













Underworld3: Mathematically Self-Describing Modelling in Python for Desktop, HPC and Cloud

Louis Moresi^{1*}, John Mansour^{2*}, Julian Giordani³, Matt Knepley⁴,
Ben Knight⁵, Juan Carlos Graciosa¹, Thyagarajulu Gollapalli², Neng
Lu¹, and Romain Beucher¹

¹ Research School of Earth Sciences, Australian National University, Canberra, Australia ² School of
Earth, Atmospheric & Environmental Science, Monash University ³ University of Sydney, Sydney,
Australia ⁴ Computer Science and Engineering, University at Buffalo ⁵ Curtin University, Perth, Australia
✉ Corresponding author * These authors contributed equally.

DOI: [10.xxxxxx/draft](https://doi.org/10.xxxxxx/draft)

Software

- [Review](#) 
- [Repository](#) 
- [Archive](#) 

Editor: [Open Journals](#) 

Reviewers:

- [@openjournals](#)

Submitted: 01 January 1970

Published: unpublished

License

Authors of papers retain copyright
and release the work under a
Creative Commons Attribution 4.0
International License ([CC BY 4.0](#))

Summary

Underworld3 is a finite element, geophysical-fluid-dynamics modelling framework designed to be both straightforward to use and highly scalable to peak high-performance computing environments. It implements the Lagrangian-particle finite element methodology outlined in Moresi et al. (2003).

Underworld3 inherits the design patterns of earlier versions of underworld including: (1) A python user interface that is inherently safe for parallel computation. (2) A symbolic interface based on sympy that allows users to construct and simplify combinations of mathematical functions, unknowns and the spatial gradients of unknowns on the fly. (3) Interchangeable Lagrangian, Semi-Lagrangian and Eulerian time derivatives with symbolic representations wrapping the underlying implementation. (4) Fast, robust, parallel numerical solvers based on PETSc (Balay et al., 2024) and petsc4py (Dalcin et al., 2011), (5) Flexible, Lagrangian “particle” swarms for handling transport-dominated unknowns that are fully interchangeable with other data-types and can also be treated as symbolic quantities. (6) Unstructured and adaptive meshing that is fully compatible with the symbolic framework.

The symbolic forms in (2,3) are used to construct a finite element representation using sympy (Meurer et al., 2017) and cython (Behnel et al., 2011). These forms are just-in-time (JIT) compiled as C functions libraries and pointers to these libraries are given to PETSc to describe the finite element weak forms (surface and volume integrals), Jacobian derivatives and boundary conditions.

Users of underworld3 typically develop python scripts within jupyter notebooks and, in this environment, underworld3 provides introspection of its native classes both as python objects as well as mathematical ones. This allows symbolic prototyping and validation of PDE solvers in scripts that can immediately be deployed in a parallel HPC environment.

Statement of need

The problems in global planetary dynamics and tectonics that underworld3 is designed to address have a number of defining characteristics: geomaterials are non-linear, viscoelastic/plastic and have a propensity for strain-dependent softening during deformation; strain localisation is very common as a consequence. Geological structures that we seek to understand are often emergent over the course of loading and are observed in the very-large deformation

limit. Material properties have strong spatial gradients arising from pressure and temperature dependence and jumps of several orders of magnitude resulting from material interfaces.

underworld3 automatically handles much of the complexity of combining the non-linearities in rheology, boundary conditions and time-discretisation, forming their derivatives, and simplifying expressions to generate an efficient, parallel PETSc script. underworld3 provides a textbook-like mathematical experience for users who are confident in understanding physical modelling. A number of equation-system templates are provided for typical geophysical fluid dynamics problems such as Stokes-flow, Navier-Stokes-flow, and Darcy flow which provide both usage and mathematical documentation at run-time.

Mathematical Framework

PETSc provides a template form for the automatic generation of weak forms (see Knepley et al., 2013). The strong-form of the problem is defined through the functional \mathcal{F} that expresses the balance between fluxes, forces, and unknowns:

$$\mathcal{F}(u) \sim \nabla \cdot F(u, \nabla u) - f(u, \nabla u) = 0 \quad (1)$$

The discrete weak form and its Jacobian derivative can be expressed as follows

$$\mathcal{F}(u) \sim \sum_e \epsilon_e^T \left[B^T W f(u^q, \nabla u^q) + \sum_k D_k^T W F^k(u^q, \nabla u^q) \right] = 0 \quad (2)$$

$$\mathcal{F}'(u) \sim \sum_e \epsilon_e^T \begin{bmatrix} B^T & D^T \end{bmatrix} W \begin{bmatrix} \partial f / \partial u & \partial f / \partial \nabla u \\ \partial F / \partial u & \partial F / \partial \nabla u \end{bmatrix} \begin{bmatrix} B^T \\ D^T \end{bmatrix} \epsilon_e \quad (3)$$

The symbolic representation of the strong-form that is encoded in underworld3 is:

$$\underbrace{\left[Du/Dt \right]}_u - \nabla \cdot \underbrace{\left[F(u, \nabla u) \right]}_F - \underbrace{\left[H(\mathbf{x}, t) \right]}_h = 0 \quad (4)$$

This symbolic form (4) contains material / time derivatives of the unknowns which are not present in the PETSc template because, after discretisation, these simplify to produce terms that are combinations of fluxes and flux history terms (which modify F) and forces (which modify h). In underworld3, the user interacts with the time derivatives themselves and sympy combines all the flux-like terms and all the force-like terms just prior to forming the Jacobians and compiling the C functions.

Discussion

The aim of underworld3 is to provide strong support to users in developing sophisticated mathematical models, and provide the ability to interrogate those models during development and at run-time. Underworld3 encodes the mathematical structure of the equations it solves and will display, in a publishable mathematical form, the derivations and simplifications that it makes as it sets up the numerical solution.

Despite this symbolic, interactive layer, underworld3 python scripts are inherently-parallel codes that seamlessly deploy as scripts in a high-performance computing parallel environment with very little performance overhead.

Underworld3 documentation is accessible in a rich, mathematical format within jupyter notebooks for model development and analysis but is also incorporated into the API documentation in the same format.

73 Acknowledgements

74 AuScope provides direct support for the core development team behind the underworld codes
75 and the underworld cloud suite of tools. AuScope is funded by the Australian Government
76 through the National Collaborative Research Infrastructure Strategy, NCRIS.

77 The development and testing of our codes is also supported by computational resources
78 provided by the Australian Government through the National Computing Infrastructure (NCI)
79 under the National Computational Merit Allocation Scheme.

80 The Australian Research Council (ARC) supported the development of novel algorithms,
81 computational methods and applications under the Discovery Project and Linkage Project
82 programs. AuScope funding was used to make these methods widely and freely available
83 in the underworld codes. Direct support for Underworld was provided by ARC Industrial
84 Transformation Research Hub Program (The Basin Genesis Hub)

85 References

86 Balay, S., Abhyankar, S., Adams, M., Benson, S., Brown, J., Brune, P., Buschelman, K.,
87 Constantinescu, E., Dalcin, L., Dener, A., Eijkhout, V., Faibussowitsch, J., Gropp, W.,
88 Hapla, V., Isaac, T., Jolivet, P., Karpeev, D., Kaushik, D., Knepley, M., ... Zhang, J.
89 (2024). *PETSc/TAO Users Manual V.3.21* (ANL-21/39-Rev-3.21, 2337606, 188499; pp.
90 ANL-21/39-Rev-3.21, 2337606, 188499). <https://doi.org/10.2172/2337606>

91 Behnel, S., Bradshaw, R., Citro, C., Dalcin, L., Seljebotn, D. S., & Smith, K. (2011). Cython:
92 The best of both worlds. *Computing in Science & Engineering*, 13(2), 31–39.

93 Dalcin, L. D., Paz, R. R., Kler, P. A., & Cosimo, A. (2011). Parallel distributed computing
94 using Python. *Advances in Water Resources*, 34(9), 1124–1139. [https://doi.org/10.1016/](https://doi.org/10.1016/j.advwatres.2011.04.013)
95 [j.advwatres.2011.04.013](https://doi.org/10.1016/j.advwatres.2011.04.013)

96 Knepley, M. G., Brown, J., Rupp, K., & Smith, B. F. (2013). Achieving High Performance
97 with Unified Residual Evaluation. *arXiv:1309.1204 [Cs]*. <https://arxiv.org/abs/1309.1204>

98 Meurer, A., Smith, C. P., Paprocki, M., Čertík, O., Kirpichev, S. B., Rocklin, M., Kumar,
99 A., Ivanov, S., Moore, J. K., Singh, S., Rathnayake, T., Vig, S., Granger, B. E., Muller,
100 R. P., Bonazzi, F., Gupta, H., Vats, S., Johansson, F., Pedregosa, F., ... Scopatz, A.
101 (2017). SymPy: Symbolic computing in Python. *PeerJ Computer Science*, 3, e103.
102 <https://doi.org/10.7717/peerj-cs.103>

103 Moresi, L., Dufour, F., & Mühlhaus, H.-B. (2003). A Lagrangian integration point finite
104 element method for large deformation modeling of viscoelastic geomaterials. *Journal*
105 *of Computational Physics*, 184(2), 476–497. [https://doi.org/10.1016/S0021-9991\(02\)](https://doi.org/10.1016/S0021-9991(02)00031-1)
106 [00031-1](https://doi.org/10.1016/S0021-9991(02)00031-1)