Journal of Computational Science 72 (2023) 102118

Contents lists available at ScienceDirect

Journal of Computational Science

journal homepage: www.elsevier.com/locate/jocs

An open-source pipeline for solving continuous reaction–diffusion models in image-based geometries of porous media

Justina Stark [a,b,c], Ivo F. Sbalzarini [a,b,c,d,*]

[a] Technische Universität Dresden, Faculty of Computer Science, Nöthnitzer Str. 46, Dresden, 01187, Germany
[b] Max Planck Institute of Molecular Cell Biology and Genetics, Pfotenhauerstr. 108, Dresden, 01307, Germany
[c] Center for Systems Biology Dresden, Pfotenhauerstr. 108, Dresden, 01307, Germany
[d] Cluster of Excellence Physics of Life, TU Dresden, Dresden, 01187, Germany

ARTICLE INFO

Dataset link: https://git.mpi-cbg.de/mosaic/reactiondiffusion_imagebased_porousmedia.git

MSC:
65-04
35K57
65M06
65D40

Keywords:
Reaction–diffusion
Complex geometry
Tortuosity
Sparse block grids
GPU computing
Level-set method

ABSTRACT

We present a versatile open-source pipeline for simulating inhomogeneous reaction–diffusion processes in highly resolved, image-based geometries of porous media with reactive boundaries. Resolving realistic pore-scale geometries in numerical models is challenging and computationally demanding, as the scale differences between the sizes of the interstitia and the whole system can lead to prohibitive memory requirements. The present pipeline combines a level-set method with geometry-adapted sparse block grids on GPUs to efficiently simulate reaction–diffusion processes in image-based geometries. We showcase the method by applying it to fertilizer diffusion in soil, heat transfer in porous ceramics, and determining effective diffusion coefficients and tortuosity. The present approach enables solving reaction–diffusion partial differential equations in real-world geometries applicable to porous media across fields such as engineering, environmental science, and biology.

1. Introduction

Reaction–diffusion partial differential equations (PDEs) describe transport and conversion processes in porous media across science and engineering [1–4]. Examples include soil contamination in geology [5], drug formulations using amorphous solid dispersion in pharmacy [6], transport of reactants in packed bed catalysts [1,7], and biological phenomena such as tissue patterning and morphogenesis [2,8–13]. The porous media in which these processes occur often have intricate, irregular geometries.

These intricate geometries make constructing accurate numerical models of transport processes in porous media challenging. To simplify the problem, transport processes are often modeled by homogenizing the geometry [14–18] and using coarse-grained geometric factors like tortuosity [19] and porosity. However, determining such factors is a non-trivial task, and simplified homogenized models often lack in accurately predicting reactive transport processes when geometric heterogeneity is lost [20]. To numerically model transport processes in porous media more accurately, or to computationally determine coarse-grained observables, the real-world geometry of the pores has to be factored in.

Realistically modeling pores in a simulation, however, is challenging. One difficulty is that the surface of real-world irregular porous media geometries cannot be described by a global parametrization. Instead, an algorithmic surface representation has to be constructed that can be either explicit, as, e.g., surface triangulation [21], or implicit, as, e.g., phase fields [22,23] or level sets [24].

Another challenge in resolving real-world pore geometries in a computer simulation is the difference in the length scales involved. While the interstitial space is often on the order of micrometers, the overall system size can be several orders of magnitude larger. A key challenge with such multiscale simulations is that they usually demand a lot of computer memory, limiting their feasibility and efficiency.

This memory requirement can be reduced in porous media simulations by taking advantage of the fact that only a fraction of the 3D space is of interest, and using a dense discretization is wasteful.

* Corresponding author at: Technische Universität Dresden, Faculty of Computer Science, Nöthnitzer Str. 46, Dresden, 01187, Germany.
E-mail address: ivo.sbalzarini@tu-dresden.de (I.F. Sbalzarini).

https://doi.org/10.1016/j.jocs.2023.102118
Received 5 April 2023; Received in revised form 25 July 2023; Accepted 27 July 2023
Available online 7 August 2023
1877-7503/© 2023 The Author(s). Published by Elsevier B.V. This is an open access article under the CC BY-NC-ND license (http://creativecommons.org/licenses/by-nc-nd/4.0/).

Several methods allow for the placement of discretization points only in regions of interest. A famous example is particle methods, where the discretization points are not bound to any grid structure [25–27]. The downside of lacking a grid structure, however, is that neighborhood information is not readily available, requiring additional data structures like cell lists [28] or Verlet lists [29]. Moreover, approximation of spatial derivatives requires computationally expensive mesh-free methods like DC-PSE [30]. Hence, particle methods reduce the memory requirement thanks to their geometrical adaptivity but increase the computational cost due to their mesh-free structure.

A data structure that combines the geometrical adaptivity of particle methods with the structured property of a grid is the sparse block grid [31]. Similar to particle methods and in contrast to dense grids, sparse grids enable the allocation of individual points in any subspace of the domain as needed. In contrast to particle methods, however, the points of a sparse grid are structured in Cartesian neighborhoods. Therefore, compared to particles, the overhead for storing positional information is smaller, neighborhood access becomes faster, and differential operators can be approximated using standard grid-based schemes. Furthermore, imposing boundary conditions on a sparse grid is computationally simpler than doing so on particles. Summarizing, owing to their grid-like structure and geometrical adaptivity, sparse block grids represent a suitable candidate for discretizing porous media geometries in a memory- and compute-efficient way.

Once the porous media geometry is numerically represented and discretized efficiently, it can be used to simulate reaction–diffusion processes. Reaction–diffusion processes can be described by PDEs, whose evolution in time requires the numerical approximation of differential operators. A wide range of methods for numerically approximating differential operators exists [32], including finite element methods [33], finite volume methods [34], finite difference methods [35], and spectral methods [36]. Spectral methods cannot be adapted to sparse grids without losing their computational advantages. Finite element and finite volume methods can use adaptive, unstructured grids. Generation of an unstructured grid for a given complex geometry, however, has a high computational cost that might be justifiable when simulating fluid flow but is disproportionate when simulating reaction–diffusion processes. Thus, we use finite-difference methods and combine them with sparse grids to solve reaction–diffusion PDEs with high memory and computational efficiency and without the need for generating unstructured meshes.

To further accelerate simulations of transport processes in porous media, we take advantage of GPUs and parallelize our simulation using the Open Framework for Particles and Meshes (OpenFPM) [37]. OpenFPM is well suited to our approach, as it provides a sparse-grid data structure for which tightly coupled scalability to multiple GPUs has been shown [31].

We extend the previous sparse block grid implementation by incorporating an implicit level-set algorithm for representing generic surfaces. This enables simulations in three-dimensional (3D), non-parametric, image-based geometries on distributed GPUs. The resulting open-source pipeline covers the entire workflow for porous media applications, from the segmentation of 3D sample images over the generation of a computational representation to solving the PDEs in the fully-resolved pore-scale geometries and visualizing the results. As inputs, the present pipeline can handle different kinds of microscopy images or micro-computed tomography scans (μCTs). From those, it generates realistic pore-scale geometries using the implemented level-set representation. On the process modeling side, we extend from homogeneous reaction–diffusion models to also include inhomogeneous reaction–diffusion processes, that is, to space-dependent diffusion coefficients and locally varying reaction rates. These can vary in function of the distance to the nearest interface, as that information is available from the level-set description of the geometry. We showcase the resulting image-based simulation pipeline in two real-world examples: inhomogeneous diffusion of fertilizer through soil and heat transfer in reticulate porous ceramics catalysts for renewable energy technology. We also benchmark the computational performance, multi-GPU scalability, and memory requirement of the present pipeline, comparing with dense-grid approaches.

2. Methods

Numerical modeling of reaction–diffusion processes in porous media requires a mathematical description of the process, a numerical description of the geometry, and a suitable data structure to discretize space. We therefore first describe the governing equations and methods we use to build predictive models of reaction–diffusion processes in the image-based geometry of porous media. We also review the concepts of tortuosity and effective diffusivity.

Reaction–diffusion processes are modeled by a PDE describing the space–time evolution of a continuous concentration or density field. For isotropic bulk reaction–diffusion processes this PDE is:

$$\frac{\partial u(\boldsymbol{x},t)}{\partial t} = \underbrace{\boldsymbol{\nabla}\cdot\left(D(\boldsymbol{x},t)\boldsymbol{\nabla}u(\boldsymbol{x},t)\right)}_{\text{diffusion}} + \underbrace{f(u(\boldsymbol{x},t),\boldsymbol{x},t)}_{\text{reactions}}, \tag{2.0.1}$$

where $u(\boldsymbol{x},t)$ is a scalar field at time t ($0 < t \leq t_{\max}$) and position $\boldsymbol{x}$ describing an intensive quantity, such as concentration or temperature. The diffusion domain Ω is enclosed by a surface $\partial\Omega$, i.e., $\boldsymbol{x} \in \{\Omega \setminus \partial\Omega\}$. $D(\boldsymbol{x},t)$ is the diffusion coefficient as a function of space and time, and $f(u(\boldsymbol{x},t),\boldsymbol{x},t)$ is the reaction term, which can be a reactive boundary, a linear combination of sources or sinks in the volume, both, or zero for pure diffusion.

Diffusion through porous media is often inhomogeneous. For example, the diffusivity can vary between different phases or when molecules interact with the solid surface, leading to spatially varying coefficients [38]. The reaction term covers sources and sinks that can vary in space and time and depend on the current concentration via the law of mass action.

The solution of this PDE depends on the initial condition $u(\boldsymbol{x},0)$, the boundary conditions, and the geometry of the diffusion domain Ω.

2.1. Tortuosity and effective diffusivity

The geometry of the diffusion space Ω affects the diffusive dispersion as it restricts the path a molecule can take [39,40]. Especially in the case of porous media with small interstitial space, a molecule has to move further on average to reach the same mean square displacement as in free space, because it has to circumvent obstacles [41–46].

The ratio between the average path length $\langle L_\mathrm{d}\rangle$ actually traveled by a diffusing molecule and the straight-line distance L_s is called diffusive tortuosity, τ_d [14,47]. The diffusive tortuosity τ_d can be expressed in terms of the diffusion coefficient in a bulk volume relative to its respective value in a tortuous medium [48], as

$$\tau_\mathrm{d} = \left(\frac{\langle L_\mathrm{d}\rangle}{L_\mathrm{s}}\right)^2 = \frac{D_\mathrm{f}}{D_\mathrm{eff}}, \tag{2.1.1}$$

where D_f is the intrinsic molecular diffusion coefficient and D_eff the effective diffusion coefficient in the tortuous medium [19,46]. The diffusive tortuosity or the effective diffusion coefficient of a material are of fundamental interest in engineering and science, because they can be used in macroscale models to make coarse-grained predictions that help design and conceptualize diffusion in a material [15,19].

Many porous media, however, are heterogeneous, and averaging approaches do not capture local variations in tortuosity and the resulting anomalies [39,49]. Synthetic pore-scale models include tortuosity to generate periodic or stochastic networks of, e.g. cylinders or spheres. However, they do not reach the same level of heterogeneity as real-world geometries due to their artificial symmetry and lack of imperfect pore connectivity [50].

Moreover, pore geometry also affects chemical reactions. In particular, reactions at the interface depend on the surface area, which

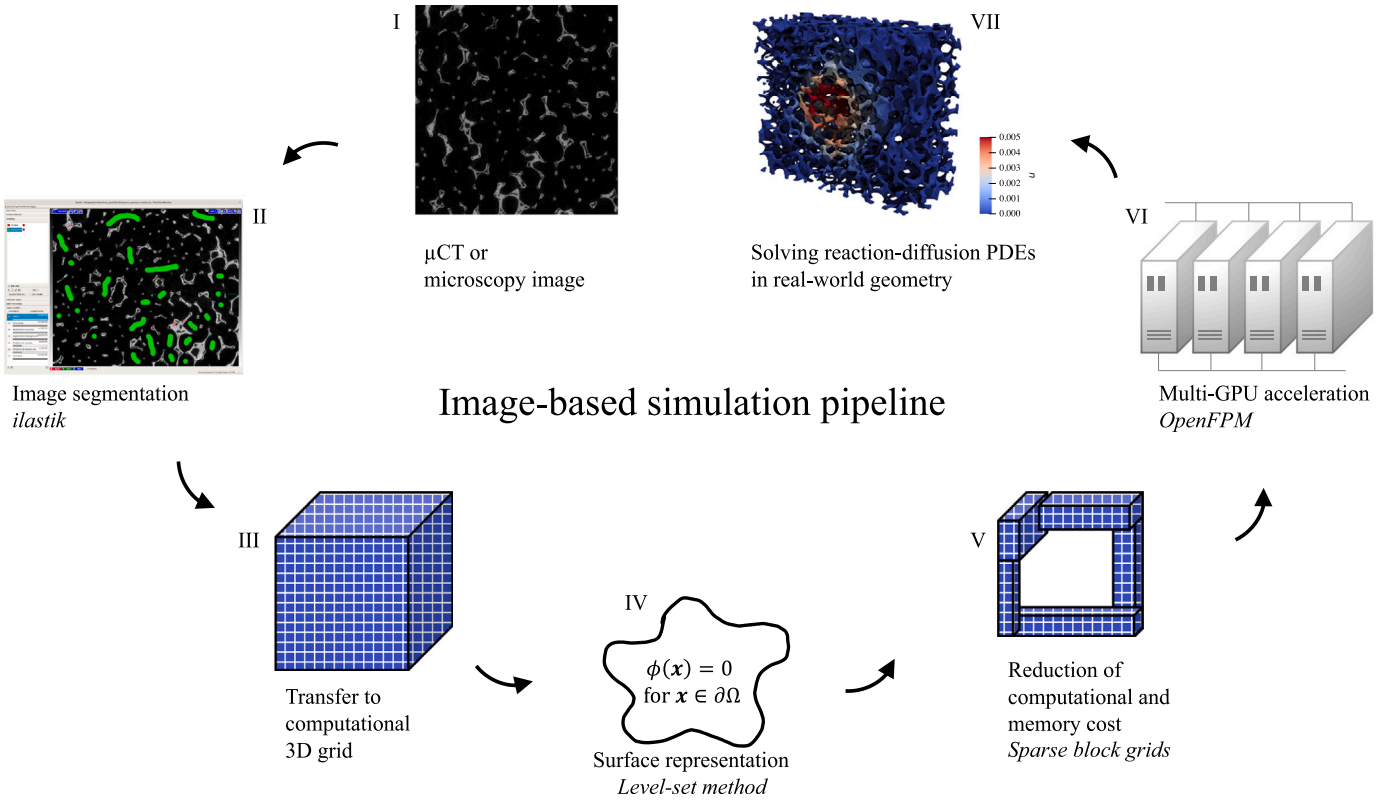


Fig. 1. Overview of the present image-based simulation pipeline. The first step is the segmentation of a 3D μCT or microscopy image (I) using the pixel-wise random-forest classifier (II) from ilastik [53]. The binary segmentation mask is converted to an indicator function from which, on a dense grid (III), the level-set signed distance function (SDF) (IV) is obtained using Sussman redistancing. Based on the SDF, a sparse block grid [31] geometry-adapted discretization is constructed (V). The sparse block grid data structure is then distributed over potentially multiple GPUs using the open-source scalable computing software OpenFPM [37] (VI). Finally, simulation results are visualized using ParaView [54] (VII).

can differ significantly between media of different grain and pore sizes [51]. This is exploited when designing catalysts like packed beds and reticulate porous ceramics, which have a highly tortuous geometry with a large surface area [7,52].

Since characteristics like tortuosity, pore size, pore connectivity, and surface area significantly impact both diffusion and reactions, accurate predictive models of reaction–diffusion processes in porous media require taking into account the real pore-scale geometry of a given sample.

2.2. Level-set method for image-based geometry reconstruction

Real-world geometries for numerical models can be derived from 3D images, such as light-microscopy volumes or micro-computed tomography scans (μCTs). For simulating transport processes in image-derived geometries, an algorithmic representation of the surface is required that forms the boundary for the transported medium.

Several methods to numerically represent surfaces exist. Explicit methods, like surface triangulation, directly discretize the surface by creating a surface grid [21]. However, generating such unstructured grids has a computational cost disproportionate to the cost of the reaction–diffusion simulation [55]. Another disadvantage of these surface grids is that they have to be re-generated when the geometry deforms.

Alternatively, surfaces can be described implicitly by a higher-dimensional function that takes a specific value at the location of the surface. Implicit methods do not require an explicit discretization of the surface and therefore enable the representation of arbitrary geometries embedded in regular grids. Handling deformations also becomes more straightforward with implicit methods, as no surface grid needs to be moved.

The most prominent implicit surface methods are phase-field and level-set methods. Phase-field methods belong to the class of diffuse-interface models with broad applications for simulating interface propagation in two-phase flows or mineral growth [56–58]. Phase-field methods were first introduced by [22] and later extended to volume-conserving phase-fields [23]. Phase-field methods represent the different phases (e.g., solid vs. fluid) of the material by a density function that varies smoothly around the interface. A main limitation of phase-field methods is their high resolution requirement near the interface, where the phase function changes rapidly [59]. Another issue is that the sharpness of the interface is limited by an arbitrarily chosen interface width parameter, leading to geometric inaccuracies and restrictions on the time-step size [60,61].

Level-set methods [24] overcome both problems by using a geometry-description function ϕ that has no additional physical meaning. The only purpose of the level function ϕ is to represent the interface as its zero iso-contour. A convenient choice for ϕ that is sufficiently flat around the surface, allows for continuous differentiation, and carries important geometric information is the signed-distance function SDF:

$$\phi_{\mathrm{SDF}}(\boldsymbol{x}) = \begin{cases} +d & \text{for} \quad \boldsymbol{x} \in \Omega \\ -d & \text{for} \quad \boldsymbol{x} \in S \backslash \Omega \\ 0 & \text{for} \quad \boldsymbol{x} \in \partial\Omega. \end{cases} \tag{2.2.1}$$

Here, d is the Euclidean distance from $\boldsymbol{x}$ to the closest point on the boundary $\partial\Omega$ of a domain Ω embedded in a space S. This implies that $|\nabla\phi_{\mathrm{SDF}}(\boldsymbol{x})| = 1$ for all $\boldsymbol{x}$.

The SDF is constructed for a geometry extracted from a 3D image. The entire workflow, as illustrated in Fig. 1, starts from binary segmentation of the image, classifying pixels into those that belong to

Ω and those that do not. For pixel classification, we use the open-source software ilastik [53], which is based on trainable random-forest classifiers. The classification result is represented on the pixels by a binary indicator function. The SDF can be obtained from such an image-based indicator function by redistancing. A classic redistancing algorithm is the Sussman method [62], which finds the SDF as the steady-state solution of the PDE

$$\frac{\partial \phi(\boldsymbol{x},\alpha)}{\partial \alpha} = \sigma(\phi(\boldsymbol{x},\alpha))(1 - |\nabla\phi(\boldsymbol{x},\alpha)|) \tag{2.2.2}$$

for large artificial "time" α. The smoothed sign function σ proposed by [63] is:

$$\sigma(\phi(\boldsymbol{x},\alpha)) = \frac{\phi(\boldsymbol{x},\alpha)}{\sqrt{\phi(\boldsymbol{x},\alpha)^2 + |\nabla\phi(\boldsymbol{x},\alpha)|^2 h^2}}, \tag{2.2.3}$$

where h is the grid spacing, typically equal to the pixel size in the input image.

Our implementation of Sussman redistancing for multi-CPUs on a dense grid is available as part of the open-source OpenFPM C++ library. At the time of writing, our implementation computes $\nabla\phi(\boldsymbol{x},\alpha)$ in Eq. (2.2.2) using upwind finite differences of order 1, 3 (ENO), or 5 (WENO) [64]. Since the level-set method imposes the limitation that neighboring interfaces must always be further apart than the width of the finite-difference stencil used for Sussman redistancing, we use first-order stencils for all results presented in this paper. They have the smallest width (namely $1h$) and therefore provide the highest spatial resolution for representing the geometry. We ensure a sufficiently large distance between interfaces by filtering geometric features of thickness $\leq 2h$ from the binary indicator function before constructing the level-set.

We also note that the SDF, while defining a mathematically sharp (i.e., not diffuse) interface, is not uniquely defined at sharp corners. During Sussman redistancing, rounding of corners with a radius of curvature $\approx h$ occurs. In image-derived geometries, however, this is not usually a concern, since the images rarely display sharp corners, due to the point-spread function of the imaging system. Moreover, the rounding error of level-set redistancing is of order $O(h)$. For these reasons, the error in the SDF converges with order one, as verified in Fig. A.1.

In our approach, we use the SDF not only to describe the interface geometry, but also to define consistent inhomogeneous diffusion coefficients, restrict reactions to the surface, and imposes no-flux Neumann boundary conditions for the diffusion. It can also be extended to describe deforming geometries, which we do, however, not consider here.

To obtain smoothly varying diffusion coefficients that depend on the distance to the surface, we make $D(\boldsymbol{x})$ a functional of $\phi_{\mathrm{SDF}}(\boldsymbol{x})$, i.e.,

$$D_{\mathrm{smooth}}(\phi_{\mathrm{SDF}}(\boldsymbol{x})) = D_{\mathrm{min}} + \frac{D_{\mathrm{max}}}{1 + \exp[-(\gamma_1 + \gamma_2\phi_{\mathrm{SDF}}(\boldsymbol{x}))]}, \tag{2.2.4}$$

in the case where the minimum and maximum diffusion coefficients (D_{min}, D_{max}) are known. The parameters γ_1 and γ_2 control the distance of the transition region from $\partial\Omega$ and the smoothing length, respectively. This transition region ensures that the diffusion coefficient can be continuously differentiated by finite difference [65].

For reactive transport processes that are restricted to one phase, it is no longer necessary to discretize the entire 3D domain S once an implicit description of the diffusion space Ω has been obtained as a SDF. For example, we do not need grid nodes in the solid phase when simulating diffusion in the fluid phase. To ensure computational efficiency, we therefore aim to use a geometry-adaptive discretization method that does not waste memory and computation where they are not needed.

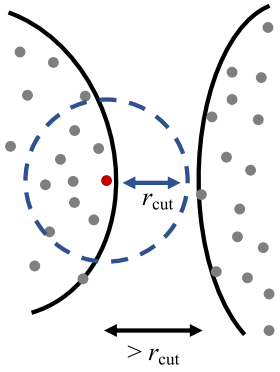


Fig. 2. Illustration of the resolution requirement of particle methods in porous media geometries. Since operator kernels of particles in one interface must not overlap with particles on neighboring interfaces, a sufficiently dense particle distribution is required. This can be achieved by upsampling.

2.3. Discretization and numerical methods

For multi-scale simulations in porous media, geometry-adaptive discretization data structures can significantly reduce memory requirements and computational costs. In particular, for processes that are restricted to one phase, using a dense grid would be wasteful. As mentioned in Section 1, particle methods are well suited for discretizing complex geometries, but require computationally expensive data structures and methods to organize neighborhood information and approximate differential operators [27,30].

The fact that operators are usually approximated over a neighborhood of particles within a specific kernel radius adds a further challenge for applications in porous media, where the interface surface is not singly connected. In particular, particle interaction neighborhoods must not overlap between different parts of the interface, as illustrated in Fig. 2. This condition can only be guaranteed when pore-scale structures are simplified or there is more than one particle per pixel. In particular, for image-based geometries, the smallest distance between two interfaces is one pixel. Thus, there have to be sufficiently many particles per pixel to ensure that kernels will never overlap with particles from a neighboring interface. This limits the use of mesh-free particle methods for simulations in fully resolved porous media geometries.

A data structure that combines the advantages of a structured grid with the efficiency of geometry-adapted methods is the sparse block grid [31]. It allows selectively allocating grid points in chunks to the phase of interest, thus considerably reducing the memory and computational overhead for discretizing sparse spaces.

Yet, sparse grids preserve the advantage of a grid data structure of implying positional information and, thus, easily finding neighboring points. Consequently, differential operators can be approximated using standard finite-difference schemes, which is faster than operator approximation on particles. Furthermore, imposing boundary conditions is simpler in finite-difference methods than in mesh-free particle methods. In particular, no mirror points in the other phase are required to impose no-flux Neumann boundary conditions. Instead, the no-flux condition is imposed within the finite-difference stencil. This is possible because the index of each neighbor within the stencil is known due to the grid structure, and the level-set function indicates on which side of the boundary a neighbor point lies. In the GPU kernel, the functor evaluating the finite difference is applied to all grid nodes. Within the functor, a stencil is constructed, and it is checked whether a neighbor is outside Ω. If it is outside, its value is set equal to that of the center point so that the gradient between the center and that outside neighbor is zero.

For determining whether a point lies outside or inside Ω, we store the level-set function on the sparse grid, along with the intensive

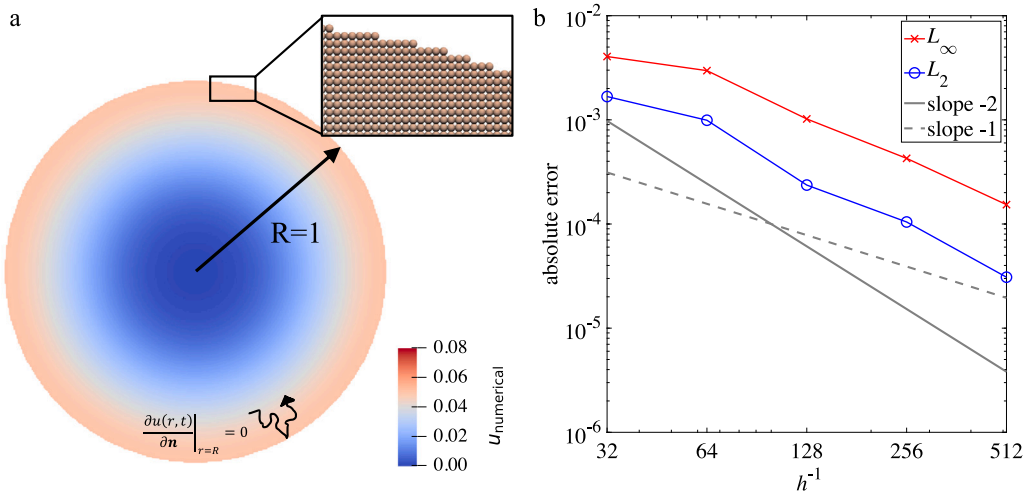


Fig. 3. Verification of the implementation of diffusion on a 2D unit disk with no-flux Neumann boundary conditions. The simulation is run until a final time $t_{\text{final}} = 0.025$ when the system is still far from steady state. The parameters are: $D = 1.0$, $\Delta t = \frac{1}{8}h^2$. **(a)** Visualization of the numerically computed concentration field at $t_{\text{final}} = 0.025$ on a sparse grid (inset zoom). The sparse grid has been created based on the analytical signed-distance function of the unit disk. Time and diffusion coefficient values are in arbitrary units. **(b)** Convergence of the absolute error of the overall method in L_2 (blue circles) and L_∞ (red crosses) norms. Gray lines show the expected slopes for first- (dashed) and second-order (solid) convergence.

scalar field u, the spatially varying diffusion coefficients, and the local reaction rates. We numerically approximate the solution of Eq. (2.0.1) with the diffusion coefficient given by Eq. (2.2.4) on the sparse grids using a Forward-Time Centered-Space (FTCS) finite-difference scheme of order two in space and order one in time. Although implicit time integrating schemes are more stable and allow for larger time step sizes, we use explicit time stepping as our focus is on GPU scalability and computational efficiency. The matrix inversions and global communication overhead incurred by implicit time stepping would quickly become prohibitive for large and complex-shaped domains. As our method addresses memory-limited problems, we chose the first-order explicit Euler time-stepping scheme because it does not require storing and communicating intermediate stages, as higher-order time-stepping schemes do. No-flux boundary conditions at the interface are imposed without a need for mirror points.

2.4. Parallel implementation using OpenFPM

The present pipeline uses sparse block grids as implemented in the OpenFPM library for scalable scientific computing [31]. OpenFPM is a C++ library for parallelizing particle- and mesh-based simulations [37]. To benefit from the automatic parallelization capabilities of OpenFPM, we implement our simulation in C++ using the OpenFPM grid and sparse-grid data structures. OpenFPM generates these data structures at compile time using template meta-programming and thus hides memory access and communication management from the application, ensuring portability across CPU and GPU architectures [66]. This greatly simplifies the implementation of the present simulation, as demonstrated by the code examples in Appendix B. More code examples and a full API documentation are available on the OpenFPM website,[1] including a documentation of our Sussman redistancing implementation and example functors for solving reaction–diffusion PDEs on sparse grids on the GPU.

In the sparse-grid implementation of OpenFPM, we use a uniform chunk size of 8 grid points along each spatial dimension [31]. These $8 \times 8 \times 8$ grid blocks are stored in an OpenFPM vector containing C++ aggregates. The chunk position in the dense grid is cached with an allocation mask for the grid points that are actually allocated in the sparse grid. Chunks are only stored if they contain at least one allocated grid point. Hence, the fewer the points needed to represent a geometry and the closer these points lie, the fewer chunks are required and, consequently, the larger the memory savings of a sparse grid compared to a dense grid implementation.

In addition to reducing memory requirements, sparse grids also speed up the computation, as the finite-difference stencil only iterates over actually allocated points. The higher the density of points within an allocated chunk, the greater the thread efficiency on the GPU [31].

The sparse grid can be distributed across multiple GPUs by domain decomposition into cuboidal sub-grids. In OpenFPM, the domain decomposition is independent of the chunk boundaries of the sparse grid. Inter-GPU communication is done using CUDA-supported asynchronous MPI over the interconnect of the computer cluster [31].

Since OpenFPM provides backends for both CUDA and HIP [66], our software implementation natively runs on GPUs from different vendors (including both Nvidia and AMD) and is scalable to distributed multi-GPU setups. Our implementation supports both 64-bit floating point (FP64) numbers and 32-bit floating point (FP32) numbers on the GPU. All results in this paper are obtained using FP32 precision, which is supported by all consumer and data-center GPUs and is typically faster [67].

Together, these design choices significantly accelerate fully resolved pore-scale simulations in image-derived geometries of porous media, as we demonstrate below.

3. Results

We verify the correctness of our implementation of the methods explained in Section 2, showcase application examples to real-world reaction–diffusion problems in porous media, quantify the parallel scalability, and compare the GPU performance and memory requirement of a sparse grid versus a dense grid.

3.1. Verification of the implementation

To verify the accuracy of the method and the convergence of the discretization, we consider a benchmark case of homogeneous diffusion of a scalar field u according to Eq. (2.0.1) without reactions, sources, or sinks, i.e., $f(u(\boldsymbol{x},t),\boldsymbol{x},t) \equiv 0$. We make the test case as relevant as possible to the application of porous media, while still possessing an analytical solution against which convergence can be verified. Therefore, the test case includes a curved boundary embedded in a

[1] At the time of writing: http://openfpm.mpi-cbg.de/.

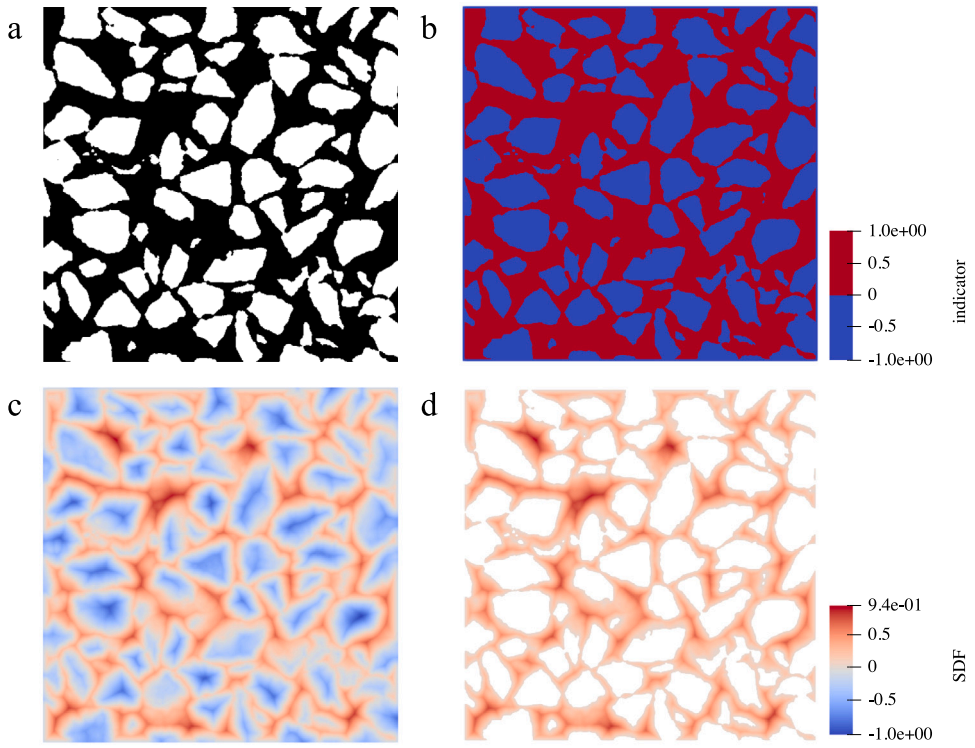


Fig. 4. Steps of the present image-based geometry reconstruction pipeline, visualized for an exemplary slice through the 3D volume of a $CaCO_3$ packed bed [7]. **(a)** Segmented μCT image showing the two phases with 45 μm voxel size [7]. **(b)** Image-based binary indicator function. **(c)** Redistanced level-set signed-distance function (SDF) (Section 2.2). **(d)** Sparse block grid containing only points in the diffusion phase, reducing storage.

Cartesian sparse block grid, represented as a level set, with no-flux Neumann boundary conditions as described in Section 2.3. Specifically, we consider diffusion inside a 2D disk of radius $R = 1$, represented by a SDF and discretized using a sparse grid, as shown in Fig. 3a. The governing equation in polar coordinates ($r = \sqrt{x^2 + y^2}$ and $\theta = \operatorname{atan}(y, x)$) is:

$$\frac{\partial u}{\partial t} = \frac{\partial^2 u}{\partial r^2} + \frac{1}{r}\frac{\partial u}{\partial r} + \frac{1}{r^2}\frac{\partial^2 u}{\partial \theta^2}\,, \tag{3.1.1}$$

with boundary condition

$$\left.\frac{\partial u(r,t)}{\partial \boldsymbol{n}}\right|_{r=R=1} = 0 \tag{3.1.2}$$

and initial condition

$$u(r,0) = \frac{r^3}{3} - \frac{r^4}{4}\,. \tag{3.1.3}$$

The polar coordinates only serve to derive the exact analytical solution. The actual simulation is done in Cartesian coordinates in 2D. The analytical solution for this case, is derived in Appendix A.1 using the method of manufactured solutions [68]. All numerical solutions are compared against this exact solution in both the L_2 and L_∞ norms of the absolute errors across all grid nodes, see Fig. 3b.

Like in the later applications, the time step size in the convergence study is adapted with the grid spacing. In particular, we fulfill the stability condition for diffusion in 2D,

$$\Delta t < \frac{1}{2\max D}\,\frac{1}{\Delta x^{-2} + \Delta y^{-2}}\,, \tag{3.1.4}$$

by setting $\Delta t = \frac{1}{8}h^2$ for isotropic grid spacings $\Delta x = \Delta y = h$ and constant diffusion coefficient $D = 1.0$.

For the smallest grid size of 32×32 nodes, the disk's diameter is covered by approximately 16 grid nodes. The simulation is run until a final time of $t_{\text{final}} = 0.025$ for all resolutions when the system is still far from steady state.

In order to separately quantify the error in solving the diffusion PDE in the domain and the error introduced by initial Sussman redistancing of the level set, we use the exact SDF for testing the PDE solver and, in addition, test the redistancing alone against the exact SDF. The convergence of the SDF error from redistancing alone is first-order, as shown in Fig. A.1.

For the finite-difference scheme used here, we expect first-order convergence in time and second-order convergence in space. The boundary conditions are imposed with first-order accuracy, as the curved boundary is approximated by the piecewise constant edge of the sparse grid, as illustrated in the inset of Fig. 3a. This is, however, not a limitation of the level-set method as such, which could in principle be used to approximate surface normals and curvature for higher-order boundary treatment. In the present workflow, however, we aim to benefit from GPU acceleration and sparse memory data structures, favoring a regular mesh. Moreover, since we use first-order Sussman redistancing, as described in Section 2.2, the error in the boundary conditions is not limiting. We therefore expect a convergence order between one, as dictated by the time stepping and Sussman redistancing, and two, as dictated by the spatial derivatives. This is confirmed in the convergence plot in Fig. 3b, where the empirical order of convergence is about 1.5, verifying the correctness of our implementation.

3.2. Generating real-world geometries from 3D images

We next illustrate the workflow of the present pipeline to go from a 3D image of a porous medium to a usable simulation domain. An overview of the full pipeline is provided in Fig. 1. Fig. 4 shows the workflow for the case of a fluid-filled $CaCO_3$ packed bed. The segmented μCT images (Fig. 4a, voxel size 45 μm) were kindly provided by Prof. Jörg Petrasch (Michigan State University, College of Engineering) [7]. Our pipeline then first converts the binary segmentation to an indicator function $\{-1, 1\}$ (Fig. 4b). This is then used as an input

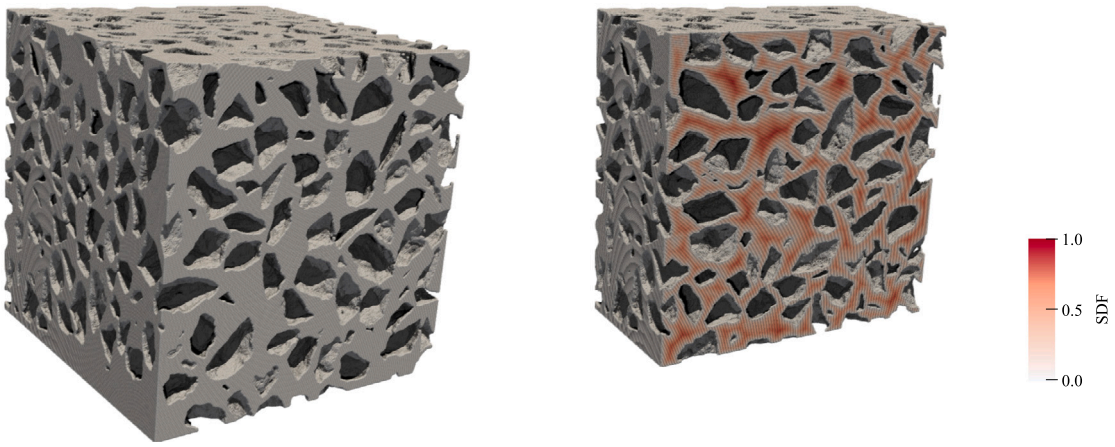


Fig. 5. 3D visualization of the sparse-grid level-set representation of the geometry from Fig. 4d. Left: full block; Right: clipped block with SDF values overlaid in color.

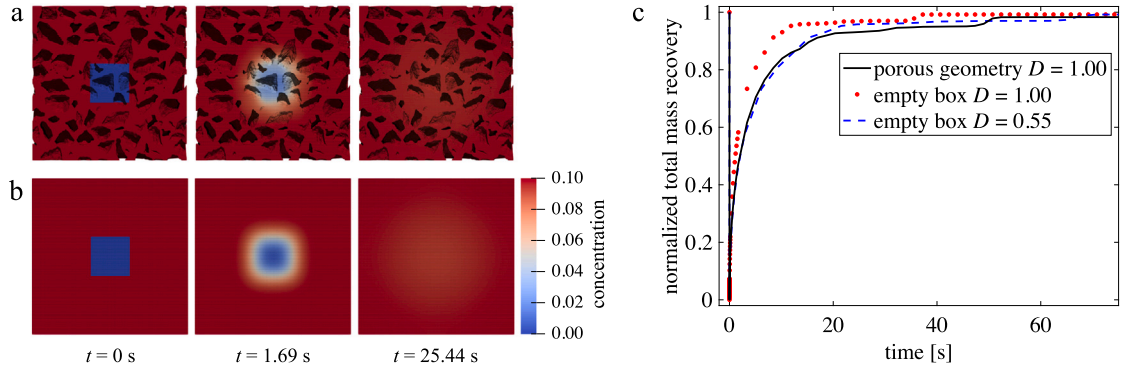


Fig. 6. Simulation of fluorescence recovery after photobleaching (FRAP) in a $CaCO_3$ packed bed for determining the diffusive tortuosity and effective diffusivity using the present pipeline. **(a)** Simulation snapshots of FRAP in the 3D image-based geometry of the porous medium with $D = 1.0\,\mathrm{mm}^2/\mathrm{s}$. **(b)** Simulation snapshot of FRAP in a bulk box of the same domain size with $D = 0.55\,\mathrm{mm}^2/\mathrm{s}$. To obtain the same recovery speed by free diffusion as in the porous structure, the diffusion coefficient had to be reduced by a factor of 1.82, thereby quantifying the diffusive tortuosity. **(c)** FRAP curves showing the normalized mass in the initially bleached box for diffusion in the porous geometry with $D = 1.0\,\mathrm{mm}^2/\mathrm{s}$ and diffusion in the box with $D = 1.0\,\mathrm{mm}^2/\mathrm{s}$ and $D = 0.55\,\mathrm{mm}^2/\mathrm{s}$. The effective diffusion coefficient $D_{\mathrm{eff}} = 0.55\,\mathrm{mm}^2/\mathrm{s}$ was obtained by least-squares fit of the FRAP curve in the box to that in the porous medium.

to Sussman redistancing to generate the level-set SDF representation of the interface (Fig. 4c) as described in Section 2.2. Based on this SDF, the sparse block grid is generated by only inserting points in the diffusion phase, e.g., in regions where $\phi_{\mathrm{SDF}} > 0$. The SDF is then only stored restricted to those points, leading to a sparse geometry-adapted representation (Fig. 4d). Allocation, however, is chunkwise meaning that a group of 8^{dims} (with dimensions dims) points belonging to one chunk is allocated when a chunk contains at least one inserted point. A 3D visualization of the sparse-grid SDF is shown in Fig. 5.

3.3. Calculation of effective diffusion coefficient and tortuosity

The effect of the pore geometry and topology on the diffusion dynamics is of interest in many applications [15,69,70] and is often studied in pore-scale numerical simulations [71–73], as experimental determination is time-consuming and difficult [43]. The effect of spatial hindrance by a porous geometry on the diffusion of a molecule can be captured by an effective diffusion coefficient, which is related to the molecular diffusion coefficient by the diffusive tortuosity (Eq. (2.1.1)). While tortuosity causes an increase in diffusivity in advection-dominated systems, known as Taylor diffusion [74], the effective diffusivity of diffusion-dominated processes in tortuous geometries is smaller than the molecular diffusivity in free space [40,42,50,75,76].

Over the past century, several theoretical and empirical correlations have been postulated that relate diffusive tortuosity τ_{d} to material porosity ψ. Many of them have the form

$$\tau_{\mathrm{d}} = \psi^{-N}, \tag{3.3.1}$$

with different exponents N depending on the type of material [1,77–79]. For instance, it was shown that Eq. (3.3.1) for $N = 2$ fits experimental data of diffusion in low-density alumina and silver pellets, where diffusion is controlled by macropores [1]. High-density pellets, where Knudsen diffusion and micropores dominate, however, could not be described by Eq. (3.3.1) [1]. This demonstrates that the quality of correlations like Eq. (3.3.1) hinges on the similarity between the geometry in question and the geometry used during the fitting process. The $CaCO_3$ packed bed studied here has low density, with only ≈50% of the sample belonging to the solid phase. We, therefore, expect the effective diffusion to be determined by the macropore tortuosity. However, the porosity used in formulas like Eq. (3.3.1) is an average value over (some parts of) a material, which does not account for size limits of the interstitial space and for poor pore connectivity.

Tortuosity–porosity relations have been comprehensively reviewed [14,46] and further refined by numerical simulations [19,50,80]. For granular media, like the $CaCO_3$ packed bed, Huang [50], for example, found in random-walk particle simulations in synthetic geometries that the tortuosity can best be correlated with the porosity ψ by the linear combination [81]

$$\tau_{\mathrm{d}} = \psi + \beta(1 - \psi), \tag{3.3.2}$$

with $\beta = 1.65$.

In order to use the present pipeline to directly measure the tortuosity and effective diffusivity of a given porous geometry, we simulate fluorescence recovery after photobleaching (FRAP). FRAP is an experiment widely used in biology to determine effective diffusion coefficients in cells and tissues [40]. While FRAP experiments are not possible for inorganic solids, the simulation of FRAP in the respective image-derived geometry can still be done with the present pipeline and used to numerically determine the tortuosity. We illustrate this by simulating FRAP in the image-based reconstruction of the fluid phase of the $CaCO_3$ from Section 3.2 with $D = 1.0\,\mathrm{mm}^2/\mathrm{s}$ (Fig. 6a). We then compare the so-simulated diffusion dynamics with diffusion in a bulk 3D box of the same edge length (Fig. 6b). Fitting (least squares) the free-space diffusion dynamics to those simulated in the porous medium, an effective diffusion coefficient of $0.55\,\mu\mathrm{m}^2/\mathrm{s}$ is found in the porous medium (Fig. 6c). According to Eq. (2.1.1), the diffusive tortuosity thus is:

$$\tau_\mathrm{d} = \frac{D}{D_\mathrm{eff}} = \frac{1.0\,\mathrm{mm}^2/\mathrm{s}}{0.55\,\mathrm{mm}^2/\mathrm{s}} = 1.82. \tag{3.3.3}$$

This means that diffusive transport is only about half as efficient in the fluid phase of a $CaCO_3$ than in a free-space bulk box.

This result agrees well with the value from Eq. (3.3.2). In particular, for the sample porosity of $\psi = 0.39$ as determined in Ref. [7], Eq. (3.3.2) predicts $\tau_\mathrm{d} = 1.40$, which is about 23% lower than what we obtain with our FRAP simulation. The difference is in agreement with the finding in Ref. [50] that, due to unconnected pores, the real medium has a larger tortuosity than the synthetic geometries for which Eq. (3.3.2) was fitted. But even the tortuosity obtained by simulations in μCT-based geometries most likely still underestimates the real hindrance to the molecular diffusion, as it ignores interactions with micropores that are not resolved in the μCTs images [38].

3.4. Application example 1: Inhomogeneous diffusion of fertilizer in soil

Understanding how solutes migrate through soil is of significant interest to several fields, including agriculture and environmental science [82–84]. For example, fertilizer and pesticides used in agriculture are desired in specific regions where they protect and nourish the crops but undesired in other areas, where they pollute groundwater [85,86] as a source of drinking water for many communities [87,88]. Therefore, fertilizer application is ideally properly planned. To help farmers manage fertilizer application and prevent a buildup of excess chemicals in the soil, it is often a prerequisite to understanding diffusive transport in porous soil. This is our first application example where we showcase the use of the present computational pipeline.

When transport processes in soil are driven by steep concentration gradients, they are diffusion dominated, i.e., the Péclet number is small, and advection can be neglected. Modeling diffusion in the soil is not trivial, as the diffusive path of a molecule is not only hindered by macropore tortuosity of the porous soil structure, but in addition, the molecules can get trapped in micropores at the solid surfaces and chemically react with molecules in the solid phase [38,72].

We therefore simulate fertilizer diffusion in soil by inhomogeneous diffusion of a concentration field in the fluid phase of a packed bed of $CaCO_3$ particles, using the image-based geometry reconstruction from Fig. 5. In the simulated scenario, a fertilizer dissolved in the liquid enters from one face of the simulation domain and diffuses through the interstitial fluid phase between the solid $CaCO_3$ grains with an inhomogeneous diffusion coefficient that models diffusive hindrance by micropores close to the surface. In particular, $D_\epsilon(\boldsymbol{x}) = D\epsilon(\boldsymbol{x})$ with molecular diffusion coefficient $D = 1.0\,\mathrm{mm}^2/\mathrm{s}$, microporosity $\epsilon = 1$ in the fluid, $\epsilon = 0$ in the solid. This follows the successful approach of locally modulating diffusion by a phase-dependent factor $\epsilon(\boldsymbol{x})$ to model boundary effects [1,89–91]. Different from previous works, however, we introduce a continuously differentiable transition zone between the phases, which depends on the distance $\phi_\mathrm{SDF}(\boldsymbol{x})$ to the fluid–solid interface according to Eq. (2.2.4). We here use smoothing parameters $\gamma_1 = \gamma_2\phi_\mathrm{min}$, $\phi_\mathrm{min} = \min_{\boldsymbol{x}\in\Omega}\phi_\mathrm{SDF}(\boldsymbol{x})$, and $\gamma_2 = 4h$. Also, in the present application, ϵ only represents microporosity at length-scales below the μCT resolution, whereas the macroporosity ψ, resolved by the imaging, is explicitly modeled by the level-set function and the geometry-adapted sparse grid. Therefore, diffusion within macropores is simulated directly, while the hindrance by unresolved micropores at the solid surface is homogenized into an inhomogeneous diffusion constant, following the classic upscaling approach. This fertilizer diffusion model can also readily be applied to contaminant diffusion.

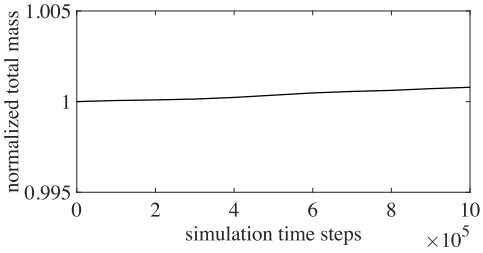


Fig. 7. Mass conservation during inhomogeneous diffusion of fertilizer through the fluid phase of a $CaCO_3$ packed bed. The average increase in normalized total mass is 7.91×10^{-10} per time step, thereby lying within the numerical round-off tolerance of the single-precision (FP32) arithmetics used.

Fig. 8 shows snapshots of the fertilizer diffusion simulation. It can be seen that the diffusion front becomes increasingly fuzzy over time. The increasing roughness of the diffusion front is caused by local variations in spatial hindrance and imperfect pore connectivity, i.e., the tortuosity of the porous geometry. Therefore, volume-averaged diffusion on the macroscale appears effectively anomalous. This anomalous behavior cannot be captured by geometry-averaging models with effective constants, demonstrating the importance of taking into account the pore-scale geometry.

We test the accuracy of our implementation of the no-flux Neumann boundary conditions and the smoothed inhomogeneous diffusion coefficient by monitoring the conservation of mass during the simulation. The results in Fig. 7 show that the total mass is conserved to within the numerical round-off tolerance of the time-stepping scheme, as the simulation is done using single-precision arithmetics on the GPU. The average increase in normalized total mass per time step is 8×10^{-10}, and the total increase after 10^6 time steps is 8×10^{-4}.

We next compare the memory requirements and simulation GPU times of a dense-grid implementation with the sparse-grid version used in this application example. The results for different grid resolutions are given in Table 1. The dense grid is an OpenFPM block grid data structure with all chunks fully occupied. Using the sparse block grid reduces the number of grid points in this particular example by 50% from 150×10^6 points to 75×10^6 points. Memory requirements are reduced by 22%, from 2.62 GB to 2.04 GB. This reduction is less than 50% due to the additional bookkeeping data structures of the sparse block grid, and because a chunk of $8 \times 8 \times 8$ points is allocated whenever there is at least one point inside it, resulting in allocation of 77% of all chunks in this example. The GPU time (excluding communication) and wall-clock time (including all necessary communication) per time step for the sparse block grid is around 4.47 ms and 4.48 ms, respectively, for the highest resolution on a single Nvidia A100 GPU. This is 13% faster than the dense grid implementation with 5.11 ms and 5.13 ms, respectively, on the same GPU. The speedup of a sparse over a dense grid in this example is modest because stencil operations are only performed for dense-grid points that lie in the fluid phase, requiring only one `if` condition more on a dense grid than on a sparse grid. Despite the $CaCO_3$ packed-bed sample being a rather dense geometry, however, the sparse grid offers some advantage, mainly in memory usage.

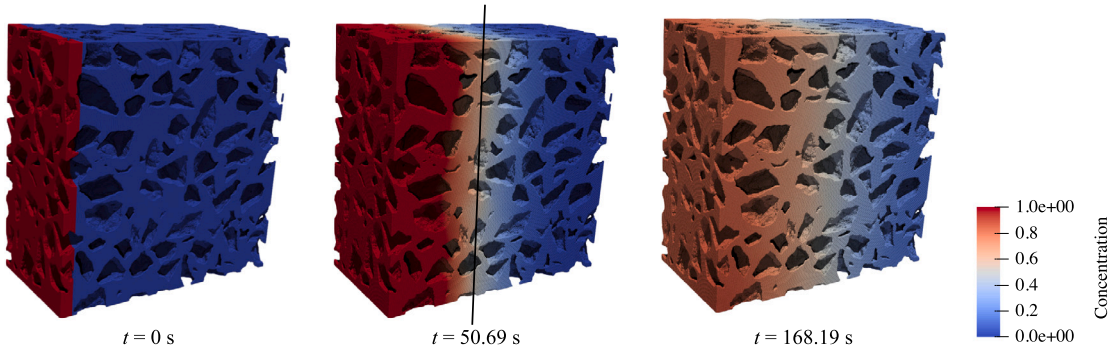


Fig. 8. Simulation snapshots after 0, 4×10^5, and 10^6 time steps of a simulation of inhomogeneous diffusion of fertilizer in the fluid phase of a $CaCO_3$ packed bed. In the simulated scenario, the fertilizer dissolved in the fluid enters from one face of the simulation domain and diffuses through the fluid phase with an inhomogeneous diffusion coefficient that models hindrance by micropores at the $CaCO_3$ surfaces. This causes the diffusion front speed to depend on the locally varying pore geometry. The black vertical line indicates the location of a flat front with homogeneous speed for comparison.

3.5. Application example 2: Heat conduction through reticulate porous ceramics

Another process that a reaction–diffusion PDE can describe is heat conduction through a solid. Heat conduction in porous media is relevant in many applications, ranging from thermal energy storage [92] to building insulation [93]. A class of materials with particularly beneficial characteristics for high-temperature applications are Reticulate Porous Ceramics (RPCs). RPCs play a role in applications including solar-thermal and thermo-chemical reactors for renewable energy technology [94], radiant burners [95], and gas filters [7,96]. We use this as a second application showcase for the present simulation pipeline.

We therefore simulate diffusive heat conduction through the solid phase of a RPC sample with heat dissipation at the surfaces. The μCT images of an RPC sample of size 1216 × 1016 × 941 voxels with a resolution of 30 μm/voxel were kindly provided by Prof. Jörg Petrasch (Michigan State University, College of Engineering) [52,97]. We segmented the images using the pixel-wise random-forest classifiers from ilastik [53] and obtained the SDF of the solid phase using the approach described in Section 3.2. In the simulated scenario, heat diffuses through the RPC solid phase with a distributed heat sink at the solid surface. Using the level-set SDF, it becomes straightforward to constrain the heat sink to the surface as a reaction term (see Fig. 9b,c).

Fig. 9d shows snapshots of the simulation. The initial temperature distribution (left) is uniform, except for a sphere of radius $235h$ in the center, where the temperature is higher. From here, temperature diffuses radially outwards with a diffusion coefficient of $D = 0.1\,\mathrm{mm}^2/\mathrm{s}$. Heat is dissipated at the solid surface, which acts as a sink with rate $f(\boldsymbol{x}) = 0.1\,\mathrm{s}^{-1}$ for $\boldsymbol{x} \in \partial\Omega$. Starting from a spherically symmetric initial condition, the temperature distribution becomes increasingly asymmetric and irregular over time (Fig. 9d, middle), although a homogeneous diffusion coefficient is used. Like in Section 3.4, the diffusion front velocity is inhomogeneous (cf. white circle in Fig. 9d, right) due to the spatially varying tortuosity and heterogeneous pore connections in the RPC sample, as well as the local variations in surface area impacting heat flux due to the surface dissipation. This heterogeneity can only be captured when considering the pore-scale geometry.

We again compare the computational cost of the present sparse-grid implementation with that of a dense block grid in a domain of 1216 × 1016 × 941 grid cells. This shows a significant reduction in allocated memory and GPU time as summarized in Table 1. This is because the RPC solid phase is truly sparse, filling only about 9% of the space, with points spread over 21% of the grid chunks. The number of mesh nodes in the sparse grid is thus reduced to 10^8 from the 1.2×10^9 of a full grid, which is an 11-fold reduction. This significantly reduces the memory requirement from 20.3 GB (dense) to 4.2 GB (sparse).

Again, the reduction in memory requirement is less than in the number of grid points, due to the 8 × 8 × 8 chunk size of the sparse block grid. The GPU time (excluding communication) and wall-clock time (including all necessary communication) per time step for the sparse grid are around 3.79 ms and 4.12 ms, respectively, for the highest resolution on two Nvidia A100 GPUs connected via NVLink. This is 76% faster than the dense grid implementation with 16.68 ms and 16.97 ms, respectively, on the same two GPUs. Compared with the example from Section 3.4, this demonstrates that a sparser geometry leads to better performance gains.

3.6. Performance and scalability

We benchmark computational performance of our code for different grid resolutions and on different numbers of GPUs. All benchmarks are performed on A100-SXM4 GPUs of the *taurus* computer of TU Dresden, which has 8 GPUs per compute node. GPUs within a node are connected by NVLink, whereas individual nodes are connected by 200 Gb/s Infiniband. We use Nvidia's *Unified Communication - X Framework* (UCX), as integrated into OpenMPI, with the `SKIP_LABELLING` option enabled, since the grid nodes do not move.

We first measure how the memory requirements and wall-clock times scale with problem size for both dense and sparse grids. The results are given in Table 1 and Fig. 10. In this benchmark, "dense grid" refers to a fully occupied (i.e., all chunks are allocated) OpenFPM block grid data structure. The grid resolution is successively halved along one dimension at a time, so the expected scaling is linear (solid line in Fig. 10). Although the fully occupied chunks of the dense grid enable better thread efficiency on the GPU than the partly filled chunks of the sparse grid, Fig. 10 confirms that the sparse grid scales almost as well as the dense grid, and both scale almost linearly.

We also benchmark the parallel scalability of the code when distributing the problem over an increasing number of GPUs. The results for both weak (problem size increases proportional to GPU count[2]) and strong (constant problem size) scaling are shown in Figs. 11 (for the $CaCO_3$ case) and 12 (for the RPC case). In both figures, the wall-clock times include communication and are averaged over all GPUs and 80 time steps, excluding the first 50 steps of GPU warm-up. For both application cases, good scalability is observed up to 8 GPUs. As expected, weak scaling is better than strong scaling. This is because in strong scaling the computation time per GPU reduces with increasing GPU count, causing the communication overhead to grow and eventually become limiting. This is exacerbated when using more than a single compute node, i.e. from 8 to 16 GPUs, requiring communication across nodes via

[2] We scale problem size by scaling the size of the background mesh. The number of actually allocated nodes in the sparse grid depends on the geometry, but never deviated more than 1% from linear scaling.

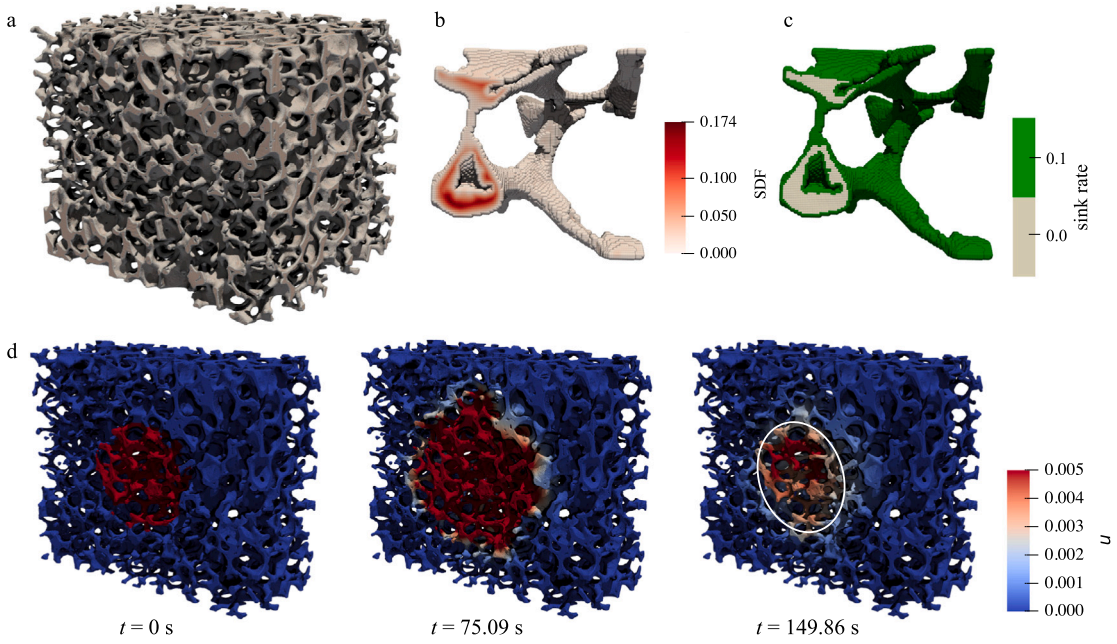


Fig. 9. Diffusive heat conduction with distributed surface sink in a sample of reticulate porous ceramics. **(a)** Surface rendering of the image-based reconstruction of the solid phase represented as a sparse-grid level set. **(b)** Magnification of one exemplary pore with the level-set SDF shown in color to demonstrate the intricate shape of the pores. **(c)** Heat-sink rate around the same pore to show how dissipation is restricted to the surface of the solid phase by the level-set function. **(d)** Simulation snapshots shown as volume renderings in a clipped domain with temperature shown by color (arbitrary units). The initial condition is a spherical heat source at the center of the domain (left). From here, heat spreads through the solid with heat dissipation at the surface (middle, right). The diffusion front significantly deforms over time due to heterogeneous pore connections and the local variations in surface dissipation. The white circle indicates the location of a symmetric front with homogeneous speed for comparison.

Table 1
Comparison of memory requirements and wall clock times for one iteration (including all necessary communication) for a dense versus sparse grid implementation of the present pipeline in the two application examples. The sparser the geometry (on the $8 \times 8 \times 8$ chunk level), the larger the savings afforded by the sparse grid, as less grid chunks are allocated. The wall clock times are averages over 100 time steps per GPU, each after 31 steps of GPU warm-up, which are excluded from the averages. The estimator of the standard deviation of the mean (i.e., the standard error) is given after the ±. All tests are performed on Nvidia A100-SXM4 GPUs using FP32 precision.

Case	Grid size	Dense grid		Sparse grid	
		DRAM (GB)	Time/step (ms)	DRAM (GB)	Time/step (ms)
$CaCO_3$ (1 GPU)	$531 \times 531 \times 531$	2.62	5.126 ± 0.001	2.04	4.484 ± 0.001
	$266 \times 531 \times 531$	1.33	2.714 ± 0.024	1.11	2.578 ± 0.025
	$266 \times 266 \times 531$	0.67	1.487 ± 0.016	0.59	1.507 ± 0.000
	$266 \times 266 \times 266$	0.34	0.852 ± 0.000	0.32	0.873 ± 0.000
RPC (2 GPUs)	$1216 \times 1016 \times 941$	20.26	16.972 ± 0.001	4.24	4.117 ± 0.009
	$608 \times 1016 \times 941$	10.26	8.739 ± 0.001	2.54	2.622 ± 0.006
	$608 \times 508 \times 941$	5.13	4.422 ± 0.003	1.48	1.765 ± 0.001
	$608 \times 508 \times 471$	2.57	2.355 ± 0.002	0.84	1.028 ± 0.003

the interconnect network. Taken together, these measurements show that our implementation of the present pipeline scales as expected for different problem sizes and numbers of GPUs.

4. Discussion and conclusions

We have presented a GPU-accelerated simulation pipeline for solving inhomogeneous reaction–diffusion PDEs in real-world, image-based geometries of porous media. Quantitatively understanding the pore-scale dynamics of reaction–diffusion processes is relevant to applications where homogenization or volume-averaging would not be informative. We have demonstrated this by direct numerical simulation of fertilizer diffusion in soil and heat conduction in reticulate porous ceramics. In both examples, pore-scale processes such as micropore adhesion or surface heat dissipation could easily and accurately be included in the simulation. The geometry-adapted sparse grid level-set discretization used in the presented pipeline was able to efficiently and effectively cope with the wide spectrum of scales present in fully resolved geometries of porous media.

The implementation of the present pipeline included extending the GPU implementation of distributed sparse block grids [31] to incorporate a level-set method for image-based modeling. The resulting workflow takes microscopy or μCT images as input, from which it generates realistic 3D geometries and represents them as a sparse level-set. On the process modeling side, we extended the capabilities of the scalable OpenFPM framework for scientific computing to include space-dependent diffusion coefficients and locally varying reactions. Together, these contributions enabled fully resolved simulations of inhomogeneous reaction–diffusion processes in non-parametric, image-based geometries of porous media.

After benchmarking the correctness and convergence of the implementation, we showcased the pipeline in two real-world examples. In the first example, we numerically solved an inhomogeneous reaction–diffusion PDE in a sparse grid that discretized the fluid phase of a $CaCO_3$ particle-packed bed. For this example, the sparse grid reduced the memory requirement by 22% and the runtime by 13% compared to a dense grid. In the second example, we simulated heat conduction with distributed surface dissipation in the solid phase of reticulate porous ceramics. Reticulate porous ceramics are extremely sparse, and the gain from using a sparse block grid was, therefore, higher than for the $CaCO_3$ particle-packed bed. In particular, the memory requirement was reduced 4.8-fold, and the simulation runtime was reduced by 76%.

For both application examples, we also benchmarked how the computational cost scales with problem size and with the number of GPUs used. We showed that the present implementation scales approximately linearly with problem size on a single GPU ($CaCO_3$) and on two GPUs (RPC). Parallelizing over more GPUs, we found almost perfect weak scaling and very good strong scaling when using up to 8 GPUs on the same compute node. As expected, the communication overhead became limiting when distributing the simulation across multiple nodes, as we showed for 16 GPUs distributed across 2 compute nodes.

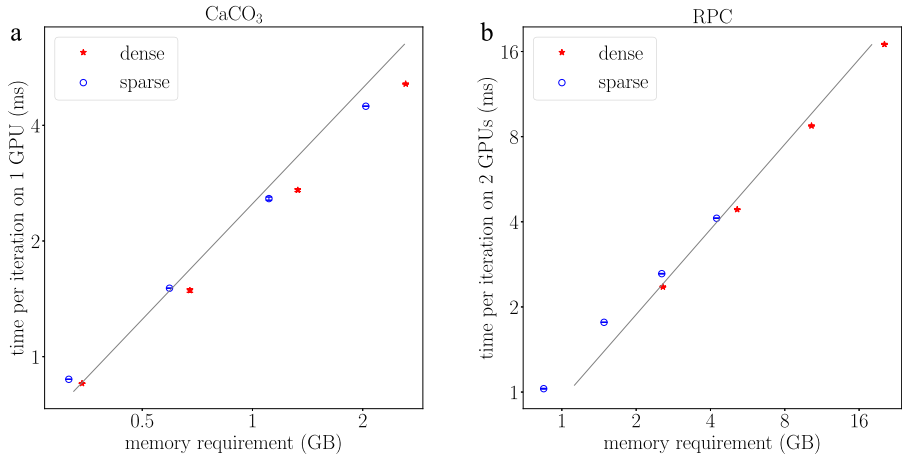
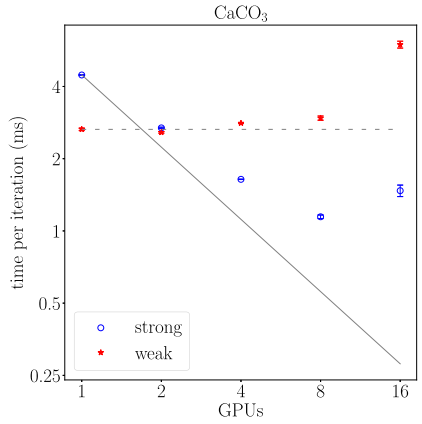


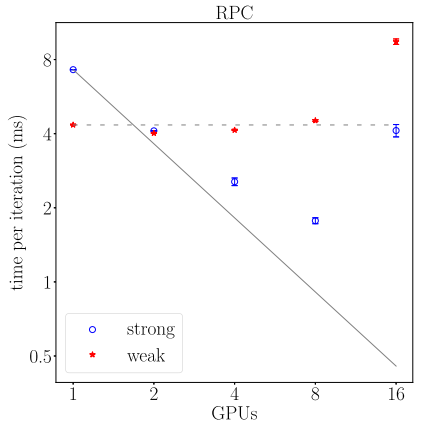
Fig. 10. Plots of the data from Table 1 comparing the wall-clock times per time step and the memory requirements for dense (red stars) and sparse (blue circles) block grids on a fixed number of GPUs. Error bars show the standard error; solid lines denote linear scaling. **(a)** Inhomogeneous diffusion in a $CaCO_3$ particle-packed bed on a single Nvidia A100 GPU. **(b)** Heat conduction in reticulate porous ceramics on 2 Nvidia A100 GPUs connected via NVLink.



strong scaling		
#	grid size	time/step (ms)
1	$531 \times 531 \times 531$	4.466 ± 0.000
2	$531 \times 531 \times 531$	2.688 ± 0.018
4	$531 \times 531 \times 531$	1.641 ± 0.003
8	$531 \times 531 \times 531$	1.145 ± 0.020
16	$531 \times 531 \times 531$	1.471 ± 0.081

weak scaling		
#	grid size	time/step (ms)
1	$266 \times 531 \times 531$	2.652 ± 0.032
2	$531 \times 531 \times 531$	2.574 ± 0.020
4	$1062 \times 531 \times 531$	2.812 ± 0.012
8	$1062 \times 1062 \times 531$	2.953 ± 0.060
16	$1062 \times 1062 \times 1062$	5.978 ± 0.197

Fig. 11. Strong and weak parallel scaling of the present implementation on multiple GPUs (#), simulating inhomogeneous diffusion in a $CaCO_3$ particle-packed bed. The wall clock times are averaged over GPUs and over 80 time steps after 50 steps of GPU warm-up, which are excluded from the averages. All measurements are performed on Nvidia A100 GPUs using FP32 precision. In the plot, blue circles denote strong and red stars weak scaling. Lines show the ideal scaling (solid: strong, dashed: weak). Error bars in the plot and error values in the table give the standard error.



strong scaling		
#	grid size	time/step (ms)
1	$1216 \times 1016 \times 941$	7.280 ± 0.000
2	$1216 \times 1016 \times 941$	4.108 ± 0.009
4	$1216 \times 1016 \times 941$	2.552 ± 0.091
8	$1216 \times 1016 \times 941$	1.772 ± 0.049
16	$1216 \times 1016 \times 941$	4.120 ± 0.238

weak scaling		
#	grid size	time/step (ms)
1	$608 \times 1016 \times 941$	4.343 ± 0.001
2	$1216 \times 1016 \times 941$	4.017 ± 0.003
4	$2432 \times 1016 \times 941$	4.138 ± 0.017
8	$2432 \times 2032 \times 941$	4.522 ± 0.038
16	$2432 \times 2032 \times 1882$	9.458 ± 0.245

Fig. 12. Strong and weak parallel scaling of the present implementation on multiple GPUs (#), simulating heat conduction in reticulate porous ceramics (RPC). The wall clock times are averaged over GPUs and over 80 time steps after 50 steps of GPU warm-up, which are excluded from the averages. All measurements are performed on Nvidia A100 GPUs using FP32 precision. In the plot, blue circles denote strong and red stars weak scaling. Lines show the ideal scaling (solid: strong, dashed: weak). Error bars in the plot and error values in the table give the standard error.

In macroscopic homogenized models, the impact of a porous geometry on diffusive transport processes can be accounted for by material-dependent constants like tortuosity and effective diffusivity. These constants are thus of great interest, but obtaining them experimentally is cumbersome [98–100]. We have therefore shown how the present image-based simulation pipeline enables determining effective diffusion coefficients and tortuosity numerically by simulating FRAP in the transport phase of the diffusion space.

While most currently available image-based simulation pipelines for transport processes in porous media involve commercial software [101], such as Avizo [102], Ansys [103], or COMSOL [104], the pipeline presented here is entirely free and open-source. The code is written in C++ using the OpenFPM library.

A current limitation of the present pipeline is the accuracy of the boundary approximation, which is first-order. This is not an intrinsic limitation of the level-set approach, though, which would allow for higher-order boundary treatment by accounting for surface normals and/or curvature. Our approach, however, only requires regular computations on a sparse grid, which is conducive to GPU acceleration as was the main goal here. While other methods for imposing boundary conditions, like immersed boundary and immersed interface methods, require transport phases on both sides of the interface [105], this is not the case for the present inhomogeneous diffusion model, where the interface is the actual domain boundary. Finally, penalization methods, which modify the force terms in the momentum equation to account for boundary effects, only apply to flow problems [106]. Therefore, we use the SDF to detect the boundary location and impose boundary conditions within the finite-difference stencil, which is well suited for solving reaction–diffusion PDEs, as it is mass-conserving and allows for reactive boundaries. Furthermore, the SDF can directly be used to represent inhomogeneous diffusion coefficients, as shown here.

Obtaining a SDF by level-set redistancing, however, can cause numerical dissipation of the interface geometry. We largely avoided this by using Sussman redistancing, which is interface-preserving [62,107] and has a discretization error that converges with the correct order (Fig. A.1). Our implementation of Sussman redistancing, however, is currently limited to a dense grid without GPU acceleration, generating a one-off computational overhead at the beginning of a simulation. Although this is not a bottleneck for reactive transport simulations in static geometries, future work will explore whether Sussman redistancing can directly be performed on a sparse grid with GPU acceleration for simulations in dynamic geometries. Addition and deletion of points in a sparse grid on the GPU, however, is not efficient, as it requires semaphores to avoid race conditions as well as restructuring or reallocation of memory. Another advantage of performing Sussman redistancing on the host is that its memory capacity is not limited to the DRAM of the GPU. Moreover, our implementation of Sussman redistancing is parallelized for multi-CPU, enabling the computation of SDFs in large grids that need not fit the memory of a single compute node.

Finally, the current parallel implementation of the present pipeline is limited to explicit time-integration methods. Benefiting from the improved numerical stability of implicit schemes, which allow for larger time-step sizes, would require addressing their larger communication overhead.

Future work will also include applying the present pipeline to other kinds of porous media. In particular, biological tissues could be good candidates, as they tend to have irregular shapes and are porous (e.g., the mineral matrix of bone [108], fibrocartilaginous structures [109] like meniscus [110], bile-canalicular networks in the liver [111,112], and the extracellular matrix between cells [113]). In all of these porous media, reaction–diffusion processes are essential to transport water, nutrients, metabolic products, and molecules with regulatory or signaling function. In addition, spatial tissue patterning by graded concentration fields during embryonic development is controlled by a complex interplay between geometry and reaction–diffusion processes [2,11,114–117]. The increasing availability of high-resolution 3D microscopy images of biological tissues provides a unique chance for understanding the organizational principles of morphogenesis [12,114,118]. We, therefore, anticipate that the generic and efficient computational pipeline presented here will also find application in this field.

CRediT authorship contribution statement

Justina Stark: Conceived and designed the analysis, Collected the data, Contributed data or analysis tools, Performed the analysis, Wrote the paper. **Ivo F. Sbalzarini:** Conceived and designed the analysis, Performed the analysis, Participated in interpretation of the data, Wrote the paper, Project supervision, Funding acquisition.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Data availability

The software is available under the GNU General Public License 3.0 (GPLv3) as open source from https://git.mpi-cbg.de/mosaic/reactiondiffusion_imagebased_porousmedia.git.

Acknowledgments

We thank Prof. Jörg Petrasch (Michigan State University, College of Engineering) for providing the μCTs of the $CaCO_3$ and RPC samples. We thank Dr. Pietro Incardona (University of Bonn) for many helpful discussions, proofreading, implementation support with OpenFPM, and the idea of the boundary treatment. From the Sbalzarini group, we thank Dr. Nandu Gopan, Dr. Johannes Pahlke, Alejandra Foggia, and Dr. Aryaman Gupta for discussions and proofreading, and Serhii Yaskovets for support with OpenFPM. We thank Dr. Michele Marass (MPI-CBG, Dresden) for editorial advice. We thank Dr. Quentin Vagne (University of Geneva) for helpful discussions. We are grateful to the Scientific Computing Facility of MPI-CBG Dresden and the Center for Information Services and High-Performance Computing (ZIH) of TU Dresden for providing their facilities for the benchmarks. We thank the anonymous reviewers for their time and constructive suggestions.

Appendix A. Additional verification results

We provide here below additional information about the analytical solution used in the convergence tests, as well as about the convergence of our implementation of Sussman redistancing of the level-set function.

A.1. Exact solution for diffusion inside a 2D disk using the method of manufactured solutions

We derive the exact solution of the test case used to verify our pipeline in Section 3.1. For this, we use the method of manufactured solutions [68]. The test case considers homogeneous diffusion of a scalar field u inside a 2D unit disk with no-flux Neumann boundary conditions and diffusion coefficient $D = 1$.

A general solution of Eq. (3.1.1) proposed in the literature for this case is [119]:

$$U(r,t) = \left(\frac{r^3}{3} - \frac{r^4}{4}\right) \mathrm{e}^{-Bt}. \tag{A.1}$$

We verify that this fulfills the boundary condition:

$$\left.\frac{\partial U(r,t)}{\partial \boldsymbol{n}}\right|_{r=1} = \frac{\partial U(R,t)}{\partial r} = (R^2 - R^3)\mathrm{e}^{-Bt} = 0\,. \tag{A.2}$$

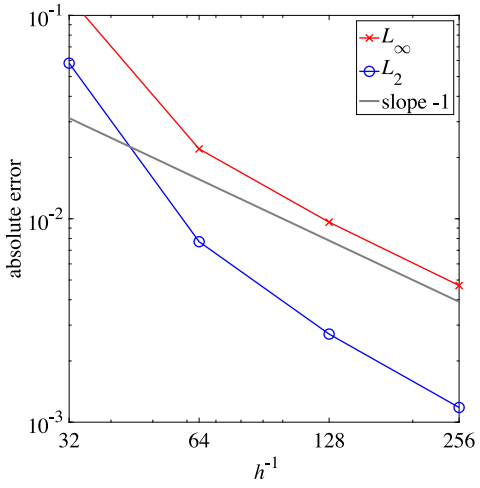


Fig. A.1. Convergence plot of Sussman redistancing for the level-set of the unit ball in 3D. The L_2 (blue circles) and L_∞ (red crosses) norms of the absolute error in the SDF across the entire narrow band of width $4h$ are plotted. Both show the expected linear convergence with the grid spacing h for a fixed time-step size. The upwind gradient computed during iterative reinitialization of the level-set function is approximated using first-order finite differences.

In order to find the particular solution that also fulfills the initial condition, the method of manufactured solutions adds a reaction term $f(r,t)$ [68]. For this, we write Eq. (3.1.1) as an operator

$$L(u) = \frac{\partial u}{\partial t} - \frac{\partial^2 u}{\partial r^2} - \frac{1}{r}\frac{\partial u}{\partial r} - \frac{1}{r^2}\frac{\partial^2 u}{\partial \theta^2} = 0. \tag{A.3}$$

and find $f(r,t)$ by applying the operator to $U(r,t)$:

$$\begin{aligned} f(r,t) &= L(U) \\ &= \frac{\partial U}{\partial t} - \frac{\partial^2 U}{\partial r^2} - \frac{1}{r}\frac{\partial U}{\partial r} - \frac{1}{r^2}\frac{\partial^2 U}{\partial \theta^2} \\ &= \left(\frac{B}{4}r^4 - \frac{B}{3}r^3 + 4r^2 - 3r\right)\mathrm{e}^{-Bt}. \end{aligned} \tag{A.4}$$

Using this reaction term in the governing equation, the problem has the exact solution given in Eq. (A.1). The scalar constant B can be arbitrarily chosen and controls the gradient steepness of the initial concentration field. We choose $B = 20$, which provides an initial concentration field with smooth gradients and values suitable to be dealt with by single-precision arithmetics.

A.2. Convergence of Sussman redistancing

In Fig. A.1, we present a convergence plot of our implementation of Sussman redistancing of the level-set SDF of the unit ball in 3D using first-order upwind finite differences. The expected order of convergence is achieved as soon as the geometry is well resolved.

Appendix B. Code examples

We provide code examples illustrating the benefits of the OpenFPM-based implementation of our simulation pipeline. The complete source code of the present implementation is available under the GNU General Public License 3.0 (GPLv3) at https://git.mpi-cbg.de/mosaic/software/parallel-computing/openfpm and https://git.mpi-cbg.de/mosaic/reactiondiffusion_imagebased_porousmedia.git with examples and documentation at http://ppmcore.mpi-cbg.de/doxygen/openfpm/index.html. For demonstration, we only show a few code examples here.

To construct a regular, dense grid in OpenFPM, the grid and domain size, ghost layer thickness, and the data-types and sizes of the grid properties have to be defined first. If the sparse will be created based on this dense grid, also the decomposition for the distribution has to be defined. Then, the grid can be allocated.

Listing 1: Distributed dense Cartesian grid construction.

```
// Grid type with property aggregate,
// here just one FP32 that will carry the level-set
    function
typedef aggregate<float> props;
// Decomposition for distribution
typedef CartDecomposition<dims,float, CudaMemory,
memory_traits_inte, BoxDistribution<dims,float> > Dec;
typedef grid_dist_id<dims, float, props, Dec >
    grid_in_type;
// Instantiate distributed regular grid
grid_in_type g_dist(sz, box, ghost);
```

The indicator function obtained from the segmentation mask is loaded onto the grid and reinitialized as a SDF using Sussman redistancing, which we implemented as a class template in OpenFPM. Code examples and documentation for this implementation are available at http://ppmcore.mpi-cbg.de/doxygen/openfpm/example_sussman_images_2D.html. Here, we show a snippet of the constructor and how to use it. Iterative redistancing is performed on a temporary grid allocated inside the class with the same domain decomposition as the input grid. Therefore, the fields needed during the redistancing to store the intermediate level-set function, ϕ_n, its upwind gradient, $\nabla\phi_n$, and the sign of the initial ϕ, sign(ϕ_0), as well as the required ghost layers for inter-process communication, are hidden from the user. In our application, the input grid has only one property, namely the pre-redistancing level-set function.

Listing 2: Sussman redistancing class snippet.

```
template <typename grid_in_type,
typename phi_type=double>
class RedistancingSussman
{
public:
        RedistancingSussman(grid_in_type & grid_in,
        Redist_options<phi_type> &redistOptions) :
        redistOptions(redistOptions),
        r_grid_in(grid_in),
        g_temp(grid_in.getDecomposition(),
        grid_in.getGridInfoVoid().getSize(),
        Ghost<grid_in_type::dims, long int>(3))
        ...
```

Listing 3: Loading the indicator function from a file onto the grid.

```
// Initialize grid with (image-based) indicator function
load_pixel_onto_grid<PHI_FULL>(g_dist,
path_to_zstack, stack_dims);
```

For running Sussman redistancing, options such as the maximum number of iterations and the convergence tolerance can be passed in the structure `Redist_options` during class instantiation. Redistancing is then executed by calling the method `run_redistancing` with the indices of the input and output properties as template arguments.

Listing 4: Sussman redistancing instantiation and execution.

```
// Instantiation of Sussman-redistancing class
RedistancingSussman<grid_type, float>
redist_obj(g_dist, redist_options);
// Run Sussman redistancing
redist_obj.run_redistancing<PHI_IN, PHI_OUT>();
```

Based on the resulting SDF, the sparse grid can be constructed by inserting points only within the diffusion domain, i.e., for points at which `lower_bound` $< \phi_{\mathrm{SDF}} <$ `upper_bound`. Here, `lower_bound` and `upper_bound` are the minimal and maximal values of ϕ_{SDF} within the diffusion domain, depending on the type of level-set function chosen and the phase considered.

Listing 5: Obtain a sparse grid based on the SDF.

```
// Create a sparse grid with four FP32 properties and same
// decomposition Dec as dense grid
typedef aggregate<float, float, float, float> props_sparse
    ;
typedef sgrid_dist_id_gpu<dims, float, props_sparse,
CudaMemory, Dec> sparse_grid_type;
sparse_grid_type g_sparse(sz, box, ghost);

// At this stage, the sparse grid is still empty. Now, we
    can loop
// over the dense grid and insert points in the sparse
    grid if they
// lie inside the diffusion domain
template <typename T>
static bool is_diffusionSpace(const T & phi_sdf,
const T & lower_bound, const T & upper_bound)
{
        const T EPSILON = std::numeric_limits<T>::epsilon
            ();
        const T _b_low = lower_bound + EPSILON;
        const T _b_up  = upper_bound  - EPSILON;
        return (phi_sdf > _b_low && phi_sdf < _b_up);
}

auto dom = grid.getDomainIterator();
while(dom.isNext())
{
        auto key = dom.get();
        if(is_diffusionSpace(grid.template get<
            PHI_SDF_FULL>(key),
        b_low, b_up))
        {
                sparse_grid.template
                insertFlush<PHI_SDF_SPARSE>(key) =
                grid.template get<PHI_SDF_FULL>(key);
        }
        ++dom;
}
```

The computations to be run on the sparse grid on the GPUs are defined as a functor. C++ functors are object-like functions, which OpenFPM can pass to the GPU as CUDA or HIP kernels. We demonstrate this exemplary for the case of inhomogeneous diffusion with explicit time stepping.

Listing 6: Defining the functor for inhomogeneous diffusion on the GPU.

```
auto epsilon = std::numeric_limits<float>::epsilon();
auto func_inhomogDiffusion =
[dx, dy, dz, dt, d_low, epsilon]
__device__ (
float & u_out,          // field out
float & D_out,          // diffusion coefficient out
float & phi_out,        // sdf of domain out
CpBlockType & u,        // field in
CpBlockType & D,        // diffusion coefficient in
CpBlockType & phi,      // sdf of domain in
auto & block, int offset, int i, int j, int k)
{
        // Stencil
        // Field
        float u_c  = u(i, j, k);
        float u_px = u(i+1, j, k);
        float u_mx = u(i-1, j, k);
        ... // and so on for y and z

        // Signed distance function
        float phi_c  = phi(i, j, k);
        float phi_px = phi(i+1, j, k);
        float phi_mx = phi(i-1, j, k);
        ... // and so on for y and z

        // Diffusion coefficient
        float D_c  = D(i, j, k);
        float D_px = D(i+1, j, k);
        float D_mx = D(i-1, j, k);
        ... // and so on for y and z

        // Impose no-flux boundary conditions
        if (phi_px <= d_low + epsilon)
        {u_px = u_c; D_px = D_c;}
        if (phi_mx <= d_low + epsilon)
        {u_mx = u_c; D_mx = D_c;}
        ... // and so on for y and z

        // Interpolate diffusion constants between points
        float D_half_px = (D_c + D_px)/2.0;
        float D_half_mx = (D_c + D_mx)/2.0;
        ... // and so on for y and z

        // Compute concentration of next time point
        u_out = u_c + dt *
        (1/(dx*dx) * (D_half_px * (u_px - u_c)
        - D_half_mx * (u_c - u_mx)) +
        ... // and so on for y and z);

        // Diffusion coefficient and SDF out=in
        D_out   = D_c; phi_out = phi_c;
};
```

To solve the diffusion equation over time, this functor and the sparse grid containing the initial condition are downloaded to the GPUs, where they are executed using the convolution functor `conv3_b` with ghost layers communicated between GPUs in a multi-GPU setting.

Listing 7: GPU convolution functor for sparse grids

```
// Copy from host to GPU
g_sparse.template hostToDevice<CONC_N, CONC_NPLUS1,
DIFFUSION_COEFFICIENT, PHI_PHASE>();
// Ghost layer communication of the signed distance
    function
// and the inhomogeneous diffusion coefficient
// The SKIP_LABELLING option is only allowed for static
    geometries
g_sparse.template ghost_get<DIFFUSION_COEFFICIENT,
    PHI_PHASE>
(RUN_ON_DEVICE | SKIP_LABELLING);
while(iter <= iterations)
{...
    // Update concentration values in ghost layer
    g_sparse.template ghost_get<CONC_N>
    (RUN_ON_DEVICE | SKIP_LABELLING);
    // Convolve functor with sparse grid on GPU
    g_sparse.template conv3_b<CONC_N,
        DIFFUSION_COEFFICIENT,
    PHI, CONC_NPLUS1, DIFFUSION_COEFFICIENT, PHI_PHASE, 1>
    ({0, 0, 0},
    {(long int) sz[x]-1, ... // and so on for y and z},
    func_inhomogDiffusion);
...}
```

References

[1] N. Wakao, J.M. Smith, Diffusion in catalyst pellets, Chem. Eng. Sci. 17 (11) (1962) 825–834, http://dx.doi.org/10.1016/0009-2509(62)87015-8.

[2] P. Müller, K.W. Rogers, B.M. Jordan, J.S. Lee, D. Robson, S. Ramanathan, A.F. Schier, Differential diffusivity of nodal and lefty underlies a reaction–diffusion patterning system, Science 336 (6082) (2012) 721–724, http://dx.doi.org/10.1126/science.1221920, URL https://www.science.org/doi/abs/10.1126/science.1221920.

[3] N. Beaudoin, A. Hamilton, D. Koehn, Z.K. Shipton, U. Kelka, Reaction-induced porosity fingering: Replacement dynamic and porosity evolution in the KBr-KCl system, Geochim. Cosmochim. Acta 232 (2018) 163–180, http://dx.doi.org/10.1016/j.gca.2018.04.026.

[4] J. Cotterell, A. Robert-Moreno, J. Sharpe, A local, self-organizing reaction-diffusion model can explain somite patterning in embryos, Cell Syst. 1 (4) (2015) 257–269, http://dx.doi.org/10.1016/j.cels.2015.10.002.

[5] P. Grathwohl, Diffusion in Natural Porous Media Contaminant Transport, Sorption/Desorption and Dissolution Kinetics, first ed., Springer, 1998.

[6] M. Alhijjaj, S. Yassin, M. Reading, J.A. Zeitler, P. Belton, S. Qi, Characterization of heterogeneity and spatial distribution of phases in complex solid dispersions by thermal analysis by structural characterization and X-ray micro computed tomography, Pharm. Res. 34 (5) (2017) 971–989, http://dx.doi.org/10.1007/s11095-016-1923-3.

[7] S. Haussener, W. Lipiński, J. Petrasch, P. Wyss, A. Steinfeld, Tomographic characterization of a semitransparent-particle packed bed and determination of its thermal radiative properties, J. Heat Transfer 131 (7) (2009) 1–11, http://dx.doi.org/10.1115/1.3109261.

[8] A.M. Turing, The chemical basis of morphogenesis, Philos. Trans. R. Soc. Lond. 237 (641) (1952) 37–72.

[9] L. Wolpert, Positional information and the spatial pattern of cellular differentiation, J. Theoret. Biol. 25 (1) (1969) 1–47, http://dx.doi.org/10.1016/S0022-5193(69)80016-0.

[10] F. Crick, E.M. Deuchar, Diffusion in embryogenesis, Nature 225 (5233) (1970) 671, http://dx.doi.org/10.1038/225671b0.

[11] O. Wartlick, A. Kicheva, M. González-Gaitán, Morphogen gradient formation, Cold Spring Harb. Perspect. Biol. 1 (3) (2009) 1–22, http://dx.doi.org/10.1101/cshperspect.a001255.

[12] A. Kicheva, T. Bollenbach, O. Wartlick, F. Jülicher, M. Gonzalez-Gaitan, Investigating the principles of morphogen gradient formation: From tissues to cells, Curr. Opin. Genetics Dev. 22 (6) (2012) 527–532, http://dx.doi.org/10.1016/j.gde.2012.08.004.

[13] S.R. Yu, M. Burkhardt, M. Nowak, J. Ries, Z. Petrášek, S. Scholpp, P. Schwille, M. Brand, Fgf8 morphogen gradient forms by a source–sink mechanism with freely diffusing molecules, Nature 461 (7263) (2009) 533–536, http://dx.doi.org/10.1038/nature08391.

[14] B. Ghanbarian, A.G. Hunt, R.P. Ewing, M. Sahimi, Tortuosity in porous media: A critical review, Soil Sci. Am. J. 77 (5) (2013) 1461–1477, http://dx.doi.org/10.2136/sssaj2012.0435.

[15] F.J. Valdés-Parada, D. Lasseux, S. Whitaker, Diffusion and heterogeneous reaction in porous media: The macroscale model revisited, Int. J. Chem. React. Eng. 15 (6) (2017) http://dx.doi.org/10.1515/ijcre-2017-0151.

[16] I. Battiato, P.T. Ferrero V, D. O'Malley, C.T. Miller, P.S. Takhar, F.J. Valdés-Parada, B.D. Wood, Theory and applications of macroscale models in porous media, Transp. Porous Media 130 (1) (2019) 5–76, http://dx.doi.org/10.1007/s11242-019-01282-2.

[17] A. Hrouda, L. Capek, M. Vanierschot, K. Denis, Macroscale simulation of the filtration process of porous media based on statistical capturing models, Sep. Purif. Technol. 266 (2021) 118577, http://dx.doi.org/10.1016/j.seppur.2021.118577, URL https://www.sciencedirect.com/science/article/pii/S138358662102277X.

[18] M.K. Bourbatache, O. Millet, T.D. Le, C. Moyne, Homogenized model for diffusion and heterogeneous reaction in porous media: Numerical study and validation, Appl. Math. Model. 111 (2022) 486–500, http://dx.doi.org/10.1016/j.apm.2022.07.001.

[19] N. Ray, A. Rupp, R. Schulz, P. Knabner, Old and new approaches predicting the diffusion in porous media, Transp. Porous Media 124 (3) (2018) 803–824, http://dx.doi.org/10.1007/s11242-018-1099-x.

[20] L. Li, C.A. Peters, M.A. Celia, Upscaling geochemical reaction rates using pore-scale network modeling, Adv. Water Resour. 29 (9) (2006) 1351–1370, http://dx.doi.org/10.1016/j.advwatres.2005.10.011.

[21] J.D. Boissonnat, Geometric structures for three-dimensional shape representation, ACM Trans. Graph. 3 (4) (1984) 266–286, http://dx.doi.org/10.1145/357346.357349.

[22] J.D. der Waals, The thermodynamic theory of capillary flow under the hipothesis of a continous variation of density, Verhandel./Konink. Akad. Weten. 1 (8) (1893).

[23] J.W. Cahn, J.E. Hilliard, Free energy of a nonuniform system. I. Interfacial free energy, J. Chem. Phys. 28 (2) (1958) 258–267, http://dx.doi.org/10.1063/1.1744102.

[24] S. Osher, J.A. Sethian, Fronts propagating with curvature-dependent speed: Algorithms based on Hamilton–Jacobi formulations, J. Comput. Phys. (1988) http://dx.doi.org/10.1016/0021-9991(88)90002-2.

[25] A. Leonard, Vortex methods for flow simulation, J. Comput. Phys. 37 (3) (1980) 289–335, http://dx.doi.org/10.1016/0021-9991(80)90040-6, URL https://www.sciencedirect.com/science/article/pii/0021999180900406.

[26] P. Degond, S. Mas-Gallic, The weighted particle method for convection–diffusion equations. I. The case of an isotropic viscosity, Math. Comp. 53 (188) (1989) 485–507.

[27] J. Pahlke, I.F. Sbalzarini, A unifying mathematical definition of particle methods, IEEE Open J. Comput. Soc. (2023) 1–12, http://dx.doi.org/10.1109/OJCS.2023.3254466.

[28] D.J. Tildesley, M.P. Allen, Computer Simulation of Liquids, Clarendon Oxford, 1987.

[29] L. Verlet, Computer experiments on classical fluids. I. Thermodynamical properties of Lennard-Jones molecules, Phys. Rev. 159 (1) (1967) 98–103, http://dx.doi.org/10.1103/PhysRev.159.98, URL https://link.aps.org/doi/10.1103/PhysRev.159.98.

[30] B. Schrader, S. Reboux, I.F. Sbalzarini, Discretization correction of general integral PSE Operators for particle methods, J. Comput. Phys. 229 (11) (2010) 4159–4182, http://dx.doi.org/10.1016/j.jcp.2010.02.004, URL https://www.sciencedirect.com/science/article/pii/S002199911000077X.

[31] P. Incardona, T. Bianucci, I.F. Sbalzarini, Distributed sparse block grids on GPUs, in: B.L. Chamberlain, A.-L. Varbanescu, H. Ltaief, P. Luszczek (Eds.), High Performance Computing, in: LNCS, vol. 12728, Springer International Publishing, Cham, 2021, pp. 272–290, http://dx.doi.org/10.1007/978-3-030-78713-4_15.

[32] J. Peiró, S. Sherwin, Finite difference, finite element and finite volume methods for partial differential equations, in: Handbook of Materials Modeling M, 2005, pp. 2415–2446, http://dx.doi.org/10.1007/978-1-4020-3286-8_127.

[33] C. Johnson, Numerical Solution of Partial Differential Equations by the Finite Element Method, Courier Corporation, 2012.

[34] F. Moukalled, L. Mangani, M. Darwish, The Finite Volume Method, Springer, 2016, URL https://link.springer.com/book/10.1007/978-3-319-16874-6.

[35] J.W. Thomas, Numerical Partial Differential Equations: Finite Difference Methods, Vol. 22, Springer Science & Business Media, 2013.

[36] D. Gottlieb, S.A. Orszag, Numerical Analysis of Spectral Methods: Theory and Applications, SIAM, 1977.

[37] P. Incardona, A. Leo, Y. Zaluzhnyi, R. Ramaswamy, I.F. Sbalzarini, OpenFPM: A scalable open framework for particle and particle-mesh codes on parallel computers, Comput. Phys. Comm. 241 (2019) 155–177, http://dx.doi.org/10.1016/j.cpc.2019.03.007, URL https://www.sciencedirect.com/science/article/pii/S0010465519300852.

[38] F. Elwinger, P. Pourmand, I. Furó, Diffusive transport in pores. Tortuosity and molecular interaction with the pore wall, J. Phys. Chem. C 121 (25) (2017) 13757–13764, http://dx.doi.org/10.1021/acs.jpcc.7b03885.

[39] J.A. Ferreira, G. Pena, G. Romanazzi, Anomalous diffusion in porous media, Appl. Math. Model. 40 (3) (2016) 1850–1862, http://dx.doi.org/10.1016/j.apm.2015.09.034.

[40] I.F. Sbalzarini, A. Mezzacasa, A. Helenius, P. Koumoutsakos, Effects of organelle shape on fluorescence recovery after photobleaching, Biophys. J. 89 (3) (2005) 1482–1492, http://dx.doi.org/10.1529/biophysj.104.057885, URL https://www.sciencedirect.com/science/article/pii/S000634950572796X.

[41] J. van Brakel, P.M. Heertjes, Analysis of diffusion in macroporous media in terms of a porosity, a tortuosity and a constrictivity factor, Int. J. Heat Mass Transfer 17 (9) (1974) 1093–1103, http://dx.doi.org/10.1016/0017-9310(74)90190-2.

[42] M.P. Bohrer, G.D. Patterson, P.J. Carroll, Hindered diffusion of dextran and ficoll in microporous membranes, Macromolecules 17 (6) (1984) 1170–1173, http://dx.doi.org/10.1021/ma00136a011.

[43] B.P. Boudreau, The diffusive tortuosity of fine-grained unlithified sediments, Geochim. Cosmochim. Acta 60 (16) (1996) 3139–3142, http://dx.doi.org/10.1016/0016-7037(96)00158-5.

[44] S. Khirevich, A. Höltzel, A. Daneyko, A. Seidel-Morgenstern, U. Tallarek, Structure-transport correlation for the diffusive tortuosity of bulk, monodisperse, random sphere packings, J. Chromatogr. A 1218 (37) (2011) 6489–6497, http://dx.doi.org/10.1016/j.chroma.2011.07.066.

[45] J.M.P.Q. Delgado, A simple experimental technique to measure tortuosity in packed beds, Can. J. Chem. Eng. 84 (6) (2006) 651–655, http://dx.doi.org/10.1002/cjce.5450840603, URL https://onlinelibrary.wiley.com/doi/abs/10.1002/cjce.5450840603.

[46] L. Shen, Z. Chen, Critical review of the impact of tortuosity on diffusion, Chem. Eng. Sci. 62 (14) (2007) 3748–3755, http://dx.doi.org/10.1016/j.ces.2007.03.041.

[47] J.M. Smith, Chemical Engineering Kinetics, second ed., McGraw-Hill, New York, 1970.

[48] C.N. Satterfield, T.K. Sherwood, The Role of Diffusion in Catalysis, Addison-Wesley publishing company, 1963.

[49] S. Gärttner, P. Frolkovič, P. Knabner, N. Ray, Efficiency and accuracy of micro-macro models for mineral dissolution, Water Resour. Res. 56 (8) (2020) http://dx.doi.org/10.1029/2020WR027585.

[50] J. Huang, F. Xiao, H. Dong, X. Yin, Diffusion tortuosity in complex porous media from pore-scale numerical simulations, Comput. & Fluids 183 (2019) 66–74, http://dx.doi.org/10.1016/j.compfluid.2019.03.018.

[51] F. Gray, B. Anabaraonye, S. Shah, E. Boek, J. Crawshaw, Chemical mechanisms of dissolution of calcite by HCl in porous media: Simulations and experiment, Adv. Water Resour. 121 (2018) 369–387, http://dx.doi.org/10.1016/j.advwatres.2018.09.007.

[52] J. Petrasch, P. Wyss, R. Stämpfli, A. Steinfeld, Tomography-based multiscale analyses of the 3D geometrical morphology of reticulated porous ceramics, J. Am. Ceram. Soc. 91 (8) (2008) 2659–2665, http://dx.doi.org/10.1111/j.1551-2916.2008.02308.x.

[53] S. Berg, D. Kutra, T. Kroeger, C.N. Straehle, B.X. Kausler, C. Haubold, M. Schiegg, J. Ales, T. Beier, M. Rudy, K. Eren, J.I. Cervantes, B. Xu, F. Beuttenmueller, A. Wolny, C. Zhang, U. Koethe, F.A. Hamprecht, A. Kreshuk, ilastik: interactive machine learning for (bio)image analysis, Nature Methods (2019) http://dx.doi.org/10.1038/s41592-019-0582-9.

[54] J. Ahrens, B. Geveci, C. Law, ParaView: An end-user tool for large-data visualization, in: The Visualization Handbook, 2005, pp. 717–962, URL https://citeseerx.ist.psu.edu/document?repid=rep1{&}type=pdf{&}doi=37289ca24fde4c63281c061815d625c7acee1e6a{#}page=736.

[55] H. Friess, S. Haussener, A. Steinfeld, J. Petrasch, Tetrahedral mesh generation based on space indicator functions, Internat. J. Numer. Methods Engrg. 93 (10) (2013) 1040–1056, http://dx.doi.org/10.1002/nme.4419, URL https://onlinelibrary.wiley.com/doi/abs/10.1002/nme.4419.

[56] M. Punke, S.M. Wise, A. Voigt, M. Salvalaglio, Explicit temperature coupling in phase-field crystal models of solidification, Modelling Simul. Mater. Sci. Eng. 30 (7) (2022) 74004, http://dx.doi.org/10.1088/1361-651X/ac8abd.

[57] J. Han, D.J. Srolovitz, M. Salvalaglio, Disconnection-mediated migration of interfaces in microstructures: I. continuum model, Acta Mater. 227 (2022) 117178, http://dx.doi.org/10.1016/j.actamat.2021.117178, URL https://www.sciencedirect.com/science/article/pii/S1359645421005589.

[58] M. Kelm, S. Gärttner, C. Bringedal, B. Flemisch, P. Knabner, N. Ray, Comparison study of phase-field and level-set method for three-phase systems including two minerals, Comput. Geosci. 26 (3) (2022) 545–570, http://dx.doi.org/10.1007/s10596-022-10142-w.

[59] J.A. Sethian, P. Smereka, Level set methods for fluid interfaces, Annu. Rev. Fluid Mech. 35 (2003) 341–372, http://dx.doi.org/10.1146/annurev.fluid.35.101101.161105.

[60] Y.-T. Kim, N. Goldenfeld, J. Dantzig, Computation of dendritic microstructures using a level set method, Phys. Rev. E 62 (2) (2000) 2471–2474, http://dx.doi.org/10.1103/PhysRevE.62.2471, URL https://link.aps.org/doi/10.1103/PhysRevE.62.2471.

[61] F. Gibou, R. Fedkiw, S. Osher, A review of level-set methods and some recent applications, J. Comput. Phys. 353 (2018) 82–109, http://dx.doi.org/10.1016/j.jcp.2017.10.006.

[62] M. Sussman, E. Fatemi, Efficient, interface-preserving level set redistancing algorithm and its application to interfacial incompressible fluid flow, SIAM J. Sci. Comput. 20 (4) (1999) 1165–1191, http://dx.doi.org/10.1137/S1064827596298245.

[63] D. Peng, B. Merriman, S. Osher, H. Zhao, M. Kang, A PDE-based fast local level set method, J. Comput. Phys. 155 (2) (1999) 410–438, http://dx.doi.org/10.1006/jcph.1999.6345.

[64] S. Osher, R. Fedkiw, Level Set Methods and Dynamic Implicit Surfaces, Vol. 153, Springer-Verlag New York, Inc., New York, 2003, http://dx.doi.org/10.1007/978-0-387-22746-7, URL http://ndl.ethernet.edu.et/bitstream/123456789/33018/1/9.pdf.pdf.

[65] N. Fiétier, Ö. Demirel, I.F. Sbalzarini, A meshless particle method for Poisson and diffusion problems with discontinuous coefficients and inhomogeneous boundary conditions, SIAM J. Sci. Comput. 35 (6) (2013) 2469–2493, http://dx.doi.org/10.1137/120889290.

[66] P. Incardona, A. Gupta, S. Yaskovets, I.F. Sbalzarini, A c++ library for memory layout and performance portability of scientific applications, in: J. Singer, Y. Elkhatib, D. Blanco Heras, P. Diehl, N. Brown, A. Ilic (Eds.), Euro-Par 2022: Parallel Processing Workshops, Springer Nature, Switzerland, Cham, 2023, pp. 109–120, http://dx.doi.org/10.1007/978-3-031-31209-0_8, URL https://rdcu.be/ddM9E.

[67] N. cuda-c-programming guide®. [link]. URL https://docs.nvidia.com/cuda/cuda-c-programming-guide/index.html#compute-capability-8-x.

[68] P.J. Roache, Code verification by the method of manufactured solutions, J. Fluids Eng. Trans. ASME 124 (1) (2002) 4–10, http://dx.doi.org/10.1115/1.1436090.

[69] C.N. Satterfield, C.K. Colton, W.H. Pitcher, Restricted diffusion in liquids within fine pores, AIChE J. 19 (3) (1973) 628–635, http://dx.doi.org/10.1002/aic.690190332.

[70] A. Bufe, M. Klee, G. Wehinger, T. Turek, G. Brenner, 3D modeling of a catalyst layer with transport pores for Fischer–Tropsch synthesis, Chem.-Ing.-Tech. 89 (10) (2017) 1385–1390, http://dx.doi.org/10.1002/cite.201700066.

[71] Y. Yong, X. Lou, S. Li, C. Yang, X. Yin, Direct simulation of the influence of the pore structure on the diffusion process in porous media, Comput. Math. Appl. 67 (2) (2014) 412–423, http://dx.doi.org/10.1016/j.camwa.2013.08.032.

[72] H. Rusinque, G. Brenner, Mass transport in porous media at the micro- and nanoscale: A novel method to model hindered diffusion, Microporous Mesoporous Mater. 280 (2019) 157–165, http://dx.doi.org/10.1016/j.micromeso.2019.01.037.

[73] J.C. Ferguson, A. Borner, F. Panerai, S. Close, N.N. Mansour, Continuum to rarefied diffusive tortuosity factors in porous media from X-ray microtomography, Comput. Mater. Sci. 203 (2021) (2022) 111030, http://dx.doi.org/10.1016/j.commatsci.2021.111030.

[74] G.I. Taylor, Dispersion of soluble matter in solvent flowing slowly through a tube, Proc. R. Soc. Lond. Ser. A Math. Phys. Sci. 219 (1137) (1953) 186–203, http://dx.doi.org/10.1098/rspa.1953.0139, URL https://royalsocietypublishing.org/doi/abs/10.1098/rspa.1953.0139.

[75] K.A. Akanni, J.W. Evans, I.S. Abramson, Effective transport coefficients in heterogeneous media, Chem. Eng. Sci. 42 (8) (1987) 1945–1954, http://dx.doi.org/10.1016/0009-2509(87)80141-0.

[76] J.J. Blum, G. Lawler, M. Reed, I. Shin, Effect of cytoskeletal geometry on intracellular diffusion, Biophys. J. 56 (5) (1989) 995–1005, http://dx.doi.org/10.1016/S0006-3495(89)82744-4.

[77] D.A.G. Bruggeman, Berechnung verschiedener physikalischer konstanten von heterogenen Substanzen. I. Dielektrizitätskonstanten und Leitfähigkeiten der Mischkörper aus isotropen Substanzen, Ann. Phys. 416 (7) (1935) 636–664, http://dx.doi.org/10.1002/andp.19354160705, URL https://onlinelibrary.wiley.com/doi/abs/10.1002/andp.19354160705.

[78] G.E. Archie, The electrical resistivity log as an aid in determining some reservoir characteristics, Trans. AIME 146 (01) (1942) 54–62, http://dx.doi.org/10.2118/942054-G.

[79] R.J. Millington, Gas diffusion in porous media, Science 130 (3367) (1959) 100–102, http://dx.doi.org/10.1126/science.130.3367.100.b.

[80] B.D. Wood, M. Quintard, S. Whitaker, Calculation of effective diffusivities for biofilms and tissues, Biotechnol. Bioeng. 77 (5) (2002) 495–516, http://dx.doi.org/10.1002/bit.10075.

[81] N. Iversen, B.B. Jørgensen, Diffusion coefficients of sulfate and methane in marine sediments: Influence of porosity, Geochim. Cosmochim. Acta 57 (3) (1993) 571–578, http://dx.doi.org/10.1016/0016-7037(93)90368-7.

[82] C.D. Shackelford, D.E. Daniel, Diffusion in saturated soil. I: Background, J. Geotech. Eng. 117 (3) (1991) 467–484, http://dx.doi.org/10.1061/(ASCE)0733-9410(1991)117:3(467).

[83] F. Degryse, M.J. McLaughlin, Phosphorus diffusion from fertilizer: Visualization, chemical measurements, and modeling, Soil Sci. Am. J. 78 (3) (2014) 832–842, http://dx.doi.org/10.2136/sssaj2013.07.0293, URL https://acsess.onlinelibrary.wiley.com/doi/abs/10.2136/sssaj2013.07.0293.

[84] H.I. Essaid, B.A. Bekins, I.M. Cozzarelli, Organic contaminant transport and fate in the subsurface: Evolution of knowledge and understanding, Water Resour. Res. 51 (7) (2015) 4861–4902, http://dx.doi.org/10.1002/2015WR017121, URL https://agupubs.onlinelibrary.wiley.com/doi/abs/10.1002/2015WR017121.

[85] Bijay-Singh, Yadvinder-Singh, G.S. Sekhon, Fertilizer-N use efficiency and nitrate pollution of groundwater in developing countries, J. Contam. Hydrol. 20 (3–4) (1995) 167–184, http://dx.doi.org/10.1016/0169-7722(95)00067-4.

[86] J. El Khattabi, B. Louche, H. Darwishe, F. Chaaban, E. Carlier, Impact of fertilizer application and agricultural crops on the quality of groundwater in the alluvial aquifer, Northern France, Water Air Soil Pollut. 229 (4) (2018) http://dx.doi.org/10.1007/s11270-018-3767-4.

[87] A.L. Srivastav, Chapter 6 - Chemical fertilizers and pesticides: role in groundwater contamination, in: M.N.V. Prasad (Ed.), Agrochemicals Detection, Treatment and Remediation, Butterworth-Heinemann, 2020, pp. 143–159, http://dx.doi.org/10.1016/B978-0-08-103017-2.00006-4, URL https://www.sciencedirect.com/science/article/pii/B9780081030172000064.

[88] I. Abd-Elaty, L. Pugliese, M. Zelenakova, P. Mesaros, A.E. Shinawi, Simulation-based solutions reducing soil and groundwater contamination from fertilizers in arid and semi-arid regions: Case study the eastern nile delta, egypt, Int. J. Environ. Res. Public Health 17 (24) (2020) 1–18, http://dx.doi.org/10.3390/ijerph17249373.

[89] C. Soulaine, H.A. Tchelepi, Micro-continuum approach for pore-scale simulation of subsurface processes, Transp. Porous Media 113 (3) (2016) 431–456, http://dx.doi.org/10.1007/s11242-016-0701-3.

[90] J.M. Etancelin, P. Moonen, P. Poncet, Improvement of remeshed Lagrangian methods for the simulation of dissolution processes at pore-scale, Adv. Water Resour. 146 (2020) 103780, http://dx.doi.org/10.1016/j.advwatres.2020.103780.

[91] L. Hume, P. Poncet, A velocity–vorticity method for highly viscous 3D flows with application to digital rock physics, J. Comput. Phys. 425 (2021) 109910, http://dx.doi.org/10.1016/j.jcp.2020.109910, URL https://www.sciencedirect.com/science/article/pii/S0021999120306847.

[92] Z. Cao, B. Wei, α-Fe2O3/single-walled carbon nanotube hybrid films as high-performance anodes for rechargeable lithium-ion batteries, J. Power Sources 241 (2013) 330–340, http://dx.doi.org/10.1016/j.jpowsour.2013.04.101, URL https://www.sciencedirect.com/science/article/pii/S037877531300709X.

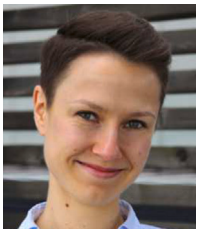
[93] A. Akolkar, N. Rahmatian, S.H. Unterberger, J. Petrasch, Tomography based analysis of conduction anisotropy in fibrous insulation, Int. J. Heat Mass Transfer 108 (2017) 1740–1749, http://dx.doi.org/10.1016/j.ijheatmasstransfer.2016.12.083.

[94] R.S. Dhamrat, J.L. Ellzey, Numerical and experimental study of the conversion of methane to hydrogen in a porous media reactor, Combust. Flame 144 (4) (2006) 698–709.

[95] J.R. Howell, M.J. Hall, J.L. Ellzey, Combustion of hydrocarbon fuels within porous inert media, Prog. Energy Combust. Sci. 22 (2) (1996) 121–145, http://dx.doi.org/10.1016/0360-1285(96)00001-9, URL https://www.sciencedirect.com/science/article/pii/0360128596000019.

[96] B. Van Setten, J. Bremmer, S.J. Jelles, M. Makkee, J.A. Moulijn, Ceramic foam as a potential molten salt oxidation catalyst support in the removal of soot from diesel exhaust gas, Catal. Today 53 (4) (1999) 613–621, http://dx.doi.org/10.1016/S0920-5861(99)00149-2.

[97] S. Haussener, P. Coray, W. Lipiński, P. Wyss, A. Steinfeld, Tomography-based heat and mass transfer characterization of reticulate porous ceramics for high-temperature processing, J. Heat Transfer 132 (2) (2010) 1–9, http://dx.doi.org/10.1115/1.4000226.
[98] Z. Li, M. Dong, Experimental study of diffusive tortuosity of liquid-saturated consolidated porous media, Ind. Eng. Chem. Res. 49 (13) (2010) 6231–6237, http://dx.doi.org/10.1021/ie901765d.
[99] C. Latrille, A. Zoia, Estimating apparent diffusion coefficient and tortuosity in packed sand columns by tracers experiments, J. Porous Media 14 (6) (2011) 507–520, http://dx.doi.org/10.1615/JPorMedia.v14.i6.40.
[100] K.L. Nguyen, V. Wernert, A.M. Lopes, L. Sorbier, R. Denoyel, Effect of tortuosity on diffusion of polystyrenes through chromatographic columns filled with fully porous and porous-shell particles and monoliths, Microporous Mesoporous Mater. 293 (September) (2020) http://dx.doi.org/10.1016/j.micromeso.2019.109776.
[101] S. An, H. Whitney Yu, Z. Wang, B. Kapadia, J. Yao, Unified mesoscopic modeling and GPU-accelerated computational method for image-based pore-scale porous media flows, Int. J. Heat Mass Transfer 115 (2017) 1192–1202, http://dx.doi.org/10.1016/j.ijheatmasstransfer.2017.08.099.
[102] Avizo®. [link]. URL https://www.thermofisher.com/de/de/home/electron-microscopy/products/software-em-3d-vis/avizo-software.html.
[103] Ansys®. [link]. URL www.ansys.com.
[104] COMSOL Multiphysics®. [link]. URL https://www.comsol.com/comsol-multiphysics.
[105] G. Tauriello, P. Koumoutsakos, A comparative study of penalization and phase field methods for the solution of the diffusion equation in complex geometries, J. Comput. Phys. 283 (2015) 388–407, http://dx.doi.org/10.1016/j.jcp.2014.11.033.
[106] Q. Liu, O.V. Vasilyev, A brinkman penalization method for compressible flows in complex geometries, J. Comput. Phys. 227 (2) (2007) 946–966, http://dx.doi.org/10.1016/j.jcp.2007.07.037.
[107] M. Sussman, A level set approach for computing solutions to incompressible two-phase flow, J. Comput. Phys. 114 (1) (1994) 146–159, http://dx.doi.org/10.1006/jcph.1994.1155.
[108] M.A. Fernández-Seara, S.L. Wehrli, F.W. Wehrli, Diffusion of exchangeable water in cortical bone studied by nuclear magnetic resonance, Biophys. J. 82 (1) (2002) 522–529, http://dx.doi.org/10.1016/S0006-3495(02)75417-9.
[109] R. Trampel, J. Schiller, L. Naji, F. Stallmach, J. Kärger, K. Arnold, Self-diffusion of polymers in cartilage as studied by pulsed field gradient NMR, Biophys. Chem. 97 (2–3) (2002) 251–260, http://dx.doi.org/10.1016/S0301-4622(02)00078-9.
[110] F. Travascio, F. Devaux, M. Volz, A.R. Jackson, Molecular and macromolecular diffusion in human meniscus: relationships with tissue structure and composition, Osteoarthr. Cartil. 28 (3) (2020) 375–382, http://dx.doi.org/10.1016/j.joca.2019.12.006.
[111] K. Meyer, O. Ostrenko, G. Bourantas, H. Morales-Navarrete, N. Porat-Shliom, F. Segovia-Miranda, H. Nonaka, A. Ghaemi, J.M. Verbavatz, L. Brusch, I. Sbalzarini, Y. Kalaidzidis, R. Weigert, M. Zerial, A predictive 3D multi-scale model of biliary fluid dynamics in the liver lobule, Cell Syst. 4 (3) (2017) 277–290.e9, http://dx.doi.org/10.1016/j.cels.2017.02.008.
[112] A.E. Ahmed, Highlight report: New insights in liver physiology: Canalicular bile flux is diffusion dominated, EXCLI J. 19 (2020) 1208–1210, http://dx.doi.org/10.17179/excli2020-2836.
[113] Y. Wang, X. Wang, T. Wohland, K. Sampath, Extracellular interactions and ligand degradation shape the nodal morphogen gradient, eLife 5 (APRIL2016) (2016) 1–19, http://dx.doi.org/10.7554/eLife.13879.
[114] P. Müller, K.W. Rogers, S.R. Yu, M. Brand, A.F. Schier, P. Muller, K.W. Rogers, S.R. Yu, M. Brand, A.F. Schier, Morphogen transport, Development 140 (8) (2013) 1621–1638, http://dx.doi.org/10.1242/dev.083519.
[115] D.M. Umulis, H.G. Othmer, The importance of geometry in mathematical models of developing systems, Curr. Opin. Genetics Dev. 22 (6) (2012) 547–552, http://dx.doi.org/10.1016/j.gde.2012.09.007.
[116] A.C. Oates, L.G. Morelli, S. Ares, Patterning embryos with oscillations: Structure, function and dynamics of the vertebrate segmentation clock, Development 139 (4) (2012) 625–639, http://dx.doi.org/10.1242/dev.063735.
[117] Z. Zhang, S. Zwick, E. Loew, J.S. Grimley, S. Ramanathan, Mouse embryo geometry drives formation of robust signaling gradients through receptor localization, Nature Commun. 10 (1) (2019) http://dx.doi.org/10.1038/s41467-019-12533-7.
[118] M.D. Multerer, L.D. Wittwer, A. Stopka, D. Barac, C. Lang, D. Iber, Simulation of morphogen and tissue dynamics, Methods Mol. Biol. 1863 (2018) 223–250, http://dx.doi.org/10.1007/978-1-4939-8772-6_13, arXiv:1806.04138.
[119] S.A. Isaacson, C.S. Peskin, Incorporating diffusion in complex geometries into stochastic chemical kinetics simulations, SIAM J. Sci. Comput. 28 (1) (2006) 47–74, http://dx.doi.org/10.1137/040605060.



Justina Stark is a doctoral candidate in computer science with Ivo Sbalzarini at the Center for Systems Biology Dresden and TU Dresden, Germany. She earned her Master's degree in chemical biology from TU Dortmund, Germany, in 2018. Before joining the International Max Planck Research School, MPI-CBG Dresden, in 2019, she completed lab rotations in the groups of Gene Myers and Florian Jug. In 2022, she had a research stay with Nicolas Beaudoin and Philippe Poncet at the University of Pau, France. Her research interests include modeling reactive transport processes in complex geometries from geology and biology.

Ivo F. Sbalzarini graduated in mechanical engineering from ETH Zurich, Switzerland, and holds a doctorate in computer science from ETH Zurich (with Petros Koumoutsakos; Chorafas Award). In 2006, he became Assistant Professor of Computational Science at ETH Zurich and in 2014 Professor of Computer Science at Technische Universität Dresden, Germany. He is also a Director of the Center for Systems Biology Dresden (CSBD) and a Senior Research Group Leader with the Max Planck Institute of Molecular Cell Biology and Genetics. His research interests include scientific computing, data-driven modeling, and parallel computing, all with applications in multi-scale problems