













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Published: unpublished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 (?).

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 (?) 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 (?) 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

Underworld is built around a general, symbolic partial differential equation solver but provides template forms to solve common geophysical fluid dynamics problems such as the Stokes equation for mantle convection, subduction-zone evolution, lithospheric deformation, glacial isostatic adjustment, ice flow; Navier-Stokes equations for finite Prandtl number fluid flow and short-timescale, viscoelastic deformation; and Darcy Flow for porous media problems including groundwater flow and contaminant transport.

These problems 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

The symbolic layer of underworld3 works with the “strong form” of a problem which is typically how the governing equations are derived and disseminated in publications and textbooks. The finite element method is based on a corresponding weak or variational form Zienkiewicz et al. (2013).

PETSc provides a template form for the automatic generation of weak forms (see ?). We start from the strong-form of the problem which is defined through the functional \mathcal{F} that expresses the balance between fluxes ($F(u, \nabla u)$), forces, $f(u, \nabla u)$, and unknowns u :

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

The discrete weak form and its Jacobian derivative would then be expressed as follows:

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

Here ϵ is the element restriction operator; B is the matrix of basis function derivatives and D is the constitutive matrix that, together, describe the relation between the unknowns and the flux. q indicates that the values are determined at a set of quadrature points, and W is a diagonal matrix of weights for these points.

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

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

$$\left[Du/Dt \right] - \nabla \cdot \left[\sigma(u, \nabla u, \mathbf{x}, t) \right] - \left[\mathbf{h}(u, \nabla u, \mathbf{x}, t) \right] = 0 \quad (4)$$

Here \mathbf{H} represents sources and sinks of u , and Du/Dt is the material time derivative of u . The material / time derivatives of the unknowns are not present in the PETSc template but, after time-discretisation, they produce terms that are combinations of fluxes and flux history terms (which combine with σ to contribute to F) and forces (which combine with \mathbf{h} to contribute to f). The explicit time / position dependence in σ is to highlight potential changes to boundary conditions or constitutive properties.

In underworld3, the user interacts with the time derivatives explicitly, and provides strong-form expressions for the template 4. Sympy automatically gathers all the flux-like terms and all the force-like terms into the form required by the PETSc template. All evaluations, derivatives and simplifications of functions in the underworld3 symbolic layer are deferred until final assembly of the PETSc template and the compilation of the C functions.

The main benefits of combining sympy with the PETSc weak form template is a user environment that 1) provides symbolic, mathematical introspection, particularly in the context of Jupyter notebooks; 2) eliminates much of the python or C coding required for complex constitutive models; 3) eliminates any need for users to compute derivatives for the Newton solvers in PETSc.

State of the Field

Underworld3 is one among a small number of specialised codes for studying Earth deformation on medium to long geological time-scales. Early geodynamics codes, of which there were too many to recite individually, were highly specialised for specific tasks. A second generation of codes are built around generic partial differential equation solvers with scriptable interfaces. These include: Aspect (C++ plugin architecture: Heister et al., 2017), Underworld 1 and 2 (xml object composition / python scripting respectively, Mansour et al., 2020; Moresi et al., 2007), Fluidity (xml combined with python scripting, Davies et al., 2011), Milamin (Matlab front end, Dabrowski et al., 2008), LaMEM (julia scripting Kaus et al., 2024), TerraFERMA (Unified Form Language, Wilson et al., 2017), GAdopt (Unified Form Language / python Davies et al., 2022).

Underworld3 uses python and the python package sympy as its scripting interface. The advantage of sympy is that it is a fully featured symbolic algebra package which allows much of the logic of the mathematical problem description to be defined symbolically and dynamically rather than as static objects.

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.

Acknowledgements

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

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

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

References

Reviewer comments

The statement of need could use one sentence at the beginning containing examples of actual applications that Underworld3 is designed to solve. Currently you mention the properties of the models (nonlinear rheology, strain weakening, ...), but applications like mantle convection, lithosphere dynamics, porous flow, crustal deformation would make it much easier to users to understand if the code would work for them.

JOSS requires a section on the state of the field and how Underworld3 fits into it. This is currently missing. From the JOSS guidelines: State of the field: Do the authors describe how this software compares to other commonly-used packages? I think you already describe in depth what makes Underworld special. One additional sentence would be sufficient to mention that there are packages fully based on python and matlab with examples, and fully compiled modeling packages with some examples, and that Underworld bridges the gap (or something similar, the content is up to you, but you need that section/comparison).

The section on the mathematical framework is pretty dense with little introduction and explanation. I know this was probably done to save space and keep a 2-page limit, but in order to be more useful I would suggest to at least:

rephrase the sentence at the beginning to something like this: 'Underworld generates the weak forms of the equations through PETSc, which provides a template form for the automatic generation

make sure in line 51 and 52 to mention which symbols in the equation represent the fluxes (F), forces (f) and unknowns (u).

equations 2 and 3 contain a number of unexplained symbols that would be necessary to spell out to make this useful to the average reader.

add one sentence at the end of the section that explains what the benefit of using this framework is in the context of underworld.

Generally I think the paper is in great shape, and with the small changes above (even if it adds half a page of length or so) should be ready to be published.

Bathe, K.-J. (2008). Finite Element Method. In B. W. Wah (Ed.), *Wiley Encyclopedia of Computer Science and Engineering* (p. ecse159). John Wiley & Sons, Inc. <https://doi.org/10.1002/9780470050118.ecse159>

Behnel, S., Bradshaw, R., Citro, C., Dalcin, L., Seljebotn, D. S., & Smith, K. (2011). Cython: The best of both worlds. *Computing in Science & Engineering*, 13(2), 31–39. <https://doi.org/10.1109/mcse.2010.118>

Dabrowski, M., Krotkiewski, M., & Schmid, D. W. (2008). MILAMIN: MATLAB-based finite element method solver for large problems. *Geochem Geophys Geosyst*, 9(4), 2007GC001719. <https://doi.org/10.1029/2007GC001719>

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

- 163 Davies, D. R., Kramer, S. C., Ghelichkhan, S., & Gibson, A. (2022). Towards automatic finite-
164 element methods for geodynamics via Firedrake. *Geosci. Model Dev.*, 15(13), 5127–5166.
165 <https://doi.org/10.5194/gmd-15-5127-2022>
- 166 Davies, D. R., Wilson, C. R., & Kramer, S. C. (2011). Fluidity: A fully unstructured
167 anisotropic adaptive mesh computational modeling framework for geodynamics: FLUIDITY-
168 MODELING GEODYNAMICAL FLOWS. *Geochem. Geophys. Geosyst.*, 12(6), n/a–n/a.
169 <https://doi.org/10.1029/2011GC003551>
- 170 Heister, T., Dannberg, J., Gassmöller, R., & Bangerth, W. (2017). High accuracy mantle
171 convection simulation through modern numerical methods – II: Realistic models and
172 problems. *Geophysical Journal International*, 210(2), 833–851. <https://doi.org/10.1093/gji/ggx195>
173
- 174 Hughes, T. J. R. (1987). *The finite element method: Linear static and dynamic finite element*
175 *analysis* (1. Dr.). Prentice Hall. ISBN: 978-0-13-317025-2
- 176 Kaus, B., Popov, A., Garrett_Ito, Spang, A., greuber, Ibragimov, I., xinxinwang01, thomasAmor-
177 row, Pusok, A., APiccolo89, ChristianSchuler, Sanan, P., sasbrune, YangJianfeng2015,
178 Bauville, A., & birand-zz. (2024). *LaMEM*. Zenodo. [https://doi.org/10.5281/ZENODO.](https://doi.org/10.5281/ZENODO.10718860)
179 [10718860](https://doi.org/10.5281/ZENODO.10718860)
- 180 Mansour, J., Giordani, J., Moresi, L., Beucher, R., Kaluza, O., Velic, M., Farrington, R.,
181 Quenette, S., & Beall, A. (2020). Underworld2: Python Geodynamics Modelling for
182 Desktop, HPC and Cloud. *JOSS*, 5(47), 1797. <https://doi.org/10.21105/joss.01797>
- 183 Moresi, L., Quenette, S., Lemiale, V., Mériaux, C., Appelbe, B., & Mühlhaus, H.-B. (2007).
184 Computational approaches to studying non-linear dynamics of the crust and mantle. *Physics*
185 *of the Earth and Planetary Interiors*, 163(1), 69–82. [https://doi.org/10.1016/j.pepi.2007.](https://doi.org/10.1016/j.pepi.2007.06.009)
186 [06.009](https://doi.org/10.1016/j.pepi.2007.06.009)
- 187 Wilson, C. R., Spiegelman, M., & Van Keken, P. E. (2017). Terra FERMA : The T ransparent
188 F inite E lement R apid M odel A ssembler for multiphysics problems in E arth sciences.
189 *Geochem Geophys Geosyst*, 18(2), 769–810. <https://doi.org/10.1002/2016GC006702>
- 190 Zienkiewicz, O. C., Taylor, R. L., & Zhu, J. Z. (2013). *The finite element method: Its basis*
191 *and fundamentals* (Seventh edition). Elsevier, Butterworth-Heinemann. ISBN: 978-1-
192 85617-633-0