Solving the Fluid Pressure Poisson Equation Using Multigrid—Evaluation and Improvements

Christian Dick, Marcus Rogowsky, and Rüdiger Westermann

Abstract—In many numerical simulations of fluids governed by the incompressible Navier-Stokes equations, the pressure Poisson equation needs to be solved to enforce mass conservation. Multigrid solvers show excellent convergence in simple scenarios, yet they can converge slowly in domains where physically separated regions are combined at coarser scales. Moreover, existing multigrid solvers are tailored to specific discretizations of the pressure Poisson equation, and they cannot easily be adapted to other discretizations.

In this paper we analyze the convergence properties of existing multigrid solvers for the pressure Poisson equation in different simulation domains, and we show how to further improve the multigrid convergence rate by using a graph-based extension to determine the coarse grid hierarchy. The proposed multigrid solver is generic in that it can be applied to different kinds of discretizations of the pressure Poisson equation, by using solely the specification of the simulation domain and pre-assembled computational stencils. We analyze the proposed solver in combination with finite difference and finite volume discretizations of the pressure Poisson equation. Our evaluations show that, despite the common assumption, multigrid schemes can exploit their potential even in the most complicated simulation scenarios, yet this behavior is obtained at the price of higher memory consumption.

Index Terms—Fluid simulation, multigrid, convergence analysis.

1 Introduction

Numerical simulation is widely adopted in films and computer games to enrich recorded imagery and virtual environments with realistic fluid effects. Yet ever higher resolutions and ever more complicated scenarios permanently increase the time required to compute these effects.

In the simulation of fluids governed by the incompressible Navier-Stokes equations—to our best knowledge the most widely used type of fluid simulation in computer graphics—a significant portion of simulation time is devoted to enforce mass conservation via solving the pressure Poisson equation (PPE). Common discretizations of the Navier-Stokes equations consider the partial differential equations at the points of a Cartesian grid and approximate the differential operators by finite differences, or subdivide the simulation domain into a finite set of subdomains (finite volumes), and reformulate the partial differential equations by considering the flows over the subdomain boundaries. The discretization of the PPE leads to a linear system of equations, which is often solved via the conjugate gradient (CG) method using simple preconditioners such as incomplete Cholesky factorization [1], and now more and more often via geometric multigrid methods (used as direct solvers or as CG preconditioners) [2], [3], [4].

In many cases (see, for instance, the scenario in Fig. 1(left)), geometric multigrid methods converge extremely well at significantly better rates than the CG method with simple preconditioners. However, multigrid methods can converge slowly in complicated domains where physically separated regions are combined on coarser grids, such as shown in Fig. 1(right). In such cases, the correction term that is computed on a coarse grid refers to a domain topology that is inconsistent with the initial topology. As a consequence the coarse grid correction becomes ineffective.

In this work we perform an exhaustive convergence study of existing geometric multigrid solvers for the PPE. This study indicates that in simple scenarios most existing multigrid solvers converge more or less equally well, yet in complicated scenarios a significant decrease of the convergence rate can be perceived. We demonstrate that in such scenarios the simulation time can be vastly dominated by the time required by the pressure solve, and that the convergence of geometric multigrid solvers can even be slower than the convergence of the CG method with simple preconditioners.

To improve multigrid convergence for a finite volume discretization of the Euler equations in a complicated domain, Aftosmis et al. [5] introduced the concept of split-cells. The key idea is to duplicate cells at coarser scales to effectively keep physically disconnected fluid parts distinct at these scales. Cell duplication is controlled by using a graph representation of the domain topology. Ferstl et al. [6] adopted this concept for solving the PPE in the context of a finite element discretization of the Navier-Stokes equations.

Our second contribution is a generalization of the concept of split-cells to common discretizations of the PPE on uniform grids: A standard finite difference discretization with the fluid boundaries being aligned along cell faces [7], and embedded boundary discretizations to handle curved free-surface and solid-wall boundaries [8], [9]. To achieve this we adapt the topology-aware approach by Aftosmis et al. and Ferstl et al. to obtain trilinear interpolation in the multigrid hierarchy when applied to discretizations with cell-centered pressure DOFs. Our approach achieves very

• *Christian Dick, Marcus Rogowsky, and Rüdiger Westermann are with the Computer Graphics and Visualization Group, Technische Universität München, Germany.*
E-mail: {dick, m.rogowsky, westermann}@tum.de

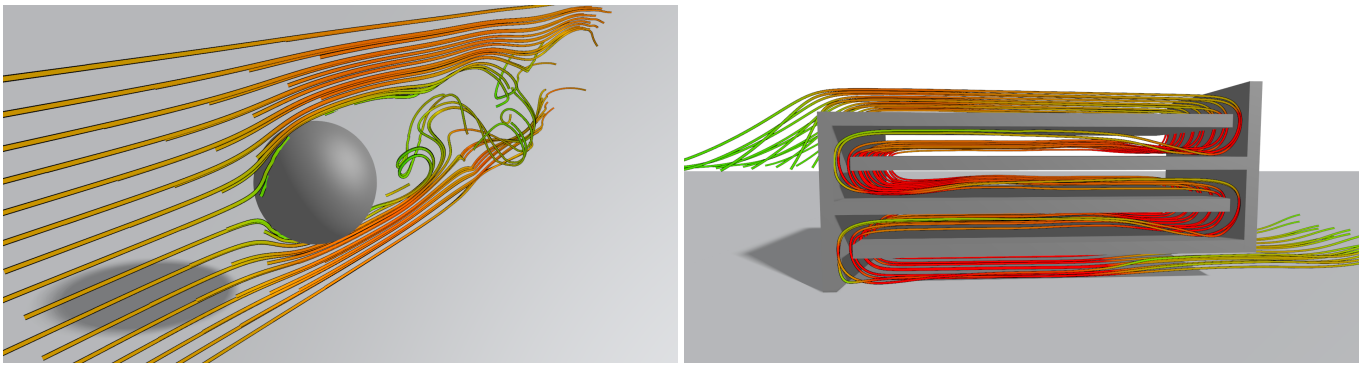

Fig. 1. The convergence rate of geometric multigrid methods for solving the pressure equation in fluid simulations can deteriorate significantly when going from a simple (left) to a more complicated domain (right). The colors of the particle trajectories encode increasing velocity values on a green to red color scale.

fast convergence independent of the geometry and topology of the simulation domain, yet it increases memory consumption. This renders the approach suitable for time-critical applications on well-equipped computing architectures that need to consider complicated domains. We also demonstrate, however, that for simple scenarios the additional investment can hardly be amortized.

For a discretization at hand, the proposed approach takes as input the computational stencils on the initial grid, and builds the hierarchy of coarse grids and the interpolation operators from the given domain geometry and topology (connectivity). The coarse grid and restriction operators are obtained algebraically using variational principles. This (primarily geometric) approach thus is related to algebraic multigrid schemes, which obtain the coarse grid and restriction operators in the same way, but in contrast use solely the information contained in the system matrix to determine the sets of coarse grid unknowns and the interpolation operators, neither explicitly taking into account the underlying differential equation nor the domain geometry and topology. We further analyze a purely algebraic multigrid approach for solving the PPE, and we compare its convergence behavior and performance to geometric multigrid approaches.

2 Related Work

We consider single-phase fluid simulation on Cartesian grids using discretizations of the incompressible Navier-Stokes equations. We cannot attempt here to survey the vast body of literature related to grid-based fluid simulation, for which our solver is designed. Yet for a thorough introduction to the field let us refer to the book by Bridson [10], where the principles and algorithms used in such simulations are discussed.

Following the work by Stam [11], we solve for the fluid velocity u in two passes. First, by ignoring the pressure term, we compute an intermediate velocity u^* by applying body forces, diffusion, and semi-Lagrangian advection. The latter uses trilinear or higher order velocity interpolation, or FLIP [12] to better compensate dissipation. Due to mass conservation, the final velocity field $u = u^* - \frac{dt}{\rho}\nabla p$ must be divergence free. This is achieved by interpreting the pressure p as a correction term, which is determined by solving the pressure Poisson equation (PPE) $-\Delta p = -\frac{\rho}{dt}\nabla \cdot u^*$. Here, dt is the length of the current time step, and ρ is the fluid density.

For the discretization of the incompressible Navier-Stokes equations on uniform grids different schemes can be employed. Due to its simplicity and resulting computational efficiency, a finite difference discretization on a staggered grid is often used [7], [13]. In its standard form, however, it models fluid boundaries that are aligned along the faces of the grid cells. Embedded boundary discretizations also handle curved (i.e., non-aligned to cell faces) free-surface and solid-wall boundaries corresponding to Dirichlet and Neumann boundary conditions in the PPE. In our work we consider the standard finite difference discretization, and the embedded boundary discretizations proposed by Batty et al. [8] and Ng et al. [9]. The former accounts for the fluid/solid coupling by using a variational (energy minimization) formulation of the pressure projection, which considers the fluid fractions within cells centered around the staggered locations of the velocity samples. The latter obtains a second-order accurate finite volume discretization of fluid/solid boundaries by considering the fractions of the cell faces occupied by the fluid. Both methods are combined with the ghost fluid method [14] to also handle curved fluid/air boundaries. The ghost fluid method introduces additional ('ghost') cell-centered pressure DOFs adjacent to these boundaries on the air side, and prescribes the pressure values obtained by linear interpolation between ghost and real pressure DOFs at the actual boundary locations.

The linear system of equations arising from the PPE discretization is often solved via the CG method using simple preconditioners, for instance, an incomplete Cholesky preconditioner [1], [10]. Although such black box solvers can be easily implemented and readily applied to any linear system of equations with symmetric positive definite system matrix, more sophisticated solvers such as multigrid methods can considerably reduce computing times (see, for instance, the recent tests performed by Setaluri and co-workers [15]). In particular, multigrid methods scale optimally in that they reach a certain residual tolerance in a number of operations that is linear in the number of unknowns (i.e., the number of required solver cycles is independent of the grid spacing). For thorough introductions to multigrid methods we recommend the books by Brandt and Livne [16], Briggs

et al. [17], and Trottenberg and co-workers [18]. For fluid simulations in computer graphics, multigrid methods were used as direct solvers [2], [4], [19], [20], or as preconditioners for the CG method [3], [6], [15].

Underlying geometric multigrid methods is the discretization of the governing equations on a hierarchy of successively coarser grids, and the prolongation and restriction of the quantities computed on these grid via specific inter-grid transfer operators. For discretizing the partial differential equation on the coarse grids, different strategies to handle the fluid boundaries were proposed.

McAdams et al. [3] treated obstacles at coarser scales via object coarsening, and performed boundary sweeps to explicitly smooth the system in the Dirichlet and Neumann boundary regions. Boundary sweeps in combination with a virtual node algorithm were later used by Hellrung et al. [21] to efficiently solve elliptic interface problems.

Johansen and Colella [22] used volume-weighted averaging of fine grid cells to handle embedded boundaries in coarse grid cells. Chentanez and Müller [4] employed the variational formulation by Batty et al. [8] combined with the ghost fluid method [14] to consider embedded boundaries on the coarse grids.

Instead of directly discretizing the partial differential equation on the coarse grids, an alternative approach is to follow a variational principle, according to which the coarse grid operators are obtained via Galerkin-based coarsening, and the restriction operators as transposes of the interpolation operators [17]. Note that in this case no explicit treatment of the boundary conditions is required on the coarse grids.

Weber et al. [23] proposed the embedding of the finite volume cut-cell formulation of Ng et al. [9] into a geometric multigrid scheme. By using piecewise constant interpolation instead of trilinear interpolation, in their approach Galerkin-based coarsening coincides with the direct discretization of the PPE on the coarse grids. We demonstrate in this paper the importance of trilinear interpolation for achieving improved convergence.

A distinct challenge is to handle complicated domain topologies on the coarse grids. Aftosmis et al. [5] duplicated cells at coarser scales by using a graph-based representation of the domain topology, in order to avoid merging physically disconnected domain parts. This strategy was later also employed by Ferstl et al. [6] to achieve improved convergence for a hexahedral finite element discretization of the PPE. To leverage multigrid for solving the Poisson equation on a mesh, Chuang and Kazhdan [24] used a nested hierarchy of regular Cartesian grids into each of which the mesh is embedded. They obtained a topology-aware finite element discretization on each level by enriching the FEM function spaces, which is mathematically equivalent to element duplication.

The paper is organized as follows: In the next section we describe how to make multigrid convergence independent of the complexity of the simulation domain for different kinds of PPE discretizations. We then perform an exhaustive convergence study using different solvers and PPE discretizations. We conclude the paper with some remarks on future work, and we give details on our specific implementation of the topology-aware multigrid approach.

3 Robust Geometric Multigrid

A geometric multigrid method combines a hierarchy of successively coarser grids, coarse grid and transfer operators, and a relaxation scheme (smoother). The idea behind multigrid is to accelerate a basic relaxation scheme by restricting the residual after a few relaxation steps to a coarser grid, computing the error on this grid, and interpolating the error back to the fine grid. Applying this principle recursively on a hierarchy of grids to determine the error leads to a basic grid traversal scheme referred to as V-cycle, which is outlined in Algorithm 1.

Algorithm 1

procedure V-CYCLE
  for $\ell = 0, \dots, L-1$ **do** ▷ Go down in the V-cycle
    if $\ell > 0$ **then** $x^\ell \leftarrow 0$ **end if**
    Relax $A^\ell x^\ell \approx b^\ell$ ▷ Apply smoother
    $r^\ell \leftarrow b^\ell - A^\ell x^\ell$ ▷ Compute residual
    $b^{\ell+1} \leftarrow R_\ell^{\ell+1} r^\ell$ ▷ Restrict residual
  end for
  Solve $A^L x^L = b^L$ directly ▷ Solve on coarsest level
  for $\ell = L-1, \dots, 0$ **do** ▷ Go up in the V-cycle
    $x^\ell \leftarrow x^\ell + I_{\ell+1}^\ell x^{\ell+1}$ ▷ Interpolate error & correct
    Relax $A^\ell x^\ell \approx b^\ell$ ▷ Apply smoother
  end for
end procedure

We propose a geometric multigrid method, which constructs a structured geometric hierarchy from a given uniform simulation grid, with the pressure DOFs being located at the centers of the grid cells. The method takes as input the information which cells carry a pressure DOF, as well as the 7-point stencils for these cells.

Whether a certain grid cell carries a pressure DOF depends on the discretization that is employed, and is determined as follows: In the standard finite difference discretization, obstacles are rasterized into the grid at the finest level, using the cells' midpoints for classification into solid and empty. In the simulation we distinguish between smoke simulation, where all empty cells are set to fluid cells, and liquid simulation, where an empty cell becomes a fluid cell when its midpoint is inside a tracked level-set boundary surface. Every fluid cell carries a pressure DOF. For curved boundaries according to Batty et al. [8] in combination with the ghost-fluid method [14], a cell is considered as solid if it is contained entirely in an obstacle. Non-solid fractions $\in [0, 1]$ are computed for every pair of face-adjacent cells. The conversion of empty cells into fluid cells is equal to the standard finite difference scheme, yet in addition to one pressure DOF for every fluid cell, we also assign a DOF to a solid cell if it is face-adjacent to a fluid cell and their associated non-solid fraction is non-zero. For curved boundaries according to Ng et al. [9] in combination with the ghost fluid method [14], a cell carries a DOF iff the intersection between the cell boundary with the interior of the fluid domain is non-empty. For simplicity we do not consider cells simultaneously filled with air, fluid, and solid, however, this can be achieved, for instance, by following the methods discussed in the book by Bridson [10].

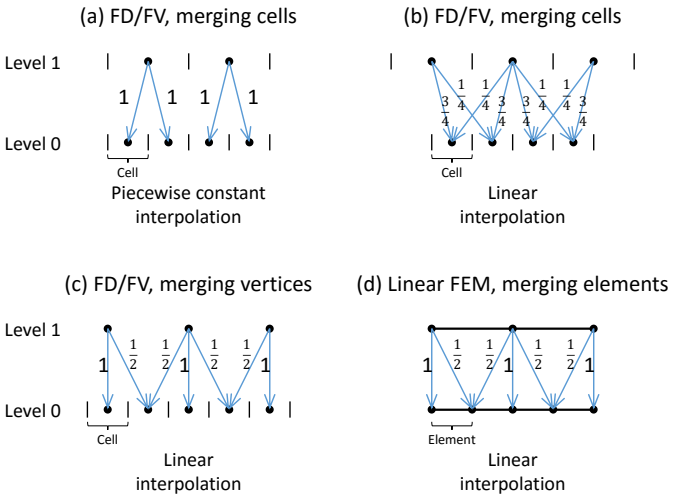

Fig. 2. Illustration of different multigrid hierarchies and accompanying interpolation operators. For simplicity, the 1D case is shown.

3.1 Multigrid Hierarchy—Principles

In general, the multigrid hierarchy is constructed by increasing the grid spacing by a factor of 2 in each dimension with each coarser level. The hierarchy construction can be described in terms of merging cells or merging vertices. For the considered finite difference/volume discretizations of the PPE, the unknowns (the pressure DOFs) are located at the cell centers of the simulation grid (referred to as a cell-centered discretization). Merging cells on a basis of adjacent 2^3-blocks, followed by introducing unknowns at the cell centers leads to a grid hierarchy where the unknowns at a certain level are located at distinct positions than the unknowns on the previous finer level (see Figure 2). Considering the multigrid interpolation operators, piecewise constant interpolation (case (a), used e.g. in [23]), or trilinear interpolation (case (b), used e.g. in [3], [4]) can be used on this hierarchy.

After introducing vertices at the cell centers of the simulation grid, a different hierarchy can be obtained by merging vertices on a basis of overlapping 3^3-blocks, followed by introducing unknowns at the vertices. For this hierarchy, the locations of the unknowns on a certain level are a subset of the locations of the unknowns on the previous finer level (case (c)). This naturally lends itself to trilinear interpolation.

Unfortunately, piecewise constant interpolation leads to reduced convergence rates in comparison to trilinear interpolation. In particular, multigrid theory suggests that the sum of the order of the restriction operator and the order of the interpolation operator (which both are 1 when using piecewise constant interpolation and the transpose of the interpolation operator as the restriction operator) should be greater than the order of the differential operator in the partial differential equation (which is 2 for the Poisson equation), in order to achieve good convergence [18, page 60]. The superior convergence resulting from trilinear interpolation over piecewise constant interpolation is also demonstrated in the convergence study presented in Section 4.

The standard approach for constructing the multigrid hierarchy is to take into account only the geometric locations of the cells (respectively vertices), i.e., a coarse grid cell (vertex) is created if at least one of the cells (vertices) of the respective 2^3-block (3^3-block) on the previous finer level exists, and all cells (vertices) of this block restrict to/interpolate from the considered coarse grid cell (vertex). Note that this approach does not take into account that cells (vertices) might belong to physically distinct parts of the simulation domain, for instance fluid regions separated by a wall. To address this issue, Aftosmis et al. [5] proposed a topology-aware coarsening strategy based on merging cells, which potentially creates multiple cells at the same location in order to represent separated domain parts on the coarser grids. This approach is limited to piecewise constant interpolation when used in the context of cell-centered finite difference/volume discretizations (case (a)), as in the original work. Note that it is not clear how to adapt the approach in order to enable trilinear interpolation as depicted in case (b). Ferstl et al. [6] adopted the strategy to enable topology-aware coarsening in the context of a finite element discretization. Using trilinear hexahedral elements automatically leads to trilinear interpolation operators (case (d)).

In the following, we present a novel topology-aware multigrid hierarchy for cell-centered finite difference/volume discretizations that leads to trilinear interpolation operators, and thus avoids the reduction of convergence rate resulting from piecewise constant interpolation. In contrast to the approach by Aftosmis et al. [5] which is based on merging cells (case (a)), our approach is based on merging vertices (case (c)).

3.2 Topology-Aware Multigrid Hierarchy Construction

We construct the geometric multigrid hierarchy successively from the finest to the coarsest level (level numbers $0, \dots, L$). On the finest level, the grid vertices exactly correspond to the pressure DOFs, which are located at the cell centers of the simulation grid and assigned as described above. By choosing the coordinate system appropriately, these vertices lie on the lattice $\mathbb{Z}^3$. With each coarser level, we double the grid spacing, such that the vertices on level ℓ lie on the lattice $2^\ell\mathbb{Z}^3$ (in particular, the vertices are aligned with the vertices on the previous levels).

To represent the domain topology on each level, we employ an undirected graph $G^\ell = (V^\ell, E^\ell)$, where V^ℓ denotes the set of grid vertices on level ℓ, and $E^\ell \subset \binom{V^\ell}{2}$ is a set of edges modeling the domain connectivity between 26-adjacent vertices. On the finest level, V^0 is given by the pressure DOFs, and two vertices are connected by an edge iff the cells containing the respective pressure DOFs are face-adjacent (see Figure 3).

To construct the respectively next coarser level ℓ from the previous level $\ell - 1$, we proceed as follows. For each potential coarse grid vertex position $x \in 2^\ell\mathbb{Z}^3$, we consider the subgraph of $G^{\ell-1}$ that is induced by the set of vertices that lie in the 3^3-block around x with respect to the lattice $2^{\ell-1}\mathbb{Z}^3$. We determine the connected components of this subgraph, and for each connected component, we create a distinct coarse grid vertex v at position x. Interpolation and restriction will later occur exactly between the vertices of a connected component, and its associated coarse grid

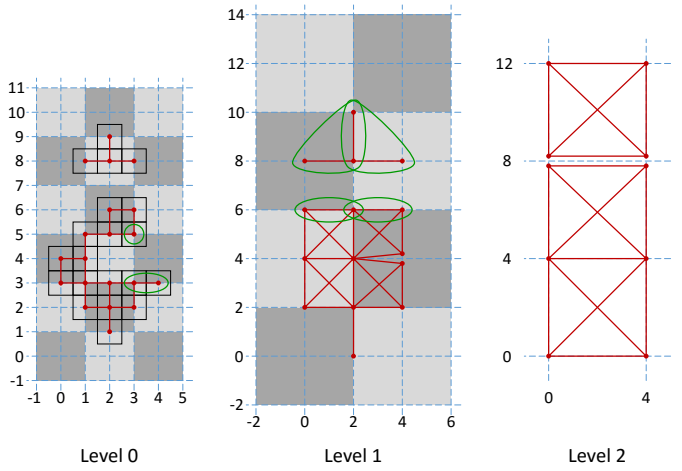

Fig. 3. Construction of the topology-aware multigrid hierarchy. For simplicity, the 2D case is shown. Red dots and lines denote the vertices and edges of the graphs used to represent the topology on each level. The blue dashed lattices indicate the possible vertex locations on each level. The black squares on level 0 depict the simulation cells carrying a pressure DOF. The overlapping 3^3-blocks of vertices (corresponding to 2^3-blocks of lattice cells) in which graph connected components are determined are indicated by the checkerboard pattern. For example, the 3^3-block centered around the vertex at position $(4, 4)$ on level 0 contains two connected components (indicated in green). For each connected component, a separate vertex at position $(4, 4)$ is created on level 1. The other 3^3-blocks on level 0 contain at most one connected component, thus there are no further duplicated vertices on level 1.

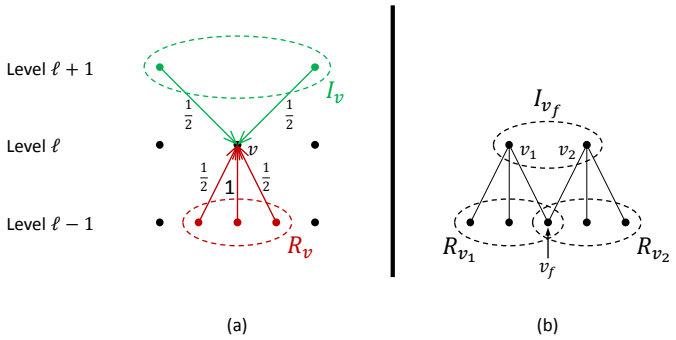

Fig. 4. (a) Illustration of the set R_v of vertices that restrict to a certain vertex v, and the set I_v of vertices from which v interpolates. (b) Illustration of $R_{v_1} \cap R_{v_2} \neq \emptyset \Leftrightarrow \exists v_f \in V^{\ell-1} : v_1, v_2 \in I_{v_f}$. Both illustrations show the 1D case for simplicity.

vertex (see Section 3.3). Since by construction no further vertex restricts to v, the connected component thus constitutes the entire set of vertices that restrict to v. This set is denoted by R_v (see Figure 4(a)). It is important to note that according to this construction principle, multiple vertices might reside at the same position, but each belongs to a topologically separated part of the simulation domain that is covered by the considered 3^3-block. After constructing the coarse grid vertices V^ℓ, we build E^ℓ by creating an edge $\{v_1, v_2\}$ between two coarse grid vertices $v_1, v_2 \in V^\ell$ iff $R_{v_1} \cap R_{v_2} \neq \emptyset$, i.e., the subgraphs associated with the two coarse grid vertices share at least one vertex. For completeness, let us here also introduce the set I_v of vertices a certain vertex v interpolates from (see Figure 4(a)). Since a vertex interpolates from exactly those vertices it restricts to, it is $I_v = \{v_c \in V^{\ell+1}; v \in R_{v_c}\}$ and $R_v = \{v_f \in V^{\ell-1}; v \in I_{v_f}\}$ for $v \in V^\ell$. Moreover, the statement that the subgraphs associated with the two coarse grid vertices share at least one vertex is equivalent to that there is a vertex on level $\ell-1$ that interpolates from both coarse grid vertices v_1 and v_2, i.e., $\{v_1, v_2\} \in E^\ell \Leftrightarrow R_{v_1} \cap R_{v_2} \neq \emptyset \Leftrightarrow \exists v_f \in V^{\ell-1} : v_1, v_2 \in I_{v_f}$ (see Figure 4(b)). As a consequence, the edges of the graph are implicitly given by the sets of interpolation vertices, and do not have do be determined and stored explicitly (see remarks on the implementation in the Appendix).

A different view of the constructed hierarchy considers the vertex associations successively over multiple levels, to explain the structure of the hierarchy in relation to the fluid domain topology. It can be shown that the sets of vertices on level ℓ that are associated with the (duplicated) coarse grid vertices on level ℓ_c ($\ell_c > \ell$) at a certain position constitute the connected components of the subgraph on level ℓ induced by the vertices in the $(2^{\ell_c-\ell+1}-1)^3$-block of the lattice $2^\ell\mathbb{Z}^3$ centered around the considered position. With respect to the finest level $\ell = 0$, this means that physically disconnected fluid parts remain disconnected on coarser scales, yet locally separated but globally connected parts are merged on one of the coarser levels. This merging seems counter-constructive at first glance, since it merges fluid parts across the boundaries of obstacles in the domain, yet it only happens on a level ℓ_c if these parts become connected in one of the $(2^{\ell_c+1}-1)^3$-blocks of the lattice $\mathbb{Z}^3$ centered around the potential coarse grid vertex positions $2^{\ell_c}\mathbb{Z}^3$.

3.3 Coarse Grid and Transfer Operators

We use trilinear interpolation operators $I^\ell_{\ell+1}$. The restriction and coarse grid operators are chosen according to the variational properties of multigrid [17], which are also used in the context of algebraic multigrid methods [18]. Considering the linear equation system $A^\ell x^\ell = b^\ell$ on level ℓ with current approximate solution $\tilde{x}^\ell$, the coarse grid correction $I^\ell_{\ell+1}x^{\ell+1}$ is determined such that the error is minimized with respect to the A-energy norm $\|x\|_A = \sqrt{x^{\mathrm{T}}Ax}$ (A symmetric, positive definite), corresponding to solving

$$\frac{\partial}{\partial x^{\ell+1}} \left\| x^\ell - (\tilde{x}^\ell + I^\ell_{\ell+1}x^{\ell+1}) \right\|^2_{A^\ell} = 0,$$

which is equivalent to solving

$$\underbrace{(I^\ell_{\ell+1})^{\mathrm{T}} A^\ell I^\ell_{\ell+1}}_{=:\, A^{\ell+1}} x^{\ell+1} = \underbrace{(I^\ell_{\ell+1})^{\mathrm{T}}}_{=:\, R^{\ell+1}_\ell} \underbrace{(b^\ell - A^\ell \tilde{x}^\ell)}_{=\, r^\ell},$$

where r^ℓ is the current residual. From this we can directly obtain the coarse grid operators by Galerkin-based coarsening according to $A^{\ell+1} = R^{\ell+1}_\ell A^\ell I^\ell_{\ell+1}$, and the restriction operators by transposition of the interpolation operators, i.e., $R^{\ell+1}_\ell = (I^\ell_{\ell+1})^{\mathrm{T}}$. The latter equation means that interpolation and restriction occur between the same pairs of vertices and using the same weights. Using this approach automatically leads to fully consistent coarse grid and transfer operators over the entire multigrid hierarchy. Note in particular that no explicit handling of boundary conditions on the coarse grids is necessary.

When using these coarse grid and transfer operators, it is required that the transfer operators have full rank, to make

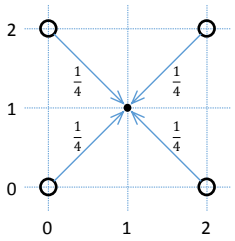

Fig. 5. 2D example showing an interpolation operator not having full rank. Here, the grid on level 0 consists of a single vertex, which is indicated by the small black dot. The vertices on level 1 are depicted by the black circles.

the resulting coarse grid operators symmetric positive definite. In combination with a converging smoothing scheme (such as Gauss-Seidel relaxation), it can then be shown that the convergence of the constructed multigrid solver is guaranteed [18]. Due to the particular construction of the coarse grid hierarchy we employ, the resulting transfer operators do not necessarily have full rank yet. A simple example is shown in Figure 5. To address this issue, we carefully remove some vertices on the coarse grids.

The transfer operators have full rank, iff the kernel of $I_\ell^{\ell-1}$ contains only the zero vector. This is achieved by means of the following algorithm, which ensures that the values at the vertices on level $\ell-1$ obtained by interpolation from given values at the vertices on level ℓ are zero, only if the latter are zero.

We process the levels successively, starting with level $\ell = 1$ and continuing with higher level numbers. Consider the processing of level ℓ. We employ a flag at each vertex on level ℓ, which can take the values 'free', 'zero', or 'removed'. The flags are initialized with 'free'. We then iterate over the vertices on level $\ell - 1$ in an order given by always choosing one of the remaining vertices for which the number of associated interpolation vertices (on level ℓ) that *in the current iteration* carry the 'free' flag is minimal. This order is obtained efficiently by using buckets and assigning the vertices according to this number. If the currently considered vertex has a 'free' interpolation vertex, we set one arbitrary 'free' interpolation vertex to 'zero', and the other 'free' interpolation vertices to 'removed'. By choosing this particular visitation order of the vertices, we seek to heuristically minimize the number of vertices that are removed, and thus to avoid a potential negative effect on the approximation quality of the coarse grids. In particular it is worth noting that for each vertex on level $\ell - 1$ that lies on the underlying lattice $2^\ell\mathbb{Z}^3$ of the next coarser level, its single interpolation vertex on this level is always preserved. Vertex removal thus can only occur at the fluid boundaries. After the iteration over the vertices is completed, the non-'zero' coarse grid vertices are removed (or masked), and the next level is processed. The interpolation weights are normalized, such that they sum up to 1 at each vertex.

It is important to note that although the coarse grid and transfer operators are written as matrices in the mathematical formulation, they are never explicitly built in our matrix-free implementation. In particular, the hierarchy is represented by means of sets of interpolation, restriction, and stencil vertices (the latter denoting the sets of vertices induced by the footprints of the numerical stencils). The interpolation and restriction operators are represented as scalar interpolation and restriction weights, associated with the interpolation and restriction vertices. Finally, the coarse grid operators are represented as scalar stencil coefficients, associated with the stencil vertices. Further details on our implementation are given in the Appendix.

3.4 Smoother and Grid Traversal

For smoothing, we use red-black Gauss-Seidel relaxation on the finest level (corresponding to 7-point stencils), and 8-color Gauss-Seidel relaxation on all other levels (corresponding to 27-point stencils). We employ standard V-cycles with one pre- and one post-smoothing Gauss-Seidel relaxation step. The colors during post-smoothing are traversed in reverse order than during pre-smoothing, which allows us to use the V-cycle as a preconditioner for the conjugate gradient method. Note that if the V-cycle is used as a CG preconditioner, x^0 must be initialized with 0 before performing the V-cycle. On the coarsest level, a conjugate gradient solver is used. The number of levels is chosen such that there are ≤ 1024 vertices on the coarsest level.

3.5 Memory Requirements

By using a topology-aware multigrid hierarchy and Galerkin-based coarsening instead of directly discretizing the partial differential equation on the individual levels, additional memory requirements arise. Specifically, using Galerkin-based coarsening, it is required to store the coefficients of the numerical stencils on levels $\ell \geq 1$, whereas for a direct discretization, the stencils can be assembled on-the-fly on all levels from the respective cell classification.

Moreover, since in the topology-aware hierarchy on levels $\ell \geq 1$ multiple vertices can reside at the same location, an indexed-based representation is required on these levels. Therefore, we enumerate the vertices and store for each vertex the indices of the stencil vertices, the indices of the restriction vertices, the respective restriction weights, as well as two offsets into the two arrays storing the indices (linearized over all vertices). To reduce the additional memory requirements, the sets of interpolation vertices are disposed after assembling the numerical stencils on the coarse grids, i.e., the respective memory can be shared by other data. In the multigrid V-cycle, interpolation therefore is realized by scattering the values from the coarse grid vertices on a certain level to the vertices on the next finer level, using the sets of restriction vertices. In contrast, for a standard hierarchy (without vertex duplication) an index-free representation can be used on all levels, since the stencil vertices, the restriction vertices, and the restriction weights are given implicitly by the regular Cartesian grid.

Considering the additional memory requirements, it is important to note that the number of vertices is reduced by a factor of $1/8$ with each coarser level. Therefore, storing the 27 coefficients per stencil on the coarse grids creates additional memory requirements of $(\frac{1}{8} + (\frac{1}{8})^2 + \dots) \cdot 27 \approx 3.9$ floating point values per pressure DOF. Storing the 27 indices of the stencil vertices, the 27 indices of the restriction vertices, the 27 restriction weights, and the 2 array offsets costs another $(\frac{1}{8} + (\frac{1}{8})^2 + \dots) \cdot (27 \cdot 4 + 27 \cdot (4 + 1) + 2 \cdot 4) \approx 36$ bytes

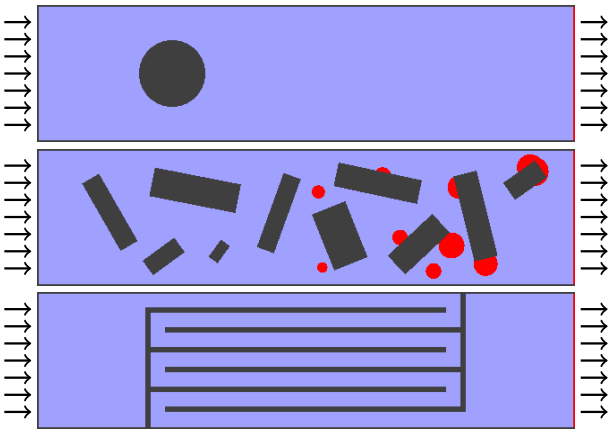

Fig. 6. Setup of the different scenarios used in the convergence tests (from top to bottom: 'sphere', 'city', 'maze' scenario). The figure shows the center horizontal cross section through the 3D wind tunnel. Light blue, dark gray, and red color denotes fluid, solids, and empty space, respectively. For the last two scenarios, the obstacles range over the entire vertical extent of the wind tunnel.

per pressure DOF, when using 32-bit indices, 8-bit weights (stored as fractions using 4 bits each for the numerator and the denominator), and 32-bit offsets. Since each vertex on average has $\frac{1+3\cdot2+3\cdot4+8}{8}$ associated interpolation vertices, storing the indices of the interpolation vertices, the interpolation weights, and 1 additional array offset would cost another $(1+\frac{1}{8}+(\frac{1}{8})^2+\dots)\cdot(\frac{27}{8}\cdot(4+1)+4) \approx 24$ bytes per pressure DOF. Note that all numbers are approximate, since duplicated vertices have not been counted. (Their number is typically rather small compared to the overall number of vertices.)

4 Convergence Study

In the following, we compare and analyze the convergence of different multigrid solvers in a number of experiments. Our tests have been run on a desktop PC, equipped with an Intel Xeon E5-1650 v2 six-core processor running at 3.5 GHz (because all compared multigrid solvers can be parallelized equally well, only a single core has been used) and 32 GB RAM (DDR3 1866 MHz). Double floating point precision has been used for all experiments. In all of our experiments, the 7-point stencils at the pressure DOFs have been precomputed and stored in main memory. As an alternative, for all solvers the stencils could also be assembled on-the-fly from a given cell classification.

4.1 Setup

The basic setup for our experiments is shown in Figure 6. We consider a virtual wind tunnel of size $16\,\mathrm{m} \times 4\,\mathrm{m} \times 4\,\mathrm{m}$, with inflow conditions on the left side (velocity $u_{\text{in}} = (30,0,0)^T\,\frac{\mathrm{m}}{\mathrm{s}}$), outflow conditions on the right side ($p = 0\,\mathrm{Pa}$), and slip conditions on all other sides. For the obstacles inside the wind tunnel, no-slip conditions are applied. The fluid density is set to $\rho = 1.2\,\frac{\mathrm{kg}}{\mathrm{m}^3}$. For the convergence experiments, the intermediate velocity before pressure projection has been set to $u^* = (-10,0,0)^T\,\frac{\mathrm{m}}{\mathrm{s}}$ over the entire domain, the time step length to $dt = 20\,\mathrm{ms}$, and the pressure has been initialized with $p = 0\,\mathrm{Pa}$.

To analyze the impact of the domain complexity on the solvers' performance, we use three different simulation scenarios (see Figure 6): In the first scenario, a single sphere-shaped obstacle is placed into the wind tunnel, a setup which is very often used in the literature. In the second scenario, the wind tunnel is populated by 10 box-shaped obstacles (which are rotated against the grid), resembling a city model. In addition, 50 sphere-shaped empty regions are inserted. While in the other two scenarios a Dirichlet boundary is present only on the right side (the outflow) of the wind tunnel, this scenario thus demonstrates the solvers' performance in the context of more complicated Dirichlet boundaries. In the third scenario, a maze-like obstacle is placed into the wind tunnel. Fluid simulations based on very similar scenarios are shown in Figures 1 and 9.

To demonstrate the generic adaptation of our approach, we use three different discretizations of the pressure Poisson equation: A standard finite difference discretization with grid-aligned fluid boundaries [7], and the embedded boundary discretizations according to Batty et al. [8] and Ng et al. [9] to handle curved solid-wall boundaries. The latter two are used in combination with the ghost-fluid method to support curved free-surface boundaries. To examine the solvers' scalability, we further employ two different grid resolutions: $300 \times 75 \times 75$ and $600 \times 150 \times 150$. Since our topology-aware multigrid hierarchy is tailored for cell-centered finite difference/volume discretizations, we do not consider finite element discretizations in our study. Note that a meaningful comparison of solver performances with respect to different discretizations would require to also take the accuracy of the respective discretizations into account.

4.2 Solvers

In our analysis, we consider the topology-aware, Galerkin-based multigrid solver presented in Section 3 as a direct solver (MG^+) or as a preconditioner for the conjugate gradient method (MGPCG^+). To analyze the impact of the topology-aware hierarchy (with vertex duplication, superscript $^+$), we have also included corresponding variants with a standard hierarchy (without vertex duplication, superscript $^-$) (see Section 3.1). To highlight the improved convergence behavior obtained via trilinear instead of piecewise constant interpolation operators, we have further included corresponding solver variants with piecewise constant interpolation operators (suffix pc). For these solvers, the hierarchy is built by merging cells (case (a) in Figure 2) rather than merging vertices (case (c) in Figure 2), and using cell duplication according to Aftosmis et al. [5] and Ferstl et al. [6] for the topology-aware variant.

For comparison, we have re-implemented and included the multigrid solvers proposed by McAdams et al. [3] and Chentanez and Müller [4]. Note that these solvers are conceptually limited to grid-aligned boundaries and the curved boundaries by Batty et al., respectively. To guarantee that all solvers use similar data structures and code optimizations, we have decided to use our own implementations. The solver implementation by McAdams et al. [25] has been extended to avoid padding—and performance losses thereof—in non-power-of-two cases. Yet we have carefully verified that the original and our implementation show

the same convergence behavior with respect to the number of performed V-cycles. The solver has been configured as described in the original work and the accompanying source code. The solver by Chentanez and Müller has been configured as described in the original work, yet only full multigrid cycles have been used for the convergence measurements (in the original work, a fixed number of 3 full multigrid cycles plus 4 V-cycles were used per simulation frame).

Moreover, we have included a conjugate gradient solver with incomplete Cholesky IC(0) preconditioner (ICPCG) and modified incomplete Cholesky MIC(0) preconditioner (MICPCG), implemented according to the book by Bridson [10].

Overall we feel confident that a fair comparison of all implementations with respect to convergence rate is possible. In particular, our implementations of the different solvers are based on the same code wherever possible (e.g., the surrounding CG solver, into which the respective multigrid cycle is embedded as a preconditioner).

The topology-aware multigrid solvers use Galerkin-based coarsening and restriction operators that are the transpose of the interpolation operators, both of which are also used in algebraic multigrid (AMG) methods. Therefore, we have also included an AMG solver in our study for comparison. Specifically, we have chosen the BoomerAMG solver from the Hypre library [26]. For this solver, the original implementation has been used. The solver has been configured as proposed in the supplied example for solving a Poisson equation (file ex5.c). We have used Falgout coarsening, hybrid symmetric Gauss-Seidel/SSOR smoothing, one pre- and one post-smoothing step on each level, and the multigrid cycle has been used as a CG preconditioner.

4.3 Results

The results of our experiments are given in the form of convergence graphs in Figures 7 and 8. Here we analyze the computing time by which a given residual tolerance is reached by a solver, rather than the number of cycles that are required. The reason is that a single cycle can be made very effective in reducing the residual, yet at the expense of significantly increasing its computing time. For example, increasing the number of smoothing steps in a geometric multigrid method can lead to a significant better residual reduction with respect to number of V-cycles. However, performing less expensive but more V-cycles might possibly lead to a better residual reduction with respect to computing time. Since computing time is the relevant factor in practical applications, we plot the $\|.\|_\infty$-norm of the residual (normalized by the $\|.\|_\infty$-norm of the residual at the start of the solver) therefore over the elapsed computing time, rather than over the cycle number. The staircases in the convergence plots correspond to the individual solver cycles.

A further important aspect is the solver setup time. For the geometric multigrid solvers, the setup time consists of the construction of the multigrid hierarchy, as well as the assembly of the numerical stencils on the coarse grids. In the convergence plots, the setup time is explicitly included in the computing time, and is indicated by a constant residual before residual reduction starts. The setup time must be spent in every simulation frame when the fluid domain and thus the system matrix changes, e.g., due to moving obstacles or a free liquid surface. For smoke simulation with fixed obstacles the system matrix does not change, i.e., in this case the setup time must be spent only in the very first simulation frame.

When considering the convergence plots in Figure 7, it can be observed that the MGPCG$^+$ solver achieves by far the best convergence in all simulation scenarios, and for all PPE discretizations and grid resolutions. It is particularly remarkable that its performance is almost independent of the complexity of the simulation scenario, works equally well for all three discretizations, and scales linearly in the number of pressure DOFs. These properties are also provided by the BoomerAMG solver, which achieves a convergence rate that is equal to that of the MGPCG$^+$ solver. However, the setup time of the algebraic multigrid solver is significantly longer than that of the geometric multigrid solver, and exceeds significantly the computing time required for performing the multigrid cycles.

Except those two solvers, the convergence behavior of all other solvers depends on the complexity of the simulation scenario. It can be observed that the 'sphere' scenario and the 'city' scenario have a similar complexity from these solvers' perspective, whereas the maze scenario has a significantly higher complexity. The multigrid solver by McAdams et al. shows only slightly slower convergence than the MGPCG$^+$ solver in the first two scenarios, yet the convergence rate is reduced significantly in the maze scenario. A similar behavior can be observed for the multigrid solver by Chentanez and Müller, which in the first two scenarios shows only a slightly lower convergence rate than the MGPCG$^+$ solver, and at the beginning is even faster due to a shorter setup time. However, in the third scenario this solver converges only very slowly.

While in the first two scenarios the standard hierarchy (without vertex duplication) and the topology-aware hierarchy (with vertex duplication) are virtually identical, in the 'maze' scenario the standard hierarchy merges locally separated fluid domains on the coarser scales, whereas the topology-aware hierarchy keeps them separated. For the multigrid solvers by McAdams et al. and Chentanez and Müller, which both use a standard hierarchy, this leads to a deterioration of the convergence rate. It is further important to note that the solver by McAdams et al. is only applicable to the grid-aligned boundary discretization of the PPE, and the solver by Chentanez and Müller only to the curved boundary discretization by Batty et al., whereas the proposed MGPCG$^+$ solver can be used in combination with any of the considered discretizations. For the ICPCG and MICPCG solvers, the computing time roughly doubles when going from the 'sphere' and 'city' scenarios to the 'maze' scenario.

When the grid resolution is doubled in each dimension, the solvers' behavior remains more or less the same. Considering the scalability of the multigrid solvers, the measurements confirm that the number of multigrid cycles required to reach a certain residual tolerance is independent of the grid resolution, resulting in a linear run-time in the number of unknowns. In contrast, for the ICPCG and MICPCG

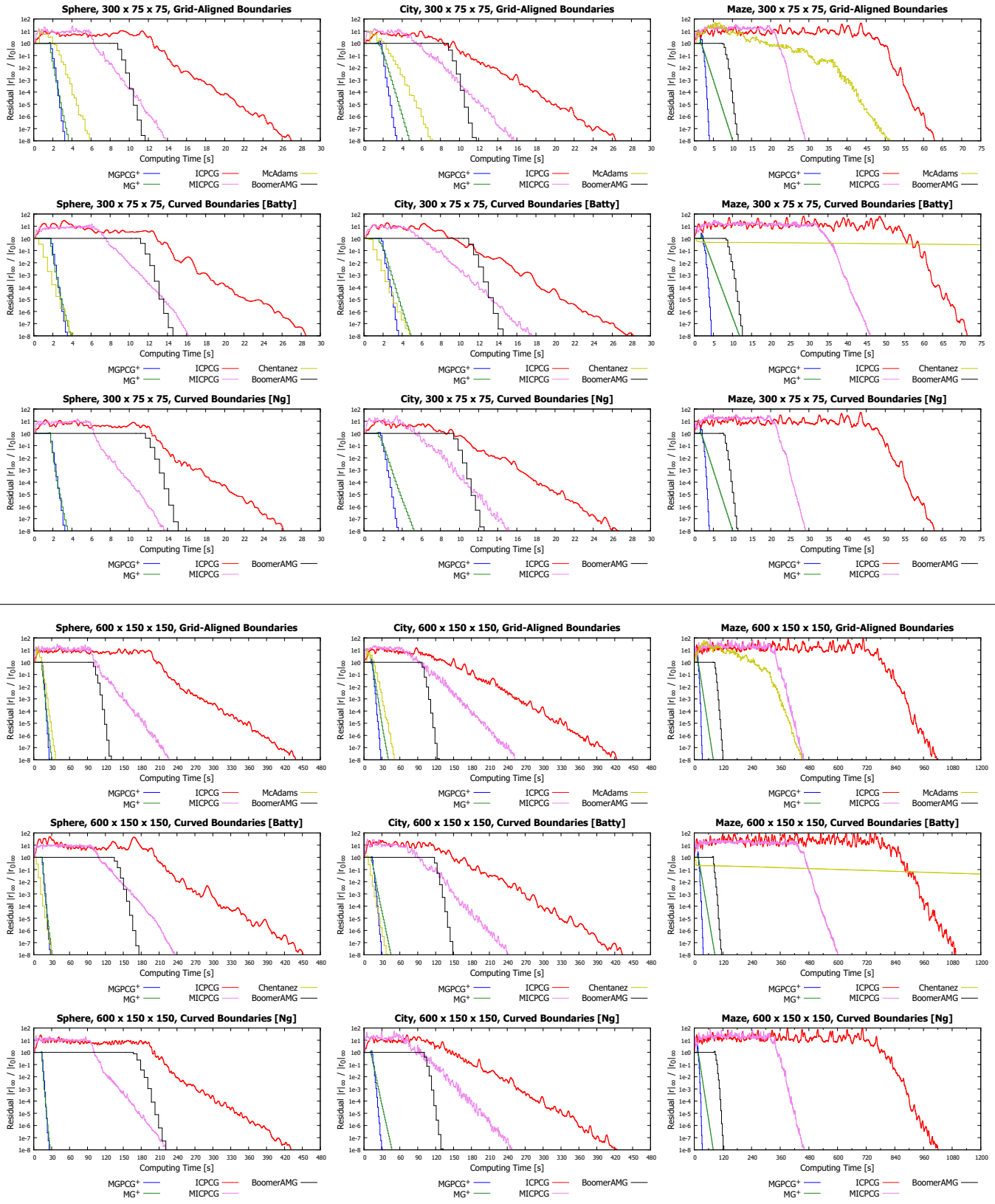

Fig. 7. Convergence behavior of different multigrid solvers for the scenarios shown in Figure 6 (columns) and three different discretizations of the PPE (rows), using two different grid resolutions (upper and lower block of 3×3 diagrams). The analysis includes the topology-aware multigrid solver as a direct solver (MG$^+$) or as a CG preconditioner (MGPCG$^+$), a CG solver with incomplete Cholesky preconditioner (ICPCG) or modified incomplete Cholesky preconditioner (MICPCG), the multigrid solvers by McAdams et al. and Chentanez and Müller, as well as the BoomerAMG solver.

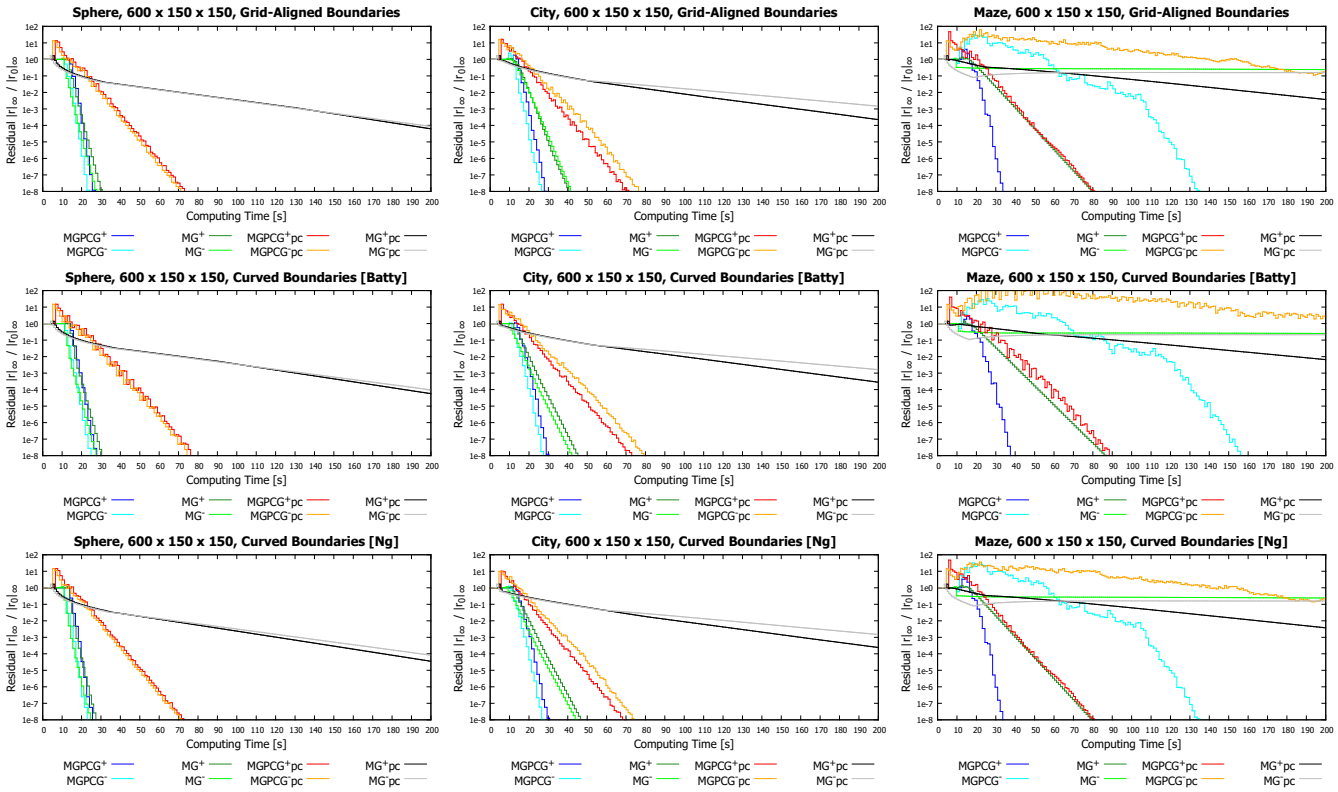

Fig. 8. Convergence behavior of Galerkin-based multigrid solvers: Usage as a direct solver (MG) vs. as a preconditioner for the CG method (MGPCG), using a topology-aware ($^+$) vs. a standard hierarchy ($^-$), and using trilinear vs. piecewise constant interpolation operators (suffix pc) as illustrated in Figure 2, cases (c) and (a).

solvers the required number of solver cycles increases with increasing grid resolution.

Figure 8 analyzes in detail the convergence rates of different variants of Galerkin-based multigrid solvers. Note that the combination of the MG$^-$pc and MGPCG$^-$pc solvers (using piecewise constant interpolation operators) with the curved boundary discretization by Ng et al. is mathematically equivalent to the approach by Weber et al. [23]. (In contrast to the rather high values for the number of pre- and post-smoothing steps that were used in this work, we have not observed an improvement of convergence rate with respect to computing time when using more than one pre- and post-smoothing step.) Again it can be observed that in the 'sphere' and the 'city' scenarios, the standard and the topology-aware hierarchy achieve the same performance, whereas in the 'maze' scenario significantly higher convergence rates are achieved by the topology-aware hierarchy. Note that the use of the topology-aware hierarchy instead of the standard hierarchy only slightly increases the solver setup time. This is due to the fact that the setup time is dominated by the computation of the numerical stencils on the coarse grids (which occurs for both hierarchies), rather than by the construction of the grid hierarchy. Using the multigrid cycle as a preconditioner for CG generally increases the convergence rate of the method. Considering the interpolation operators, the experiments show that trilinear interpolation yields significantly better convergence rates than piecewise constant interpolation.

5 Conclusion, Discussion and Future Work

We have presented a convergence analysis of geometric multigrid schemes for solving the pressure Poisson equation in fluid simulation. We have shown that in complicated domains existing multigrid solvers converge slowly, yet a topology-aware multigrid solver achieves high convergence rates and performance gains thereof. Key to this approach is the accurate representation of the domain topology and geometry throughout the multigrid hierarchy. In combination with the blackbox-style construction of the coarse grid operators from the stencils of the used discretization, alternative discretizations of the pressure Poisson equation can be incorporated flexibly and solved at high convergence rates. For moderately complicated scenarios all compared multigrid solvers converge equally well, and the additional memory requirements and coding effort imposed by the topology-aware approach cannot be amortized.

Since the topology-aware solver introduces a memory overhead, it might not be well suited for architectures with low memory capacities such as mobile devices, and it may reduce the maximum resolution that can be simulated compared to alternative approaches. Due to vertex duplication, it also cannot directly be incorporated into available libraries providing hierarchical data structures comprising regular grids per level, such as OpenVDB [27].

Considering the future parallelization of the solvers, all multigrid solvers included in this study provide virtually the same amount of parallelism and can take advantage of many advancements in solver parallelization.

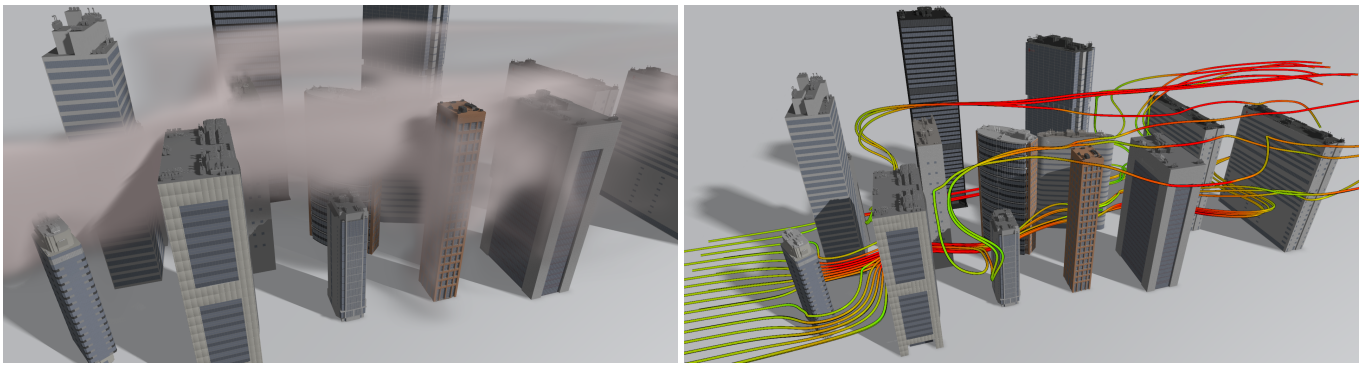

Fig. 9. Simulation of smoke in a city. Slight variants of this and the other two scenarios shown in Figure 1 are employed in our convergence study.

Multi-color Gauss-Seidel relaxation, residual computation, restriction, interpolation (using the existing 8-color multi-coloring, when realized as scattering), and the matrix-vector and dot products (requiring a reduce operation) in the surrounding CG solver and the coarsest-level CG solver can be parallelized in a straightforward way: All operations can be performed independently at each vertex, and all data structures are pre-allocated. However, the parallelization of the solver setup, including the construction of the topology-aware hierarchy and the computation of the stencils, is more complicated. These steps require allocating data structures that store an irregular amount of data per vertex, necessitating the use of appropriate parallelization primitives such as stream compaction.

Considering the multigrid cycles, it can be expected that the memory overhead of the topology-aware approach *on the coarse levels* has a negative effect on multi-core scalability due to memory bandwidth limitations between CPU and main memory. However, since the major workload is on the finest level ($7/8$ of the total work) where no memory overhead occurs, we expect that the negative impact on scalability is rather small.

In the future we will furthermore investigate strategies to reduce the memory requirements imposed by the topology-aware approach, such as using an index-based representation only in regions where vertex duplication occurs (and a regular grid representation in all other regions), and/or storing only the irregular coarse grid stencils.

APPENDIX A
IMPLEMENTATION DETAILS

An efficient construction of the topology-aware hierarchy might seem difficult at first glance, but is possible by using appropriate algorithms and data structures. Here we describe our implementation as it has been used in the convergence experiments. We focus on the construction of the multigrid hierarchy and the assembly of the numerical stencils on the coarse grids by means of Galerkin-based coarsening.

A.1 Hierarchy Construction

We describe the algorithm to efficiently construct the topology-aware hierarchy without explicitly determining the set of edges E^ℓ (see Section 3.2).

The main idea is to determine the connected components of a graph by constructing an auxiliary directed graph with the same vertex set, but with each connected component of the original graph being replaced by a spanning tree of this connected component. This spanning forest can be represented efficiently (for each vertex, only a pointer to its parent vertex is required, with each root vertex pointing to itself), and each connected component is uniquely identified by the root of its corresponding spanning tree. For each vertex the respective connected component can be determined by following the parent pointers to the root of the respective spanning tree.

We generate the individual levels successively from fine to coarse. Specifically, in pass $\ell = 1, \dots, L$, we create the vertices V^ℓ, the sets of restriction vertices $R_v, v \in V^\ell$, and the sets of interpolation vertices $I_v, v \in V^{\ell-1}$. The sets of interpolation vertices are disposed after the assembly of the coarse grid stencils.

The creation of a level is further subdivided into 8 passes, one for each $\vec{x}_0 \in \{0,1\}^3$, such that in pass $i = 1, \dots, 8$ we create all coarse grid vertices on the lattice $L_i = 2^\ell(\vec{x}_0 + 2\mathbb{Z}^3)$, i.e., we skip every second coarse grid vertex position in each dimension. Note that this directly yields the subsets of vertices that can be processed in parallel during 8-color Gauss-Seidel relaxation. Conceptually, we proceed as follows: Let $\vec{x}_v$ denote the position of a vertex v, and $V_i = \{v \in V^{\ell-1}; \vec{x}_v \in L_i + 2^{\ell-1}\{-1,0,1\}^3\}$ the subset of vertices on the previous finer level that lie within 3^3-blocks around the considered potential coarse grid vertex positions L_i. Note that by skipping every second coarse grid vertex position in each dimension, the considered 3^3-blocks are separated by one layer of lattice vertices in each dimension. Thus, the connected components of $G^{\ell-1}[V_i]$ (denoting the restriction of $G^{\ell-1}$ to V_i) correspond to coarse grid vertices on the lattice L_i.

To actually determine the connected components, we build an auxiliary directed graph G_i that is a spanning forest of $G^{\ell-1}[V_i]$, as described above.

Identifying the vertices by integer numbers, i.e., $V^{\ell-1} = \{1, \dots, N^{\ell-1}\}$, the forest is represented by an array $parent[1, \dots, N^{\ell-1}]$, such that $parent[v]$ specifies the parent of vertex v. Note that a vertex v is a root of a tree, if $parent[v] = v$. The array is initialized with $parent[v] \leftarrow v$ for all $v \in V^{\ell-1}$, i.e., initially, each vertex is a separated tree. We then successively merge trees in the forest.

Algorithm 2

function ROOT(v)
  $current \leftarrow v$ ▷ Walk from v towards root
  $next \leftarrow parent[current]$
  while $next \neq current$ **do** ▷ Stop when root reached
    $current \leftarrow next$; $next \leftarrow parent[current]$
  end while
  return $current$
end function

procedure MERGETREES(v_1, v_2)
  $root_1 \leftarrow$ ROOT(v_1) ▷ Determine root of v_1
  $root_2 \leftarrow$ ROOT(v_2) ▷ Determine root of v_2
  $parent[root_2] \leftarrow root_1$ ▷ Make $root_2$ child of $root_1$
end procedure

For $\ell = 1$, we iterate over the simulation cells and consider each cell's face-adjacent neighbors. If both the cell and a neighbor carry a pressure DOF, these cells correspond to connected vertices on the finest level, i.e., to an edge $\{v_1, v_2\} \in E^0$. If $v_1, v_2 \in V_i$, we merge the trees in G_i containing v_1 and v_2, respectively. This is achieved by executing MERGETREES(v_1, v_2) in Algorithm 2.

For $\ell \geq 2$, recall that $\{v_1, v_2\} \in E^{\ell-1} \Leftrightarrow \exists v_f \in V^{\ell-2} : v_1, v_2 \in I_{v_f}$ (see Section 3.2). We thus iterate over $V^{\ell-2}$, and for each $v_f \in V^{\ell-2}$, we merge all trees in G_i containing one of the vertices $I_{v_f} \cap V_i$ into a single tree. After picking an arbitrary vertex $v' \in I_{v_f} \cap V_i$, this is achieved by executing MERGETREES(v', v) for each $v \in (I_{v_f} \cap V_i) \setminus \{v'\}$.

The desired information can now be readily determined from the spanning forest. Since we identify the connected components of $G^{\ell-1}[V_i]$ by the root vertices of the corresponding spanning trees, we create a coarse grid vertex $v_c \in V^\ell$ for each $v \in V^{\ell-1} \cap V_i$ for which $parent[v] = v$. The set of restriction vertices associated with v_c is obtained by $R_{v_c} = \{v_f \in V^{\ell-1}; \text{ROOT}(v_f) = v\}$. After all 8 passes have been performed, the construction of the coarse grid vertices V^ℓ is completed. Then, the sets of interpolation vertices $I_v, v \in V^{\ell-1}$ are derived from the sets of restriction vertices $R_v, v \in V^\ell$ (see Section 3.2).

A.2 Assembly of Coarse Grid Stencils

After constructing the hierarchy, we assemble the numerical stencils on the coarse grids, starting with level 1 and proceeding with higher level numbers. As input we take the multigrid hierarchy, as well as the stencils on level 0 given by the respective discretization of the PPE. The stencil at a vertex v is denoted by $\sum_{v' \in S_v} A^v_{v'} x_{v'}$, where S_v denotes the set of stencil vertices, $A^v_{v'}$ the stencil coefficients, and $x_{v'}$ the unknowns at the stencil vertices.

The construction of the stencil at a coarse grid vertex $v_c \in V^\ell$ on level $\ell \geq 1$ is performed in two steps. Firstly, we compute a weighted sum of the stencils at the restriction vertices of v_c on the previous finer level:

$$\sum_{v \in R_{v_c}} \left(w_{v \leftrightarrow v_c} \sum_{v' \in S_v} A^v_{v'} x_{v'} \right) = \sum_{v \in R_{v_c}} \sum_{v' \in S_v} (w_{v \leftrightarrow v_c} A^v_{v'}) x_{v'},$$

where $w_{v \leftrightarrow v_c}$ denotes the trilinear interpolation and restriction weight between vertex v and vertex v_c. By accumulating the contributions at the vertices $V = \bigcup_{v \in R_{v_c}} S_v$, we obtain an intermediate stencil

$$\sum_{v' \in V} B_{v'} x_{v'},$$

where the unknowns still reside on the previous finer level. Secondly, we replace these unknowns by means of interpolation from unknowns at coarse grid vertices according to $x_{v'} = \sum_{v'_c \in I_{v'}} w_{v' \leftrightarrow v'_c} x_{v'_c}$, leading to

$$\sum_{v' \in V} B_{v'} \left(\sum_{v'_c \in I_{v'}} w_{v' \leftrightarrow v'_c} x_{v'_c} \right) = \sum_{v' \in V} \sum_{v'_c \in I_{v'}} (w_{v' \leftrightarrow v'_c} B_{v'}) x_{v'_c}.$$

Again, by accumulating the contributions at the vertices $S_{v_c} = \bigcup_{v' \in V} I_{v'}$, we finally obtain the stencil at the coarse grid vertex v_c

$$\sum_{v'_c \in S_{v_c}} A^{v_c}_{v'_c} x_{v'_c}.$$

To enable the efficient accumulation of the coefficients, we employ an array that stores for each vertex in V (first step) or S_{v_c} (second step) a pair consisting of the vertex index and the vertex's accumulated coefficient, as well as a look-up table that maps the vertices $V^{\ell-1}$ (first step) or V^ℓ (second step) to array indices. Since the vertex sets V and S_{v_c} are not known a priori, we start with an array of length 0, and extend the array during the accumulation process. The look-up table initially maps each vertex to -1 (indicating that an array index has not (yet) been assigned to a vertex), and is updated accordingly.

ACKNOWLEDGMENTS

The work was partly funded by the European Union under the ERC Advanced Grant 291372: Safer-Vis—Uncertainty Visualization for Reliable Data Discovery.

REFERENCES

[1] R. Fedkiw, J. Stam, and H. W. Jensen, "Visual simulation of smoke," in *Proc. ACM SIGGRAPH*, 2001, pp. 15–22.
[2] J. Molemaker, J. M. Cohen, S. Patel, and J. Noh, "Low viscosity flow simulations for animation," in *Proc. Eurographics/ACM SIGGRAPH Symposium on Computer Animation*, 2008, pp. 9–18.
[3] A. McAdams, E. Sifakis, and J. Teran, "A parallel multigrid Poisson solver for fluids simulation on large grids," in *Proc. Eurographics/ACM SIGGRAPH Symposium on Computer Animation*, 2010, pp. 65–73.
[4] N. Chentanez and M. Müller, "A multigrid fluid pressure solver handling separating solid boundary conditions," in *Proc. Eurographics/ACM SIGGRAPH Symposium on Computer Animation*, 2011, pp. 83–90.
[5] M. J. Aftosmis, M. J. Berger, and G. Adomavicius, "A parallel multilevel method for adaptively refined Cartesian grids with embedded boundaries, AIAA 2000-0808," in *Proc. 38th AIAA Aerospace Sciences Meeting and Exhibit*, 2000.
[6] F. Ferstl, R. Westermann, and C. Dick, "Large-scale liquid simulation on adaptive hexahedral grids," *IEEE TVCG*, vol. 20, no. 10, pp. 1405–1417, 2014.
[7] N. Foster and D. Metaxas, "Realistic animation of liquids," *Graphical Models and Image Processing*, vol. 58, no. 5, pp. 471–483, 1996.
[8] C. Batty, F. Bertails, and R. Bridson, "A fast variational framework for accurate solid-fluid coupling," *ACM TOG*, vol. 26, no. 3, pp. 100:1–100:7, 2007.
[9] Y. T. Ng, C. Min, and F. Gibou, "An efficient fluid-solid coupling algorithm for single-phase flows," *Journal of Computational Physics*, vol. 228, no. 23, pp. 8807–8829, 2009.

[10] R. Bridson, *Fluid Simulation for Computer Graphics*. A K Peters/CRC Press, 2008.
[11] J. Stam, "Stable fluids," in *Proc. ACM SIGGRAPH*, 1999, pp. 121–128.
[12] Y. Zhu and R. Bridson, "Animating sand as a fluid," *ACM TOG*, vol. 24, no. 3, pp. 965–972, 2005.
[13] F. H. Harlow and J. E. Welch, "Numerical calculation of time-dependent viscous incompressible flow of fluid with free surface," *The Physics of Fluids*, vol. 8, no. 12, pp. 2182–2189, 1965.
[14] F. Gibou, R. P. Fedkiw, L.-T. Cheng, and M. Kang, "A second-order-accurate symmetric discretization of the Poisson equation on irregular domains," *Journal of Computational Physics*, vol. 176, no. 1, pp. 205–227, 2002.
[15] R. Setaluri, M. Aanjaneya, S. Bauer, and E. Sifakis, "SPGrid: A Sparse Paged Grid structure applied to adaptive smoke simulation," *ACM TOG*, vol. 33, no. 6, pp. 205:1–205:12, 2014.
[16] A. Brandt and O. Livne, *Multigrid Techniques: 1984 Guide with Applications to Fluid Dynamics, Revised Edition*. SIAM, 2011.
[17] W. L. Briggs, V. E. Henson, and S. F. McCormick, *A Multigrid Tutorial*, 2nd ed. SIAM, 2000.
[18] U. Trottenberg, C. W. Oosterlee, and A. Schüller, *Multigrid*. Elsevier Academic Press, 2001.
[19] N. Goodnight, C. Woolley, G. Lewin, D. Luebke, and G. Humphreys, "A multigrid solver for boundary value problems using programmable graphics hardware," in *Proc. ACM SIGGRAPH/EUROGRAPHICS Conference on Graphics Hardware*, 2003, pp. 102–111.
[20] N. Chentanez and M. Müller, "Real-time Eulerian water simulation using a restricted tall cell grid," *ACM TOG*, vol. 30, no. 4, pp. 82:1–82:10, 2011.
[21] J. L. Hellrung Jr., L. Wang, E. Sifakis, and J. M. Teran, "A second order virtual node method for elliptic problems with interfaces and irregular domains in three dimensions," *Journal of Computational Physics*, vol. 231, no. 4, pp. 2015–2048, 2012.
[22] H. Johansen and P. Colella, "A Cartesian grid embedded boundary method for Poisson's equation on irregular domains," *Journal of Computational Physics*, vol. 147, no. 1, pp. 60–85, 1998.
[23] D. Weber, J. Mueller-Roemer, A. Stork, and D. Fellner, "A cut-cell geometric multigrid poisson solver for fluid simulation," *Computer Graphics Forum*, vol. 34, no. 2, pp. 481–491, 2015.
[24] M. Chuang and M. Kazhdan, "A connectivity-aware multi-level finite-element system for solving Laplace-Beltrami equations," *ArXiv e-prints*, 2015.
[25] A. McAdams, E. Sifakis, and J. Teran, "A parallel multigrid Poisson solver for fluids simulation on large grids, accompanying source code version 1.0," http://pages.cs.wisc.edu/~sifakis/project_pages/MGPCG-1.0.zip, 2010.
[26] Lawrence Livermore National Laboratory, "Hypre 2.10.0b," http://acts.nersc.gov/hypre/, 2015.
[27] K. Museth, "VDB: High-resolution sparse volumes with dynamic topology," *ACM TOG*, vol. 32, no. 3, pp. 27:1–27:22, 2013.

Christian Dick is a PostDoc in the Computer Graphics and Visualization Group at the Technische Universität München, Germany. He received a diploma in computer science in July 2007 and a PhD in January 2012, both from Technische Universität München. His research interests include physics-based simulation of deformable objects and fluids, simulation and visualization in the context of medical applications, as well as the visualization of very large scientific data sets.

Marcus Rogowsky is a graduate student at the Technische Universität München. He received a BSc in computer science in 2014. His research interests include multigrid methods for physics-based simulation of deformable bodies and fluids.

Rüdiger Westermann studied computer science at the Technical University Darmstadt, Germany. He pursued his Doctoral thesis on multiresolution techniques in volume rendering, and he received a PhD in computer science from the University of Dortmund, Germany. In 2002, he was appointed the chair of computer graphics and visualization at the Technical University Munich. His research interests include scalable simulation and visualization algorithms, GPU computing, real-time rendering of large data, and uncertainty visualization.