XIX International Conference on Water Resources
CMWR 2012
University of Illinois at Urbana-Champaign
June 17-22, 2012

ADVANCED ALGEBRAIC MULTIGRID APPLICATIONFOR THE ACCELERATION OF GROUNDWATER SIMULATIONS

Peter Thum*, Klaus Stüben*

*Fraunhofer Institute for Algorithms and Scientific Computing (SCAI)
Schloss Birlinghoven, D-53754 Sankt Augustin, Germany
e-mail: peter.thum@scai.fraunhofer.de, web page: http://www.scai.fraunhofer.de/samg

Key words: Advanced numerical methods, linear solver, algebraic multigrid

Summary. The computationally most intensive process during a groundwater simulation is the solution of linear systems of equations. This can take more than 80 percent of the overall runtime. Hence, an efficient linear solver can save an enormous amount of runtime. However, it may strongly depend on the overall simulation environment which linear solver is the most efficient one for specific linear systems. In time-dependent processes, for instance, the type of the most suitable solver may even change over time. We present an advanced application of algebraic multigrid (AMG) which automatically adapts and optimizes its solving strategy, yielding a robust and very efficient process. This is achieved through two steering mechanisms. The first one controls the AMG method: it attempts to save computing time by reusing setup data from previous runs. The second one controls the switching between AMG and an alternative solver which is complementary to AMG. In groundwater simulation, a reasonable choice for this alternative solver is ILU-CG. The resulting solution process behaves like a "one-for-all" solver, at least from the user's point of view.

1 INTRODUCTION

The computationally most intensive process during a groundwater simulation is the solution of the linear systems, which can take more than 80 percent of the overall runtime. Hence, an efficient linear solver can save an enormous amount of the computational time.

Today, usually classical ILU preconditioned CG methods (PCG) are used to solve the linear systems. However, these methods are not of optimal complexity, which means that a doubling of the model size will result in much more than a doubling in computational time.

Algebraic multigrid methods use a hierarchical approach to solve the linear systems. Through this approach, they reach an optimal computational complexity. Hence, in general, AMG is highly superior to standard one-level methods such as PCG. However, there are two drawbacks: a relatively costly overhead (setup phase) and a memory requirement which is about twice as high as that for PCG.As a consequence of the setup overhead, AMG may lose its superiority in certain situations, for instance, for relatively small linear systems and/or strongly diagonally dominant matrices. In such situations, PCG may be faster than AMG. In fact, a solution by AMG may then simply be overkill. Since in complex transient linear and/or nonlinear situations the character of the individual matrices may change from step to step, it is a not a priory clear if and

when AMG will be faster than PCG. This depends on various details such as the discretization in time, the time stepping, and the complexity of the model considered as well as its size.

It is clear that, in general, we cannot expect a single solver to be optimal throughout a whole simulation. Hence, we introduce a "solver control" which provides an adaptive optimization of the solving strategy, i.e., depending on some given criteria, the most suitable solver (either AMG or PCG) is automatically and dynamically selected at run time. Additionally, whenever AMG is the currently "active" solver, AMG's setup overhead will automatically be reduced by reusing setup data from previous linear systems whenever reasonable.

The presented solver control is available as part of the SAMG solver library developed by Fraunhofer SCAI. Apart from efficient AMG-based solver modules, SAMG includes also various one-level solvers such as PCG as well as direct methods. The control switches between two predefined solvers and decides which one will be better suited for a linear system given.

Furthermore, the AMG-modules of SAMG are parallelized by state-of-the-art techniques, allowing making efficient use of today's multi-core computers and CPU clusters.

2 ADVANCED ALGEBRAIC MULTIGRID FOR GROUNDWATER SIMULATION

The basic equation in groundwater flow simulation in the steady state case writes

$$-\nabla \cdot K \nabla p = W, \tag{1}$$

where p is the hydraulic head, K is a hydraulic conductivity tensor, and W is a source/sink term. Considering common discretization methods like finite elements differences or finite volumes, this elliptic partial differential equation (PDE) usually leads to an M-matrix. M-matrices are perfectly suited for the algebraic multigrid method (AMG).

2.1 Brief introduction to algebraic multigrid

AMG accelerates the convergence of the iterative solution of large sparse linear systems by creating a hierarchical process. The motivation for such methods arises from the inability of classical one-level iterative solvers to efficiently reduce the approximation error from iteration to iteration. Classical one-level iterative solvers typically experience a slow convergence, as they cannot handle all error frequencies effectively. To be more specific, the high frequency error components are reduced much faster than the low frequency components. AMG constructs a sequence of lower dimensional systems only based on the matrix itself. Thus, no information about the grid or physical geometry is required beyond what is already contained in the coefficient matrix. On the coarser levels, the low error frequencies of the fine level become higher frequency and, hence, each error frequency can be reduced efficiently at an appropriate level. If combined with some reasonable smoothing process, the lower dimensional systems can be used to correct the higher dimensional ones.

Algebraic multigrid methods proceed in two phases for solving a linear system of equations: a setup phase in which the hierarchical components are set up, and a cycling phase in which the linear system is iteratively solved. Due to its hierarchy, AMG provides optimal complexity,

which means that the time for solving a linear system depends on its size only linearly. AMG can be implemented as a plug-in solver, provided that the underlying matrix satisfies certain properties (like sparsity and ellipticity).

Although the development of AMG goes back to the early eighties, it still provides one of the most efficient, and notable robust, algebraic methods to solve elliptic problems[1]. One computational advantage of this process is that only the coarsest system requires a direct solution and the operations on all the other levels are simply matrix-vector multiplications. The only drawbacks of AMG are the before-mentioned overhead in terms of the setup phase and its increased memory requirement.

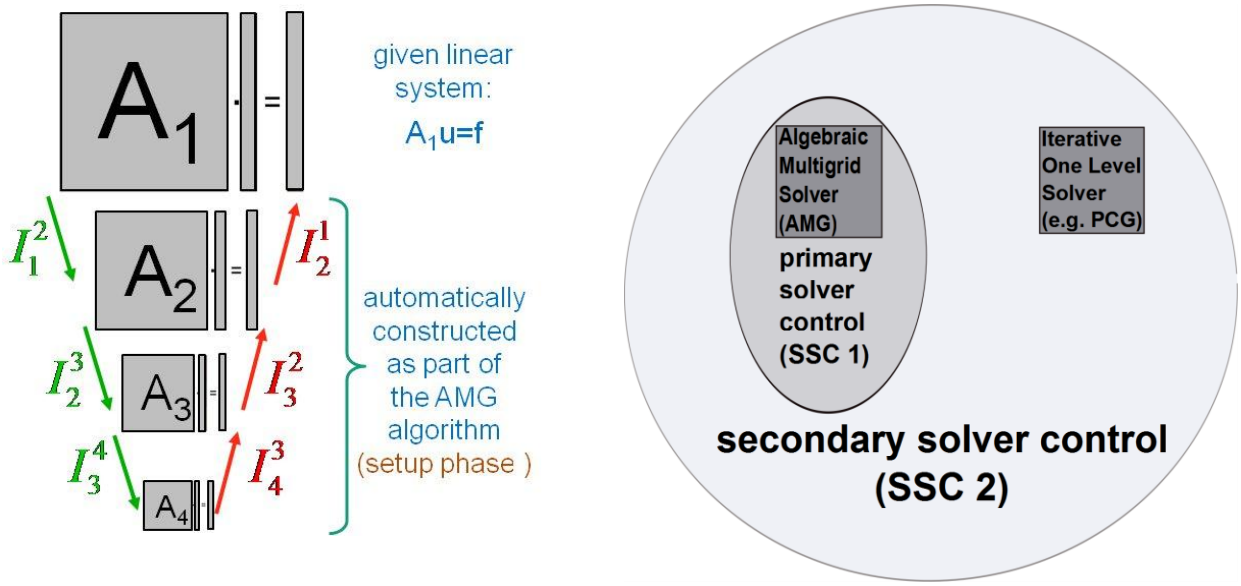

Figure 1: Algebraic multigrid hierarchy (left) and SAMG solver control mechanism

2.2 The SAMG solver control

The SAMG solver control (SSC) is a framework, which attempts to optimize its solving strategy for classes of similar linear systems automatically. The goal of this is to obtain a "one-for-all" solver, at least from the user's point of view.

The SAMG solver control consists of two control mechanisms: One solely for the AMG method used (primary control, SSC1) and one for the switching between solvers (secondary control, SSC2), see Figure 1.

- SSC 1optimizes the AMG performance by analyzing the solver behavior of previous time steps and reuses parts of the old solver setup.
- SSC 2 analyzes the behavior of the selected solver and switches to a different pre-selected solver (e.g. iterative one-level), if this turns out to be more efficient.

SSC1 is able to analyze AMG's behavior of previous linear systems and (automatically) detect if setup information of previous runs can be reused in order to minimize overall runtime[2].

The reuse of setup information is possible because subsequent linear systems in non-linear and/or transient simulations typically change only slightly regarding their algebraic properties. This way SAMG is able to reuse parts or all of previous expensive setup phases whenever possible, which significantly reduces overall runtimes.

SSC2 switches between two different predefined solvers. These two solvers should complement each other so that runtime can be saved if the properties of the linear system change during a simulation. In groundwater simulation, two suited solvers are AMG and PCG. Considering transient simulation with very small time-steps, for instance, the linear systems typically get very diagonally dominant. Hence, ILU-CG (PCG) solvers are very effective since they usually need only very few iterations to find a solution. In such cases, AMG is overkill since an iteration of AMG already costs as much as 2 to 3 ILU-CG iterations. On the other hand, considering large linear systems, big time-steps, or steady-state simulations AMG usually turns out to be considerably faster and more robust than PCG.

In order to ensure an efficient solving approach, the secondary controlSSC2takes the following aspects into account to decide whether AMG or PCG shall be used:

- memory requirement,
- runtime,
- convergence rate,

where the convergence rate is defined as r= $\left(\frac{r_{out}}{r_{in}}\right)^{\frac{1}{i}}$ where r_{out} denotes the output residual, r_{in} the input residual, and *i*the number of linear iterations.

SSC2 keeps track of the performance of the two linear solvers. If necessary, it switches between solvers. Hence, considering a transient model that is most efficiently solved with just a fixed one of the solvers, the control mechanism will detect this at the expense of a small overhead (up to a few percent, say). However, for models with matrix properties changing strongly during the simulation, the solver control might become faster than either one of the solvers solely, since the role of the solvers in terms of better efficiency might change during the simulation.

Especially for simulations with many linear systems, runtime can usually be reduced. We will investigate the performance of the SAMG solver library on several industrial relevant models for the popular simulation codes MODFLOW 2005 and FEFLOW. More details of the models can be found in previous publications[2, 3].

In the following, we refer to SSC if both control mechanisms, SSC1 and SCC2, are activated, and we refer simply to SAMG if neither of the control mechanisms is activated (i.e., plain AMG is used).

3 NUMERICAL EXPERIMENTS

3.1 Linear steady-state

For steady state models, SAMG is usually faster than common PCG solvers by one to two orders of magnitude. The reason for this is the relatively weak diagonal dominance of the linear systems, which effects PCG's convergence negatively. AMG's convergence, on the contrary, is nearly independent of the diagonal dominance. Hence, the performance gain can be enormous. Due to AMG's optimality, the performance gain increases with increasing model size, see Figure 1. Since in the case of linear steady-state models only one linear system has to be solved SSC1 is not used. SSC2 is also deactivated due to the superior behavior of SAMG.

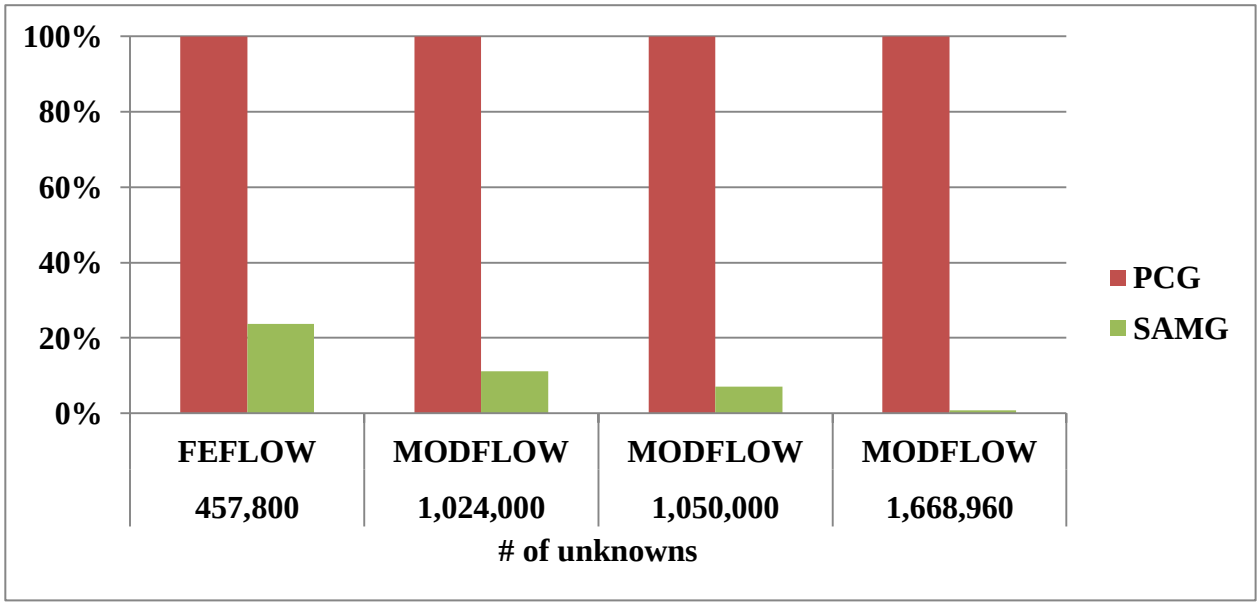

Figure1: Steady state models: Runtime comparison between PCG and SAMG

3.2 Nonlinear steady-state

Considering non-linear problems, the performance gain depends not only on the size of the linear system but also on the linearization method used and on the strength of the nonlinearity. Highly nonlinear models usually require the linear systems to be solved much more accurately than in the case of weakly nonlinear models. As a rule of thumb, the more accurately the linear system has to be solved, the higher the performance gain of SAMG compared to PCG will be. The reason for this is that one AMG iteration costs as much as 3 to 5 PCG iterations. Hence, if PCG only needs two iterations to reach the required accuracy, say, AMG is not able to beat it.

For non-linear models, more than one linear system has to be solved. It pays often off to reuse setup data from previous runs. Hence, we activate SSC1 for non-linear steady-state models.

Model	Cells/Unknowns	PCG	SSC1
1a	158,550	1	0.06
1b	158,550	1	0.26
2	1,476,225	1	0.43
3	1,728,000	1	0.24

Table1: Non-linear steady state MODFLOW models: Runtime comparison between PCG and SSC1. Runtimes scaled so that PCG's runtime is always 1.

Table 1 shows the results for four non-linear steady-state models. The models were set up for MODFLOW 2005, which uses Picard iteration for the linearization of the system. Models 1a and 1b differ only in the well properties and hydraulic conductivities.

3.3 Transient case

For transient simulations, we use SSC=SSC1+SSC2 which means that SAMG automatically tries to find the better of the two solvers in terms of runtime and robustness, and that AMG's setup will be reused if possible. This "full control" is reasonable for transient cases since it is not a priori clear whether PCG or AMG is suited best. For instance, AMG is perfectly suited for linear systems corresponding to relatively large time steps, or for systems that need a high accuracy, particularly in the non-linear case. PCG, on the other hand, tends to win for very small time-steps.

Figure 2 shows the results of PCG, SSC1 alone and the combination of SSC1 and SSC2 for various models. Model 1 makes use of an automatic time stepping and SSC is the fastest method the reason being the changing properties and time step lengths of the model over time.

Model 2 uses a relatively big constant time step length, which usually is beneficial for AMG methods. The PCG method is not shown here since it did not converge properly and led to a wrong simulation result for this model. SSC is able to detect this and decides to use the AMG solver, which – due to the small control overhead - results in a small overhead of 3%compared to SSC1 alone. Model 3 uses an automatic time-stepping, with many very small time-steps. Hence, overall, PCG is the best-suited solver in this case. Again SSC detects this at the expense of a small control overhead of 12%. The fourth as well as the fifth model use both predefined time steps of variable length. For both models, SSC1 turns out to be faster. As before, SSC detects the best solving process at a small control overhead of 12 and 3%, respectively.

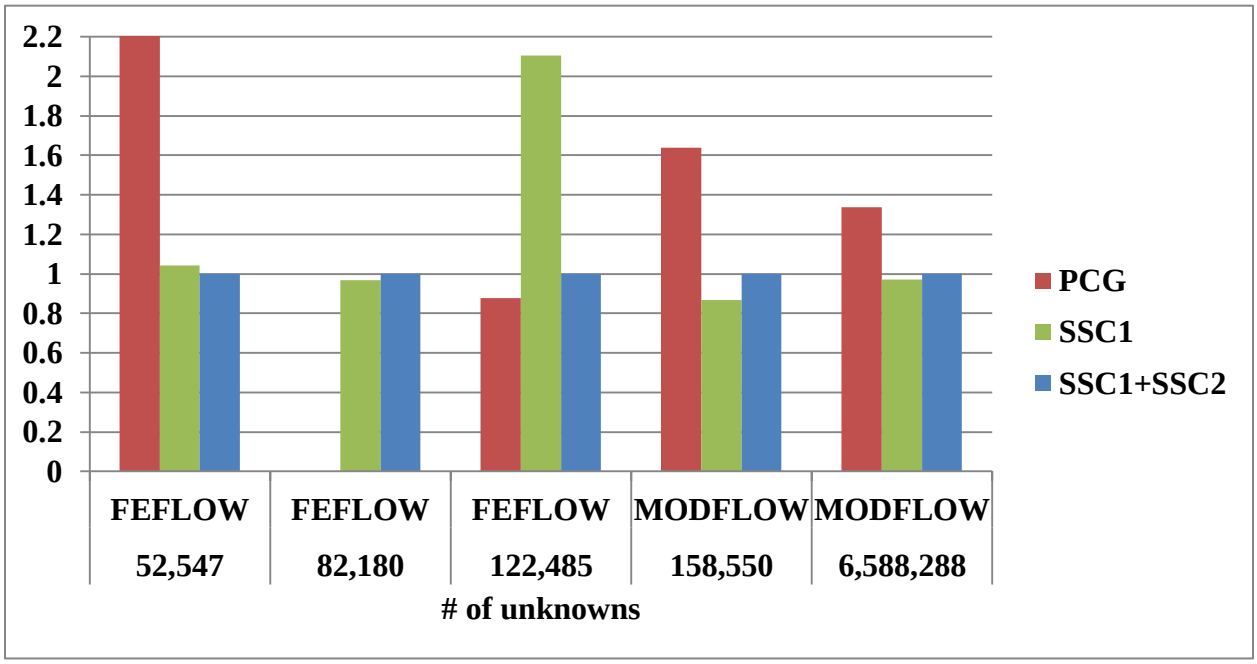

Figure 2: Runtime comparison for transient models between PCG,SSC1, andSSC 2

3.4 Parallel Performance

When investigating parallel performance one often uses the notion parallel speed-up. Parallel speed-up on p threads is defined as the ratio of the execution times needed on one thread and on p threads. The higher the parallel speedup of a program is the better its parallel performance will be.

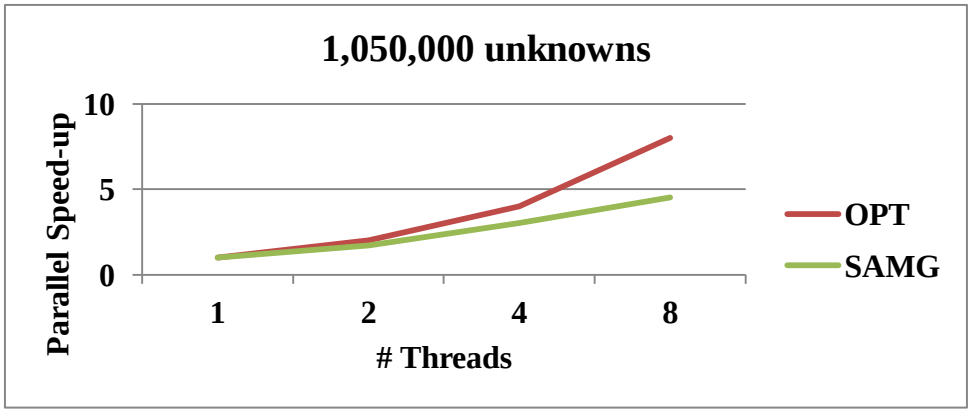

Figure 3: Parallel speedup of SAMG if applied to a typical linear system extracted from a MODFLOW simulation. The ideal (OPT) parallel speed-up is also drawn

For demonstration, a medium sized model was chosen and a single matrix was extracted. The model has been computed on up to 8 cores on an Intel Nehalem (2x4 cores, 24 GB Memory) computer. Figure 3 shows the corresponding parallel speed-up. For comparison, the ideal speedup curve is drawn. The SAMG speedup obtained on 8 cores is around 4.5. This is not yet optimal but close to the best possible due to the limited memory bandwidth of the multi-core architecture. The limit applies to virtually all numerical solvers using of sparse matrix-vector operations and is not specific to AMG.

4 CONCLUSIONS

Algebraic multigrid methods (AMG) are very well suited for steady-state models and transient models with big time-steps. In our test cases, up to 99 percent of the solving time could be saved. In transient or/and non-linear models, SAMG's solver control SSC1attempts to reuse AMG's setup data which to a good deal compensates for AMG's relatively high setup cost.

Nonetheless, there are situations where ILU-CG (PCG) solvers are more beneficial than AMG even with SSC1. This is, for instance, the case for models using very small time-steps and/or very low accuracy requirements. However, if a model makes use of various time step lengths and eventually uses an adaptive time stepping strategy, it is not a priory clear if AMG or PCG will be the faster solver. In this case, the automatic SAMG solver control (SSC) is able to detect the best-suited solver with (at most) little overhead. In case of strongly changing matrix properties and time step sizes, SSC might even be faster than either AMG or PCG. The results presented were obtained using the MODFLOW and FEFLOW groundwater simulators. However, such benefits will carry over to other (groundwater) simulators with similar discretization techniques based on the pressure equation (1).

REFERENCES

[1]K. Stüben, "An Introduction to Algebraic Multigrid", Appendix in the book „Multigrid" by U. Trottenberg; C.W. Oosterlee; A. Schüller, Academic Press, pp. 413-532, (2001).

[2] P. Thum, W. Hesch and K. Stüben, "Lmg2: accelerating the samg multigrid-solver in mudflow", *MODFLOW and MORE Conference Proceedings*, Colorado School of Mines, Golden (Co), USA, (2011).
http://www.scai.fraunhofer.de/fileadmin/download/samg/paper/modflow11.pdf

[3] P. Thum, "One for all – the new samg solver control within feflow",*2nd international FEFLOW User Conference*, Potsdam (Germany), (2009).
http://www.scai.fraunhofer.de/fileadmin/download/samg/paper/feflow.pdf