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

Louis Moresi [1*¶], **John Mansour** [2*], **Julian Giordani** [3], **Matt Knepley** [4], **Ben Knight** [5], **Juan Carlos Graciosa** [1], **Thyagarajulu Gollapalli** [2], **Neng Lu** [1], and **Romain Beucher** [1]

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.

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

Typical problems in geodynamics usually require computing material deformation, damage evolution, and interface tracking in the large-deformation limit. These are typically not well supported by standard engineering finite element simulation codes. Underworld is a python software framework that is intended to solve geodynamics problems that sit at the interface between computational fluid mechanics and solid mechanics (often known as *complex fluids*).

40 It does so by putting Lagrangian and Eulerian variables on an equal footing at both the user
41 and computational levels.

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

48 These problems have a number of defining characteristics: geomaterials are non-linear, vis-
49 coelastic/plastic and have a propensity for strain-dependent softening during deformation;
50 strain localisation is very common as a consequence. Geological structures that we seek to
51 understand are often emergent over the course of loading and are observed in the very-large
52 deformation limit. Material properties have strong spatial gradients arising from pressure and
53 temperature dependence and jumps of several orders of magnitude resulting from material
54 interfaces.

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

Mathematical Framework

63 The symbolic layer of underworld3 works with the "strong form" of a problem which is typically
64 how the governing equations are derived and disseminated in publications and textbooks. The
65 finite element method is based on a corresponding weak or variational form Zienkiewicz et al.
66 (2013).

67 PETSc provides a template form for the automatic generation of weak forms (see Knepley et al.,
68 2013). We start from the strong-form of the problem which is defined through the functional
69 $\mathcal{F}_s$ that expresses the balance between fluxes ($F(u, \nabla u)$), forces, $f(u, \nabla u)$, and unknowns u:

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

70 The discrete weak form and its Jacobian derivative would then be expressed through the related
71 functional $\mathcal{F}_w$ as follows:

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

$$\mathcal{F}_w'(u) \sim \sum_e \epsilon_{e^T} \left[\begin{array}{cc} B^T & D^T \end{array} \right] W \left[\begin{array}{cc} \partial f/\partial u & \partial f/\partial \nabla u \\ \partial F/\partial u & \partial F/\partial \nabla u \end{array} \right] \left[\begin{array}{c} B^T \\ D^T \end{array} \right] \epsilon_e \tag{3}$$

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

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

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

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

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

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

State of the Field

94 Underworld3 is one among a small number of codes for studying Earth deformation on medium
95 to long geological time-scales. Early geodynamics codes, of which there were too many to
96 recite individually, were highly specialised for specific tasks with little flexibility for user-defined
97 problems. A subsequent generation of codes, currently in use, was built around generic partial
98 differential equation solvers with scriptable interfaces.

99 C++

100 These include: Aspect (C++ plugin architecture: Heister et al., 2017), Underworld 1 and 2 (xml
101 object composition / python scripting respectively, Mansour et al., 2020; Moresi et al., 2007),
102 Fluidity (xml combined with python scripting, Davies et al., 2011), Milamin (Matlab front end,
103 Dabrowski et al., 2008), LaMEM (julia scripting Kaus et al., 2024), TerraFERMA (Unified
104 Form Language, Wilson et al., 2017), GAdopt (Unified Form Language / python Davies et al.,
105 2022).

106 Underworld3 uses python and the python package sympy as the scripting interface that overlies
107 the generic partial differential equation layer. The advantage of sympy is that it is a fully
108 featured symbolic algebra package which allows much of the logic of the mathematical problem
109 description to be defined symbolically and dynamically rather than as static relationships
110 between objects. It also provides deep, mathematical introspection when developing and
111 debugging models.

Discussion

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

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

121 Underworld3 documentation is accessible in a rich, mathematical format within jupyter note-
122 books for model development and analysis but is also incorporated into the API documentation

123 in the same format.

Acknowledgements

References

140 Balay, S., Abhyankar, S., Adams, M., Benson, S., Brown, J., Brune, P., Buschelman, K.,
141   Constantinescu, E., Dalcin, L., Dener, A., Eijkhout, V., Faibussowitsch, J., Gropp, W.,
142   Hapla, V., Isaac, T., Jolivet, P., Karpeev, D., Kaushik, D., Knepley, M., … Zhang, J.
143   (2024). *PETSc/TAO Users Manual V.3.21* (ANL–21/39-Rev-3.21, 2337606, 188499; pp.
144   ANL-21/39-Rev-3.21, 2337606, 188499). https://doi.org/10.2172/2337606

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

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

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

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

157 Davies, D. R., Kramer, S. C., Ghelichkhan, S., & Gibson, A. (2022). Towards automatic finite-
158   element methods for geodynamics via Firedrake. *Geosci. Model Dev.*, *15*(13), 5127–5166.
159   https://doi.org/10.5194/gmd-15-5127-2022

160 Davies, D. R., Wilson, C. R., & Kramer, S. C. (2011). Fluidity: A fully unstructured
161   anisotropic adaptive mesh computational modeling framework for geodynamics: FLUIDITY-
162   MODELING GEODYNAMICAL FLOWS. *Geochem. Geophys. Geosyst.*, *12*(6), n/a–n/a.
163   https://doi.org/10.1029/2011GC003551

164 Heister, T., Dannberg, J., Gassmöller, R., & Bangerth, W. (2017). High accuracy mantle
165   convection simulation through modern numerical methods – II: Realistic models and
166   problems. *Geophysical Journal International*, *210*(2), 833–851. https://doi.org/10.1093/
167   gji/ggx195

168 Hughes, T. J. R. (1987). *The finite element method: Linear static and dynamic finite element*
169 *analysis* (1. Dr.). Prentice Hall. https://doi.org/10.1016/0045-7825(87)90013-2

170 Kaus, B., Popov, A., Garrett_Ito, Spang, A., greuber, Ibragimov, I., xinxinwang01, thomasAmor-
171 row, Pusok, A., APiccolo89, ChristianSchuler, Sanan, P., sasbrune, YangJianfeng2015,
172 Bauville, A., & birand-zz. (2024). *LaMEM*. Zenodo. https://doi.org/10.5281/ZENODO.
173 10718860

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

176 Mansour, J., Giordani, J., Moresi, L., Beucher, R., Kaluza, O., Velic, M., Farrington, R.,
177 Quenette, S., & Beall, A. (2020). Underworld2: Python Geodynamics Modelling for
178 Desktop, HPC and Cloud. *JOSS*, *5*(47), 1797. https://doi.org/10.21105/joss.01797

179 Meurer, A., Smith, C. P., Paprocki, M., Čertík, O., Kirpichev, S. B., Rocklin, M., Kumar,
180 A., Ivanov, S., Moore, J. K., Singh, S., Rathnayake, T., Vig, S., Granger, B. E., Muller,
181 R. P., Bonazzi, F., Gupta, H., Vats, S., Johansson, F., Pedregosa, F., … Scopatz, A.
182 (2017). SymPy: Symbolic computing in Python. *PeerJ Computer Science*, *3*, e103.
183 https://doi.org/10.7717/peerj-cs.103

184 Moresi, L., Dufour, F., & Mühlhaus, H.-B. (2003). A Lagrangian integration point finite
185 element method for large deformation modeling of viscoelastic geomaterials. *Journal*
186 *of Computational Physics*, *184*(2), 476–497. https://doi.org/10.1016/S0021-9991(02)
187 00031-1

188 Moresi, L., Quenette, S., Lemiale, V., Mériaux, C., Appelbe, B., & Mühlhaus, H.-B. (2007).
189 Computational approaches to studying non-linear dynamics of the crust and mantle. *Physics*
190 *of the Earth and Planetary Interiors*, *163*(1), 69–82. https://doi.org/10.1016/j.pepi.2007.
191 06.009

192 Wilson, C. R., Spiegelman, M., & Van Keken, P. E. (2017). Terra FERMA : The T ransparent
193 F inite E lement R apid M odel A ssembler for multiphysics problems in E arth sciences.
194 *Geochem Geophys Geosyst*, *18*(2), 769–810. https://doi.org/10.1002/2016GC006702

195 Zienkiewicz, O. C., Taylor, R. L., & Zhu, J. Z. (2013). *The finite element method: Its*
196 *basis and fundamentals* (Seventh edition). Elsevier, Butterworth-Heinemann. https:
197 //doi.org/10.1016/C2009-0-24909-9