SPE 105789

An Efficient Algebraic Multigrid Solver Strategy for Adaptive Implicit Methods in Oil Reservoir Simulation

T. Clees, Fraunhofer SCAI, and L. Ganzer, SPE, SMT Alps*

* Now with U. of Leoben

This paper was prepared for presentation at the 2007 SPE Reservoir Simulation Symposium held in Houston, Texas, U.S.A., 26–28 February 2007.

Abstract

We propose a new, efficient, adaptive algebraic multigrid (AMG) solver strategy for the discrete systems of partial differential equations arising from structured or unstructured grid models in reservoir simulation. The proposed strategy has been particularly tailored to linear systems of equations arising in adaptive implicit methods. The coarsening process of the AMG method designed automatically employs information on the physical structure of the models; as a smoother, an adaptive ILUT method is employed, taking care of an efficient solution of the hyperbolic parts whilst providing adequately smooth errors for the elliptic parts. To achieve a good compromise of high efficiency and robustness for a variety of problem classes – ranging from simple, small black-oil to challenging, large compositional models – an automatic, adaptive ILUT parameter and AMG solver switching strategy, α-SAMG, has been developed. Its efficiency is demonstrated for eight industrial benchmark cases by comparison against standard one-level and AMG solvers as well as the pure one-level variant of the proposed new strategy. In addition, very promising results of first parallel runs are shown.

Introduction

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.

To avoid huge computational costs, adaptive implicit methods (AIM) have been developed, offering a good compromise between computational speed, memory requirements and accuracy. Still, comparably to FIM, the mixed hyperbolic-elliptic character of AIM matrices[a] aggravates the development of efficient linear solvers.

On the one hand, iterative hierarchical solvers are necessary for the optimal solution of mixed hyperbolic-elliptic problems; only a hierarchy allows for a solver complexity which scales linearly with the problem size. On the other hand, algebraic multigrid (AMG) methods, providing optimal solvers for elliptic problems, loose efficiency if applied without modification to mixed hyperbolic-elliptic systems.

In this paper, we develop AMG methods suitable for matrices arising from adaptive and also fully implicit methods. In particular, we will see that so-called point-based AMG methods employing a pressure-based coarsening with robust, ILUT-based smoothers and a special treatment of well equations provide efficient preconditioners. Moreover, we introduce an adaptive ILUT-parameter and solver switching strategy which can efficiently and robustly handle matrices arising from gas-water, black-oil and compositional fluid characterization in single- and dual-porosity simulation models. This solver strategy automatically adapts itself to the magnitude and physical complexity of each matrix to be solved during a simulation, making use of experiences gained with the series of matrices solved so far, so that matrices ranging from simple and/or small up to physically challenging and/or large are handled efficiently without the need to change solver settings manually.

The outline of the paper is as follows. We start with an overview of adaptive implicit methods. In particular, properties of the arising matrices, as far as important for their efficient numerical solution, are described. We give an overview of the state-of-the-art solution methods for these matrices and point out advantages and disadvantages of these solvers. Afterwards, we explain a well-known class of multi-level solvers, namely algebraic multigrid methods (AMG), which have the potential to solve matrices arising in AIM with (nearly) optimal numerical complexity. These methods have been developed for scalar elliptic partial differential equations. Since AIM matrices correspond to a PDE system of mixed elliptic-hyperbolic character, we summarize available AMG methods for PDE systems in general and methods promising for an application to AIM matrices. As the main contribution of this paper, we describe in detail our new automatic,

[a] Matrices arising in adaptive implicit methods are called AIM matrices in the following.

adaptive ILUT parameter and AMG solver switching strategy for AIM matrices, α-SAMG, and show its robustness and efficiency for selected industrial gas-water, black-oil and compositional benchmark cases of different physical complexity and model sizes ranging from ten thousand to more than one million active grid blocks. In the outlook, we will present very promising results of first parallel runs for representative matrices from the largest model case analyzed so far.

Adaptive Implicit Methods (AIM)

The fluid flow simulator used in this work is based on a general-purpose, compositional formulation. A mole balance equation in compositional formulation can be written for each component in the following form:

$$-\nabla\left[\sum_{p=1}^{P}\left(x_{pc}D_p\vec{u}_p\right)+\vec{j}_{Dc}\right]+\sum_{p=1}^{P}\left(q_p x_{pc}\right) = \frac{\partial}{\partial t}\left[\phi\sum_{p=1}^{P}\left(x_{pc}D_p S_p\right)\right] \quad (1)$$

The mass transport in Eq.1 consists of two terms. The first term represents the convective flux, the second term represents the combined transport by dispersion and diffusion. In addition, the left hand side of the balance equation contains a source/sink contribution from the wells. The right hand side represents the rate of mass accumulation. Eq. 1 is valid for arbitrary grid block geometries. Darcy's law is defined by

$$\vec{u}_p = -\lambda_p \overline{\overline{k}}\nabla\Phi_p \ , \quad (2)$$

where

$$\lambda_p = \frac{k_{rp}}{\mu_p} \text{ and } \nabla\Phi_p = \nabla p_p - \rho_p\vec{g} \ . \quad (3)$$

Combined with Darcy's law, the mole balances yield highly non-linear partial differential equations of mixed hyperbolic-elliptic type.

Using a control volume finite difference (CVFD) discretization, the general mole balance equation in difference form for a component *c* in a gridblock *i* is given by

$$\sum_{j=1}^{N}\tau_{ij}\left[\sum_{p=1}^{P}\left(x_{pc}\lambda_p D_p\right)_{ij}^{n+1}\left(\Phi_{pj}-\Phi_{pi}\right)^{n+1}\right] + \sum_{p=1}^{P}\left(q_p D_p x_{pc}\right)_i^{n+1} = \frac{V_i}{\Delta t}\Delta_t\left[\phi\sum_{p=1}^{P}\left(S_p D_p x_{pc}\right)\right]_i \quad (4)$$

For simulating the reservoir, two alternatives have been pursued in the past: keep the grid structured and set up the equations in either IMPES or fully implicit mode. Using this approach, the resulting matrices have regular bands and structure principally allowing for highly cache-efficient solver algorithms.

Alternatively, one may use unstructured Voronoi (or PEBI) grids with irregular number of connections and also permit both IMPES and fully implicit discretizations simultaneously within the model. This type is termed adaptive implicit formulation (AIM), because IMPES formulation is applied for some parts of the reservoir, but fully implicit formulation in other areas (e.g. at perforated grid blocks or in areas with large variable changes). As a result, the user gains a maximum in flexibility, but the linear solver encounters matrices with highly irregular structures.

All benchmark cases illustrated in this work follow the latter description, use unstructured gridding and adaptive implicit formulation introduced by Thomas and Thurnau[1]. Rather than providing a fixed number of implicit unknowns in a grid block, this method operates with different numbers of unknowns (level of implicitness) in adjacent grid blocks. The goal is to restrict the expensive fully implicit formulation to that part of the model that requires it, leaving the other parts in the less expensive IMPES mode. In 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. This division into implicit and IMPES blocks within the simulation model may vary from one solution step to the next based on a switching criterion and leads to a distortion of the otherwise banded structure of the coefficient matrix.

Properties of AIM matrices. The AIM matrices to be solved in the Newton-Raphson process are highly nonsymmetric and contain varying degrees of freedom caused by the variation in the number of implicit variables treated in a grid block. Individual matrix coefficients can be highly 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 (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 AIM matrices. Finally, well equations which are fully coupled to the system of equations relate grid blocks that otherwise are not geometrically connected.

As analyzed by Klie[2] and by Stüben, Clees, Klie et al.[3,4], the pressure block is basically elliptic. Saturations add hyperbolic behavior to the system. Depending on the concrete situation, the system's matrices can be of elliptic up to nearly hyperbolic character. Physically more involved models additionally change the eigenvalues of the matrices and make them more difficult to solve. In particular, ellipticity and diagonal dominance can suffer considerably from well equations, but also from larger time steps, which are nevertheless desirable in order to reduce computational time – as long as the linear solver employed can efficiently cope with the resulting matrices. A good compromise has to be found here.

Possibilities for solving AIM matrices. Linear complexity of the linear solver employed is strongly desired in order to reduce computing times and to cope with increasingly large models. The application of direct solution methods (based on Gaussian elimination) to matrices arising in oil reservoir simulation is not feasible since these methods exhibit a complexity far away from being linear – often, they are of quadratic complexity $O(N^2)$ or even worse – and the number of grid nodes and thus variables easily exceeds several hundred thousands already.

Usually, standard iterative one-level methods are therefore used. Most often, an incomplete LU factorization method, as for instance ILUT[5] or ILU(k)[6], together with a standard accelerator, as for instance BiCGstab[7], GMRes[8] or ORTHOMIN[9], is employed.

However, also iterative one-level methods do not exhibit a linear complexity of $O(N)$. It is common opinion that, in order to approach optimality here, multi-level methods are necessary. Among the well-known and most promising approaches are algebraic multigrid methods.

Algebraic Multigrid (AMG)

In the following, we briefly characterize approaches for scalar PDEs, before we give an overview of approaches for PDE systems, followed by a discussion of AMG for AIM matrices.

AMG for scalar partial differential equations (variable-based AMG, VAMG). Two main classes of algebraic multigrid methods are known today, namely classical AMG and aggregation- or agglomeration-based AMG.

Classical AMG[10,11,12] is known to provide very efficient and robust solvers or preconditioners for large classes of matrix problems $Av = b$, an important one being the class of (sparse) linear systems with matrices A which are "close" to being M-matrices. Problems like this widely occur in connection with discretized scalar elliptic partial differential equations (PDEs). In such cases, classical AMG is very mature and can handle millions of variables much more efficiently than any one-level method. Since explicit information on the geometry (such as grid data) is not needed, AMG is especially suited for unstructured grids both in 2D and 3D. In fact, the coarsening process is directly based on the connectivity pattern reflected by the matrix, and interpolation is constructed based on the matrix entries. Restriction is simply defined to be the transpose of interpolation, regardless whether the matrix to be solved is symmetric or not. The Galerkin coarse-level matrix for level $n+1$ is computed as

$$A_{n+1} = I_n^{n+1} A_n I_{n+1}^n \tag{5}$$

with I_{n+1}^n being the interpolation from level $n+1$ to n, and

$$I_n^{n+1} = \left(I_{n+1}^n\right)^T \tag{6}$$

the restriction from level n to $n+1$, starting from level 1 which represents the original matrix equation $Av = b$.

Aggregation- or agglomeration-based AMG differ from classical AMG in the way coarse-level variables and interpolation formulas are constructed. Aggregation means splitting the set of variables into disjoint subsets (the supernodes or macro variables), for each of which a constant interpolation formula is constructed. This approach yields a simple-to-compute Galerkin operator as well as low operator complexities[11,13] for the hierarchy constructed. Interpolation in case of agglomeration-based AMG is also piecewise constant. However, agglomerates are sets of neighboring finite elements glued together. Here, variables on the interfaces of elements belong to more than one agglomerate. Their interpolation is just the "average interpolation" of the surrounding agglomerates. For both, aggregation- and agglomeration-based AMG, smoothing of interpolation[13] can be used in order to increase the quality of interpolation and thus the robustness of the overall method which is especially necessary for matrices stemming from second-order discretizations.

Classical AMG with aggressive coarsening[10,11,12], roughly characterized, is a means to provide a compromise between the robustness of interpolation of classical AMG and the low grid and operator complexity resulting from aggregation-based AMG.

AMG for PDE systems. Extensions of these "scalar" AMG methods are required to efficiently solve systems of PDEs involving two or more scalar functions (called unknowns in the following). This is because classical AMG realizes a variable-based approach which does not distinguish between different unknowns. Unless the coupling between different unknowns is very weak, such an approach cannot work efficiently for systems of PDEs where, in general, the corresponding matrix is far from being an M-matrix. In the past, several ways to generalize AMG have been investigated, and there is still an ongoing rapid development of new AMG and AMG-like approaches. For a review, we refer to Clees[14]. However, there is no unique and best approach yet. All approaches seem to have their range of applicability but all of them may fail to be efficient in certain other applications.

Unknown-Based AMG (UAMG). We first want to recall a rather popular AMG approach to solve systems of PDEs, the so-called unknown-based approach, which is very similar to the variable-based approach except that all unknowns are treated separately. To be more specific, let us assume the variables to be ordered by unknowns, that is, $Av = b$ has the form

$$\begin{pmatrix} A_{[1,1]} & \cdots & A_{[1,n_u]} \\ \vdots & \ddots & \vdots \\ A_{[n_u,1]} & \cdots & A_{[n_u,n_u]} \end{pmatrix} \begin{pmatrix} v_{[1]} \\ \vdots \\ v_{[n_u]} \end{pmatrix} = \begin{pmatrix} b_{[1]} \\ \vdots \\ b_{[n_u]} \end{pmatrix}, \tag{7}$$

where n_u denotes the number of unknowns of the given system of PDEs, $v_{[n]}$ denotes the vector of variables corresponding to the n-th unknown, $b_{[n]}$ the respective part of the right-hand side, and the matrices $A_{[m,n]}$ reflect the couplings of the m-th to the n-th unknown. Using this notation, coarsening the set of variables which correspond to the n-th unknown is strictly based on the connectivity structure reflected by the submatrix $A_{[n,n]}$, and interpolation is based on the corresponding matrix entries. In particular, interpolation to any variable v_i involves

only coarse-level variables corresponding to the same unknown v_i belongs to. The Galerkin matrices, however, are usually computed w.r.t. all unknowns.

This unknown-based approach has been proposed already in the very early papers on AMG (see references given in Stüben[11]). It is certainly the simplest approach for solving PDE systems, which nevertheless works quite efficiently for some important practical applications. Compared to the variable-based approach, the only additional information required is information about the correspondence between variables and unknowns.

The unknown-based approach is mainly used for applications where the matrices $A_{[n,n]}$ are close to being M-matrices. The essential additional condition for the approach to work is that smoothing results in an error which is smooth separately for each unknown. One advantage of this approach is that it can easily cope with anisotropies which are different between the different unknowns. Another advantage is that unknowns can virtually be distributed arbitrarily across mesh points. However, this approach will become inefficient, for instance, if the cross-unknown couplings are too strong.

Point-Based AMG (PAMG): A General Framework. Many PDE systems of practical importance are too strongly coupled for VAMG and also UAMG. This is especially the case for oil reservoir simulation. Clees[14] developed a flexible framework for constructing so-called "point-based" AMG approaches to solve various types of strongly coupled PDE systems. This framework for PAMG approaches along with (classical) VAMG and UAMG approaches is integrated in Fraunhofer SCAI's (parallel) linear solver library SAMG(p)[15,16].

In contrast to the unknown-based approach, a point-based approach operates (i.e. coarsens and/or interpolates) on the level of points rather than variables as in VAMG and UAMG. Since we have the solution of PDEs in mind, we think of points as being real physical grid nodes (in space). However, from AMG's point of view, it is sufficient to think of the nodes of a graph representing the connectivity structure of A. Regarding a point-based approach, it is only relevant whether there are (disjoint!) "blocks" of variables (corresponding to different unknowns) which may be coarsened, maybe also interpolated, simultaneously.

To be more specific, we assume a reasonable splitting of variables into points to be given. Let $Av = b$ then be ordered point-wise, i.e.

$$\begin{pmatrix} A_{(1,1)} & \cdots & A_{(1,n_p)} \\ \vdots & \ddots & \vdots \\ A_{(n_p,1)} & \cdots & A_{(n_p,n_p)} \end{pmatrix} \begin{pmatrix} v_{(1)} \\ \vdots \\ v_{(n_p)} \end{pmatrix} = \begin{pmatrix} b_{(1)} \\ \vdots \\ b_{(n_p)} \end{pmatrix}, \qquad (8)$$

where n_p denotes the number of points, $v_{(k)}$ the vector of variables corresponding to the k-th point, $b_{(k)}$ the respective part of the right-hand side, and the matrices $A_{(k,l)}$ reflect the couplings of the k-th to the l-th point.

In order to coarsen A, a so-called primary matrix P of dimension n_p is constructed. Its entries can be seen to result from a "condensation" of the point-coupling matrices $A_{(k,l)}$ to scalar values of P in such a way that the resulting P reflects the couplings between the points reasonably well. A VAMG coarsening process is applied to P. The resulting hierarchy of points is then used for all unknowns. Note that this is different from the unknown-based approach where each unknown is associated with its own hierarchy.

Many different approaches for constructing P, and subsequently a suitable interpolation are possible and available in SAMG. For a brief overview on SAMG's main components, see Fig. 1. It should be noted that a reasonable choice of components strongly depends on the class of applications at hand.

Results for industrial applications in semiconductor device simulation are presented in Clees[14], showing that (different) suitable PAMG approaches yield efficient solution processes for three very different and important types of PDE systems, namely Lamé equations (linear elasticity), reaction-diffusion and drift-diffusion equations.

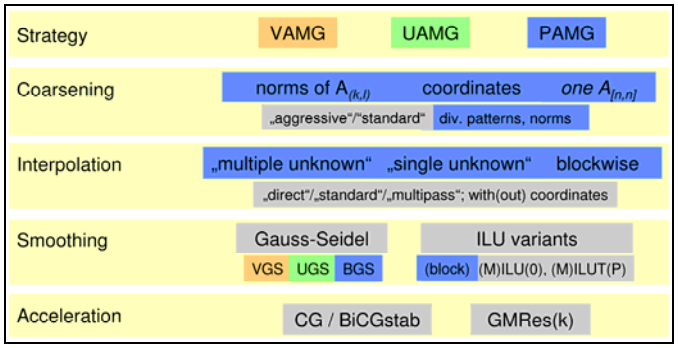

Fig. 1: Main components of the linear solver library SAMG[15].

AMG Methods for AIM matrices

It is known that classical AMG is an efficient and robust preconditioner for matrices arising in IMPES and streamline methods[11,17], as long as the algebraic manipulations used for IMPES as well as the integration of well equations do not destroy the M-matrix property too much. This is because the discrete pressure equations to be solved then are nearly elliptic, and AMG has been designed for such matrices.

Since AIM matrices - as FIM matrices - stem from a discretization and linearization of a strongly coupled PDE system, AMG methods for scalar PDEs do not work efficiently for them. At the moment, there are two general ways for employing AMG in AIM or FIM:

- Use inside a two-stage approach as, for instance, the commonly used CPR[18] method: a VAMG method with standard components (in particular, Gauss-Seidel relaxation) as used for a discrete Poisson equation is employed as a preconditioner for an IMPES-like equation. In addition, an incomplete LU factorization[19], multi-level ILU[3] or even a simpler method as LSOR, for instance[3], is used as a preconditioner for (the remaining variables or, typically) the whole system. This two-stage preconditioner is accelerated by a standard Krylov-subspace method, for instance BiCGstab[7] or GMRes[8], to

yield the overall approach, called CPR-VAMG in the following.
- Use of a coupled AMG solver, i.e. direct application of AMG to the whole matrix: here, an AMG method suitable for the discrete PDE systems arising in AIM or FIM is necessary.

Two-stage approach (CPR-VAMG). Whereas such approaches are already quite common in FIM, little experience seems to have been gained for AIM. In general, the efficiency of CPR (-like)-VAMG methods is limited by three factors:
- the efficiency of the AMG preconditioner for the IMPES-like equation. AMG is perfectly suited for anisotropic diffusion equations (with a standard second-order discretization). However, IMPES-like equations might substantially deviate from this situation. They are obtained from the overall system's matrix by algebraic manipulations. As an effect, they might not be M-matrices any more [2,3,4]. In particular, negative eigenvalues can arise, which usually decrease AMG's efficiency drastically. Several attempts have been made to circumvent these problems (see Klie[2], for instance). However, a solution of this problem is not known. Moreover, an appropriate integration of well equations in the overall process (maybe via approximate decoupling as another stage) is an open issue.
- the efficiency of the incomplete LU factorization (or LSOR or a multi-level ILU) as a preconditioner for (the remaining variables or, typically) the whole system. In general, the optimal choice of the preconditioner for the second stage is an open question.
- the interplay of both methods being part of the two-stage preconditioner. Both preconditioners are only weakly coupled. It is crucial for the efficiency, though not obvious at all, which stopping criteria shall be used for the different stages.

Coupled AMG solver for AIM and FIM matrices. Solving FIM or AIM matrices directly (in one stage) by means of an AMG-based approach is – besides first simple attempts[14,20] for FIM – apparently new. In Stüben, Clees et al.[3,] first results for some standard, yet physically rather simple benchmark cases are presented. We here concentrate on the description and discussion of an approach which is capable of solving a wide variety of AIM matrices efficiently *and* robustly. We make use of SAMG(p)[15,16]. The main components of the approach finally chosen, used as a preconditioner for BiCGstab and called (ILUT-)pc-PAMG, are discussed in the following.

PAMG as the principal strategy. We already indicated that AIM and FIM matrices cannot be treated with VAMG and are too strongly coupled for UAMG, unless only simple models are considered. Since in oil-reservoir simulation finite-volume discretizations are used, the technical prerequisites for SAMG's PAMG approach are fulfilled: points simply correspond to grid nodes here. This also works for AIM since SAMG allows for a changing number of variables per point. With the pressure, the discrete AIM/FIM PDE system contains an elliptic component which drives – at least to a certain amount – the long-range couplings of the whole system. Hence, we now analyze the applicability of PAMG approaches, in particular based on the pressure unknown, to AIM/FIM matrices.

ILUT Smoothing. Since AIM/FIM matrices stem from PDE systems of mixed elliptic-hyperbolic character, one should choose a smoother which additionally acts as a solver for the "characteristic saturation directions", i.e. the (more or less) hyperbolic part here. Block-Gauss-Seidel, ILU(0) and block-ILU(0)[b] do not provide enough robustness since they only work for some models and do even then not perform efficiently enough (see also Table 4). Especially, if one tries to enlarge time steps without loosing efficiency of the linear solver, one has to use stronger smoothers. A good candidate is ILUT[5]. Its two parameters are lfil, which is the level of absolute fill-in, as well as droptol, which is a threshold for dropping paths belonging to small couplings during elimination. As other ILU-type approaches, ILUT's efficiency strongly depends on the ordering of variables of the matrix equation. We found that, for instance, the ordering provided by the simulator used is very appropriate for ILUT.

Pressure-based coarse-level correction with special treatment of critical matrix rows. Numerical experiments have shown that both, norm-based and pure pressure-based PAMG face problems. Norm-based methods, even with strong block-smoothers do not converge here usually. Pure pressure-based PAMG, on the other hand, seems to suffer considerably from more complicated physical situations (especially well equations). Matrix rows strongly violating diagonal dominance (also for the pressure-to-pressure-coupling sub-matrix of A), which can result, for instance, from well equations, seem to be the main reason. However, these matrix rows can be marked, and corresponding information can be submitted to SAMG. If we force SAMG to exclude these rows from the coarsening process, but include them in smoothing (on the finest level)[c] and acceleration, a good convergence is obtained again.

The coupled AMG preconditioner pc-PAMG proposed in comparison to a multi-stage preconditioner, as for instance CPR-VAMG:
- No artificial IMPES-like pressure matrix is constructed. Hence, we avoid problems with "additional" indefiniteness and "unphysical" eigenvalues.
- Moreover, extra time/memory needed in CPR-VAMG, for creating/storing such a matrix is spared.
- A complicated tuning of the interplay between the different stages of a multi-stage method is omitted.
- A special, global treatment of "special" rows, for instance coming from well equations, can be integrated directly. Algebraic (pre-)manipulations can be avoided also here.

[b] Abbreviated below by BGS, ILU, BILU, respectively.

[c] Another possibility would be treating them with a(n overlapping) Schwarz method or an approximate Schur complement approach. Numerical tests have shown that this was not necessary for the cases analyzed so far.

- By even forcing all variables not belonging to the (main) pressure unknown to stay on the finest level, an approach being a compromise between the pc-PAMG approach detailed above and CPR-VAMG would result. However, a main difference being that, due to the Galerkin operator, cross-unknown couplings are still taken into account on coarser levels in pc-PAMG approaches.

Further important properties of pc-PAMG(-BiCGstab):

- Due to ILUT smoothing, the approach can be made more robust also for quite tightly coupled systems as well as larger time steps.
- We use aggressive coarsening on all levels which produces very reasonable operator complexities (see also Table 6) which further contributes to low memory requirements (see also Table 5).
- The accelerator used, BiCGstab, is free of parameters.
- The ILUT smoother cannot be used with a fixed setting of its parameters lfil and droptol. Hence, a control mechanism which performs a "forecast" of suitable values seems mandatory.
- The overhead (AMG's setup phase as well as ILUT's decomposition phase) of the approach proposed pays off only if the matrix exceeds a certain magnitude (number of variables).
- The real multi-level variant needs a certain level of (discrete) ellipticity since this part is the one accelerated by the hierarchy.
- Couplings of "essentially hyperbolic" matrices are strongly driven by characteristic directions. Hence, the above approach in its one-level variant should be appropriate.

α-SAMG, an Automatic and Adaptive Parameter and Solver Switching Strategy

The properties of the coupled solver mentioned above are confirmed by practically relevant test cases, as for instance (but not restricted to) the ones presented in this paper. Benchmarks with many different matrices and full simulation runs for different models have shown that – at least so far – not one single solver with a fixed set of parameters is able to solve all matrices efficiently and robustly. This is because the character of the matrices can change drastically from nearly elliptic to more and more hyperbolic. A practical way to circumvent this problem very efficiently *and* robustly is sketched now.

ILUT switching. Our numerical experiments have shown that all matrices can be solved by means of our basic solver described above, in both its multi-level or one-level variant, if suitable values for ILUT's lfil and droptol are chosen. ILUT's robustness can be increased by means of a larger lfil and/or smaller droptol. By decreasing lfil or increasing droptol, ILUT can be made less memory- or time-consuming, respectively. However, a careful adjustment of these parameters is necessary since a forecast of ILUT's effective performance for a given matrix without further information or "online testing", i.e. testing during the simulation run, is virtually impossible. Hence, a carefully designed control mechanism should allow for solving all matrices efficiently. The mechanism we have developed is integrated in the solver switching outlined in the following.

Online Solver switching. In order to allow for the employment of the most efficient solver possible for differently large matrices, we divide the range of (theoretically) possible numbers of variables for the application at hand into so-called "dimension classes (D-classes)". A rough upper limit for the maximum number of variables is given by the maximum number of points times the maximum number of physical unknowns.

During a simulation run, for each D-class, a history of convergence factors, timings, AMG memory complexities and overall solver memory requirement is continuously being stored and updated. Within a given D-class, comparisons between different solvers (currently the one-level and multi-level variant of our basic solver) and ILUT settings are made from time to time (e.g. for every 50th matrix within its D-class) in order to define the direction of higher efficiency and switch the solver and/or ILUT settings appropriately.

Treatment of large matrices. If the overall memory requirement for a concrete run and matrix reaches the limits of the machine used, the switching mechanism changes its default strategy. In particular, it tries to maintain efficiency by means of a lower lfil combined with a smaller droptol.

Treatment of small matrices and pc-PAMG's coarsest level solver. For AIM matrices with up to 100,000 rows, we also tested a powerful sparse direct solver, namely PARDISO[21,22]. Our numerical tests have shown that our α-SAMG solver in its one- or multi-level variant is more efficient here than PARDISO if, roughly, the number of variables for the *original* matrix exceeds only 10,000. A(ny) powerful direct solver can, however, be used as a *coarsest-level solver* inside SAMG so that coarse-level matrices might have up to several thousand rows. This way, the performance of a good direct solver and (pc-P)AMG can be combined in an efficient way.

Overall approach (α-SAMG). In summary, we propose an automatic, adaptive ILUT parameter and solver switching strategy based on a special PAMG method, pc-PAMG, suitable for handling matrices arising in oil reservoir simulations based on AIM (and FIM). This approach is called α-SAMG in the following. The switching is accomplished by three types of adaptivity:

- For each D-class, the basic solver used is switched according to results of online solver testing. The results "automatically" depend not only on the magnitude (number of variables) of the matrix, but also on the physical complexity of the underlying model.
- The parameters lfil and droptol of the ILUT smoother employed are changed according to the convergence, timings and memory requirement history. In particular, it is checked based on various heuristics whether more or less robust settings shall be used for the actual and/or following matrices of the same D-class.

- The underlying pc-PAMG method adapts its coarsening and interpolation automatically to the coupling structure reflected by the matrix entries. Special rows (currently the ones strongly violating diagonal dominance) are marked and (currently) excluded from coarsening. Memory consumptions (grid, operator, interpolation complexities[11]) are continuously being adapted during a simulation run.

Benchmark Cases

In order to test our solver switching strategy, α-SAMG has been integrated into the simulator mentioned above. Detailed results are presented and discussed for eight different industrial benchmark cases:

- Model M1 is a small compositional simulation model with 7 hydrocarbon components used in the equation-of-state computations. The fluid composition changes with depth, the reservoir fluid exhibits dew points at the top and bubble points at the grid blocks closer to the water-oil contact. A gas recycling operation is simulated.
- Model M2 is a two-phase gas water model with about 200 wells and frequent abrupt changes in flow direction. The process simulated is a gas storage system with many cycles of gas injection and production.
- Model M3 represents a 3-phase, black-oil model with about 150 wells. The model contains local grid refinement and faults. This is a classical history match simulation model with production processes and typical well reactions due to excessive water-cut or gas-oil ratio.
- Model M4 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, because the natural fractures in this reservoir are modelled with a dual porosity approach applied to selected parts of the model.
- Models M5 and M6 represent a fine gridded, single well simulation model with more than 400 layers. Model M5 simulates the processes with a gas-water formulation, model M6 uses a 7-component EOS compositional formulation. The well is an extended-reach type horizontal well.
- Model M7 is a fairly large simulation model with faults, local grid refinement in an extended black-oil formulation, where the water phase consists of a variable salt concentration. With this model it is possible to trace movement of reservoir and injected waters with different salinities.
- Model M8 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.

General properties of these benchmark cases can be found in Tables 1 and 2. Here and in the following, “avr.” means average number, “max.” maximum, “min.” minimum, and “rel.” relative to the number of functions.

model	Type	points	avr. rows	max. rows
M1	compositional	6999	29821	69997
M2	gas-water	26536	28523	36200
M3	black-oil	69933	75488	93712
M4	black-oil	70742	84608	110201
M5	gas-water	265389	265795	269738
M6	compositional	265389	267788	317925
M7	black-oil + salt	678351	683935	699445
M8	black-oil	1103334	1147276	1332839

Table 1: Properties of the benchmark cases.

model	fn.	avr. doi	max. doi	rel. max. doi
M1	10	4,26	10,00	100%
M2	3	1,07	1,36	45%
M3	5	1,08	1,34	27%
M4	5	1,20	1,56	31%
M5	3	1,00	1,02	34%
M6	10	1,01	1,20	12%
M7	6	1,01	1,03	17%
M8	5	1,04	1,21	24%

Table 2: Further properties of the benchmark cases. “fn.” means number of physical functions, “doi” degree of implicitness (equal to the number of rows divided by the number of points).

Numerical Results

In the following, different aspects are analyzed in order to judge the efficiency and robustness of α-SAMG and to find ways to improve the switching mechanisms currently implemented even further. In particular, we show that α-SAMG is more efficient than ORILU[23], an adaptive ILU(k)-ORTHOMIN solver, originally integrated into the simulator employed. We discuss timings in comparison with several relevant solvers and smoothers, memory consumptions of the total simulation run as well as the linear solver part, and the ILUT and solver switching behaviour.

General Remarks. The benchmark cases react very differently during simulation due to their different “physical” nature. This is not only due to the different magnitudes and the general application classes (types) they belong to, but also due to the “physical complexity” as well as the amount of (effective discrete) ellipticity.

In this respect, models M5 and M6 are extreme since both are medium-sized and contain only one extended horizontal well. These models are interesting for testing the solver and parameter switching since they are (effectively discrete) hyperbolic. Hence, both can be solved very quickly by means of even a very cheap one-level method if (and only if) it correctly “surfs” along the characteristic directions.

The small gas-water model M2 and the small compositional model M1 are a bit extreme w.r.t. size because their matrices fit into (a 2 Mbytes sized L2-)cache in single precision, but not completely in double precision. Since ORILU is a mixed single and double precision solver, a “speedup” between 0.5 and 1 for α-SAMG compared to

ORILU would be achieved if both solvers are comparable despite precision.

These four models, M1, M2, M5 and M6, thus also show whether ORILU and the one-level variant of α-SAMG are comparable in a "fair" way.
The other models are intended to test especially the multi-level part of α-SAMG. Note, however, that M3, M4, M7, and M8 cannot directly be compared so that a "scalability study" (i.e. how close is α-SAMG to an O(N) method) based on the models available is not possible.

Robustness. For most test cases and at several stages, matrix properties can drastically change from one step to the next. In such a case, it sometimes happens that pc-PAMG (or the one-level method) does not converge with the ILUT parameters chosen. α-SAMG then adapts the settings and performs a second run (if necessary, this process would be repeated). This way, all matrix equations arising for all models could be solved without problems, proving the robustness of α-SAMG's concept.

Timings of α-SAMG compared to ORILU. Moreover, α-SAMG outperforms ORILU for each model (see Table 3). As discussed above, for the small models M1 and M2 and also the quite hyperbolic M5 and M6 a performance comparable to ORILU is very reasonable. In fact, α-SAMG is even slightly more efficient here. For the medium-sized models, M3 and M4, the performance of α-SAMG is already reasonably higher, and for the largest model, M8, with a speedup of more than 15 very convincing. M4 is due to the dual porosity feature more involved than the similarly sized M3, correspondingly α-SAMG performs a bit better for M3. Only M7 seems to be a bit poor considering its size. However, one should take into account here that M7 contains another physical unknown due to the salt component on one hand. On the other hand, it seems to be more hyperbolic. The discussion on M7 will be continued in the section on ILUT settings.

model	matrices	time ORILU	time α–SAMG	speedup
M1	941	930	928	1.00
M2	8120	5833	4489	1.30
M3	2007	6674	4159	1.60
M4	935	3205	2188	1.46
M5	4454	16822	14123	1.19
M6	3998	77477	65047	1.19
M7	473	14034	11604	1.21
M8	703	391070	25216	15.51

Table 3: Total simulator run times [sec] for ORILU and α-SAMG. The speedup values displayed refer to these total run times. The speedup of the linear solver part itself is higher.

Comparison to other solvers and different smoothers. Table 4 endorses the discussion on appropriate smoothers from above. Other smoothers, namely the parameter-free methods ILU, BILU and BGS defined above, are not as robust and efficient as the adaptive ILUT employed in α-SAMG. In addition, a pure ILU- or BILU-preconditioner does also work neither robustly nor efficiently (only the ratio for the best out of these two methods is shown in Table 4).

Comparison to α-SAMG's pure one-level variant (OL-α-SAMG). α-SAMG is at least as fast as OL-α-SAMG, see Table 4. In three cases, namely M1, M5 and M6, α-SAMG results - for the reasons (size or hyperbolic behavior) explained above - in a pure one-level method, see also Table 7. Interestingly, OL-α-SAMG is already slower for the small M2 than the full α-SAMG. Again, the fact that M4 features dual porosity is reflected also here to some extend: OL-α-SAMG performs better for M3 than for M4.

model	restr. α-SAMG	(B)ILU	ILU-pc-PAMG	BILU-pc-PAMG	BGS-pc-PAMG
M1	ML: 0.61	0.52	0.48	0.21	div
M2	0.87	0.31	0.29	0.45	0.32
M3	0.41	div	0.43	0.23	0.29
M4	0.83	0.25	0.57	0.41	0.53
M5	ML: 0.92	0.58	div	0.46	div
M8	0.45	div	div	div	div

Table 4: Total simulator run times for different preconditioners divided by respective simulator run time for α-SAMG (see Table 1). Accelerator always BiCGstab. "restr. α-SAMG" means α-SAMG restricted to OL-α-SAMG if α-SAMG does not result in a pure one-level method itself. Exceptions are marked by "ML" (pure multi-level variant of α-SAMG). "div" means unrecoverable solver problems during the simulator run.

Memory consumptions. Although α-SAMG employs a hierarchy for all models except of M1, M5, M6, the memory consumptions of α-SAMG are principally the same or even less than those of ORILU, as can be seen from Table 5. This is because of two reasons: the adaptive ILUT strategy slightly favors memory-cheap variants, i.e. a small lfil, if possible. On the other hand, the AMG method employs an aggressive coarsening strategy, so that coarser levels only add a small amount of memory. For concrete operator-complexities, memory consumptions of the linear solver part only, and ILUT parameters, see Table 6.

	total memory [MBytes]			avr. memory [Mbytes]		
model	ORILU	α–SAMG	factor	ORILU	α–SAMG	factor
M1	210	95	2.21	146	53	2.75
M3	108	99	1.09	102	73	1.40
M4	163	135	1.21	158	100	1.58
M5	261	274	0.95	260	168	1.55

Table 5: Peak memory requirements for the simulator runs; average requirements including all overhead for each Newton step; both for ORILU and α-SAMG. Values for M6 are nearly identical to M5. Both M7 and M8 can be run within 2 GBytes RAM.

	complex.		peak mem.		lfil		$\log_{10}$(droptol)	
model	min.	max.	min.	max.	avr.	max.	min.	avr.
M1	1.00	1.00	3.0	43.6	12.4	24	-6.2	-3.6
M2	1.00	1.38	7.1	28.7	4.7	16	-4.1	-2.2
M3	1.11	1.24	23.6	45.7	5.0	13	-2.6	-2.1
M4	1.00	1.35	19.4	53.9	4.1	7	-2.7	-2.0
M5	1.00	1.00	68.7	100.4	4.1	6	-2.7	-2.0
M6	1.00	1.00	69.3	131.6	4.0	12	-3.4	-2.0
M7	1.30	1.33	246.4	319.3	4.0	6	-2.7	-2.0
M8	1.15	1.46	432.0	923.3	7.2	16	-6.9	-3.0

Table 6: Characteristic values for runs with α-SAMG: Operator complexities (including finest-level matrix), peak memory requirements of the linear solver, ILUT parameters lfil and droptol. Minimum for lfil is always 4, maximum for droptol always 1e-2.

ILUT Parameter and Solver Switching behavior. Summaries of important statistics on the ILUT parameter switching as well as the solver switching can be found in Tables 6 and 7.

More detailed results on the development of the switching during simulation are presented for the models M4 and M8. M4 is the largest model, for which both one-level and multi-level solvers are employed during simulation. M8 is the largest model in the benchmark set presented, and at the same time the largest one for which only multi-level solvers are used by α-SAMG. Detailed information on the series of ILUT parameters resulting for models M4 and M8 are presented in Figs. 2 and 3. BiCGstab iterations needed by the solver chosen for each of the matrices are shown in Figs. 4 and 5.

In all benchmark cases, the average number of BiCGstab iterations is very reasonable, although for the runs with multi-level solvers less lfil than the average number of entries per matrix row is needed. Hence, the resulting smoother is less memory-expensive than an ILU(0) would theoretically be (which still would be very reasonable). High peaks of the iteration numbers are due to the effect described in the section on robustness above and Table 7.

Continuing the discussion from above for Model M7, one can see that the ILUT settings employed result in a rather cheap smoother. Obviously, M7 is not too complicated to be solved by means of ILU-type preconditioners. This should explain the fact that α-SAMG is only a bit faster than ORILU for M7.

Slightly more iterations seem to be necessary for M8, at least for the settings of the ILUT parameters as resulting from the current switching mechanism. However, one has to take into account here first that M8 has been run on a machine with 2 GBytes RAM only: α-SAMG's control mechanism has to restrict the maximum lfil here and switches to a droptol which seems to be smaller than really necessary, just adding superfluous run time. Second, although the lfil needed is a bit higher than for the other models, the smoother employed is still cheaper than ILU(0) in terms of memory consumptions. Hence, iterations can be slightly higher. Third, the model contains several faults and grid refinement. Both facts seem to distort favorable characteristic directions so that the ordering of the variables, as produced by the simulator, might suffer. Still, overall, α-SAMG is more than 15 times faster than ORILU for the large M8 model which demonstrates the high potential of α-SAMG for even larger models.

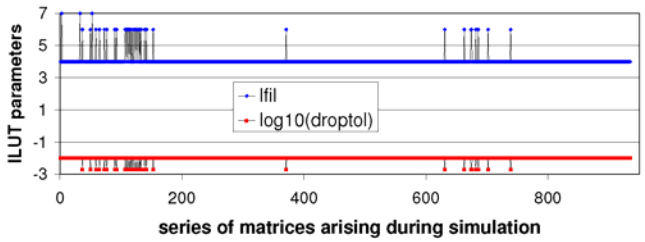

Fig. 2: ILUT parameters chosen by α-SAMG for M4.

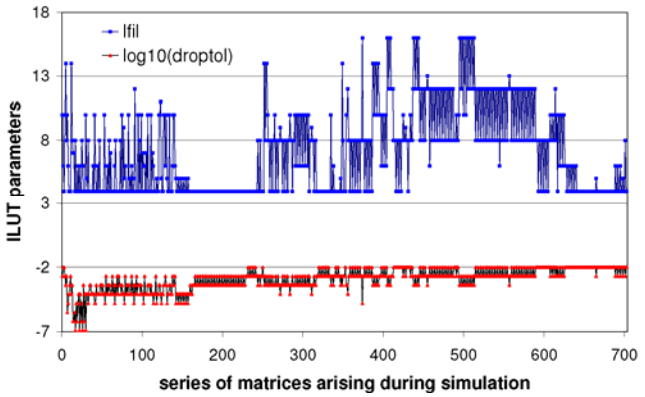

Fig. 3: ILUT parameters chosen by α-SAMG for M8.

	all: iters.		OL: iters.		ML: iters.		runs with	
model	avr.	max.	avr.	max.	avr.	max.	OL	ML
M1	30.7	186	30.7	186	0	0	941	0
M2	16.0	106	17.5	100	6.8	106	7008	1112
M3	7.6	28	0	0	7.6	28	0	2007
M4	11.1	52	20.3	52	5.0	12	374	561
M5	7.0	11	7.0	11	0	0	4454	0
M6	5.0	114	5.0	114	0	0	3998	0
M7	6.3	12	0	0	6.3	12	0	473
M8	16.4	155	0	0	16.4	155	0	703

Table 7: Number of BiCGstab iterations per matrix for runs with α-SAMG; number of iterations and runs separated into cases where ILUT-BiCGstab (OL) or ILUT-pc-PAMG-BiCGstab (ML) are used, respectively. Note that high peaks for iteration numbers are due to the fact that sometimes matrices have to be re-run, and iterations are summed up over all attempts then.

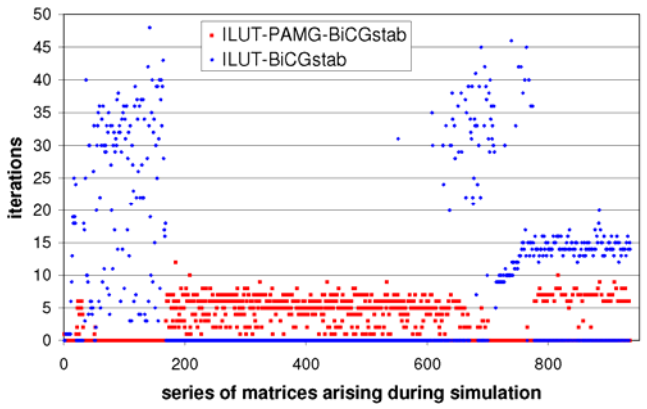

Fig. 4: Number of iterations of α-SAMG for M4. Both, one-level (depicted by blue dots) and multi-level (red dots) methods are chosen by α-SAMG. A dot on the zero line means that the respective other solver is used.

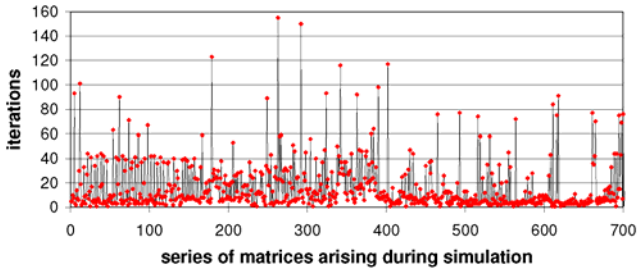

Fig. 5: Number of iterations of α-SAMG for M8. Only multi-level methods are chosen by α-SAMG.

Conclusions and Outlook

The results presented in this paper clearly indicate the efficiency and robustness of the new solution method α-SAMG proposed. For a wide variety of test cases from different application types – gas-water, black-oil, and compositional models – as well as different levels of physical complexity, α-SAMG performs better in terms of overall computing time as well as memory requirements than the original adaptive ORILU solver as well as other one-level and multi-level variants. The performance tends to be the better the larger and/or more physically involved the model considered is. For a reasonably complex and large black-oil model, namely M8, a speedup of more than 15 could already be achieved keeping memory requirements reasonably low at the same time. However, if the model is still of mixed elliptic-hyperbolic type, but features a larger hyperbolic behavior than M8, as is the case for some models presented, full advantage of multi-level approaches is expected to be seen for model sizes beyond 1 million grid points.

Possible further improvement of the switching strategy. Our strategy α-SAMG already takes D-classes into account. Special a-priori or online adjustments with respect to physical properties, such as application type (A-classes, including gas-water, black-oil, compositional) or physical complexity (P-classes, among them being concrete well models, dual-porosity dual-permeability, etc.) and level of ellipticity could help to further improve efficiency. In addition, further possibilities for improving the underlying AMG method are under investigation[3,4].

A parallel version of the approach proposed. A full MPI-based parallel version of α-SAMG's solver switching strategy is currently under development. In particular, the control mechanism has to be extended to take into account different numbers of processors, different characteristics (e.g. w.r.t. communication) of the parallel cluster, and so on.

However, the underlying pc-PAMG solver including a first parallel version of our ILUT smoother (ILUT with overlap) is already available in the parallel variant SAMGp[16] of the linear solver library SAMG. First promising results are given in Tables 8 and 9 for two typical matrices from M8, the largest model analyzed so far. We used lfil=8, droptol=1e-4, an overlap[d] of 1 and up to 4 processors. Runs on more processors do not really make sense here since the matrices only have about 1.1 million variables (see Table 1).

A detailed analysis and benchmarking of a full parallel variant of α-SAMG will be published in a subsequent paper.

	total run time			cycling phase only			total memory	
p	time	sp.	eff.	cycles	time	sp.	memory	factor
1	82.67			10	75.86		624.17	
2	47.79	1.73	0.86	9	39.24	1.93	347.80	1.79
4	33.61	2.46	0.61	10	24.40	3.11	184.64	3.38

Table 8: Results of parallel runs with SAMGp for a representative matrix of M8. "p" is the number of processors, "sp." the speed-up, "eff." the parallel efficiency (equal to "sp." divided by "p").

	total run time			cycling phase only			total memory	
P	time	sp.	eff.	cycles	time	sp.	memory	factor
1	94.89			12	88.13		613.94	
2	61.18	1.35	0.68	13	52.51	1.68	343.26	1.79
4	41.95	1.97	0.49	13	31.67	2.78	182.12	3.37

Table 9: Results for another matrix of M8, analogously to Table 8.

Acknowledgements

The authors would like to thank SMT Alps for permission to use the SURE simulator for the numerical experiments. They are grateful to Klaus Stüben and Arnold Krechel from Fraunhofer SCAI for fruitful discussions and support especially for the parallel tests.

[d] An overlap of 0 or more than 1 would just be slightly more expensive for these matrices. Also the level of overlap has to be controlled inside a parallel version of α-SAMG.

Nomenclature

D_p = phase molar density
k = permeability
q_p = production/injection rate
S_p = phase saturation
V_i = block volume
x_{pc} = molar fraction of component *c* in phase *p*
τ_{ij} = inter-block transmissibility
λ_p = phase mobility
Φ_p = phase potential
Δt = time step length
ϕ = porosity

Subscripts/Superscripts

c = component
i, j = blocks
p = phase
N = number of neighbors
P = number of phases

References

1. Thomas, G.W. and Thurnau, D.H.: "Reservoir Simulation Using an Adaptive Implicit Method," SPEJ **23** (Oct. 1983), 759.
2. Klie, H.: *Krylov-Secant Methods for Solving Large Scale Systems of Coupled Nonlinear Parabolic Equations*. PhD thesis, Dept. of Computational and Applied Mathematics, Rice University, Houston, TX (1996).
3. Stüben, K., Clees, T., Klie, H., Lou, B. and 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.
4. M., Klie, H., Wheeler, M.F., Clees, T. and 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.
5. Saad, Y.: "ILUT: A dual threshold incomplete ILU factorization," Numer. Lin. Alg. Appl. **1** (1994), 387.
6. 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.
7. 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** (1992), 631.
8. Saad, Y. and Schultz, M.H.: "GMRes: a generalized minimal residual algorithm for solving nonsymmetric linear systems," SIAM J. Sci. Stat. Comp. **7** (1986), 856.
9. 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.
10. Stüben, K.: "A review of algebraic multigrid," J. Comp. Appl. Math. **128** (2001), 281.
11. Stüben, K.: "An Introduction to Algebraic Multigrid," in *Multigrid*[12], 413.
12. Trottenberg, U., Oosterlee, C.W. and Schüller, T.: *Multigrid*, Academic Press, London, UK (2001).
13. Vanek, P., Mandel, J. and Brezina, M.: "Algebraic Multigrid by Smoothed Aggregation for Second and Fourth Order Problems," Computing **56** (1996), 179.
14. Clees, T.: *AMG Strategies for PDE Systems with Applications in Industrial Semiconductor Simulation*, Ph.D. thesis, University of Cologne, Nov. 2004; Shaker, Aachen, Germany (2005).
15. Stüben, K. and Clees, T.: *SAMG User's Manual, Release 22c*, Fraunhofer SCAI, Sankt Augustin, Germany (June 2005), http://www.scai.fraunhofer.de/samg.html.
16. Krechel, A. and Stüben, K.: *SAMGp User's Manual, Release 21z*, Fraunhofer SCAI, Sankt Augustin, Germany (Oct. 2005), http://www.scai.fraunhofer.de/samg.html.
17. Stüben K., Delaney P. and 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.
18. Wallis, J.R., Kendall, R.P. and Little, T.E.: "Constrained Residual Acceleration of Conjugate Residual Methods," paper SPE 13563 presented at the 8th SPE Symposium on Reservoir Simulation, Dallas, TX, Feb. 10-13, 1985.
19. Cao, H., Tchelepi, H.A., Wallis, J. and Yardumian, H.: "Parallel Scalable Unstructured CPR-Type Linear Solver for Reservoir Simulation," paper 96809 presented at the 2005 SPE Annual Technical Conference and Exhibition, Dallas, TX, 9-12 Oct.
20. Papadopoulos, A. and Tchelepi, H.: "Block smoothed aggregation AMG preconditioning for oil reservoir simulation systems," *Computing Laboratory Numerical Analysis Report* **03/04** (2003), Oxford University, Oxford, UK.
21. Schenk, O. and Gärtner, K.: "Solving Unsymmetric Sparse Systems of Linear Equations with PARDISO," Journal of Future Generation Computer Systems **20** (2004), 475.
22. Schenk, O. and Gärtner, K.: "On fast factorization pivoting methods for symmetric indefinite systems," Elec. Trans. Numer. Anal. **23** (2006), 158.
23. Brand, C. W. and 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.