LMG2: Accelerating the SAMG Multigrid-Solver in MODFLOW

Peter Thum[1], Wayne Hesch[2], Klaus Stüben[1]
[1] *Fraunhofer Institute for Algorithms and Scientific Computing (SCAI), peter.thum@scai.fraunhofer.de, klaus.stueben@scai.fraunhofer.de,St. Augustin, Germany*
[2] *Schlumberger Water Services, WHesch@slb.com, Waterloo, Ontario, Canada*

ABSTRACT

This report documents the new Link-AMG (LMG2) interface that links the USGS-MODFLOW-2000,2005 groundwater flow model to an Algebraic Multigrid Solver, SAMG. SAMG is a very mature program that is used for the efficient solution of linear systems of equations in many commercial simulation codes. The LMG2 interface provides some distinct advantages compared to former LMG versions. The main advantages are an enhanced memory management, the ability to re-use setup data as well as parallel computing support for an efficient use on state-of-the-art multi-core machines. LMG2 outperforms PCG and GMG especially for large grids and highly variable conductivity fields. For the applications in this paper we demonstrate that LMG2 is up to 100 times faster than PCG and up to twice as fast as and even faster than GMG on state of the art multi-core machines.

INTRODUCTION

In recent years, computing hardware has grown exponentially (memory and hard drive capacity, number of processors, speed, and introduction of the 64-bit platform), while at the same time becoming more affordable. This has allowed groundwater modelers to build bigger, more realistic groundwater models, with higher resolution, leading to increasing grid sizes and simulation run times.

Commonly used classical linear solvers are not scalable; their runtime rises exponentially with the size of the model. Carefully designed algebraic multigrid methods anticipate this effect, they promise optimality and robustness, and can easily be plugged into existing simulation software. For instance, SAMG has been introduced as the primary numerical solver module in FEFLOW 6 and has been linked into MODFLOW-2000 or MODFLOW-2005 as an alternative solver. This is done via the so-called Link-AMG (LMG) interface.

The “original” LMG interface was introduced by the USGS in 2001 in connection with the academic forerunner of SAMG, namely, the very first algebraic multigrid code, AMG1R5. The new interface, LMG2, which links SAMG with MODFLOW-2000 and -2005, has been drastically improved and its applicability has been widely extended. The following describes the novelties in LMG2 and demonstrates its performance for a representative collection of models.

LMG AND ALGEBRAIC MULTIGRID METHODS

Algebraic Multigrid methods, and as such SAMG, proceed in two phases for solving a linear system of equations: first, an expensive setup phase in which all hierarchical components are set up, and secondly a cycling phase in which the linear system is solved iteratively, further details can be found in [Stüben et al]. The new LMG2 interface is able to analyze the solver behavior for previous linear systems and (automatically) exploits this information to minimize overall runtime. This is possible due to the fact that subsequent linear systems in non-linear and/or transient simulations typically change only slightly regarding their algebraic properties. This way LMG2 enables SAMG to reuse parts or all of previous expensive setup phases whenever possible which significantly reduces overall run times.

This automatic optimization process is the most important advantage of LMG2. In addition, it implements an enhanced memory management and is OpenMP parallel, leading to additional performance gains on state-of-the-art multi-core computers. Finally, in contrast to previous LMG versions, LMG2 now uses exactly the same residual-based stopping criteria as the alternative MODFLOW solvers, PCG and GMG.

In the past, differences in these stopping criteria made it difficult to compare different solvers realistically. Note that the damping strategies did not change.

NUMERICAL EXPERIMENTS

Benchmark tests were performed for 18 different models, two of them were only suited for MODFLOW-2000. The models represent common groundwater models and cover all possible variations, i.e. small/big time steps, steady state, high/low anisotropies, big/small size, linear and nonlinear. The brief characterization of the models can be found in Table 1.

MODEL	NLAY	NROW	NCOL	# cells	NPER	Time steps (TS)
A1	7	151	150	158,550	3	30 TS in the first SP, 10 TS in the following SPs
A2	7	151	150	158,550	3	10 TS in each SP
C1	70	120	120	1,008,000	2	First SP is SS. 50 TS in second SP
E1	13	94	101	123,422	18	10 TS in each SP
H1	10	177	106	187,620	1	SS
I1	9	405	405	1,476,225	1	SS
L1	21	500	500	5,250,000	1	SS
R1	7	1032	912	6,588,288	7	1 TS in first SP, 10 TS in the following SPs
S1	7	151	150	158,550	1	SS
S2	7	151	150	158,550	1	SS
T1	60	240	120	1,728,000	1	SS
T2	1	1500	700	1,050,000	1	SS
T3	4	190	194	147,440	49	First SP is SS. 1 TS in the following SPs
T4	15	194	160	465,600	1	SS
T5	3	163	153	74,817	1	SS
T6	40	160	160	1,024.000	1	SS
T7	1	855	1952	1,668.960	1	SS
V1	7	172	152	183,008	201	1 TS in first SP, 10 TS in the following SPs

Table 1. Overview of the test models. NLAY: number of layers; NROW/NCOL: number of rows/columns; #cells: degrees of freedom of the linear system; NPER: number of stress periods; SP: stress period; SS: steady state.

Linear Solver

SAMG is compared to the commonly used PCG and GMG solvers. PCG is an ILU-preconditioned CG solver which is well-suited for small models and/or very small time steps. GMG is favorable for large models and relatively large time steps. The OpenMP parallel version of PCG, described in [Dong and Li], is also evaluated, allowing for a comparison between the parallel performance of SAMG and PCG.

Comparison of LMG1 and LMG2

The main features of LMG2 have been described before. In particular, the smart and automatic reuse of setup data typically reduces CPU time significantly. This can clearly be seen by comparing LMG2 with its forerunner LMG1 which did not feature the reuse of setup data. The runtime for many models between LMG1 and LMG2 has improved significantly. Additionally, LMG2 makes use of an improved AMG strategy which uses less memory than previously.

Figure 1 shows results from LMG2, LMG1, and GMG, using the models documented in [Wilson and Naff]. Note that the CPU time is normalized, i.e. LMG2's CPU time is set to 100% highlighted by the blue line. The results were obtained with MODFLOW-2000. With the reuse of setup, LMG2 is able to outperform

GMG's CPU time for all considered models, with the exception of T5 and T6. Model T6 is a linear steady-state model, hence only one linear system has to be solved, and, consequently, LMG2 is not able to save setup time. However, due to LMG2's improved memory management, CPU-time is reduced substantially compared to LMG1. On the other hand, model T5 is small, causing the setup time to be over-proportionally large relative to the actual solution time. Although LMG2 performs only one setup for this model, the cost for this setup is so high that GMG outperforms in this special situation.

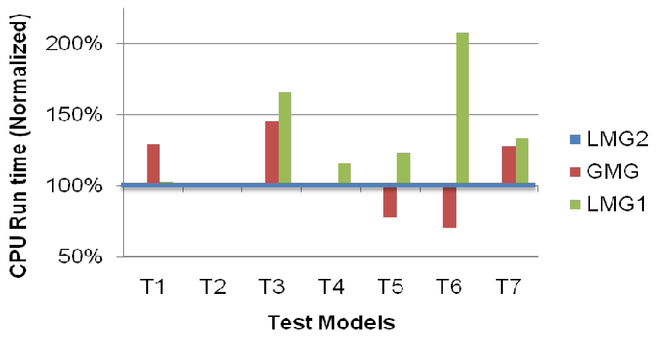

Figure 1. Comparison of CPU times between GMG, LMG1, and LMG2. LMG2's CPU time is set to 100 percent.

Comparison of LMG2 to GMG and PCG

In this section, tests are run using MODFLOW-2005. Hence, T4 and T5 are skipped since they use packages (SEN and OBS) which are no longer available within MODFLOW 2005.

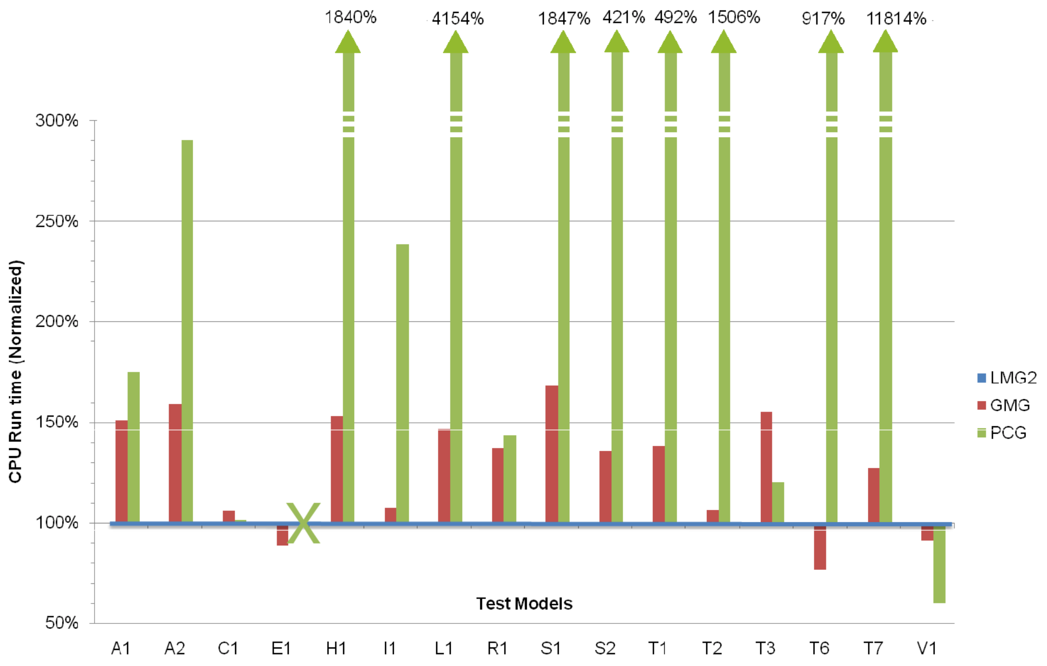

Figure 2: CPU time comparison of LMG2, GMG, and PCG. LMG2's CPU time is set to 100 percent.

Figure 2 shows that LMG2 is faster than PCG for all models, with the exception V1. This model makes use of relatively small time steps which usually is very beneficial for PCG. Note that the model E1 did not converge with PCG.

The CPU times of all solvers for C1 are very similar. C1 also uses small time steps. Hence, PCG should be far better than the other two solvers. However, the model also has a very small head and residual closure criteria (i.e. HCLOSE=1e-5, RCLOSE=1e-8). The strong closure criteria lead to over-proportionally more iterations of the PCG solver than of the other two solvers. Hence, the CPU time of all solvers is very similar. These results are in line with trends noticed in [Hill and Mehl].

When comparing GMG and LMG2, GMG is better for only three cases: T6, E1, and V1. Model T6 is linear and steady state. Hence, the setup time is relatively high compared to the overall solution time of the linear systems. Models E1 and V1 make use of relatively small time steps, which seem to be also beneficial for GMG's convergence. When the models feature high conductivity differences and/or anisotropies, GMG necessarily looses efficiency. This is in contrast to LMG2, which adaptively tunes its components to such situations. This is clearly visible in the results: While model V1 has no big variance in the hydrologic conductivities, models T1, T7, S1, S2, H1, A1, and A2 feature strong anisotropies. Hence, for the latter models, SAMG's performance is extremely good.

Generally, we would expect an enormous save CPU time for LMG2 compared to the other solvers for very big models. The reason for the low speed-up of 1.4 for model R1 is the small time step size used, which counteracts the model size.

The memory usage of SAMG compared to GMG is significantly higher because all transfer operators, such as restriction and interpolation, have to be stored. However, with modern machines operating on several gigabytes of memory, the difference in memory consumption is a minor tradeoff given the significant improvement in runtime, especially for models with large grids.

Parallel Performance

In this section, we first analyze the parallel performance of LMG and PCG. 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, the better its parallel performance.

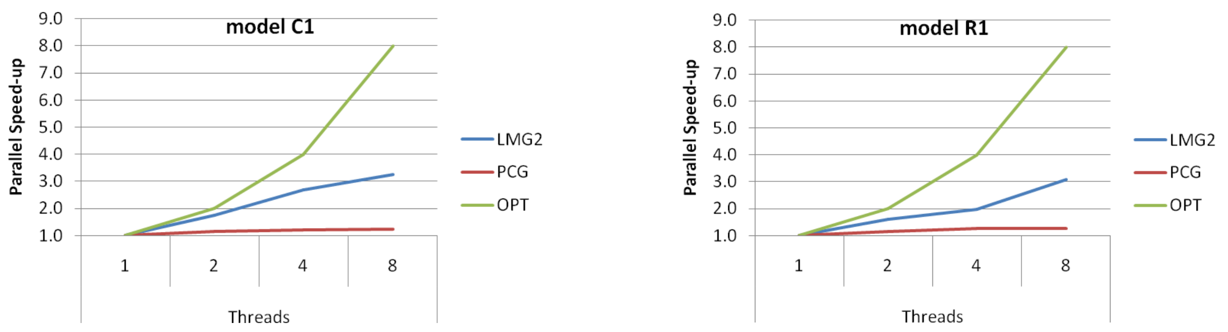

Figure 3. Parallel Speedup for the models C1 and R1 of LMG2 and PCG. The ideal (OPT) parallel speed-up (green) is also drawn.

For demonstration, the test problems C1 and R1 were run up to 8 cores on an Intel Nehalem (2x4 cores, 24 GB Memory) computer. Figure 3 shows the corresponding parallel speed-up for the parallel MODFLOW solvers LMG2 and PCG. Additionally, a line for the ideal speedup is drawn. Generally, the memory bandwidth provides a limitation in the achievable speedups. Hence, even if more threads are used, no significant further improvements in runtime are to be expected. For both PCG and LMG2, the ideal performance is not reached. The reason for this is inherent to the respective algorithms: PCG as well as LMG2 are not parallelizable to a full extent. Generally, the parallel performance of LMG2 seems better than that of PCG, which is seen in Figure 3. Note that the parallel speed-up strongly depends on the compiler and hardware used. The parallel speed-up of PCG seems to be bounded for the Intel Fortran Compiler and the Nehalem machine used.

Figure 4 shows a runtime comparison of GMG, PCG and LMG2 on 2 threads. The results demonstrate that LMG2 outperforms both GMG and PCG for all models but V1. Here again PCG is better for the reasons already mentioned in the previous section. Note the more cores are used the better is LMG2's performance compared to GMG and PCG, due to its significantly better scalability.

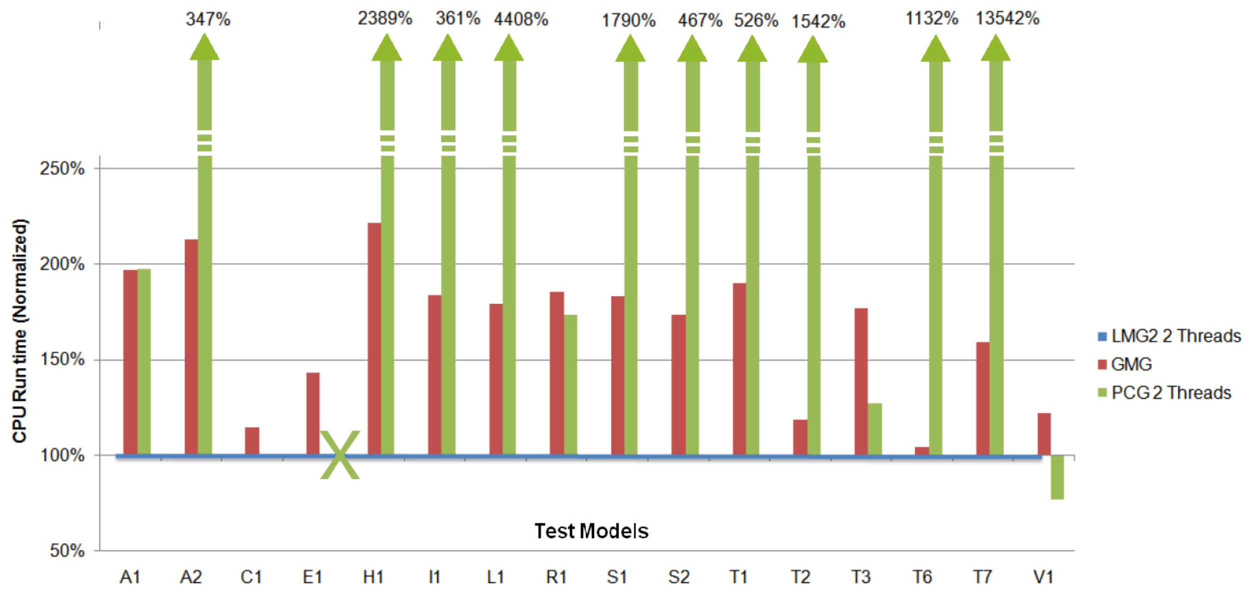

Figure 4. CPU time comparison between LMG2 and PCG using two threads and the sequential GMG . LMG2's CPU-time on one thread is set to 100 percent.

SUMMARY

The performance of the LMG2 solver compared to the PCG and GMG solvers has been demonstrated for a large set of problems. LMG2 is faster than PCG and GMG for nearly all considered models. LMG2 is extraordinarily efficient for large grids and highly variable conductivity fields. The results indicate that LMG2 is up to 100 times faster than PCG and up to 2 times faster than GMG. Only in the case of very homogenous conductivity distributions, GMG is able to perform slightly better than LMG2. Only for very small time steps and low accuracy requirements PCG is better than both GMG and LMG2. There are plans to extend LMG2 so that it detects situations which are particularly suited for PCG (very small meshes and/or very small time steps) and automatically switches to a PCG variant. With such a control mechanism LMG2 should eventually become the solver of choice for all MODFLOW models.

LMG2 is very robust and easy to use. In contrast to LMG2, GMG has to be tuned manually in order to handle anisotropies correctly; otherwise GMG may run into convergence problems. Furthermore, we have seen that the parallelization of LMG2 further improves its performance. In particular, LMG2 on 2 threads causes better run-times than the GMG solver for all problems. The performance gain will increase further with rising cores.

Future developments for LMG2 include expanding to support USGS MODFLOW-LGR and SEAWAT engines.

REFERENCES

Dong Y., Li G. A Parallel PCG Solver for MODFLOW. Ground Water 47, no. 6:845-850, 2009.

Stüben, K., Delaney, P., Chmakov, S. Algebraic Multigrid (AMG) for Ground Water Flow and Oil Reservoir Simulation. "MODFLOW and MORE", Colorado School of Mines, Golden, Colorado, 2003.

Detwiler L., Mehl S., Rajaram H., and Cheung W. Comparisons of an algebraic multigrid algorithm to two iterative solvers used for modeling ground water flow and transport. Ground Water, (v. 40, no. 3), 2002.

S.W. Mehl and M.C. Hill. MODFLOW-2000, the u.s. geological survey modular ground-water model-user guide to the link-amg (lmg) package for solving matrix equations using an algebraic multigrid solver. Open-File Report, 01-177, 2001.