Seventh International Conference on Computational Fluid Dynamics (ICCFD7), Big Island, Hawaii, July 9-13, 2012

ICCFD7-2905

HPC Fluid Flow Simulations in Porous Media Geometries

M. Lieb, M. Mehl, T. Neckel, K. Unterweger
Corresponding author: liebm@in.tum.de

Institut für Informatik, Technische Universität München, Germany

Abstract: The simulation of flows in porous media on the microscale level requires a high number of unknowns to resolve the geometry of the fissures in-between the sand grains. For the geometric representation of the sand grains we use fracture network-like structures in 2D and sphere packings in 3D. These geometries are created with an in-house scenario generator, based on a modified and extended version of the Lubachevsky-Stillinger algorithm in 3D. The incompressible Navier-Stokes flow solver of the PDE framework Peano is used to compute the overall flow through the reference volume. The throughput results allow, in combination with suitable criteria, for the estimation of the necessary resolution of a simulation setup. Highly parallel runs have been performed with several thousands of compute nodes on a HPC system showing good scalability results even for this kind of unbalanced geometry data.

Keywords: Computational Fluid Dynamics, Cartesian Grids, HPC, Porous Media, Geometry Generation, Adaptive Grids.

1 Introduction

Simulating flow in complex geologic formations such as oilfields or saline aquifers to enable enhanced applications like oil recovery by CO_2 sequestration is a complex and challenging task. To understand the underlying phenomena and optimise the design of the application, sufficiently accurate numerical simulations of the corresponding processes are necessary. Even with the help of more and more powerful supercomputers, a full resolution of the microscopic scale within the complex geologic formations is, however, not affordable since the computational domains are huge compared to the studied simulation scale. The domains of interest are typically by a factor of about $10^5 - 10^8$ larger than the underlying micro scales; in a 3D setup, about 10^{18} degrees of freedom would be necessary. Since a direct solution of such systems is out of reach, homogenisation techniques are applied to reduce the computational intensity of the problem. At the macro scale, additional assumptions are introduced which typically increase the complexity of the underlying model.

For homogenisations, different approaches exist. Using one combined mesh of macro scale cells, fully coupled techniques resolve the overall domain by a mixture of different discretisations. Thus, mere porous media parts of the domain are solved using Darcy's law, e.g., while special non-porous areas of the domain—such as relatively large cavities or channels—are treated as full flow regions (see [1], e.g.). Another approach are the so-called upscaling techniques. Here, the modelling on the coarse scale is (considerably) improved by flow computations on smaller areas of reference. Based on the upscaling approach presented in [2], we developed a similar variant by generating explicitly resolved porous media-like geometries (fracture network-like structures via channels in 2D and sphere packings in 3D) and computing the corresponding Navier-Stokes flow in order to obtain the respective permeability tensor values (see [3]).

Within this paper, we focus on the explicit computation of flow in representative equivalent volumes (REVs). Using a set of results for the two-dimensional scenarios, the flow in the fracture network-like channel structures is analysed in detail. Together with first results on the flow in 3D sphere packings, these results allow to estimate the necessary resolution and, thus, the resource and runtime requirements for REV flow simulations. The streamlines and velocity field of such a simulation is depicted in Figure 1.

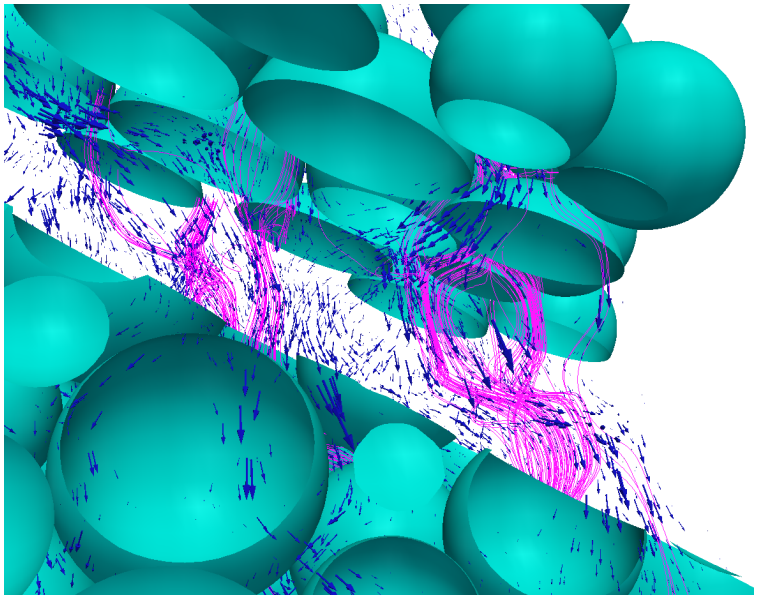


Figure 1: Streamlines and arrows indicating the velocity field of a steady state flow through a 40 spheres REV setup.

In the three-dimensional setup, the domain may be approximated by different sphere packings representing subdomain patches and different sediment layers. Hence, we extended the Lubachevsky-Stillinger algorithm [4] such that radii variations in the packings are now possible and can be configured by prescribing built-in or user-defined distribution functions.

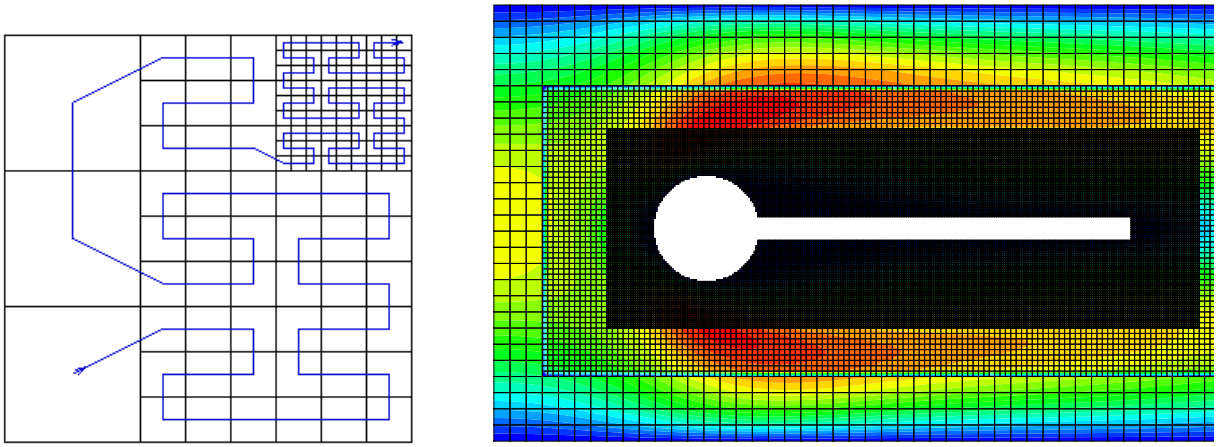
Since fully representative scenarios require thousands of spheres and hundreds of millions of grid elements, high parallel efficiency of the simulation environment is essential. Due to the porous media-like geometry, a considerable amount of cells and vertices are outside the flow volume in our Eulerian approach. Hence, load imbalance increases much faster in parallel setups with a growing number of processes and degrees of freedom compared to usual channel-like flow scenarios. Therefore, a scaling analysis up to several thousand processors has been performed showing both the validity of our approach and the relevance of such porous media-like setups for High-Performance Computing (HPC).

The remainder of this paper is organised as follows: The solver for the Navier-Stokes equations and the underlying software framework are briefly described in Sec. 2. In Sec. 3, we present the generator tool for the different porous media-like geometries. Numerical results of flow throughputs in 2D fracture network-like channel scenarios and in 3D as well as of large-scale, parallel runs are provided in Sec. 4. A conclusion and an outlook on future work in Sec. 5 close this contribution.

2 The PDE Framework Peano

To run incompressible flow simulations on the small-scale porous media-like geometries, we use the Navier-Stokes flow solver of Peano. The framework Peano is implemented in C++ using a strictly object-oriented manner relying on relevant approaches of Software Engineering such as a clear design concept, automated tests, and continuous integration (cf. [5, 6, 7]). Peano addresses important challenges concerning performance and re-usability of code and offers regular and adaptive Cartesian meshes. In order to automatically generate the mesh for computations, a given surface geometry is embedded into the domain—typically a hypercube root cell—and the corresponding "brick-world" volume geometry for the PDE solver is created efficiently. The adaptive Cartesian grids, available for arbitrary dimensions, are realised via spacetrees (similar to octrees), space-filling curves and stack data structures. For iterating over the grid, the underlying spacetree of cells is serialised using the so-called space-filling curves ([8]) which are recursively defined and, thus, fit perfectly to the generation and structure of the spacetree and the corresponding grid. A two-dimensional example

of an adaptive grid is shown in Figure 10(a) together with the corresponding discrete iterate of the space-filling Peano curve. A very cache-efficient mechanism relying on simple stack data structures is used for accessing vertex data in a cell-wise grid iteration which shows exactly the necessary linear "pile-up-pile-down" structure of accesses to the degrees of freedom. The combined usage of the Peano curve together with stack data structures results in very high cache-hit rates due to the inherent temporal and spatial locality of data access independent of the dimension, the underlying architecture, or the concrete type of application using Peano. In addition, no adjacency information has to be stored explicitly since the operator evaluation is realised in a strictly cell-wise manner, a fact that seems to be a restriction at first sight but actually fits perfectly to the approach and may be used also in combination with higher-order discretisation schemes. In particular, avoiding the storage of adjacency information leads to very low memory requirements of the code. Finally, the insertion or deletion of grid cells or vertices during refinement or coarsening of the mesh throughout the computations is straightforward to realise.



(a) (b)

Figure 2: (a) Example of a 2D adaptive Cartesian grid and its corresponding discrete iteration of the Peano curve. (b) Velocity distribution of the FSI benchmark CFD 1 (cf. [9]) at a Reynolds number of Re=20.

Peano's approach to adaptive Cartesian grids does neither depend on the discretisation scheme in use nor on the concrete PDE to be solved. Therefore, Peano represents a framework for different solvers, offering data handling, grid generation, an encapsulated shared and distributed memory parallelisation etc. Currently, the framework comprises several solvers, such as for the Poisson equation, for the continuity equation, for heat equation simulations, and for the incompressible Navier-Stokes equations (NSE) (cf. [10]).

The Navier-Stokes solver in Peano supports two- and three-dimensional spatial discretisation using low-order finite elements (FEM) or higher-order interpolated differential operators (IDO, cf. [11, 12]). Among the different explicit and implicit time integration methods available, the solver heavily relies on the explicit Chorin projection method (see [13]). A large number of benchmark computations showed the successful combination of the various features necessary for a modern flow solver and keeping good performance results of Peano such as a low memory footprint and high cache-hit rates (cf. [6, 5, 10], e.g.). Furthermore, the flow solver has been used for partitioned fluid-structure interaction simulations (such as proposed in [9], see Fig. 10(b) and [14]) where moving geometries and, hence, an efficient realisation of dynamically changing adaptive grids play an important role. Thus, Peano represents a powerful tool for efficient CFD simulations.

3 Scenario Generation

Setting up a flow scenario for porous media structures requires a considerable complex geometry. If measured data of the underlying geometry is not (yet) available or doesn't have the necessary resolution, or if simulations shall be calibrated, a computational construction of porous media-like structures is advantageous or even necessary. Creating such structures manually is not feasible; we therefore use generic geometry setups

which are generated by a scenario generator. This is a modular programme with a Qt[1] based graphical user interface (GUI) as a frontend. The algorithms are realised as dynamic libraries which are loaded during the start-up phase of the GUI. For dynamic generation of setups on supercomputers, a command line version is available as well. Currently we provide two sets of geometry setups which are discussed in the following: fracture networks in 2D and sphere packings in 3D.

3.1 Fracture network scenarios

A typical 2D porous media scenario is a crack network of a fractured rock. In our generator the cracks are represented by rotated channels. The insertion procedure is controlled by the user's specifications:

- The number of channels to be inserted in the domain.
- The length of the channels.
- The width of the channels.
- The minimum and maximum orientation of the channels in the domain.

The user frontend for such a setup is depicted in Figure 3. The user defined parameters are based on our

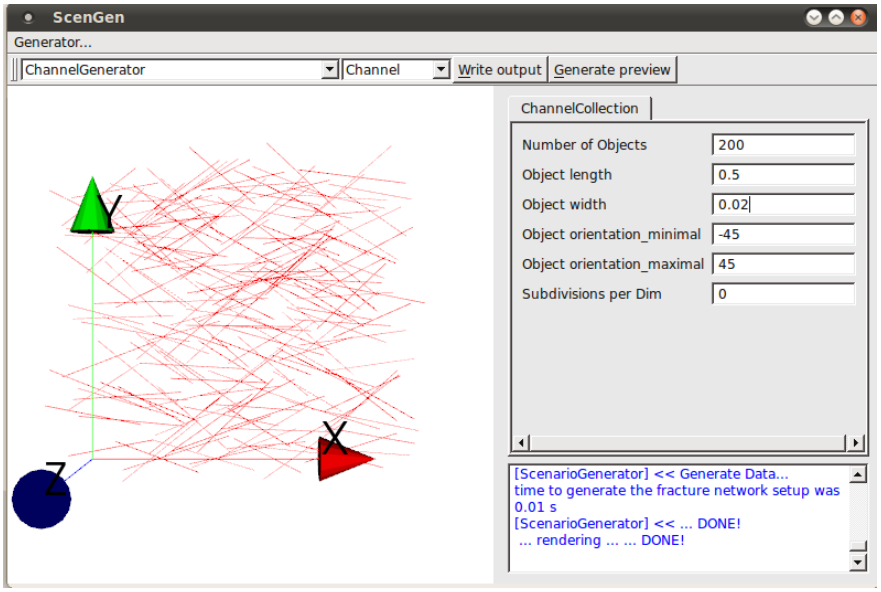


Figure 3: Screen shot of the ScenarioGenerator GUI with the fracture network generator. The channels of length 0.5 and width 0.02 are randomly oriented in the range of $(-\frac{\pi}{4}, \frac{\pi}{4})$ and embedded in the unit square in this example.

current needs and can easily be extended if a generation algorithm of higher complexity is desired, e.g. The basis of our setup is the model discussed in [2]. There, channels of fixed length l, fixed width w, a random orientation in the range of $(-\pi/2, \pi/2)$, and a random centre point $0 \leq x, y \leq 1$ are inserted into a unit square.

3.2 3D scenarios

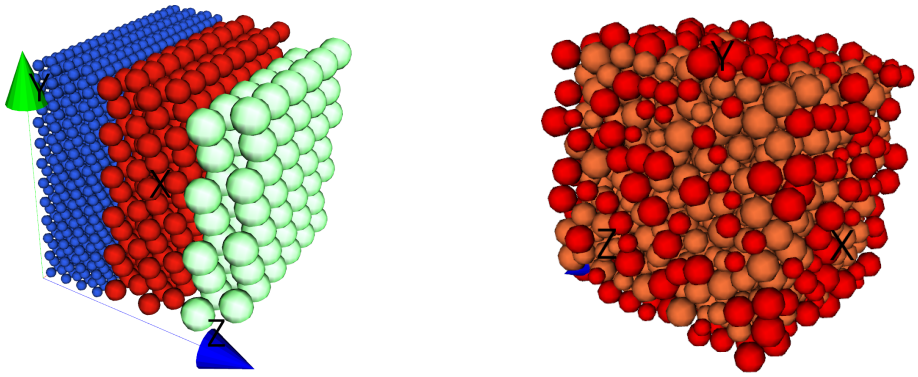
For the generation of 3D porous media-like geometries such as sand grain structures we use three different sphere packing variants:

- hexagonal close packing,

[1] http://qt.nokia.com/products/

- layers of regular sphere packings, and
- random sphere packing.

The first is a standard packing known from material science where the spheres are distributed uniformly in the domain. The layered packing represents layers of the regular packings with different porosity (see Fig. 4(a)). Each packing is specified by the radius of the spheres, the thickness of the layer in the stack, and the volume fraction which should be covered by spheres; this can be used as a simple model for multiple soil simulations. The random sphere packing represents an irregular but spatially periodic packing, as shown in Fig. 4(b). It is based on an algorithm of Lubachevsky and Stillinger [4]. We modified the algorithm to increase the inhomogeneity and to enforce a fully close packing prohibiting freely moving spheres. A positive side effect of these modifications is a reduced runtime: In the average, we could measure a speed-up factor of $9 - 11$ compared to the initial algorithm given the same number of spheres and equal distribution of $90 - 100\%$ of the reference radius. Whereas the original approach worked with homogeneous spheres, we allow a variation of the grain size among a packing. By default this will be a uniform distribution based on a reference radius together with a user-defined lower bound relative to the reference radius. Furthermore, it is now possible to use user-defined distribution functions for the grain size distributions (variations of the Gaussian distribution, e.g.).



(a) (b)

Figure 4: 3D porous media equivalent structures. (a) Example of a multilayer scenario. (b) Artificially generated sphere packing with periodic domain boundaries resulting in an average maximum density of 63.4%.

4 Numerical Results

In the following, we investigate the throughput through the domain depending on the resolution of the computational grid. During the setup phase of the computational domain, we ignore the cells that are not part of the domain. In our notation, a cell is defined to be outside (red) of the domain, if one of its adjacent vertices is outside of the fluid domain. A part of a rotated channel depicted in Fig. 5(a) shows a relatively coarse grid resolution with at most 3 fluid cells (blue) along the channels width. This grid is now refined by bisecting the cells dimension-wisely. As a result, a subset of the children of the boundary cells turn now into inner cells. If we take the former identification pattern we would expect 6 fluid cells across the channel. However, we end up with 8 inner cells along the channel's width. By decreasing the mesh width with a factor of 2, the width of the discretised channel increased by 30%. This increase of the fluid domain size can be observed for all non-mesh-aligned geometric structures. Porous media structures are complicated and thus practically never mesh-aligned. In the following we will analyse the effect of the resolution-based domain change for fracture networks (2D) and the flow through sand grains represented by irregular sphere packings. Our context is the permeability computation of micro-scale porous structures. Therefore, we compare the results based on the steady-state throughput computed by our flow solver. In the second part of the results,

we analyse the parallel efficiency of the flow solver and the imbalance effects formed by the non-homogeneous geometry.

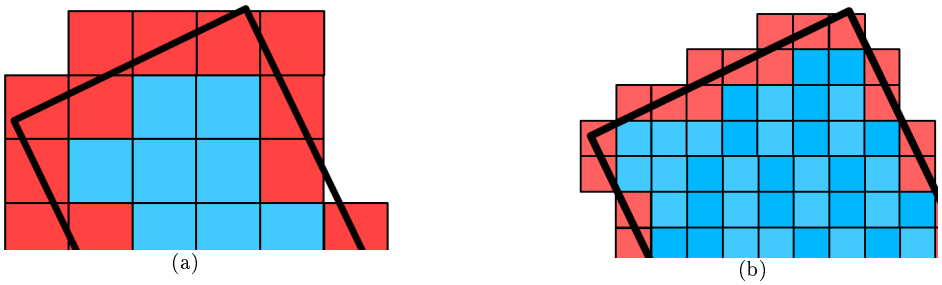


Figure 5: Effect of bisection the resolution on the general structure of the fluid domain. The red cells are defined as outside of the fluid domain. The blue cells represent the computational fluid domain.

4.1 2D Fracture Network Simulations

The requirement for our initial configuration is that the discretisation allows for a flow in the complete pore space. In the fracture network setups we started with a discrete width of three cells for a 45° rotated channel. The generation setup was as follows:

- width fixed: 0.02
- length fixed: 0.5
- orientation: $-45° - 10°$

The resulting fracture network is depicted in Figure 6(a). It is used as input data for our fluid solver. Applying no-slip boundary conditions at the top an the bottom and a constant positive pressure difference from the left to the right of the domain we run our simulation which computes the resulting flow. From this steady-state solution we compute the mass flow Q through our REV by integrating the velocities at the outflow. The incompressibility implies that the outflow equals the inflow:

$$Q = \rho \int_{\Gamma_{out}} u \cdot \vec{n} = - \int_{\Gamma_{in}} u \cdot \vec{n}. \tag{1}$$

Thus, we can compute the discrete outflow by summing up the normal velocities at the outflow:

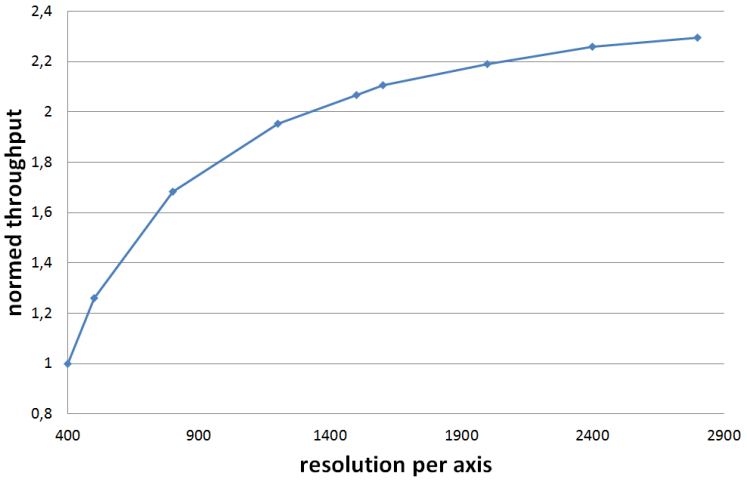
$$Q = \rho h \sum_{i=1}^{h^{-1}} u_{out,i} \tag{2}$$

Starting from the base resolution of 400×400 we simulate this scenario repeatedly, while we continuously refine the grid's resolution. In order to allow for a better comparison between the results we have normalised the outflow results based on the outflow of the coarsest scenario. The resulting behaviour pattern (Figure 6(b)) shows that the flow through the domain increases. However, the second derivative is negative and we can assume a convergence towards the continuous solution for $h \to 0$.

4.2 3D Sphere Packing Simulations

In our flow simulations we want to make sure that for all possible cuts through the geometry formed by our sphere packing there is at least one velocity degree of freedom. The densest possible packing is the hexagonal close packing. Here, we cut a slice through the narrowest area and examine the remaining area.

(a)



(b)

Figure 6: The channel scenario depicted in (a) is used to investigate the relative throughput. The development of the throughput over the number of time steps (b) is normalised with respect to the coarsest resolution (400×400).

In the centre we insert a sphere. The radius of this sphere r_{min} depends on the radius r of the spheres in the dense packing:

$$r_{min} = r\sqrt{2}\left(\frac{2}{\sqrt{3}} - 1\right) \tag{3}$$

We can simplify the problem and use this as the radius of a circumcircle of a quadrangle. Now we subdivide this quadrangle into 9 elements. The length of an edge of one of these subelements will now be the minimal mesh width required. It can be formulated as follows:

$$h_{min} \leq \frac{r}{3}\left(\frac{2\sqrt{2}}{\sqrt{3}} - \sqrt{2}\right) \tag{4}$$

Using equation (4) we now determine the starting discretisation h_{min} width for all sphere packings. As we are dealing with inhomogeneous packings we assume, that a close packing of some of the smallest spheres may exist within the domain and use r_{min} as the reference sphere radius.

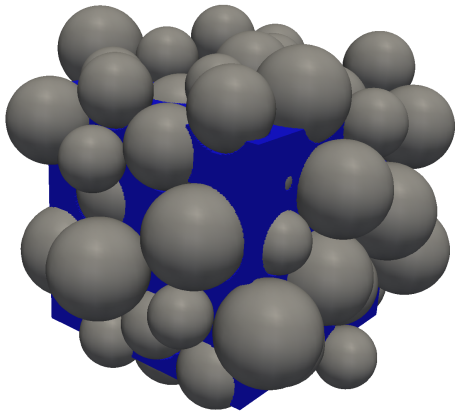
The graph in Figure 7(b) shows the change of throughput for a sphere packing scenario, depending on the grid resolution. From this we can deduct that the behaviour of the flow through a sphere packing is similar to that in the 2D fracture network case in the previous sections. However, further investigations of various setups and higher resolutions are needed to see, if this is a general tendency.

4.3 Parallelisation and HPC

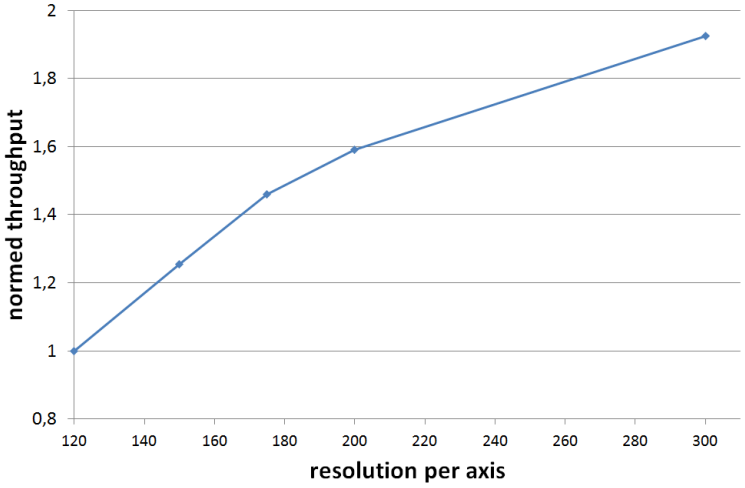
For simulating large domains with a feasible resolution, a huge amount of unknowns is needed. Especially in three-dimensional scenarios the number of grid points easily exceeds 10^9, leading to a memory demand of several hundreds of gigabytes and a computational amount that cannot be handled in a reasonable time on a single processing core. Up-to-date multicore machines also mostly lack the computing power and memory to cope with high-resolved simulations. Thus, a code used for such simulations must yield a parallelisation for distributed or hybrid, shared- and distributed-memory, machines.

4.3.1 Inhomogeneity

While the Peano framework offers a parallel implementation of the Navier-Stokes solver, scenarios with large porous geometries cause problems when computing on several hundreds or thousands of compute nodes.



(a)



(b)

Figure 7: The random sphere packing scenario with 40 fully inner spheres depicted in (a) is used to investigate the relative throughput. The development of the throughput over the number of time steps (b) is normalised with respect to the coarsest resolution($h = 125^{-1}$).

If the domain is decomposed only into few subdomains, the ratios between the parts of the subdomains that are considered as being inside and outside of the geometry are likely to correspond to the ratio of the complete domain. With increasing number of nodes the area of the computational domain that is covered by one node decreases. In the same time the difference between the ratios of subdomains increases, since some subdomains will probably have mostly fluid cells, while others will have mostly cells which are located in the geometry and thus outside of the fluid. That means that the subdomains become more and more inhomogeneous. Grid cells that reside in the geometry require only operations for the mere grid traversal, while grid cells that are outside of the geometry and thus inside the fluid domain, require additional operations for the actual solver. Therefore, the inhomogeneity between different subdomains leads to an imbalance between the work loads of the corresponding compute nodes which might influence the scalability of the application.

4.3.2 Experiments

All parallel simulations were performed on an IBM Blue Gene/P system with 16.384 compute nodes, each containing a quad-core PowerPC 450 clocked at 850MHz. There are 4GB of memory available per compute node.
The computed scenario has a cube-shaped domain and a geometry consisting of 200 spheres, shown in Figure 10 (a). We compute this scenario eight times with different numbers of compute nodes and appropriate resolutions to keep a fixed number of cells per node. That is, we are performing a weak scaling analysis. Since the number of solver iterations needed to gain a given accuracy threshold, depends on the grid resolution, we are only taking into account one single solver iteration. The parallel efficiency with respect to a parallel run with 8 compute nodes is shown in Figure 8. The given setup yields a parallel efficiency of approximately 70% for a parallel run with 4096 compute nodes and a grid with 847^3 cells.

To get an estimation of the impact of the subdomain's imbalance we measured the time each compute node needed for one Gauss-Seidel iteration. We estimate the imbalance as the difference between the time of the slowest compute node and the time of the fastest compute node, relative to the time of the slowest compute node: $imbalance = \frac{t_{max} - t_{min}}{t_{max}}$. The result can be found in Figure 9. In our scenario the imbalance increases from about 8% on 8 compute nodes to over 30% on 4096 compute nodes. The overview over the maximal, minimal and average runtimes is given in Table 1. The average runtime for the test case with 4096 compute nodes indicates that a perfectly balanced simulation would take 20% less time.

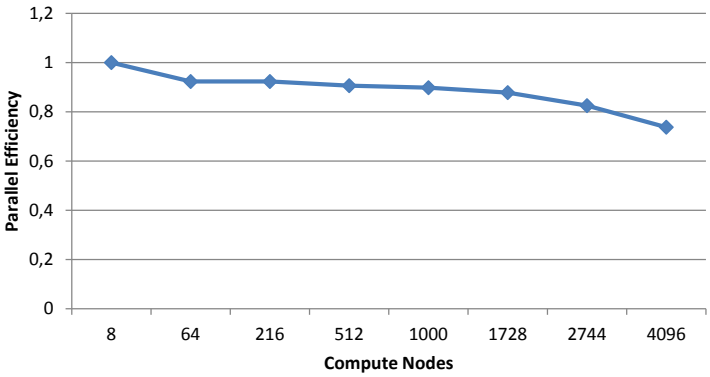


Figure 8: Parallel efficiency for the sphere packing scenario.

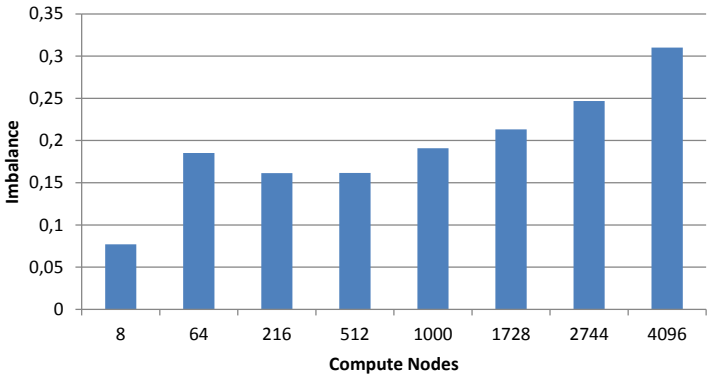
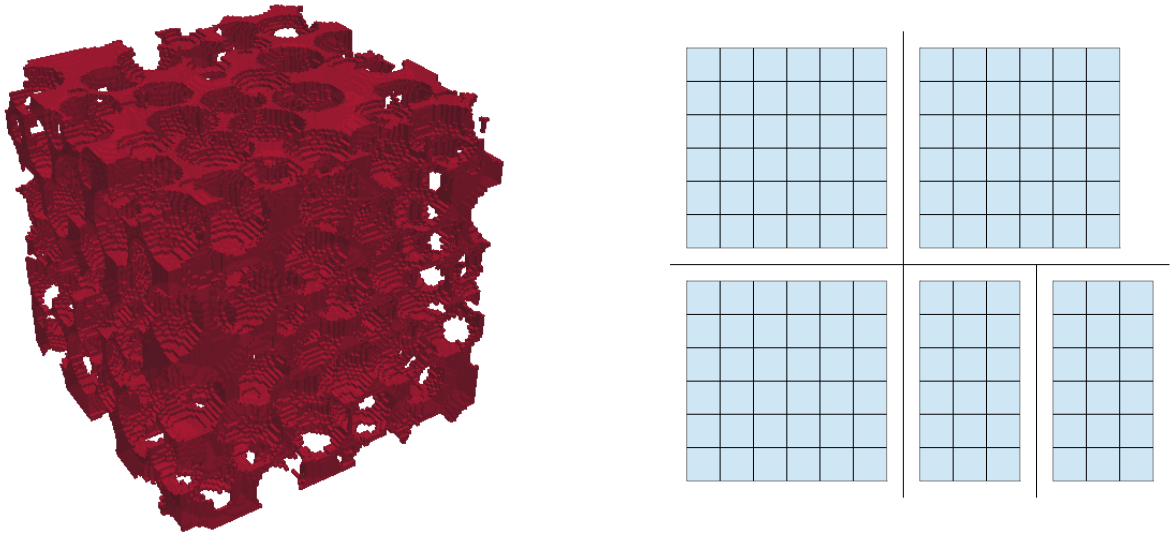


Figure 9: Measured imbalance with increasing number of compute nodes.

Table 1: Measured runtimes on compute nodes for running one Gauss-Seidel iteration

Compute nodes	Maximum time	Minimum time	Average time	Imbalance
8	3.74	4.05	3.90	0.08
64	3.54	4.35	3.98	0.19
216	3.62	4.31	4.03	0.16
512	3.63	4.33	4.06	0.16
1000	3.58	4.43	4.08	0.19
1728	3.61	4.59	4.09	0.21
2744	3.64	4.83	4.14	0.25
4096	3.68	5.34	4.25	0.31



(a) (b)

Figure 10: The random sphere packing scenario for the scalability tests using 200 spheres (a). Example of a non-uniform decomposition of a grid consisting of 12×12 cells (b).

4.3.3 Load Balancing

The domain decomposition in Peano does not necessarily follow the common concept for decomposing a rectangular domain into a rectangular array of compute nodes. It rather implements a kd-tree-like subdivision of the domain. So, initially the grid is decomposed in a regular manner, but any resulting subdomain may be further split along any coordinate axis. A resulting decomposition is shown in the right image of Figure 10. Here, a grid of 12×12 cells was initially split up into 2×2 subgrids with each 6×6 cells. Afterwards the lower right subdomain was further split into two subdomains of 3×6 cells, each. The black lines between the subdomains indicate the separating edges.

This kd-tree approach can be used for balancing the work load between the compute nodes. By examining the runtime for a grid iteration on the initial decomposition the compute nodes with the highest loads can be found and their subdomains can be further subdivided and redistributed on idling compute nodes. Hence, the imbalance can be decreased, which will lead to a better scalability for scenarios with complex geometry.

5 Conclusion and Outlook

For the simulation of a flow in porous media-like geometries on the microscopic level, we have explicitly discretised the domain using Cartesian grids. Using suitable criteria for detecting necessary mesh resolutions, we computed the flow throughput relevant for permeability tensor data for a number of representative elementary volumes. The simulations were executed in parallel on a HPC system (IBM Blue Gene System Shaheen[2]) and show good parallel efficiency. The next step will be, in collaboration with the group of Shuyu Sun[3], to use physical input data both to validate the simulations with reference data and to use these tensors in combination with a Darcy solver on a coarser spatial scale. Furthermore, we plan to use 3D rock scans as input data for the flow simulation.

6 Acknowledgements

The work presented in this paper has been supported by the International Graduate School of Science and Engineering (IGSSE) of the Technische Universität München. This support is gratefully acknowledged.

References

[1] K.-H. Hoffmann and N.D. Botkin. Multiphase flows with phase transitions: models and numerics. *JP Journal of Heat and Mass Transfer*, 6(2):73–86, 2012.

[2] A. A. Rodriguez, H. Klie, S. Sun, X. Gai, M. F. Wheeler, H. Florez, and U. Simon Bolivar. Upscaling of hydraulic properties of fractured porous media: Full permeability tensor and continuum scale simulations. In *Proceedings of the 2006 SPE Symposium on Improved Oil Recovery, Oklahoma, USA*, 2006.

[3] M. Lieb, T. Neckel, H.-J. Bungartz, and S. Sun. Towards a navier stokes-darcy upscaling based on permeability tensor computation. *Procedia Computer Science*, 9(0):717 – 726, 2012. Proceedings of the International Conference on Computational Science, ICCS 2012.

[4] B. D. Lubachevsky and F. Stillinger. Geometric Properties of Random Disk Packings. *Journal of Statistical Physics*, 60(5,6):561–583, 1990.

[5] H.-J. Bungartz, M. Mehl, T. Neckel, and T. Weinzierl. The PDE framework Peano applied to fluid dynamics: An efficient implementation of a parallel multiscale fluid dynamics solver on octree-like adaptive Cartesian grids. *Computational Mechanics*, 46(1):103–114, June 2010. Published online.

[6] T. Neckel. *The PDE Framework Peano: An Environment for Efficient Flow Simulations*. Verlag Dr. Hut, 2009.

[7] T. Weinzierl. *A Framework for Parallel PDE Solvers on Multiscale Adaptive Cartesian Grids*. Verlag Dr. Hut, 2009.

[8] H. Sagan. *Space-filling curves*. Springer-Verlag, New York, 1994.

[9] J. Hron and S. Turek. Numerical benchmarking of fluid-structure interaction between elastic object and laminar incompressible flow. In H.-J. Bungartz and M. Schäfer, editors, *Fluid-Structure Interaction, Part II*, Lecture Notes in Computational Science and Engineering. Springer-Verlag, 2010. to appear.

[10] Hans-Joachim Bungartz, Bernhard Gatzhammer, Michael Lieb, Miriam Mehl, and Tobias Neckel. Towards multi-phase flow simulations in the pde framework peano. *Computational Mechanics*, 48(3):365–376, 2011.

[11] T. Aoki. Interpolated Differential Operator (IDO) scheme for solving partial differential equations. *Comput. Phys. Comm.*, 102:132–146, 1997.

[12] Y. Imai, T. Aoki, and K. Takizawa. Conservative form of interpolated differential operator scheme for compressible and incompressible fluid dynamics. *Journal of Computational Physics*, 227:2263–2285, 2008.

[13] A. J. Chorin. Numerical solution of the Navier-Stokes equations. *Math. Comp.*, 22:745–762, 1968.

[2] `http://www.kaust.edu.sa/research/labs/supercomputing.html`

[3] Center for Subsurface Imaging and Fluid Modeling consortium, King Abdullah University, Thuwal 23955-6900, Kingdom of Saudi Arabia

[14] B. Gatzhammer. *Efficient and Flexible Partitioned Simulation of Fluid-Structure Interactions*. to appear. PhD thesis.