematic view of two-grid method. (right) Recursive extension of the two-grid method to a multigrid method.

First, at the beginning of a cycle, all high frequency error components on Ω_h are reduced by a factor independent of *h* (the smoothing factor). The remaining (low frequency) error components are effectively reduced by means of the coarser grids. In fact, any of these low frequencies is high frequency w.r.t. the scale of a particular intermediate grid. Relaxation on this particular grid will reduce the corresponding error component efficiently. That is, *the primary purpose of each grid is to reduce those error components which are high frequency with respect to its own scale*. Putting all this together we understand that, within each multigrid cycle, all error frequencies will *uniformly* (i.e., at the same rate, namely, the one given by the smoothing factor) be reduced. Of course, all this is under the assumption that the transfer operators between levels do not interfere too much with the smoothing processes on all levels. For further mathematical investigations of the multigrid process we refer to the multigrid literature (e.g., [50] and the many references given therein)

The resulting method is not only scalable, each cycle is also very cheap. This is because the numerical work per level decays by a factor of 1/4 from level to level. The total work, W_{cycle}, required by a single multigrid cycle is asymptotically

$$W_{cycle} \approx W_1 + \frac{1}{4}W_1 + \frac{1}{16}W_1 + \ldots \to \frac{4}{3}W_1$$

where W_1 denotes the computational work performed on the finest level (for smoothing as well as data transfers between Ω_h and Ω_{2h}). If W_{GS} denotes the computational work for one single GS relaxation step on the finest level, and if we assume two smoothing steps to be done (i.e., $\nu_1=\nu_2=1$), we have $W_1 \approx 4W_{GS}$. That is, each multigrid cycle costs about the same as just a handful of GS sweeps on the finest level,

$$W_{cycle} \approx \frac{4}{3}4W_{GS} \approx 5.3W_{GS}\,.$$

Table 1 shows timings demonstrating the dramatic gain by using GS relaxation as a smoother in a multigrid process compared to using it as a solver. For the multigrid cycle, we use the components shown in Figure 10. As expected (due to its $O(N^2)$ complexity mentioned earlier), GS relaxation as a solver is virtually of no practical use, whereas the corresponding multigrid process is highly efficient: less than 30 sec on a standard laptop to solve Poisson equation on a mesh with more than 16 million variables! For completeness, the last row shows the performance of a particularly efficient multigrid approach known as "Full Multigrid[6] (FMG)". This approach is again three times faster than plain multigrid cycling.

mesh size	256x256	512x512	1024x1024	2048x2048	4096x4096
plain GS relaxation	86	1250	18750	---	---
multigrid cycling	0.09	0.44	1.69	6.77	27.41
Full multigrid (FMG)	0.03	0.16	0.53	2.14	8.30

Table 1: Computational time (in sec) to solve Poisson's equation

2.5 Robust multigrid

The multigrid technique as outlined for Poisson's equation carries over to general elliptic PDEs as well as coupled systems of PDEs. Usually, however, the smoothing properties of plain GS relaxation will not be sufficient any more to ensure high multigrid efficiency. Instead, suitable block-relaxations (e.g. line-relaxation) may be required, or even completely different iterative methods with robust smoothing properties such as iterative methods based on incomplete LU decomposition ("ILU"). Here and in the following, we use the term "smoother" to denote any method which can be used as a smoother in the multigrid context[7].

In the remainder of this section we will summarize a few important aspects regarding the generalization of GMG which are of particular relevance for reservoir simulation applications. These aspects will later be used to demonstrate the conceptual differences between geometric and algebraic multigrid and make clear AMG's major advantages for a practical application. For more details regarding the generalization of GMG we refer to [50].

2.5.1 Two characteristic model cases

Any reasonable multigrid process requires an efficient interplay between smoothing and coarse-grid correction. Roughly, the following properties need to be satisfied:

A. On each level, after having applied a few smoothing steps, the resulting error can sufficiently well be approximated on the next coarser level.
B. The discretization operators on the coarser grids approximate the respective finer ones sufficiently well.
C. The intergrid data transfer (in particular, the interpolation) is accurate enough.

We briefly discuss two model cases demonstrating the meaning of properties A-C. Both models are relevant for the efficient multigrid treatment of reservoir equations.

[6] Essentially, FMG employs regular multigrid cycles in a nested way. Unfortunately, there is no analog of this particularly fast multigrid variant in the context of algebraic multigrid to be introduced later.
[7] In principle, all iterative methods can be used as a smoother. Typical robust smoothers used in practice are (block-) relaxation methods, collective relaxation, distributive relaxation, ILU-type methods, approximate inverses, etc.

Anisotropic problems

Given a smoother, the details of its smoothing properties depend on the application such as its degree of anisotropy. For instance, while plain GS relaxation is a very efficient smoother for isotropic Poisson equations (2), this is no longer true for anisotropic equations,

(7) $$-\varepsilon u_{xx} - u_{yy} = f \quad \text{with } \varepsilon \ll 1,$$

discretized by the standard 5-point stencil. Instead of (6) we now obtain

(8) $$e_{i,j} \leftarrow (\varepsilon e_{i-1,j} + \varepsilon e_{i+1,j} + e_{i,j-1} + e_{i,j+1}) / 2(1+\varepsilon)$$

showing that smooth error is essentially a weighted average of its neighbors *in y-direction only* whereas it is hardly influenced by values in *x*-direction. This is illustrated in Figure 12.

$$\frac{1}{h^2}\begin{bmatrix} & -1 & \\ -\varepsilon & 2(1+\varepsilon) & -\varepsilon \\ & -1 & \end{bmatrix}$$

$$0 < \varepsilon \ll 1$$

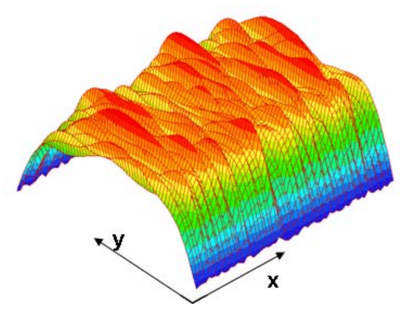


Figure 12: Error smoothing by GS relaxation applied to the anisotropic Poisson equation (7). Starting from a random first guess, the error gets quickly smooth only in the y-direction (i.e. in the direction of "strong connections").

Obviously, such error cannot reasonably be approximated on Ω_{2h} (i.e., Property A from above is violated), that is, the smoothing properties of plain GS relaxation are not sufficient in connection with standard *h→2h* coarsening. We need a different smoother, namely, one which smoothes the error *uniformly*, that is, in *all* spatial directions. We just mention that this can easily be achieved by a block-variant of GS relaxation which relaxes (ie, solves for) all *strongly* coupled points *simultaneously*. In our model case, this corresponds to *line relaxation* with the lines being parallel to the y-axis. Similarly, x-line relaxation has perfect uniform smoothing properties if $\varepsilon \gg 1$. In more realistic cases of strongly varying coefficients, we can combine both approaches: *Alternating line-relaxation* (i.e. one sweep of x-line relaxation followed by one sweep of y-line relaxation) has robust uniform smoothing properties for all kinds of anisotropy.

While alternating line-relaxation provides an efficient smoother for general anisotropic applications in 2D, this is no longer true for general 3D applications. In fact, the 3D analog of alternating *line*-relaxation is alternating *plane*-relaxation, employing robust 2D multigrid processes within each plane (using alternating *line*-relaxation for smoothing). This shows that, even in relatively simple model cases, complex smoothers may be required in order to ensure sufficient smoothing properties.

Discontinuous coefficients

In reservoir simulation, drastic changes in permeabilities lead to PDEs with strongly discontinuous coefficients. In such applications, care has to be taken to satisfy properties B and C from above. To illustrate this, let us consider an exemplary diffusion problem,

(9) $$-\nabla(a\nabla u) = f \,,$$

on the unit square with the discontinuous coefficient *a* being defined in Figure 13. The figure also shows the shape of the error after having performed a few GS relaxation steps. Obviously, the error is not really smooth any more; it reflects the discontinuity along the interior line of discontinuity. Clearly, in contrast to the anisotropic problems, this behavior cannot be avoided by just choosing a different smoother. As a consequence, *linear* interpolation as employed in the Poisson case (Figure 10) is not suitable near the line of discontinuity. In fact, linear interpolation is feasible only if we can assume error after smoothing to be continuous. Moreover, unless the line of discontinuity coincides with grid lines on all coarser levels (which, generally cannot be ensured in real-life applications, cf. Figure 14), the discretization accuracy of the coarse level operators will be quite bad. Hence, both property B and C are violated.

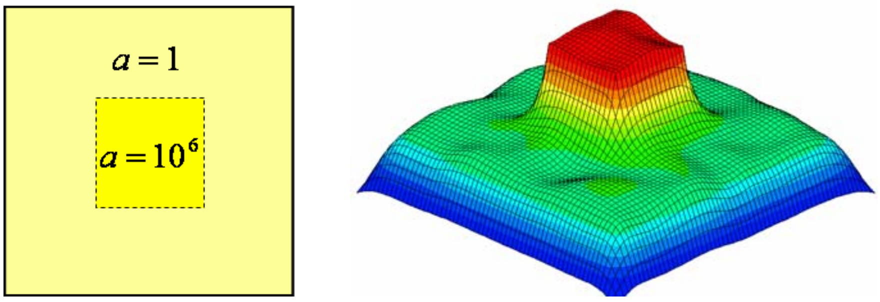


Figure 13: (left) Definition of discontinuous coefficient a in (9). (right) "Smooth" error obtained after a few sweeps of GS relaxation starting with a random guess.

Remedies to such kind of situation (cf. also Section 3.2.2 on Algebraic Multigrid):

- A reasonable interpolation needs to "reflect" the discontinuities. This can most easily be achieved by basing interpolation directly on the difference equations. (Such kind of stencil-based interpolation is called "operator-dependent interpolation" and has been introduced in [1] for the first time.)
- Rather than using the "same" discretization on each level, the so-called Galerkin operator,

(10) $$A_{2h} = I_h^{2h} A_h I_{2h}^h \,,$$

provides a robust alternative. This particular coarse-grid operator has its origin in finite elements[8]. Note that (10) is a simple multiplication between matrices which can be done purely algebraically.

[8] In case of symmetric and positive definite matrices, this operator can also be motivated independently: "Galerkin-based" multigrid satisfies a *variational principle* which, essentially, ensures optimal corrections from coarser levels (as good as they can be, given the current coarser levels and transfer operators) if "optimality" is measured w.r.t. the energy norm, that is, $||e||=(Ae,e)^{1/2}$.

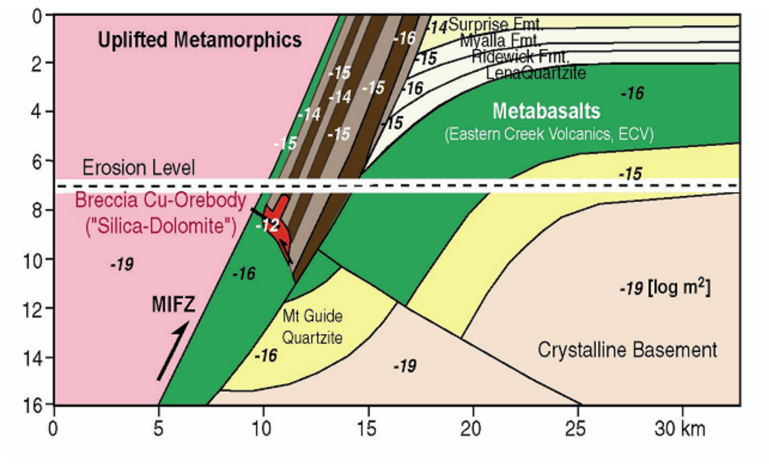


Figure 14: Drastic changes in permeabilities causing discontinuous coefficients

2.5.2 Towards Algebraic Multigrid

By now we have only considered standard *h→2h* coarsening. However, to satisfy Property A, this is not necessary. In fact, Property A only says that error after relaxation needs to be smooth *relative to the coarse grids* used. This indicates that we can loosen the requirements on the smoother and still maintain an efficient interplay with the coarse-grid correction if we introduce more flexibility in the selection of coarser grids. For the anisotropic case (7), for instance, we can continue to use plain GS relaxation for smoothing if we coarsen only in the direction of smoothness, that is, if we replace standard coarsening by “semi-coarsening” in *y*-direction (cf. Figure 15). Similarly, if $\varepsilon \gg 1$, we can use semi-coarsening in *x*-direction. This idea immediately generalizes to 3D applications.

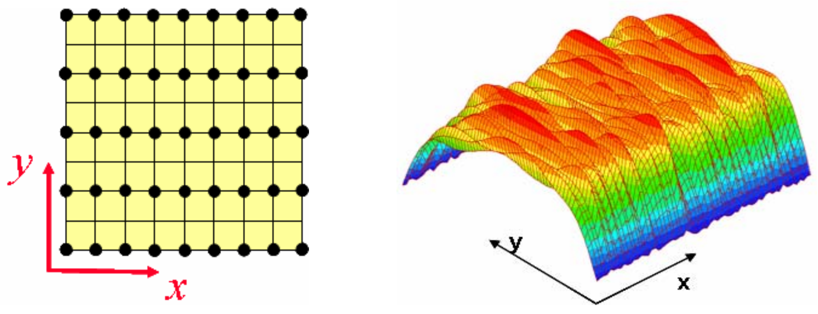


Figure 15: Semi-coarsening in y-direction for anisotropic applications

Even though smoothing can indeed be simplified to quite some extent, the usefulness of semi-coarsening in real-life problems seems rather limited: In case of anisotropies of strongly varying directions, neither *x*-semi coarsening nor *y*-semi coarsening will do. In fact, it can be shown that the two types of coarsening need to be employed simultaneously: each multigrid level needs to employ more than one coarser grid, leading to the fairly complex “multiple semi-coarsening” technique (i.e. semi-coarsening in multiple directions, [33], see Figure 16).

Summarizing, a *robust* GMG solver for general elliptic PDEs requires either complex smoothers or sophisticated multiple semi-coarsening techniques. Moreover, to cover also discontinuous coefficients, both operator-dependent interpolation as well as Galerkin coarse-grid operators (10) are required. Hence, robust and general GMG methods may get rather complicated, even in case of structured meshes. In case of fully unstructured meshes, the development of GMG solvers gets even more troublesome, if possible at all: On the one hand, a natural, easy-to-construct grid hierarchy may simply not exist. On the other hand, even if such a hierarchy was available, neither the above mentioned robust (line- and plane-) smoothers nor the multiple semi-coarsening technique carry over to the unstructured case.

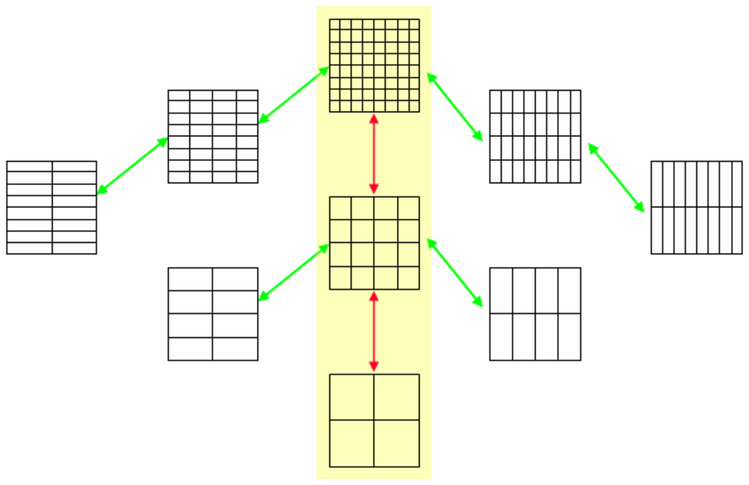


Figure 16: Outline of multiple semi-coarsening strategy for anisotropic problems in connection with plain GS smoothing

One should observe that a major technical hurdle in constructing robust geometric multigrid methods is due to the fact that, generally, a grid hierarchy is assumed to be *pre-defined*. Since smoothness on a grid is always to be understood relative to a next coarser grid, this may require complex smoothing methods of the type mentioned further above. On the other hand, the semi-coarsening technique can principally be used to considerably weaken the requirements on the smoother; but as long as this technique also assumes pre-defined grid hierarchies, it seems still far from being flexible enough for an efficient treatment in "real-life".

As a way out one needs still more flexibility in constructing coarser levels. For instance, a *dynamic* semi-coarsening strategy, allowing *local adaptations* to the true smoothing properties of the current smoother, would cure many of the difficulties mentioned before. Clearly, it is hardly possible to realize this in a structured geometric environment. However, observing that the operator-dependent interpolation and the Galerkin operator mentioned in Section 2.5.1 are essentially independent of the geometry, one might attempt to formulate a generalized multigrid method in purely algebraic terms, that is, without any direct reference to underlying grids. In fact, this is essentially what Algebraic Multigrid is all about.

3 Algebraic Multigrid (AMG)

We have seen that GMG can be used to solve elliptic PDEs very efficiently. For instance, Poisson equation on a 16 million point mesh can be solved in a few seconds on a standard laptop (see Table 1)! Since around 1970, world-wide research has demonstrated that the multigrid principle - the combination of smoothing and coarse-grid correction - is not only efficient but also very general. The development of multigrid can actually be regarded as the most important development in numerical mathematics over the last 40 years.

However, in spite of its revolutionary success in academia, GMG had only very limited impact on the development of industrial simulation software. From a practical point of view, there seem to be essentially two reasons for this:

1. GMG solvers are not suited to simply replace classical solvers in existing simulators. This is because GMG is a general *technique* to solve PDEs, and a GMG-based simulator has to be conceptually tailored to this technique, it has to be "multigridded".
2. The alternative - to develop a GMG-based simulator from scratch - is possible in principle. However, the technical realization may, under industrial requirements, become rather cumbersome. In particular, in case of unstructured meshes the development of GMG-based simulators gets even more troublesome, if possible at all (cf. Figure 19).

The algebraic multigrid[9] (AMG) approach was driven by the attempt to automate and generalize GMG so that it can be applied directly to certain (sparse) matrix equations without explicitly referring to geometry and without requiring any pre-defined hierarchy. The construction of a hierarchy is actually part of AMG and is done fully automatically. That is, AMG solvers attempt to combine the advantages of GMG with those of easy-to-use plug-in solvers. In contrast to GMG, AMG-based solvers are easy to integrate into existing simulators, actually one of the major reasons for their industrial success.

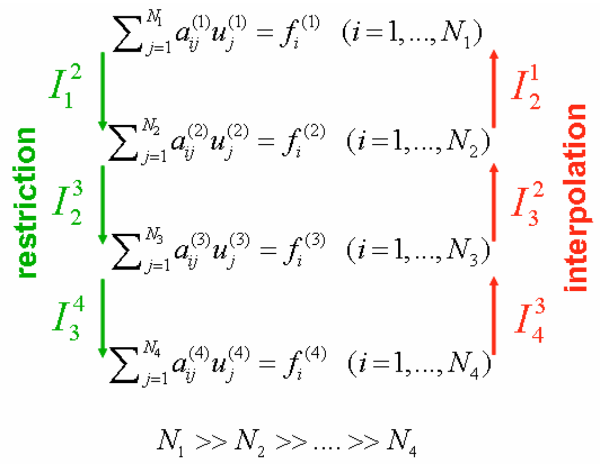

$\sum_{j=1}^{N_1} a_{ij}^{(1)} u_j^{(1)} = f_i^{(1)} \quad (i=1,\dots,N_1)$
$\sum_{j=1}^{N_2} a_{ij}^{(2)} u_j^{(2)} = f_i^{(2)} \quad (i=1,\dots,N_2)$
$\sum_{j=1}^{N_3} a_{ij}^{(3)} u_j^{(3)} = f_i^{(3)} \quad (i=1,\dots,N_3)$
$\sum_{j=1}^{N_4} a_{ij}^{(4)} u_j^{(4)} = f_i^{(4)} \quad (i=1,\dots,N_4)$
restriction
I_1^2
I_2^3
I_3^4
interpolation
I_2^1
I_3^2
I_4^3
$N_1 >> N_2 >> .... >> N_4$

Figure 17: Being based on the same basic principles as GMG, AMG operates on a hierarchy of increasingly coarse matrix problems all of which are constructed fully automatically as part of the numerical algorithm ("setup phase").

AMG can formally be described purely algebraically as an approximate Schur complement approach. This approach has the advantage of providing a unified abstract description of various hierarchical methods including, for instance, multilevel variants of ILU (see, e.g., [34]). From a practical point of view, however, aspects regarding the heuristic motivation for certain algorithmic details may get less transparent or even get lost. Instead, we prefer a more intuitive description which allows pointing out the analogy between AMG and GMG. For that purpose we "visualize" a matrix by its graph representation (nodes correspond to variables, edges stand for non-vanishing matrix entries). This way, at least formally, we can argue and describe AMG in a GMG-like terminology. For instance, we can still think in terms of grids, grid points, grid functions, etc.

[9] Since algebraic multigrid is applied to matrices rather than grid structures, 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*.

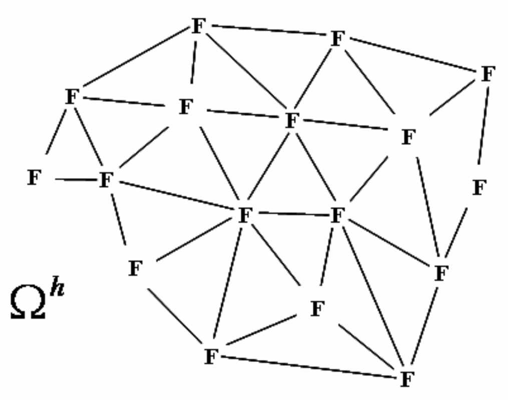


Figure 18: Graph representation of a matrix: The nodes of the graph stand for the variables; edges for non-vanishing couplings between variables. In analogy to the finest grid in the context of GMG, we denote this graph by Ω^h. "F" stands for fine level nodes.

In the next subsections, we briefly introduce the basic ideas of "classical" AMG. Note that, although AMG is formally more general, we generally still have the solution of PDEs in mind. Also, in spite of the fact that geometry is not explicitly exploited, the heuristics behind the algorithmic details is often still motivated by geometric arguments. Some results will demonstrate AMG's efficiency and potential.

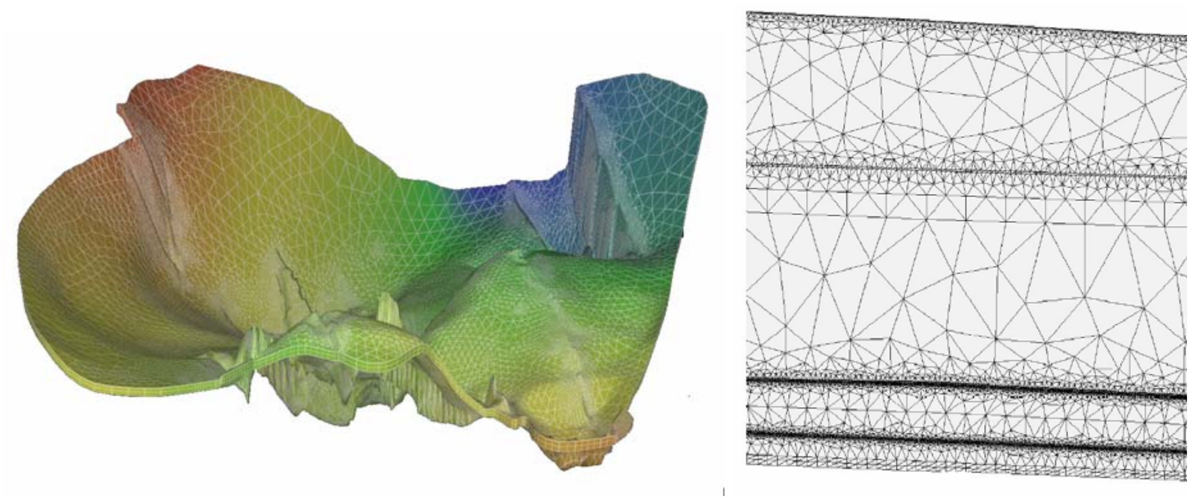


Figure 19: (left) Large-scale 3D basin flow model with unstructured mesh. The pentahedral prismatic mesh necessarily incorporates a number of faulty zones which leads to a vertical distortion of the prisms along these locations. (right) A complex cross-sectional 2D problem. The triangular mesh is fully unstructured and locally refined in a layered geometry. The problem models an aquifer-aquitard system with heterogeneous distribution of conductivity and storativity. (Courtesy of Wasy GmbH)

3.1 History

The development of AMG started in the early nineteen-eighties [9,10,11,35,36,42] when Galerkin-based coarse-grid correction processes and, in particular, operator-dependent interpolation were introduced into GMG ([1], see also Section 2.5.1). One of the motivations for AMG was the observation that reasonable operator-dependent interpolation and the Galerkin operator (10) 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 [20]. However, regarding the selection of coarser levels, this code was still geometrically based. In a purely algebraic setting, the coarsening process itself is also defined algebraically, i.e., strictly based on information contained in the given matrix.

The very first AMG program is described and investigated in [35,36,42], see also [16]. At that time, the primary interest was of a scientific nature, namely, to demonstrate that - under certain conditions - multigrid methods can be designed even when no grids are available or, if available, without exploiting them. The original AMG approach was designed for the class of linear algebraic systems (1) with matrices *A* being symmetric, positive definite and "close to" weakly diagonally dominant M-matrices[10]. Problems like this widely occur in connection with discretized *scalar* elliptic PDEs of 2nd order, the simplest one being the Poisson equation (2).

Since the resulting code, AMG1R5, was made publicly available in the mid eighties, there had been no substantial further research and development in AMG for many years. In spite of its potential, it took until around 1995 before there was a remarkable increase of interest in AMG and, more generally, in algebraically oriented hierarchical methods, both in science and applications. Among other reasons, this interest was fed by the increasing geometrical complexity of applications which, technically, limited the immediate use of alternate fast solvers such as those based on GMG. Another reason was the steadily increasing demand for efficient "plug-in" solvers. In reservoir simulation this demand was driven by ever-increasing problem sizes, complex structures, heterogeneities, multiphase flows, and wells which made clear the limits of classical one-level solvers used in industrial simulators. This development was similar in many other areas of numerical simulation.

Fostered by this situation, sophisticated extensions of the original AMG method have been developed aiming at increasing its range of applicability. In particular, substantial progress has been achieved towards the efficient treatment of *coupled* systems of PDEs (see Section 4). Moreover, various other hierarchical algebraic approaches have been developed which differ substantially from the original AMG[11].

3.2 Basics

We recall that, from a practical point of view, a major drawback of GMG is that it requires a *pre-defined* hierarchy of grids. This is not only the main reason for its technical difficulties in treating unstructured grid problems (if treatable at all), it is also responsible for the fact that particularly strong smoothers need to be employed in order to achieve a high robustness.

Once we give up the request of a pre-defined hierarchy, we can get along with much simpler smoothers and, at the same time, gain more flexibility in treating unstructured meshes. Formally, this is the starting point of AMG and the main reason for its flexibility. In fact, AMG fixes the smoother to some simple scheme - typically just plain GS relaxation - and attempts to ensure an efficient interplay with the coarse-grid correction by *locally* and *dynamically* adapting coarser levels and (operator-based) interpolations to the smoothing properties of the smoother. Geometrically speaking, AMG attempts to coarsen only "in directions" in which relaxation really smoothes the error for the problem at hand. For many classes of applications, the relevant information is contained in the (discretization) matrix

[10] More specifically: $a_{ii} > 0,\ a_{ij} \le 0\ (i \ne j)$ and $\sum_j a_{ij} \ge 0$.

[11] AMG, as we understand it, is structurally completely analogous to GMG in the sense that the basic algorithmic components - smoothing and coarse-grid correction - play the same role.

itself, so that the complete coarsening process can be performed based only on information contained in the matrix, without any reference to the grid.

Hence, the application of AMG to solve a given problem (1) is a two-part process: First, there is the *setup phase* which consists of recursively choosing the coarser levels and dynamically defining the transfer and coarse-grid operators. Second, the *solution phase* just uses the resulting components in order to perform normal multigrid cycling until a desired level of tolerance is reached. Since the solution phase is straightforward, in the following we need to consider only the setup phase.

3.2.1 Smoothing and coarsening

In order to motivate the basic AMG idea, we first need to define "smoothness" in an algebraic context and understand how AMG can exploit this in constructing coarser levels. Recall first that smoothness in GMG is defined *relative to a given coarser grid*. For example, an error may be smooth with respect to a semi-coarsened grid but not with respect to a standard *h→2h* coarsened grid (see Figure 15). Since there are no pre-defined coarser grids in AMG (there may even be no grids at all), smoothing in the previous sense becomes meaningless. Instead, we exploit the relation between smoothness and slow convergence as discussed in Section 2.2 and simply *define* an error e to be "algebraically smooth" *if it is slow to converge* with respect to the selected smoother. In other words, we call an error "smooth" if it *has to be* approximated by means of a coarser level[12] (which then needs to be properly constructed!). From an algebraic point of view, this is the important point in distinguishing smooth and non-smooth errors.

Note that it is not important whether relaxation really smoothes the error in any geometric sense. What is important, though, is that *algebraically smooth error can be characterized to a degree that makes it possible to perform an automatic coarsening process.* In order to demonstrate what this means, let us reconsider the arguments of Section 2.2 for the class of matrix problems with weakly diagonally dominant M-matrices[10], assuming GS relaxation to be used for smoothing:
In one GS relaxation sweep, each variable u_i in turn will be replaced by the new value, $\bar{u}_i$,

(11) $$\bar{u}_i = (f_i - \sum_{j \neq i} a_{ij} u_j) / a_{ii} = u_i + r_i / a_{ii} \quad \text{with} \quad r_i = f_i - \sum_j a_{ij} u_j$$

denoting the i-th residual *before* relaxing u_i. In terms of the error, $e_i = u_i^* - u_i$, this means

(12) $$\bar{e}_i = e_i - r_i / a_{ii}$$

which follows immediately by subtracting (11) from the fixed point equation, $\bar{u}_i^* = u_i^*$. By definition, error is algebraically smooth if it is slow to converge, i.e., if $\bar{e}_i \approx e_i$. From this and (12) we heuristically conclude that algebraically smooth error is characterized by the relation

(13) $$| r_i | \ll a_{ii} | e_i |,$$

[12] Remember that, roughly speaking, any component of the error which cannot be reduced by smoothing necessarily needs to be reduced by computations on coarser levels and vice versa.

meaning that the (scaled) residuals are much smaller than the errors themselves. This simple relation expresses the most important property of algebraically smooth error: while such error may still be very large globally, *locally* we can approximate every e_i fairly well as a function of its neighboring error values e_j,

$$(r_i =)\ \ a_{ii}e_i + \sum\nolimits_{j\neq i} a_{ij}e_j \approx 0$$

or

$$e_i \approx \frac{1}{a_{ii}} \sum_{j\neq i} |a_{ij}|\, e_j \ . \tag{14}$$

The latter gives us a simple and practical characterization: At any point, i, the value of algebraically smooth error is well approximated by the weighted average of its neighboring values with weights being given by the corresponding coupling coefficients, a_{ij}. This immediately implies that algebraically smooth error "changes slowly in the direction of strong couplings", that is, from i to j if $|a_{ij}|$ is large (cf. Figure 20).

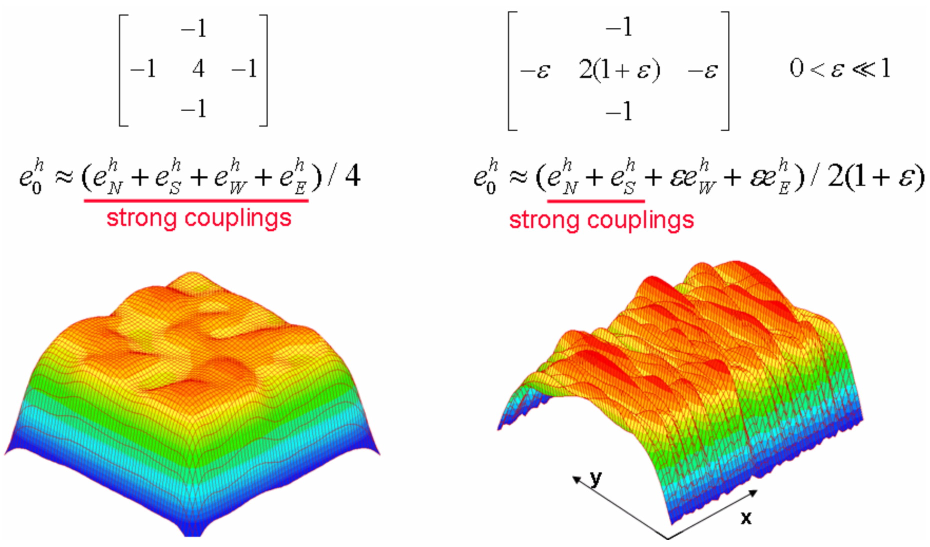


Figure 20: Visualization of algebraic smoothness for isotropic and anisotropic problems

This sounds very familiar from the GMG discussion in Section 2.2. However, while the previous characterization was just a *conclusion* in the case of GMG, in the context of AMG it plays a different role in that it provides the basis for explicitly performing the coarsening process: Coarsening will only be in directions where smooth error changes slowly, i.e., "in the direction of strong couplings". Even more, it will also be the basis for defining interpolation between levels.

3.2.2 The two-level method

Based on the characterization of algebraic smoothness, this section outlines the main idea of how to design a two-level method. Generally, the following three steps need to be followed:

1. Subdivide Ω^h, the set of all variables, into F- and C-variables: while the F-variables remain on the finest level, the C-variables will be taken down to the next coarser one.

That is, the next coarser level, formally denoted by Ω^H, consists of just the C-variables. We call this process the "C/F-splitting".

2. Define transfer operators ("interpolation" and "restriction") which map coarse vectors (defined on Ω^H) into fine ones (defined on Ω^h) and vice versa.
3. Finally, define the matrix operator ("coarse-grid correction operator") on Ω^H.

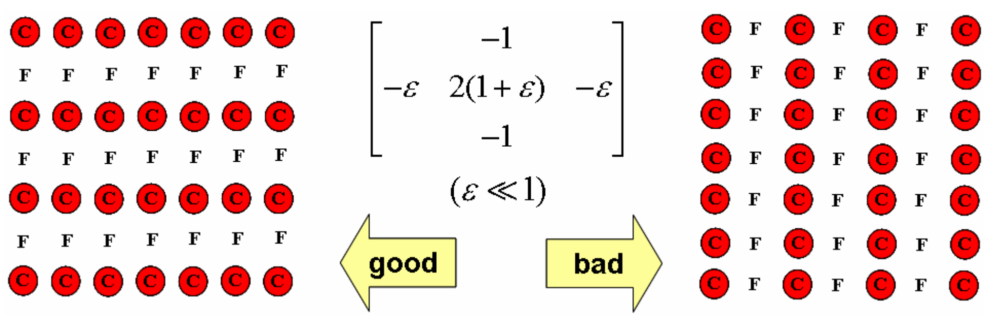

$\begin{bmatrix} & -1 & \\ -\varepsilon & 2(1+\varepsilon) & -\varepsilon \\ & -1 & \end{bmatrix}$
$(\varepsilon \ll 1)$
good
bad
C
F

Figure 21: Meaning of "coarsening in the direction of strong couplings" in case of the anisotropic model operator.

As motivated before, coarsening should be "in the direction of strong couplings". What this means in a simple model case is illustrated in Figure 21. Applying this idea in a more general context means that the C/F-splitting should be done so that F-variables are "sufficiently strongly" coupled to C-variables. Conflicting to this is the goal that we want a rapid coarsening (producing low fill-in on coarser levels in later steps). This gives rise to the following two criteria in performing a C/F-splitting:

1. The set of C-variables should be *independent*, that is, C-variables should not be strongly coupled among each other.
2. Under the previous constraint, the set of C-variables should be a *maximal* set.

While these criteria reflect the most important goals of a C/F-splitting, obviously, there is much arbitrariness to realize a splitting concretely. We cannot go into further details here but rather refer to [36,43,44] for concrete algorithms and additional motivations.

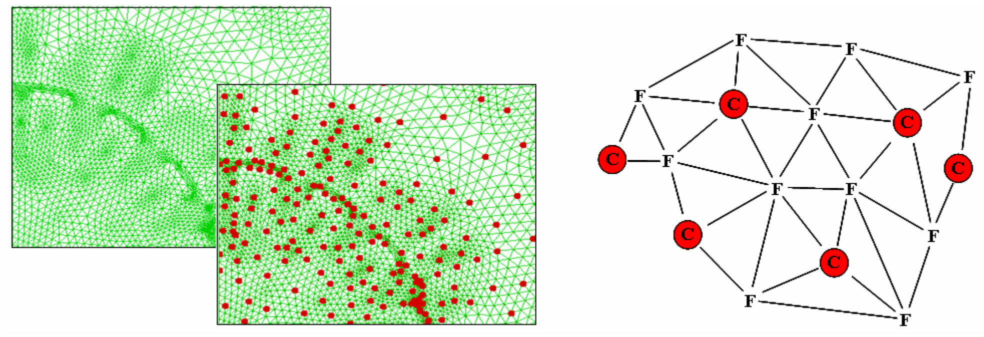


Figure 22: Illustration of the C/F-splitting[13]

Once a coarse level has been constructed, we need to define a reasonable coarse-to-fine interpolation. In the simplest case, we directly exploit (14) saying that the value of a smooth

[13] We here make use of the graph representation (see Figure 18), the grids are just for illustration. To simplify the description, we assume that only edges are shown which correspond to *strong* couplings (based on some reasonable threshold); *weak* couplings are ignored in the graph.

error is essentially determined by the values of its strongly coupled neighbors (those with large coefficients). Applying this to an F-variable, we obtain a simple interpolation formula,

(15) $$e_i^h = \alpha_i \sum_{j \in C} |a_{ij}|\, e_j^H \quad (i \in F) \quad \text{and} \quad e_i^h = e_i^H \quad (i \in C)$$

where the scaling factor, α_i, is defined by collapsing all F-to-F couplings to the diagonal (cf. Figure 23). Since this simple interpolation formula uses only "direct" couplings, it is called "direct" interpolation. While useful in simple applications, in general more complex types of operator-dependent interpolation are preferable both for reasons of efficiency and robustness. However, (15) shows the main principle; for more details and variants, see [43].

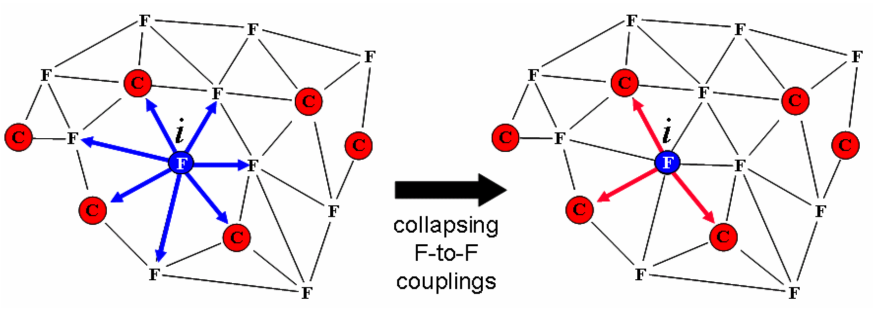


Figure 23: Principle of interpolation

Denoting the interpolation matrix by I_H^h, the restriction is generally defined as its transpose, that is, $I_h^H = (I_H^h)^T$. Once both transfer matrices are available, the coarse-level matrix is defined to be the *Galerkin* operator,

(16) $$A_H = I_h^H A_h I_H^h \, .$$

We have introduced this kind of algebraically defined coarse-grid correction operator already in GMG, see (10). We here just recall that the Galerkin operator is mathematically justified by its optimality property regarding the quality of coarse-grid corrections[8].

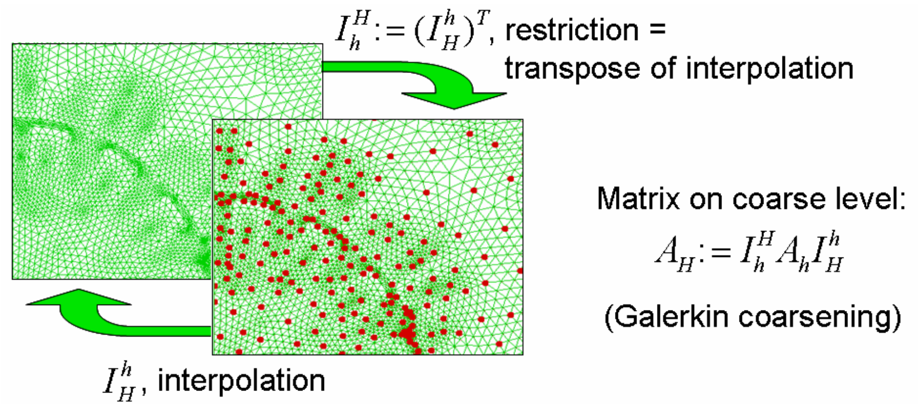


Figure 24: The ingredients of a two-level method

Summarizing, we obtain the two-grid components as illustrated in Figure 24. Using these components, the final two-grid method runs exactly as in GMG, see Figure 9, just replacing 2*h* by *H*. Note that there are only two components which actually have to be constructed, namely, the C/F-splitting and the interpolation matrix, I_H^h. We point out that the "operator-dependency" of interpolation, together with the optimality property of the Galerkin operator, makes AMG an efficient tool also for discontinuous coefficient applications. This is in contrast to standard GMG, cf. Section 2.5.1. In fact, AMG's interpolation (15) can be regarded as a generalization of the operator-dependent GMG interpolation introduced in [1,20].

3.2.3 Two-level theory

AMG methods used in practice[14], are largely heuristically motivated. However, under certain assumptions, in particular symmetry and positive definiteness ("s.p.d."), some theory is available. In particular, 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. In the following, we give an exemplary result.

Theorem: Assume the weakly diagonally dominant M-matrix A to be s.p.d. Furthermore, for any fixed $0 < \tau \leq 1$, assume a C/F-splitting to be selected so that, for all $i \in F$,

$$\sum_{k \in C} |a_{ik}| \geq \tau \sum_{j \neq i} |a_{ij}| \tag{17}$$

and define interpolation according to (15). Then the two-level method with GS smoothing has a convergence factor $0 \leq \rho < 1$ which does *not* depend on the matrix (neither the coefficients nor its dimension) but only on τ. While in the limit case $\tau = 1$ the resulting method degenerates to a direct solver, decreasing $\tau \to 0$ causes increasingly slow convergence, $\rho \to 1$.

To satisfy (17) with both a *low* number of C-variables and a reasonably *large* τ means that only few of the strongest neighbors should be put into C; the use of weak couplings would increase the computational work but hardly affect the convergence. Note that this implicitly means that coarsening will be in the direction of smoothness.

The more strong connections are put into C (and used for interpolation), the better the convergence can be. In the limit case $\tau = 1$, for which the two-grid method degenerates to a direct solver[15], (17) means that, for each $i \in F$, *all of its neighbors* are in *C*. In this case, interpolation is "most accurate" but also most expensive. In fact, the resulting direct solvers are of no practical relevance since, in terms of computational work and memory requirement, they will generally be extremely inefficient if recursively extended to a hierarchy of levels.

However, this limit shows something important, namely, that the *convergence* of AMG is not generally a problem; the more effort is put into its coarsening, the faster the convergence can be. More specifically, by suitably selecting τ, AMG can always be *forced* to converge rapidly. Unfortunately, however, the benefit of an improved interpolation in terms of convergence speed is offset by the expense in terms of additional computational work (which is also directly related to the memory requirement). That is, from a practical point of view, a major problem in designing efficient AMG algorithms is not convergence but rather the tradeoff between convergence and numerical workload. Keeping the balance between these aspects is the ultimate goal of any practical algorithm! Note that this is, in a sense, just opposite to

[14] Remember that there is a lot of arbitrariness in both the C/F-splitting and the definition of interpolation.
[15] As a matter of fact, the limit case $\tau = 1$ results in a direct solver for *any* non-singular matrix *A*.

GMG where the numerical work per cycle is known and "controllable" but the convergence may not be satisfactory.

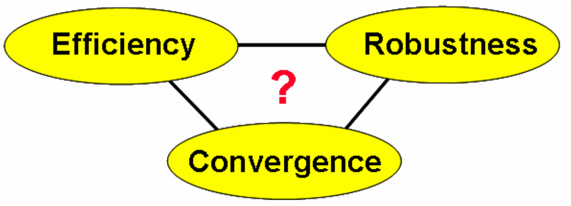


Figure 25: The key to an efficient AMG approach: The tradeoff between convergence, workload and robustness.

3.2.4 Recursive definition

The recursive application of two-grid cycles finally leads to real multigrid cycles. Since this is, formally, completely analogous to GMG (cf. Figure 11), we do not need to explain this in more detail but rather refer to Figure 26.

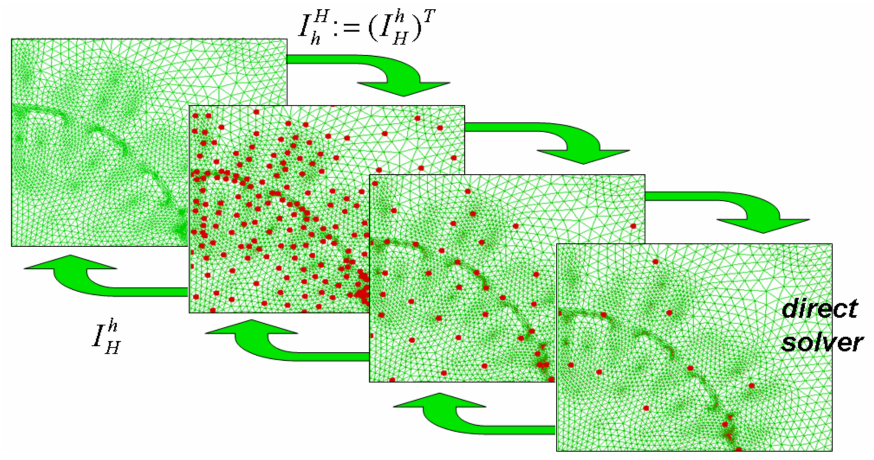


Figure 26: Complete AMG cycle recursively defined two-grid cycles

3.3 *AMG in practice*

Before an AMG cycle can finally be composed and the actual solution phase can start, the setup phase - in which all AMG components are recursively computed - has to be concluded. This overhead is the price for the flexibility of AMG and its simplicity of use and is one reason for the fact that AMG is usually less efficient than GMG (if applied to problems for which GMG *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 convergence. Nevertheless, if applied to typical elliptic test problems, the computational cost of AMG's solution phase (ignoring the setup cost) should be comparable to the cost of a *robust* GMG solver.

However, AMG should not be regarded as a competitor of GMG. The strengths of AMG are its robustness, its applicability in complex geometric situations and its applicability to even solve certain problems which are beyond the reach of GMG, in particular, problems with no geometric or continuous background at all (as long as the underlying matrices have similar properties as those derived from elliptic PDEs). That is, AMG provides an attractive multi-level variant whenever GMG 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 (one-level) methods.

All results given in this section are obtained by the SAMG software library [47].

3.3.1 Remarks on generalizations

Up to now, for ease of motivation, we have confined ourselves to the case of weakly diagonally dominant M-matrices. Although representative for some important class of problems, most applications will not strictly be of that type. "Mild" deviations from the M-matrix type or the weak diagonal dominance can simply be ignored. However, if deviations become too strong, AMG's efficiency may substantially deteriorate.

It is difficult to say precisely, to which type of problems (matrices) AMG can efficiently be generalized. The most essential requirement for a generalization is that we are able to define "strength of connectivity" in a physically correct way. Without this, AMG will not be able to perform any reasonable coarsening. Note that this definition was obvious in weakly diagonally dominant M-matrix cases. Moreover, operator-dependent interpolation should be so that algebraically smooth error is interpolated accurately enough. Again, while the definition of a reasonable interpolation is fairly obvious in the M-matrix case, this is not at all true in other cases.

In the following, we list a few typical situations for which the AMG processes need to be extended. None of these situations will cause a problem just by itself; whether AMG can efficiently be extended, or whether it may fail, strongly depends on the concrete situation, in particular, the physical origin of the respective new aspect. For the sake of brevity, we here cannot discuss these cases in more detail (except for the important case of coupled PDE systems which will be treated in Section 4).

- If matrix rows contain *(large) positive off-diagonal entries,* there is no simple and unique rule of how to deal with them in a physically correct way. In fact, what is correct and what not depends to a large extent on the origin of these positive entries; different situations require different procedures. In general, one has to be careful in defining strength of connectivity as well as the interpolation.
- For problems with *near-zero eigenvalues*, sufficiently accurate AMG interpolation may often not be computable based only on matrix information. The major difficulty is caused by the fact that the smaller an eigenvalue of a given problem, the more accurately the corresponding eigenvector needs interpolating. Clearly, unless such eigenvectors are close to being constants, the accuracy of interpolation for these eigenvectors will be limited without exploiting additional information.
- For (slightly) *indefinite problems,* AMG is often applicable without any change. However, whether this is really true depends not only on the number of negative eigenvalues but also on whether or not there are near-zero eigenvalues (cf. the previous case). Since it is difficult to easily detect "bad cases" by merely looking at matrix entries, the application of AMG on indefinite problems may or may not work.

- If matrices are *far from weakly diagonal dominant*, it strongly depends on the physical origin whether or not AMG can efficiently cope with such a case. Without any adaptation, AMG will most probably loose much of its efficiency or will even fail.
- In practice, matrices A are often *re-scaled* by applying diagonal matrices, D_1 and D_2,

(18) $$A \leftarrow D_1^{-1} A D_2^{-1} .$$

There are various reasons for doing this. In the context of AMG, however, such re-scalings need to be done with great care; at best they should be avoided altogether. In particular, re-scalings from the right (column scaling) may completely destroy matrix properties which are important for an efficient AMG treatment. For instance, while A may correspond to a nicely elliptic discretization, a re-scaling from the right may completely destroy this property, causing a fatal failure of AMG. Re-scalings from the left are usually somewhat less critical. But still, such re-scalings may negatively influence the quality of AMG's coarsening to quite some extent.

- Finally, the discretization of *coupled systems of PDEs* leads to matrices A which are usually far from weakly diagonally dominant M-matrices. The AMG treatment of such matrix problems requires special generalizations. Because of the importance of coupled PDE systems for practice, we will outline a generalized AMG strategy to deal with such applications in Section 4.

3.3.2 Increasing robustness

In order to increase the robustness of AMG approaches, it has become very popular to not use AMG-based solvers "stand-alone" but rather combine them with acceleration methods such as conjugate gradient (CG), BiCGstab or GMRES [37,40, 52]. Practical experience has clearly shown that AMG is a very good preconditioner, much better than standard (one-level) pre-conditioners. Heuristically, the major reason is due to the fact that AMG, in contrast to any one-level preconditioner, aims at the efficient reduction of *all* error components, short-range as well as long-range. However, although AMG tries to capture all relevant influences by proper coarsening and interpolation, its interpolation will hardly ever be optimal. It may well happen that error reduction is significantly less efficient for some very specific error components. This may cause a few eigenvalues of the AMG iteration matrix to be considerably closer to 1 than all the rest. If this happens, AMG's stand-alone convergence factor would be limited by the slow convergence of just a few exceptional error components whereas the majority of the error components is reduced very quickly. Acceleration by, for instance, conjugate gradient typically eliminates these particular components very effectively. 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 are avoided.

We finally want to mention that, although AMG generally gets along with much "weaker" smoothers than GMG, it often pays to add robustness to AMG by using somewhat "stronger" smoothers than just GS relaxation. Often used in practice are smoothers of ILU- or ILUT-type. Such smoothers will be extensively used, for example, in oil reservoir simulation (FIM and AIM approaches, see Section 5).

3.3.3 Exemplary coarsening process

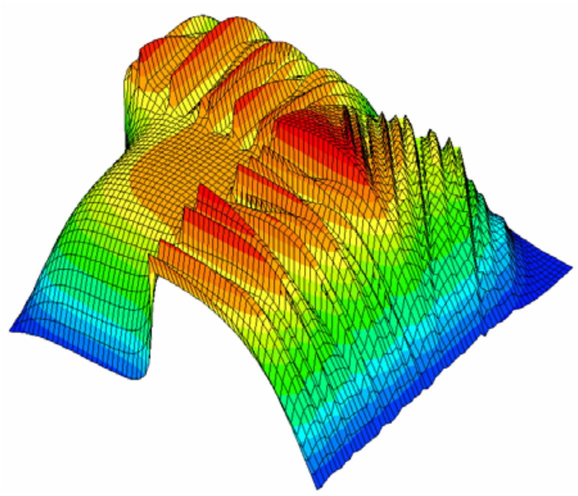
In this section we demonstrate AMG's flexibility in adjusting its coarsening process locally to the requirements of a given problem. The underlying problem is the differential problem

$$-(au_x)_x - (bu_y)_y + cu_{xy} = f(x, y) \tag{19}$$

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.

$a=1$ $b=10^3$ $c=0$	$a=1$ $b=1$ $c=2$
$a=1$ $b=1$ $c=0$	$a=10^3$ $b=1$ $c=0$

The diffusion part is discretized by the standard 5-point and the mixed derivative by the (left-oriented) 7-point stencils. 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. The left part of Figure 27 shows what a "smooth" error looks like on the finest level after having applied a few GS 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.



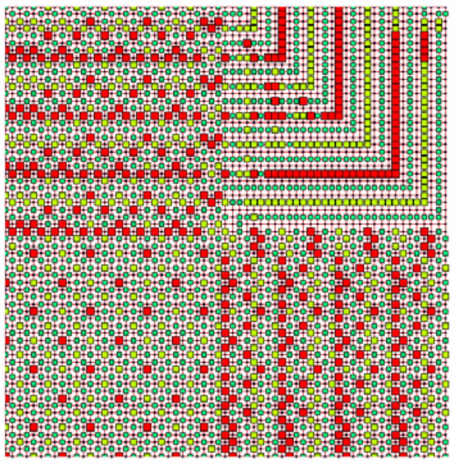


Figure 27: (left) "Smooth" error in case of Problem (19). (right) The finest and three consecutive levels created by the standard SAMG coarsening algorithm.

We have learned that such error can effectively be reduced by means of a coarser grid only 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, this is exactly how SAMG operates: First, its operator-based interpolation ensures the correct treatment of the discontinuities. Second, SAMG coarsening is in the direction of strong connectivity, that is, in the direction of smoothness.

To illustrate this further, the right part of Figure 27 depicts the finest and three consecutive grids created by using standard SAMG coarsening and interpolation. The smallest dots mark grid points which are contained *only* on the finest grid, the squares mark those points which are also contained on the coarser levels, the bigger the square, the longer the corresponding grid point stays in the coarsening process. The picture shows that coarsening is uniform in the lower left quarter where the problem is isotropic. In the other quarters, AMG adjusts itself to the different anisotropies by locally coarsening in the proper direction. For instance, in the lower right quarter, coarsening is in *x*-direction only. Since SAMG takes only *strong* connections in coarsening into account and since all connections in the *y*-direction are *weak*, the individual lines are coarsened *independently of each other*. Consequently, the coarsening of neighboring *x*-lines is not “synchronized”; it is actually a matter of “coincidence” where coarsening starts within each line. This has to be observed also in interpreting the coarsening pattern in the upper right quarter: within each diagonal line, coarsening is essentially in the direction of this line.

3.3.4 Performance and scalability

As a benchmark, we consider the application of SAMG to solve the pressure-correction equation which occurs as the most time-consuming part of the segregated solution approach to solve the Navier-Stokes equations. The discretization is based on finite-volumes. The concrete application is the industrial computation of the exterior flow over a complete Mercedes-Benz E-Class model (see left picture in Figure 28). The underlying mesh consists of several million cells and is highly complex.

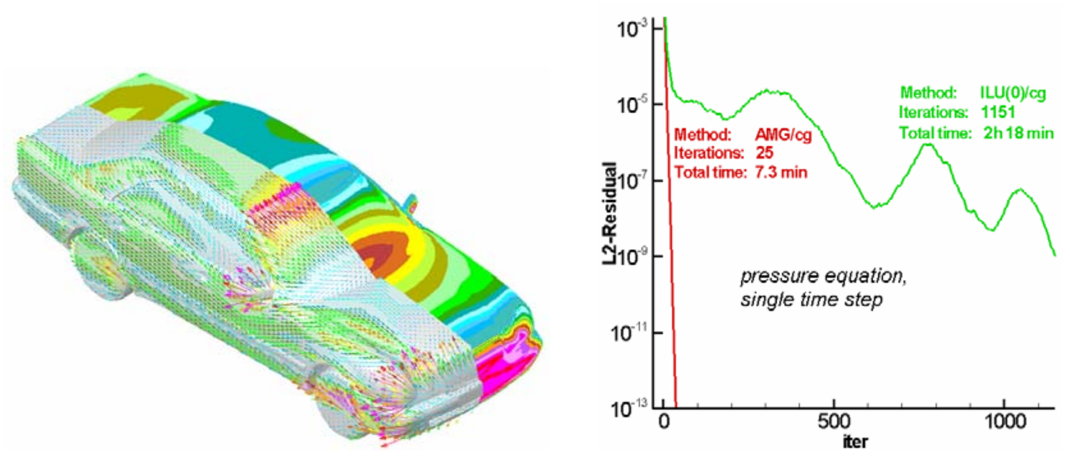


Figure 28: (left) Mercedes-Benz E-Class model. (right) Comparison of convergence histories between SAMG and a standard one-level solver. (Courtesy of Daimler-Benz and Computational Dynamics)

The right picture of Figure 28 refers to the solution of a single pressure-correction equation at one particular time step taken from a normal production run. It compares SAMG's convergence history with that one of a standard solver (ILU(0) pre-conditioned conjugate gradient, ILU/cg). In terms of total computational time, SAMG is about 19 times faster. This reflects the typical performance of SAMG in geometrically complex applications of the type and size considered here. Since, due to its scalability, SAMG's convergence is virtually independent of the problem size, the gain by employing SAMG grows further with increasing problem size.

Finally, we want to demonstrate SAMG’s scalability properties. For this purpose we consider a simple model problem which can easily be scaled up, the so-called Durlofsky flow problem. This corresponds to a square domain with a conductivity field as shown in Figure 29. The flow enters on the left side of the domain and exists on the right side with impervious boundaries at the top and the bottom.

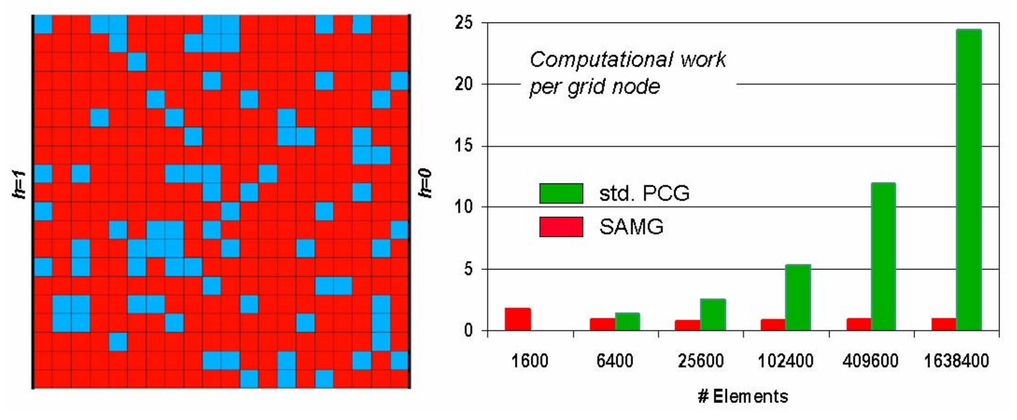


Figure 29: Durlofsky problem: (left) Conductivity pattern (red=1m/s; blue=10^{-6} m/s), basic 20x20 mesh. (right) Performance of SAMG compared to that of a standard one-level solver for increasing mesh size.

The problem is steady-state, characterized by a structured regular gridding and heterogeneous conductivity distribution. The parameter contrast refers to a power of six. The domain is initially discretized by a 20x20 square quadrilateral mesh representing the first refinement. To test SAMG's scalability, we perform a stepwise global refinement by recursively subdividing quadrilaterals into four equally sized quadrilaterals. For the sequence of meshes obtained this way, the bar chart in Figure 29 compares the performance of SAMG with that of a standard pre-conditioned conjugate gradient solver, PCG. The results clearly indicate the superiority of SAMG for this problem as well as its scalability property. In fact, the required number of iterations does not increase for SAMG if the problem enlarges whereas it increases significantly for the standard solver. For large problems, in the order of a million variables, SAMG is more than ten times faster than the standard solver.

4 AMG for coupled PDE systems

Classical AMG as outlined in the previous section has been designed for the solution of linear systems (1), obtained by discretizing *scalar* elliptic PDEs. Such matrices have characteristic properties - in simple cases they are weakly diagonally dominant M-matrices - which are exploited by the AMG algorithm. Unfortunately, matrices obtained by discretizing *coupled systems* of PDEs are usually far from satisfying these properties. Hence, if applied directly to such systems, classical AMG will fail or, at least, seriously deteriorate. Except for applications with very weak cross-couplings between different physical unknowns[16], the coarsening and interpolation rules of classical AMG are no longer appropriate. Without (at least) being able to distinguish between different unknowns, no efficient AMG processing can be expected.

Notation: Linear systems (1) corresponding to discretized coupled systems of PDEs will simply be called *coupled systems* as opposed to *scalar systems* corresponding to scalar PDEs.

[16] For historical reasons, in the context of AMG, the different physical functions defined by a coupled system of PDEs are usually called "unknowns".

Correspondingly, AMG approaches which are directly applicable to coupled systems are called *coupled AMG* as apposed to *scalar AMG* which is applicable only to scalar systems.

In practice, the solution of a coupled system of PDEs is very often reduced to the solution of a series of scalar problems. This is very popular, for example, in computational fluid dynamics ("pressure-correction methods") or oil reservoir simulation ("two-stage preconditioning", see Section 5.2.1). In such cases, scalar AMG approaches can immediately be applied to elliptic sub-problems. However, whether such a "decoupled" approach is efficient, depends on the application. In many cases it will be more efficient to treat coupled systems "directly" by a suitably generalized AMG approach. Before we will discuss procedures of how to obtain such coupled AMG approaches, we want to make a general remark:

Although AMG finally operates only on matrices, we must generally not ignore the physical background of a given problem. Generally speaking, the following three basic assumptions necessarily have to be satisfied for an AMG approach to be meaningful:

1. For any linear system to be solved by AMG, *a natural hierarchy (of resolution scales) has to **exist***. Otherwise, there cannot exist an efficient AMG approach either. However, since we are only considering PDEs, this request is naturally satisfied.
2. *AMG is able to **construct** a reasonable hierarchy.* This essentially means that AMG is able to correctly distinguish between weak and strong couplings. This was easy and straightforward for M-matrices. In general, however, this is not straightforward at all.
3. *AMG is able to **exploit** the hierarchy.* This means that AMG is able to use coarse-level approximations to effectively correct fine-level ones. This is essentially a requirement regarding the "accuracy" of AMG's interpolation.

Unfortunately, there is no simple answer on how to satisfy these assumptions in general and there is no straightforward extension of classical AMG resulting in a black-box which can efficiently solve any coupled system of PDEs. In fact, concrete approaches need to be specifically designed for certain classes of applications and black-box usage can only refer to such classes.

Already in the early work [36], but more systematically during the last years [17,18,23,47], extensions of the classical AMG have been considered. In the following, the goal is not to present a final algorithm, but rather to describe a general framework which is flexible enough to exploit additional (including user-provided) information and which can easily be adjusted to specific requirements of a given problem class. For demonstration, we mention some typical examples.

4.1 General idea: the primary matrix

The basic idea to define coarsening processes under very general assumptions relies on the introduction of some auxiliary (sparse) control matrix, the so-called *primary matrix,* generally denoted by P: Rather than defining the strength of connectivity between variables directly via the entries of the given matrix A, it is defined via the entries of P. Formally, this provides a very general framework allowing the method to be adjusted to many different situations[17].

[17] We point out that, in most practical cases, a primary matrix does not need to be explicitly assembled and stored. The concept of primary matrices is rather used for a simple description of the approaches.

While primarily introduced for the treatment of coupled systems of PDEs, the concept of primary matrices makes sense also in case of scalar PDEs. Classical AMG as described in the previous section corresponds to selecting *P=A*, that is, *P* represents the connectivity structure between variables as provided directly by the given matrix. However, in non-standard scalar applications, one may also define *P* differently, for instance, based on geometric distances of the variables (if known), or on alternate (for instance, less accurate but simpler) discretization schemes. One may also define rows of *P* by suitably combining certain neighboring rows of the matrix *A* ("local elimination"). This provides, for instance, a natural way to analyze the influence of "critical" (e.g., large positive) off-diagonal matrix entries and treat them correctly in terms of strength of connectivity.

The main purpose of a primary matrix is to allow for a reasonable coarsening process; in many situations it may also be useful in constructing interpolation. That is, coefficients of interpolation are defined based on the matrix entries of *P* rather than those of *A*. Obviously, one can imagine many possible algorithmic combinations and which one is best, strongly depends on certain characteristics of a given application.

In the following, we concentrate on the "direct" treatment of coupled systems, outlining two different approaches, *unknown-based* and *point-based*.

4.2 Unknown-based approaches

Let us consider a coupled system for *nf* physical unknowns such as the pressure, the saturation of a particular species, or a velocity component. We require that, besides the linear system (1) itself, information is available allowing to distinguish between the different unknowns. Since this is the only additional information required, the resulting AMG approaches are called "unknown-based". The idea is to perform coarsening and interpolation separately for each unknown[18]; the basis AMG approach applied to each of the unknowns is analogous to the scalar AMG approach. Hence, unknown-based AMG approaches are both simple and powerful for many classes of applications, at least if the cross-couplings between unknowns do not exceed a certain strength. Typical application classes are given by *diffusion problems* or *linear elasticity*.

The general form of an unknown-based approach is most easily described if we assume all variables to be ordered unknown-wise, so that the matrix *A* takes the form

$$A = \begin{pmatrix} A_{[1,1]} & \cdots & A_{[1,nf]} \\ \vdots & \ddots & \vdots \\ A_{[nf,1]} & \cdots & A_{[nf,nf]} \end{pmatrix}.$$

Them, a general primary matrix has the form

$$\begin{pmatrix} P_1 & & \\ & \ddots & \\ & & P_{nf} \end{pmatrix}.$$

[18] This does *not* mean that all unknowns are treated independently of each other. In fact, on each level of the multigrid hierarchy the Galerkin operator re-introduces cross-couplings between different unknowns.

The sparse block sub-matrices P_i are assumed to have the same dimension as $A_{[i,i]}$ and to be suitable for scalar AMG. Hence, the connectivity structure reflected by the auxiliary matrix P_i can be used to coarsen the i-th unknown. Clearly, for this to make sense, P_i should reflect the physical connectivity between variables corresponding to the i-th unknown reasonably well. For instance, for all diagonal blocks $A_{[i,i]}$ which are appropriate for scalar AMG (eg, which are close to being weakly diagonally dominant M-matrices), a standard choice would be $P_i=A_{[i,i]}$. However, analogous to the scalar case outlined above, various other choices are possible.

In the unknown-based approaches, interpolation is kept separate for the different unknowns, that is, variables corresponding to the i-th unknown, say, are interpolated from variables of the same type only. Concrete weights may, for example, be based on the entries of P_i, $A_{[i,i]}$ or on geometric distances (if available).

Note that unknown-based approaches have no analog in GMG where coarsening is always on the basis of grids and, hence, all physical functions live on the same hierarchy.

4.3 Point-based approaches

In contrast to unknown-based approaches, point-based ones additionally need to distinguish between different points. More precisely, point-based approaches need to know which variables are sitting at the same physical point of a given grid (they generally do *not* need to know the *location* of grid points, though). For this to make sense, we require all unknowns to be defined at the same grid (i.e., we do not consider “staggered” grids). Note, however, that not all unknowns are required to be defined everywhere, allowing for adaptive strategies such as those realized in the AIM approach in reservoir simulation (see Section 5.3).

For an outline of point-based approaches, we assume all variables to be numbered point-wise so that the matrix A takes the form

$$A=\begin{pmatrix} A_{(1,1)} & \cdots & A_{(1,np)} \\ \vdots & \ddots & \vdots \\ A_{(np,1)} & \cdots & A_{(np,np)} \end{pmatrix}$$

where np denotes the number of grid points. In contrast to unknown-based approaches, point-based ones are controlled by a single primary (sparse) $np \times np$ matrix P which is defined on the level of “points” rather than variables, and all unknowns are coarsened based on the same P. That is, all unknowns are coarsened simultaneously and live on the same coarse grids. Clearly, for this to make sense, the connectivity structure defined by P should represent the connectivity of *all* unknowns in the coupled system reasonably well.

In the standard case that all unknowns are defined at all grid points, point-based coarsening with the primary matrix P can actually be regarded as a special unknown-based coarsening with $P_i \equiv P$. The fact that point-based approaches produce only a single hierarchy of grids makes them particularly flexible in taking care of strong cross-couplings between unknowns. For instance, smoothing can be performed in a block sense (e.g. block Gauss-Seidel or block ILU). If required, interpolation can also be performed block-wise with blocks being induced by the system’s point block coupling (see below). Hence, in applications with strong couplings between different unknowns, point-based AMG may be advantageous compared to unknown-based AMG.

There are many possibilities of choosing a primary matrix *P*, some of them being sketched in the following. Often, the connectivity inherent to one of the given PDE system's unknowns, *k* say, can be regarded as being representative also for the other unknowns of the coupled system. In such cases, provided that the *k*-th unknown is defined at each point of the grid, a possible choice is

$$P = A_{[k,k]}\,.$$

In reservoir modeling, for instance, we will use the pressure block for that purpose (see Sections 5.2.2 and 5.3). In other applications, depending on certain characteristics of the class of PDEs under consideration, potential choices are $P=(p_{ij})$ with

$$p_{ij} = -\| A_{(i,j)} \| \quad \text{or} \quad p_{ij} = -1/\,dist(i,j)^2$$

where $i \neq j$ and, for instance,

$$p_{ii} = -\sum\nolimits_{j \neq i} p_{ij}\;.$$

While the first choice leads to what is sometimes called „block approach“, the second choice (assuming point coordinates to be available) is closely related to geometric coarsening[19]. One can also imagine applications for which *P* can be defined based on some natural physical quantity for which there is no obvious equation contained in the given system.

Once a primary matrix has been selected for coarsening, there are still many ways to define interpolation. In particular, interpolation may be different for each physical unknown (e.g., based on the original matrix blocks $A_{[i,i]}$), it may be the same for each unknown (e.g., based on the primary matrix *P* or on coordinates), or it may be defined based on the point-wise block couplings.

4.4 Software package SAMG

Formally, by combining the various coarsening and interpolation possibilities outlined above, one obtains a general framework to define concrete AMG approaches. It seems clear, however, that there is no single approach which will work satisfactorily for *all* systems of PDEs. Instead, different approaches will be required for different classes of applications.

During the last years, a software library, SAMG [30,47], has been developed at Fraunhofer Institute SCAI which realizes most of the ideas introduced above. Hence, SAMG is not just a solver but rather a very flexible multilevel framework which can be adapted to specific requirements of various problem classes. Primary matrices are either defined internally to SAMG (i.e. are constructed automatically) or they are user-provided. The latter option makes particular sense in applications where SAMG cannot construct a reasonable primary matrix automatically, based solely on algebraic information contained in the given linear system. However, in many such cases, a user may still be able to define a reasonable matrix himself, based on the underlying physics.

[19] Note that the use of distances requires the additional definition of a non-zero pattern in order for the primary matrix to be sparse. Clearly, this pattern should be related to the connectivity patterns of the matrix itself.

Over the years, SAMG has been applied to various types of coupled systems of PDEs from such diverse areas as fluid flow, structural mechanics, oil reservoir and ground water simulation, casting and molding, process and device simulation in solid state physics, electro-chemistry, and circuit simulation. In particular, in oil reservoir simulation, SAMG has become a well-established tool for various software providers as well as major oil companies. Regarding some more discussion in various areas, see, for example, [17,18,19,22,45,46]. For further information, see also http://www.scai.fraunhofer.de/samg.

5 AMG in reservoir simulation

Modern reservoir simulation faces increasingly complex physical models and highly resolved, unstructured grids. Both trends result in ever growing and more difficult to solve matrix equations. Especially compositional models for heterogeneous or fractured reservoirs provide a considerable challenge to solver efficiency, in particular, if fully implicit methods (FIM) are used.

The core of the computation at each time step is governed by the successive solution of coupled linear (Jacobian) systems representing the behavior of different physical entities sharing the same discretization element. Generally, the underlying “FIM matrices” are highly nonsymmetric and indefinite. Individual matrix coefficients are typically strongly anisotropic and/or discontinuous. This is due to geological settings as well as due to numerical effects, among them different vertical and horizontal permeabilities, high porosity contrasts between adjacent grid blocks or other properties with drastic variations (typically by several orders of magnitude). In addition, unstructured gridding, local grid refinement, fault modeling, large variations in grid spacing and adverse grid block aspect ratios (ratio of lateral to vertical block dimensions) cause additional distortion to the FIM matrices making them even more difficult to treat numerically. Finally, well equations which are fully coupled to the system of equations relate grid blocks that otherwise are not geometrically connected. The condition number and degree of coupling of these systems may be subject to dramatic changes due to abrupt flow variations induced by the high-heterogeneity and complex well operations during the simulation process.

Advanced AMG solvers offer an efficient technology for solving linear systems that are “sufficiently elliptic”. Since, when applicable, AMG solvers are both scalable and easy to use, interest in incorporating AMG into industrial oil reservoir simulation codes has been steadily increasing during the last years. In this section, we summarize overall strategies to numerically tackle reservoir simulation and demonstrate the use of AMG in all cases:

1. Classical *IMPES* and modern *streamline approaches* implicitly solve only for the pressure. Hence, in principle, scalar AMG can immediately be applied.
2. In contrast to this, *fully implicit modeling (FIM)* requires solving coupled systems.
   a. The most popular way to do this is via a two-stage preconditioning approach which numerically decouples the treatment of pressure and non-pressure variables. As a result, the computationally most expensive part consists of solving pressure systems for which, again, scalar AMG seems most natural.
   b. As an important alternative, we introduce a coupled AMG approach to “directly” solve FIM systems. We demonstrate that AMG can be applied to efficiently deal with such systems although they are not of elliptic type. In fact, FIM systems are essentially of mixed elliptic-hyperbolic character.

3. To reduce computational cost, *adaptive implicit modeling (AIM)* has been developed, offering a good compromise between computational speed, memory requirements and accuracy. However, as for FIM, the mixed elliptic-hyperbolic character of AIM matrices aggravates the development of efficient linear solvers. We demonstrate how coupled AMG has efficiently been used in real-life simulations.

All results shown in this section have been obtained with the SAMG package [18,43,47] as part of cooperations with StreamSim Technologies, SMT Alps and the University of Texas at Austin.

5.1 IMPES and streamline approach

Streamline-based flow simulation is an effective and complementary technology to more traditional flow modeling approaches such as finite differences (see Thiele [48] and the references given therein). This is because streamline-based flow simulation is particularly effective in solving large, geologically complex and heterogeneous systems, where fluid flow is dictated by well positions and rates, rock properties (permeability, porosity, and fault distributions), fluid mobility (phase relative permeabilities and viscosities), gravity, and voidage replacement ratios close to one. These are the class of problems more traditional modeling techniques have difficulties with. More diffusive mechanisms, such as capillary pressure effects and expansion-dominated flow, on the other hand, are not modeled efficiently and accurately by streamlines.

Streamline-based simulation is an IMPES-type formulation and therefore involves the implicit solution for a single variable only, namely, pressure on the global 3D scale (see Figure 30). Hence, *scalar* AMG is well suited for solving the pressure equation. The high efficiency of AMG in solving the pressure equations makes streamline-based simulations computationally highly efficient, one important reason making this approach so attractive (Figure 31).

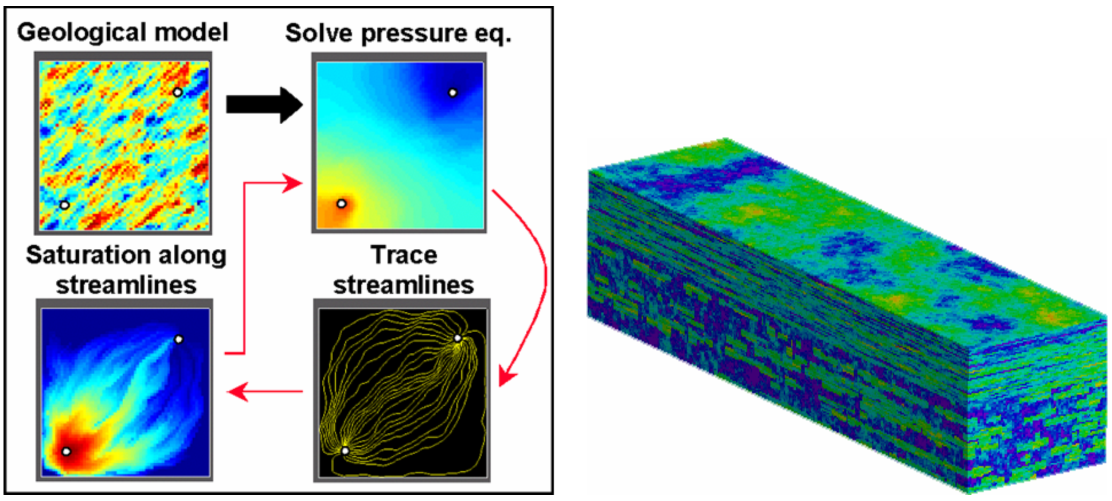


Figure 30: (left) Schematic view of the streamline approach. (right) SPE 10 upscaling benchmark model, 1.12 million cells.

Some characteristics of the pressure equations may cause difficulties for a linear solver, namely, the *variation of permeabilities* (leading to strongly discontinuous and anisotropic coefficients) as well as the presence of *well equations*. While strongly varying permeabilities may substantially slow down the convergence speed of any standard linear solver, they do not provide any particular difficulty for AMG. It is the dynamic and automatic features of AMG which allow a flexible adaptation to any variation of coefficients.

On the other hand, the non-PDE character of well equations disturbs the locality and ellipticity of a system to some extent. Without providing any extra information to AMG, such equations are treated just like any other equation of the discretized PDE, causing the quality of the well equation's coarse-level correction to be limited. Whether this will slow down the overall convergence, and to which extent, depends on the number and type of wells. While often no special treatment of the well equations is needed, in more complex situations a reasonable "well decoupling" may pay. The idea is to apply AMG only to the reservoir part and couple the well equations differently. To explain three straightforward approaches, we re-write the linear system (1) as

(20) $$\begin{pmatrix} A_{rr} & A_{rw} \\ A_{wr} & A_{ww} \end{pmatrix}\begin{pmatrix} u_r \\ u_w \end{pmatrix} = \begin{pmatrix} f_r \\ f_w \end{pmatrix}$$

where the indices r and w mark the reservoir and well quantities, respectively.

Coupling via the finest-level smoother. This means that all "critical" variables (e.g. those corresponding to wells) are excluded from any coarsening process. That is, they are simply forced to stay on the finest level, coupled to the remaining variables only via the finest-level smoothing process. This is the simplest procedure which should work if the number of critical variables is small and if the coupling between critical and non-critical variables is not too strong. Clearly, the finest level smoother should be sufficiently strong.

Schwarz alternating method (with overlap). This essentially means that, starting with some first approximation, $u_w^{(0)}$, we solve the two sub-systems

$$A_{rr}u_r^{(i+1)} = f_r - A_{rw}u_w^{(i)} \quad \text{and} \quad A_{ww}u_w^{(i+1)} = f_w - A_{wr}u_r^{(i+1)}$$

one after the other for i=0,1,2,...... While the reservoir part is approximately solved by applying one or more AMG cycles, the well part is solved by a block solver (eg, sparse Gauss elimination). Since the number of wells is limited, the block solve should be inexpensive compared to the solution of the reservoir part. This type of iterative process is called "Schwarz alternating method". To speed up the "outer convergence", the block solve should include some "overlap" variables.

Approximative Schur complement approach. The Schur complement approach essentially corresponds to a block elimination of (20):

$$\begin{pmatrix} \bar{A}_{rr} & 0 \\ A_{wr} & A_{ww} \end{pmatrix}\begin{pmatrix} u_r \\ u_w \end{pmatrix} = \begin{pmatrix} \bar{f}_r \\ f_w \end{pmatrix} \text{ with } \begin{cases} \bar{A}_{rr} = A_{rr} - A_{rw}A_{ww}^{-1}A_{wr} \\ \bar{f}_{rr} = f_{rr} - A_{rw}A_{ww}^{-1}f_w \end{cases}.$$

Assuming the number of wells to be not too large, the most costly part in solving this block tridiagonal system is the solution of

(21) $$\bar{A}_{rr}u_r = \bar{f}_r.$$

In principle, $\bar{A}_{rr}$ can explicitly be evaluated. Unfortunately, due to the algebraic manipulation of the block elimination process (disturbing the ellipticity), $\bar{A}_{rr}$ has to be expected to be "less favorable" for an AMG solution than A_{rr}. Hence, we do not solve (21) directly, but rather by an iterative "pre-conditioning" process of the form

$$(22) \qquad u_r^{(i+1)} = u_r^{(i)} + M^{-1} r^{(i)} \quad \text{with} \quad r^{(i)} = \bar{f}_r - \bar{A}_{rr} u_r^{(i)} .$$

The „pre-conditioner", *M*, is some reasonable approximation to $\bar{A}_{rr}$ which is more suitable for an AMG solution[20]. In many cases, $M = A_{rr}$ (possibly scaled so that the row sums of *M* equal those of $\bar{A}_{rr}$) is a reasonable choice. In practice, $M^{-1} r^{(i)}$ in (22) is not evaluated explicitly but rather stands for the application of one or more AMG cycles to the system with matrix *M* and right hand side $r^{(i)}$.

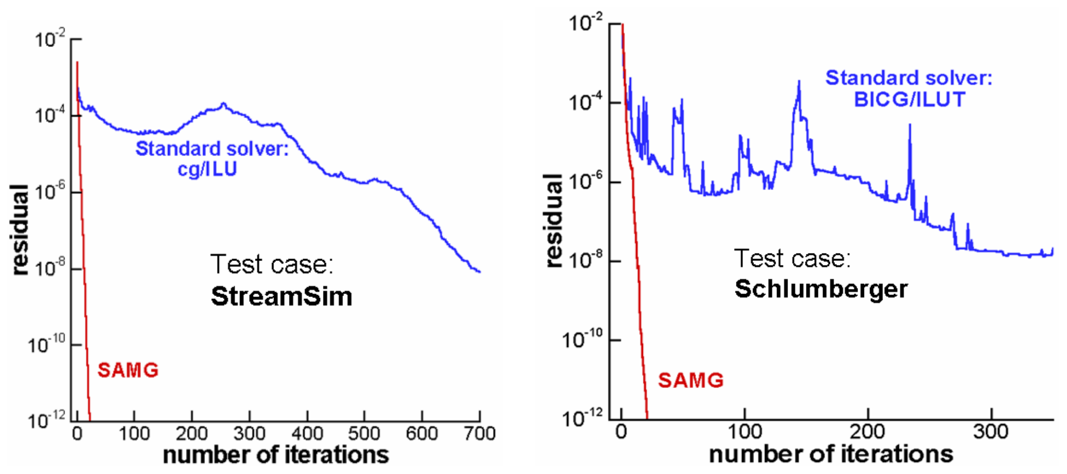


Figure 31: Comparison of SAMG's convergence history with that of a standard one-level method in solving the pressure equation in the streamline approach. The figures correspond to a symmetric (left) and non-symmetric (right) test case.

5.2 Fully implicit modeling (FIM)

To demonstrate the principle of how to apply AMG, let us consider the black-oil model,

$$q_a = -k \frac{k_a(S_a)}{\mu_a} (\nabla p_a \text{-} \rho_a g \nabla D) \qquad \text{Darcy's law}$$

$$\phi \frac{\partial S_a}{\partial t} + \nabla \cdot q_a + Q_a = 0 \qquad \text{phase conservation law}$$

$$S_o + S_w + S_g = 1 \qquad \text{phase-constraint equation}$$

$$\left.\begin{aligned} p_o - p_w &= p_{cow}(S_o) \\ p_g - p_o &= p_{cgo}(S_o) \end{aligned}\right\} \quad \text{capillary pressure relations}$$

[20] Note that, if M equals $\bar{A}_{rr}$, one iteration of (22) yields the exact solution of $\bar{A}_{rr} u_r = \bar{f}_r$.

supplemented by *well equations* and *boundary conditions*. The letters *S*, *p*, *q* and *k* stand for saturation, pressure, velocity and permeabilities, respectively. The index *a* stands for the phases oil (*o*), water (*w*) and gas (*g*).

The discretization by finite differences results in coupled linear systems of the form

$$(23) \qquad \begin{pmatrix} A_{ss} & A_{sp} \\ A_{ps} & A_{pp} \end{pmatrix} \begin{pmatrix} u_s \\ u_p \end{pmatrix} = \begin{pmatrix} f_s \\ f_p \end{pmatrix}$$

where the indices p and s denote the pressure and saturation quantities, respectively. For the question of how to efficiently apply AMG to solve coupled systems (23), it is important to note that the block A_{pp} has the character of a purely elliptic problem, whereas the block A_{ss} behaves essentially hyperbolic. This motivates two different (but in a sense related) possibilities of using AMG, namely, the "two-stage preconditioning" and the "direct" approach. We will outline both approaches in the following. For a more detailed discussion as well as an extensive list of references, we refer to [19,46].

5.2.1 Two-stage preconditioning

Two-stage preconditioners are based on the idea that coupled system solutions (23) are mainly determined by the solution of their elliptic component (i.e., pressure). The procedure consists of "extracting" and accurately solving pressure subsystems. Residuals associated with this intermediate solution are corrected with an additional step that recovers part of the global information contained in the original system. Two-stage preconditioners have been investigated by, for instance, Behie, Vinsome, Forsyth, Wallis, Klie and Dawson (see [46] for a list of references).

Generally, to help convergence when iterating between the two stages, the procedure is not applied directly to (23). Instead, some "decoupling" is performed aiming at weakening the existing coupling between pressure and non-pressure blocks. That is, based on some reasonable decoupling operators, D_1 and D_2, the original system (23) is transformed into

$$(24) \qquad \tilde{A}\tilde{u} = \tilde{f} \quad \text{with} \quad \tilde{A} = D_1^{-1} A D_2^{-1} = \begin{pmatrix} \tilde{A}_{ss} & \tilde{A}_{sp} \\ \tilde{A}_{ps} & \tilde{A}_{pp} \end{pmatrix}.$$

Clearly, a "full" decoupling, forcing $\tilde{A}_{sp}$ or $\tilde{A}_{ps}$ (or even both) to vanish, is prohibitive. For reasons of numerical efficiency, the decoupling process should be computationally inexpensive but still provide a good preconditioning effect on the original system. One of the most popular decoupling strategies is known as quasi-IMPES which sets $D_2=I$ and selects D_1 so that the pressure at a grid point is decoupled from the saturations at the same point. Another approach also sets $D_2=I$ but selects D_1 so that pressure and saturation values sitting at the same grid point are completely decoupled from each other ("point-block diagonal scaling"). For more details on these decoupling operators, see, for instance, [46].

A two-stage preconditioner for solving (24) consists of iteratively applying the following two steps, describing how a new approximation $\tilde{u}^{(1)}$ is computed from an old one, $\tilde{u}^{(0)}$:

(25) $\tilde{u}^{(1/2)} = \tilde{u}^{(0)} + \delta^{(0)}, \quad \delta^{(0)} = M_1^{-1} r^{(0)}$ and $\tilde{u}^{(1)} = \tilde{u}^{(1/2)} + \delta^{(1/2)}, \quad \delta^{(1/2)} = M_2^{-1} r^{(1/2)}$

with the residuals $r^{(i)} = \tilde{f} - \tilde{A}\tilde{u}^{(i)}$. That is, the approach is analogous to (22), except that we now select two different preconditioners, M_1 and M_2. Both of them should approximate $\tilde{A}$ in some sense but we will choose them so that, roughly, the first pre-conditioning step improves the current pressure approximation whereas the second step updates the saturation. More specifically, one iteration step, computing $\tilde{u}^{(1)}$ from $\tilde{u}^{(0)}$, runs as follows[21]:

1. Compute $\quad r^{(0)} = \begin{pmatrix} r_s^{(0)} \\ r_p^{(0)} \end{pmatrix} = \tilde{f} - \tilde{A}\tilde{u}^{(0)}$

2. Solve $\quad \delta^{(0)} = \begin{pmatrix} \delta_s^{(0)} \\ \delta_p^{(0)} \end{pmatrix} = \begin{pmatrix} 0 \\ \tilde{A}_{pp}^{-1} \tilde{r}_p^{(0)} \end{pmatrix}$ (1st preconditioning step)

3. Compute $\quad r^{(1/2)} = \tilde{f} - \tilde{A}\tilde{u}^{(1/2)} = \tilde{f} - \tilde{A}(\tilde{u}^{(0)} + \delta^{(0)}) = r^{(0)} - \tilde{A}\delta^{(0)}$

4. Solve $\quad \delta^{(1/2)} = M_2^{-1} r^{(1/2)}$ (2nd preconditioning step)

5. Compute $\quad \tilde{u}^{(1)} = \tilde{u}^{(0)} + \delta^{(0)} + \delta^{(1/2)}$

Various aspects need to be balanced to ensure robustness and highest efficiency. Although the two-stage preconditioning approach is widely considered as the currently most successful approach for FIM simulations, there are still several open questions some of which are mentioned in the following. In general, the performance of the above two-stage process is influenced by three factors:

The efficiency of the 1st preconditioning step. The 1st preconditioning step requires the (approximate) solution of the (transformed) pressure equation,

$$\tilde{A}_{pp}\delta_p^{(0)} = r_p^{(0)}.$$

Since this is related to the elliptic part of the original system (23), scalar AMG seems most promising as a solver. However, two aspects have to be taken into account.

- The main purpose of the decoupling process mentioned further above is to help the "outer" convergence. Unfortunately, the decoupling has an unwanted side effect: it influences the elliptic properties of the pressure block to some extent. As a consequence, the transformed matrix, $\tilde{A}_{pp}$, may be "less favorable" than A_{pp} for an efficient AMG-treatment. Several attempts have been made to minimize this unwanted side effect by choosing the decoupling carefully. However, a robust, final solution to this problem seems not yet known.

[21] Note that the description (25) is somewhat sloppy: M_1 approximates only $\tilde{A}_{pp}$ and, hence, is not invertible in the sense of (25). However, it is obvious how to interpret this.

- Second, well equations may require careful attention. To obtain highest efficiency, well equations should be appropriately integrated into the overall solver process, for instance, by some kind of well-decoupling strategy as described in Section 5.1.

The efficiency of the 2nd preconditioning step. The purpose of the 2nd preconditioning step is to update the non-pressure variables to a sufficient extent. Since the non-pressure subsystem behaves essentially hyperbolic (the variables have a directional dependence), this is often much simpler to accomplish than solving the pressure system. In practice, M_2 is typically selected as an ILU-type factorization of $\tilde{A}$. In simple applications, it may be sufficient to restrict the factorization just to the non-pressure variables or to use an even simpler method such as a relaxation method. In general, the optimal choice of the preconditioner M_2 is an open question.

The interplay between the two preconditioning steps. The number of AMG cycles and the number of ILU steps performed in the first and second preconditioning step, respectively, is crucial for the overall efficiency. However, it is not obvious how to optimally control these “inner” iterations, in particular, when to terminate them.

5.2.2 Direct approach

To treat FIM systems (23) “directly” by coupled AMG, we may consider the general approaches introduced in Section 4. However, in composing a concrete approach, one has to take into account that the FIM matrices are not purely elliptic but rather of mixed elliptic-hyperbolic type.

It is important to recall that it is mainly the elliptic part which needs accelerating by hierarchical processing. Hence, a possible point-based AMG approach should be “driven” by the elliptic pressure relations (i.e. based on a primary matrix $P=A_{pp}$, see Section 4.3), combined with a strong smoother which adds some properties as an approximate solver for the hyperbolic part. Often smoothers of type block Gauss-Seidel, ILU(0) or block ILU(0) work sufficiently well. However, more robust and flexible in this context is ILUT [38], an ILU-version which is controlled by two parameters, generally known as *lfil* and *droptol*:

- *lfil* - defines the maximum level of absolute fill-in;
- *droptol* - defines a threshold for dropping small couplings during elimination.

These parameters can be used to flexibly adjust the ILUT-process to the requirements. However, as with any ILU-type approach, ILUT’s efficiency depends to some extent on the ordering of variables of the matrix equation.

Numerical experiments have shown that, based on these ingredients, robust coupled AMG solvers can be obtained if, in addition, “critical” equations are treated with care. In particular, matrix rows strongly violating diagonal dominance (resulting, for instance, from well equations), may cause convergence problems. As before, decoupling processes similar to those described in Section 5.1 might be installed to exclude critical equations from coarsening.

Remark: Similar to the two-stage preconditioning approach described in Section 5.2.1, the direct AMG approach is motivated by the fact that solutions of (23) are mainly determined by their elliptic component. Moreover, the two-stage and the direct approach are related if one regards the second stage of the two-stage preconditioning approach as a smoothing process (restricted to the finest level only): By forcing all non-pressure variables to stay on the finest level, the direct AMG approach becomes very similar to a two-stage preconditioning approach, except that no algebraic manipulation is performed on the pressure block A_{pp}.

5.2.3 Some comparisons

In this section we compare both AMG approaches to solve FIM problems, the two-stage preconditioning ("AMG-2stage") and the direct approach ("AMG-direct"). All results have been obtained with the IPARS simulator [55]. For the direct approach, we use the simplest possible smoother, namely, plain Gauss-Seidel relaxation. The two-stage preconditioning uses the quasi-IMPES decoupling and line SOR (LSOR) as 2nd preconditioning step.

We consider five test cases with size and type as follows:

1. 60x220 (1st slice of the non-fluvial reservoir, Tarbert formation);
2. 60x220 (1st slice of the fluvial reservoir, Upper Ness formation);
3. 17x12x44 (upscaled version of the original data, including Tarbert and Upper Ness formations);
4. 25x15x55 (Upper Ness formation, upscaled version);
5. 50x30x110 (Upper Ness formation, upscaled version).

For all cases we define one water injection well at the center of the reservoir, and four production wells at the four corners. All are bottomhole pressure specified. For each of the considered problem sizes, we performed tests for an oil-water and a black-oil system. For the black-oil system, the PVT and saturation-dependent curves data were adapted from the SPE 9th Comparative Project.

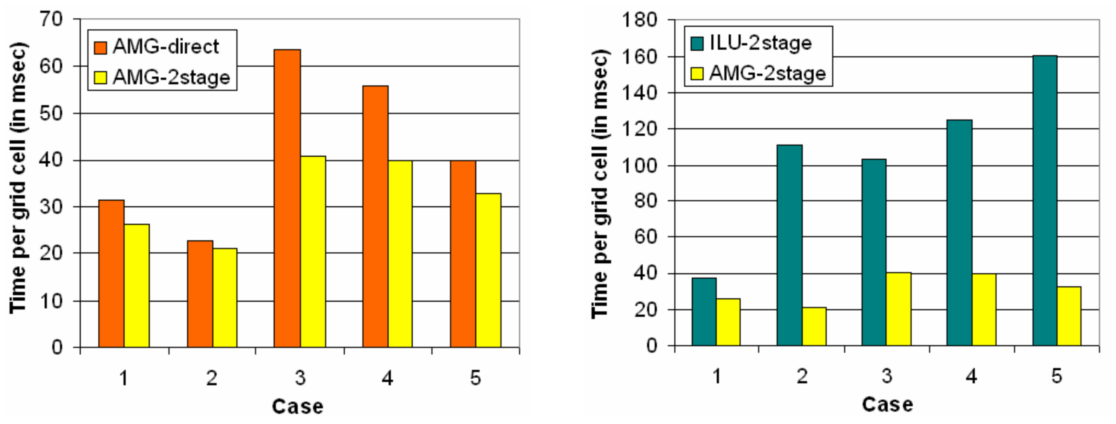


Figure 32: Oil-water simulations: (left) Comparing AMG solver strategies. (right) Comparing AMG with multilevel ILU [3] for solving the pressure equation in a two-stage preconditioning approach.

Figure 32 summarizes timings for oil-water simulations. Simulations were carried out for 2000 days except for Case 5 which was simulated for 500 days only. We first observe that, for all test cases and solver approaches considered here, AMG is efficient as a basic linear solver. According to the left figure, AMG-2stage seems to be slightly more efficient than AMG-direct, at least for the cases considered here. The right figure clearly shows the substantial gain in performance in switching from multilevel ILU to AMG. Figure 33 shows similar timings for the black-oil case. Again, there appears to be no striking difference between AMG-direct and AMG-2stage, with a slight preference for the latter.

These results seem to indicate that there is no real advantage of AMG-direct as compared to AMG-2stage. However, one should observe that many factors may affect the algebraic properties of the FIM matrices and, through this, the solver comparison. For instance, a serious violation of diagonal dominance - for instance, caused by well equations - may have a strong impact on solver performance. More testing is certainly needed to draw some well-founded and final conclusion.

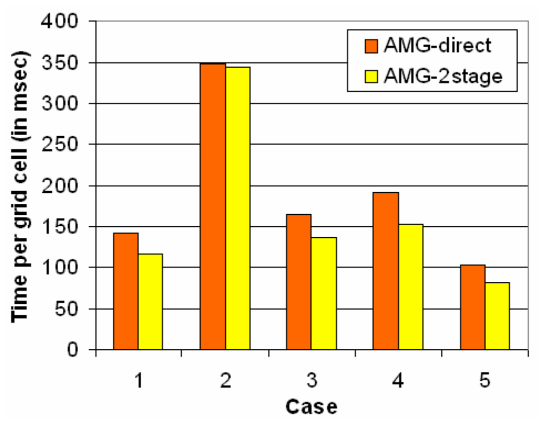


Figure 33: Black-oil simulations comparing AMG solver strategies.

For the specific cases tested here it is actually not surprising that AMG-direct is slightly more expensive than AMG-2stage. This is because AMG-2stage works efficiently with simplest algorithmic components such as plain Gauss-Seidel relaxation for smoothing and LSOR as 2nd preconditioner, M_2. In such cases it is plausible that AMG-direct is not more efficient than AMG-2stage. However, we want to recall that the performance of AMG-2stage is strongly tied to the decoupling process, the pressure solution and the 2nd preconditioner, M_2. In general, stronger and more costly preconditioners M_2 as well as more robust smoothers will be required. Most importantly, however, algebraic manipulations such as those done in the decoupling process, may negatively influence the ellipticity of the pressure block to an extent which causes AMG to deteriorate significantly. For instance, depending on the situation, row sums may get increasingly negative and, eventually, $\tilde{A}_{pp}$ may become indefinite. The risk of a substantial performance drop can certainly be reduced by a proper choice of the decoupling. However, a rigorous analysis, taking all aspects into account, seems fairly difficult.[22]

In contrast to AMG-2stage, AMG-direct attempts to get along without any algebraic manipulations except possibly for well-decoupling etc. This makes the direct approach particularly interesting and promising regarding its robustness. In fact, AMG-direct is applied to the unmodified system (23), with the coarsening process being driven by the fully elliptic pressure block, A_{pp}. Convergence of the direct AMG approach is expected to be less dependent on the strength of cross-couplings between pressure and saturation. This is due to its hierarchical nature, combined with its potential in employing strong block-wise smoothing and interpolation.

[22] We recall that re-scalings may substantially influence AMG's performance (see (18) in Section 3.3.1).

5.3 Adaptive implicit modeling (AIM)

In contrast to the fully implicit modeling approach as discussed in the previous section, adaptive implicit modeling (AIM, [49]) permits both explicit and implicit discretizations simultaneously within a model. The goal is to restrict the expensive fully implicit formulation to that part of a model that really requires implicitness (e.g. at perforated grid blocks or in areas with large variable changes), leaving the other parts in the much less expensive IMPES mode. At any given time step, pressure will be calculated implicitly *everywhere* in the reservoir model, but other variables, such as phase saturations and concentrations will be implicit in selected grid blocks only and explicit elsewhere.

As a result, the user gains a maximum in flexibility, but the linear solver (which is naturally applied only to the implicitly coupled variables) encounters matrices with highly irregular and dynamically changing structures. This is because the division into implicit and IMPES blocks typically varies from one simulation step to the next, based on some automatic control mechanism.

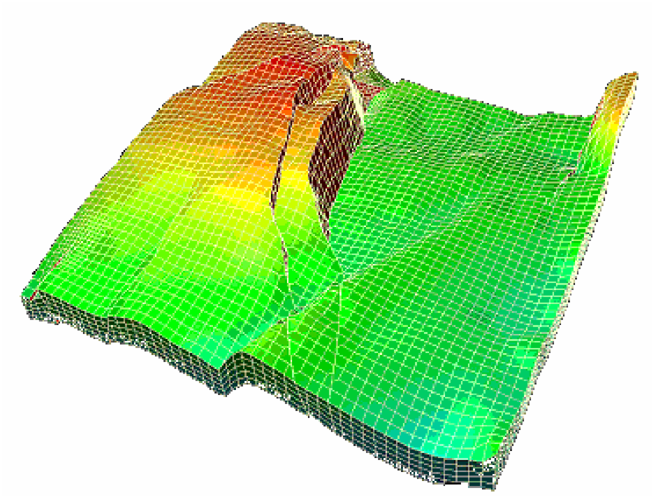


Figure 34: Type of grid used by the SURE simulator (Voronoi grid; courtesy SMT Alps)

5.3.1 A self-adapting direct AMG strategy

Regarding the employment of AMG, FIM carries over to AIM. In the following, we consider the direct application of AMG, using basically the same algorithmic components as before (cf. Section 5.2.2): Point-based AMG with pressure-driven coarsening[23] combined with robust ILUT-type smoothers, accelerated with some Krylov method (usually BiCGstab). As before, depending on the concrete situation, a special treatment of “critical” equations may be necessary (cf. well-decoupling in Section 5.1).

However, compared to the FIM case, there is a major difference in that the size and the character of the AIM matrices may change drastically from one simulation step to the next. In particular, their numerical character may change from nearly elliptic to increasingly hyperbolic. As a consequence, there is no fixed solver which solves all AIM-systems within a complete simulation run robustly *and* efficiently. Unfortunately, a theoretical forecast of what

[23] Note that pressure-driven coarsening is possible since the pressure is implicitly computed *everywhere*.

is “the best” solver for a given AIM-system seems virtually impossible without further information.

Hence, from a practical point of view, a strategy is required which *automatically* adapts the linear solver to the size and physical complexity of each AIM-system to be solved during a simulation run. Based on the above AMG approach (including its one-level variant), we have developed an automatic, self-adapting parameter and solver switching strategy. Separately for specific problem sizes, this strategy continuously keeps track of the histories of convergence factors, timings, AMG memory complexities and overall solver memory requirements. Based on this information, further information contained in the matrices and some “online testing”, the final solver is dynamically selected.

The overall strategy combines three types of adaptivity:

1. Switching between one-level and multilevel solver variants.
2. Switching of the ILUT parameters *lfil* and *droptol* (see Section 5.2.2). These parameters are dynamically changed according to the convergence, timings and memory requirement history. This refers to both ILUT as a one-level method and as a smoother in a multi-level cycle. Various heuristics check for “optimality”.
3. Automatic adaptation of AMG’s coarsening and interpolation to the coupling structure reflected by the AIM matrix entries. Special rows (typically the ones strongly violating diagonal dominance) are excluded from coarsening and coupled differently.

5.3.2 Numerical experience

The above strategy has been realized based on the SAMG library, called “α-SAMG” in the following. We have tested the feasibility and effectivity of α-SAMG in close cooperation with SMT Alps on the basis of their SURE simulator.

For a wide variety of real-life test cases from different application types – gas-water, black-oil, and compositional models – as well as different levels of physical complexity, α-SAMG outperforms the original solver[24] as well as other one-level and multi-level variants in terms of overall computing time as well as memory requirements. As expected, the performance gain tends to be the better the larger and/or more physically involved the considered model is. For a reasonably complex and large black-oil model, for instance, a speedup of more than 15 could be achieved keeping memory requirements reasonably low at the same time (see Case C2, below). Clearly, concrete performance details depend on the situation, in particular, on the dominance of the elliptic components. Roughly speaking, the more dominant the elliptic components are in the overall numerical solution, the higher the overall gain to be expected.

For a complete discussion of a list of benchmark cases and further details on the numerical procedures considered, we refer to [19]. In the following, we just summarize on two black-oil models, a relatively small and a larger one:

- **Case C1** represents an undersaturated black-oil simulation model with faults, local grid refinement and fractures. There are more than 50 wells (vertical and horizontal). The model was chosen to investigate solver performance on dual porosity model,

[24] The original linear SURE solver, ORILU, efficiently combines ILU-type approaches with ORTHOMIN.

because the natural fractures in this reservoir are modelled with a dual porosity approach applied to selected parts of the model.

- **Case C2** is a conventional 3-phase black-oil model, initially undersaturated, but due to production, the pressure drops below bubble point. The model contains more than one million active grid blocks and includes local grid refinement and faults.

Some further information on these models is found in Table 2. Besides the grid sizes and the number of functions involved, the table shows the total number of AIM matrix problems to be solved during a full simulation run as well as the average and maximum number of matrix rows. Note that, at any given simulation step, the number of rows equals the number of variables which are coupled implicitly. The ratio between the number of rows and the number of points is called the degree of implicitness (DOI). Finally, the achieved speedups[25] are shown.

	Case C1	**Case C2**
number of grid points	70,742	1,103,334
number of functions	5	5
number of matrix problems	935	703
average number of rows	84,608	1,147,276
maximum number of rows	110,201	1,332,839
average DOI	1.20	1.04
maximum DOI	1.56	1.21
speedup compared to ORILU	1.46	15.51

Table 2: Black-oil benchmark cases

The below figures show detailed results on the switching as performed by α-SAMG during full simulation runs corresponding to the above two cases.

The model C1 is fairly small so that there is a relatively high probability that, due to their lower overhead, one-level solvers may be more efficient than multi-level solvers. In fact, Figure 35 shows that α-SAMG switches between one-level and multi-level solvers during the course of the simulation. The concrete switching behavior as well as the number of iterations required for each matrix problem can be seen from the figure. The number of iterations ranges between just a couple and around 50, the average being 11.1 which is very reasonable. Obviously, the higher iteration numbers are attained by the one-level solvers whereas the multi-level solvers require much less iteration. To avoid misunderstandings, we should recall that the primary guiding principle of α-SAMG is to optimize *efficiency*, that is, to minimize overall computing time. Clearly, in comparing different types of linear solvers, the mere number of iterations provides hardly any indication for efficiency.

While Figure 35 details the solver switching for Case C1, Figure 36 details the corresponding ILUT parameter switching[26]. One sees that, most of the time, the fill-in parameter *lfil* is 4 except for some cases where it increases to 6 or even 7. At the same time, *droptol* is decreased by one order to magnitude. This shows that, during the course of the simulation, some more problematic matrices occur. Overall, however, parameters do not change dramatically during the simulation.

[25] Speedup values refer to the total simulation times; the speedup of the linear solver part alone is higher.
[26] The minimum fill-in value *lfil* is 4, and the maximum threshold value *droptol* is 0.01.

By replacing the original ORILU solver[24] by α-SAMG, a speedup[25] of 1.5 is observed for Case 1. Recalling that this particular problem is relatively small (around 70,000 grid points only) and that the overall switching process will always cause some overhead (in particular, due to its online testing), this is very acceptable.

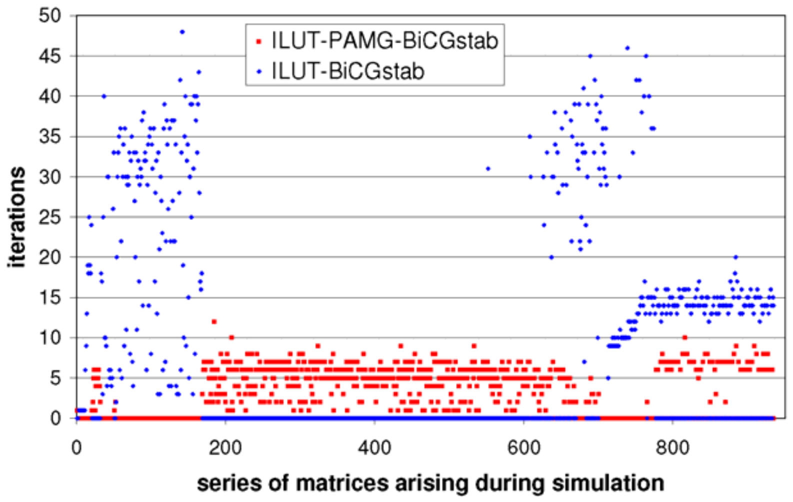


Figure 35: Solver switching and number of iterations for Case C1. Both, one-level (blue dots) and multi-level (red dots) methods are chosen by α-SAMG.

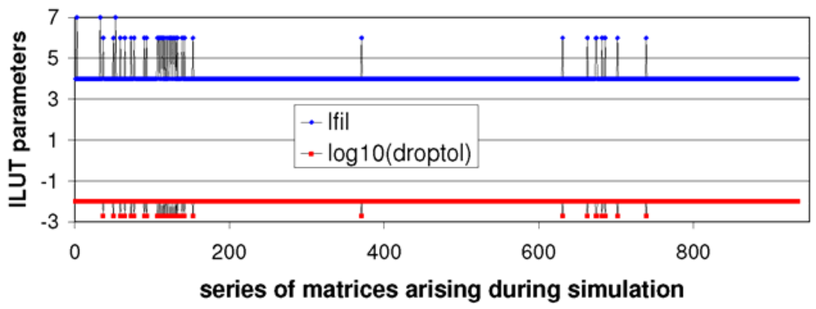


Figure 36: ILUT parameter switching for Case C1

For larger problems, the speedup may be much higher. This is demonstrated by Case 2 for which a speedup of approximately 15 is observed. For this large case, one-level methods are not competitive and α-SAMG decides to only use multi-level solvers. Figure 37 shows the corresponding number of iterations per simulation step. Although the average number of iterations needed is very reasonable, namely 16, there are some peaks where the number of iterations gets close to 100 or even higher. Note that these high iteration numbers are due to the fact that sometimes problems have to be re-run (e.g., with different ILUT parameters), and iterations are summed up over all attempts then. Indeed, as can be seen from Figure 38, there is a strong switching of ILUT parameters. The figure shows that the fill-in parameter *lfil* changes from around 4 to around 16, with 7.2 being the average. At the same time, also the *droptol* parameters changes strongly, namely, between 10^{-2} and 10^{-7}.

Cases C1 and C2 give some indication about how the general switching process works and what can be gained. Many more results and discussions can be found in [19]. The results clearly indicate that a reasonable and automatic switching mechanism is necessary since the type of problems to be solved during a full simulation run changes drastically and there is no hope to find a single parameter setting which is not only robust but also efficient.

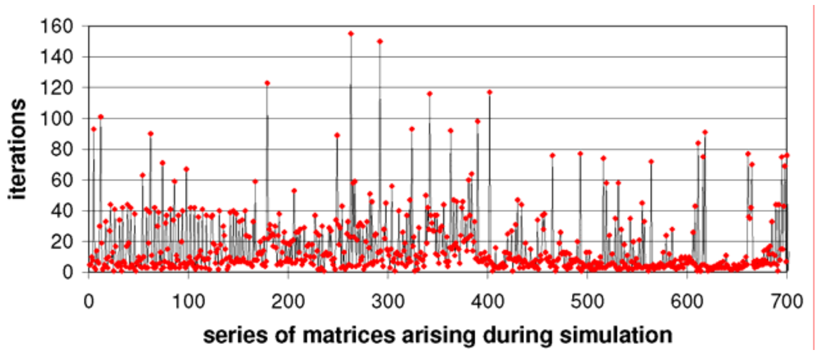


Figure 37: Number of SAMG iterations for Case C2. Only real multi-level methods are chosen by α-SAMG, one-level variants are not competitive.

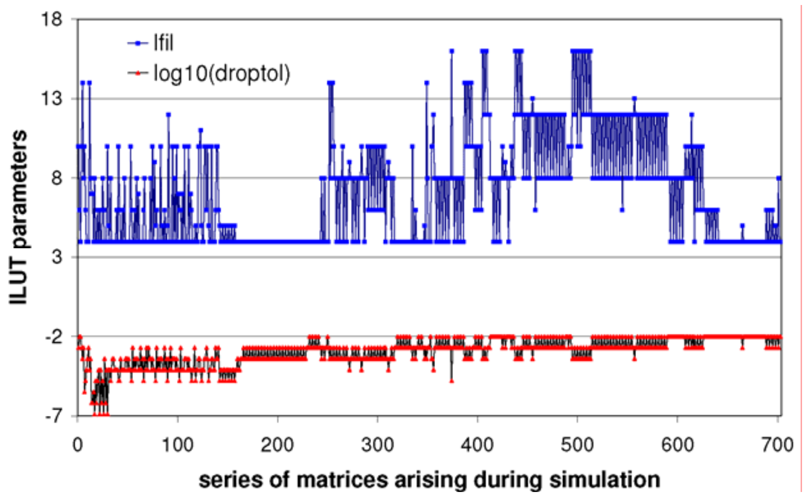


Figure 38: ILUT parameter switching for Case C2

In spite of the fact that a switching strategy as realized here causes quite some overhead, we have seen that the speedup achieved may be substantial. Of course, in each individual simulation case, the concrete numerical solution process could be optimized further. However, this is not the intention of α-SAMG. The intention is actually to demonstrate that, based on the set of solvers and parameters selected here, an automatic process can be designed which is efficient as a "pitch-black box" without the need of manually changing any further parameter.

6 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. Bank, R.E.; Smith, R.K.: *The incomplete factorization multigraph algorithm*, SIAM J. Sci. Comput. 20, pp 1349-1364, 1999.
3. Bank, R.; Wagner, Ch.: *Multilevel ILU decomposition*, Numer. Math. 82, pp. 543-576, 1999.
4. Barrett, R. et.al.: *Templates for the solution of linear systems: building blocks for iterative methods*, SIAM, Philadelphia, 1994.

5. Brand, C. W.; Ganzer, L.: *Iterative Solvers for Dynamically Implicit Reservoir Flow Equations on Irregular Grids*, paper presented at the 5th European Conference on the Mathematics of Oil Recovery (ECMOR), Leoben, Austria. Sep. 3-6, 1996.

6. Brandt, A.: *Multi-level adaptive technique (MLAT) for fast numerical solution to boundary value problems*, Lecture Notes in Physics 18, Springer 1973.

7. Brandt, A.: *Multi-level adaptive solutions to boundary value problems*, Math. Comp. 31, 1977.

8. Brandt, A.: *Multigrid techniques: 1984 Guide with applications to fluid dynamics*, GMD-Studie No. 85, 1984.

9. 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.

10. 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.

11. Brandt, A.: *Algebraic multigrid theory: the symmetric case*, Appl. Math. Comp. 19, pp 23-56, 1986.

12. Brandt, A., *The scope of multiresolution iterative computations*, SIAM News 23, pp 8-9, 1990.

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. 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)*, SIAM J. Sc. Comp. 22, 2000.

15. Briggs, W.: *A multigrid tutorial*, SIAM, Philadelphia (1987). New edition: 2001.

16. 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.

17. Clees, T.; Stüben, K.: *Algebraic multigrid for industrial semiconductor device simulation*, Proceedings of the First International Conference on Challenges in Scientific Computing, Berlin, Germany, Oct 2-5, 2002. Lecture Notes in Computational Science and Engineering 35, Springer, Heidelberg, Berlin, 2003.

18. Clees, T., *AMG Strategies for PDE Systems with Applications in Industrial Semiconductor Simulation.* Dissertation, University of Cologne, Shaker Verlag, Aachen, Germany, October 2005.

19. Clees, T. and Ganzer, L.: *An Efficient Algebraic Multi-Grid Solver Strategy for Adaptive Implicit Methods in Oil Reservoir Simulation*, paper SPE 105789 presented at the 2007 SPE Reservoir Simulation Symposium, Houston, TX, Feb. 26-28, 2007.

20. Dendy, J.E. (Jr.): *Black box multigrid*, J. Comp. Physics 48, pp. 366-386,1982.

21. Falgout, R.D.: *An Introduction to Algebraic Multigrid*, Computing in Science and Engineering, Special issue on Multigrid Computing, 8 (2006), pp. 24-33, UCRL-JRNL-220851.

22. Füllenbach, T.; Stüben, K.; Mijalkovic, S.: *Application of an algebraic multigrid solver to process simulation problems*, Proceedings of the International Conference on Simulation of Semiconductor Processes and Devices, Seattle (WA), USA, Sep 6-8, 2000. IEEE, Piscataway (NJ), USA, pp. 225-228, 2000.

23. Füllenbach, T.; Stüben, K.: *Algebraic multigrid for selected PDE systems*, Proceedings of the Fourth European Conference on Elliptic and Parabolic Problems, Rolduc (The Netherlands) and Gaeta (Italy), 2001. World Scientific, New Jersey, London, pp. 399-410, 2002.

24. Hackbusch, W.: *On the multigrid method applied to difference equations*, Computing 20, 1978.

25. Hackbusch, W.: *Multigrid methods and applications*, Springer Series in Comp. Math. 4, Springer, 1985.

26. Häfner, F.; Stüben, K.: *Simulation and Parameter Identification of Oswald's Saltpool Experiments with the SAMG Multigrid-Solver in the Transport code MODCALIF*, Proceedings of the Conference "Finite-Element Models, MODFLOW, and More 2004: Solving Groundwater Problems", September 13-16, 2004, Karlovy Vary (Carlsbad), Czech Republic.

27. Klie, H., Wheeler, M.F., Clees, T., Stüben, K.: *Deflation AMG Solvers for Highly Ill-Conditioned Reservoir Simulation Problems*, paper SPE 105820 presented at the 2007 SPE Reservoir Simulation Symposium, Houston, TX, Feb. 28–30.

28. Krechel, A.; Stüben, K.: *Operator dependent interpolation in algebraic multigrid*, Lecture Notes in Computational Science and Engineering 3, Springer Verlag, 1998.

29. Krechel, A.; Stüben, K.: *Parallel algebraic multigrid based on subdomain blocking*, Parallel Computing 27, pp. 1009-1031, 2001.

30. Krechel, A., Stüben, K.: *SAMGp User's Manual*, Fraunhofer SCAI, can be downloaded from http://www.scai.fraunhofer.de/samg.

31. 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.

32. McCormick, S. (ed.).: *Multigrid methods*, Frontiers in Applied Mathematics, Vol. 5, SIAM, Philadelphia, 1987.

33. 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.

34. Notay, Y.: *Algebraic multigrid and algebraic multilevel methods: a theoretical comparison*, Numer. Linear Algebra Appl. 12, pp.419-451, 2005.

35. 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.

36. Ruge, J.W.; Stüben, K.: *Algebraic Multigrid (AMG)*, In „Multigrid Methods“ (McCormick, S.F., ed.), SIAM, Frontiers in Applied Mathematics, Vol 5, Philadelphia, 1986.

37. Saad, Y.; Schultz, M.H.: *GMRes: a generalized minimal residual algorithm for solving nonsymmetric linear systems*, SIAM J. Sci. Stat. Comp. **7**, 856, 1986.

38. Saad, Y.: *ILUT: A dual threshold incomplete ILU factorization*, Numer. Lin. Alg. Appl. 1, 387, 1994.

39. Saad, Y.: *ILUM: a multi-elimination ILU preconditioner for general sparse matrices*, SIAM J. Sci. Comput. 17, pp 830-847, 1996.

40. Saad, Y.: *Iterative Methods for Sparse Linear Systems*, 2nd ed., SIAM Society for Industrial & Applied Mathematics, 2003, http://www-users.cs.umn.edu/~saad/books.html.

41. Stüben, K.; Trottenberg, U.: *Multigrid methods: Fundamental algorithms, model problem analysis and applications*, Lecture Notes in Mathematics 960, Springer, 1982.

42. Stüben, K.: *Algebraic multigrid (AMG): Experiences and comparisons*, Appl. Math. Comp. 13, pp. 419-452, 1983.

43. Stüben, K.: *An Introduction to Algebraic Multigrid*, Appendix in the book „Multigrid“ by U. Trottenberg; C.W. Oosterlee; A. Schüller, Academic Press, pp. 413-532, 2001.

44. Stüben, K.: *A Review of Algebraic Multigrid*, Journal of Computational and Applied Mathematics 128, pp. 281-309, 2001.

45. Stüben, K.; Delaney, P.; Chmakov, S.: *Algebraic Multigrid (AMG) for Ground Water Flow and Oil Reservoir Simulation*, Proceedings of the Conference "MODFLOW and More 2003: Understanding through Modeling", International Ground Water Modeling Center (IGWMC), Colorado School of Mines. Golden, Colorado, Sept 17-19, 2003.

46. Stüben, K., Clees, T., Klie, H., Lou, B., Wheeler, M.F.: *Algebraic Multigrid Methods (AMG) for the Efficient Solution of Fully Implicit Formulations in Reservoir Simulation*, paper SPE 105832 presented at the 2007 SPE Reservoir Simulation Symposium, Houston, TX, Feb. 28–30, 2007.

47. Stüben, K. and Clees, T.: *SAMG User's Manual*, Fraunhofer Institute SCAI, can be downloaded from http://www.scai.fraunhofer.de/samg.

48. Thiele, M.: *Streamline Simulation*, 8th International Forum on Reservoir Simulation, Stresa / Lago Maggiore, Italy, June 20-24, 2005 (see also 9th International Forum on Reservoir Simulation, Abu Dhabi, United Arab Emirates, Dec 9-13, 2007).

49. Thomas, G.W.; Thurnau, D.H.: *Reservoir Simulation Using an Adaptive Implicit Method*, SPEJ 23, 759, 1983.

50. Trottenberg; U.; Oosterlee, C.W.; Schüller, A.: *Multigrid*, Academic Press, 2001 (with appendices by Brandt, A., Oswald, P. and Stüben, K.).

51. Vanek, P.; Mandel, J.; Brezina, M.: *Algebraic Multigrid by Smoothed Aggregation for Second and Fourth Order Problems*, Computing 56, 179, 1996.

52. Van der Vorst, H.A.: *Bi-CGSTAB: A fast and smoothly converging variant of Bi-CG for the solution of non-symmetric linear systems*, SIAM J. Sci. Stat. Comp. 13, 631, 1992.

53. Vinesome, P.K.W.: *Orthomin, an Iterative Method for Solving Sparse Banded Sets of Simultaneous Linear Equations*, paper SPE 5729 presented at the 4th SPE Symposium on Reservoir Simulation, Los Angeles, CA. Feb. 19-20, 1976.

54. Wesseling, P.: *An introduction to multigrid methods*, Pure and Applied Mathematics Series, John Wiley and Sons, 1992.

55. Wheeler, J.: *IPARS User's Manual*, Tech Report CSM, ICES, The University of Texas at Austin, Austin, TX, 2000.

56. Wienands, R.; Joppich, W.: *Practical Fourier Analysis for Multigrid Methods*, CRC Press, Bocaraton, Florida, USA, 2004.

57. Yavneh, I.: *Why Multigrid Methods are so Efficient*, Computing in Science and Engineering, Special issue on Multigrid Computing, 8 (2006), pp. 12-22, UCRL-JRNL-220851.