PAPERS | OCTOBER 01 2024

Numerical simulation projects in micromagnetics with Jupyter

Martin Lonsky; Martin Lang; Samuel Holt; Swapneel Amit Pathak; Robin Klause; Tzu-Hsiang Lo; Marijan Beg; Axel Hoffmann; Hans Fangohr

Am. J. Phys. 92, 794–800 (2024)
https://doi.org/10.1119/5.0149038

Numerical simulation projects in micromagnetics with Jupyter

Martin Lonsky,[1,2,a)] Martin Lang,[3,4,b)] Samuel Holt,[3,4,c)] Swapneel Amit Pathak,[3,4,d)] Robin Klause,[1,e)] Tzu-Hsiang Lo,[1,f)] Marijan Beg,[5,g)] Axel Hoffmann,[1,h)] and Hans Fangohr[3,4,6,i)]

[1]*Materials Research Laboratory and Department of Materials Science and Engineering, University of Illinois Urbana-Champaign, Urbana, Illinois 61801, USA*
[2]*Institute of Physics, Goethe University Frankfurt, 60438 Frankfurt, Germany*
[3]*Max Planck Institute for the Structure and Dynamics of Matter, 22761 Hamburg, Germany*
[4]*Faculty of Engineering and Physical Sciences, University of Southampton, Southampton SO17 1BJ, United Kingdom*
[5]*Department of Earth Science and Engineering, Imperial College London, London SW7 2AZ, United Kingdom*
[6]*Center for Free-Electron Laser Science, 22761 Hamburg, Germany*

(Received 3 March 2023; accepted 15 July 2024)

We report a case study where an existing materials science course was modified to include numerical simulation projects on the micromagnetic behavior of materials. The Ubermag micromagnetic simulation software package is used in order to solve problems computationally. The simulation software is controlled through the Python code in Jupyter notebooks. Our experience is that the self-paced problem-solving nature of the project work can facilitate a better in-depth exploration of the course contents. We discuss which aspects of the Ubermag and the project Jupyter ecosystem have been beneficial for the students' learning experience and which could be transferred to similar teaching activities in other subject areas.
https://doi.org/10.1119/5.0149038

I. INTRODUCTION

Traditionally, science curricula at the university level consist of theory-focused classes and experimental courses. However, not only in the natural sciences but also within the engineering community, computation has emerged as a third fundamental methodology.[1] Both experimentalists and theorists make use of computational techniques in their activities. Oftentimes, a system of interest is too complex to be solved analytically, or certain experiments cannot be carried out in a laboratory. In such cases, numerical studies can help to improve our understanding.

In STEM education, computational modeling has an important role,[2] and it is widely argued that more computational content in curricula would be desirable [e.g., Ref. 3] Anecdotal evidence on undergraduate programs at numerous universities worldwide suggests that computational contents remain severely underrepresented in the respective curricula,[3,4] despite recent studies presenting evidence that computational methods education in undergraduate coursework may lead to the development of multiple essential skills.[5,6] For example, the American Association of Physics Teachers (AAPT) has identified competency in computation to be vital for success at the workplace or PhD research activities for physicists.[7,8] Computational modeling and numerical simulations appear crucial for obtaining a complete picture of the modern STEM disciplines, and therefore, adequate ways need to be found to embed this branch into teaching curricula.

We can distinguish between at least two approaches toward incorporating computational methods in a curriculum:[9] first, there are courses that solely focus on programming, numerical methods, modeling, and simulations. Second, computational content can also be introduced by embedding it in existing courses. The latter approach is at the core of this case study.

Here, we report on the introduction and implementation of numerical simulation group projects in an elective course within the materials science and engineering curriculum of the University of Illinois Urbana-Champaign (UIUC). The course is also available to students in other fields, such as electrical and computer engineering or physics, and requires basic knowledge in condensed matter physics. The students set up and perform micromagnetic simulations by using the open computational environment* Ubermag.[11] In this article, we describe our experiences using Ubermag and related computational software packages in STEM instruction. We discuss our insights from the teaching delivery, student evaluations, and personal interview surveys.

This paper is structured as follows: Section II contains a description of the course on magnetic materials and applications at UIUC, a brief introduction to computational micromagnetics, and a detailed presentation of the Ubermag software and its application in the classroom. Based on the students' feedback and our own experience, we give recommendations for the implementation of computational projects in other courses in Sec. III. We provide supplementary material along with this article, including a general overview on the incorporation of computational contents into STEM programs; a description of simulation projects and the corresponding problem sheets; additional practical considerations; and a thematic analysis of the feedback from students and the teaching staff.

II. MICROMAGNETIC SIMULATION PROJECTS IN A MATERIALS SCIENCE AND ENGINEERING COURSE

There exist many approaches to integrating computational contents into undergraduate or graduate STEM degree curricula (see Sec. I of the supplementary material). Here, we

*Open computational environments allow students to directly see and control the underlying algorithm of the computational model, while closed computational environments such as simulation applets are considered as a black box with no or little information about the exact model.[10]

present a case study that we conducted within the framework of a class on "Magnetic Materials and Applications." We utilized the software package Ubermag,[11] developed at the University of Southampton, United Kingdom, and the Max Planck Institute for the Structure and Dynamics of Matter, Germany, which provides a Python interface to existing micromagnetic simulation packages.

We introduced group projects that make use of the Ubermag software package. Ubermag offers an easy-to-learn approach to create, control, and run simulation scripts that solve the underlying partial differential equation that describes the temporal evolution of the magnetic field in a specified materials system.

In Subsection II A, we begin with a detailed description of the Magnetic Materials and Applications course in which we have conducted our case study. This is followed by an introduction to (analytical) micromagnetic theory in Subsection II B and the numerical computation of solutions in Subsection II C. Finally, Subsection II D introduces the Ubermag software.

A. The elective course on magnetic materials and applications

The Magnetic Materials and Applications class (MSE 598/498/464) at UIUC is an elective course aimed at both undergraduate and graduate students at the Department of Materials Science and Engineering, but other students from the physics, chemical engineering, and electrical engineering departments have also attended this class. The total enrollment ranges from 7 to 15 students per semester.

The lecture introduces the fundamental concepts with regard to the practical use of magnetic materials. The course objectives are:

- Understand how different magnetic interactions determine static and dynamic magnetic properties.
- Quantify essential magnetic materials properties.
- Design components of magnetism-based devices.
- Use basic micromagnetic simulations.

The class is held over the span of about 16 weeks, and it is recommended that students dedicate 6–8 hours per week to working on the course. Aside from the live lectures, online discussions are encouraged via Canvas (an online course and learning management system), weekly homework is assigned, literature review presentations are delivered by the students, and a micromagnetic simulation project has to be completed successfully.

We have designed five distinct simulation projects. Students are asked to work in groups of two to four, since each project is divided into several subprojects, which are mostly independent of each other but do have a certain overlap such that it is beneficial for the students to interact with their peers and discuss their solutions. More detailed information about the contents of the projects and problem sheets can be found in Sec. II of the supplementary material. A sample timeline of the projects is illustrated in Fig. 1. Before the problems are handed out to the students, we give a brief introduction (about 30 min) to Ubermag as part of a standard 90-min class that focuses on micromagnetic simulations. Furthermore, we provide them with additional materials such as video tutorials by the Ubermag developers and the accompanying software documentation, which is very comprehensive and includes numerous examples of Jupyter notebooks that enable to run Ubermag. Due to the students' diverse backgrounds, their exposure to programming in general and Python in particular prior to working on the computational micromagnetics projects has been vastly different. For instance, the Department of Materials Science and Engineering at UIUC has computational methods embedded in several core classes of the curriculum,[12–16] while students from other majors who attend our course may never have written their own code. Furthermore, it is reasonable to assume a discrepancy in the average computational literacy between undergraduate and graduate students in the class.

Project reports are due around two months after the projects have been assigned. It is prudent to set up meetings between students and the instructor together with a teaching assistant halfway through the duration of the computational project. First, this provides preliminary feedback to the students and helps to prevent them from getting lost in detail. Furthermore, it also enables students to ask questions about the subject matter, programming in general, and the instructor's expectation with regard to their report and presentation. Finally, it may be perceived as an intermediate deadline and thereby encourages students to get started with the projects as early as possible. We also ensure that students always have the possibility of reaching out to the teaching assistants via email as well as on a Canvas discussion forum. A few weeks after the intermediate discussions, students are required to present their results to the class and then hand in a project report a few days later. After each presentation, we aim to stimulate a technical discussion and then solicit feedback from the audience on the presentation content and style.

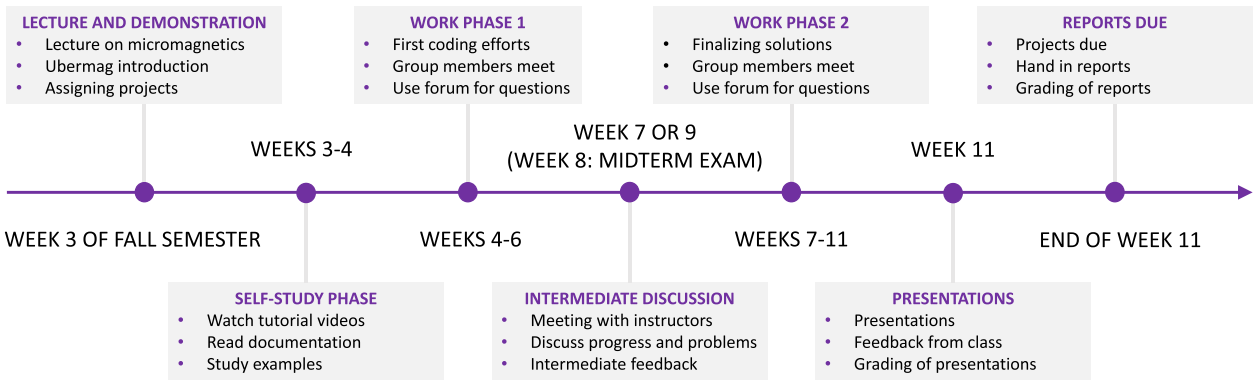

Fig. 1. (Color online) Example timeline for the computational micromagnetics projects.

B. Introduction to micromagnetics

Micromagnetics is concerned with magnetization processes on length scales large enough to overlook atomic structure details of a material but small enough to resolve magnetic textures like domain walls. Examples of relevant applications include magnetic data storage devices and nanoparticles for medical purposes. The basis of time-dependent micromagnetics is the equation of motion of the magnetization vector field (Landau–Lifshitz–Gilbert, LLG equation)[17]

$$\frac{\mathrm{d}\mathbf{m}}{\mathrm{d}t} = -|\gamma_0|\mathbf{m}\times\mathbf{H}_{\mathrm{eff}}(\mathbf{m}) + \alpha\mathbf{m}\times\frac{\mathrm{d}\mathbf{m}}{\mathrm{d}t}. \tag{1}$$

Here, γ_0 denotes the gyromagnetic constant, and α is the Gilbert damping constant. The entity of interest is the magnetization vector field $\mathbf{m}(\mathbf{r},t)\in\mathbb{R}^3$ defined as a function of position $\mathbf{r}\in\mathbb{R}^3$ and time $t\in\mathbb{R}$. For a time-dependent problem, one generally knows an initial magnetization vector field $\mathbf{m}_0=\mathbf{m}(t_0)$ at time t_0 and wants to compute $\mathbf{m}(\mathbf{r},t)$ for $t>t_0$.

A significant part of the complexity originates from the effective field, $\mathbf{H}_{\mathrm{eff}}$, which is itself a function of the magnetization vector field. The effective field can be computed from the energy E of the system,

$$\mathbf{H}_{\mathrm{eff}} = -\frac{1}{\mu_0}\frac{\delta E}{\delta\mathbf{m}}. \tag{2}$$

Different phenomena of material physics can be described by including different contributions to the energy E, for example,

$$\begin{aligned}E(\mathbf{m}) = {}& E_{\mathrm{Ex}}(\mathbf{m}) + E_{\mathrm{Z}}(\mathbf{m}) + E_{\mathrm{Dem}}(\mathbf{m})\\ &+E_{\mathrm{Anis}}(\mathbf{m}) + E_{\mathrm{DMI}}(\mathbf{m}),\end{aligned} \tag{3}$$

where E_{Ex} denotes the exchange energy, E_{Z} is the Zeeman energy, E_{Dem} is the demagnetization energy, E_{Anis} is the anisotropy energy, and E_{DMI} is the Dzyaloshinskii–Moriya interaction (DMI). All energy terms involve integrals over the volume, some involve vector analysis operators, and the demagnetization term contains a double integral over the volume due to the long-range nature of demagnetization effects.

In summary, the micromagnetic problem—summarized through Eqs. (1) and (3)—is complex. Mathematically, this is reflected in Eq. (1) being a non-linear integro-partial differential equation. A rich variety of phenomena are described by this model, ranging from dynamic effects such as ferromagnetic resonance and spin-wave propagation[18–20] to static equilibrium configurations of the magnetization field such as magnetic domains and vortices. It is this complexity and richness that makes the model a fruitful ground for advanced materials physics education.

C. Introduction to computational micromagnetics

The micromagnetic model can only be solved analytically for a small number of cases (often in geometries with particular symmetries). In general, a numerical approach is required to obtain a solution. A typical numerical approach toward the solution of the LLG equation (1) is given by discretizing it in space using finite elements or finite differences. The time-dependent problem then becomes numerically tractable by solving the spatial partial differential equation for a time t, then advancing the solution from t to $t+\Delta t$ through solving a set of ordinary differential equations. We note that this iterative solution over steps in time is algorithmically similar to the time integration method that is frequently used in VPython simulations.

There are at least two widely used software packages that solve the complex computational micromagnetics problem using finite differences and relying on the same physical model: the Object Oriented MicroMagnetic Framework (OOMMF)[21] and mumax^3.[22] The OOMMF software operates on the computer's CPU, whereas mumax^3 is GPU-accelerated and requires an Nvidia GPU card to be installed. OOMMF is written in C++ and Tcl, and mumax^3 is based on the programming languages Go and CUDA. The input scripts for the simulations need to be defined by the user in Tcl and a Go-like scripting language. The learning curve for either package is long; while clearly acceptable in professional research activities, it is a challenge for occasional users such as students in an educational setting. In the remainder of this article, we will demonstrate that the Ubermag software has a significantly shorter learning curve, making it more suitable for educational use. Ubermag has been developed to offer a Python interface[23] to OOMMF with the goal of providing an improved environment for researchers to support computational science investigations of magnetic materials and devices. Later, Ubermag was extended to also interface with mumax^3.[24] In what follows, we will provide more detailed information on Ubermag, and how it can be used for teaching activities.

D. The Ubermag software and its utilization in the classroom

The Python packages provided by Ubermag allow the user to specify micromagnetic models, run simulations, and analyze and visualize data in interactive Jupyter notebooks, see Fig. 1 of the supplementary material. Only the computational solving of micromagnetic problems is delegated to the *micromagnetic calculators* (i.e., OOMMF or mumax^3), while all other steps are independent from these simulation packages.

The Ubermag Python packages (Sec. III C 1) are structured to mirror the computational modeling concepts: define a physics model (`micromagneticmodel`), discretize space (`discretisedfield`), and compute the numerical solution (`oommfc` and `mumax3c`).

Students can control and run their Ubermag simulations via browser-based interactive Jupyter notebooks. The modular structure of Jupyter notebooks allows running blocks of code (so-called "cells") individually instead of running the entire simulation script. Students can obtain an in-depth understanding of the underlying physics by iteratively modifying and exploring the system (Sec. III B 1).

The installation of software for teaching purposes can be challenging: the university's or the students' personal laptops may be running a variety of operating systems (typically Windows, MacOS or Linux) with different versions. More complex simulation software environments may need multiple libraries of compatible versions to be installed simultaneously. For the Ubermag software, there are fortunately multiple ways to install it: using conda-forge, the three main operating systems are supported. An installation using Python's standard installation tool pip is also possible but

requires the user to manually install a micromagnetic calculator (such as OOMMF or mumax[3]).

All simulation projects in our course are carefully designed such that each calculation runs for a reasonably short period of time, i.e., seconds to minutes. For computational problems that can be computed within a few minutes on a single-core CPU, there is another *zero-install* way of using Ubermag through a service called MyBinder available at `mybinder.org`. In short: Ubermag can be executed in the cloud and controlled from any browser; no installation on the computer is necessary. This has been very popular with students (see Sec. III B 2 for more details).

We discuss the value of using open-source software in education in Sec. III of the supplementary material. Furthermore, we present a qualitative thematic analysis of the learning experience from the student and teacher perspective in Sec. IV of the supplementary material. In the following section, we offer suggestions for embedding computational projects into other courses based on the feedback and our experience.

III. RECOMMENDATIONS FOR COMPUTATIONAL PROJECTS

The feedback we have obtained from students and teachers suggests that computational problem solving can improve the learning experience. In this section, we discuss the teaching setup with the objective to extract insights that could be useful in other subjects (i.e., outside micromagnetics and materials physics more generally). We want to comment on three points here: The choice of programming language (Sec. III A), the opportunities from the Jupyter Notebook for use in education (Sec. III B), and aspects of the Ubermag design that are beneficial for teaching (Sec. III C).

A. Choice of programming language

The use of Python as the language to both drive the simulation and to carry out the analysis of the data extracted from the simulations appears to be a good choice. Python is easy to learn yet very powerful.[25] Of particular relevance is the wide set of Python libraries available for science and engineering—including sophisticated data analysis and data visualization tools.

B. Project Jupyter tools for education

The Jupyter notebook[26] emerged from the Interactive Python (IPython)[27] environment. A recent review[28] by the original authors makes the observation that the notebook has been designed to help scientists think. As such, it is perhaps not surprising that the Jupyter notebook has become the standard in data science[29] and is increasingly used in science for data exploration and analysis.[30] Students can benefit in similar ways as data scientists and scientists from the Jupyter notebook, which is increasingly used in educational settings.[31,32]

1. Jupyter notebook

The combination of computer code (as input) and the output from the execution (be it textual, or visualizations, for example) together with equations typeset in LaTeX and free-text in one document helps the thinking process. The notebook captures exactly the protocol that was used (i.e., order of commands for simulation and analysis) to achieve a certain result.[33] The ability to re-execute a command or simulation easily (because the relevant commands are readily available in the document) encourages exploration and verification and, thus, supports a learning process that is driven by experimentation and exploration[28] of the behavior of a complex system.

While we have not seen this done by our students, it is also possible to author a project report within the Jupyter notebook. It is, thus, possible to transition gradually from a set of instructions for the simulation to run and data to be plotted to a report that puts those activities and results in context. We hypothesize that this may lower the barrier toward starting the report writing. Moreover, as demonstrated in previous reports,[34] Jupyter notebooks represent a platform that supports and fosters students' epistemic agency as well as reproducibility of the result.[33]

2. Zero-install software provision with Binder

In our study, we have made use of the publicly accessible and free Binder software,[35] which is part of project Jupyter.[†] The Binder software takes the URL of a data repository,[‡] scans the repository for files that specify which software is required, installs this software—together with a Jupyter notebook server—in a (Docker) container image, starts the container, and connects the notebook server from the container with the user's browser. None of the technical steps described above is visible to the user: After selecting the appropriate URL,[§] it takes a couple of minutes until the desired notebooks session appear in the browser. The major benefit for our teaching experience is that students can connect to a Binder session from their desktop, laptop, or mobile device and access the computational environment in which to experiment (numerically) from their browser, which helps lowering the usability barrier.

The public MyBinder service, which hosts the hardware on which the container is executed, comes with some limitations: For example, a notebook session that is idle (i.e., no computation and no user activity) for 10 min will be stopped from the MyBinder site, and all changes will be lost. The notebook and other files can be downloaded before the session is stopped (and later uploaded if a continuation of the work is desired). The computing hardware offered by MyBinder is relatively weak (for example at most two CPU cores).

Despite these limitations, MyBinder has been very useful for our teaching experience in providing a zero-install computational environment: Most students have carried out all of their simulation computation on the MyBinder service. The reason the MyBinder service works well for our projects is that the computation required for the student exercises is relatively modest and can be completed within minutes to hours on single-core CPUs. If one wanted to offer the same no-install computational environment for projects that have more substantial computational demands, the university could host and run their own Binder service: the BinderHub[35] is designed for this.[**] However, the skills required to set this up

[†]URL: https://mybinder.org/.
[‡]Ubermag repository https://github.com/ubermag/tutorials on Github.
[§]Ubermag on mybinder.org: https://mybinder.org/v2/gh/ubermag/tutorials/latest.
[**]We note that JupyterHub is the part of BinderHub responsible for running the server (after BinderHub builds the image), and that—given appropriate skill sets—it can be configured to have additional functionality, such as required user authentication, persistent storage, or control of one or more software environments that can be launched.

exceed those of most academics, and help from the local computing or IT team is likely to be required.

A local install of Ubermag on the student's computer is also possible, and some students have chosen to follow this path. Once the installation is completed, this is more convenient for ongoing and extended studies.

3. Zero-install and zero-hosting with JupyterLite

Looking ahead, the just emerging JupyterLite project[††] circumvents the shortcomings of the MyBinder service. JupyterLite makes it possible to execute Jupyter notebooks and many Python packages in the user's browser (using WebAssembly) and holds great potential for the use of software in the classroom in the future: (i) like Binder, JupyterLite is a zero-install approach, and (ii) the JupyterLite approach does not need other centralized computing resources (i.e., it is a zero-hosting approach).

In the JupyterLite set up, the complete and preconfigured software environment is provided for the learners on a (static) web page. Once the webpage is opened by the learner, the software environment is executed in the browser of the learner's own device (computer, laptop, chromebook, etc.), which provides the computing power. Such consumer devices are generally powerful enough and have no limit in run-time, and there is no dependency on cloud-hosted or other compute resources. (At the moment, the micromagnetic simulation software is not available as WebAssembly.)

C. User interface design for simulation software in education

Our hypothesis is that the use of simulation packages in advanced STEM classes can have educational value. We had a positive experience using the Ubermag software. In this section, we describe two important aspects of the user interface design.

1. Expose concepts of computational modeling

When a computer simulation is used to study a science or engineering problem, there are multiple layers of decision making and simplifications of the problem taking place (we assume that the model equations include differential equations):

(1) Decide on the model to be used and express the model in equations.
(2) Discretize the model in some form (on grid).
(3) Solve the discretized equation.

Many simulation packages are written for a particular model description and provide all the steps 1–3. In particular, the separation between these different aspects is not visible to the user. In Ubermag, this separation of concerns is more clearly exposed and accessible, and, thus, the meaning of the individual steps is easier for the learner to understand:

(1) Decide on physics approximations and the model to be used: Within the Ubermag framework, the user selects the relevant physics through the terms that contribute to the energy and dynamics of the system from the `micromagneticmodel` Python package. This creates a *machine-readable* definition[‡‡] of a micromagnetic problem. Computer scientists would express this so that the `micromagneticmodel` Python package provides a *Domain Specific Language* for micromagnetic models of the real world.[11]
(2) Discretize the model in some form: This requires splitting space into smaller parts such as cuboidal cells for finite difference discretization and a wider choice of geometrical objects for finite elements. Within the Ubermag framework, the `discretisedfield` package is used to define a (finite difference) discretization of space, and scalar and vector fields defined on that discretized space.
(3) Using the micromagnetic model definition together with the discretization, the problem can be solved numerically. Ubermag translates the information from the micromagnetic model and the discretized field into a configuration file that is understood by one of the micromagnetic calculators that it supports. Using the OOMMF Calculator (`oommfc`) or the mumax[3] Calculator (`mumax3c`) Python package, Ubermag then delegates the actual numerical solution to OOMMF or mumax[3].

Through the use of different packages—with clearly defined and orthogonal concerns—the concepts of computational science become easier to grasp than if all of those aspects are grouped together in the black-box simulation software.

2. Focus on physics, not the package-specific syntax

A potential user of the software needs to learn and understand what physics model choice and discretization is implemented in the software and needs to learn how to instruct the software (often through a configuration file) to use the right model, and to combine this with the required geometry, material and other parameters, initial configuration, time-dependent or spatially resolved external effects, etc. Generally, the required configuration file syntax is simulation package dependent: A scientist (or student), thus, needs to learn this syntax for every new simulation package they want to use, which contributes to the *usability barrier*.

The Ubermag framework provides an abstraction from the specific simulation package configuration file syntax in the domain of micromagnetics. The `micromagneticmodel` package provides the machine-readable definition of the problem using a syntax that scientists perceive as somewhat intuitive, and Ubermag can automatically translate this into the package-specific configuration file syntax. It is, thus, much easier to define a micromagnetic problem with Ubermag than it would be if the packages OOMMF and mumax[3] were used directly. We believe this reduction in complexity (of specifying a problem in a particular syntax) makes it possible to explore a much wider set of topics within the teaching module and the computational projects. This idea—to provide more "user-friendly" high-level interface to existing simulation software—is certainly

[††] https://jupyterlite.readthedocs.io (accessed May 5, 2023).

[‡‡] The machine-readable problem definition means that a computer (or researcher, educator, or learner) can read it and extract all needed information to fully define the physics problem of interest. For the learning context, the machine-readability ensures completeness of information. In a research and industry context, machine-readability is a pre-requisite for increasing automation of simulation-based work. It also supports reproducibility.

transferable to other domains. Examples include the atomic simulation environment (ASE)[36] and the materials science workflow tool AiiDA.[37]

IV. CONCLUSIONS

We have introduced computer simulation into a materials science class and describe approaches we found beneficial for the learning experience. It would be interesting to evaluate these in the context of different subject areas and educational settings:

- Choice of Python as one language for simulation and analysis, with broad library support.
- Design and use of user interfaces that focus on the learner's interest: expose modeling concepts and hide peculiarities and complexity of underlying simulation engines.
- Use of Jupyter Notebooks to encourage interactive exploration of the (simulated) system under study.
- Binder capabilities of project Jupyter, which make it possible to execute simulations in the cloud rather than on the students' computers. This overcomes the (sometimes significant) challenge of installing the software locally.

Extensions of work described here include combinations of Jupyter based simulation teaching with computational essays,[34,38] the "nbgrader"' tool supporting the grading in Jupyter notebooks,[32] and the opportunities for the JupyterLite (Sec. III B 3) based zero-install zero-hosting provisioning of software at scale.

SUPPLEMENTARY MATERIAL

Please click on this link to access the supplementary material, which includes a detailed overview of five simulation projects, the problem sheets, a general overview of computational methods in education, a thematic analysis of student and teacher feedback on the computational projects, and additional practical considerations. Print readers can see the supplementary material at https://doi.org/10.60893/figshare.ajp.c.7349842

ACKNOWLEDGMENTS

The authors would like to thank Thomas Wilhelm (Goethe University Frankfurt) for fruitful discussions on numerical methods in higher education and Min Ragan-Kelley for helpful discussions on the project Jupyter. M.L. acknowledges the financial support by the German Science Foundation (Deutsche Forschungsgemeinschaft, DFG) through the research fellowship LO 2584/1-1 for the development of the simulation projects 1 and 2, the implementation of student interviews and questionnaires as well as the manuscript preparation. S.H., S.P., and H.F. were supported by the Engineering and Physical Sciences Research Council's United Kingdom Skyrmion Programme Grant (EP/N032128/1). The development of the MSE 598/498/464 course and specifically the simulation projects 3 and 4 as well as the efforts from R.K. and A.H. were partially supported by the NSF through the University of Illinois Urbana-Champaign Materials Research Science and Engineering Center Grant No. DMR-1720633. The development of the simulation project 5 by T-HL was supported by the U.S. Department of Energy, Office of Science, Materials Science and Engineering Division, under Contract No. DE-SC0022060.

AUTHOR DECLARATIONS

Conflict of Interest

The authors have no conflicts to disclose.

[a]Electronic mail: lonskymartin@gmail.com, ORCID: 0000-0002-8955-3095.
[b]ORCID: 0000-0001-7104-7867.
[c]ORCID: 0000-0003-3323-8958.
[d]ORCID: 0000-0003-3840-955X.
[e]ORCID: 0000-0001-9878-041X.
[f]ORCID: 0000-0003-3747-2979.
[g]ORCID: 0000-0002-6670-3994.
[h]ORCID: 0000-0002-1808-2767.
[i]Electronic mail: hans.fangohr@mpsd.mpg.de, ORCID: 0000-0001-5494-7193.

[1]B. Skuse, "The third pillar," Phys. World **32**(3), 40–43 (2019).
[2]J. Weber and T. Wilhelm, "The benefit of computational modelling in physics teaching: A historical overview," Eur. J. Phys. **41**(3), 034003 (2020).
[3]M. D. Caballero and L. Merner, "Prevalence and nature of computational instruction in undergraduate physics programs across the United States," Phys. Rev. Phys. Educ. Res. **14**(2), 020129 (2018).
[4]G. Kortemeyer, "Incorporating computational exercises into introductory physics courses," J. Phys. **1512**(1), 012025 (2020).
[5]A. L. Graves and A. D. Light, "Hitting the ground running: Computational physics education to prepare students for computational physics research," Comput. Sci. Eng. **22**(4), 50–60 (2020).
[6]D. M. Cook, "Computation in undergraduate physics: The Lawrence approach," Am. J. Phys. **76**(4), 321–326 (2008).
[7]L. McNeil and P. Heron, "Preparing physics students for 21st-century careers," Phys. Today **70**(11), 38–43 (2017).
[8]AAPT Undergraduate Curriculum Task Force, "AAPT recommendations for computational physics in the undergraduate physics curriculum" Report (2016); available at http://www.aapt.org/Resources/upload/AAPT_UCTF_CompPhysReport_final_B.pdf.
[9]N. Chonacky and D. Winch, "Integrating computation into the undergraduate curriculum: A vision and guidelines for future developments," Am. J. Phys. **76**(4), 327–333 (2008).
[10]M. D. Caballero, M. A. Kohlmyer, and M. F. Schatz, "Implementing and assessing computational modeling in introductory mechanics," Phys. Rev. ST Phys. Educ. Res. **8**(2), 020106 (2012).
[11]M. Beg, M. Lang, and H. Fangohr, "Ubermag: Toward more effective micromagnetic workflows," IEEE Trans. Magn. **58**(2), 7300205 (2022).
[12]A. Kononov, P. Bellon, T. Bretl, A. Ferguson, G. Herman, K. Kilian, J. Krogstad, C. Leal, R. Maass, A. Schleife, J. Shang, D. Trinkle, and M. West, "Computational curriculum for MatSE undergraduates," in ASEE Annual Conference & Exposition Proceedings, 2017.
[13]R. Mansbach, A. Ferguson, K. Kilian, J. Krogstad, C. Leal, A. Schleife, D. R. Trinkle, M. West, and G. L. Herman, "Reforming an undergraduate materials science curriculum with computational modules," J. Mater. Educ. **38**, 161–174 (2016); available at https://icme.unt.edu/journal.
[14]R. Mansbach, G. Herman, M. West, D. Trinkle, A. Ferguson, and A. Schleife, "WORK IN PROGRESS: Computational modules for the MatSE undergraduate curriculum," ASEE Annual Conference & Exposition Proceedings, 2016.
[15]X. Zhang, A. Schleife, A. Ferguson, P. Bellon, T. Bretl, G. Herman, J. Krogstad, R. Maass, C. Leal, D. Trinkle, J. Shang, and M. West, "Computational curriculum for MatSE undergraduates and the influence on senior classes," in ASEE Annual Conference & Exposition Proceedings, 2018.
[16]C.-W. Lee, A. Schleife, D. Trinkle, J. Krogstad, R. Maass, P. Bellon, J. Shang, C. Leal, M. West, T. Bretl, G. Herman, and S. Tang, "Impact of computational curricular reform on non-participating undergraduate courses: Student and faculty perspective," in ASEE Annual Conference & Exposition Proceedings, 2019.
[17]T. L. Gilbert, "Classics in magnetics a phenomenological theory of damping in ferromagnetic materials," IEEE Trans. Magn. **40**(6), 3443–3449 (2004).

[18]M. Lonsky and A. Hoffmann, "Dynamic excitations of chiral magnetic textures," APL Mater. **8**(10), 100903 (2020a).
[19]M. Lonsky and A. Hoffmann, "Coupled Skyrmion breathing modes in synthetic ferri- and antiferromagnets," Phys. Rev. B **102**(10), 104403 (2020).
[20]M. Lonsky and A. Hoffmann, "Dynamic fingerprints of synthetic antiferromagnet nanostructures with interfacial Dzyaloshinskii–Moriya interaction," J. Appl. Phys. **132**(4), 043903 (2022).
[21]M. J. Donahue and D. G. Porter, "OOMMF User's Guide, Version 1.0," Interagency Report No. NISTIR 6376 (National Institute of Standards and Technology, Gaithersburg, MD, 1999).
[22]A. Vansteenkiste, J. Leliaert, M. Dvornik, M. Helsen, F. Garcia-Sanchez, and B. V. Waeyenberge, "The design and verification of MuMax3," AIP Adv. **4**(10), 107133 (2014).
[23]M. Beg, R. A. Pepper, and H. Fangohr, "User interfaces for computational science: A domain specific language for OOMMF embedded in Python," AIP Adv. **7**(5), 056025 (2017).
[24]H. Fangohr, M. Lang, S. J. R. Holt, S. A. Pathak, K. Zulfiqar, and M. Beg, "Vision for unified micromagnetic modeling (UMM) with Ubermag," AIP Adv. **14**(1), 015138 (2024).
[25]H. Fangohr, "A comparison of C, MATLAB, and Python as teaching languages in engineering," in *Computational Science – ICCS 2004,* Lecture Notes in Computer Science, edited by M. Bubak, G. D. van Albada, P. M. A. Sloot, and J. Dongarra (Springer, 2004), Vol. 3039.
[26]T. Kluyver, B. Ragan-Kelley, F. Pérez, B. Granger, M. Bussonnier, J. Frederic, K. Kelley, J. Hamrick, J. Grout, S. Corlay, P. Ivanov, D. Avila, S. Abdalla, and C. Willing, "Jupyter notebooks—A publishing format for reproducible computational workflows," in *Positioning and Power in Academic Publishing: Players, Agents and Agendas*, edited by F. Loizides and B. Schmidt (IOS Press, 2016), pp. 87–90.
[27]F. Perez and B. E. Granger, "IPython: A system for interactive scientific computing," Comput. Sci. Eng. **9**(3), 21–29 (2007).
[28]B. E. Granger and F. Perez, "Jupyter: Thinking and storytelling with code and data," Comput. Sci. Eng. **23**(2), 7–14 (2021).
[29]J. M. Perkel, "Why Jupyter is data scientists' computational notebook of choice," Nature **563**(7729), 145–146 (2018).
[30]H. Fangohr, M. Beg, M. Bergemann, V. Bondar, S. Brockhauser, A. Campbell, C. Carinan, R. Costa, F. Dall'Antonia, C. Danilevski, J. E, W. Ehsan, S. Esenov, R. Fabbri, S. Fangohr, E. F. del Castillo, G. Flucke, C. Fortmann-Grote, D. F. Marsa, G. Giovanetti, D. Goeries, A. Götz, J. Hall, S. Hauf, D. Hickin, T. H. Rod, T. Jarosiewicz, E. Kamil, M. Karnevskiy, J. Kieffer, Y. Kirienko, A. Klimovskaia, T. Kluyver, M. Kuster, L. L. Guyader, A. Madsen, L. Maia, D. Mamchyk, L. Mercadier, T. Michelat, I. Mohacsi, J. Möller, A. Parenti, E. Pellegrini, J. Perrin, M. Reiser, J. Reppin, R. Rosca, D. Rück, T. Rüter, H. Santos, R. Schaffer, A. Scherz, F. Schlünzen, M. Scholz, M. Schuh, J. Selknaes, A. Silenzi, G. Sipos, M. Spirzewski, J. Sztuk, J. Szuba, J. Taylor, S. Trojanowski, K. Wrona, A. Yaroslavtsev, and J. Zhu, "Data exploration and analysis with Jupyter notebooks," in *Proceedings of the 17th International Conference on Accelerator and Large Experimental Physics Control Systems, ICALEPCS'19* (JACoW Publishing, Geneva, Switzerland, 2020), pp. 799–806.
[31]L. A. Barba, L. J. Barker, D. S. Blank, J. Brown, A. Downey, T. George, L. J. Heagy, K. Mandli, J. K. Moore, D. Lippert, K. Niemeyer, R. Watkins, R. West, E. Wickes, C. Willling, and M. Zingale (2022). "Teaching and learning with Jupyter," Figshare. https://doi.org/10.6084/m9.figshare.19608801.v1
[32]P. Jupyter, D. Blank, D. Bourgin, A. Brown, M. Bussonnier, J. Frederic, B. Granger, T. Griffiths, J. Hamrick, K. Kelley, M. Pacer, L. Page, F. Pérez, B. Ragan-Kelley, J. Suchow, and C. Willing, "nbgrader: A tool for creating and grading assignments in the Jupyter notebook," J. Open Source Educ. **2**(11), 32 (2019).
[33]M. Beg, J. Taka, T. Kluyver, A. Konovalov, M. Ragan-Kelley, N. M. Thiery, and H. Fangohr, "Using Jupyter for reproducible scientific workflows," Comput. Sci. Eng. **23**(2), 36–46 (2021).
[34]T. O. B. Odden, D. W. Silvia, and A. Malthe-Sørenssen, "Using computational essays to foster disciplinary epistemic agency in undergraduate science," J. Res. Sci. Teach. **60**(5), 937–977 (2023).
[35]Project Jupyter, Matthias Bussonnier, Jessica Forde, Jeremy Freeman, Brian Granger, Tim Head, Chris Holdgraf, Kyle Kelley, Gladys Nalvarte, Andrew Osheroff, M. Pacer, Yuvi Panda, Fernando Perez, Benjamin Ragan Kelley, and Carol Willing, "Binder 2.0—Reproducible, interactive, sharable environments for science at scale," in *Proceedings of the 17th Python in Science Conference*, edited by Fatih Akici, David Lippa, Dillon Niederhut, and M. Pacer (2018), pp. 113–120.
[36]A. H. Larsen, J. J. Mortensen, J. Blomqvist, I. E. Castelli, R. Christensen, M. Dułak, J. Friis, M. N. Groves, B. Hammer, C. Hargus, E. D. Hermes, P. C. Jennings, P. B. Jensen, J. Kermode, J. R. Kitchin, E. L. Kolsbjerg, J. Kubal, K. Kaasbjerg, S. Lysgaard, J. B. Maronsson, T. Maxson, T. Olsen, L. Pastewka, A. Peterson, C. Rostgaard, J. Schiøtz, O. Schütt, M. Strange, K. S. Thygesen, T. Vegge, L. Vilhelmsen, M. Walter, Z. Zeng, and K. W. Jacobsen, "The atomic simulation environment—A Python library for working with atoms," J. Phys. **29**(27), 273002 (2017).
[37]S. P. Huber, S. Zoupanos, M. Uhrin, L. Talirz, L. Kahle, R. Häuselmann, D. Gresch, T. Müller, A. V. Yakutovich, C. W. Andersen, F. F. Ramirez, C. S. Adorf, F. Gargiulo, S. Kumbhar, E. Passaro, C. Johnston, A. Merkys, A. Cepellotti, N. Mounet, N. Marzari, B. Kozinsky, and G. Pizzi, "AiiDa 1.0, a scalable computational infrastructure for automated reproducible workflows and data provenance," Sci. Data **7**, 300 (2020).
[38]A. diSessa, *Changing Minds: Computers, Learning, and Literacy* (MITPress, Cambridge, MA, 2000).