

# Developing computational skills through simulation based problem-solving in science

Martin Lonsky,<sup>1,2,\*</sup> Martin Lang,<sup>3,4</sup> Samuel Holt,<sup>3,4</sup> Swapneel Amit Pathak,<sup>3,4</sup> Robin Klause,<sup>1</sup> Tzu-Hsiang Lo,<sup>1</sup> Marijan Beg,<sup>5</sup> Axel Hoffmann,<sup>1</sup> and Hans Fangohr<sup>4,3,6,†</sup>

<sup>1</sup>*Materials Research Laboratory and Department of Materials Science and Engineering, University of Illinois at Urbana-Champaign, Urbana, Illinois 61801, USA*

<sup>2</sup>*Institute of Physics, Goethe University Frankfurt, 60438 Frankfurt, Germany*

<sup>3</sup>*Faculty of Engineering and Physical Sciences, University of Southampton, Southampton, SO17 1BJ, United Kingdom*

<sup>4</sup>*Max Planck Institute for the Structure and Dynamics of Matter, 22761 Hamburg, Germany*

<sup>5</sup>*Department of Earth Science and Engineering, Imperial College London, London SW7 2AZ, United Kingdom*

<sup>6</sup>*Center for Free-Electron Laser Science, 22761 Hamburg, Germany*

(Dated: March 6, 2023)

Computational modeling and numerical simulations have become indispensable tools in science, technology, engineering and mathematics (STEM), and in industrial research and development. Consequently, there is growing demand for computational skills in undergraduate and postgraduate students and a need to revise existing curricula. We report from a case study where an existing materials science module was modified to contain numerical simulation of the materials: students use simulation software to explore the material's behavior in simulated experiments. In particular, the Ubermag micromagnetic simulation software package is used by the students in order to solve problems computationally that are linked to current research in the field of magnetism. The simulation software is controlled through Python code in Jupyter notebooks. This setup provides a computational environment in which simulations can be started and analyzed in the same notebook. A key learning activity is a project in which students tackle a given task over a period of approximately 2 months in a small group. Our experience is that the self-paced problem-solving nature of the project work — combined with the freedom to explore the system's behavior through the simulation — can facilitate a better in-depth exploration of the course contents. We report feedback from students and educators both on the training in material science and the Jupyter notebook as a computational environment for education. Finally, we discuss which aspects of the Ubermag and the Jupyter notebook have been beneficial for the students' learning experience and which could be transferred to similar teaching activities in other subject areas.

## I. INTRODUCTION

Traditionally, disciplines such as physics, chemistry and materials science have been divided into two strongly interrelated parts, namely theory and experiment. The experiment constitutes a means to confirm or falsify hypotheses that are typically arising from a carefully developed theory. At the same time, experiments can also initiate the development of novel theories. More generally, scientific investigations can be characterized by a cyclical process of connecting experiment and theory [1]. It may appear natural to reflect the classification into theory and experiment in teaching curricula: We often find theory-focused classes alongside experimental courses at university level.

However, significant advances in technology and computing have entailed a novel building block of science, namely computational modeling and numerical simulations. Not only in the natural sciences, but also within the engineering community, this new branch has rapidly evolved into the third fundamental pillar besides theory and experiment [2]. Both experimentalists and theorists make use of computational techniques in their research

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 a case, numerical studies can help to improve our understanding despite the above challenges. Prominent examples of computer simulations include climate and weather forecasting [3], computational fluid dynamics [4], the modeling of chemical reactions [5] and most recently the SARS-CoV-2 pandemic [6].

In STEM education, computational modeling has an important role [7] and it is widely argued that more computational content in curricula would be desirable [e.g. 8, 9]. Anecdotal evidence on undergraduate programs at numerous universities worldwide suggests that computational contents remain severely underrepresented in the respective curricula [9–12], despite recent studies presenting evidence that computational methods education in undergraduate coursework may lead to the development of multiple essential skills [13, 14]. 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 and thus recommends the incorporation of extensive computational physics contents into undergraduate programs [15, 16]. Addressing pre-university education, high school classrooms are being discussed to make computation a part of science learning [17]. Computational modeling and numerical simulations appear

\* mlonsky@physik.uni-frankfurt.de

† hans.fangohr@mpsdl.mpg.de

crucial for obtaining a complete picture of the modern STEM disciplines, and therefore adequate ways need to be found to embed this young branch into teaching curricula.

In the context of teaching, we can distinguish between at least two approaches toward incorporating computational methods in a curriculum [8]: Firstly, there are modules which solely focus on programming, numerical methods, or modeling and numerical simulations (e.g., a class on computational physics). Secondly, computational content can also be introduced by embedding it in existing modules. The latter approach is at the core of the case study presented in this work.

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 at Urbana-Champaign (UIUC). The course is available to students in different fields, such as electrical and computer engineering or physics. In more detail, the students set up and perform micromagnetic simulations by using the open computational environment [18] *Uebermag* [19]. In this article, we describe our methodology, characteristics, and experiences of *Uebermag* and related computational software packages when used in STEM instruction. We discuss our insights from the teaching delivery, student evaluations and personal interview surveys.

The paper is structured as follows. In Sec. II, we review the current status of teaching computational modeling and numerical simulations at university level. This involves several examples of already existing classes and tools, as well as suggestions for further applications in undergraduate courses. Section III contains a brief introduction to computational micromagnetics, which is followed by a detailed description of the fundamental aspects of the *Uebermag* software package and its application in the classroom at UIUC. Subsequently, in Sec. IV we present the evaluation of student feedback and personal interview surveys. Based on the feedback and our own experience, we then give recommendations for the implementation of computational projects in other courses in Sec. V. Finally, we summarize our findings in Sec. VI.

## II. INCORPORATION OF COMPUTATIONAL METHODS INTO STEM PROGRAMS

There are various ways how computational methods can be embedded into traditional modules that form an existing undergraduate or graduate curriculum. In general, this may be done in the form of computational assignments, computational lectures, computational laboratories and computational research projects [20]. In the case of physics, an excellent platform for relevant course materials, teaching approaches, workshops and interaction with fellow teachers has been established by the Partnership for Integration of Computation into Under-

graduate Physics (PICUP) organization [21]. A further comprehensive compilation of useful resources on computational physics can be found in Ref. [22]. In the remainder of this section, we will give several examples on how to embed computational methods into STEM programs.

A popular tool used in introductory STEM classes is given by Visual Python (VPython), a visual extension of Python that can be utilized for simulating simple experiments with only a few lines of code [11]. For instance, VPython has been used in the aerospace engineering program at the University of Southampton, where second-year students were assigned computational projects related to three-dimensional real time visualization [23]. In physics, VPython has been shown to be particularly useful for classical mechanics classes and can nowadays also be run in Jupyter Notebooks and in browsers by using Web VPython. VPython can also be introduced simultaneously with an introduction to programming if students have no prior programming experience [23]. In a survey conducted by Zhang *et al.*, a majority of materials science and engineering students have indicated that learning computational skills in the second undergraduate year is optimal [24]. On the other hand, one can argue that contents which enhance students' computational literacy should be conveyed as early as possible so they can be used, exploited and extended in all parts of the degree program in the subsequent years. Caballero and Merner point out that faculty frequently do realize the importance of computation and programming, however, the majority is still reluctant to include these topics into their courses due to a variety of reasons, such as time constraints and insufficient (perceived) expertise, since typically most of the faculty do not specialize in computation [9, 25].

Further examples of computational contents in STEM curricula are given in the following. In terms of closed computational environments, Physlets [26] and the Physics Education Technology (PhET) Interactive Simulations project at the University of Colorado Boulder [27] are two noteworthy products for introductory physics courses. A major portion of open computational environments are focused on more advanced and specialized topics, such as the interactive molecular dynamics simulation code developed at Weber State University [28], and the nanoHUB platform [29] which is used in the course "ME 581 – Numerical Methods in Engineering Using Jupyter Notebooks" at Purdue University [30, 31]. Aside from the aforementioned course, the nanoHUB instructional materials are also available as open content under a Creative Commons license and can be used for classes anywhere in the world. Various simulation software packages are easily accessible and can be controlled through a graphical user interface. At UIUC, instructors have started using some of the resources for various undergraduate courses [20, 24, 32–34], e.g., in a class on the "Electronic Properties of Materials" (MSE 304). This involves several homework assignments where students have to hand in reports on solving various com-

putational problems. In detail, the students use density functional theory (DFT) to investigate properties such as the density of states, band structure and effective masses of electrons and holes in different materials by utilizing the Quantum ESPRESSO computer code [35, 36] on nanoHUB. Furthermore, in another assignment, students are asked to use ABACUS (Assembly of Basic Applications for the Coordinated Understanding of Semiconductors) [37] to obtain a qualitative and quantitative understanding of a semiconducting diode. Aside from the nanoHUB set of tools, a comprehensive overview of relevant methods in the context of teaching computational materials science and engineering can be found in Ref. [38].

Further advances concerning the inclusion of computational methods in existing courses will benefit from simulation environments such as Ubermag [19] and the Atomistic Simulation Environment (ASE) [39]. ASE is a software package for atomistic simulations provided by researchers at the Technical University of Denmark. Ubermag is described in more detail in Sec. III. Both Ubermag and ASE provide a high-level and convenient interface to multiple simulation packages: Controlling these simulation packages directly requires more specialized skills than using the high-level interface. Both ASE and Ubermag expose a Python interface to the user and are supported by extensive documentation, which includes tutorials, frequently asked questions and contents of a workshops. A key contribution to the value that these simulation environments offer for education is the *lowering of the usability barrier*: How much of a (often cryptic and historically grown) syntax does the student need to know before they can get different simulation results? The learning takes place through changing the simulation inputs and understanding how the outputs change in response. This is more efficient if the changes of the input are concise and cognitively not too demanding.

Other examples of software with high-level interfaces and thus good potential for teaching include COMSOL Multiphysics [40, 41] and the Einstein Toolkit which offers computational tools for relativistic astrophysics and gravitational physics applications [42].

There are many ways in which computational activities can be embedded in our teaching curricula, including incorporation of computational exercises into introductory courses, see for example Refs. [11, 14, 43–47]. In laboratory courses experiments can be combined with computer simulations, for example by using VPython [47], Mathematica [48] or Microsoft Excel [49]. It was reported that it may be beneficial to establish dedicated computational physics or engineering laboratories [14]. A further interesting method to embed computational contents into the curriculum is given by so-called computational essays [50]. Originally proposed in a work by diSessa [51], computational essays consist of a combination of text, executable code, interactive diagrams and other computational tools. For the realization of such essays, interactive Jupyter notebooks [52] represent an ideal tool.

Following this overview of previous efforts on incorporating computational contents into STEM curricula, we now turn to the specific case of micromagnetic simulation projects in a materials science and engineering course.

### III. MICROMAGNETIC SIMULATION PROJECTS IN A MATERIALS SCIENCE AND ENGINEERING COURSE

We present a case study that we conducted within the framework of a class on “Magnetic Materials and Applications” in the Department of Materials Science and Engineering at UIUC. In this context, we utilized the software package Ubermag [19], 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 simulation package. Small-group computational projects represent an ideal means to complement a traditional lecture course by a more student-centered aspect — it has been shown in previous studies that such (technology-enabled) active-learning approaches [53] can give rise to an increased student performance [54–56]. Ubermag offers a simple and easy-to-learn approach to create, control and run simulation scripts that solve the underlying partial differential equation which describes the temporal evolution of the magnetic field in a specified materials system.

We start with an introduction to (analytical) micromagnetic theory in Subsec. III A, followed by the numerical computation of solutions to the analytical micromagnetic problem using so-called *computational micromagnetics* in Subsec. III B. Subsection III C introduces the Ubermag software and Subsec. III D describes the teaching activity in detail.

#### A. Introduction to Micromagnetics

The basis of time-dependent micromagnetics is the equation of motion of the magnetization vector field (Landau-Lifshitz-Gilbert, LLG equation) [57]:

$$\frac{d\mathbf{m}}{dt} = -|\gamma_0|\mathbf{m} \times \mathbf{H}_{\text{eff}}(\mathbf{m}) + \alpha\mathbf{m} \times \frac{d\mathbf{m}}{dt}. \quad (1)$$

Here,  $\gamma_0$  denotes the gyromagnetic constant,  $\alpha$  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,  $\mathbf{H}_{\text{eff}}$ , which is itself a function of the mag-

netization vector field. The effective field can be computed from the energy  $E$  of the system ( $\mathbf{H}_{\text{eff}} = -\frac{1}{\mu_0} \frac{\delta E}{\delta \mathbf{m}}$ ).

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

$$E(\mathbf{m}) = E_{\text{Ex}}(\mathbf{m}) + E_{\text{Z}}(\mathbf{m}) + E_{\text{Dem}}(\mathbf{m}) + E_{\text{Anis}}(\mathbf{m}) + E_{\text{DMI}}(\mathbf{m}), \quad (2)$$

where  $E_{\text{Ex}}$  denotes the exchange energy,  $E_{\text{Z}}$  the Zeeman energy,  $E_{\text{Dem}}$  the demagnetization energy,  $E_{\text{Anis}}$  the anisotropy energy, and  $E_{\text{DMI}}$  the Dzyaloshinskii-Moriya interaction (DMI). All energy terms involve integrals over the volume, some involve vector analysis operators, and the demagnetization term ( $E_{\text{Dem}}$ ) contains a double integral due the long-range nature of the demagnetization effects [58].

In summary, the micromagnetic problem — summarized through Eqs. (1) and (2) — is complex. Mathematically, this is reflected in Eq. (1) being a non-linear integro partial differential equation. In terms of the physics that is described by the model, there is a rich variety of phenomena ranging from fundamental physics to applied engineering. It is this complexity and richness which results in thousands of publications, and that makes the model a fruitful ground for advanced materials education.

## B. Introduction to Computational Micromagnetics

The micromagnetic model can be solved analytically only for very few 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, and then repeating (where  $t \rightarrow t + \Delta t$ ). Relevant magnetization dynamics phenomena include ferromagnetic resonance and spin-wave propagation [59–61]. However, the LLG can also be used to determine equilibrium configurations of the magnetization field, or conventional energy minimization can be used.

There are at least two widely used software packages that solve the complex computational micromagnetics problem using finite differences: the Object Oriented MicroMagnetic Framework (OOMMF) [62] and mumax<sup>3</sup> [63]. The OOMMF software operates on the CPU of the utilized computer, whereas mumax<sup>3</sup> is a GPU-accelerated program and requires an NVIDIA GPU card to be installed. OOMMF is written in C++ and Tcl, mumax<sup>3</sup> 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, respectively. The learning curve for either

package is steep; while clearly acceptable in professional research activities, it is a challenge for occasional users such as students in an educational setting.

In what follows, we will discuss how Ubermag is related to the aforementioned simulation packages and how it can be used for teaching activities.

## C. The Ubermag Software and its Utilization in the Classroom

Ubermag has been developed to provide a Python interface [64] to OOMMF with the goal of providing an improved environment for researchers to support computational science investigations of magnetic materials and devices (recent examples include [65–67]). Later, Ubermag was extended to also interface with mumax<sup>3</sup> [19].

The Python packages provided by Ubermag allow to specify micromagnetic models, run simulations, and analyze as well as visualize data in interactive Jupyter notebooks. Only the computational solving of micromagnetic problems is delegated to the *micromagnetic calculators* (i.e. OOMMF or mumax<sup>3</sup>), while all other steps are independent from these simulation packages.

The main Ubermag Python packages that are typically used for a majority of simulations are termed `micromagneticmodel`, `discretisedfield`, `oommfc` and `mumax3c`. Here, `micromagneticmodel` is used to define micromagnetic models (i.e., describing the energy and dynamics equations that determine the underlying physics), `discretisedfield` is used to construct and visualize finite-difference fields to enable and prepare a numerical solution, and `oommfc` and `mumax3c` handle communication with the OOMMF and mumax<sup>3</sup> calculators, respectively.

This structured approach — with the aim of supporting scientists — is also beneficial in a teaching context: The structure of the software interface follows that of the physics model.

By using Ubermag, students can control and run their simulations via browser-based interactive Jupyter notebooks in an exploratory manner by re-executing code as often as it is desired (see Fig. 1). The modular structure of Jupyter notebooks allows to run blocks of code (so called “cells”) individually instead of running the entire simulation script. In general, such a notebook can contain the problem description, the (triggering of the) numerical computation as well as data analysis in a self-contained way. The environment invites exploration through modification of cells and their repeated and interactive execution: Students can obtain an in-depth understanding of the underlying physics by iteratively modifying and probing the system.

In addition to code cells, “Markdown” cells can be used in Jupyter notebooks to annotate the document. Markdown is a common markup language used for type setting. Equations, using LaTeX typesetting, are also supported in the Markdown cells. 2D, 3D and interac-

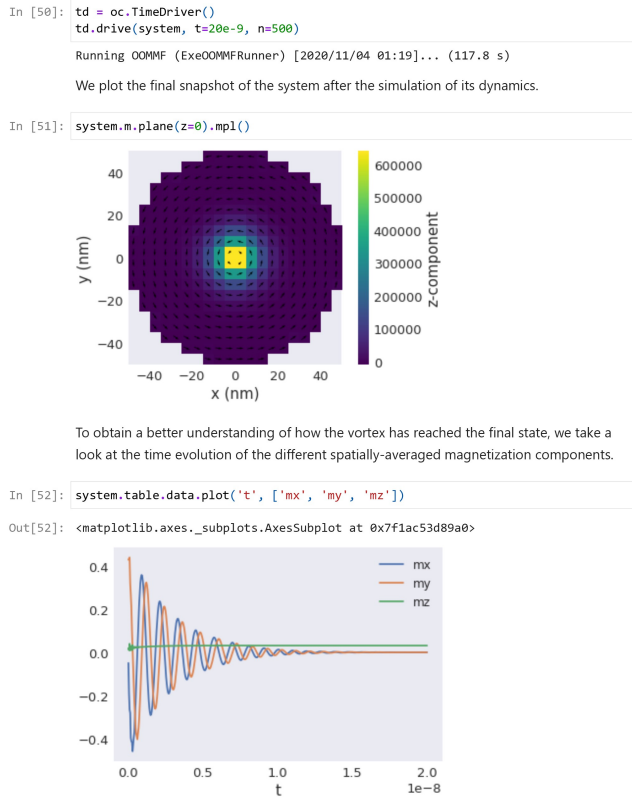


FIG. 1. Screenshot of a Jupyter notebook containing Ubermag-based micromagnetic simulations used in a web browser via Binder. This excerpt contains Python code, Markdown text and two diagrams. See main text for details.

tive plots can be included as well, for example, by using the (third-party) packages `matplotlib`, `k3d` and `pyvista`.

An exemplary screenshot of a (part of a) Jupyter notebook is depicted in Fig. 1. This particular notebook contains Python code, Markdown text, an image to visualize the magnetic configuration of a nanodisk with a diameter  $d = 100$  nm, and a plot of the time-dependence of the (spatially averaged) magnetization components  $m_x$ ,  $m_y$  and  $m_z$  after excitation by a magnetic field pulse. More information about this particular simulation project can be found in the supplementary material [68].

The consistent use of Python in the Ubermag research environment — for problem definition, execution of simulations, and data analysis — saves students from having to learn multiple programming languages (such as Tcl and Go). Instead, they can focus on the underlying physics while developing and expanding their skills in one of the most popular and powerful programming languages in the world.

The installation of software for teaching purposes can be a demanding topic: the university's or the students' personal laptops may be running a variety of operating systems (typically Windows, MacOS or Linux) with dif-

ferent 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 calculators (such as OOMMF or mumax<sup>3</sup>).

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](https://mybinder.org). In short: Ubermag can be executed in the cloud and controlled from the browser of the student. No installation on the student's machine is necessary. This has been very popular with the students. (We discuss this option in more detail in section VB2).

A last aspect we want to mention is the open-source nature of Ubermag. In contrast to, for example, COMSOL Multiphysics [40, 41] and other commercial simulation packages, Ubermag and Python-based Jupyter notebooks offer enhanced value, transparency and unlimited accessibility for students, instructors, and departments: No licenses need to be purchased as all software is open source and is available for download [69]. There is also no tie-in to a particular vendor: Should the university dislike the way the Ubermag package develops in the future, it could take the current Ubermag version and either keep using it as is, or modify it to suit its own needs best.

The ideal scenario is of course that users of the package feed back any requests for improvements (or actual code changes that implement these improvements) to the open source team. There are well established protocols for such contributions, although it may need skills beyond those of the average academic. Increasingly, universities employ research software engineers who can provide such skills [70].

#### D. The Elective Course on Magnetic Materials and Applications

Our case study has been conducted in the framework of the Magnetic Materials and Applications class (MSE 598/498/464) at UTUC. This elective course is aimed at both undergraduate and graduate students at the Department of Materials Science and Engineering, but also other students from the physics, chemical engineering and electrical engineering departments have been attending this class in the past. Over the past years, the class has seen an enrollment of 7 to 15 students per semester.

In the lecture, the fundamental concepts with regard to the practical use of magnetic materials are introduced. It is held over the span of about 16 weeks and in the syllabus 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 Can-

vas (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.

In total, we have designed five distinct simulation projects. We ask students to work in groups of two to four, since each project is divided into several subprojects. Therefore, each student can work on one of the subprojects. The subprojects are to some extent independent of each other, but they do have a certain overlap such that it is beneficial for the students to interact with their peers and discuss their solutions. An exemplary timeline of the computational projects in the fall semester 2021 at UIUC is illustrated in Fig. 2. Before the problems are handed out to the students, we give an introduction to Ubermag and micromagnetic simulations in a 90-minute class. Furthermore, we provide them with supplementary material such as video tutorials presented by the Ubermag developers and the accompanying software documentation, which is very comprehensive and includes numerous exemplary Jupyter notebooks. Due to the students' diverse background, their exposure to programming in general and Python in particular prior to working on the micromagnetic simulation 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 [20, 24, 32–34], while students from other majors who attend the course on Magnetic Materials and Applications may never have written their own code. Furthermore, it is plausible to assume a discrepancy in the average computational literacy between undergraduate and graduate students, who are both allowed to attend the class. The extensive Ubermag documentation and collection of example notebooks has been very beneficial to address this variety of existing skills.

The project reports are due around two months after the projects have been assigned. After the completion of the first delivery of the course (fall 2020) into which we had incorporated the simulation projects, we learned that 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 due to multiple reasons. Firstly, this discussion allows to provide preliminary feedback to the students and helps to prevent them from getting lost in details, which did happen to one group in the fall semester of 2020. 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. Lastly, it may be perceived as an intermediate deadline and thereby encourages students to get started with the projects as early as possible. We also ensured 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 several days

later. After each presentation, we aimed to stimulate a technical discussion and then solicited feedback from the audience on the presentation content and style.

We note that the Ubermag numerical simulations fit perfectly into the syllabus, since micromagnetic modeling had been discussed superficially in our course prior to the introduction of the simulation projects and is also typically presented in magnetism courses at other universities. Due to the complexity of the competing energy terms presented in Eq. (2), micromagnetic simulations constitute a rich and instructive playground for the students. Furthermore, the simulation projects are carefully designed such that each calculation runs for a reasonably short period of time, i.e., seconds to minutes. In combination with the intuitive control of Ubermag through Jupyter notebooks, students can develop and follow an exploratory approach by running simulations iteratively and visualizing their results immediately within the notebook.

Students work in groups of two to four, and thereby we encourage collaborations, which have been shown to enhance the learning outcome of computational problems [43]. Assigning programming exercises to groups of students is also expected to circumvent the problem of unauthorized collaboration and plagiarism, as well as to increase the student satisfaction [11].

A relatively new package termed `mag2exp` of Ubermag [71] allows to simulate physical quantities identical to those obtained from experimental techniques that are used to study magnetism, such as Lorentz transmission electron microscopy or small angle neutron scattering. While we have not implemented this in our simulation projects yet, this approach is expected to further reduce the gap between experiments and numerical modeling and thus represents a possible extension of the existing exercises.

The physics of the simulation projects and the individual projects are summarized in the supplementary material [68]. In the next section, we discuss the learning experience from the student and teacher perspective.

## IV. FEEDBACK ON LEARNING EXPERIENCE

### A. Feedback from student interviews

Personal interview surveys were carried out in October 2022 with six students of the classes offered in Fall 2020 and 2021 in order to assess their experience with the simulation projects. Prior to the interviews, we had reached out to 20 former students whose email addresses were still known and received a response by six individuals. The conversations took place on a video conferencing platform and typically lasted around 30 minutes per student. The interviews were recorded and then analyzed in further detail. During the personal interview survey we asked open-ended questions that touched upon the following central points:

## EXEMPLARY TIMELINE FOR MICROMAGNETIC SIMULATION PROJECTS (FALL 2021, UIUC)

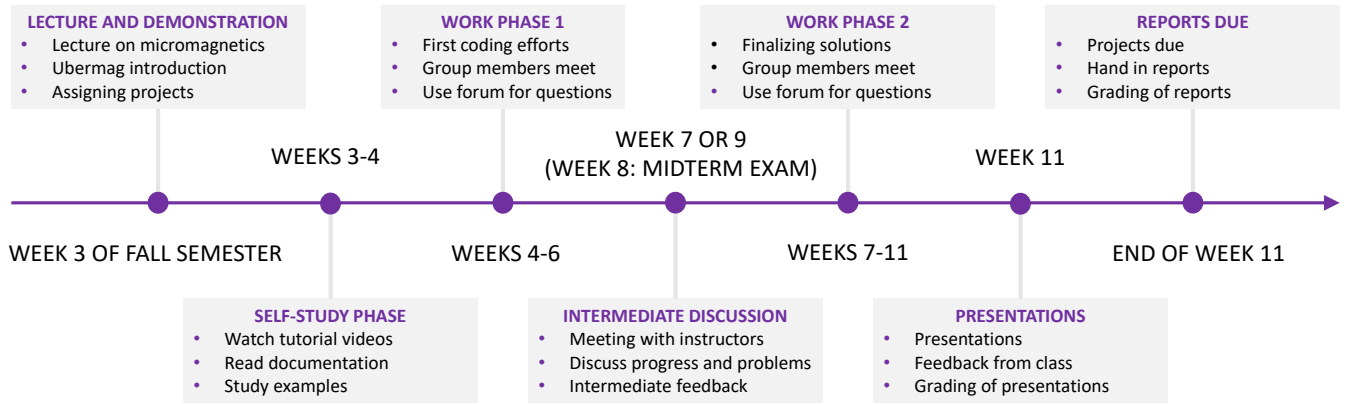


FIG. 2. Exemplary timeline of computational micromagnetics projects from the fall semester 2021.

- Overall experience with simulation projects, positive and negative aspects, suggestions for improvements;
- Difficulty of projects, taking into account students' proficiency in programming, especially with Python, and their prior experience in the field of magnetism;
- Assessment of the importance of computational methods for research and whether micromagnetic simulation projects changed students' attitude;
- Communication and collaboration within the student groups as well as the communication between the students and the instructional staff.

In what follows, we provide an overview of feedback obtained from the students. Due to the limited number of students who participated in the class on magnetic materials and applications over the course of the past years, a quantitative study is not possible and we focus on a qualitative analysis.

Several students mentioned that varying the different micromagnetic energy terms and visualizing the static and dynamic magnetization states helped them to obtain a better understanding of the subject matter in comparison to problem sheets or demonstrations in the lecture. One student with rather limited prior knowledge on both magnetism and computational methods said that the simulations gave them a much better understanding of the basics, which they could not obtain in the conventional lecture due to time constraints and the fact that the instructor has to cater to the needs of both undergraduate and graduate students whose levels of understanding can be vastly different.

Another student stated that their physical intuition for magnetism has significantly improved by working on the computational projects. Interestingly, the micromagnetic

simulation projects and especially the gained experience on programming with Python became a central topic in one student's job interview and appeared to be viewed as positive by the interviewers. Two students noted that micromagnetic simulations allowed them to obtain an experimentalist's perspective on magnetism without the need of being in a laboratory environment and, for example, fabricating expensive thin films or potentially facing safety issues. They do not think that computational methods should replace experimental lab courses, but certainly they are a good addition that does not require too many resources. One student brought up the issue that the class is for both undergraduate and graduate students. Therefore, the students' background, skills and prior knowledge (especially with regard to programming) may be too diverse and makes it difficult for the instructor to provide a successful learning experience to everybody in the classroom.

The communication within the project groups was typically perceived to be very good and was facilitated through students setting up virtual meetings, writing emails or sending text messages. One student mentioned that sometimes it was more convenient to ask the instructor or teaching assistants for help, since they would have a better background knowledge about programming and the underlying physics.

Different groups had different approaches to solving the problems. Some groups frequently exchanged ideas, comments and concerns about the different subtasks, but there was also one case where students had assigned tasks to each other initially and then no further significant interaction occurred. The communication between the students and the instructor as well as the teaching assistants was assessed positively. It was pointed out by several students that especially the intermediate discussions were highly useful for them to obtain preliminary feedback and talk about potential issues. Some students



said that they would have preferred a more detailed introduction to the Ubermag software package within the actual classroom environment in contrast to our comparably brief presentation and the subsequent reference to YouTube tutorials and the Ubermag documentation. Another session with the teaching assistants that could include short exercises prior to the assignment of the actual simulation projects may be beneficial in the future. Another suggestion given by a student is to design several smaller projects instead of one large computational project, as it is the case in the course on electronic properties of materials that is taught at UIUC. This would enable students to work on more than just one specific topic.

One student had been intimidated by computational methods a few years ago, but the micromagnetic simulation projects turned out to be a major reason why the student became more comfortable with the underlying concepts and numerical approaches. Moreover, some students have learned to recognize the importance of computational methods in science after working on the simulation projects, while other students stated that the projects are relatively limited in terms of their scope and duration, and thus their mindset has not changed significantly. We note that the Department of Materials Science and Engineering at UIUC has already incorporated computational methods into several core classes of the undergraduate curriculum and therefore students in this program may have a clearer picture with regard to the importance of computational methods in research compared to students at other departments and universities. A majority of students were slightly more engaged for the simulation projects than for other parts of the class. The underlying reason was not only the students' interest in computational methods, but also, for example, peer pressure (they did not want to disappoint their group members) and the relevance for their final grade. Overall, the students appeared to enjoy working on the simulation projects. For instance, one student wanted to explore another project that they were not assigned to on their own time. The student said that this other project also sounded very interesting and that they wanted to explore it in their free time. This is just an example of some students wanting to go beyond what is required of them.

One interviewee suggested that it would be beneficial to discuss in more detail the capabilities and limitations of computational methods. For instance, this student would have been interested to learn about the validity of computational methods, the approximations and corrections that are required, and whether artifacts such as the choice of the simulation cell size play an important role. Another student with prior experience in both OOMMF and mumax<sup>3</sup> noted that Ubermag is more suitable as a teaching tool compared to the other software packages. Its intuitive handling and straightforward visualization capabilities make discussions with other group members and the instructor considerably easier.

## B. Feedback from student questionnaires

In preparation for a future quantitative analysis, we also conducted an anonymous and paper-based survey among the participants of the class in the fall semester of 2022. Seven students in the class participated and were asked questions similar to those in the above-mentioned personal interview surveys.

We carried out two distinct surveys throughout the semester: One survey was conducted before assigning the simulation projects and another slightly modified survey was done after the completion of the two-month projects. The motivation for using questionnaires at two different points in time was to get a first idea about possible changes and trends in the students' perception and knowledge as a result of carrying out the computational projects. In the following, we briefly summarize the feedback and results.

Prior to working on the micromagnetic simulation projects, most students indicated that they were only barely or somewhat familiar with numerical methods, but on average reasonably comfortable with programming, especially with Python. All respondents recognized that computational methods are important in today's research landscape and could elaborate on the relevance of numerical approaches in research. A major portion of students claimed to have a very limited experience with magnetism. This semester's class was composed of students who gave a mixed response on whether they prefer experimental or theoretical research, underlining the diverse background of the participants. After having completed the simulation projects, a majority of students now respond that they are fairly or even very familiar with numerical methods. There is a slight trend towards being more comfortable with programming and comparably stronger indications for a perceived increase in the students' knowledge of magnetism. Nearly all students thought that Python is well suitable for such a project and, more specifically, indicate a similar attitude towards Jupyter notebooks and Ubermag. Almost all students stated that the difficulty of the projects was just right. Strikingly, each respondent said that the projects were much more enjoyable than regular problem sheets assigned in the class. An open-ended question on the students' overall experience was answered in a similar way to the personal interview surveys. Most people clearly enjoyed the projects and working with Ubermag. One person criticized ambiguity in some questions and another person suggested balancing out the difficulty of the various subprojects. Two people said that they would have preferred to install a local version of Ubermag, since they viewed the online version as buggy.

## C. Feedback from Teaching Staff

Initially, the design of the five simulation projects from scratch was laborious and time-consuming for the teach-



ing assistants. Suitable problems had to be developed and example solutions needed to be tested carefully in Jupyter notebooks. Once we had a reasonable problem sheet and solution for each computational project, it became easier in the subsequent years, since everything can be reused while gradually improving possible issues that were found in previous semesters. It was noticed that even slight ambiguities in the description of the problems may lead to students being confused or getting lost in rather irrelevant details. This is one of the main reasons why we introduced the intermediate discussions when we had students working on the projects for the second time (fall 2021). One of the teaching assistants mentioned that this type of meeting was helpful for them to identify a minor mistake in the problem definition that had caused students to obtain erroneous simulation results. Despite the challenging circumstances in the fall 2020 and 2021 semesters due to the COVID-19 pandemic, we felt that the communication between the instructional team and students as well as among the students themselves was better and more intensive than in courses without such computational projects. By working on these projects, students were encouraged to actively discuss about the subject matter and ask questions to their peers as well as instructors. Therefore, the instructional team regards these computational projects as a successful realization of a technology-enabled active-learning approach and a suitable supplement to the traditional lecture format. Apart from designing the projects, leading intermediate discussions and moderating the session with the students' final presentations, the work load for the teachers was not exceedingly high compared to other courses. Therefore, we have decided to further pursue and optimize this approach.

Overall the simulation projects (see supplementary material [68]) were designed to relate directly to current research topics, as well as distinct topics of the regular lectures. For example, synthetic antiferromagnets are integral components of modern spintronics devices to minimize complications from stray fields. Similarly, the complex gyrotropic nature of magnetization dynamics gives rise to complex phenomena such as Walker breakdown for moving domain walls, or distinct chiralities in the motion of topological solitons such as vortices and skyrmions.

Unfortunately, given the extended timeframe of the micromagnetic simulation projects lasting two months, it was not always possible to align the projects well with when these concepts are discussed in class. We consider the use of small example notebooks and shorter homework problems to achieve that alignment in the future. For instance, at the moment the projects do not include any current-driven dynamics, since spin transfer torques are only discussed in class after the conclusion of the micromagnetic simulation projects.

## D. Summary feedback

In conclusion, the micromagnetic simulation projects have seen a mostly positive perception by the students that were interviewed. This leads us to the hypothesis that such computational projects help students to obtain a better understanding and intuition of the underlying physics.

Computational micromagnetics with Ubermag constitutes a suitable approach to strengthen and develop students' programming skills while at the same time facilitating a more in-depth understanding of the subject matter. We hope to conduct a more quantitative investigation in the future to examine the hypothesis that micromagnetic simulation projects based on Ubermag are a valuable addition to the syllabus of a course on magnetic materials and applications.

## V. RECOMMENDATIONS FOR COMPUTATIONAL PROJECTS

The feedback we have obtained from students and teachers on the course suggests that computational projects and computational problem solving can improve the learning experience and effectiveness. In this section, we discuss the teaching setup with the objective to extract insights that could be used and built on in other subjects (i.e., outside micromagnetics and materials science more generally).

We want to comment on three points here: the choice of programming language (V A), the opportunities from the Jupyter Notebook for use in education (V B), and aspects of the Ubermag design that are beneficial for its use in teaching (V C).

### A. Choice of Programming Language

The use of Python as the language to both drive the simulation and to carry out the data analysis appears to be a good choice. The Python language is widely used, some students did know it already, others will benefit from learning about it. It can be argued that out of the widely used languages, Python is one that is easy to learn yet very powerful [72, 73]. Of particular relevance is the wide set of libraries available for science and engineering — including sophisticated data analysis and data visualization tools — which can be used immediately in Python.

The use of the *same* language to direct the computation (here using a Python interface to an existing simulation package) and the data analysis is also useful, as it reduces the cognitive load on the students.

## B. Project Jupyter Tools for Education

The Jupyter notebook [74] emerged from the Interactive Python (IPython) [75] environment. A recent review [52] 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 defacto standard in data science [76], and is increasingly used in science for data exploration and analysis [77]. Students can benefit in similar ways as data scientists and scientists from the Jupyter notebook. Jupyter notebooks are increasingly used in educational settings [78, 79].

### 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. Part of this is that the notebook captures exactly the protocol that was used (i.e., order of commands for simulation and analysis) to achieve a certain result [80]. 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 [52] 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 towards starting the report writing. We know from research settings that using a notebook as the document to create a report or publication provides good reproducibility [80].

We find that interactive Jupyter notebooks (using the programming language Python in our study), represent an effective tool for students to craft and execute numerical simulations individually or in groups and thereby develop a functional understanding of STEM topics.

### 2. Binder can provide the computational environment

In our study, we have made use of the Binder software [81], which is part of Project Jupyter. There is a service running the Binder software that is publicly accessible and free to use at [mybinder.org](https://mybinder.org), and which we have used for our teaching delivery.

The Binder software takes the URL of a data repository [82], scans the repository for files that specify which software is required to install, installs this software — together with a Jupyter notebook server — in a Docker

container, and then connects the notebook server with the user's browser on the user's computer. The user's computer is connected to the Binder service via the network. All the technical steps described above are not visible to the user: after selecting the appropriate URL [83], it takes a couple of minutes until the desired notebooks session appears in the user's 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. This is in particular useful because (i) they do not need to install any software and (ii) they can use a variety of computer devices as long as a web browser is available.

The public MyBinder service comes with some limitations: For example a notebook session that is idle (i.e. no computation and no user activity) for 10 minutes will be stopped from the MyBinder site and all changes created in the notebooks and on the file system of the container 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). However, for ongoing and extended studies a local install of Ubermag on the student's computer would be more convenient.

Furthermore, the computing hardware offered by MyBinder is relatively weak (for example at most two CPU cores). Those measures of restricted session length and restricted CPU cores are taken by the MyBinder team to keep the global service they offer as economic as possible. MyBinder is financed through donations [84] and sponsorship from companies [85].

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, while others have installed the Ubermag software on their own computer. 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 run their own Binder service: the BinderHub [81] is designed for this. However, the skills required to set this up exceed those of most academics, and help from the local computing or IT team is likely to be required. A local install of Ubermag is possible, and some students have chosen to follow this path.

## C. Using Simulation Software in Education — Design of Ubermag

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 try to describe which aspects of the Ubermag design have been helpful. Ubermag provides a separation of multiple concerns which is beneficial for the systematic teaching and learning of computational science. Before we provide a more detailed analysis, we need to provide some context.

### 1. Concepts in simulation driven problem solving

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

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

1. Decide on physics approximations and the model to be used (this determines the relevant equations): Within the Ubermag framework, the user selects the relevant physics through 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 [19].
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 (currently OOMMF and mumax<sup>3</sup>). Using the

OOMMF Calculator (`oommf`) or the mumax<sup>3</sup> Calculator (`mumax3c`) Python package, Ubermag then delegates the actual numerical solution to OOMMF or mumax<sup>3</sup>.

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. Specifying the simulation problem to solve

Many simulation packages show similar traits: They may have developed over many years or decades; often by scientists who may not have expected that the software would be used for decades. The user interface is often some kind of configuration file. Sometimes the configuration file uses the syntax of a language or protocol that was fashionable at the time of inception of the simulation package. Or the configuration file uses a set of syntax rules that were invented by the simulation package developers; sometimes being extended over time as the capabilities of the package grew beyond initial plans.

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 through the 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.

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 appear to 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<sup>3</sup> 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 a “user-friendly” high-level interface to existing simulation software — is certainly transferable to other domains. At least in the domain of atomic simulation there is a related effort (the atomic simulation environment ASE) which also provides a (Python-based) abstraction layer and can delegate computation to multiple simulation packages [39].

#### D. Recommendation Summary

We recommend the use of the Jupyter notebook: It helps thinking and documenting the studies, and encourages experimentation. Using the notebook, there is the option to provide a zero-install computational environment using Binder. The choice of Python for the data analysis seems reasonable as there are many libraries that allow students to achieve a lot with very few commands, the language is wide spread, easy to learn and powerful if required. Ubermag’s design helps to understand the different steps taken in computational science, going from a real-world problem to a particular numerical solution. For the use of simulation software, we see value in Ubermag’s approach of providing a high-level instruction language to drive the simulation packages; in particular where the packages may require a very specific syntax otherwise.

### VI. CONCLUSIONS

We report from the modification of an existing teaching module in higher education where we have introduced group projects based on the use of computer simulation software (Sec. III): The Ubermag micromagnetic simulation environment has been used to solve self-paced group project challenges in the area of magnetic materials education and research.

Qualified teaching assistants (e.g., experienced PhD students) or even a dedicated staff member can support the course instructor with the implementation of such projects. At the Department of Materials Science and Engineering at UIUC, the computational curriculum is supported by such a dedicated assistant who provides support for the development of modules, computational lecture delivery and supplementary office hours [20, 24]. More generally, the growing movement of Research Software Engineers at Universities [70] may provide opportunities to contribute software expertise into design and delivery of teaching activities with computational components.

For the magnetic materials course under consideration here, the course instructor’s research group typically has one or two scientists that utilize micromagnetic simulations in their own research. Therefore, they can also contribute to the success of the computational projects in the course on magnetic materials and applications.

Feedback from students and teachers suggests the change was an improvement to the learning experience (Sec. IV). Students value the ability to explore a complex system (that is the magnetic material here) through the medium of computer simulation, where properties of the material and the environment can be changed arbitrarily by the student, and the response of the material can be investigated (by executing the simulation calculation). Students also reported that the programming skills they have developed along the way are an impor-

tant enhancement of their skill set. Teachers enjoy the enthusiasm of the students, and the possibility to align the student exercises — albeit at different levels — with topics of current research questions by using a simulation tool used by researchers routinely.

Attempting to look beyond the particular study and generalize, we make the following observations (Sec. V):

- Simulation can provide a virtual experiment and thus offer the learning experience of carrying out experiments in order to better understand a system.
- Simulations are often more controllable than a real experiment (within limitations of the underlying model assumption), cheaper and safer.
- It is important to empower the students to control the simulation effectively within the time frame available in the educational setting:
  - Fine grained control of parameters is good.
  - Simulation configuration through written commands or configuration files are preferred over use of graphical user interfaces (GUIs). This way all inputs are recorded, reproducible and can be retrospectively linked to the outputs.
  - The cognitive load for expressing inputs for the simulation should be kept minimal (so students can focus on the main content of the module rather than the — potentially complex — syntax of the simulation package).
- We have used Python as the programming language (Sec. V A):
  - Python is perceived as easy to learn by scientists; yet it is a very powerful language.
  - There are many scientific libraries — this makes the use of the language much more effective.
  - The use of the same language for instructing the simulation and analyzing the outputs is beneficial.
  - Python is also fashionable at the moment: Some competency in using it is a benefit to the students’ skills for the modern employment market.
- In the simulation based experiments, the Jupyter notebook (Sec. V B) can replace the traditional experiment logbook. By design, it can support the thinking and learning process.
- The Binder software — as a part of Project Jupyter — allows zero-install provisioning of the simulation environment. However, this is only practical where the simulation times are short (and thus the public

mybinder.org instance can be used), or where there is local IT know-how available to, for example, set up a university-wide BinderHub instance.

- Where a well structured interface to the simulation exists (such as in Ubermag with the breakdown of simulation packages into model definition, numerical discretization, and numerical solver), this can support achieving of the learning objectives (Sec. VC).

We note that the Jupyter-notebook based interface lends itself straightforwardly to the learning (and assessment) method of computational essays [50, 51]. We may consider this in the future if, for example, time constraints do not allow for the assignment of more extensive simulation projects, or a self-paced learning method for individual students is sought in addition to the group projects.

## ACKNOWLEDGEMENTS

The authors would like to thank Thomas Wilhelm (Goethe University Frankfurt) for fruitful discussions

on numerical methods in higher education. MLo 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. SH, SP and HF 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 RK and AH were partially supported by the NSF through the University of Illinois at 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 US Department of Energy, Office of Science, Materials Science and Engineering Division, under Contract No. DE-SC0022060. The authors have no conflicts to disclose.

- 
- [1] R. Duit and M. Tesch, On the role of the experiment in science teaching and learning – visions and the reality of instructional practice, *Proceedings of the 7th International Conference on Hands-on Science*, 17 (2010).
  - [2] B. Skuse, The third pillar, *Physics World* **32**, 40 (2019).
  - [3] M. J. Miller and P. K. Smolarkiewicz, Predicting weather, climate and extreme events, *Journal of Computational Physics* **227**, 3429 (2008).
  - [4] M. H. Zawawi, A. Saleha, A. Salwa, N. H. Hassan, N. M. Zahari, M. Z. Ramli, and Z. C. Muda, A review: Fundamentals of computational fluid dynamics (CFD), *AIP Conference Proceedings*, AIP Conference Proceedings **2030**, 020252 (2018).
  - [5] J. R. Gissinger, B. D. Jensen, and K. E. Wise, Modeling chemical reactions in classical molecular dynamics simulations, *Polymer* **128**, 211 (2017).
  - [6] J. Dehning, J. Zierenberg, F. P. Spitzner, M. Wibrall, J. P. Neto, M. Wilczek, and V. Priesemann, Inferring change points in the spread of COVID-19 reveals the effectiveness of interventions, *Science* **369**, 10.1126/science.abb9789 (2020).
  - [7] J. Weber and T. Wilhelm, The benefit of computational modelling in physics teaching: a historical overview, *European Journal of Physics* **41**, 034003 (2020).
  - [8] N. Chonacky and D. Winch, Integrating computation into the undergraduate curriculum: A vision and guidelines for future developments, *American Journal of Physics* **76**, 327 (2008).
  - [9] M. D. Caballero and L. Merner, Prevalence and nature of computational instruction in undergraduate physics programs across the United States, *Physical Review Physics Education Research* **14**, 020129 (2018).
  - [10] R. F. Martin, Undergraduate computational physics education: uneven history and promising future, *Journal of Physics: Conference Series* **759**, 012005 (2016).
  - [11] G. Kortemeyer, Incorporating computational exercises into introductory physics courses, *Journal of Physics: Conference Series* **1512**, 012025 (2020).
  - [12] K. Thornton and M. Asta, Current status and outlook of computational materials science education in the US, *Modelling and Simulation in Materials Science and Engineering* **13**, R53 (2005).
  - [13] A. L. Graves and A. D. Light, Hitting the ground running: Computational physics education to prepare students for computational physics research, *Computing in Science and Engineering* **22**, 50 (2020).
  - [14] D. M. Cook, Computation in undergraduate physics: The lawrence approach, *American Journal of Physics* **76**, 321 (2008).
  - [15] L. McNeil and P. Heron, Preparing physics students for 21st-century careers, *Physics Today* **70**, 38 (2017).
  - [16] AAPT Undergraduate Curriculum Task Force, AAPT Recommendations for Computational Physics in the Undergraduate Physics Curriculum (2016).
  - [17] P. C. Hamerski, D. McPadden, M. D. Caballero, and P. W. Irving, Students' perspectives on computational challenges in physics class, *Physical Review Physics Education Research* **18**, 020109 (2022).
  - [18] 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.

- [19] M. Beg, M. Lang, and H. Fangohr, Ubermag: Toward more effective micromagnetic workflows, *IEEE Transactions on Magnetics* **58**, 1 (2022).
- [20] 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, 2017 ASEE Annual Conference & Exposition Proceedings 10.18260/1-2-28060 (2017).
- [21] M. D. Caballero, N. Chonacky, L. Engelhardt, R. C. Hilborn, M. L. del Puerto, and K. R. Roos, PICUP: A community of teachers integrating computation into undergraduate physics courses, *The Physics Teacher* **57**, 397 (2019).
- [22] T. J. Atherton, Resource Letter CP-3: Computational physics, *American Journal of Physics* **91**, 7 (2023), <https://doi.org/10.1119/5.0106476>.
- [23] H. Fangohr, Exploiting real-time 3d visualisation to enthuse students: A case study of using visual python in engineering, in *Computational Science – ICCS 2006*, edited by V. N. Alexandrov, G. D. van Albada, P. M. A. Sloot, and J. Dongarra (Springer Berlin Heidelberg, Berlin, Heidelberg, 2006) pp. 139–146.
- [24] 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, 2018 ASEE Annual Conference & Exposition Proceedings 10.18260/1-2-30213 (2018).
- [25] N. T. Young, G. Allen, J. M. Aiken, R. Henderson, and M. D. Caballero, Identifying features predictive of faculty integrating computation into physics courses, *Physical Review Physics Education Research* **15**, 010114 (2019).
- [26] W. Christian and A. Titus, Developing web-based curricula using java physlets, *Computers in Physics* **12**, 227 (1998).
- [27] K. Perkins, W. Adams, M. Dubson, N. Finkelstein, S. Reid, C. Wieman, and R. LeMaster, PhET: Interactive simulations for teaching and learning physics, *The Physics Teacher* **44**, 18 (2006).
- [28] D. V. Schroeder, Interactive molecular dynamics, *American Journal of Physics* **83**, 210 (2015).
- [29] K. Madhavan, L. Zentner, V. Farnsworth, S. Shivarajapura, M. Zentner, N. Denny, and G. Klimeck, nanohub.org: cloud-based services for nanoscale modeling, simulation, and education, *Nanotechnology Reviews* **2**, 107 (2013).
- [30] M. Hunt, M. Koslowski, P. Kolis, M. Meyer, A. Dandekar, and C. Duarte-Cordon, Purdue ME 581-Numerical Methods in Engineering Using Jupyter Notebooks, nanoHUB 10.21981/FGKG-7P21 (2021).
- [31] T. Faltens, A. Strachan, and G. Klimeck, nanoHUB as a platform for implementing ICME simulations in research and education, *Proceedings of the 3rd World Congress on Integrated Computational Materials Engineering (ICME 2015)*, 269 (2015).
- [32] R. Mansbach, A. Ferguson, K. Kilian, J. Krogstad, C. Leal, A. Schleife, D. Trinkle, M. West, and G. Herman, Reforming an undergraduate materials science curriculum with computational modules, *Journal of Materials Education* **38**, 161 (2016).
- [33] R. Mansbach, G. Herman, M. West, D. Trinkle, A. Ferguson, and A. Schleife, WORK IN PROGRESS: Computational modules for the MatSE undergraduate curriculum, 2016 ASEE Annual Conference & Exposition Proceedings 10.18260/p.27214 (2016).
- [34] 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, 2019 ASEE Annual Conference & Exposition Proceedings 10.18260/1-2-32926 (2019).
- [35] P. Giannozzi, S. Baroni, N. Bonini, M. Calandra, R. Car, C. Cavazzoni, D. Ceresoli, G. L. Chiarotti, M. Cococcioni, I. Dabo, A. D. Corso, S. de Gironcoli, S. Fabris, G. Fratesi, R. Gebauer, U. Gerstmann, C. Gougoussis, A. Kokalj, M. Lazzeri, L. Martin-Samos, N. Marzari, F. Mauri, R. Mazzarello, S. Paolini, A. Pasquarello, L. Paulatto, C. Sbraccia, S. Scandolo, G. Sclauzero, A. P. Seitsonen, A. Smogunov, P. Umari, and R. M. Wentzcovitch, QUANTUM ESPRESSO: a modular and open-source software project for quantum simulations of materials, *Journal of Physics: Condensed Matter* **21**, 395502 (2009).
- [36] P. Giannozzi, O. Andreussi, T. Brumme, O. Bunau, M. B. Nardelli, M. Calandra, R. Car, C. Cavazzoni, D. Ceresoli, M. Cococcioni, N. Colonna, I. Carnimeo, A. D. Corso, S. de Gironcoli, P. Delugas, R. A. DiStasio, A. Ferretti, A. Floris, G. Fratesi, G. Fugallo, R. Gebauer, U. Gerstmann, F. Giustino, T. Gorni, J. Jia, M. Kawamura, H.-Y. Ko, A. Kokalj, E. Küçükbenli, M. Lazzeri, M. Marsili, N. Marzari, F. Mauri, N. L. Nguyen, H.-V. Nguyen, A. O. de-la Roza, L. Paulatto, S. Poncé, D. Rocca, R. Sabatini, B. Santra, M. Schlipf, A. P. Seitsonen, A. Smogunov, I. Timrov, T. Thonhauser, P. Umari, N. Vast, X. Wu, and S. Baroni, Advanced capabilities for materials modelling with quantum ESPRESSO, *Journal of Physics: Condensed Matter* **29**, 465901 (2017).
- [37] G. Klimeck and D. Vasileska, ABACUS and AQME: Semiconductor device and quantum mechanics education on nanoHUB.org, 2009 13th International Workshop on Computational Electronics, 1 (2009).
- [38] K. Thornton, S. Nola, R. E. Garcia, M. Asta, and G. B. Olson, Computational materials science and engineering education: A survey of trends and needs, *JOM* **61**, 12 (2009).
- [39] A. H. Larsen, J. J. Mortensen, J. Blomqvist, I. E. Castelli, R. Christensen, M. Dulak, 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, *Journal of Physics: Condensed Matter* **29**, 273002 (2017).
- [40] M. Pieper and S. Schulz, Teaching Simulation Methods with COMSOL Multiphysics, in *Proceedings of the 2014 COMSOL Conference* (Cambridge, United Kingdom, 2014).
- [41] J. Dedulle, Pedagogic using of COMSOL Multiphysics for learning Numerical Method and Numerical Modelling, in *Conférence COMSOL Multiphysics* (Paris, France, 2007).
- [42] F. Löffler, J. Faber, E. Bentivegna, T. Bode, P. Diener, R. Haas, I. Hinder, B. C. Mundim, C. D. Ott, E. Schnetter, G. Allen, M. Campanelli, and P. Laguna,

- The einstein toolkit: a community computational infrastructure for relativistic astrophysics, *Classical and Quantum Gravity* **29**, 115001 (2012).
- [43] G. Kortemeyer and A. F. Kortemeyer, The nature of collaborations on programming assignments in introductory physics courses: a case study, *European Journal of Physics* **39**, 055705 (2018).
  - [44] R. Chabay and B. Sherwood, Computational physics in the introductory calculus-based course, *American Journal of Physics* **76**, 307 (2008).
  - [45] T. Timberlake and J. E. Hasbun, Computation in classical mechanics, *American Journal of Physics* **76**, 334 (2008).
  - [46] M. D. Caballero and S. J. Pollock, A model for incorporating computation without changing the course: An example from middle-division classical mechanics, *American Journal of Physics* **82**, 231 (2014).
  - [47] M. D. Caballero, M. A. Kohlmyer, and M. F. Schatz, Implementing and assessing computational modeling in introductory mechanics, *Physical Review Special Topics - Physics Education Research* **8**, 020106 (2012).
  - [48] S. V. Samsonau, Computer simulations combined with experiments for a calculus-based physics laboratory course, *Physics Education* **53**, 055013 (2018).
  - [49] D. Sachmpazidi, M. Bautista, Z. Chajecki, C. Mendoza, and C. Henderson, Integrating numerical modeling into an introductory physics laboratory, *American Journal of Physics* **89**, 713 (2021).
  - [50] T. O. B. Odden, E. Lockwood, and M. D. Caballero, Physics computational literacy: An exploratory case study using computational essays, *Physical Review Physics Education Research* **15**, 020152 (2019).
  - [51] A. diSessa, *Changing Minds - Computers, Learning, and Literacy* (MITPress, Cambridge, MA, 2000).
  - [52] B. E. Granger and F. Perez, Jupyter: Thinking and storytelling with code and data, *Computing in Science & Engineering* **23**, 7 (2021).
  - [53] K. Fisher, Technology-Enabled Active Learning Environments, CELE Exchange, Centre for Effective Learning Environments 10.1787/5kmbjxrmc0p-en (2010).
  - [54] S. Freeman, S. L. Eddy, M. McDonough, M. K. Smith, N. Okoroafor, H. Jordt, and M. P. Wenderoth, Active learning increases student performance in science, engineering, and mathematics, *Proceedings of the National Academy of Sciences* **111**, 8410 (2014).
  - [55] R. R. Hake, Interactive-engagement versus traditional methods: A six-thousand-student survey of mechanics test data for introductory physics courses, *American Journal of Physics* **66**, 64 (1998).
  - [56] C. Hoellwarth and M. J. Moelter, The implications of a robust curriculum in introductory mechanics, *American Journal of Physics* **79**, 540 (2011).
  - [57] T. Gilbert, Classics in magnetism a phenomenological theory of damping in ferromagnetic materials, *IEEE Transactions on Magnetics* **40**, 3443 (2004).
  - [58] C. Abert, Micromagnetics and spintronics: models and numerical methods, *The European Physical Journal B* **92**, 120 (2019).
  - [59] M. Lonsky and A. Hoffmann, Dynamic excitations of chiral magnetic textures, *APL Materials* **8**, 100903 (2020).
  - [60] M. Lonsky and A. Hoffmann, Coupled skyrmion breathing modes in synthetic ferri- and antiferromagnets, *Physical Review B* **102**, 104403 (2020).
  - [61] M. Lonsky and A. Hoffmann, Dynamic fingerprints of synthetic antiferromagnet nanostructures with interfacial Dzyaloshinskii–Moriya interaction, *Journal of Applied Physics* **132**, 043903 (2022).
  - [62] M. J. Donahue and D. G. Porter, OOMMF User’s Guide, Version 1.0, Interagency Report NISTIR 6376, National Institute of Standards and Technology, Gaithersburg, MD (1999).
  - [63] A. Vansteenkiste, J. Leliaert, M. Dvornik, M. Helsen, F. Garcia-Sanchez, and B. V. Waeyenberge, The design and verification of MuMax3, *AIP Advances* **4**, 107133 (2014).
  - [64] 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).
  - [65] M. Ponsudana, R. Amuda, A. Brinda, and N. Kanimozhi, Investigation on the excitation of magnetic skyrmionium in a nanostructure, *Journal of Superconductivity and Novel Magnetism* **35**, 805 (2022).
  - [66] T. J. Hicken, M. N. Wilson, S. J. R. Holt, R. Khasanov, M. R. Lees, R. Gupta, D. Das, G. Balakrishnan, and T. Lancaster, Magnetism in the néel-skyrmion host  $\text{GaV}_4\text{S}_8$  under pressure, *Physical Review B* **105**, 134414 (2022).
  - [67] M. Xu, J. Zhang, D. Meng, Z. Zhang, and G. Jiang, The influence of introducing holes on the generation of skyrmions in nanofilms, *Physics Letters A* **433**, 128034 (2022).
  - [68] See supplementary material at <https://aapt.scitation.org/journal/ajp> for a detailed overview of five simulation projects. (link to be inserted by the journal staff).
  - [69] Ubermag source code at <https://github.com/ubermag>.
  - [70] J. Cohen, D. S. Katz, M. Barker, N. C. Hong, R. Haines, and C. Jay, The four pillars of research software engineering, *IEEE Software* **38**, 97 (2021).
  - [71] S. J. R. Holt, M. Lang, J. C. Loudon, T. J. Hicken, D. Cortés-Ortuño, S. A. Pathak, M. Beg, and H. Fangohr, Virtual experiments in computational magnetism: mag2exp, In preparation for submission (2023).
  - [72] H. Fangohr, A Comparison of C, MATLAB, and Python as Teaching Languages in Engineering, *Computational Science - ICCS 2004*, 1210 (2004).
  - [73] D. G. Balreira, T. L. T. da Silveira, and J. A. Wickboldt, Investigating the impact of adopting python and c languages for introductory engineering programming courses, *Computer Applications in Engineering Education* 10.1002/cae.22570 (2022).
  - [74] 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.
  - [75] F. Perez and B. E. Granger, IPython: A system for interactive scientific computing, *Computing in Science & Engineering* **9**, 21 (2007).
  - [76] J. M. Perkel, Why jupyter is data scientists’ computational notebook of choice, *Nature* **563**, 145 (2018).



- [77] 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. Repin, 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. Szutuk, J. Szuba, J. Taylor, S. Trojanowski, K. Wrona, A. Yaroslavtsev, and J. Zhu, Data Exploration and Analysis with Jupyter Notebooks, in *Proc. ICALEPCS’19*, International Conference on Accelerator and Large Experimental Physics Control Systems No. 17 (JACoW Publishing, Geneva, Switzerland, 2020) pp. 799–806.
- [78] 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. Willing, and M. Zingale, Teaching and Learning with Jupyter, Figshare 10.6084/m9.figshare.19608801.v1 (2022).
- [79] 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. Perez, B. Ragan-Kelley, J. Suchow, and C. Willing, nbgrader: A tool for creating and grading assignments in the Jupyter notebook, *Journal of Open Source Education* **2**, 32 (2019).
- [80] M. Beg, J. Taka, T. Kluyver, A. Konovalov, M. Ragan-Kelley, N. M. Thiery, and H. Fangohr, Using Jupyter for reproducible scientific workflows, *Computing in Science & Engineering* **23**, 36 (2021).
- [81] Project Jupyter, Matthias Bussonnier, Jessica Forde, Jeremy Freeman, Brian Granger, Tim Head, Chris Holdgraf, Kyle Kelley, Gladys Nalvarte, Andrew Osherooff, 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.
- [82] Ubermag repository <https://github.com/ubermag/tutorials> on Github.
- [83] Ubermag on mybinder.org: <https://mybinder.org/v2/gh/ubermag/tutorials/latest>.
- [84] Donations to Binder: <https://numfocus.salsalabs.org/donate-to-binder/index.html>.
- [85] Current MyBinder sponsors include Google Cloud, OVH, GESIS Notebooks and the Turing Institute, see <https://mybinder.org/>.