A Massively Parallel Multigrid Method with Level Dependent Smoothers for Problems with High Anisotropies

Sebastian Reiter, Andreas Vogel, Arne Nägel, and Gabriel Wittum

Abstract Anisotropic layers, as often seen in biological and geological domains, impose difficulties to several aspects of numerical simulations. In this article we examine how the highly scalable approach to massively parallel geometric multigrid solvers presented in Reiter et al. (Comput Vis Sci 16(4):151–164, 2013) can be extended to problem domains featuring such anisotropies. Considering the real world problem of drug diffusion through the human skin we combine hierarchically distributed multigrids, anisotropic refinement, and level dependent smoothing strategies to create a robust and highly scalable multigrid solver for anisotropic domains.

Keywords Multigrid • Parallelization • Anisotropy • Smoothing

1 Introduction

The development of algebraic solvers for discretizations of partial differential equations on massively parallel computers is an active research field. Multigrid methods [7] have been employed with great efficiency for elliptic PDEs on large super-computers [1, 3, 4, 6, 9, 10, 13–15, 17]. In [13] we demonstrated that the geometric multigrid solver of the software package UG4 [16] has nearly optimal weak scaling properties for up to 262,144 processes and more than 10^{10} unknowns. The study indicates that very good scalability should be achievable for even higher numbers of processes and unknowns with the given approach, once the required resources are available.

An issue that had not been fully addressed in previous studies are the difficulties to solve those equations on massively parallel computers in the presence of anisotropic coefficients or anisotropic elements in the underlying grid. Techniques exist to address those problems for general settings for lower process numbers, e.g., involving the use of anisotropic refinement to construct a specialized grid

S. Reiter (✉) • A. Vogel • A. Nägel • G. Wittum
G-CSC, Goethe-Universität Frankfurt, Kettenhofweg 139, 60325 Frankfurt (M.), Germany
e-mail: sebastian.reiter@gcsc.uni-frankfurt.de; andreas.vogel@gcsc.uni-frankfurt.de; arne.naegel@gcsc.uni-frankfurt.de; wittum@gcsc.uni-frankfurt.de

W.E. Nagel et al. (eds.), *High Performance Computing in Science and Engineering '16*, DOI 10.1007/978-3-319-47066-5_45

hierarchy [2]. While parallelism is considered in [2], massively parallel systems as todays supercomputers with hundred thousands of computing cores did not exist at that time and no optimizations regarding massive scalability have thus been performed. In [10] massively parallel multigrid is described for the solution of elliptic PDEs for the special case with strong vertical anisotropies on structured grids.

Considering the real world problem of drug diffusion through the human skin [11], we extended the methods described in [2] to construct a method that employs geometric multigrid on massively parallel computers for problems with highly anisotropic elements using a combination of specialized refinement techniques and smoothers resulting in a robust and highly scalable solver for anisotropic problems. The special grid layout of the model problem thereby requires a solver which can handle anisotropies in all spatial directions on unstructured grids.

2 Problem Description

The motivating biological question for the construction of our solver is the numerical simulation of substance transport through the human skin. Using simulations of such processes helps to estimate the risk assessment of chemical exposures and at the same time the need for in vitro and in vivo testing can be reduced. However, the special structure of the human skin imposes several numerical challenges due to the anisotropic geometry and physical coefficients varying by orders of magnitude, cf., e.g., [8, 11]. The uppermost part of the skin, called *stratum corneum (SC)*, consists of multiple layers of cells (*corneocytes*) which are connected by thin channels (*lipid layers*) (cf. Fig. 1). Since parameters affecting transport and diffusivity of a substance vary strongly in those subdomains, both have to be considered in a simulation. For benchmark purposes we solve a modified heat equation

$$\frac{\partial}{\partial t}(Ku) + \nabla \cdot (-\mathbf{D}K\nabla u) = 0$$

in a computational domain $\Omega = \Omega_{lip} \cup \Omega_{cor}$. Here, u corresponds to the chemical activity, $K = K(x)$ and $\mathbf{D} = \mathbf{D}(x)$ are spatially dependent partition and diffusion coefficients respectively. Concentrations are given by $c = Ku$ and may undergo discontinuities which reflect domain dependent variations in lipophilicity and hydrophilicity. For reasons of simplicity we used $K = 1$ and

$$D(x) = \begin{cases} 1 & \text{if } x \in \Omega_{lip} \\ 10^{-3} & \text{if } x \in \Omega_{cor}. \end{cases}$$

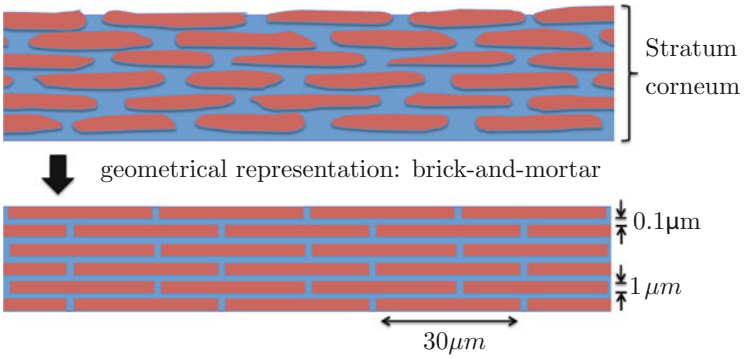

Fig. 1 Idealized model of the *stratum corneum*: *brick-and-mortar* grid

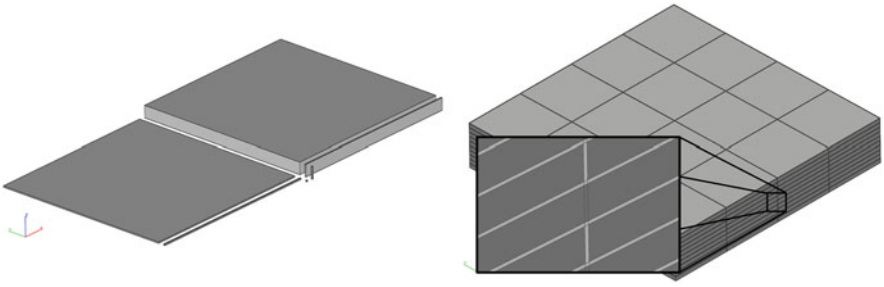

Fig. 2 *Left*: Individual hexahedral elements used for the *brick-and-mortar* FE-grid. *Right*: Anisotropic coarse grid of the 3d *brick-and-mortar* geometry (1280 elements)

For this study we focused on the idealized but still realistic *brick-and-mortar* domain. To this end, we used finite element grids consisting of highly anisotropic elements. This level of anisotropy is required to reduce the number of elements in the coarse grids, while still capturing the topology and morphology of the underlying domain.

For the considered geometry only hexahedral elements with varying degrees of anisotropy were used. The elements used to construct the coarse grid and an overview over the resulting FE-grid are depicted in Fig. 2. In the given example the highest aspect ratios are 1:15, however, the presented methods work for much higher aspect ratios, too.

The steady state solution of the regarded problem setup is depicted in Fig. 3.

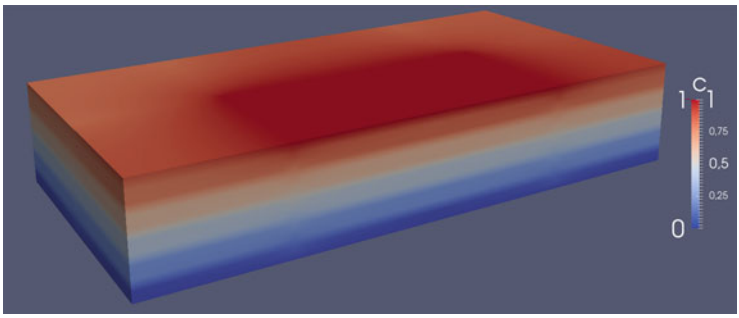

Fig. 3 Cut through the domain showing the steady state solution on level 5

3 Solver Setup

In [13] we employed a massively parallel geometric multigrid solver for the Poisson problem on grids with isotropic cells. While the rather simple Jacobi smoother used in those studies is fast and perfectly scalable, it is not suitable for anisotropic problems, since its smoothing properties deteriorate in this case. Iterative methods like the ILU method on the other hand are known to possess good smoothing properties also for highly anisotropic problems (cf. [2, 5, 18]). However, for most applicable methods an efficient parallel implementation is not feasible. The typical strategy for parallelization is then to employ those methods in Block-Jacobi-type fashion, i.e., on each process the more sophisticated smoother is executed locally, while interprocess couplings are treated using a Jacobi method.

While this setup works nicely for smaller process numbers, the iteration numbers typically increase with the number of processes being involved. As a second aspect, load imbalances can have a severe impact on the runtime of solver initializations and such load imbalances are typically given for massively parallel simulations on unstructured grids. This in particular holds true, when setup times do not grow like $O(n)$, as, e.g., for a threshold based ILUT.

In order to construct an optimal solver which can handle high anisotropies while still providing nearly optimal scalability, we combined the benefits of the Jacobi smoother for isotropic elements with the efficiency and robustness of the ILU smoothers for anisotropic problems. This combination is possible using a special refinement technique, which reduces the anisotropy of elements with each refinement until the resulting grid can be considered isotropic. The anisotropic and isotropic refinement rules used for the elements from Fig. 2 are depicted in Fig. 4.

In each refinement step only edges which are longer than a certain threshold are refined. Starting with a threshold of half of the length of the longest edge, the threshold is halved after each step so that shorter edges will be refined in the next iteration. For the shown brick-and-mortar geometry this technique leads to an isotropic grid after a certain number of refinements, depending on the highest aspect ratio.

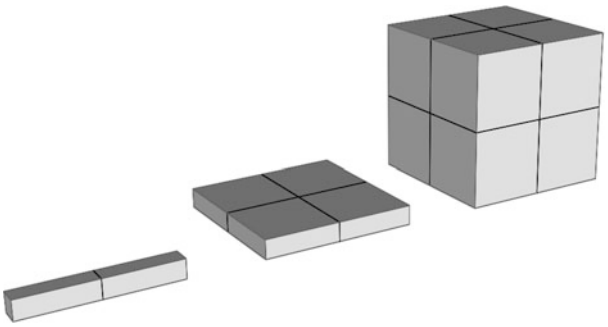

Fig. 4 Anisotropic refinement schemes for different shapes. *Black edges* are introduced during refinement

Table 1 Solver components: solvers on lower levels serve as base solvers for the preconditioners on higher levels

Levels	2	2–4	4-top
Solver	LU (exact)	CG	CG
Preconditioner	–	GMG	GMG
Smoother	–	ILU	Jacobi
Cycle	–	V (3,3)	V (3,3)

On the lower levels in which anisotropic elements are present we employ a multigrid method with robust ILU smoothing, whereas on higher levels, which contain nearly isotropic elements, a highly scalable multigrid method with Jacobi smoothing is used. This setup is even more justified considering the fact, that lower levels can be distributed to only a subset of the available processes, since they only contain a fraction of the elements of higher levels. We can thus make sure that complex ILU smoothing is only performed on a fixed number of processes, thus not interfering with the scalability on higher levels. To this end, we are using the hierarchical distribution approach described in [13]. Technically this setup is realized in the software package UG4 [16] by employing a multigrid method with ILU smoothing on lower levels as a base solver for the multigrid method with damped Jacobi smoothing ($omega = 0.5$) on higher levels. The prescribed tolerance for the intermediate coarse grid solver was a relative reduction by three orders of magnitude. Table 1 shows the different properties of the involved solver components.

4 Parallelization

Parallelization is performed using the approach detailed in [13]. Hierarchical distribution hereby plays an important role, since we apply the multigrid method with ILU smoothing on a smaller subset of processes only. The base solver for this lower ILU smoothed multigrid method may finally run on one process only.

Since anisotropic refinement may lead to load-imbalances, redistributions of the grid hierarchy may be necessary to improve those balances. To this end, we are using a fast parallel bisection strategy for distributed multigrid hierarchies as described in [12].

All communication between different processes is realized through the message passing interface *MPI*.

5 Results

A weak scaling study was performed on the Cray XC40 super computer *Hazel Hen* at the HLRS Stuttgart which features 7712 compute nodes, each with 128 GB of memory, 24 cores per node (virtually 48 through hyperthreading) and a peak performance of 7420 TFlops.

The study was performed by solving the aforementioned human skin brick-and-mortar model using the described solver setup. To allow for better comparability of the different runs, the number of outer CG-iterations was thereby fixed to 12, which resulted in a relative reduction of the defect by approximately 10^{-6} in all runs. The study starts on 1 process and for each subsequent run we refine the grid once more using regular refinement, thus increasing the number of elements by a factor of 8. At the same time we also increase the number of involved processes by a factor of 8 to guarantee a constant workload per process for all runs. Since we executed the parallel base-solver of the outer multigrid method on level 4, our study starts with level 5. Table 2 shows the number of unknowns and the run times of the different runs. The scaling behavior of assembly, solver initialization, and solving is also shown in Fig. 5.

Table 3 gives an overview over levelwise distribution qualities. The distribution quality q_l of a level l of the hierarchy is computed as

$$q_l := \frac{n_l^{total} - n_l^{max}}{n_l^{max} \cdot (P_l - 1)},$$

Table 2 Each line corresponds to an individual run. Recorded are the number of processes (*PEs*), the number of levels (*Levels*), the number of unknowns (*DoFs*), the run times of assembly (T_{ass}), solver initialization (T_{ini}), and solving (T_{sol})

PEs	Levels	DoFs	T_{ass} (s)	T_{ini} (s)	T_{sol} (s)
8	6	522,720	0.48	1.38	6.58
64	7	4,181,760	0.80	2.03	6.95
512	8	33,454,080	0.85	2.10	7.25
4096	9	267,632,640	0.87	2.10	7.15
32,768	10	2,141,061,120	0.86	2.13	7.60

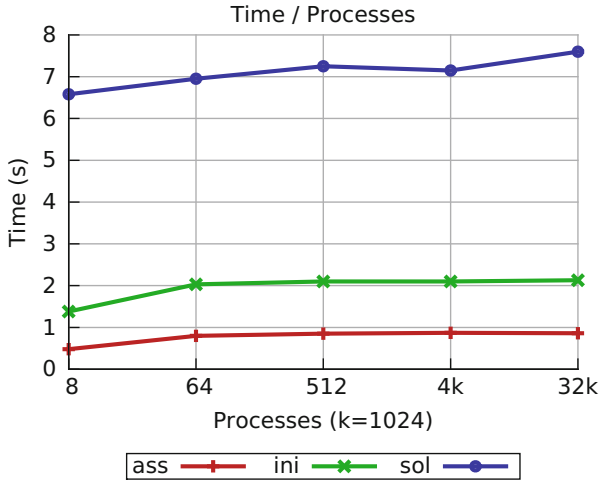

Fig. 5 Scaling of the run times of assembly (*ass*), solver initialization (*ini*), and solving (*sol*)

Table 3 Distribution qualities for each level of the multigrid hierarchy for the different runs

PE	0	1	2	3	4	5	6	7	8	9
8	1	1	1	1	1	1	–	–	–	–
64	1	1	0.92	0.93	0.94	0.99	0.99	–	–	–
512	1	1	1	0.91	0.95	0.94	0.94	0.94	–	–
4096	1	1	1	0.91	0.95	0.89	0.89	0.89	0.89	–
32,768	1	1	1	0.74	0.83	0.93	0.78	0.78	0.8	0.8

where $P_l > 1$ is the number of processes of the given process-hierarchy on level l, n_l^p is the number of elements in level l on process p, and

$$n_l^{total} := \sum_{p=1}^{P_l} n_l^p,$$

$$n_l^{max} := \max_{p=1,\ldots,P_l} n_l^p.$$

For $P_l = 1$ numerator and denominator both vanish and we define $q_l = 1$. q_l is thus in the range $[0, 1]$, where $q_l = 0$ means that all elements of level l are contained on one process only and $q_l = 1$ reflects an equal share of elements amongst all processes.

In Table 4 the number of processes used on each level for the individual runs are specified. They were chosen so that each level on each process would at least contain 32 elements (given an ideal load-balance). The first redistribution is performed for up to 256 processes. For each further redistribution the number of processes is multiplied by 64 and capped, if the maximum number of processes is reached.

Table 4 Number of processes used on each level for the individual runs

PE	0	1	2	3	4	5	6	7	8	9
8	1	1	8	8	8	8	–	–	–	–
64	1	1	64	64	64	64	64	–	–	–
512	1	1	1	256	256	512	512	512	–	–
4096	1	1	1	256	256	4096	4096	4096	4096	–
32,768	1	1	1	256	256	256	16,384	16,384	32,768	32,768

Both matrix assembly and solver initialization are performed process locally. The increase in run time T_{ass} and T_{ini} from 8 to 64 processes is related to the slight load-imbalance which can be observed for higher process numbers (cf. Table 3). Nevertheless, the scaling behavior of both assembly and initialization is very good and perfectly suited for large scale parallel runs.

The solver scalability is satisfactory as well. The run times T_{sol} increase slightly the more processes are involved and two effects are in play here: The distribution quality of the grid hierarchy deteriorates the larger the number of processes involved. This reflects the fact that some processes have more work to do than others in each program section due to the slight load imbalance. The slight imbalance is to be expected for an unstructured grid in which no special properties can be exploited for partitioning. However, for runs up to 256 processes the increasing parallelization of the intermediate base-solver on level 4 has a positive effect on total solver run times.

As demonstrated in [13], the underlying multigrid implementation in UG4 has nearly optimal scaling properties for perfectly balanced grids. The slightly worse scaling properties in the study at hand are thus likely to be linked to the observed load-imbalance. Nevertheless, given the complexity of the problem at hand we think that the achieved run-times are still convincing. The achieved results demonstrate the applicability of the presented approach to gain insight into complex biological processes through high-resolution numerical simulations on massively parallel computers.

Acknowledgements We thank the HLRS for the opportunity to use *Hazel Hen* and their kind support.

References

1. Baker, A.H., Falgout, R.D., Kolev, T.V., Yang, U.M.: Multigrid smoothers for ultra-parallel computing. SIAM J. Sci. Comput. **33**, 2864–2887 (2011)
2. Bastian, P., Wittum, G.: Adaptive multigrid methods: the UG concept. In: Adaptive Methods – Algorithms, Theory and Applications: Proceedings of the Ninth GAMM-Seminar, Kiel, 22–24 Jan 1993, pp. 17–37. Vieweg+Teubner Verlag, Wiesbaden (1994)
3. Bastian, P., Blatt, M., Scheichl, R.: Algebraic multigrid for discontinuous Galerkin discretizations of heterogeneous elliptic problems. Numer. Linear Algebra Appl. **19**(2), 367–388 (2012)

4. Bergen, B., Gradl, T., Rude, U., Hulsemann, F.: A massively parallel multigrid method for finite elements. Comput. Sci. Eng. **8**(6), 56–62 (2006)
5. Bramble, J., Zhang, X.: Uniform convergence of the multigrid v-cycle for an anisotropic problem. Math. Comput. **70**(234), 453–470 (2001)
6. Gmeiner, B., Köstler, H., Stürmer, M., Rüde, U.: Parallel multigrid on hierarchical hybrid grids: a performance study on current high performance computing clusters. Concurr. Comput.: Pract. Exp. **26**(1), 217–240 (2014)
7. Hackbusch, W.: Multi-grid Methods and Applications, vol. 4. Springer, Berlin/New York (1985)
8. Heisig, M., Lieckfeldt, R., Wittum, G., Mazurkevich, G., Lee, G.: Non steady-state descriptions of drug permeation through stratum corneum. I. The biphasic brick-and-mortar model. Pharm. Res. **13**(3), 421–426 (1996)
9. Heppner, I., Lampe, M., Nägel, A., Reiter, S., Rupp, M., Vogel, A., Wittum, G.: Software framework ug4: parallel multigrid on the hermit supercomputer. In: High Performance Computing in Science and Engineering 12, pp. 435–449. Springer, Berlin/London (2013)
10. Müller, E.H., Scheichl, R.: Massively parallel solvers for elliptic partial differential equations in numerical weather and climate prediction. Q. J. R. Meteorol. Soc. **140**(685), 2608–2624 (2014)
11. Nägel, A., Heisig, M., Wittum, G.: Detailed modeling of skin penetration–an overview. Adv. Drug Deliv. Rev. **65**(2), 191–207 (2013) Modeling the human skin barrier – towards a better understanding of dermal absorption
12. Reiter, S.: Effiziente Algorithmen und Datenstrukturen für die Realisierung von adaptiven, hierarchischen Gittern auf massiv parallelen Systemen. PhD thesis, Universität Frankfurt am Main (2014)
13. Reiter, S., Vogel, A., Heppner, I., Rupp, M., Wittum, G.: A massively parallel geometric multigrid solver on hierarchically distributed grids. Comput. Vis. Sci. **16**(4), 151–164 (2013)
14. Sampath, R.S., Biros, G.: A parallel geometric multigrid method for finite elements on octree meshes. SIAM J. Sci. Comput. **32**, 1361–1392 (2010)
15. Sundar, H., Biros, G., Burstedde, C., Rudi, J., Ghattas, O., Stadler, G.: Parallel geometric-algebraic multigrid on unstructured forests of octrees. In: Proceedings of the International Conference on High Performance Computing, Networking, Storage and Analysis, SC '12, pp. 43:1–43:11, Los Alamitos. IEEE Computer Society Press (2012)
16. Vogel, A., Reiter, S., Rupp, M., Nägel, A., Wittum, G.: UG 4: a novel flexible software system for simulating PDE based models on high performance computers. Comput. Vis. Sci. **16**(4), 165–179 (2013)
17. Williams, S., Lijewski, M., Almgren, A., Van Straalen, B., Carson, E., Knight, N., Demmel, J.: s-step Krylov subspace methods as bottom solvers for geometric multigrid. In: 28th International Parallel and Distributed Processing Symposium, pp. 1149–1158. IEEE, Piscataway (2014)
18. Wittum, G.: On the robustness of ILU smoothing. SIAM J. Sci. Stat. Comput. **10**(4), 699–717 (1989)