ISSN: 3067-2317
Research Article
World Journal of Artificial Intelligence and Robotics Research

MATWERK: A Web-Based Framework for Physics-Informed Thermal Conductivity Simulation and Interactive Microstructure Analysis

Ioseb Kachiashvili

[1]*Faculty of Informatics and Control Systems, Georgian Technical University, 77, st. Kostava, Tbilisi, 0160, Georgia*
[2]*AI department the Muskhelishvili Institute of Computational Mathematics, Georgian Technical University, Tbilisi, Georgia*

***Corresponding author:** Ioseb Kachiashvili, Faculty of Informatics and Control Systems, Georgian Technical University, 77, st. Kostava, Tbilisi, 0160, Georgia.

Submitted: 10 August 2026 **Accepted:** 17 August 2026 **Published:** 03 September 2026

Citation: *Kachiashvili, I. (2026). MATWERK: A Web-Based Framework for Physics-Informed Thermal Conductivity Simulation and Interactive Microstructure Analysis. Wor Jour of Arti inte and Rob Res, 3(5), 01-03.*

Abstract

MATWERK is a web-based computational framework developed for thermal conductivity simulation and microstructure analysis in heterogeneous materials. The platform integrates FastAPI services, PyTorch-based numerical computation, HDF5 dataset management, and interactive browser-based visualization into a unified scientific environment. The system enables users to upload or generate binary microstructures, execute thermal simulations, visualize heat distribution fields, and evaluate effective conductivity properties through an accessible web interface. The framework additionally incorporates surrogate-model-oriented workflows designed to accelerate computational analysis for research and educational applications. This paper presents the system architecture, implementation methodology, computational workflow, visualization subsystem, and engineering significance of the platform.

Keywords: Web-based Computational Framework, Conductivity Simulation, Microstructure Analysis, Heterogeneous Materials, Numerical Computation, Dataset Management, Visualization.

Introduction

Computational material science increasingly relies on scalable software systems capable of integrating numerical simulation, machine learning, data visualization, and user interaction. Traditional thermal conductivity simulation pipelines are frequently constrained by desktop-oriented workflows, limited accessibility, and fragmented visualization tooling. MATWERK was developed to address these limitations through a browser-accessible framework that combines modern web technologies with scientific computing infrastructure.

The framework focuses on binary microstructure conductivity analysis, where thermal transport properties are estimated from spatially distributed material phases. The system supports interactive microstructure generation, numerical simulation, dataset processing, and visualization within a unified architecture. By combining FastAPI, PyTorch, HDF5-based storage, and interactive frontend components, the platform provides a modular environment suitable for rapid experimentation and educational demonstration [1- 4].

System Architecture

The MATWERK architecture follows a layered design composed of frontend presentation components, RESTful backend services, scientific computation modules, and data management utilities (see Figure 1). The frontend layer is implemented using HTML5, CSS, JavaScript, and Plotly visualization libraries [5]. The backend layer is built on FastAPI and exposes endpoints responsible for simulation execution, file management, and result retrieval.

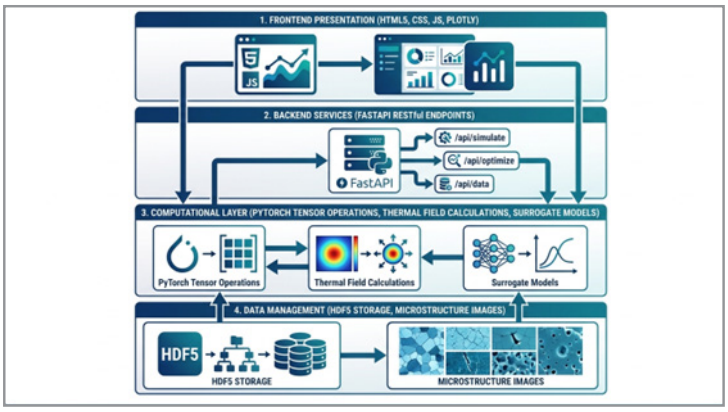

Figure 1: The MATWERK Architecture.

The computational layer integrates PyTorch-based tensor operations for thermal field calculations and surrogate-model-oriented workflows. CUDA (Compute Unified Device Architecture) acceleration is supported when compatible hardware is available. Scientific datasets are managed using HDF5 storage, enabling scalable handling of large collections of microstructure images and conductivity fields. The modular separation between presentation, simulation, and visualization components improves maintainability and enables future integration of additional numerical methods and machine learning models.

Backend Implementation

The backend subsystem is implemented primarily within the main.py module. The application initializes FastAPI services, configures CORS (Cross-Origin Resource Sharing) middleware, manages simulation routing, and coordinates dataset interaction. Device selection logic dynamically enables GPU (Graphics Processing Unit) acceleration through CUDA when available. The backend exposes endpoints for thermal simulation execution, parameter submission, image processing, and conductivity prediction.

The API (Application Programming Interface) structure enables browser-based interaction while maintaining compatibility with external scientific workflows and future distributed deployment strategies. PyTorch integration plays a central role within the computational workflow. Tensor-based numerical representations enable efficient matrix operations and facilitate future integration of Fourier Neural Operator architectures and surrogate modeling techniques.

Data Processing and Utilities

The utilities subsystem provides reusable abstractions for dataset management, tensor preparation, and numerical preprocessing. The utilities module defines classes and helper functions responsible for loading HDF5 datasets, constructing parameter fields, generating coordinate meshes, and preparing simulation inputs. The Microstructure Image Dataset abstraction provides compatibility with PyTorch data pipelines, enabling batch-based processing and machine learning integration. The modular organization of utility functions improves reproducibility and simplifies future extension of the computational workflow. The use of HDF5 storage is particularly important for large-scale scientific computation because it supports structured storage of high-dimensional thermal field data while maintaining efficient input and output performance.

Visualization Framework

Scientific visualization functionality is implemented through both frontend and backend plotting components (see Figure 2). The plotting subsystem uses Matplotlib-based rendering and interactive widget abstractions for thermal field visualization and conductivity analysis. The Thermal Widget component encapsulates interactive plotting behavior and supports dynamic updates of simulation fields, temperature distributions, and heat-flow visualizations. Interactive visualization significantly improves interpretability of thermal simulation results and enhances educational usability. Frontend visualization components additionally integrate Plotly-based rendering for browser-native interaction. Responsive layouts and structured CSS (Cascading Style Sheets) styling allow the platform to maintain usability across different display environments.

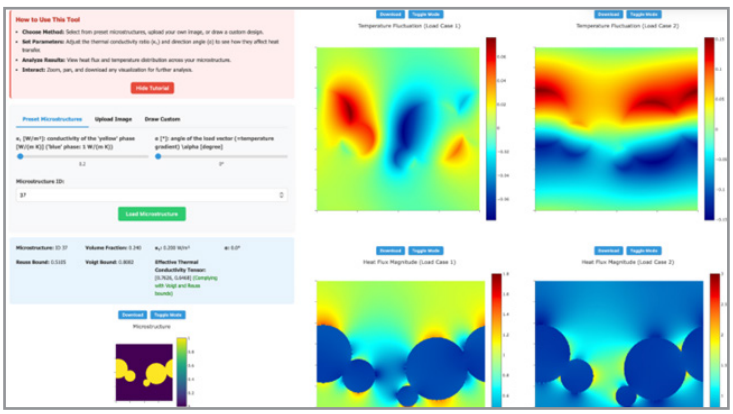

Figure 2: Scientific Visualization of the Results.

Frontend Design

The frontend interface is implemented through the index.html and landing_page.html components. These interfaces provide responsive user interaction workflows for simulation control, dataset exploration, and result visualization (see Figure 2). The frontend design emphasizes accessibility and modularity. Structured layouts organize visualization panels, parameter controls, and simulation outputs into intuitive interaction regions. CSS variable-based styling simplifies customization and improves maintainability. Interactive browser-based simulation environments reduce the installation complexity associated with conventional scientific software and improve accessibility for both educational and research-oriented use cases.

Engineering Significance

MATWERK demonstrates how modern API-oriented software engineering practices can be integrated with computational material science workflows. The platform bridges the gap between scientific computation and web-native interaction models, enabling lightweight deployment and broader accessibility. The integration of machine-learning-compatible data pipelines and tensor-based numerical computation establishes a foundation for accelerated conductivity prediction and surrogate modeling research.

The architecture additionally supports future extension toward distributed inference, uncertainty quantification, and topology optimization workflows. From an educational perspective, the platform provides an accessible environment for demonstrating thermal transport phenomena, numerical simulation principles, and scientific visualization techniques.

Conclusion

MATWERK provides a modular and extensible framework for thermal conductivity simulation and interactive microstructure analysis. The platform successfully combines scientific computing infrastructure, browser-based interaction, and visualization technologies within a unified architecture. The framework demonstrates the feasibility of integrating FastAPI services, PyTorch-based computation, HDF5 dataset management and interactive visualization into a web-accessible scientific environment. Future development may extend the system toward distributed computation, advanced surrogate modeling, and large-scale material informatics applications.

References

1. Tiangolo, S. (2025). FastAPI documentation. URL: https://fastapi.tiangolo.Com.
2. Contributors, T. (2019). Pytorch documentation. URL: https://pytorch.org/docs/stable/index.html.
3. Hoar, C. (2024). Designing a Creative Coding Course Syllabus (Doctoral dissertation, Zurich University of Applied Sciences).
4. The HDF Group. (1998). HDF5 documentation. https://support.hdfgroup.org/documentation/.
5. Plotly Technologies Inc. (2013). Plotly Python documentation. https://plotly.com/python/.

Appendix Selected Implementation Elements

Fast API Backend Initialization

```
app = FastAPI()
app.add_middleware(
   CORSMiddleware,
   allow_origins=["*"],
   allow_credentials=True,
   allow_methods=["*"],
   allow_headers=["*"],)
```

PyTorch Dataset Abstraction

```
class MicrostructureImageDataset(Dataset):
   def __init__(self, h5_file):
      self.h5_file = h5_file
```

Interactive Thermal Visualization

```
class ThermalWidget:
   def __init__(self, ...):
      self.fig, self.ax = plt.subplots()
```


Copyright: *©2026 Ioseb Kachiashvili. This is an open-access article distributed under the terms of the Creative Commons Attribution License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original author and source are credited.*