Journal of Computational Physics 281 (2015) 970–981

Contents lists available at ScienceDirect

Journal of Computational Physics

www.elsevier.com/locate/jcp

Parallel domain decomposition method with non-blocking communication for flow through porous media

Andreas Lemmer, Rudolf Hilfer

Institute for Computational Physics, University of Stuttgart, Allmandring 3, 70569 Stuttgart, Germany

ARTICLE INFO

Article history:
Received 19 December 2013
Received in revised form 8 August 2014
Accepted 21 August 2014
Available online 10 September 2014

Keywords:
Domain decomposition method
Parallel computing
Porous media
Stokes equation
SIMPLE algorithm

ABSTRACT

This paper introduces a domain decomposition method for numerically solving the Stokes equation for very large, complex geometries. Examples arise from realistic porous media. The computational method is based on the SIMPLE (Semi-Implicit Method for Pressure Linked Equations) algorithm which uses a finite-differences approach for discretizing the underlying equations. It achieves comparable speed and efficiency as lattice Boltzmann methods. The domain decomposition method splits a large three-dimensional region into slices that can be processed in parallel on multi-processor computation environments with only minimal communication between the computation nodes. With this method, the flow through a porous medium with grid sizes up to 2048^3 voxel has been calculated.

1. Introduction

Although many algorithms and methods exist to solve the Stokes equation for creeping flow through porous media only few of these can be used for large systems such as computer tomograms representing three-dimensional complex geometries with $2048 \times 2048 \times 2048$ discrete volume elements (voxel) arranged on a cubic grid. Such large three-dimensional images are routinely generated by synchrotron or other imaging devices. Recently, three-dimensional images as large as $32\,768 \times 32\,768 \times 32\,768$ voxel have become available to the scientific community [1]. Finite difference approximations of Stokes equations for these large geometries result in linear systems with 10^{10}–10^{13} degrees of freedom. Because of these very large numbers of unknowns a method that does not require the system matrix is needed. In this paper, a pressure correction algorithm on a staggered grid [2,3], [4, pp. 179–187, 209–218], [5, pp. 96–98, 109–122], [6,7] is used, which is based on the SIMPLE algorithm [8,9].

Depending on the system size, the numerical solution of a large system with 2048^3 voxel cannot be computed on single-processor computers because of the huge memory requirements and computational effort. Instead it has to be distributed on many cores of a multi-processor computation environment. To benefit from all advantages of such an environment and to achieve an acceptable performance, an efficient domain decomposition method for this distribution is necessary.

The new domain decomposition method presented in this paper realizes an efficient parallelization with only non-blocking communication between the computation nodes. It was developed for the Stokes systems following ideas in [10,11] and then optimized to have the lowest possible memory requirement. With this algorithm, we are indeed able to numerically solve problems with system sizes up to $4096^3 = 68\,719\,476\,736$ voxel. Here we present results for $2048^3 = 8\,589\,934\,592$ voxel. To our knowledge this is the largest such computation at present.

http://dx.doi.org/10.1016/j.jcp.2014.08.032
0021-9991/

2. Definition of the problem

2.1. Continuum formulation

Consider a three-dimensional porous sample shaped as a cube with side length L $(0 < L < \infty)$

$$\mathbb{S} = \left\{(x, y, z) \in \mathbb{R}^3 : 0 \le x \le L,\ 0 \le y \le L,\ 0 \le z \le L\right\} \tag{1}$$

as a subset $\mathbb{S} \subset \mathbb{R}^3$. The sample contains two closed subsets $\mathbb{P}, \mathbb{M}$, the pore space $\mathbb{P} \subset \mathbb{S}$ and the matrix (rock) phase $\mathbb{M} \subset \mathbb{S}$ such that $\mathbb{S} = \mathbb{M} \cup \mathbb{P}$. It is assumed that both sets are path-connected. The two closed sets have the common boundary[1] $\partial\mathbb{P} = \partial\mathbb{M} = \mathbb{P} \cap \mathbb{M}$. The boundary of the sample is denoted by

$$\partial\mathbb{S} = \partial\mathbb{S}_{x,0} \cup \partial\mathbb{S}_{x,L} \cup \partial\mathbb{S}_{y,0} \cup \partial\mathbb{S}_{y,L} \cup \partial\mathbb{S}_{z,0} \cup \partial\mathbb{S}_{z,L} \tag{2}$$

with

$$\partial\mathbb{S}_{x,0} = \left\{(x, y, z) \in \mathbb{S} : x = 0\right\} \tag{3a}$$
$$\partial\mathbb{S}_{x,L} = \left\{(x, y, z) \in \mathbb{S} : x = L\right\} \tag{3b}$$
$$\partial\mathbb{S}_{y,0} = \left\{(x, y, z) \in \mathbb{S} : y = 0\right\} \tag{3c}$$
$$\partial\mathbb{S}_{y,L} = \left\{(x, y, z) \in \mathbb{S} : y = L\right\} \tag{3d}$$
$$\partial\mathbb{S}_{z,0} = \left\{(x, y, z) \in \mathbb{S} : z = 0\right\} \tag{3e}$$
$$\partial\mathbb{S}_{z,L} = \left\{(x, y, z) \in \mathbb{S} : z = L\right\}. \tag{3f}$$

Applying a small pressure gradient across a fluid filled sample $\mathbb{S}$ induces slow laminar flow through the pore space $\mathbb{P}$. In the long-time limit the flow becomes stationary. If the fluid is incompressible the resulting stationary and laminar flow field is also solenoidal and governed by the three-dimensional time-independent Stokes equation with no-slip boundary conditions

$$-\vec{\nabla} p(\vec{x}) + \Delta \vec{v}(\vec{x}) = \vec{0}, \quad \vec{x} \in \mathbb{P} \tag{4a}$$
$$\vec{\nabla} \cdot \vec{v}(\vec{x}) = 0, \quad \vec{x} \in \mathbb{P} \tag{4b}$$
$$\vec{v}(\vec{x}) = \vec{0}, \quad \vec{x} \in \partial\mathbb{P} \tag{4c}$$

where $p(\vec{x})$ is the dimensionless pressure field and $\vec{v}(\vec{x})$ the dimensionless velocity field. If applied body forces, sources, sinks or more general boundary conditions with slip are allowed the inhomogeneous Stokes problem would result where the right hand sides of Eqs. (4) would be nonzero.

For the calculations in this paper the stationary Stokes equations (4) have been made nondimensional in such a way that the gradient term and the Laplacian term in Eq. (4a) are of equal magnitude. There are several ways to do this. One method, used here, is to employ the macroscopic side length L in the gradient term, a typical microscopic "pore size" in the Laplacian term and to choose the units for velocity as $(P/L)(\ell^2/\mu)$, where P is the pressure drop across the sample and μ is the dynamic viscosity of the fluid.[2] This choice leads to Eq. (4a)

The boundary conditions on the six sides of the sample

$$p(\vec{x}) = p_0, \quad \vec{x} \in \partial\mathbb{S}_{z,0} \tag{5a}$$
$$p(\vec{x}) = p_L, \quad \vec{x} \in \partial\mathbb{S}_{z,L} \tag{5b}$$
$$\vec{v}(\vec{x}) = \vec{0}, \quad \vec{x} \in \partial\mathbb{S}_{i,j},\ i = x, y,\ j = 0, L \tag{5c}$$

correspond to an applied pressure drop of magnitude $|p_0 - p_L|/L$ in the z-direction driving the flow. There are closed impermeable walls on the four remaining faces, represented as no-flow conditions in Eq. (5c). The pressures p_0, p_L at the inlet and outlet are constant across the face.

The mathematical problem consists in finding solutions of these equations in a suitable space of functions. For the pressure field this is the Lebesgue space $L^2(\mathbb{P})$ of real valued square integrable functions on $\mathbb{P}$. For the velocity field it is the space $[H^1(\mathbb{P})]^3$ where $H^1(\mathbb{P})$ is the space of square integrable functions on $\mathbb{P}$ whose gradients are also square integrable. If the boundary $\partial\mathbb{P}$ is sufficiently smooth (e.g. a C^2-manifold) then existence and uniqueness of solutions follows from the variational formulation of the problem using the Lax–Milgram theorem [12].

[1] The boundary of a set is defined as the difference between its closure and its interior, the closure being the intersection of all closed sets containing the set, the interior being the union of all open sets contained in the set.

[2] For rock samples of $L = 10$ cm size with pore sizes around $\ell = 10^{-5}$ m the flow of water with $\mu = 10^{-3}$ Pa s under a pressure gradient of 1000 Pa will result in velocities measured as multiples of 10^{-3} m s^{-1}. For field scale simulations with grid blocks of size $L = 100$ m the same pressure gradient results in velocities of order 10^{-6} m s^{-1}.

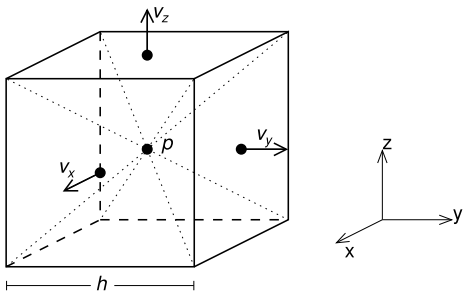

Fig. 1. One cell of the cubic lattice with lattice constant h illustrating the staggered arrangement of pressure and velocity. The pressure p is defined in the center of the cell at the position $\vec{x}$. The components of the velocity v_i are defined on their corresponding cell faces at the positions $\vec{x} + \frac{h}{2} \cdot \vec{e}_i$.

2.2. Discretization

The boundary value problem (4), (5) is discretized using finite differences and a three-dimensional cubic lattice with lattice spacing h. The cubic voxel $\mathbb{V}(k,l,m)$ are numbered by the integers $(k,l,m) \in \mathbb{Z}^3$ with $1 \le k,l,m \le N$ according to their position in the three-dimensional grid. The discretized pore space geometry is represented by a binary pore space indicator such that

$$\chi(k,l,m) = \begin{cases} 1, & |\mathbb{V}(k,l,m) \cap \mathbb{P}| > \alpha h^3 \\ 0, & \text{otherwise} \end{cases} \tag{6}$$

whenever the pore space volume inside a voxel $|\mathbb{V}(k,l,m) \cap \mathbb{P}|$ exceeds αh^3 where $0 \le \alpha \le 1$ is a threshold value. The notation $|\mathbb{G}| = \int_{\mathbb{G}} \mathrm{d}^3x$ denotes the volume of a set $\mathbb{G} \subset \mathbb{R}^3$. The threshold $\alpha = 0$ approximates the pore space volume from above, while $\alpha = 1$ approximates from below. In practice α is chosen using a segmentation algorithm that ensures a good approximation of the original and the discretized geometry according to one or several criteria. Here and in the following we assume the discretization $\chi(k,l,m)$ to be given.

The center of the voxel $\mathbb{V}(k,l,m)$ has the position vector

$$\vec{x}(k,l,m) = \left(k - \frac{1}{2}\right)h \cdot \vec{e}_x + \left(l - \frac{1}{2}\right)h \cdot \vec{e}_y + \left(m - \frac{1}{2}\right)h \cdot \vec{e}_z \tag{7}$$

where the unit vectors $\vec{e}_x$, $\vec{e}_y$ and $\vec{e}_z$ form the standard orthonormal basis in $\mathbb{R}^3$. The unknown pressure p and velocity fields v_i are then discretized in a staggered arrangement as

$$p(k,l,m) = p\big(\vec{x}(k,l,m)\big) \tag{8}$$

$$v_i(k,l,m) = v_i\left(\vec{x}(k,l,m) + \frac{h}{2} \cdot \vec{e}_i\right), \quad i = x, y, z \tag{9}$$

where by an abuse of notation we use the same symbol for the discretized and continuous fields. The staggered arrangement of the discretized fields is illustrated in Fig. 1.

The differential expressions $\vec{\nabla} p$ and $\Delta \vec{v}$ and $\vec{\nabla} \cdot \vec{v}$ are discretized in their lowest-order symmetrical form. Thus

$$(\vec{\nabla} p)_x(k,l,m) = p(k+1,l,m) - p(k,l,m) \tag{10}$$

where for the y- resp. z-components $k+1$ becomes k and l resp. m become $l+1$ resp. $m+1$. The discretized Laplacian applied to v_x reads

$$\begin{aligned}(\Delta v_x)(k,l,m) = {} & -6v_x(k,l,m) + v_x(k+1,l,m) + v_x(k-1,l,m) \\ & + v_x(k,l+1,m) + v_x(k,l-1,m) \\ & + v_x(k,l,m+1) + v_x(k,l,m-1)\end{aligned} \tag{11}$$

and analogously for y- and z-components. The divergence becomes

$$\begin{aligned}(\vec{\nabla} \cdot \vec{v})(k,l,m) = {} & v_x(k,l,m) - v_x(k-1,l,m) \\ & + v_y(k,l,m) - v_y(k,l-1,m) \\ & + v_z(k,l,m) - v_z(k,l,m-1).\end{aligned} \tag{12}$$

With this discretization, Eqs. (4a) and (4b) read

$$-(\vec{\nabla} p)_i(k,l,m) + (\Delta v_i)(k,l,m) = 0, \quad i = x, y, z \tag{13a}$$

$$(\vec{\nabla} \cdot \vec{v})(k,l,m) = 0. \tag{13b}$$

These are four equations per voxel for four unknowns per voxel, resulting in a linear system with $4N^3$ unknowns and the same number of equations. In the following when we refer to a "solution" or speak of "solving" the equations this means finding the unknowns $p(k,l,m), v_i(k,l,m)$ with $i = x, y, z$ for all voxel $\mathbb{V}(k,l,m)$.

The boundary conditions are implemented by introducing impermeable walls $\mathbb{V}(0,l,m), \mathbb{V}(N+1,l,m)$ in the x-direction at $k = 0, k = N+1$ with $0 \le l, m \le N+1$, impermeable walls $\mathbb{V}(k,0,m), \mathbb{V}(k,N+1,m)$ in the y-direction at $l = 0, l = N+1$ with $0 \le k, m \le N+1$ and fully permeable walls $\mathbb{V}(k,l,0), \mathbb{V}(k,l,N+1)$, called "inlet" and "outlet" planes, at $m = 0, m = N+1$ with $1 \le k, l \le N$ in the z-direction by setting

$$\chi(0,l,m) = 0, \quad 0 \le l, m \le N+1 \tag{14a}$$

$$\chi(N+1,l,m) = 0, \quad 0 \le l, m \le N+1 \tag{14b}$$

$$\chi(k,0,m) = 0, \quad 0 \le k, m \le N+1 \tag{14c}$$

$$\chi(k,N+1,m) = 0, \quad 0 \le k, m \le N+1 \tag{14d}$$

$$\chi(k,l,0) = 1, \quad 1 \le k, l \le N \tag{14e}$$

$$\chi(k,l,N+1) = 1, \quad 1 \le k, l \le N. \tag{14f}$$

With these specifications, the no-flow and no-slip boundary conditions (4c) and (5c) for the velocity field are implemented by setting

$$\left.\begin{aligned} v_x(k,l,m) &= 0 \\ v_y(k+1,l,m) &= -v_y(k,l,m) \\ v_z(k+1,l,m) &= -v_z(k,l,m) \end{aligned}\right\} \quad \text{if} \quad \left\{\begin{aligned} &\chi(k,l,m) = 1 \\ &\text{and} \\ &\chi(k+1,l,m) = 0, \end{aligned}\right. \tag{15a}$$

$$\left.\begin{aligned} v_x(k,l,m) &= 0 \\ v_y(k,l,m) &= -v_y(k+1,l,m) \\ v_z(k,l,m) &= -v_z(k+1,l,m) \end{aligned}\right\} \quad \text{if} \quad \left\{\begin{aligned} &\chi(k,l,m) = 0 \\ &\text{and} \\ &\chi(k+1,l,m) = 1, \end{aligned}\right. \tag{15b}$$

$$\left.\begin{aligned} v_x(k,l+1,m) &= -v_x(k,l,m) \\ v_y(k,l,m) &= 0 \\ v_z(k,l+1,m) &= -v_z(k,l,m) \end{aligned}\right\} \quad \text{if} \quad \left\{\begin{aligned} &\chi(k,l,m) = 1 \\ &\text{and} \\ &\chi(k,l+1,m) = 0, \end{aligned}\right. \tag{15c}$$

$$\left.\begin{aligned} v_x(k,l,m) &= -v_x(k,l+1,m) \\ v_y(k,l,m) &= 0 \\ v_z(k,l,m) &= -v_z(k,l+1,m) \end{aligned}\right\} \quad \text{if} \quad \left\{\begin{aligned} &\chi(k,l,m) = 0 \\ &\text{and} \\ &\chi(k,l+1,m) = 1, \end{aligned}\right. \tag{15d}$$

$$\left.\begin{aligned} v_x(k,l,m+1) &= -v_x(k,l,m) \\ v_y(k,l,m+1) &= -v_y(k,l,m) \\ v_z(k,l,m) &= 0 \end{aligned}\right\} \quad \text{if} \quad \left\{\begin{aligned} &\chi(k,l,m) = 1 \\ &\text{and} \\ &\chi(k,l,m+1) = 0 \end{aligned}\right. \tag{15e}$$

$$\left.\begin{aligned} v_x(k,l,m) &= -v_x(k,l,m+1) \\ v_y(k,l,m) &= -v_y(k,l,m+1) \\ v_z(k,l,m) &= 0 \end{aligned}\right\} \quad \text{if} \quad \left\{\begin{aligned} &\chi(k,l,m) = 0 \\ &\text{and} \\ &\chi(k,l,m+1) = 1 \end{aligned}\right. \tag{15f}$$

The pressure boundary conditions (5a), (5b) are implemented by enforcing $(1 \le k, l \le N)$

$$p(k,l,0) = 2p_0 - p(k,l,1) \tag{16a}$$

$$p(k,l,N+1) = 2p_L - p(k,l,N) \tag{16b}$$

in the inlet and outlet planes. These conditions are enforced for each step throughout the iterative algorithm described in the next section and they ensure that the pressure is constant at the boundaries $\partial\mathbb{S}_{z,0}$ and $\partial\mathbb{S}_{z,L}$.

3. Method of solution

3.1. Iterative numerical solution (SIMPLE algorithm)

The discretized equations (13)–(16) are solved numerically using a pressure-correction method based on the so called Semi-Implicit Method for Pressure Linked Equations (SIMPLE) [9]. The iterative SIMPLE algorithm for incompressible fluids effectively replaces Eq. (4a) with the time-dependent Stokes equation. With the choice of units in (4) the dimensions are fixed and the dimensionless form of the time-dependent Stokes equation reads

$$\frac{\rho \ell^3 P}{\mu^2 L} \frac{\partial \vec{v}(\vec{x},t)}{\partial t} = -\vec{\nabla} p(\vec{x},t) + \Delta \vec{v}(\vec{x},t), \tag{17}$$

where ρ is the density of the fluid.[3] The dimensionless number $\frac{\rho \ell^3 P}{\mu^2 L} \approx 10^{-2}$ rescales the dimensionless time which is measured in units of L/V where V is the typical velocity (see footnote 2). In the SIMPLE algorithm this rescaling factor is set to unity. The time evolution of the solution $p(\vec{x},t), \vec{v}(\vec{x},t)$ converges towards the stationary solutions $p(\vec{x}), \vec{v}(\vec{x})$ in the limit of large times.

The solution algorithm consists of the following schematic steps:

1. Initialize (at step $n=0$) the pressure and velocity fields as $p^0(k,l,m)$ and $\vec{v}^0(k,l,m)$ for all voxel as detailed (in Section 4) below.
2. Given the velocity $\vec{v}^n$ and pressure p^n (at step n), calculate an intermediate velocity $\vec{v}^*$ according to
$$\vec{v}^* = \vec{v}^n + \delta t \cdot \left(-\vec{\nabla} p^n + \Delta \vec{v}^n\right) \tag{18}$$
for a sufficiently small time interval δt. Calculate an error indicator
$$\epsilon_{v^*} = \sqrt{\sum_{i=x,y,z} \left\| v_i^* - v_i^n \right\|^2}. \tag{19}$$
3. Given $\vec{v}^*$, solve the Poisson equation (also called pressure-correction equation) for the correction field p'
$$\Delta p' = \frac{1}{\delta t} \vec{\nabla} \cdot \vec{v}^* \tag{20}$$
up to a certain accuracy $\epsilon_{SOR} \leq \beta \epsilon_{v^*}$ with β an empirically determined parameter and ϵ_{SOR} the error given by the PSOR-algorithm employed to solve this equation (see below).
4. Update the pressure and velocities to the timestep $n+1$ according to
$$p^{n+1} = p^n + \alpha_p \cdot p' \tag{21a}$$
$$\vec{v}^{n+1} = \vec{v}^* - \alpha_v \cdot \delta t \vec{\nabla} p' \tag{21b}$$
with underrelaxation factors for pressure α_p and velocity α_v.
5. Calculate suitable "error" resp. convergence measures
$$\epsilon_p = \left\| p^{n+1} - p^n \right\| \tag{22a}$$
$$\epsilon_v = \sqrt{\sum_{i=x,y,z} \left\| v_i^{n+1} - v_i^n \right\|^2} \tag{22b}$$
for the deviation of the unknown fields from the $n \to \infty$ limit.
6. If both errors $\epsilon_p, \epsilon_v < \eta$ are below a given level of tolerance η, then stop the iteration, else continue and go to step 2 above.

The underrelaxation factors α_p, α_v for pressure and velocity in Eqs. (21) are empirically determined beforehand. They are kept constant for all system sizes and geometries.

Steps 2–6 of the algorithm are iterated until the difference between consecutive steps (here called "error") falls below a certain threshold or level of tolerance, i.e. until the stationary solution is approached sufficiently accurately. The timestep δt cannot be chosen arbitrarily large. In [5] it was shown that for reasons of stability $\delta t \leq 1/6$ must hold, because the scheme is only semi-implicit but not fully implicit. It was found here that the theoretical restriction $\delta t \leq \frac{1}{6}$ can be loosened slightly because of the smoothing character of the SOR algorithm. A value of $\delta t = 0.19$ was found to give convergence and the fastest convergence rate.

Most of the computation time is spent on the solution of the pressure-correction equation (20). Therefore, it is important to choose an algorithm that performs well for this task. Here, an adapted (3D) version of the Parallel Successive Overrelaxation (PSOR, see [10,11]) algorithm is used, because it has the least memory requirements, a good convergence rate and shows the most efficient parallelization. It is superior to gradient methods because of the lack of efficient parallel preconditioners for gradient methods. To achieve good performance with the SOR algorithm, the asymptotically optimal overrelaxation factor has to be determined [13,14] as described in Section 4. For additional performance gain, the overrelaxation factor is Chebyshev accelerated [15].

There exist several well-known improvements over the original SIMPLE algorithm such as the SIMPLER algorithm [16–18] and other variations [19–22]. These variations have been tested in this work, but were found to be less efficient than the original algorithm when combined with our domain decomposition method discussed next. This is due to the fact that

[3] For water ($\rho = 1000\ \mathrm{kg\,m^{-3}}$) the dimensionless number is $\frac{\rho \ell^3 P}{\mu^2 L} \approx 10^{-2}$ (or 10^{-5} for field scale) if the same values as in footnote 2 are used for the other parameters. The typical time scale for lab scale flows is $L/V \approx 10^{-1}/10^{-3} = 10^2$ s (resp. $10^2/10^{-6} = 10^8$ s for flows on the field scale).

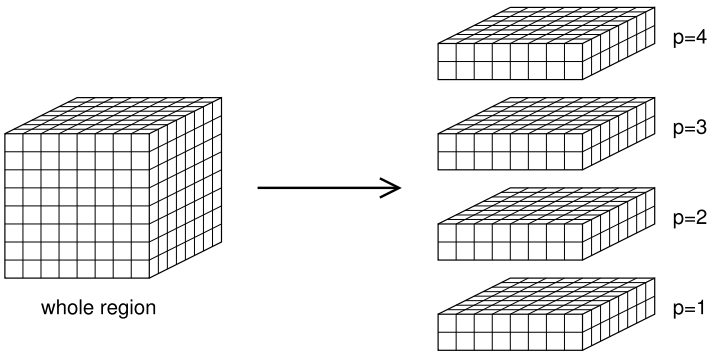

Fig. 2. Decomposition of a cubic domain with 8^3 grid cells on 4 nodes. Each slice with $8*8*2$ cells is assigned to one computation node.

they require either additional temporary variables or additional computation steps thereby reducing the parallel efficiency of these more advanced methods.

Other applications of similar algorithms for similar problems can be found in [23,24]. Alternative methods for solving the Stokes equation or equivalent equations are for example the Lattice–Boltzmann method [6,25,26] or other particle-based methods [27].

3.2. Domain decomposition

Let Π denote the number of available nodes, each node containing Ξ processors which can access the shared memory efficiently, and choose N such that

$$M = \frac{N}{\Pi} \tag{23}$$

is an even number. The sample $\mathbb{S}$ is decomposed along the z-axis into Π *slices* (or layers)

$$\mathbb{A}_p = \bigcup_{k=0}^{k=N+1} \bigcup_{l=0}^{l=N+1} \bigcup_{m=(p-1)M+1}^{pM} \mathbb{V}(k,l,m) \tag{24}$$

with $p = 1, \ldots, \Pi$ the number ("rank") of the node. These slices then have a size of $M(N+2)^2$ voxel. Each slice is assigned to one computation node. The communication between the nodes is implemented via the Message Passing Interface (MPI) (Fig. 2).

The slice $\mathbb{A}_p$ on computation node p with $p = 1, \ldots, \Pi$ is decomposed into M *planes*

$$\mathbb{B}_{pq} = \bigcup_{k=0}^{k=N+1} \bigcup_{l=0}^{l=N+1} \mathbb{V}\big(k,l,(p-1)M+q\big) \tag{25}$$

with $q = 1, \ldots, M$. The size of each plane is $(N+2)^2$ voxel. On each node, one ghost plane *above* (except for the uppermost slice) and one *below* (except for the lowermost slice) are added for parallelization. These ghost planes contain the data of the last plane of the node above and the data of the first plane of the node below, respectively.

Each plane $\mathbb{B}_{pq}$ is decomposed into $N+2$ *lines*

$$\mathbb{C}_{pqr} = \bigcup_{k=0}^{k=N+1} \mathbb{V}\big(k,r,(p-1)M+q\big) \tag{26}$$

with $r = 0, \ldots, N+1$. The size of each line is $N+2$ voxel.

Because the discretized Laplacian involves only next neighbor voxel (as seen in Eq. (11)), the decomposition into layers, planes and lines allows an efficient twofold parallelization as discussed in the next section. The layers will be solved in parallel on computer nodes containing several processors which in turn allow to treat planes and lines in parallel.

4. Program description

Fig. 3 shows the systematic program flow with the SIMPLE algorithm and the communication with MPI between the nodes.

The program is started with an MPI command that spawns one instance of the program on each computation node p. Each instance computes its computation region $\mathbb{A}_p$ based on its rank p in the parallel environment and the total number of

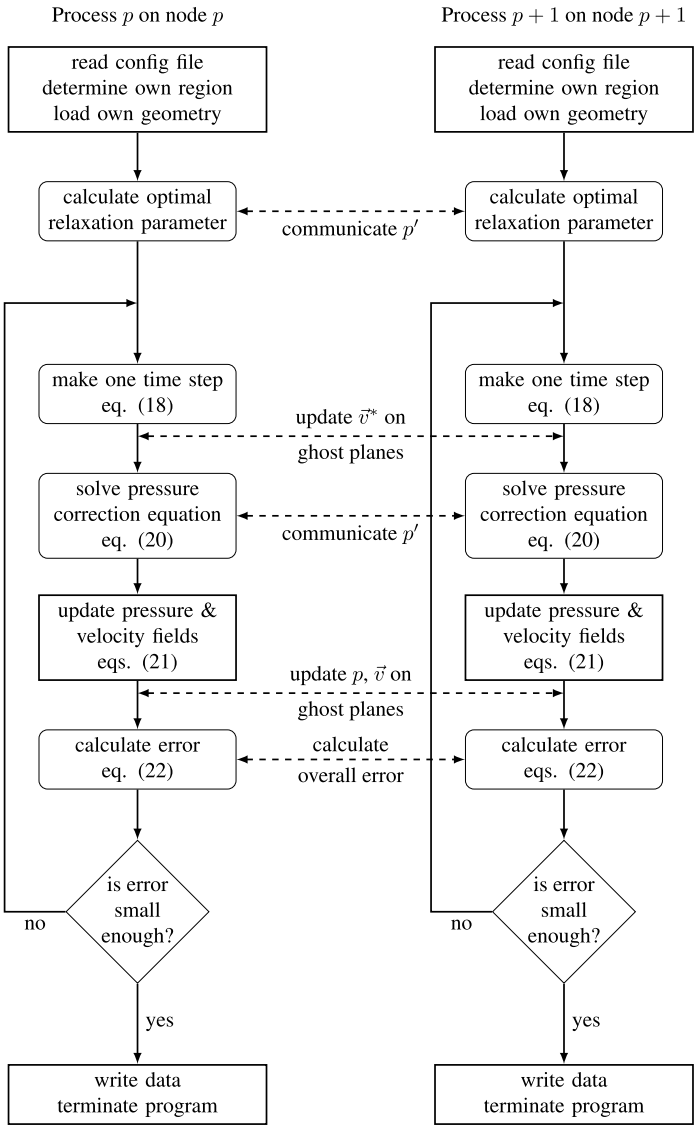

Fig. 3. Program description shown for the computation on nodes p and $p+1$. One instance of the program is started on each computation node. The communication between the nodes via MPI is shown as dashed arrows.

nodes Π according to Eq. (24). It loads its geometry from a binary file. One plane below $\mathbb{B}_{p\,0}$ and one plane above $\mathbb{B}_{p\,M+1}$ are added on each node that represent the ghost planes and the inlet and outlet planes.

After this is done, the optimal relaxation parameter is determined. This is done here by computing the spectral radius of the system matrix which highly depends on the underlying geometry of $\mathbb{P}$. The spectral radius is estimated by solving the Laplace equation

$$\Delta p'(k,l,m) = 0 \tag{27}$$

with the initial condition

$$p'(k,l,m) = 10^{50} \quad \text{for } 1 \le k,l,m \le N \text{ and } \chi(k,l,m) = 1 \tag{28}$$

and the boundary conditions

$$p'(k,l,0) = 0, \quad 1 \le k,l \le N \tag{29a}$$

$$p'(k,l,N) = 0, \quad 1 \le k,l \le N \tag{29b}$$

with the same SOR algorithm and the same parallelization as used for the pressure correction equation that is described below, using an arbitrary overrelaxation parameter ω. The value of p' decreases rapidly during the iterative estimation of

the optimal relaxation factor and should ideally be renormalized after each step. Here we have avoided this renormalizing of p' by choosing a sufficiently high initial value such that there is no danger of numerical underflows. Because the exact solution is known to be zero on all grid cells, the error after each iterative step can be given exactly as $\|p'\|$. This allows a precise specification of the error evolution from which follows the spectral radius ρ. After determining the spectral radius, the optimal overrelaxation factor is given by the equation

$$\omega_{opt} = \frac{2}{1+\sqrt{1-\rho^2}} \tag{30}$$

(see [13,14]).

Because the asymptotically optimal overrelaxation factor is not a good choice in the beginning of this SOR algorithm, the algorithm is started with a different (lower) factor and then is Chebyshev accelerated [15] towards the asymptotically optimal factor. A starting parameter of $\omega_0 = 2(\omega_{opt} - 1)$ has been found empirically as a good value.

In the beginning of the computation, the pressure field p is initialized with a linear pressure gradient along the z-axis

$$p(k,l,m) = \frac{2m-1}{2N}, \quad \chi(k,l,m) = 1 \tag{31a}$$

$$p(k,l,m) = 0, \quad \chi(k,l,m) = 0 \tag{31b}$$

that corresponds to the choices $p_0 = 0$ in Eq. (5a) and $p_L = 1$ in Eq. (5b). The velocity fields v_x, v_y, v_z are initialized to 0 on all cell faces:

$$v_x(k,l,m) = v_y(k,l,m) = v_z(k,l,m) = 0 \tag{32}$$

After these preparations the main loop of the SIMPLE algorithm starts. The solution of the pressure-correction equation (20) is the most time-consuming part of the SIMPLE algorithm. Its efficient parallelization is achieved by processing the planes $\mathbb{B}_{pq}$ in a specific sequence and by scheduling the communication between the computation nodes such that the idle times are minimized. This is achieved by the following sequence which is executed on each node p. Here "to process a plane (or line)" means to compute all unknowns for the collocation points corresponding to the voxels of the plane (or line).

1. Process the first plane $\mathbb{B}_{p1}$.
2. Except for the first node $p = 1$, start the communication of the data of $\mathbb{B}_{p1}$ to the preceding node $p-1$.
3. Process all remaining odd planes $\mathbb{B}_{pq}$, $q = 3, 5, \ldots, M-3, M-1$.
4. Except for the last node $p = \Pi$, wait for the communication from the succeeding node $p+1$ to be finished.
5. Process the last plane $\mathbb{B}_{pM}$.
6. Except for the last node $p = \Pi$, start the communication of the data of $\mathbb{B}_{pM}$ to the succeeding node $p+1$.
7. Process all remaining even planes $\mathbb{B}_{pq}$, $q = 2, 4, \ldots, M-4, M-2$.
8. Except for the first node $p = 1$, wait for the communication from the preceding node $p-1$ to be finished.

When processing a plane $\mathbb{B}_{pq}$, its lines $\mathbb{C}_{pqr}$, $r = 0, \ldots, N+1$ can also be processed in parallel with the Ξ processor cores of the node. For this intra-node parallelization, at first all odd lines $\mathbb{C}_{pqr}$, $r = 1, 3, \ldots, N+1$ are processed in parallel followed by all even lines $\mathbb{C}_{pqr}$, $r = 0, 2, \ldots, N$. This parallelization is realized via OpenMP in order to take advantage of the shared memory on a node. This results in a parallelization without communication or copying of memory regions. The whole sequence of calculations is represented graphically in Fig. 4.

With this domain decomposition and ordering, the time available for communication is maximized without interrupting the actual calculation. It has been shown [10,11] that this ordering has the same convergence rate as the Red–Black ordering for the Laplace equation. The minimal height of a slice is $M = 2$ planes (then with blocking communication), the optimal height depends on the processing speed of the planes, the communication rate and the latency and is at least $M = 4$ planes.

The time-evolution equation (18) is processed using the same ordering and scheduling of communication as described above.

After the pressure and velocity fields are updated to the next time step according to Eqs. (21), the error is estimated as described in Eqs. (22). When the error drops below a certain threshold, the SIMPLE algorithm stops and the solution is stored for a further investigation of the physical properties.

5. Results

The computations were performed with the parameters listed in Table 1.

All results are computed on the cluster bwGrid at HLR Stuttgart, Germany. The system is built of Intel Xeon E5440 processors, 2 processors (8 cores) per node, 16 GB RAM per node, Infiniband interconnect.

The underlying geometry is a discretization of a stochastic model for a Fontainebleau sandstone with a real side length of 1.5 cm [1], discretized at different resolutions $a = 8, 16, 32, 64, 128 \cdot a_0$, $a_0 = 0.91552734$ µm and a subsample of the same geometry which contains the first octant discretized at resolutions $a = 4, 8, 16, 32, 64 \cdot a_0$.

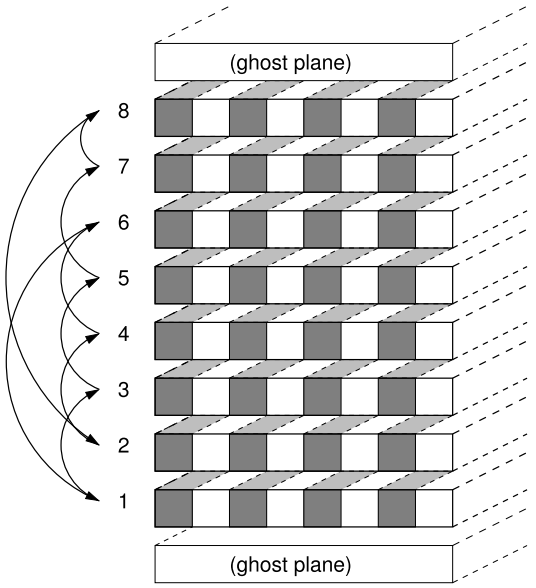

Fig. 4. Computational area of one node with 8 planes, each with a height of 1 voxel. Each plane consists of 8 lines that are shown as gray and white regions and are extended in the third dimension. The planes are computed sequentially in the order 1–3–5–7–8–2–4–6. In each plane, all odd (gray) lines or all even (white) lines can be computed in parallel.

Table 1
List of parameters for the numerical computations.

Parameter	Value
L	7.5 mm..15 mm
ϕ	≈15%
N	128..2048
Π	1..64
Ξ	4..8
$\Pi \cdot \Xi$	4..512
η	10^{-6}
δt	0.19
α_p	0.3
α_v	0.9
β	0.05

Results reported here are limited to results from the cluster bwGrid at HLR Stuttgart, because within this study only a short test run was available on the local CRAY XE6 (Hermit). In the test run a system of 4096^3 voxel was started on 256 nodes and 8192 processors. This amounted to 18 planes per slice and required roughly 16 GB per node. The test run had to be stopped before convergence due to a lack of computing time allocation.

From the domain decomposition strategy it is clear that every node should work on at least four planes, optimal are six or more planes. Accordingly the scaling is expected to remain optimal for any number of nodes as long as the number of nodes is a factor of 6 or more smaller than the number of planes (voxels in z-direction). The limiting factor for practical computations is the memory available per node. On the CRAY XE6 (Hermit) these are maximally 64 GB. The maximum number of nodes resp. processors are 3552 resp. 113 664. For a system of size $8192 \times 8192 \times 8192$ we expect optimal scaling to persist given that roughly 44 000 processors on 1365 nodes are available for computation. Such a calculation would require 28 GB per node, falling well within the capabilities of the CRAY XE6 (Hermit).

5.1. Dependence of speed and efficiency on system size

The total CPU time (time*procs) needed for obtaining the solution of the discretized Stokes equation is approximately proportional to $N^{4.5}$, where N is the side length of the sample (Fig. 5). The actual number of grid cells for which a computation is performed is the number of cells with $\chi(k,l,m) = 1$. It is approximately given by ϕN^3. This is multiplied by the effort for solving the pressure correction equation using the SOR method. The latter increases proportional to $N^{1.5}$ thus resulting in a total of $N^{4.5}$ for the algorithm as a whole.

It has been found that the prefactor in front of the $N^{4.5}$ law depends strongly on the geometry of the region to solve. This is apparently due to the fact that the pressure-correction algorithm converges more slowly when pore throats and pore

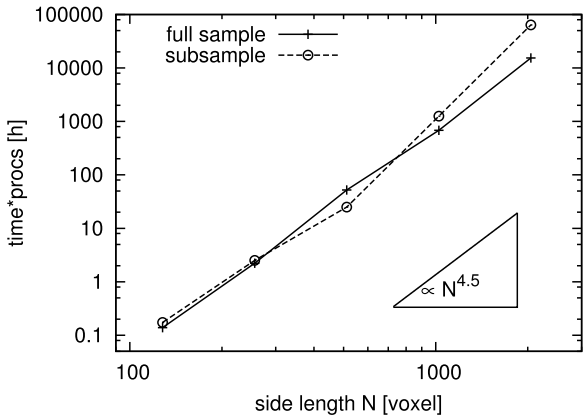

Fig. 5. Increasing total computation time with system side length.

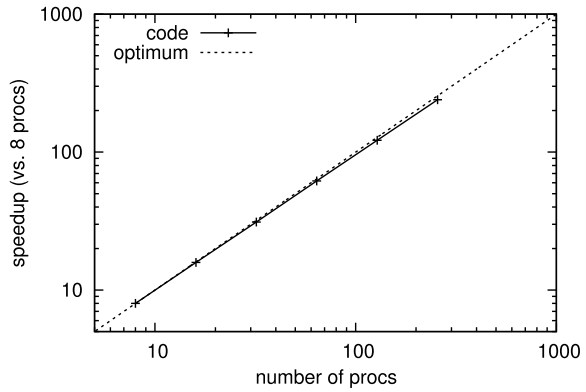

Fig. 6. Strong scaling. One region has been solved on 2, 4, 8, 16 and 32 nodes. The scaling shows a linear dependence as expected from an efficient algorithm.

bodies are resolved with more voxel, i.e. collocation points. This occurs here for subsamples with the same side length as the super-sample, because the resolution is twice as high.

5.2. Parallel efficiency

A cubic region with a side length of 256 voxel has been solved on 16, 32, 64, 128 and 256 processors to investigate the strong scaling behavior of the algorithm. As can be seen in Fig. 6, the algorithm shows a quasi-perfect behavior and the speedup is nearly linear with the number of processors used. This is due to the efficient parallelization with non-blocking communication.

Furthermore, the Stokes equation on a sample discretized at different resolutions resulting in system sizes of 256^3, 512^3 and 1024^3 voxel was calculated on 4, 32 and 256 processors to investigate the weak scaling. As can be seen in Fig. 7, this scaling behavior is suboptimal because of the effort for solving the pressure correction equation with the SOR algorithm increases as $N^{1.5}$.

5.3. Accuracy

The accuracy of the implementation has been tested by comparing the numerical solution to the well known quasi-analytical solution for a tube with cubic cross section [28]. It was found that the discretization error decreases with resolution. Its value is around one percent for a cross section of 16×16 voxel. For each voxel both the error in the velocity field $\vec{v}$ and in $\vec{\nabla} \cdot \vec{v}$ are computed. The program stops when either of them falls below $\eta = 10^{-6}$ (cf. Table 2).

6. Conclusion

The domain decomposition into planar slices and linear columns in combination with the SOR pressure-correction algorithm shows the good scaling behavior due to the implementation of strictly non-blocking communication. However, because of the method chosen for solving the Stokes equation, there is a strong dependence of the performance on the underlying geometry. If the throats in the geometry of the porous medium are very small (<8 grid nodes), there is a precision

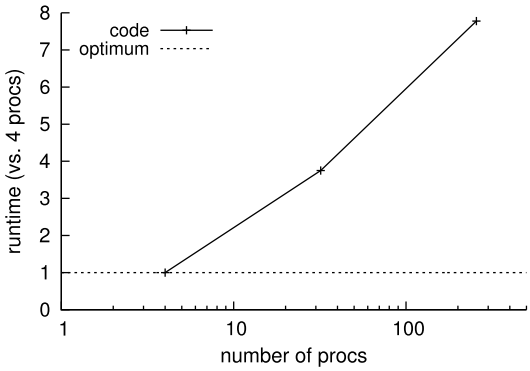

Fig. 7. Weak scaling. Because of the system size dependence, a constant (optimal) scaling cannot be achieved.

Table 2
Computation time for solving Stokes equation on geometries with different sizes. In the first column, f means full sample, s means subsample. The second column shows the side length of the cubic sample in voxel. The third column shows the resolution of the sample in units of $a_0 = 0.91552734$ µm. The forth column lists the total numbers of processors used. In the fifth column, the computation time can be found.

Sample	Side length N	Resolution $[a_0]$	Procs $(\Pi \cdot \Xi)$	Time [h]
f	128	128	4	0.03
f	256	64	4	0.56
f	512	32	64	0.79
f	1024	16	256	2.7
f	2048	8	256	60
s	128	64	4	0.04
s	256	32	32	0.08
s	512	16	64	0.39
s	1024	8	256	4.9
s	2048	4	512	125

loss in the result, but the algorithm performs extremely fast. If there are many large throats in the geometry, the algorithm is more precise, but significantly slower.

The Stokes flow through a realistic porous medium with a system size of $2048^3 = 8\,589\,934\,592$ voxel and about $5 \cdot 10^9$ unknowns was successfully computed. To the best of our knowledge this represents the largest such computations known at present.

Acknowledgement

This work has been supported by the Deutsche Forschungsgemeinschaft (SFB 716).

References

[1] R. Hilfer, T. Zauner, High-precision synthetic computed tomography of reconstructed porous media, Phys. Rev. E 84 (6) (2011) 062301.
[2] S. Armfield, Finite difference solutions of the Navier–Stokes equations on staggered and non-staggered grids, Comput. Fluids 20 (1) (1991) 1–17.
[3] C. Hirsch, Computational Methods for Inviscid and Viscous Flows, Numerical Computation of Internal and External Flows, vol. 2, Wiley, Chichester, 1988.
[4] J.H. Ferziger, M. Perić, Computational Methods for Fluid Dynamics, vol. 3, Springer, Berlin, 1996.
[5] C. Manwart, Geometrische Modellierung und Transporteigenschaften poröser Medien, Ph.D. thesis, Universität Stuttgart, 2001.
[6] C. Manwart, U. Aaltosalmi, A. Koponen, R. Hilfer, J. Timonen, Lattice–Boltzmann and finite-difference simulations for the permeability for three-dimensional porous media, Phys. Rev. E 66 (1) (2002) 016702.
[7] C. Manwart, R. Hilfer, Numerical simulation of creeping fluid flow in reconstruction models of porous media, Phys. A, Stat. Mech. Appl. 314 (1) (2002) 706–713.
[8] S.V. Patankar, D.B. Spalding, A calculation procedure for heat, mass and momentum transfer in three-dimensional parabolic flows, Int. J. Heat Mass Transf. 15 (10) (1972) 1787–1806.
[9] S.V. Patankar, Numerical Heat Transfer and Fluid Flow, Taylor & Francis, 1980.
[10] D. Xie, L. Adams, New parallel SOR method by domain partitioning, J. Sci. Comput. 20 (6) (1999) 2261–2281.
[11] D. Xie, A new block parallel SOR method and its analysis, SIAM J. Sci. Comput. 27 (5) (2006) 1513–1533.
[12] R. Dautry, J. Lions, Mathematical Analysis and Numerical Methods for Science and Technology, Springer, Berlin, 1993.
[13] D. Young, Iterative Solution of Large Linear Systems, Academic Press, New York, 1971.
[14] S. Yang, M.K. Gobbert, The optimal relaxation parameter for the SOR method applied to the Poisson equation in any space dimensions, Appl. Math. Lett. 22 (3) (2009) 325–331.
[15] D.M. Young, L.A. Hageman, Applied Iterative Methods, Academic, New York, 1981.
[16] S.V. Patankar, A calculation procedure for two-dimensional elliptic situations, Numer. Heat Transf. 4 (4) (1981) 409–425.
[17] J. Van Doormaal, G. Raithby, Enhancements of the SIMPLE method for predicting incompressible fluid flows, Numer. Heat Transf. 7 (2) (1984) 147–163.

[18] Y. Cheng, T. Lee, H. Low, W. Tao, Improvement of SIMPLER algorithm for incompressible flow on collocated grid system, Numer. Heat Transf., Part B, Fundam. 51 (5) (2007) 463–486.
[19] D. Jang, R. Jetli, S. Acharya, Comparison of the PISO, SIMPLER, and SIMPLEC algorithms for the treatment of the pressure-velocity coupling in steady flow problems, Numer. Heat Transf., Part A, Appl. 10 (3) (1986) 209–228.
[20] K. Karki, S. Patankar, Calculation procedure for viscous incompressible flows in complex geometries, Numer. Heat Transf., Part A, Appl. 14 (3) (1988) 295–307.
[21] R.-H. Yen, C.-H. Liu, Enhancement of the SIMPLE algorithm by an additional explicit corrector step, Numer. Heat Transf., Part B, Fundam. 24 (1) (1993) 127–141.
[22] H. Tao, W.-Q. Ozoe, A modified pressure-correction scheme for the SIMPLER method, MSIMPLER, Numer. Heat Transf., Part B, Fundam. 39 (5) (2001) 435–449.
[23] J.P. De Angeli, A.M. Valli, N.C. Reis Jr., A.F. De Souza, Finite difference simulations of the Navier–Stokes equations using parallel distributed computing, in: Proceedings of 15th Symposium on Computer Architecture and High Performance Computing, 2003, IEEE, 2003, pp. 149–156.
[24] B. Manshoor, M.N. Wan Hassan, N. Alias, Incompressible flow simulation using SIMPLE method on parallel computer, J. Sci. Technol. 1 (1) (2011).
[25] Z. Guo, T. Zhao, Lattice Boltzmann model for incompressible flows through porous media, Phys. Rev. E 66 (2002) 036304.
[26] J. Wang, X. Zhang, A.G. Bengough, J.W. Crawford, Domain-decomposition method for parallel lattice Boltzmann simulation of incompressible flow in porous media, Phys. Rev. E 72 (1) (2005) 016706.
[27] S. Ovaysi, M. Piri, Direct pore-level modeling of incompressible fluid flow in porous media, J. Comput. Phys. 229 (19) (2010) 7456–7476.
[28] K. Wieghardt, Theoretische Strömungslehre, 2. auflage ed., Teubner, Stuttgart, 1974.