RESEARCH ARTICLE | JANUARY 23 2024

Vision for unified micromagnetic modeling (UMM) with Ubermag

Special Collection: 68th Annual Conference on Magnetism and Magnetic Materials

Hans Fangohr ; Martin Lang ; Samuel J. R. Holt ; Swapneel Amit Pathak ; Kauser Zulfiqar ; Marijan Beg

AIP Advances 14, 015138 (2024)
https://doi.org/10.1063/9.0000661

Vision for unified micromagnetic modeling (UMM) with Ubermag

Cite as: AIP Advances 14, 015138 (2024); doi: 10.1063/9.0000661
Submitted: 29 September 2023 • Accepted: 25 November 2023 •
Published Online: 23 January 2024

Hans Fangohr,[1,2,3,a)] Martin Lang,[1,2,3] Samuel J. R. Holt,[1,2] Swapneel Amit Pathak,[1,2] Kauser Zulfiqar,[1,2,4] and Marijan Beg[5]

AFFILIATIONS

[1] Max Planck Institute for the Structure and Dynamics of Matter, Luruper Chaussee 149, 22761 Hamburg, Germany
[2] Center for Free-Electron Laser Science, Luruper Chaussee 149, 22761 Hamburg, Germany
[3] University of Southampton, Southampton SO17 1BJ, United Kingdom
[4] Department of Physics, University of Hamburg, Jungiusstrasse 9, 20355 Hamburg, Germany
[5] Department of Earth Science and Engineering, Imperial College London, London SW7 2AZ, United Kingdom

Note: This paper was presented at the 68th Annual Conference on Magnetism and Magnetic Materials.
[a)]**Author to whom correspondence should be addressed:** hans.fangohr@mpsd.mpg.de

ABSTRACT

Scientists who want to use micromagnetic simulation packages, need to learn how to express the micromagnetic problem of interest in a "language" such as a configuration file, script or GUI-clicks that the simulation software understands. This language varies from software to software. If the researchers need to use another package, they need to learn a new language to re-express their (unchanged) micromagnetic problem for the next software. For research-project specific pre- or post-processing, scientists often need to write additional software. We propose the vision of a *unified micromagnetic modeling* (*UMM*) approach with which researchers can express the micromagnetic problem *once* and from which multiple simulation packages can be instructed *automatically* to carry out the actual numerical problem solving. Furthermore, by providing defined interfaces that are embedded in the Python data science ecosystem and which are used to communicate with the simulation packages, we can create an open research framework in which simulation runs and additional computation can be arbitrarily combined and orchestrated. Where analysis tools are missing from simulators, these can conveniently be created at Python level. Advantages of this approach include reduced effort for scientists to familiarize themselves with multiple simulation configuration languages, easier exploitation of complementary features of the different simulation packages, the ability to compare results computed with different simulation packages more easily, and the option to easily extend analysis functionality of the existing simulators. With recent updates of Ubermag we present a prototype of such a UMM framework. Ubermag provides a unified interface (expressed in Python) to solve micromagnetic problems using, currently, OOMMF and mumax3. After a simulation has finished, the results are made available to the researcher for analysis within the Python ecosystem of scientific and data science libraries. We discuss the current state of capabilities and challenges associated with the proposed approach.

I. MOTIVATION AND INTRODUCTION

Micromagnetic simulations are widely used. As of September 2023, there are more than 3500 citations of OOMMF[1] reported,[2] and the Web of Science Core database lists more than 2800 citations for mumax3.[3] Over time, a significant number of micromagnetic simulation packages have become available. Many of those are published as open source and thus, *in principle*, accessible for the whole community. However, learning how to use each simulation is time-demanding.

Attempting to address this issue, we described[4] in 2021 Ubermag: a human-centered research environment to conduct OOMMF micromagnetic simulations and subsequent analysis from a Python-based framework.

In this manuscript, we report on subsequent progress that makes it now possible to use both OOMMF and mumax3 as micromagnetic calculators within Ubermag. (We refer to micromagnetic simulation packages as *micromagnetic calculators* as they are able to do the calculations needed to solve a micromagnetic problem.) This progress provides evidence that multiple micromagnetic calculators could potentially be directed using the same interface. For scientists using Ubermag, this is a significant step forward. We report on design decisions, benefits and challenges below.

II. UNIFIED MICROMAGNETIC MODELING VISION

The vision for a unified (micromagnetic) modeling is a computational environment from which micromagnetic problems can be defined in a user-friendly and somewhat intuitive way. In this environment existing micromagnetic simulation packages can be used to solve these problems; analysis, interactive and scriptable visualization are available and integrated, the results of simulator A can be used as input for simulator B; and additional analysis tools can be quickly implemented in the same framework if needed. (It is, of course, subjective what is "intuitive", "user-friendly", and "quick".)

Such a framework would enable us to carry out computational research in new ways: perhaps compute the ground state of a micromagnetic system (using one micromagnetic calculator), and to then compute Eigenmodes of that ground state by calling a (Python) library that can exploit the power of a GPU. Or train a neural network within one of the Python-based machine learning environments by calling a micromagnetic simulator of our choice many times to generate suitable training data. If simulation of experimental measurements is implemented in the framework (such as started in Ref. 5), users of all simulation packages can immediately use this to better interpret or plan experimental studies.

In that unified micromagnetic modeling vision, micromagnetic calculators can be used like library functions and researchers can freely mix and match their use with other tailored code (from other libraries or created by themselves).

Outside of magnetism, there are related initiatives such as the atomic simulation environment ASE[6] and the computational material science environment pyiron.[7]

III. UMM IN UBERMAG

The *Ubermag* micromagnetic modeling environment[9] provides functionality that starts to implement such a unified micromagnetic modeling framework. Ubermag uses a domain specific micromagnetic language[10] as the input required from the user and carries out an automatic translation of this micromagnetic problem definition into configuration files for OOMMF[1] and mumax3.[3] Figure 1 illustrates these capabilities schematically. Figure 2 shows an example code snippet demonstrating that the change of a single line is sufficient to choose the micromagnetic calculator.

Not all features of OOMMF and mumax3 are currently accessible from Ubermag (see also discussion in Sec. IV A). An overview of the implemented functionality is available as part of the Ubermag documentation.[11]

A. Framework for computational micromagnetics

Ubermag is embedded in the Python language and uses established data formats (such as NumPy and xarray arrays and pandas tables) that seamlessly connect to a multitude of analysis tools

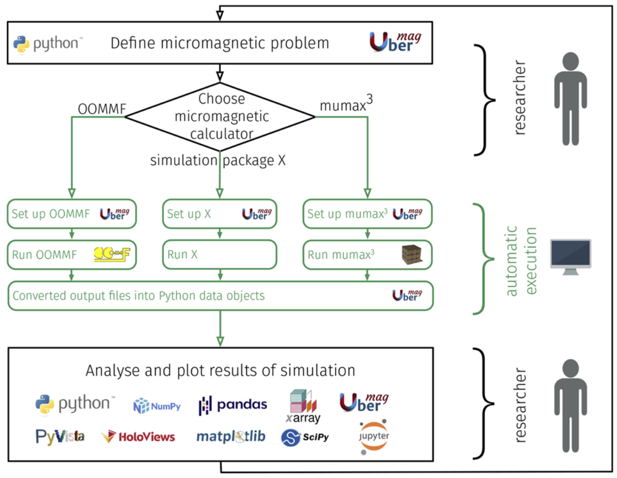

FIG. 1. Unified micromagnetic modeling (UMM) capabilities implemented in Ubermag. The "simulation package X" is a placeholder for additional micromagnetic calculators that could be added in the future.

such as SciPy and scikit-learn along with visualization libraries such as Matplotlib, HoloViews, and PyVista. These libraries and Ubermag itself can be used in Jupyter Notebooks,[12] which are popular in data science and computational science[13] and can improve reproducibility.[4]

B. Data representation, analysis and extensibility

Ubermag provides a unified interface to analyze simulation output independent of the underlying calculator. To work with spatially resolved data such as the magnetization, `discretisedfield` (one of the Ubermag packages) defines a `Field` class. The `Field` class combines the mesh, that is, information about spatial location and discretization of the simulated object, and the vector/scalar data (e.g., magnetization) on that mesh. Data files generated from the individual calculators can automatically be converted into instances of this class. The same class is also used in the initial simulation set up, hence providing a coherent user experience.

The `Field` class offers a wide range of analysis and post-processing functionality. (i) It contains wrappers around commonly used plotting libraries such as Matplotlib and HoloViews to visualize simulation results. (ii) It contains functions to compute commonly required derived quantities such as the topological charge. (iii) The `Field` class implements mathematical operations, ranging from basic operations (such as addition) to vector operations (such as dot products) to integration, derivatives, and fast Fourier transforms (FFTs). These operations can be used to implement further post-processing functionality to compute derived quantities. Where spatial Fourier transforms are computed, the `Field` class provides the position in k-space for every `Field` point, in the same way that it provides the position in real space for real-space data. (iv) The `Field` class is based on NumPy `ndarrays`. Users get direct access to the underlying data and can use it to do further post-processing outside Ubermag using other libraries such as PyTorch. (v) The `Field` class object can be exported into standard formats such as xarray DataArrays.

```
1  # use OOMMF as micromagnetic CALCulator
2  import oommfc as calc
3  # start actual numerical simulation:
4  # 1. minimize energy
5  md = calc.MinDriver()
6  md.drive(system)
7  # 2. time integration
8  td = calc.TimeDriver()
9  td.drive(system, t=0.2e-9)
```

```
# use mumax3 as micromagnetic CALCulator     1
import mumax3c as calc                       2
# start actual numerical simulation:         3
# 1. minimize energy                         4
md = calc.MinDriver()                        5
md.drive(system)                             6
# 2. time integration                        7
td = calc.TimeDriver()                       8
td.drive(system, t=0.2e-9)                   9
```

FIG. 2. Solving a micromagnetic problem using Ubermag. By only changing line 2, OOMMF (left) and mumax3 (right) are used as the micromagnetic calculator to compute the solution. Complete source is shown in the supplementary information.[8]

C. Adding additional micromagnetic calculators

Currently, Ubermag provides a unified interface to the OOMMF and mumax3 calculators. Figure 3 shows the class hierarchy to facilitate the integration of different micromagnetic calculators. Abstract classes contain abstract methods that define function signatures—and hence define the interfaces—but do not provide an implementation. The concrete `TimeDriver` class of the `mumax3c` package inherits from the abstract base class `ExternalDriver` as indicated by the arrow and provides the implementation. The base class `ExternalDriver` has a concrete public `drive` method, which guarantees the unique interface within Ubermag, and which makes use of the abstract methods `_write_inputfiles`, `_call` and `_read_data` to write the required input files for the calculator, to execute the calculator, and to retrieve the output produced by the calculator back into the Python-based Ubermag environment. Within the concrete classes in packages `oommfc` and `mumax3c`, the three methods need to be implemented.

The software design of Ubermag anticipates the addition of further micromagnetic calculators. Calculators based on cuboidal discretization of space are currently easier to integrate than calculators that can operate on less regular meshes due to the central role of the `discretisedfield.Field` object.

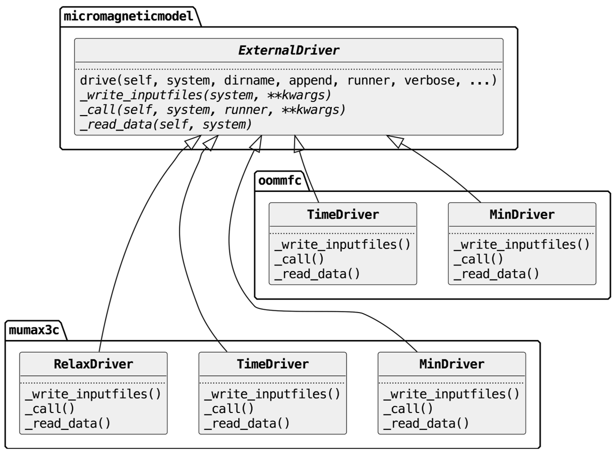

FIG. 3. The abstract *ExternalDriver* class defines the interface. The classes in the `mumax3c` and the `oommfc` package provide the calculator-specific implementations. Abstract classes and methods are shown in italics.

D. Agile development and testing

The Ubermag software is tested through unit, integration and system tests. There are over 4000 tests (based on a parametrization of about 500 tests) executed after every code change. Comprehensive automated testing makes refactoring of the software possible: when adding a new micromagnetic calculator needs changes in the existing design and code, tests are essential to avoid breaking existing functionality. This agile development approach (in contrast to plan-driven development) is appropriate here, in particular in a research environment where new calculators to integrate may not exist yet.

IV. DISCUSSION

A. Unified interface for diverging feature sets

One of the issues we have experienced in our efforts to realize a unified micromagnetic modeling approach in Ubermag is that, on the one hand, we try to design a unified interface (primarily through the machine-readable micromagnetic problem specification). On the other hand, the features of the two micromagnetic calculators are different, so that some capabilities are available in OOMMF but not mumax3, and vice versa.

Examples of deviating capabilities include: mumax3 is limited to one external applied field whereas OOMMF allows to specify multiple external fields that act simultaneously (same for anisotropy). In mumax3 at most 256 different material parameter sets can be used. OOMMF is less flexible in the use of periodic boundary conditions. In mumax3, a DMI term cannot be used without an exchange term.

The design and implementation of the generic description and its translation is time consuming because the software authors need to (i) find a user interface that allows a convenient yet generic description of the desired micromagnetic problem, (ii) understand the behavior and syntax of each calculator and (iii) implement the translation of the generic description into each calculator's syntax in the concrete `Driver` classes. While from the user interface it may appear trivial to change the calculator (see example Fig. 2), the underlying software may be of high complexity to realize this.

It is an open research question how to best address these challenges. Our current approach within Ubermag is to aim to make relatively standard micromagnetic calculations easily possible (ideally for all supported micromagnetic calculators). For more unusual/advanced calculations which, perhaps, make use of a unique capability that is only given in one calculator, it may be necessary that the user provides calculator specific instructions. For example, OOMMF allows specification of external magnetic fields that depend in more arbitrary form on space and time if these are described as a Tcl function. If a researcher needs to use this, they can provide the string embedding the Tcl function to Ubermag when defining the Zeeman field. Ubermag will insert it in the right place in the OOMMF configuration file. While not generic, it enables advanced users to use this advanced feature of OOMMF.

Ubermag can also be used to just write the input file(s) for a simulation package. With this approach the user can subsequently modify the config file manually to use a feature of the calculator that is not supported by Ubermag.

B. Deviating simulation results

The ability of Ubermag to easily compare simulation results from different micromagnetic calculators makes it feasible to routinely check that different calculators produce the same result. In the development of Ubermag, we have conducted many such tests—predominantly with the aim to ensure we have no bugs in the translation of the micromagnetic problem description to the calculator specific instructions.

It is not uncommon to discover discrepancies between the calculators (at the time of writing OOMMF and mumax3). In general, it is difficult to identify the origin of such deviations of two (or more) packages. Possible reasons include the somewhat expected floating point inaccuracies[14] and the non-associative properties of floating point arithmetic that can matter in parallel execution,[14] but also deviating assumptions, the use of different numerical methods, bugs in the calculator implementation or the translator implementation.

Investigating such questions is time consuming. However, having the option to quickly compare the behavior of different calculators offers the community additional transparency and new data points to improve the quality of our collective work.

V. SUMMARY

From the current version of Ubermag we see evidence for the value of a unified micromagnetic modeling environment: (i) Researchers can *choose* which micromagnetic calculator to use depending on the available hardware. For example, if a NVIDIA GPU is available one may want to use mumax3 to maximize execution performance, but if no GPU is available the same problem could be solved using OOMMF. (We note that OOMMF 1 also supports GPU-accelerated computation). (ii) Researchers can *compare and validate* different micromagnetic calculators against each other easily. (iii) Researchers can *combine* unique features of calculator A with unique features of calculator B. (iv) Researchers do *not need to learn* the syntax of each micromagnetic calculator but just the syntax of the UMM problem definition language which, by design, is meant to be accessible to scientists. (v) If a micromagnetic calculator does not provide a required analysis feature, its capabilities can be *extended* by implementing the feature in the Ubermag environment (and then be used with all calculators).

(vi) Some of the software tests (Sec. III D) of Ubermag are ‘system’ or ‘smoke’ tests that ask a micromagnetic calculator to solve a given micromagnetic problem (package `micromagnetictests`), including some of the micromagnetic standard problems or limiting cases where an analytical solution is known. Through the unified interface, the same set of tests can be *re-used* for all micromagnetic calculators. (vii) Where researchers want to create new micromagnetic calculators, they can *re-use* the user interface.

How feasible is the integration of more micromagnetic calculators into a unified micromagnetic modeling framework? We have shown that for two calculators, interoperability of practical value can be achieved. Looking ahead it is possible to include more calculators from a technical point of view (Sec. III D), however, at the cost of increased implementation complexity and maintenance effort. This is conflated by the lack of a business model for such academic open source software: while beneficial for the users of the software, there is little (academic or financial) benefit for the developers within commonly used career metrics and funding systems.

Ubermag does not only make micromagnetic calculators interoperable. It also starts to define standards for machine readable micromagnetic problem definitions, calculations and data. It is increasingly recognized that such (FAIR[15]) standards are important, for example, for research efficiency, data mining, and reproducibility. With such standards, a unified modeling environment can become a *lingua franca* for data-supported research both for experiments and simulation.

While the discussion in this paper is focused on micromagnetic simulators, the creation of a unified modeling approach for atomistic simulations, i.e. extended Heisenberg models, is closely related and would probably offer similar value.

ACKNOWLEDGMENTS

This work was supported by EPSRC UK Skyrmion Grant EP/N032128/1 and OpenDreamKit Horizon 2020 European Research Infrastructure projects (676541).

AUTHOR DECLARATIONS

Conflict of Interest

The authors have no conflicts to disclose.

Author Contributions

Hans Fangohr: Funding acquisition (lead); Investigation (equal); Methodology (lead); Project administration (lead); Software (equal); Supervision (lead); Writing – original draft (lead); Writing – review & editing (equal). **Martin Lang**: Investigation (equal); Software (equal); Writing – review & editing (equal). **Samuel J. R. Holt**: Investigation (equal); Software (equal); Writing – review & editing (equal). **Swapneel Amit Pathak**: Investigation (equal); Software (equal). **Kauser Zulfiqar**: Software (supporting); Writing – review & editing (supporting). **Marijan Beg**: Funding acquisition (supporting); Investigation (equal); Software (equal); Supervision (equal).

DATA AVAILABILITY

The Ubermag software[9] and full version of the Fig. 2 example are available online.[8]

REFERENCES

[1] M. J. Donahue and M. Donahue, Oommf user’s guide, v 1.0, 1999.
[2] https://math.nist.gov/oommf/oommf_cites.html.

[3]A. Vansteenkiste, J. Leliaert, M. Dvornik, M. Helsen *et al.*, "The design and verification of mumax3," AIP Advances **4**, 107133 (2014).
[4]M. Beg *et al.*, "Using Jupyter for reproducible scientific workflows," Computing in Science & Engineering **23**, 36–46 (2021).
[5]S. J. R. Holt, M. Lang, J. C. Loudon, T. J. Hicken, D. Suess, D. Cortés-Ortuño, S. A. Pathak, M. Beg, and H. Fangohr, "Virtual experiments in computational magnetism: mag2exp," under review (2023).
[6]A. Hjorth Larsen *et al.*, "The atomic simulation environment—A Python library for working with atoms," J. of Phys.: Condensed Matter **29**, 273002 (2017).
[7]J. Janssen *et al.*, "pyiron: An integrated development environment for computational materials science," Comp. Materials Science **163**, 24–36 (2019).
[8]Supplementary materials, see https://gitlab.mpcdf.mpg.de/ubermag/paper-2023-unified-micromagnetic-modelling, https://doi.org/10.17617/3.D5FT5Y.
[9]M. Beg, M. Lang, and H. Fangohr, "Ubermag: Toward more effective micromagnetic workflows," IEEE Transactions on Magnetics **58**, 1–5 (2022).
[10]M. Beg, R. A. Pepper, and H. Fangohr, "User interfaces for computational science: A domain specific language for OOMMF embedded in Python," AIP Advances **7**, 056025 (2017).
[11]Supported calculator features in Ubermag, see https://ubermag.github.io/documentation/compatibility.html.
[12]B. E. Granger and F. Perez, "Jupyter: Thinking and storytelling with code and data," Computing in Science & Engineering **23**, 7–14 (2021).
[13]H. Fangohr *et al.*, "Data exploration and analysis with Jupyter notebooks," in *Proceedings of the ICALEPCS'19*, 2020, http://www.jacow.org, Vol. 17 pp. 799–806.
[14]D. Goldberg, "What every computer scientist should know about floating-point arithmetic," ACM Comput. Surv. **23**, 5–48 (1991).
[15]M. D. Wilkinson *et al.*, "The FAIR guiding principles for scientific data management and stewardship," Scientific Data **3**, 160018 (2016).