MaterForge: Materials Formulation Engine with Python

Rahil Miten Doshi[1,2], **Harald Koestler**[1,3], **and Matthias Markl**[2]

1 Chair for System Simulation, Friedrich-Alexander-Universität Erlangen-Nürnberg, Germany **2** Chair of Materials Science and Engineering for Metals, Friedrich-Alexander-Universität Erlangen-Nürnberg, Germany **3** Erlangen National High Performance Computing Center (NHR@FAU), Erlangen, Germany

DOI: 10.xxxxxx/draft

Software

- Review
- Repository
- Archive

Editor: Sarath Menon

Reviewers:

- @jan-janssen
- @JosePizarro3

Submitted: 20 August 2025
Published: unpublished

Summary

MaterForge is an extensible, open-source Python library that streamlines the definition and use of material properties in numerical simulations. The library supports complex material behaviors, from simple constants to experimental data in user-friendly YAML configurations. These are internally converted into symbolic mathematical expressions for scientific computing frameworks. MaterForge supports various material types, provides flexible property definitions, and automatically resolves dependency order for derived properties while detecting cycles. It is designed for high-performance computing (HPC) applications and serves as a bridge between experimental data and numerical simulation.

Statement of Need

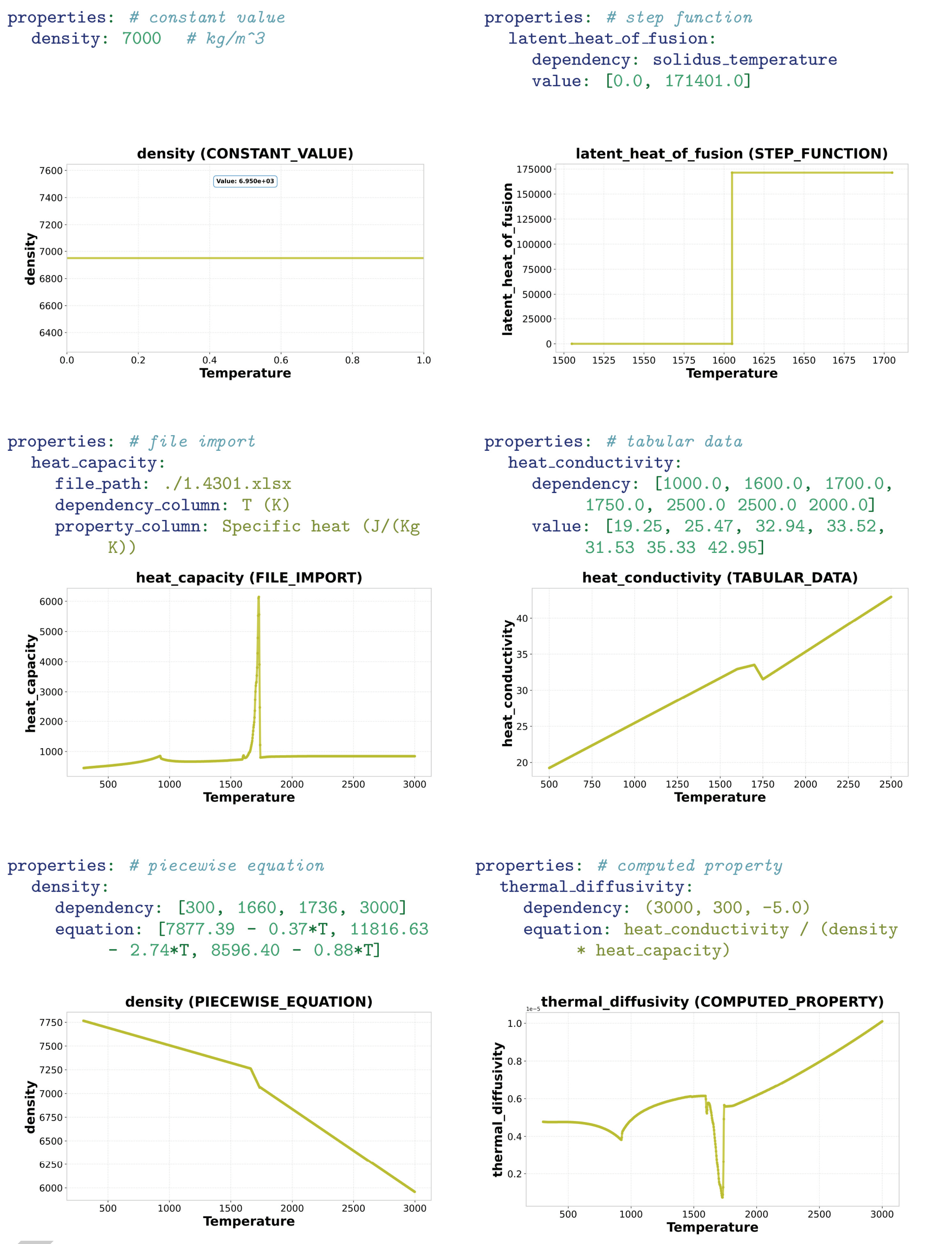
Accurate numerical simulation requires material properties such as thermal conductivity, density, and viscosity that depend on variables like temperature, pressure, or strain rate (Lewis et al., 1996). This challenge is compounded by the wide variation in data availability, from well-characterized models for established materials to sparse experimental points for novel materials. Property definitions consequently range from simple constants to complex tabular datasets or sophisticated equations, creating significant integration hurdles for researchers.

To manage this complexity, researchers often resort to manual interpolation, custom scripting, or proprietary software, which compromises reproducibility and standardization (Ashby & Johnson, 2014). While valuable resources like the NIST WebBook (Linstrom & Mallard, 2001) and CoolProp (Bell et al., 2014) provide valuable raw data, they lack integrated processing to unify these varied formats. CALPHAD databases (Lukas et al., 2007) are powerful but often require proprietary software and do not easily integrate with general-purpose simulation codes.

This leads to ad hoc solutions, hindering workflow efficiency and FAIR data adoption (Wilkinson et al., 2016). MaterForge bridges this gap by providing a unified framework that leverages symbolic mathematics, automatic regression, and dependency resolution to standardize and simplify the integration of realistic material behavior into scientific simulations.

Key Functionality

- **Flexible Input Methods**: The library supports various property definition methods such as constant values, step functions, file-based data (.xlsx, .csv, .txt), tabular data, piecewise equations, and computed properties (Figure 1). This versatility allows users to leverage data from diverse sources.



```
properties: # constant value
  density: 7000   # kg/m^3
```

```
properties: # step function
  latent_heat_of_fusion:
    dependency: solidus_temperature
    value: [0.0, 171401.0]
```

```
properties: # file import
  heat_capacity:
    file_path: ./1.4301.xlsx
    dependency_column: T (K)
    property_column: Specific heat (J/(Kg
         K))
```

```
properties: # tabular data
  heat_conductivity:
    dependency: [1000.0, 1600.0, 1700.0,
         1750.0, 2500.0 2500.0 2000.0]
    value: [19.25, 25.47, 32.94, 33.52,
         31.53 35.33 42.95]
```

```
properties: # piecewise equation
  density:
    dependency: [300, 1660, 1736, 3000]
    equation: [7877.39 - 0.37*T, 11816.63
         - 2.74*T, 8596.40 - 0.88*T]
```

```
properties: # computed property
  thermal_diffusivity:
    dependency: (3000, 300, -5.0)
    equation: heat_conductivity / (density
         * heat_capacity)
```

Figure 1: MaterForge's property definition methods with corresponding YAML examples and automatically generated validation plots.

- **Extensible Material Support**: The framework supports any material type through its extensible architecture. Currently implemented for pure metals and alloys, its modular design allows straightforward extension to materials such as ceramics, polymers, or composites.
- **Automatic Dependency Resolution**: For dependent properties (e.g., density calculated from thermal expansion coefficient), MaterForge automatically determines the correct processing order, resolves mathematical dependencies, and detects circular references.
- **Regression and Data Reduction**: The library performs piecewise regression for large

datasets, simplifying complex property curves into efficient mathematical representations with configurable polynomial degrees and segments, reducing computational overhead while maintaining accuracy.

- **Intelligent Simplification Timing**: MaterForge provides sophisticated control over when data simplification occurs via the `simplify` parameter. `simplify: pre` optimizes performance by simplifying properties before being used in dependent calculations, while `simplify: post` defers simplification until all dependent properties have been computed, maximizing numerical accuracy.
- **Configurable Boundary Behavior**: Users can define how properties behave outside their specified ranges, choosing between `constant`-value or `extrapolation` to best match the physical behavior of the material. The boundary behavior options work seamlessly with the regression capabilities to provide comprehensive data processing control (Figure 2).

```
bounds: [constant, extrapolate]
regression:
  simplify: pre
  degree: 2
  segments: 3
```

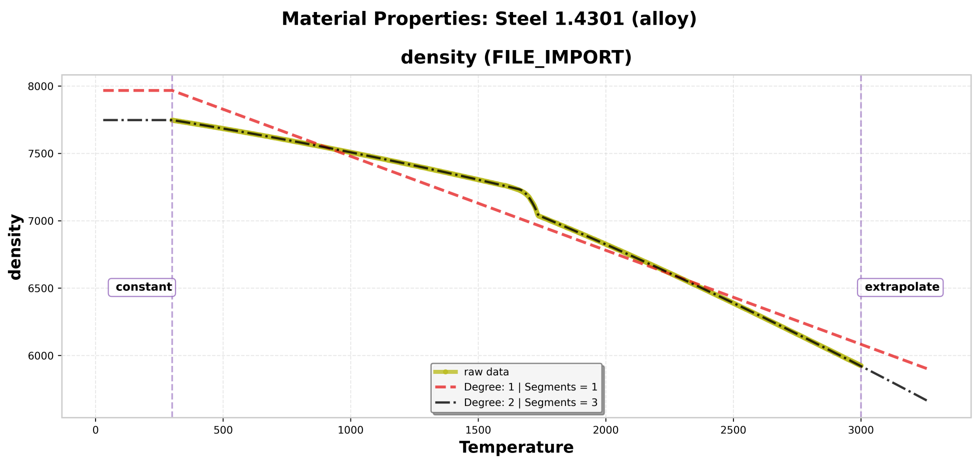


Figure 2: MaterForge's data processing capabilities: regression and data reduction showing raw data (points) fitted with different polynomial degrees and segment configurations, and configurable boundary behavior options demonstrating constant versus extrapolate settings for the same density property, illustrating how MaterForge can reduce complexity while maintaining physical accuracy and providing flexible boundary control.

- **Inverse Property Computation**: The library can generate inverse piecewise-linear functions, enabling the determination of independent variables from known property values. This capability is essential for energy-based numerical methods (Voller & Prakash, 1987), where temperature is computed via the inverse function of the enthalpy.
- **Built-in Validation Framework**: A comprehensive validation framework checks YAML configurations for correctness, including composition sums, required fields, and valid property names, preventing common configuration errors (Roache, 1998).
- **Integrated Visualization**: An integrated visualization tool automatically generates plots to verify property definitions, with the option to disable visualization for production workflows.

Usage

Materials are defined in YAML files and loaded via `create_material`, which returns a fully configured material object.

YAML Configuration Example: Alloy (`steel.yaml`)

```yaml
name: Steel 1.4301
material_type: alloy
composition: {Fe: 0.675, Cr: 0.170, Ni: 0.120, Mo: 0.025, Mn: 0.010}
solidus_temperature: 1605.0
liquidus_temperature: 1735.0
initial_boiling_temperature: 3090.0
final_boiling_temperature: 3200.0
properties:
  density:
    file_path: ./1.4301.xlsx
    dependency_column: T (K)
    property_column: rho (kg/m^3)
    bounds: [constant, extrapolate]
    regression:
      simplify: pre
      degree: 1
      segments: 3
```

Python Integration

```python
import sympy as sp
from materforge.parsing.api import create_material

# Define temperature symbol and load material definition from YAML
T = sp.Symbol('T')
steel = create_material('steel.yaml', T, enable_plotting=True)

# Access symbolic property expressions
density_expr = steel.density

# Evaluate density at 500 K
density_500K = evaluate_material_properties(steel, 500.0, ['density'])
```

Comparison with Existing Tools

Feature	MaterForge	CoolProp	NIST WebBook	CALPHAD Tools
Symbolic Integration	Yes	No	No	Limited
Dependency Resolution	Automatic	No	No	No
Input Methods	6 types	1	1	1
Custom Properties	Any	No	No	Limited
Variable Support	Any	T, P only	Static	T, P, Comp.
Solid Materials	Yes	Limited	Yes	Yes
Python Integration	Native	Yes	API only	Limited
Open Source	Yes	Yes	No	Mixed

Key Advantage: MaterForge's native symbolic mathematics via SymPy (Meurer et al., 2017), automatic dependency resolution, and multiple input methods provide flexibility and integration not found in existing tools, enabling more reproducible and sophisticated scientific simulations.

Research Applications

MaterForge is applicable to alloy design (Callister & Rethwisch, 2018), finite element analysis (Hughes, 2012), multiscale modeling (Tadmor & Miller, 2011), computational fluid dynamics, and heat transfer. Its architecture promotes reproducible science and is well-suited for HPC environments, with demonstrated integrations into frameworks like pystencils (Bauer et al., 2019) and waLBerla (Bauer et al., 2021).

Availability

MaterForge is distributed under the BSD-3-Clause License. The source code is hosted on GitHub, with full documentation and YAML examples. The package can be installed via PyPI using `pip install materforge`.

Acknowledgements

This work was funded by the European High Performance Computing Joint Undertaking (Grant No. 101093457) and the Deutsche Forschungsgemeinschaft within Research Unit FOR-5134 (Grant No. 434946896). We thank Carola Forster for providing the material data for Steel 1.4301 using JMatPro.

References

Ashby, M., & Johnson, K. (Eds.). (2014). Materials and design. In *Materials and design (third edition)* (Third Edition, p. i). Butterworth-Heinemann. https://doi.org/10.1016/B978-0-08-098205-2.00011-1

Bauer, M., Hötzer, J., Ernst, D., Hammer, J., Seiz, M., Hierl, H., Hönig, J., Köstler, H., Nestler, B., & Rüde, U. (2019). Code generation for massively parallel phase-field simulations. *Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis*, 1–12. https://doi.org/10.1145/3295500.3356186

Bauer, M., Köstler, H., & Rüde, U. (2021). waLBerla: A block-structured high-performance framework for multiphysics simulations. *Computers & Mathematics with Applications*, *81*, 478–501. https://doi.org/10.1016/j.camwa.2020.01.007

Bell, I. H., Wronski, J., Quoilin, S., & Lemort, V. (2014). Pure and pseudo-pure fluid thermophysical property evaluation and the open-source thermophysical property library CoolProp. *Industrial & Engineering Chemistry Research*, *53*(6), 2498–2508. https://doi.org/10.1021/ie4033999

Callister, W. D., & Rethwisch, D. G. (2018). *Materials science and engineering: An introduction* (10th ed.). John Wiley & Sons. ISBN: 978-1119405498

Hughes, T. J. R. (2012). *The finite element method: Linear static and dynamic finite element analysis*. Dover Publications. https://doi.org/10.1016/0045-7825(87)90013-2

Lewis, R. W., Morgan, K., Thomas, H. R., & Seetharamu, K. N. (1996). *Finite element analysis of heat transfer and fluid flow*. John Wiley & Sons. ISBN: 978-0471943617

Linstrom, P. J., & Mallard, W. G. (2001). The NIST chemistry WebBook: a chemical data resource on the internet. *Journal of Chemical & Engineering Data*, *46*(5), 1059–1063. https://doi.org/10.1021/je000236i

Lukas, H. L., Fries, S. G., & Sundman, B. (2007). *Computational thermodynamics: The calphad method*. Cambridge University Press. ISBN: 978-0521868112

Meurer, A., Smith, C. P., Paprocki, M., Čertík, O., Kirpichev, S. B., Rocklin, M., Kumar, A., Ivanov, S., Moore, J. K., Singh, S., Rathnayake, T., Vig, S., Granger, B. E., Muller, R. P., Bonazzi, F., Gupta, H., Vats, S., Johansson, F., Pedregosa, F., … Scopatz, A. (2017). SymPy: Symbolic computing in python. *PeerJ Computer Science*, *3*, e103. https://doi.org/10.7717/peerj-cs.103

Roache, P. J. (1998). *Verification and validation in computational science and engineering*. Hermosa Publishers. ISBN: 978-0913478080

Tadmor, E. B., & Miller, R. E. (2011). *Modeling materials: Continuum, atomistic and multiscale techniques*. Cambridge University Press. ISBN: 978-0521856980

Voller, V. R., & Prakash, C. (1987). A fixed grid numerical modelling methodology for convection-diffusion mushy region phase-change problems. *International Journal of Heat and Mass Transfer*, *30*(8), 1709–1719. https://doi.org/10.1016/0017-9310(87)90317-6

Wilkinson, M. D., Dumontier, M., Aalbersberg, Ij. J., Appleton, G., Axton, M., Baak, A., Blomberg, N., Boiten, J.-W., Silva Santos, L. B. da, Bourne, P. E., & others. (2016). The FAIR guiding principles for scientific data management and stewardship. *Scientific Data*, *3*(1), 1–9. https://doi.org/10.1038/sdata.2016.18