One for all – the new SAMG solver control within FEFLOW

Peter Thum
peter.thum@scai.fraunhofer.de
Fraunhofer Institute SCAI, Germany

ABSTRACT: The use of an efficient linear solver is very important to reduce the overall runtime of a flow simulation. However, if a linear solver will be very fast e.g. in case of a time dependent model it does not have to be efficient in case of a steady-state model. The efficiency of the linear solver may even change during a simulation run. Hence, there is the need to develop a solver control which chooses an appropriate linear solver for each respective situation. This paper describes the new SAMG solver control available with FEFLOW version 5.4. Numerical results are presented for a range of industrially relevant models, demonstrating the robustness and efficiency of the new solver control.

INTRODUCTION

In today's groundwater simulation the complexity and size of models has increased reflecting the advancement in computer hardware as well as parallel programming techniques and software memory management. In hand with this also the simulation time increases. Since the amount of time spend for the solution of linear systems within a simulation can be up to 50 per cent of the whole simulation run, optimizing the runtime of the linear solver has a major impact on the whole simulation time.

This paper describes the improvements of the SAMG solver package. The package is now included within FEFLOW without the need of further licensing, and provides a solver switching, which chooses the optimal solution strategy automatically. Hence, the new SAMG solver package is now able to serve as a "one-for-all" linear solver.

PROBLEM DEFINITION

The SAMG (algebraic multigrid methods for systems) package is a library of subroutines for the highly efficient solution of large linear systems of equations with sparse matrices. In particular, it includes algebraic multigrid (AMG). AMG is a, so-called, optimal solver, which means that the computational cost depends linearly on the number of unknowns. However, AMG cannot be applied to all types of systems of partial differential equations. Hence, the performance of AMG methods strongly depends on the properties of the linear system to be solved for.

AMG proceeds in two steps. In a first step, so-called, *setup* hierarchies of variables as well as so – called transfer operators are imposed. The second step, so-called, *cycling* then applies these information and operators in order to solve the linear system.

In previous versions of the SAMG solver package for FEFLOW already a reuse of AMG's setup phase was introduced. The reuse of the setup helped to save a lot of computing time. However, SAMG was not the fastest solver for each situation. In particular, transient models with an automatically controlled time-stepping by FEFLOW's adaptive predictor-corrector technique could often be solved very quickly with the FEFLOW-PCG solver. On the other hand, as long as nearly steady-state models have to be solved for each time step, the SAMG package often turned out to be superior.

Due to the situation dependent performance of the former SAMG solver package, the idea was to create an automatic solver control. The control shall switch between different linear solvers and in this way find an optimal solving strategy in terms of robustness and computing time for each model. Hence, it should turn out to be a "one-for-all" solver and thus simplify the work of the FEFLOW user.

DESCRIPTION OF THE NEW SOLVER CONTROL

The new solver control should be efficient and robust. In particular, the overhead due to finding an optimal linear solver for a specific situation should be as small as possible. Hence, we only switch between two different linear solvers. The switching is applied between a robust solver which tends to

be fast in case of small time steps and a solver which tends to be superior for nearly steady-state simulations.

The PCG method was the default solver in previous FEFLOW versions, due that it turned out to be a very robust solver. PCG denotes either modified incomplete Cholesky factorization preconditioned conjugate gradients (ICCG) in the symmetric case, or incomplete lower-upper factorization (ILU) preconditioned biconjugate gradient method ILU-BICGSTAB for unsymmetrical systems.

Due to its robustness and fastness for small time steps, we choose the ILU-BICGSTAB solver as one of the solvers to be switched between. The implemented ILU-BICGSTAB solver is the one available with SAMG solver library. In comparison to the ILU-BICGSTAB solver of previous FEFLOW versions it is fully OpenMP parallel. In hand with this, the new method is superior compared to the old one in terms of runtime on state-of-the-art multi-core computers. However, since the employed ILU-BICGSTAB is an inherently unsymmetrical method, it might be slower than the FEFLOW ICCG for symmetrical linear systems.

Due to the fastness in the case of nearly steady-state simulations and its robustness even for very unstructured grids, we chose SAMG's multigrid method as our second linear solver for the new solver control. To be more specific, we chose AMG-BICGSTAB. This method is an algebraic multigrid preconditioned stabilized biconjugate gradient Krylov method using standard interpolation and Gauss-Seidel smoothing. It is of linear complexity. Hence, especially for very big models this solver should turn out to be highly efficient. Moreover, this method is also OpenMP parallel.

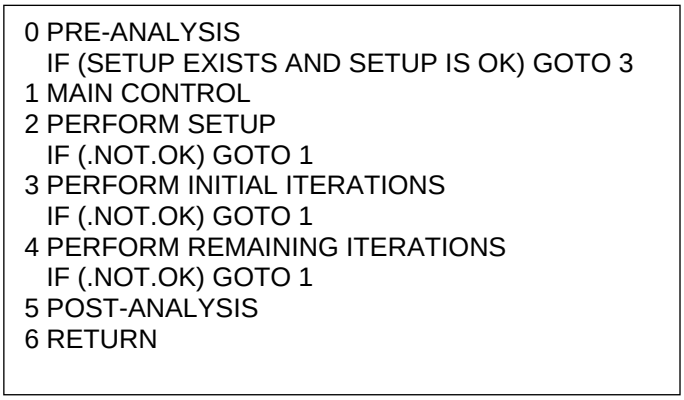

Figure 1: Sketch of the SAMG solver control algorithm

In order to ensure an efficient solving approach, the automatic solver control decides upon various criteria if the AMG or the PCG method shall be used.
The criteria are:

- Memory restrictions of the computing architecture
- Errors of the linear solvers
- Runtime
- Convergence Rate - The convergence rate is defined to be: r=(R_out/R_in)^(1/i), where R_out denotes the output residual, R_in the input residual, and i the number of linear iterations

These criteria are considered in the main control, post- and pre-analysis, see Figure 1.

Figure 1 shows a sketch of the control algorithm. Initially, the solver control performs a pre-analysis to decide upon the various properties of the linear system which of the two employed solvers shall be activated. The second step proceeds as follows. If we do not have an appropriate setup, the main control routine will be called. Otherwise, we start with the performance of the initial iterations (3).

The main control keeps track of the performance of the integrated linear solvers. If necessary, it switches to a different solver. If the main control switches the solver a new setup will be performed. During the setup all components required by the linear solver are set up. To be more specific, considering PCG the factorization of the matrix is computed, in terms of AMG its setup phase

respectively. The set up components might also be reused for the following linear systems to be solved for.

If an appropriate setup is available, the control will start the initial iterations. After performing these iterations, the situation is analyzed by the solving control. If the convergence rate turns out to be sufficiently well and no other exceptions occur we will proceed with the remaining iterations. In case of an unsatisfying performance or error during the initial iterations, we return to the main control routine. At the end of the solver control routine, the post-analysis recapitulates the overall performance and develops guidelines for further linear solver runs.

The two solvers integrated into the new SAMG solver control shall complement one another perfectly. Due to the splitting of the cycling phase into two subsets and the analysis after the initial iterations, an insufficient solving of one of the solvers is detected very fast. Hence, the overhead due to deficient attempts is reduced.

Considering a model which is efficiently solvable with one of the solvers, there should be only a small overhead. Considering models which change their properties during the simulation, the new solver control might even be faster than one of the solvers solely, since the efficiency of one solver might change during the simulation.

NUMERICAL EXPERIMENTS

In this section, we examine the performance of the new SAMG solver control (SSC) on a range of industrially relevant test cases. In particular, we compare the Performance of the SSC to the performance of the old SAMG solver package, denoted with AMG and the commonly employed preconditioned conjugate gradient (PCG) method.

Test cases

TRANSIENT
This model is an unsteady 3D regional finite-element flow problem. The mesh at a moderate resolution consisting of 221,210 pentahedral prismatic elements and 122,485 nodes is locally refined, well-formed and the parameter contrast remains moderate (conductivity ranges over five orders of magnitude). It can be considered as a typical 3D transient finite-element groundwater model used in practical water resources simulations. Of specific interest here are boundary conditions applied to rivers and pumping wells that possess a short-term dynamic (e.g., pumping capacity changes each day). The time-steps are automatically controlled by FEFLOW's adaptive predictor-corrector technique.

BASIN
This test refers to a 3D large-scale basin flow model. The 123,726 element pentahedral prismatic mesh with a moderate resolution has to incorporate a number of faulty zones, which leads to a vertical distortion of the prisms along these locations. The model is transient; fixed time-steps of 1 day are used. The parameter contrast is three orders of magnitude.

CROSS-SECTION
This model is a cross-sectional 2D problem which has a fully unstructured locally refined triangular mesh. The problem models an aquifer-aquitard system with heterogeneous distribution of conductivity and storativity. We consider the steady-state simulation with 903,872 elements and 457,800 nodes.

REAL1
This is a real world model. It makes use of a 2D triangular mesh with 52,547 nodes and 104,456 elements. The characteristic is that it is a combined groundwater flow and mass transport simulation. Therefore, it makes use of two, so-called, states of the linear solver. This means, in particular, that the solution strategy for the mass transport and the one of the flow transport are solved fully independent of each other. It turned out to be very important to proceed in this way since the systems of partial differential equations describing the flow and the transport respectively are very different. This model also makes use of FEFLOW's automatic time-stepping.

REAL2

We also examine another real word model. This model is a 3D transient groundwater flow. It uses five layers with 108,846 nodes and 178,420 pentahedral prismatic elements. Again, the automatically controlled time-stepping by the adaptive predictor-corrector technique is employed.

Numerical Results

In this section, we examine the performance of the new SAMG solver control (SSC) on the basis of the models introduced above. The performance of SSC is compared to the old solver package, denoted with AMG and the commonly employed FEFLOW preconditioned conjugate gradient (PCG) method.

Most of the models considered are flow problems. Due to the discretization, the linear systems which arise within the flow simulation are all symmetric. This symmetry is exploited by FEFLOW. In particular, the PCG solver for the flow equations only deals with the symmetric part of the linear systems.

In contrast, SAMG's ILU-BICGSTAB solver, as well as SAMG's AMG-BICGSTAB solver, always deals with the full linear system. Hence, if the FEFLOW-PCG solver works perfectly, we will see a relative underperformance of the SAMG build-in ILU-BICGSTAB solver of up to 50 percent on a single core. Since the SAMG ILU-BICGSTAB solver is OpenMP parallel this will not show up with this impact on state-of-the-art multi-core computers.

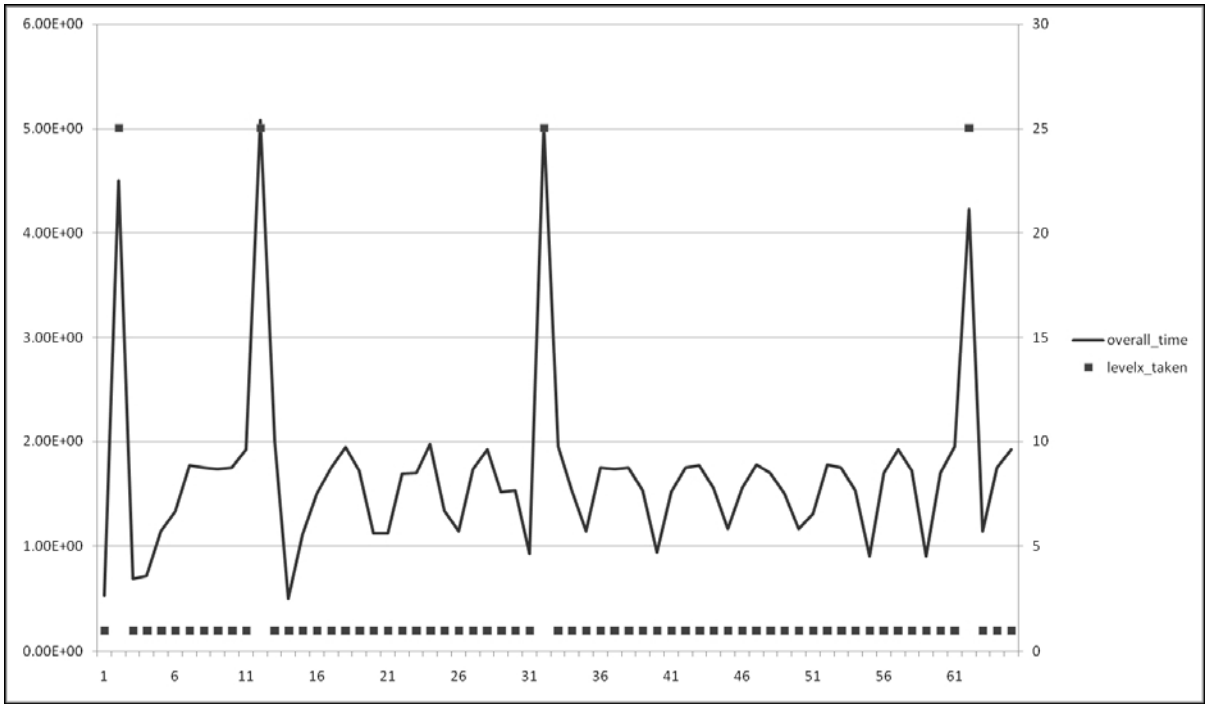

Figure 2: Illustration of the control mechanism for TRANSIENT (cutoff)

Figure 2 illustrates the work of the SSC for the REAL2 model. For this model, the ILU-BICGSTAB method turns out to be superior. This is most properly caused by the sufficiently small time steps introduced by the automatic time stepping employed for this model. The dots in the figure show the method used. 1 denotes the ILU-BICGSTAB, 25 the AMG-BICGSTAB solver respectively (right scale). The black line shows the runtime of the SSC. It increases if AMG-BICGSTAB is used. This increase is detected by the SSC. Hence, the SSC sticks to the ILU-BICGSTAB solver. The figure shows that the testing of the AMG-BICGSTAB method is performed several times. This is done in order to ensure that the behavior of the solvers does not change during the run. The time between the tests will be increased linearly if the other method remains inferior to the current one.

Figure 3 shows a cutoff of the SSC behavior during the flow simulation of the REAL1 model. For the part shown in the Figure, the AMG-BICGSTAB method turns out to be superior. The dashed line gives information on the number of iterations the setup of the AMG-BICGSTAB method has been reused for. We see that there were only three setups performed within about 330 calls of the SSC during the time shown by the figure. The gray line shows the run time for one call to the control routine. We see two things: First of all, we find that for longer time steps, indicated by the black line also the run time of the control routine rises. As a second thing we figure out that a new setup cost some extra time.

Another mechanism of the SSC also shows up in Figure 3. Sometimes a new setup has to be done if the currently employed solver does not work sufficiently anymore. This is detected by the convergence rate comparison. Hence, considering the third performed setup shown in the figure which was initiated by the convergence rate comparison, we see a significantly lower run time of the following control calls.

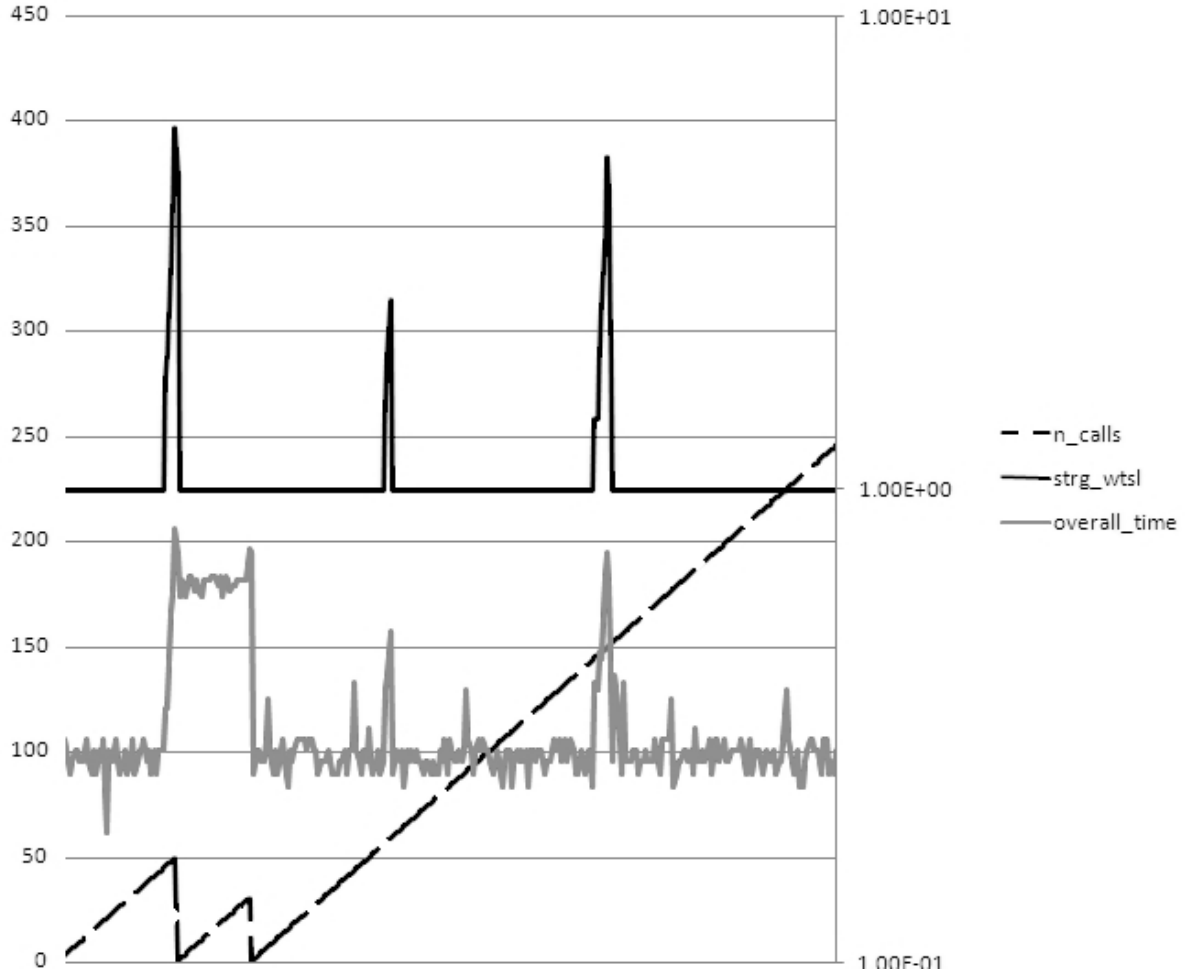

Figure 3: Illustration of the control for the REAL1 model (only flow) (cutoff)

The REAL2 model shows a considerable out performance of the FEFLOW-PCG solver, see Table 1. This out performance is most properly caused by two things. First of all the dynamic time stepping employed in the model leads to linear systems which are easily solvable by ICCG and ILU-BICGSTAB. The second thing is that the FEFLOW-PCG solver employed in this case is the ICCG method, which fully exploits the symmetry of the linear system. The SSC, in contrast, makes use of the ILU-BICGSTAB solver which does not exploit the symmetry. Table 1 clearly shows that the overhead due to the inefficient handling of the symmetry is 17 per cent while the overall overhead of the Solver control is only 19 per cent. This leads to the conclusion that the overhead due to the control is only 2 percent.

AMG	PCG	ILU-BICGSTAB	SSC
1.81	1.00	1.17	1.19

Table 1: Comparison of the relative run times for REAL2

Table 2 shows the run times for the other models described above. Considering the TRANSIENT model we find a clear out performance of the PCG solver compared to AMG. This is again most properly due to the dynamic time stepping employed for this model. The overhead for the SAMG solver control origins again from two things. For this model the PCG runtime turns out to be superior for each time step. Since, the considered linear systems are symmetric; SSC's ILU-BICGSTAB is strongly inferior, like we already figured out for the REAL2 model. A second, but rather small overhead is caused by the fact that the SSC tests the AMG-BICGSTAB methods several times in order to ensure its insufficiency for the model.

Model	AMG	PCG	SSC	Overhead
TRANSIENT	3489s	1456s	1658s	14%
BASIN	701s	*	724s	3%
CROSS-SECTION	24s	101s	24s	0%
REAL1	73230s	8129s	7801s	-4%

Table 2: Run time comparison; * denotes that this method failed

Considering the BASIN problem, the PCG solver completely failed. It was impossible to get convergent solutions. Continuing the simulation in time, after a small number of time-steps, the solution became fully instable. In contrast to the PCG, the AMG showed much better computational properties. AMG was able to solve the problem. Although the residual error could not be reduced to the given error criteria, the simulation could be successfully continued in time with stable and sufficiently accurate solutions. The same is true for the SAMG solver control. Therefore, the overhead of 3 percent can be fully traced back to the testing of the PCG solver, see Table 2.

The CROSS-SECTION model is a steady-state simulation. Hence, the control routine is called only once. The routine decides upon the properties of the matrix to choose the AMG-BICGSTAB solver. Therefore, the runtime for AMG and the control routine do not differ, see Table 2. For this model, the PCG is clearly not efficient. It was not able to reach the error criterion within the predefined number of iterations.

The REAL1 model shows that the control might even be faster than one of the methods solely. For the REAL1 model the AMG performs poorly with respect to run time. However, the convergence rates are very good. The bad run time of the AMG method can be explained by the overhead which is due to the setup and also the symmetry of the system, which is not exploited by AMG. The performance of PCG compared to AMG looks on a first sight highly superior. However, taking a closer look to PCG's performance we figure out that the PCG solver sometimes needs many iterations. Hence, AMG is superior for certain linear systems of the simulation. Due to the intelligent solver switching of SSC, we can exploit this fact. This results in a considerably good runtime.

Conclusion and Outlook

We investigated the performance of the new SAMG solver library, which is available with FEFLOW version 5.4. The results indicate the efficiency of the new solver control. In particular, the overhead due to the control mechanism showed to be bounded by five percent.

However, due to the use of the unsymmetrical variant of PCG in the SAMG solver control the SAMG-PCG solver might be up to 50 percent slower than the FEFLOW-PCG solver for flow calculations on a single core. On the other hand, the employed solvers in the SAMG solver control are OpenMP parallel, which leads to performance gains on state-of-the-art computing architectures. Hence, the overall performance on state-of-the-art multi-core machines turned out to be fairly well.

For some models, the control even turned out to be faster than the other solvers considered. This can be explained by the properties of linear systems. During a simulation, the systems can drastically

change and lead to inefficiencies of the currently employed linear solver. In such a situation, the control is able to detect the lack of performance and switch to another solver.

A future task is to replace the ILU-BICGSTAB solver in the case of symmetric linear systems by a parallelized variant of the incomplete Cholesky factorization. This modification shall result in an efficient linear solver for all models. The modification is planned to be released within one of the next patches of FEFLOW.

REFERENCES

Stüben, K., Algebraic Multigrid (AMG): Experiences and comparisons, Appl.Math. Comp. 13 (1983), 23-56
DHI-WASY GmbH, White papers Vol.III
DHI-WASY GmbH, unpublished