evoxels: A differentiable physics framework for voxel-based microstructure simulations

Simon Daubner[*1], Alexander E. Cohen[2], Benjamin Dörich[3], and Samuel J. Cooper[1]

[1]Imperial College London, United Kingdom
[2]Massachusetts Institute of Technology, United States
[3]Karlsruhe Institute of Technology, Germany

1 Summary

Materials science inherently spans disciplines: experimentalists use advanced microscopy to uncover micro- and nanoscale structure, while theorists and computational scientists develop models that link processing, structure, and properties. Bridging these domains is essential for inverse material design where you start from desired performance and work backwards to optimal microstructures and manufacturing routes. Integrating high-resolution imaging with predictive simulations and data-driven optimization accelerates discovery and deepens understanding of process–structure–property relationships.

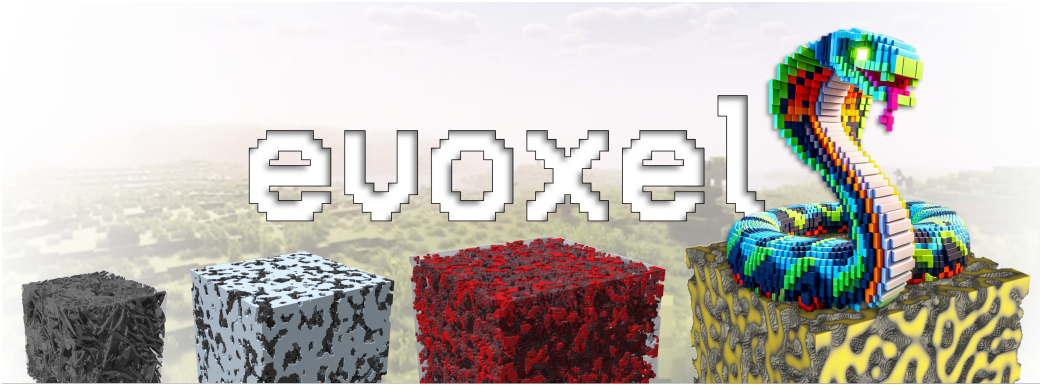

Figure 1: Artwork visualizing the core idea of evoxels: a Python-based differentiable physics framework for simulating and analyzing 3D voxelized microstructures.

The differentiable physics framework **evoxels** is based on a fully Pythonic, unified voxel-based approach that integrates segmented 3D microscopy data, physical simulations, inverse modeling, and machine learning.

- At its core is a voxel grid representation compatible with both pytorch and jax to leverage massive parallelization on CPU, GPU and TPU for large microstructures.
- Both backends naturally provide high computational performance based on just-in-time compiled kernels and end-to-end gradient-based parameter learning through automatic differentiation.

*Corresponding author: s.daubner@imperial.ac.uk

- The solver design based on advanced Fourier spectral time-stepping and low-RAM in-place updates enables scaling to hundreds of millions of DOFs on commodity hardware (e.g. forward Cahn-Hilliard simulation with 400^3 voxels on NVIDIA RTX 500 Ada Laptop GPU with 4GB memory) and billions of DOFs on high end data-center GPUs (1024^3 voxels on NVIDIA RTX A6000; more details see Figure 3).
- Its modular design includes comprehensive convergence tests to ensure the right order of convergence and robustness for various combinations of boundary conditions, grid conventions and stencils during rapid prototyping of new PDEs.

While not intended to replace general finite-element or multi-physics platforms, it fills a unique niche for high-resolution voxel workflows, rapid prototyping for structure simulations and materials design, and fully open, reproducible pipelines that bridge imaging, modeling, and data-driven optimization.

From a high-level perspective, evoxels is organized around two core abstractions: VoxelFields and VoxelGrid. VoxelFields provides a uniform, NumPy-based container for any number of 3D fields on the same regular grid, maximizing interoperability with image I/O libraries (e.g. tifffile, h5py, napari, scikit-image) and visualization tools (PyVista, VTK). VoxelGrid couples these fields to either a PyTorch or JAX backend, offering pre-defined boundary conditions, finite difference stencils and FFT libraries as sketched in Figure 2. The implemented solvers leverage advanced Fourier spectral timesteppers (e.g. semi-implicit, exponential integrators), on-the-fly plotting and integrated wall-time and RAM profiling. A suite of predefined PDE "problems" (e.g. Cahn–Hilliard, reaction-diffusion, multi-phase evolution) can be solved out of the box or extended via user-defined ODEs classes with custom right-hand sides. Integrated convergence tests ensure each discretization achieves the expected order before it ever touches real microscopy data.

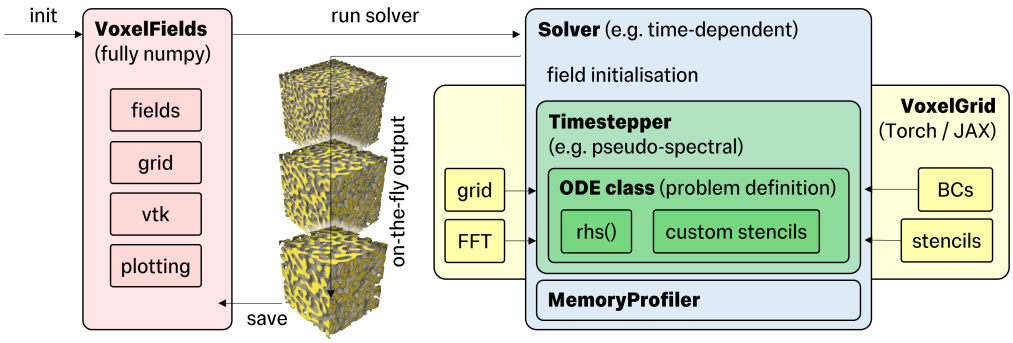

Figure 2: Visualisation of package concept. The VoxelFields class acts as the user interface for organising 3D fields on a regular grid including plotting and export functions. Solvers are assembled in a modular fashion. The chosen timestepper and ODE class are just-in-time compiled (green becomes one kernel) based on the given VoxelGrid backend.

evoxels is aimed squarely at researchers who need a "plug-in-your-image, get-your-answer" workflow for digital materials science and inverse design. Experimentalists can feed segmented FIB-SEM or X-ray tomograms directly into high-performance simulations; computational scientists and modelers benefit from a truly open, reproducible framework. It speaks to anyone who wants special-purpose solvers for representative volume elements - without the overhead of mesh generation - while still offering the flexibility to develop new solvers, test boundary conditions, and incorporate machine-learning-driven optimization. evoxels provides both the turnkey usability of a specialized package and the extensibility of a low-level research toolkit for e.g. benchmarking tortuosity, fitting diffusion coefficients, or prototyping novel phase-field models.

2 Statement of Need

Understanding the link between microstructure and material properties is a central challenge in materials science which increasingly relies on high-resolution 3D imaging, large-scale simulations, and data-driven optimization. Despite the growing availability of segmented volumes from FIB-SEM, X-ray CT, or synchrotron tomography, and data augmentation through generative AI [1, 2] the pipeline from experimental data to simulation remains fragmented. Existing simulation tools rarely operate directly on voxelized microscopy data, instead requiring costly meshing or complex preprocessing. While boundary-conforming meshes (finite element/finite volume method) can better capture complex geometries, voxel-based methods (finite difference and Fourier pseudospectral methods) – especially in combination with smoothed boundary techniques [3, 4] – offer a robust and practical alternative for computing effective material properties. In many materials science applications, small numerical or geometric errors (e.g. 5–10 %) are acceptable, as modeling assumptions are often approximate and the goal is to capture the correct order of magnitude or understand factors like tortuosity or relative transport rates – that is, how much better or worse a given microstructure performs. Furthermore, many commercial codes rely on proprietary data formats, complicating data exchange and reproducibility. In addition to these technical hurdles, significant domain expertise is typically required to configure simulations i.e. choosing appropriate time-stepping schemes, numerical discretizations, and boundary conditions. Even for well-studied problems such as the Cahn–Hilliard equation [5, 6], no scalable 3D Python implementation exists, highlighting a broader lack of open, reusable simulation frameworks in the field. These gaps in data interoperability, code availability, and accessible expertise continue to hinder progress in understanding process-structure-property relationships and limit the practical deployment of inverse design methodologies.

The **evoxels** package enables large-scale forward and inverse simulations on uniform voxel grids, ensuring direct compatibility with microscopy data and harnessing GPU-optimized FFT and tensor operations. This design supports forward modeling of transport and phase evolution phenomena, as well as backpropagation-based inverse problems such as parameter estimation and neural surrogate training - tasks which are still difficult to achieve with traditional FEM-based solvers. This differentiable-physics foundation makes it easy to embed voxel-based solvers as neural-network layers, train generative models for optimal microstructures, or jointly optimize processing and properties via gradient descent. By keeping each simulation step fast and fully backpropagatable, evoxels enables data-driven materials discovery and high-dimensional design-space exploration.

There remains significant untapped potential in applying FFT-based semi-implicit schemes [6] and exponential integrators [7] across the broader landscape of digital materials science. Although these methods are well-established in areas such as spectral homogenization and phase-field modeling, their adoption has largely been limited to specialized research codes. For example in [8], a C++-CUDA implementation of exponential integrators combined with FFT on a GPU was shown to outperform state-of-the-art exponential integrators implementations by fully exploiting the tensor structure of the spatial discretizations. However, few open-source frameworks incorporate these methods into modern simulation pipelines that support automatic differentiation and GPU acceleration—capabilities increasingly critical for inverse design and data-driven workflows.

To evaluate performance against state-of-the-art python libraries, we benchmark the stiff, fourth-order Cahn–Hilliard spinodal-decomposition problem using torchode and Diffrax. As shown in Figure 3, evoxels' native pseudo-spectral IMEX solver achieves runtimes one to two orders of magnitude shorter than general-purpose ODE integrators and requires substantially less GPU memory. By contrast, the TSIT5 integrator with PID-controlled timestepping which is available in both torchode and Diffrax demands finer timesteps, increasing both computation time and memory use to impractical levels for parameter optimization or inverse-design tasks. We also provide a custom Diffrax pseudo-spectral IMEX implementation fully integrated into the evoxels framework; while its wall time matches the native evoxels solver, it incurs higher memory overhead. Finally, fully implicit schemes (e.g., Diffrax's Implicit Euler) exhaust GPU memory on moderate-sized 3D grids (even $< 100^3$), underlining their unsuitability for high-resolution microstructure simulations.

evoxels positions itself as a lightweight, accessible, and rigorously tested tool for prototyping voxel-based PDE solvers. Compared to domain-specific tools like taufactor [10] and magnum.np [11],

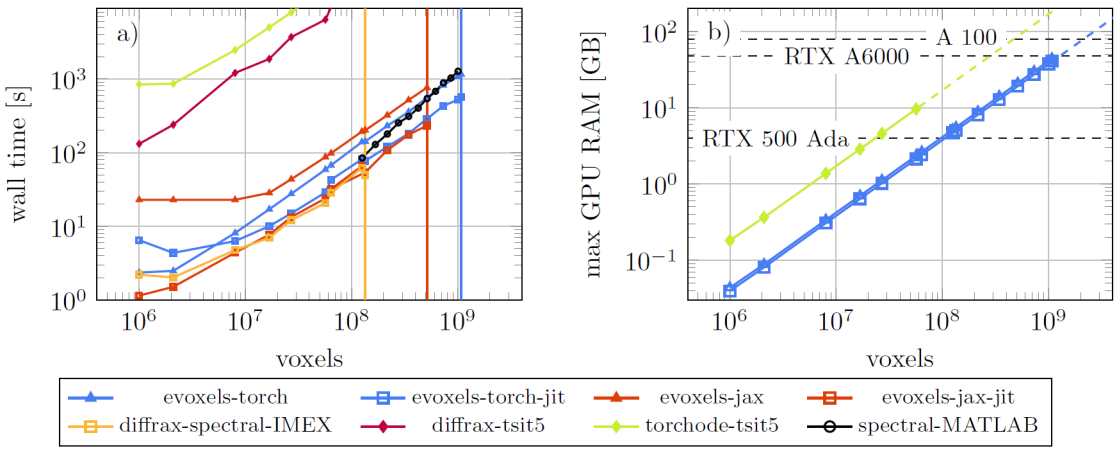

Figure 3: Comparison of wall time and maximum GPU memory usage for the Cahn-Hilliard (CH) problem. Wall time for solving 1000 timesteps with fixed stepsize $\Delta t = 1$ based on pseudo-spectral IMEX scheme with evoxels-torch (blue) and evoxels-jax (red) - both with and without just-in-time (jit) compilation; pseudo spectral IMEX scheme as custom diffrax solver (orange); and tsit5 scheme in combination with a PID timestep controller in torchode (green) and diffrax (purple). Vertical lines denote maximum problem size on Nvidia RTX A6000 for reference. Black datapoints refer to spectral element simulation of CH using MATLAB on Nvidia A100 [9]. GPU memory footprint of all pytorch-based simulations shown in b) shows linear scaling with amount of voxels.

evoxels supports a broader range of problems, boundary conditions, and numerical methods while maintaining a modular, user-friendly interface for imaging-driven workflows. At the same time, it is more specialized and efficient for problems on uniform grids with fixed physics than general-purpose solvers like FiPy or FEniCS. evoxels is not intended to replace multiphysics platforms such as COMSOL or MOOSE, but to complement them by filling a niche in high-resolution, imaging-driven, and differentiable simulations.

Building on prior advances in microstructure characterization [4], phase-field modeling for battery materials [12] and the inverse learning of physics from image data [13], evoxels integrates these capabilities into a unified, extensible codebase. It is currently being used by researchers and students alike to advance inverse-learning capabilities and to develop advanced time integration methods. In a field where open-source simulation tools remain underdeveloped, it provides a practical blueprint for reproducible digital materials science, helping to democratize capabilities that have long been confined to specialist groups or proprietary codebases.

Acknowledgements

We acknowledge computational resources and support provided by the Imperial College Research Computing Service (http://doi.org/10.14469/hpc/2232). This work has received financial support from the European Union's Horizon Europe research and innovation programme under grant agreement No 101069726 (SEATBELT project). Views and opinions expressed are however those of the author(s) only and do not necessarily reflect those of the European Union or CINEA. Neither the European Union nor the granting authority can be held responsible for them.

Appendix: Microstructure modelling

This section provides an insight into the PDEs behind the scenes, i.e. the types of computational problems that are commonly encountered in digital materials science and materials design. These problems can largely be classified into two categories: one set governs the calculation of effective properties in complex heterogeneous structures; the other describes microstructural evolution via phase transformations, or reaction–diffusion mechanisms. In inverse-design scenarios - where one prescribes target properties and then tailors the material's developmental pathways - these two PDE classes become tightly coupled. Understanding and efficiently solving both types is essential for predictive modeling and optimized materials innovation.

Reaction-diffusion systems

The time-dependent evolution of a single species undergoing reaction and diffusion given by Fick's law is described by the PDE

$$\frac{\partial c}{\partial t} = \boldsymbol{\nabla} \cdot (D(x,c)\boldsymbol{\nabla} c) + f(c,t) \tag{1}$$

where the diffusivity D can be a function of space x and/or composition c. Within the more general framework of linear irreversible thermodynamics, the driving force for diffusion is given by the gradient in chemical potential $\nabla\mu$ and with the chemical mobility M, the concentration evolution is then governed by

$$\frac{\partial c}{\partial t} = \boldsymbol{\nabla} \cdot (M(x,c)\boldsymbol{\nabla}\mu(c)) + f(c,t). \tag{2}$$

Note that the chemical potential is a function of concentration as in the case of the Cahn-Hilliard problem.
The Gray-Scott model is an example of two species coupled by an interaction term which can lead to surprisingly complex pattern formation. It can be expressed as two coupled PDEs

$$\frac{\partial c_A}{\partial t} = \boldsymbol{\nabla} \cdot (D_A\boldsymbol{\nabla} c_A) - c_A c_B^2 + f(1-c_A), \tag{3}$$

$$\frac{\partial c_B}{\partial t} = \boldsymbol{\nabla} \cdot (D_B\boldsymbol{\nabla} c_B) + c_A c_B^2 - k c_B. \tag{4}$$

The model describes two species A and B which are dispersed in another medium assuming dilute solution (pairwise diffusion coefficients are equal to zero). c_A is added at a given feed rate f but it can maximally reach a concentration of $c_A = 1$. A reaction converts c_A into c_B in the presence of c_B (i.e. reaction term $c_A c_B^2$). c_B is continuously removed with a given kill rate k until it reaches $c_B = 0$. Both species are also diffusing within the solution with given chemical diffusivities D_A and D_B respectively.

Phase transformations of first and second order

We start from a free energy functional for a two-phase system

$$F_{\text{int}} = \int_V f_{\text{int}} \mathrm{d}V = \gamma_0 \int_V \varepsilon |\boldsymbol{\nabla}\phi|^2 + \frac{9}{\varepsilon}\phi^2(1-\phi)^2 \mathrm{d}V$$

where $\gamma_0\,[\mathrm{J/m^2}]$ denotes the interfacial energy and $\varepsilon\,[\mathrm{m}]$ scales the width of the diffuse interface. Note that in this case, the homogeneous free energy is given by a double-well potential while in the context of the Cahn-Hilliard equation a regular free energy involving logarithmic terms is often employed and, in the context of the multiphase field method, an obstacle-type potential can be found. However, the following procedure would be identical in all cases.

The second order phase transition problem - also known as the *Cahn-Hilliard equation* [5, 14] - can be derived by inserting the chemical potential $\mu = \delta f_{\text{int}}/\delta\phi$ given by the functional derivative of the given free energy formulation into the mass conservation equation which yields

$$\frac{\partial\phi}{\partial t} = \boldsymbol{\nabla}\cdot(M\boldsymbol{\nabla}\mu) + f(\phi,t), \quad \mu = \gamma_0 g(\phi) - 2\gamma_0\varepsilon\boldsymbol{\nabla}^2\phi \tag{5}$$

which is a fourth-order PDE in space. Specifically, we use a variable mobility of the form $M = D_0\phi(1-\phi)$ and the homogenous part of the chemical potential is given by the polynomial expression $g(\phi) = \frac{18}{\varepsilon}\phi(1-\phi)(1-2\phi)$. Additionally, a forcing (source/sink) term f can be considered.
Alternatively, the kinetics of a first-order phase transformation can be derived from the linear relaxation of the system free energy towards its minimum

$$\frac{\partial\phi}{\partial t} = -L\,\frac{\delta f_{\text{int}}}{\delta\phi} = \frac{M}{\varepsilon}\big(2\gamma_0\varepsilon\boldsymbol{\nabla}^2\phi - \gamma_0 g(\phi)\big) \tag{6}$$

also known as the *Allen-Cahn equation* for the non-conserved order parameter ϕ. Note that the kinetic coefficient L is chosen as M/ε such that M represents a physical mobility of the migrating interface which is independent of its diffuse width scaled by ε [15]. Eq. (6) leads to the formation of a diffuse interface and an evolution of the microstructure driven by curvature minimization.
An interesting variation of Eq. (6) is obtained by subtracting the curvature-driven forces from the laplacian of ϕ [16, 17, 18] such that

$$\frac{\partial\phi}{\partial t} = M\gamma_0\bigg(2\bigg(\boldsymbol{\nabla}^2\phi - |\boldsymbol{\nabla}\phi|\boldsymbol{\nabla}\cdot\frac{\boldsymbol{\nabla}\phi}{|\boldsymbol{\nabla}\phi|}\bigg) - \frac{1}{\varepsilon}g(\phi)\bigg).$$

This will create a smooth transition in the direction of the surface normal while the shape is not altered by curvature minimization [16]. Instead of subtracting the curvature the normal part of the laplacian can be computed directly [18]

$$\boldsymbol{\nabla}^2\phi - |\boldsymbol{\nabla}\phi|\boldsymbol{\nabla}\cdot\frac{\boldsymbol{\nabla}\phi}{|\boldsymbol{\nabla}\phi|} = \boldsymbol{\nabla}(\boldsymbol{\nabla}\phi\cdot\boldsymbol{n})\cdot\boldsymbol{n} = \boldsymbol{\nabla}|\boldsymbol{\nabla}\phi|\cdot\frac{\boldsymbol{\nabla}\phi}{|\boldsymbol{\nabla}\phi|}.$$

A multi-phase generalization of Eq. (6) is given by a set of N coupled PDEs [19, 20] given as the sum of pairwise interactions

$$\frac{\partial\phi_\alpha}{\partial t} = -\sum_{\beta\neq\alpha}^{\tilde{N}}\frac{M_{\alpha\beta}}{\varepsilon\tilde{N}}\bigg(\frac{\delta f_{\text{int}}}{\delta\phi_\alpha} - \frac{\delta f_{\text{int}}}{\delta\phi_\beta}\bigg) \tag{7}$$

where $\tilde{N}$ denotes the amount of locally present phases and $M_{\alpha\beta}$ denotes a matrix of pairwise interfacial mobilities.

Smoothed boundary method

Oftentimes, the equations discussed above are confined to a given microstructure like lithium ion transport in the pore space of an electrode microstructure. Following [3, 21], we re-write the diffusion equation (1) with the indicator function ψ as

$$\psi\frac{\partial c}{\partial t} = \boldsymbol{\nabla}\cdot(\psi D\boldsymbol{\nabla}c) + |\boldsymbol{\nabla}\psi|j_N + \psi f(c,t) \tag{8}$$

where j_N is the normal boundary flux. The flux j_N can vary spatially and/or temporally and for a closed system $j_N = 0$ holds. If the microstructure does not evolve over time ($\partial\psi/\partial t = 0$) the equality $\psi\partial c/\partial t = \partial\psi c/\partial t$ holds. Therefore, we can re-formulate the PDE using $\psi c = z$

$$\frac{\partial z}{\partial t} = \boldsymbol{\nabla}\cdot\bigg(D_0\boldsymbol{\nabla}z - D_0\frac{z}{\psi}\boldsymbol{\nabla}\psi\bigg) + |\boldsymbol{\nabla}\psi|j_N + \psi f(c,t) \tag{9}$$

This formulation is beneficial in terms of generality and the use of FFT based semi-implicit timestepping as discussed in the next section.

Semi-implicit timestepping

Consider the (mass / heat) conservation equation (see Eq. (1))

$$\frac{\partial u}{\partial t} = \nabla \cdot (\Gamma(x,u)\nabla u) + f(u,t)$$

with variable mobility (diffusivity/conductivity) Γ and a forcing (sink/source) term f. Following the procedure sketched in [6, 22] we can re-formulate the equation to

$$\frac{\partial u}{\partial t} = \nabla \cdot ((\Gamma_0 + \Gamma(x,u) - \Gamma_0)\nabla u) + f(u,t)$$

to then apply the semi-implicit Fourier spectral method to discretize the PDE in Fourier space

$$\frac{\hat{u}^{n+1} + \hat{u}^n}{\Delta t} = -k^2\Gamma_0\hat{u}^{n+1} + k^2\Gamma_0\hat{u}^n + \mathcal{F}\{\nabla \cdot (\Gamma(x,u)\nabla u) + f(u,t)\}^n$$

where $\mathcal{F}(u) = \hat{u}$ denotes the Fourier transform. This yields the following equation for the new timestep $\hat{u}^{n+1}$

$$\hat{u}^{n+1}(1 + \Delta t k^2\Gamma_0) = \hat{u}^n(1 + \Delta t k^2\Gamma_0) + \Delta t\mathcal{F}\{\nabla \cdot (\Gamma(x,u)\nabla u) + f(u,t)\}^n$$

which corresponds to the following time-stepping scheme in real space

$$u^{n+1} = u^n + \mathcal{F}^{-1}\left\{\frac{\Delta t}{1 + \Delta t k^2\Gamma_0}\mathcal{F}\{\nabla \cdot (\Gamma(x,u)\nabla u) + f(u,t)\}\right\}.$$

Note that this procedure is relatively simple as it only involves coding the finite difference approximation of the original right-hand side of the problem $\text{rhs} = \nabla \cdot (\Gamma\nabla u) + f(u,t)$ as would be common to use any predefined timestepping scheme from existing python packages such as torchode and diffrax. In comparison to the explicit Euler scheme, this procedure additionally involves one forward FFT and one inverse FFT but leads to much larger stable timesteps [6, 22]. Note that as u^n and u^{n+1} both must satisfy the boundary conditions, the update $\mathcal{F}^{-1}\{\dots\}$ itself fulfills certain boundary constraints. Critically, as long as Dirichlet boundary conditions on u are not a function of time the update exhibits constant zero boundary conditions which enables the efficient use of FFT.
This procedure can be equally applied to all reaction-diffusion and phase evolution problems given above, e.g. when applied to the fourth order Cahn-Hilliard problem, this results in

$$\phi^{n+1} = \phi^n + \mathcal{F}^{-1}\left\{\frac{\Delta t}{1 + 2\Delta t\varepsilon M_0 k^4}\mathcal{F}\{\nabla \cdot (M\nabla\mu) + f(\phi,t)\}\right\}$$

and, similarly, the smoothed boundary formulation of the diffusion equation then reads

$$z^{n+1} = z^n + \mathcal{F}^{-1}\left\{\frac{\Delta t}{1 + \Delta t k^2 D_0}\mathcal{F}\left\{\nabla \cdot \left(D\nabla z - D\frac{z}{\psi}\nabla\psi\right) + |\nabla\psi| j_\text{N} + f(z,t)\right\}\right\}.$$

The prefactor inside the inverse Fourier transform always takes the form $\Delta t/(1 - \Delta t\,\mathcal{S})$ where $\mathcal{S}$ is the symbol of the spatial operator i.e. its representation in the Fourier (spectral) domain. For instance, the diffusion operator $D\nabla^2$ corresponds to $-k^2 D$. Within the evoxels framework, the symbol must be defined together with the numerical right-hand side of the PDE for applying pseudo-spectral timesteppers to a given problem definition.

References

[1] Steve Kench and Samuel J. Cooper. Generating three-dimensional structures from a two-dimensional slice with generative adversarial network-based dimensionality expansion. *Nature Machine Intelligence*, 3(4):299–305, 2021.

[2] Donal P. Finegan, Isaac Squires, Amir Dahari, Steve Kench, Katherine L. Jungjohann, and Samuel J. Cooper. Machine-Learning-Driven Advanced Characterization of Battery Electrodes. *ACS Energy Letters*, 7(12):4368–4378, dec 2022.

[3] Hui-Chia Yu, Hsun-Yi Chen, and K Thornton. Extended smoothed boundary method for solving partial differential equations with general boundary conditions on complex boundaries. *Modelling and Simulation in Materials Science and Engineering*, 20(7):075008, oct 2012.

[4] S Daubner and B. Nestler. Microstructure Characterization of Battery Materials Based on Voxelated Image Data: Computation of Active Surface Area and Tortuosity. *Journal of The Electrochemical Society*, 171(12):120514, dec 2024.

[5] J. W. Cahn and J. E. Hilliard. Free Energy of a Nonuniform System. I. Interfacial Free Energy. *The Journal of Chemical Physics*, 28(2):258–267, feb 1958.

[6] Jingzhi Zhu, Long-Qing Chen, Jie Shen, and Veena Tikare. Coarsening kinetics from a variable-mobility Cahn-Hilliard equation: Application of a semi-implicit Fourier spectral method. *Physical Review E*, 60(4):3564–3572, oct 1999.

[7] M. Hochbruck and A. Ostermann. Exponential integrators. *Acta Numerica*, 19:209–286, may 2010.

[8] M. Caliari, F. Cassini, L. Einkemmer, A. Ostermann, and F. Zivcovich. A μ-mode integrator for solving evolution equations in Kronecker form. *J. Comput. Phys.*, 455:Paper No. 110989, 16, 2022.

[9] Xinyu Liu Xinyu Liu, Jie Shen Jie Shen, and Xiangxiong Zhang Xiangxiong Zhang. A Simple GPU Implementation of Spectral-Element Methods for Solving 3D Poisson Type Equations on Rectangular Domains and Its Applications. *Communications in Computational Physics*, 36(5):1157–1185, jan 2024.

[10] Steve Kench, Isaac Squires, and Samuel Cooper. TauFactor 2: A GPU accelerated python tool for microstructural analysis. *Journal of Open Source Software*, 8(88):5358, aug 2023.

[11] Florian Bruckner, Sabri Koraltan, Claas Abert, and Dieter Suess. magnum.np: a PyTorch based GPU enhanced finite difference micromagnetic simulation framework for high level development and inverse design. *Scientific Reports*, 13(1):12054, jul 2023.

[12] Simon Daubner, Marcel Weichel, Martin Reder, Daniel Schneider, Qi Huang, Alexander E Cohen, Martin Z. Bazant, and Britta Nestler. Simulation of intercalation and phase transitions in nano-porous, polycrystalline agglomerates. *npj Computational Materials*, 11(1):211, jul 2025.

[13] Hongbo Zhao, Haitao Dean Deng, Alexander E. Cohen, Jongwoo Lim, Yiyang Li, Dimitrios Fraggedakis, Benben Jiang, Brian D. Storey, William C. Chueh, Richard D. Braatz, and Martin Z. Bazant. Learning heterogeneous reaction kinetics from X-ray videos pixel by pixel. *Nature*, 621(7978):289–294, sep 2023.

[14] J. W. Cahn. On spinodal decomposition. *Acta Metallurgica*, 9:795–801, 1961.

[15] Britta Nestler, Harald Garcke, and Björn Stinner. Multicomponent alloy solidification: Phase-field modeling and simulations. *Physical Review E*, 71(4):041609, apr 2005.

[16] Ying Sun and Christoph Beckermann. Sharp interface tracking using the phase-field equation. *Journal of Computational Physics*, 220(2):626–653, 2007.

[17] Tomohiro Takaki and Junji Kato. Phase-field topology optimization model that removes the curvature effects. *Mechanical Engineering Journal*, 4(2):16–00462–16–00462, 2017.

[18] Ephraim Schoof. *Chemomechanische Modellierung der Wärmebehandlung von Stählen mit der Phasenfeldmethode*. Dissertation, Karlsruhe Institute of Technology (KIT), 2020.

[19] Simon Daubner, Paul W Hoffrogge, Martin Minar, and Britta Nestler. Triple junction benchmark for multiphase-field and multi-order parameter models. *Computational Materials Science*, 219:111995, feb 2023.

[20] P W Hoffrogge, S Daubner, D Schneider, B Nestler, B Zhou, and J Eiken. Triple junction benchmark for multiphase-field models combining capillary and bulk driving forces. *Modelling and Simulation in Materials Science and Engineering*, 33(1):015001, jan 2025.

[21] X. Li, J. Lowengrub, A. Ratz, and A. Voigt. Solving PDEs in complex geometries. *Communications in Mathematical Sciences*, 7(1):81–107, 2009.

[22] L.Q. Chen and Jie Shen. Applications of semi-implicit Fourier-spectral method to phase field equations. *Computer Physics Communications*, 108(2-3):147–158, feb 1998.