The Journal of Open Source Software

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 ¶ Corresponding author

DOI: 10.21105/joss.09733

Software

- Review
- Repository
- Archive

Editor: Philip Cardiff

Reviewers:

- @meyer-nils
- @david-zwicker

Submitted: 15 July 2025
Published: 25 February 2026

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.

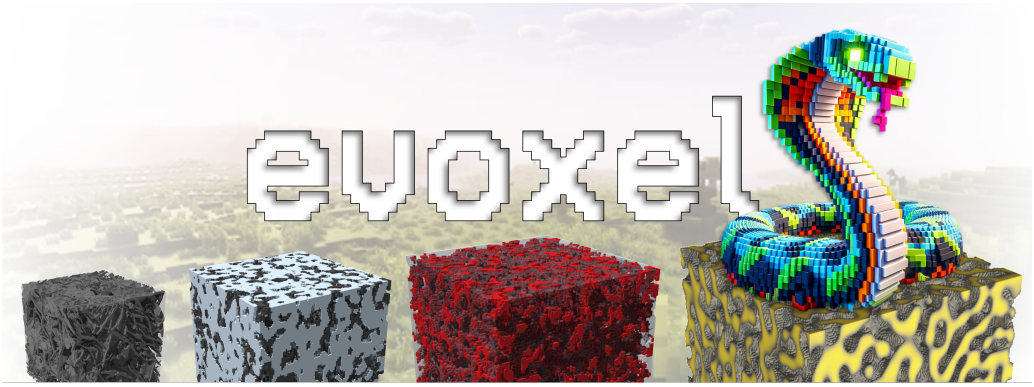

Figure 1: Artwork visualizing the core idea of evoxels: a Python-based differentiable physics framework for simulating and analyzing 3D voxelized microstructures.

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.
- 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 4 GB 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.

At a high level, 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 (napari contributors, 2019), scikit-image (Walt et al., 2014)) and visualization tools (e.g., PyVista (Sullivan & Kaszynski, 2019) for visualization and VTK export). 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 ODE classes with custom right-hand sides. Integrated convergence tests ensure that each discretization achieves the expected order before it is applied to real microscopy data.

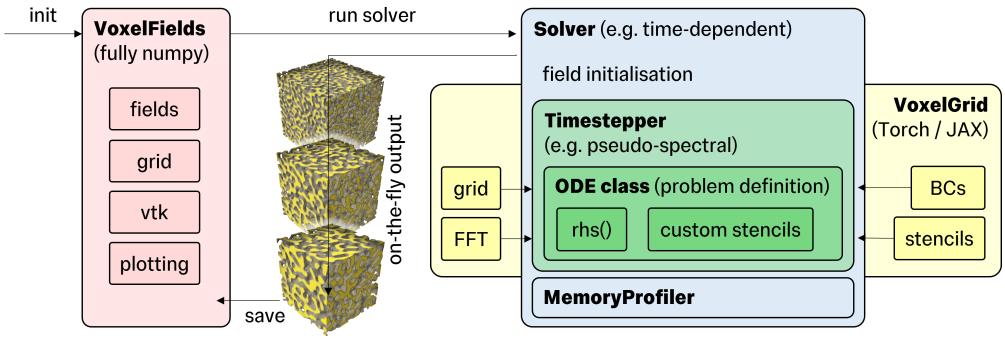

Figure 2: Visualisation of package concept. The VoxelFields class acts as the user interface for organising 3D fields on a regular grid, including plotting and export functions. Solvers are assembled in a modular fashion. The chosen timestepper and ODE class are just-in-time compiled (green becomes a single kernel) based on the specified 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, e.g., benchmarking tortuosity, fitting diffusion coefficients, or prototyping novel phase-field models.

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. Concrete examples include microstructure-resolved simulations on tomographic reconstructions of battery electrodes to quantify structure–transport relationships and guide electrode design (Lu et al., 2020). Another example is the combination of operando imaging with physics-based inversion to infer spatially heterogeneous kinetics from image sequences (Zhao et al., 2023). A further example is phase-field post-processing of 3D grain-label maps from X-ray diffraction microscopy to clean up noisy reconstructions by filling gaps and smoothing the interfaces between neighboring grains (Chlupsa et al., 2024). All of these workflows benefit from scalable voxel-based solvers that operate directly on imaging data and can be embedded in optimization loops. Despite the growing availability of segmented volumes from FIB-SEM, X-ray CT, or synchrotron tomography, and data augmentation through generative AI (Finegan et al., 2022; Kench & Cooper, 2021), 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 pseudo-spectral methods) – especially in combination with smoothed boundary techniques (Daubner & Nestler, 2024; Yu et al., 2012) – 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 (Cahn & Hilliard, 1958; Zhu et al., 1999), few user-friendly and scalable 3D Python implementations combine robustness, GPU acceleration, and suitability for large voxelized domains (Zwicker, 2020). Existing open-source solvers either struggle to scale efficiently to large 3D problems, lack GPU support, or do not support automatic differentiation. The broader lack of open, reusable simulation frameworks is even more pronounced in the field of inverse design and image-based parameter estimation, where many state-of-the-art approaches rely on bespoke, problem-specific codes that are not released or are difficult to reuse and extend (Zhao et al., 2023). 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 enables embedding voxel-based solvers as neural-network layers, training generative models to optimize microstructures, and jointly optimizing 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 (Zhu et al., 1999) and exponential integrators (Hochbruck & Ostermann, 2010) 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 (Caliari et al., 2022), a C++-

CUDA implementation of exponential integrators combined with FFT on a GPU was shown to outperform state-of-the-art exponential integrator 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 (Lienen & Günnemann, 2022) and Diffrax (Kidger, 2021). 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, available in both torchode and Diffrax, requires 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; although its wall time matches that of 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 $< 100^3$), highlighting their unsuitability for high-resolution microstructure simulations.

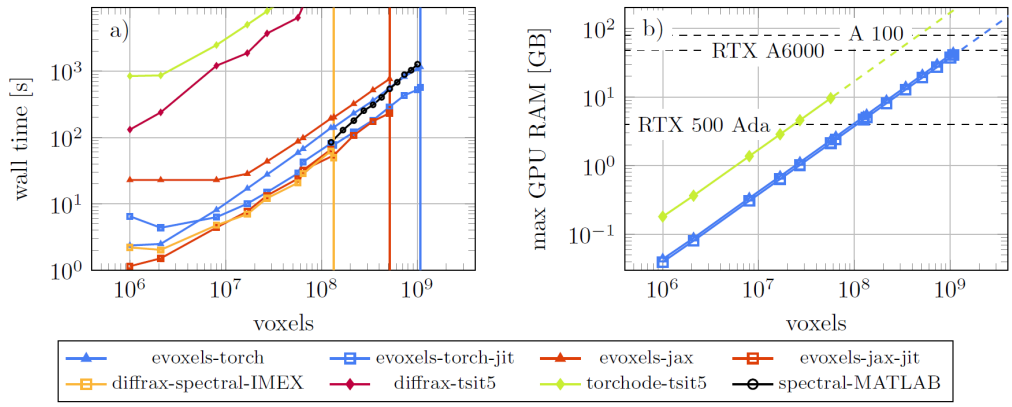

Figure 3: Comparison of wall time and maximum GPU memory usage for the Cahn-Hilliard (CH) problem. Wall time for solving 1000 timesteps with fixed stepsize $\Delta t = 1$ based on pseudo-spectral IMEX scheme with evoxels-torch (blue) and evoxels-jax (red) - both with and without just-in-time (jit) compilation; 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 the maximum problem size on Nvidia RTX A6000 for reference. Black data points refer to spectral element simulations of CH using MATLAB on an NVIDIA A100 (Liu et al., 2024). The GPU memory footprint of all PyTorch-based simulations shown in b) scales linearly with the number of voxels.

The evoxels package is a lightweight, accessible, and rigorously tested tool for prototyping voxel-based PDE solvers, and is compared to domain-specific tools such as taufactor (Kench et al., 2023) and magnum.np (Bruckner et al., 2023), it supports a broader range of problems, boundary conditions, and numerical methods while maintaining a modular, user-friendly interface for imaging-driven workflows. Compared with py-pde (Zwicker, 2020), evoxels places greater emphasis on image-based materials science applications and natively supports differentiable physics via automatic differentiation in PyTorch and JAX, enabling gradient-based inverse problems such as parameter learning and microstructure design. 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 (Daubner & Nestler, 2024), phase-field modeling for battery materials (Daubner et al., 2025), and the inverse learning of physics from image data (Zhao et al., 2023), evoxels integrates these capabilities into a unified, extensible codebase. It is currently used by researchers and students alike to advance inverse learning capabilities and 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.

References

Bruckner, F., Koraltan, S., Abert, C., & Suess, D. (2023). magnum.np: a PyTorch based GPU enhanced finite difference micromagnetic simulation framework for high level development and inverse design. *Scientific Reports*, *13*(1), 12054. https://doi.org/10.1038/s41598-023-39192-5

Cahn, J. W., & Hilliard, J. E. (1958). Free Energy of a Nonuniform System. I. Interfacial Free Energy. *The Journal of Chemical Physics*, *28*(2), 258–267. https://doi.org/10.1063/1.1744102

Caliari, M., Cassini, F., Einkemmer, L., Ostermann, A., & Zivcovich, F. (2022). A μ-mode integrator for solving evolution equations in Kronecker form. *Journal of Computational Physics*, *455*, Paper No. 110989, 16. https://doi.org/10.1016/j.jcp.2022.110989

Chlupsa, M., Croft, Z., Thornton, K., & Shahani, A. J. (2024). Enhancing polycrystalline-microstructure reconstruction from X-ray diffraction microscopy with phase-field post-processing. *Scripta Materialia*, *252*(July), 116228. https://doi.org/10.1016/j.scriptamat.2024.116228

Daubner, S., & Nestler, B. (2024). 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. https://doi.org/10.1149/1945-7111/ad9a07

Daubner, S., Weichel, M., Reder, M., Schneider, D., Huang, Q., Cohen, A. E., Bazant, M. Z., & Nestler, B. (2025). Simulation of intercalation and phase transitions in nano-porous, polycrystalline agglomerates. *Npj Computational Materials*, *11*(1), 211. https://doi.org/10.1038/s41524-025-01707-1

Finegan, D. P., Squires, I., Dahari, A., Kench, S., Jungjohann, K. L., & Cooper, S. J. (2022). Machine-Learning-Driven Advanced Characterization of Battery Electrodes. *ACS Energy Letters*, *7*(12), 4368–4378. https://doi.org/10.1021/acsenergylett.2c01996

Hochbruck, M., & Ostermann, A. (2010). Exponential integrators. *Acta Numerica*, *19*, 209–286. https://doi.org/10.1017/S0962492910000048

Kench, S., & Cooper, S. J. (2021). Generating three-dimensional structures from a two-

dimensional slice with generative adversarial network-based dimensionality expansion. *Nature Machine Intelligence*, *3*(4), 299–305. https://doi.org/10.1038/s42256-021-00322-1

Kench, S., Squires, I., & Cooper, S. (2023). TauFactor 2: A GPU accelerated python tool for microstructural analysis. *Journal of Open Source Software*, *8*(88), 5358. https://doi.org/10.21105/joss.05358

Kidger, P. (2021). *On Neural Differential Equations* [PhD thesis]. University of Oxford.

Lienen, M., & Günnemann, S. (2022). Torchode: A parallel ODE solver for PyTorch. *The Symbiosis of Deep Learning and Differential Equations II, NeurIPS*. https://openreview.net/forum?id=uiKVKTiUYB0

Liu, X., Shen, J., & Zhang, X. (2024). 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. https://doi.org/10.4208/cicp.OA-2024-0072

Lu, X., Bertei, A., Finegan, D. P., Tan, C., Daemi, S. R., Weaving, J. S., O'Regan, K. B., Heenan, T. M. M., Hinds, G., Kendrick, E., Brett, D. J. L., & Shearing, P. R. (2020). 3D microstructure design of lithium-ion battery electrodes assisted by X-ray nano-computed tomography and modelling. *Nature Communications*, *11*(1), 2079. https://doi.org/10.1038/s41467-020-15811-x

napari contributors. (2019). *a multi-dimensional image viewer for python*. https://doi.org/10.5281/zenodo.3555620

Sullivan, B., & Kaszynski, A. (2019). PyVista: 3D plotting and mesh analysis through a streamlined interface for the Visualization Toolkit (VTK). *Journal of Open Source Software*, *4*(37), 1450. https://doi.org/10.21105/joss.01450

Walt, S. van der, Schönberger, J. L., Nunez-Iglesias, J., Boulogne, F., Warner, J. D., Yager, N., Gouillart, E., Yu, T., & contributors, the scikit-image. (2014). Scikit-image: Image processing in python. *PeerJ*, *2*, e453. https://doi.org/10.7717/peerj.453

Yu, H.-C., Chen, H.-Y., & Thornton, K. (2012). 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. https://doi.org/10.1088/0965-0393/20/7/075008

Zhao, H., Deng, H. D., Cohen, A. E., Lim, J., Li, Y., Fraggedakis, D., Jiang, B., Storey, B. D., Chueh, W. C., Braatz, R. D., & Bazant, M. Z. (2023). Learning heterogeneous reaction kinetics from X-ray videos pixel by pixel. *Nature*, *621*(7978), 289–294. https://doi.org/10.1038/s41586-023-06393-x

Zhu, J., Chen, L.-Q., Shen, J., & Tikare, V. (1999). Coarsening kinetics from a variable-mobility Cahn-Hilliard equation: Application of a semi-implicit Fourier spectral method. *Physical Review E*, *60*(4), 3564–3572. https://doi.org/10.1103/PhysRevE.60.3564

Zwicker, D. (2020). py-pde: A Python package for solving partial differential equations. *Journal of Open Source Software*, *5*(48), 2158. https://doi.org/10.21105/joss.02158