UNIVERSITY OF SOUTHAMPTON

Research Software Engineering in Micromagnetics

by

Marc-Antonio Bisotti

A thesis submitted for the degree of
Doctor of Philosophy

in the
Faculty of Engineering and Physical Sciences
Computational Engineering and Design

December 2020

UNIVERSITY OF SOUTHAMPTON

ABSTRACT

FACULTY OF ENGINEERING AND PHYSICAL SCIENCES
COMPUTATIONAL ENGINEERING AND DESIGN

Doctor of Philosophy

RESEARCH SOFTWARE ENGINEERING IN MICROMAGNETICS

by Marc-Antonio Bisotti

Software is fundamental to research. Seven out of ten researchers in the United Kingdom report that their work would be impossible without it [1]. Yet the scientific community faces a credibility crisis, and this is publicly known. The crisis is a multistakeholder problem where no single solution will suffice. It has been shown that code quality is strongly related to the quality of the scientific results.

The work presented in this thesis pursues a two-fold strategy. On one hand, in response to the problem just outlined, it explores concepts from software engineering impacting quality in an academic context. This includes version control, software documentation, testing, continuous integration, or software distribution. To render this exploration concrete, a field of study was chosen and two case studies were performed. Like most of the computational sciences, micromagnetics benefits greatly from fast and accurate numerical tools. Thus on the other hand, this work contributes to the body of methodology for micromagnetic simulations.

Two micromagnetic simulators have been developed in preparation of this thesis. One simulator is a finite element code called FinMag. The author created the initial implementation. The other simulator is a finite difference code called Fidimag. For Fidimag, the author participated in the initial implementation. The continuing development of these two packages is now a collaborative effort with many developers.

The two simulators enabled published, novel, reproducible research on magnonics and the study of magnetic skyrmions. Further, the outcome of two numerical studies increasing the performance of the micromagnetic simulators are presented.

[1] "S.J. Hettrick et al, UK Research Software Survey 2014", DOI:10.5281/zenodo.1183562

Contents

Nomenclature **xiii**

1 Introduction **1**

2 Software Engineering **5**
2.1 The Software Development Process 6
2.1.1 The Waterfall Model 7
2.1.2 The V-Model 8
2.1.3 Agile Methods 9
2.2 Software Design and Architecture 12
2.3 Software Testing 13

3 Simulations of Ferromagnetic Nanodevices **15**
3.1 Magnetism at the Nanoscale 15
3.2 The Landau-Lifshitz-Gilbert (LLG) Equation 17
3.3 Energy Contributions to the LLG Equation 19
3.3.1 Exchange Energy 20
3.3.2 Demagnetisation Energy 20
3.3.3 Dzyaloshinskii-Moriya energy 22
3.3.4 Magnetocrystalline Anisotropy Energy 24
3.3.5 Zeeman Energy 24
3.3.6 Thermal Fluctuations 25
3.4 Spin-Transfer Torque 25
3.4.1 The Giant Magnetoresistance Effect 25
3.4.2 Spin-Transfer Torque extension of the LLG Equation 26
3.4.3 Spin-Torque Oscillators 29
3.5 Numerical Methods 32
3.5.1 The Finite Differences Method 32
3.5.2 The Finite Elements Method 34
3.5.3 Linear Systems of Equations 37
3.5.4 Time Integration of the LLG Equation 40
3.5.5 Nudged Elastic Band Method 42
3.6 Micromagnetic Standard Problems 43

4 Fidimag – a finite difference software for atomistic and micromagnetic simulations **45**
4.1 Finite Difference Simulations of Magnetic Materials 46
4.2 Implementation 50
4.3 Software Testing 55
4.4 Solution to Standard Problem #4 56
4.5 Time Integration of the LLG in Finite Difference Schemes 59

5 FinMag – a finite element software for micromagnetic simulations 69
5.1 Introduction . . . 69
5.2 Implementation of the Dzyaloshinskii-Moriya Energy . . . 71
5.3 Assessing FinMag's Reliability . . . 72
5.4 Solution to Standard Problem #3 . . . 78
5.5 Performance of Krylov Methods in Demagnetising Field Computation . . . 80
5.6 Use of FinMag in Research . . . 84

6 Research Software Engineering 91
6.1 Research Software Engineering in an Academic Setting . . . 91
6.1.1 Source Control . . . 92
6.1.2 Agile Method . . . 93
6.2 Software Quality Assurance . . . 93
6.2.1 Software Testing . . . 93
6.2.2 Continuous Integration with Jenkins and Travis CI . . . 95
6.3 Software Documentation with Jupyter Notebooks . . . 96
6.4 Software Distribution using Docker . . . 97
6.5 Reproducible Science . . . 98

7 Conclusion 101

References 105

List of Figures

2.1 The waterfall model. Validation and verification make up the feedback step. . . . 8
2.2 The V model. . . . 9

3.1 The precession and damping of the magnetic moment $\mathbf{M}$ in $\mathbf{H}_{\text{eff}}$. . . . 19
3.2 Rendering of skyrmions. Hairy sphere on the top and "vortex" on the bottom. The latter object being what we identify as a skyrmion in solid-state physics. Reproduced from [1]. . . . 23
3.3 Trilayer system with a non-magnetic metal between two ferromagnetic layers in two possible configurations with different conductivity. (left) In the absence of an external field, the system relaxes into an antiparallel configuration. (right) The two magnetisations are parallel when an external field is applied. . . . 26
3.4 Schematics of a typical spin valve device. A thin non-ferromagnetic spacer is sandwiched between two ferromagnetic layers of different thickness. The magnetisation in the thinner free layer is inclined by an angle of θ in respect to the magnetisation in the thicker fixed layer. . . . 27
3.5 The precession and damping of the magnetic moment $\mathbf{M}$ in $\mathbf{H}_{\text{eff}}$ of Figure 3.1 with added spin-transfer torque. Depending on the sign of the current, the spin-transfer torque either reinforces the damping, or acts in an opposite direction (shown). . 28
3.6 Schematics of a bilayer spin-torque oscillator (reproduced from [2]). When magnetic damping is compensated by the spin torque resulting from a spin-polarised current flowing from the fixed layer, the free layer magnetisation M precesses around the direction of an applied magnetic field H0. . . . 30
3.7 A sphere in a regular grid made of $20 \times 20 \times 20 = 8000$ cells. . . . 34
3.8 A triangulated sphere. The mesh is comprised of 736 volume elements. . . . 35

4.1 Architecture of Fidimag as reflected by the directory structure and Python modules. Code useful to micromagnetic and atomistic simulations was extracted into a `common` namespace. The red arrows point to user-facing parts of the system. . . . 51
4.2 The cuboid mesh (a) is used in the continuum approximation. In the atomistic case, the ordering of the atoms or molecules is given by the crystal structure of the magnetic solid. The meshes then represent the 2D/3D square lattice (a) and the 2D hexagonal lattice (b). . . . 52
4.3 The initial unit magnetisation of the bar of Permalloy described in standard problem #4. . . . 58
4.4 The magnetisation configuration at the moment of the reversal of the average magnetisation. . . . 59
4.5 The magnetisation after a nanosecond of simulation time has elapsed. . . . 59
4.6 The components of the average magnetisation over time as computed with Fidimag and by OOMMF. . . . 59
4.7 Comparison of the average magnetisation as computed by OOMMF and Fidimag for standard problem #4. . . . 63
4.8 Number of evaluations of the LLG equation during the simulation of dynamical part of standard problem #4 for `RTOL` = `ATOL` = 10^{-8}. `max_ord` controls the maximum order of the BDF. . . . 64

4.9 Number of evaluations of the LLG equation during the simulation of dynamical part of standard problem #4 for RTOL = ATOL = 10^{-10}. 65
4.10 Walltime elapsed during the simulations for RTOL = ATOL = 10^{-8} for each of the integrators. 66
4.11 Walltime elapsed during the simulations for RTOL = ATOL = 10^{-10} for each of the integrators. 67

5.1 Starting magnetisation $\boldsymbol{m}$ for the exchange field comparison. 73
5.2 Starting magnetisation $\boldsymbol{m}$ for the anisotropy field comparison. 74
5.3 Convergence of FinMag's and OOMMF's solution for fine meshes. 75
5.4 Starting magnetisation $\boldsymbol{m}$ for the demagnetising field comparison. 75
5.5 Time development of the macrospin for two values of the damping parameter α. The plots show FinMag's solution drawn with markers and the analytical solution as lines. 77
5.6 Example with exchange and demagnetising field. Average unit magnetisation $\boldsymbol{m}$ over time for FinMag and Nmag. 78
5.7 Example with exchange and demagnetising field. Exchange energy density through the x-axis of the bar. 78
5.8 Example with exchange and demagnetising field. Demagnetisation energy density through the x-axis of the bar. 79
5.9 Time spent solving the first linear system for the magnetised sphere. For each solver, the runtime for the fastest, slowest, and default preconditioners are shown, along with a run without preconditioner. 83
5.10 Time spent solving the second linear system for the magnetised sphere. 83
5.11 Time spent solving the first linear system for the thin film. For each solver, the runtime for the fastest, slowest, and default preconditioners are shown, along with a run without preconditioner. 84
5.12 Time spent solving the second linear system for the thin film. 84
5.13 Geometry used for the simulations in Ref. [3], composed of a magnetic nanoparticle and an elliptical magnetic disc. Note the empty space between the particle and the disc, which still needs to be accounted for by the simulation code. Image from Ref. [3] reproduced with permission. 88

6.1 Screenshot from pivotaltracker.com project management tool. 94

List of Tables

2.1 Summary of the 15 knowledge areas identified by the software body of knowledge. 6

4.1 Butcher tableau of the forward Euler method. 61
4.2 Butcher tableau of the classic Runge-Kutta method. Using the third row of the tableau as an example, k_3 is obtained by evaluating f at $t_n + \frac{1}{2}h$ ($\frac{1}{2}$ in leftmost column), $y_n + \frac{1}{2}hk_2$ (0 indicating no contribution from k_1). 61
4.3 Butcher tableau of the Runge-Kutta method due to Dormand and Prince. The first row below the line yields the fifth-order solution. The error estimate is computed by subtracting an alternative solution given by the second row below the line. . . 62
4.4 Complete list of numerical integrators used in study. 62

5.1 Comparison of the exchange field computed with FinMag against Nmag and OOMMF. 73
5.2 Comparison of the anisotropy field computed with FinMag against Nmag, Magpar and OOMMF. 74
5.3 Comparison of the demagnetising field computed with FinMag, Nmag, and Magpar against the analytical solution followed by a comparison between FinMag and Nmag and Magpar. 75
5.4 Comparison of the solution of the LLG equation of FinMag with OOMMF. . . . 76
5.5 Single domain limit L for standard problem 3. Contrary to what is said in the problem definition, J. Martins found that L does depend on the discretisation size [4]. 79
5.6 Energy contributions in the flower and vortex state in the standard problem 3. The energies are relative to K_m. 79
5.7 Components of the spatially averaged unit magnetisation $\boldsymbol{m}$ in the single domain limit. Empty cells have the value 0. While the magnetisation points in the direction of the easy axis in the flower state, the vortex state has two possible configurations. 79
5.8 List of linear solvers used in the numerical study of magnetised sphere and thin film. 80
5.9 List of preconditioners used in the numerical study of magnetised sphere and thin film. 80
5.10 Time spent solving the first linear system for the magnetised sphere as a function of preconditioner (rows) and Krylov subspace method (columns). 81
5.11 Time spent solving the second linear system for the magnetised sphere as a function of preconditioner (rows) and Krylov subspace method (columns). 81
5.12 Time spent solving the first linear system for the thin film as a function of preconditioner (rows) and Krylov subspace method (columns). 82
5.13 Time spent solving the second linear system for the thin film as a function of preconditioner (rows) and Krylov subspace method (columns). 82

6.1 Modules with software engineering content in selected programmes of the University of Southampton [5]. 92
6.2 Lines of source code for Fidimag, FinMag, Nmag, and OOMMF. Data generated with cloc version 1.74. 93

6.3 Number of test cases in source codes of Fidimag, FinMag, Nmag, and OOMMF. Counted using `grep -R def 'test_' | wc -l` for Fidimag, FinMag, and Nmag, and by counting for OOMMF. 95
6.4 Documentation included with Fidimag, FinMag, Nmag, and OOMMF. The Fidimag manual, by incorporating Jupyter notebooks, also includes the standard output of its contained programs, somewhat inflating its page count. 97

List of Algorithms

4.1 Solution to the standard problem #3. The code that computes the energy difference between the two possible magnetic configurations has been abstracted into the function `energy_difference` in the module `cube_sim` that is imported in line 6. The function `energy_difference` can then be handed off to a bisect method provided by SciPy to find the cube size for which both configurations have the same energy. 49

4.2 Definition of a 2 nm wide sphere geometry for a micromagnetic Fidimag simulation. 53

4.3 A unit test in which a scalar field `s` is created on a hexagonal mesh, saved to a VTK file and compared to a reference file. 55

4.4 Solution to the standard problem #4. The simulation defined in the function `compute_initial_magnetisation` returns the relaxed magnetisation configuration in absence of an applied magnetic field. In the function `compute_dynamics` that makes up the second simulation, the magnetic field that causes the magnetisation reversal is added on line 35. 57

4.5 Solution to the standard problem #4 (cont.) The routines defined in Listing 4.4 are called to carry out the simulation. The snapshots in Figures 4.3 to 4.5 were made using the function `plot_quiver` defined on line 50. 58

4.6 Python implemention of the classic Runge-Kutta method of fourth order in Fidimag. 61

5.1 Implementation of the Dzyaloshinskii-Moriya interaction. FinMag provides several numerical methods to compute the field from the definition of the energy density, one of which was chosen as default in the class's initialising method. 71

5.2 Creation of a simulation and addition of the DMI to it. 71

5.3 A snippet of the code used to create FinMag simulations in Ref. [6] showing the programmatic mesh creation. FinMag uses the NETGEN mesh generator internally [7]. The simulations are created in a function called `reversal_sim`. Some of its parameters have been ommited for brevity. 86

6.1 Dockerfile for the creation of a container image of Fidimag. 99

Nomenclature

A	exchange energy constant
α	Gilbert damping constant
$\mathbf{B}$	magnetic flux density (or magnetic field)
$\mathbf{d}$	Dzyaloshinskii vector
e	elementary charge
ϵ_{jkl}	Levi-Civita symbol
E	energy
FMR	ferromagnetic resonance
GMR	giant magnetoresistance
γ	gyromagnetic ratio
$\hbar$	(reduced) Planck constant
$\mathbf{H}$	magnetic field
$\mathbf{H}_{\text{eff}}$	effective field
J	exchange energy constant
k	uniaxial anisotropy constant
k_B	Boltzmann constant
LLG	Landau-Lifshitz-Gilbert (equation)
λ	Landau-Lifshitz damping constant
l_{ex}	exchange length
m_e	electron mass
$\mathbf{M}$	magnetisation
$\mathbf{m}$	unit magnetisation
M_{S}	saturation magnetisation
μ	magnetic moment
μ_B	Bohr magneton
μ_0	vacuum permeability
ϕ	magnetic scalar potential
R	resistance
STT	spin-transfer torque
$\mathbf{S}$	spin
t	time
$\hat{\mathbf{u}}$	uniaxial anisotropy axis
w	energy density

Research Contributions

Marijan Beg, Ryan A. Pepper, David Corts-Ortuo, Bilal Atie, **Marc-Antonio Bisotti**, Gary Downing, Thomas Kluyver, Ondrej Hovorka, and Hans Fangohr. Stable and manipulable Bloch point. *Scientific Reports* 9, 7959, (2019).

Marc-Antonio Bisotti, David Cortés-Ortuño, Ryan Pepper, Weiwei Wang, Marijan Beg, and Hans Fangohr. Fidimag – a finite difference atomistic and micromagnetic simulation package. *Journal of Open Research Software*, 6(1), 22 (2018).

Marc-Antonio Bisotti, Marijan Beg, Weiwei Wang, Maximilian Albert, Dmitri Chernyshenko, David Cortés-Ortuño, Ryan Pepper, Mark Vousden, Rebecca Carey, Hagen Fuchs, Anders Johansen, Gabriel Balaban, and Hans Fangohr. FinMag – finite-element micromagnetic simulation tool. DOI: http://doi.org/10.5281/zenodo.1216011 (2018).

Ryan Pepper, Marijan Beg, David Cortés-Ortuño, Thomas Kluyver, **Marc-Antonio Bisotti**, Rebecca Carey, Mark Vousden, Maximilian Albert, Weiwei Wang, Ondrej Hovorka, and Hans Fangohr. Skyrmion states in thin confined polygonal nanostructures. *Journal of Applied Physics* 123, 093903 (2018).

Benny Malengier, Pavol Kišon, James Tocknell, Claas Abert, Florian Bruckner and **Marc-Antonio Bisotti**. ODES : a high level interface to ODE and DAE solvers. *Journal of Open Source Software* 3, 22 (2018).

Alexander Baker, Marijan Beg, Gregory Ashton, Maximilian Albert, Dmitri Chernyshenko, Weiwei Wang, Shilei Zhang, **Marc-Antonio Bisotti**, Matteo Franchin, Chun Lian Hu, Robert Stamps, Thorsten Hesjedal, and Hans Fangohr. Proposal of a micromagnetic standard problem for ferromagnetic resonance simulations. *Journal of Magnetism and Magnetic Materials* 421, 428-439 (2017).

Marijan Beg, Maximilian Albert, **Marc-Antonio Bisotti**, David Cortés-Ortuño, Weiwei Wang, Rebecca Carey, Mark Vousden, Ondrej Hovorka, Chiara Ciccarelli, Charles S. Spencer, Christopher H. Marrows, and Hans Fangohr. Dynamics of skyrmionic states in confined helimagnetic nanostructures. *Physical Review B* 95, 014433 (2017).

David Cortés-Ortuño, Weiwei Wang, Marijan Beg, **Marc-Antonio Bisotti**, Ryan Pepper, Rebecca Carey, Mark Vousden, Thomas Kluyver, Ondrej Hovorka, and Hans Fangohr. Thermal stability and topological protection of skyrmions in nanotracks. *Scientific Reports* 7, 4060 (2017).

Mark Vousden, Maximilian Albert, Marijan Beg, **Marc-Antonio Bisotti**, Rebecca Carey, Dmitri Chernyshenko, David Cortés-Ortuño, Weiwei Wang, Ondrej Hovorka, Christopher H. Marrows,

and Hans Fangohr. Skyrmions in thin films with easy-plane magnetocrystalline anisotropy. *Applied Physics Letters* 108, 132406 (2016).

Rebecca Carey, Marijan Beg, Maximilian Albert, **Marc-Antonio Bisotti**, David Cortés-Ortuño, Mark Vousden, Weiwei Wang, Ondrej Hovorka, and Hans Fangohr. Hysteresis of nanocylinders with Dzyaloshinskii-Moriya interaction. *Applied Physics Letters* 109, 122401 (2016).

Maximilian Albert, Marijan Beg, Dmitri Chernyshenko, **Marc-Antonio Bisotti**, Rebecca Carey, Hans Fangohr, Peter J. Metaxas. Frequency-based nanoparticle sensing over large field ranges using the ferromagnetic resonances of a magnetic nanodisc. *Nanotechnology* 27, 455502 (2016).

Mark Vousden, **Marc-Antonio Bisotti**, Maximilian Albert, Hans Fangohr. Virtual Micromagnetics: A Framework for Accessible and Reproducible Micromagnetic Simulation. *Journal of Open Research Software* 4:e41 (2016).

Weiwei Wang, Maximilian Albert, Marijan Beg, **Marc-Antonio Bisotti**, Dmitri Chernyshenko, David Cortés-Ortuño, Ian Hawke, and Hans Fangohr. Magnon driven domain wall motion with the Dzyaloshinskii-Moriya interaction. *Physical Review Letters* 114, 087203 (2015).

Weiwei Wang, Mykola Dvornik, **Marc-Antonio Bisotti**, Dmitri Chernyshenko, Marijan Beg, Maximilian Albert, Arne Vansteenkiste, Bartel V. Waeyenberge, Andriy N. Kuchko, Volodymyr V. Kruglyak, and Hans Fangohr. Phenomenological description of the nonlocal magnetization relaxation in magnonics, spintronics, and domain-wall dynamics. *Phys Rev B* 92, 054430 (2015).

Marijan Beg, Rebecca Carey, Weiwei Wang, David Cortés-Ortuño, Mark Vousden, **Marc-Antonio Bisotti**, Maximilian Albert, Dmitri Chernyshenko, Ondrej Hovorka, Robert L. Stamps, and Hans Fangohr. Ground state search, hysteretic behaviour, and reversal mechanism of skyrmionic textures in confined helimagnetic nanostructures. *Scientific Reports* 5, 17137 (2015).

Declaration of Authorship

I, Marc-Antonio Bisotti, declare that this thesis entitled Research Software Engineering in Micromagnetics and the work presented in it are my own and has been generated by me as the result of my own original research.

I confirm that:

1. This work was done wholly or mainly while in candidature for a research degree at this University;
2. Where any part of this thesis has previously been submitted for a degree or any other qualification at this University or any other institution, this has been clearly stated;
3. Where I have consulted the published work of others, this is always clearly attributed;
4. Where I have quoted from the work of others, the source is always given. With the exception of such quotations, this thesis is entirely my own work;
5. I have acknowledged all main sources of help;
6. Where the thesis is based on work done by myself jointly with others, I have made clear exactly what was done by others and what I have contributed myself;
7. Chapter 4 is based on work published as "Marc-Antonio Bisotti, David Cortés-Ortuño, Ryan Pepper, Weiwei Wang, Marijan Beg, Thomas Kluyver, Hans Fangohr (2018). Fidimag – a finite difference atomistic and micromagnetic simulation package. Journal of Open Research Software, 6(1), 22."

Signed:

Date:

Acknowledgements

First of all, I would like to express my sincere gratitude to my supervisor Prof Hans Fangohr for his support and guidance throughout the duration of my PhD. Similarly, I would like to thank Dr Ian Hawke, the late Prof Peter de Groot, and Prof Simon Cox for agreeing to be my co-supervisors and for their advice.

To my colleagues Dr Maximilian Albert, Dr Marijan Beg, Rebecca Carey, Dr Dmitri Chernyshenko, Dr David Cortés-Ortuño, Dr Ryan Pepper, Dr Mark Vousden, and Dr Weiwei Wang – thank for you for making our office a nice and welcoming place to work. I enjoyed the many fruitful discussions I had with you about physics, software engineering and the world at large. Your companionship and friendship over the years means a lot to me. Dr Matteo Franchin, thank you for patiently answering the many questions I first had about Nmag.

To my wife Dr Carmen Bisotti, thank you for always being there for me.

Last but not least, I am indebted to my family for their unwavering belief in me over the years. Without their unconditional love and support I would never have been able to embark on, or complete, the work presented here.

Chapter 1

Introduction

Due to advances in technology and the rapid spread of computational methods across the research landscape, reproducibility has become an important issue for recent research [8]. Reproducible research means that published results can be verified and subjected to alternative analyses. This requires that the data gathered by the scientist and the computer code that reproduces the findings be made available to others. In a more general sense replicability counts among the main beliefs of the scientific method. And yet, the scientific community faces a reproducibility crisis, the echoes of which have reached mainstream media [9]. A survey of 1500 scientists conducted by Nature in 2016 found that 70% of them had failed to reproduce at least one other's scientist experiment, with the exact numbers differing between disciplines [10]. Roughly 40% of respondents agreed that either the raw data or the code being unavailable feature among the top-rated factors contributing to irreproducible research. Indeed, the situation is not better in disciplines where reproducibility should in theory be easiest to achieve, such as computer science. Collberg et al. report that having examined 402 papers from ACM conferences and journals backed by code, roughly half included code that could not be made to compile or run [11].

Begley and Ioannidis remark that new data and publications are being generated at an unprecedented rate [12]. Further, they offer evidence that the majority of these findings will not stand the test of time. The desperation to "publish or perish" and failure to adhere to good scientific practice are offered as possible causes for this. They conclude that the current predicament is a multistakeholder problem where no single solution will suffice.

Research software is fundamental to research; seven out of ten researchers in the United Kingdom report that their work would be impossible without it [13]. A large number of competing and inter-related constraints characterise the software development process. The classic example being the "Quality Triangle Dilemma" [14], or trilemma, between fast, cheap, and good software development. Where a software is optimised for performance, there is also a continuing tradeoff with source legibility and software maintainability. In a more academic context it might be the balancing act of developing a software that is novel, yet also comes with enough features to be genuinely useful. Some of the constraints are clearly defined, while others are poorly specified and notoriously vague.

Balancing the concerns is difficult, as changes to one parameter can have a large impact upon several related areas. As Clarke et al. note, this is what makes software engineering, engineering [15]. There is no single solution to a typical set of software engineering constraints. Instead, the software engineer faces problems which consist not in finding the solution, but rather in engineering an acceptable or near optimal solution from a large number of alternatives. An emphasis on good research software engineering is however poorly incentivised due to a lack of recognition for the value of the software itself, and an underestimation of its long life cycle. The accumulated knowledge of the researchers that is embodied in the software makes it a long-time investment [16]. This problem is compounded by the fact that computational scientists draw from a variety of scientific and engineering disciplines, but are generally not trained software engineers.

In 1994, Hatton and Roberts [17] published measurements of defects and accuracy in seismic data processing packages and showed that software quality is strongly related to the quality of the scientific results. Software quality models such as ISO/IEC 25010:2011 [18] exist, that break up software quality into a number of characteristics such as the functional suitability assessed by Hatton and Roberts. Yet, many years after their study, how to raise the quality of scientific software is still an open question [19, 20, 21]. Publishing source code along with scientific articles is certainly one step to ensure reproducibility of in silico experiments, but a case can be made that the computational sciences further stand to gain from adopting more of the practices and ideas of modern software engineering. We believe this approach is necessary to tackle the underlying problems of both the productivity and credibility crises.

To put the ideas presented to test, the approach outlined in the last paragraph must be applied to a particular research area. In the work presented in this thesis, the field of study is computational micromagnetics. In micromagnetics, ferromagnetic nanostructures are studied with regard to applications in data storage, as an ever increasing need for data storage results in great challenges for the development of devices that are cheap, reliable and have a high-capacity. The limitations of existing technologies such as perpendicular recording media is known as the magnetic recording trilemma [22]: a trade-off between the conflicting requirements of low energy required to imprint information, high thermal stability and high signal-to-noise ratio. The dynamics of ferromagnetic samples can only be analytically solved for a very limited number of systems. The complexity of common problems requires the use of micromagnetic simulation codes such as OOMMF [23], Mumax [24], Micromagnum [25, 26], and Micromagus [27], which use the finite difference approach to spatial discretisation, and Nmag [28] and Magpar [29], which use a finite element approach. Since the creation of many of the cited codes, the state of the open-source landscape in scientific computing has changed significantly, for example thanks to efforts such as the FEniCS Project [30] for solving differential equations, or Docker [31] for software distribution. By leveraging open-source softwares a code can be kept smaller, thereby lowering the defect rate and the maintenance burden. The best practices around software testing have evolved as well. Software tests are incorporated from the start instead of as an afterthought, and automated to a high degree, to avoid relying on the discipline of the developers. Software quality can be further raised by chosing an appropriate programming model. If instead of being created interactively or using an ad-hoc configuration language, simulation scripts are programs in their own right (following a code as configuration approach) they can reproduced more easily and tested.

Two micromagnetic packages have been developed to support the research described in this thesis. One is a finite element code called FinMag. The author created the initial implementation. The other package is a finite difference code called Fidimag. For Fidimag, the author participated in the initial implementation. A report on the simulator was published in preparation of this thesis [32]. The continuing development of these two packages is now a collaborative effort with many developers. For both codes the author is a leading contributor by number of lines, and number of commits in version control.

In micromagnetic systems, complex behaviour emerges as a result of the dynamic interplay of the energies involved. They will be presented briefly. First, the exchange energy, which for two neighbouring magnetic moments (spins) $\mathbf{S}_i$ and $\mathbf{S}_j$ is equal to $-J\mathbf{S}_i \cdot \mathbf{S}_j$, where J is the exchange energy constant [33]. If we assume that J is positive, we can see that the exchange energy favours a parallel alignment of the two moments in what is called the ferromagnetic order. The exchange interaction does not have a preferential direction in space. The magnetic moments will align with an external magnetic field $\mathbf{B}$ and the corresponding energy for a single magnetic moment μ is $-\mu\mathbf{B}$ [33]. Another source of a preferential direction is introduced by the magnetocrystalline anisotropy. One type of magnetocrystalline anisotropy is the uniaxial anisotropy. With the uniaxial anisotropy constant k, the energy for a spin reads $-k\left(\mathbf{S}\cdot\hat{\mathbf{u}}\right)^2$, where $\hat{\mathbf{u}}$ is the uniaxial anisotropy axis [34]. Depending on the anisotropy constant, two different kinds of uniaxial anisotropy are possible. If $k > 0$, the energy is minimised when the spin is either parallel or antiparallel to $\hat{\mathbf{u}}$. This is the easy-axis anisotropy. For $k < 0$, the energy is minimised for a spin perpendicular to $\hat{\mathbf{u}}$ in what is called easy-plane anisotropy. Certain magnetic systems that lack some type of inversion symmetry exhibit the Dzyaloshinskii-Moriya interaction. For two neighbouring spins, its energy is $\mathbf{d}\cdot(\mathbf{S}_i \times \mathbf{S}_j)$, where $\mathbf{d}$ is the Dzyaloshinskii vector [35]. It favours a perpendicular arrangement of neighbouring spins and as such, competes with the exchange energy that favours a parallel alignment. The compromise is a helical modulation. For the right set of parameters J, k, B and $\mathbf{d}$ a chiral skyrmion is a stable minimum energy configuration which can be manipulated by electrical currents. A review of recent advances in physics and potential applications is given by Fert et al. in Ref. [36]. The author made contributions to research in skyrmionic textures, for which FinMag was used. They are presented in Section 5.6. The last interaction discussed here is the demagnetising field which acts on the magnetisation so as to reduce the total magnetic moment. It gives rise to shape anisotropy and helps form magnetic domains in larger ferromagnets. Of the listed interactions, it is by far the most expensive computationally. For the finite element case, we implement a FEM/BEM method by Fredkin and Koehler as was used in Nmag [37]. In Section 5.5 we show how its performance in the study of thin magnetic samples can be significantly improved with preconditioning. The micromagnetic problem presents itself as an ordinary differential equation (ODE) with $3n$ degrees of freedom in the form of an initial value problem, where n is the number of mesh nodes. With the finite difference code, we systematically compare formulas used to integrate the ODE for their runtime characteristics. The findings, presented in Section 4.5, are contrasted with the finite element case.

The thesis is structured as follows. After this introduction, Chapter 2 provides the background on software engineering and testing procedures. Chapter 3 introduces the micromagnetic model and gives an overview over the numerical methods needed to succesfully implement a micromagnetic simulation tool. The packages Fidimag and FinMag are described in Chapter 4 and Chapter 5 respectively. It is also there that the numerical studies are presented. Then, Chapter 6 presents

the challenges identified in research software engineering and the lessons learned during the software development that was part of this work. The thesis concludes with a summary in Chapter 7.

Chapter 2

Software Engineering

The term software engineering can be traced to a set of conferences organised by the NATO in 1968 and 1969 [38, 39]. At the time, the title *software engineering* of the conferences was deemed provocative and it was accepted among the participants that the name expressed a need rather than a reality [40]. In fact, with computers only being a few decades old, the 1960s already saw what was called a software crisis: Faulty software that is delivered late and does not meet its specifications. Moreover, with the impact of non-technical factors often underestimated.

At its root, software engineering deals with methods and techniques used in successfully developing reliable software systems. More precisely, it concerns itself with large software systems, which means they are developed by more than one person for more than half a year [41], which is a sharper metric than merely counting lines of code. Most research software projects however, especially in the context of PhD programmes, are not large collaborations from scratch. As we saw in Chapter 1 this does not mean they wouldn't benefit greatly from better software engineering practice. Our focus should thus be on approaches that do not impose a large overhead and can be adapted over the course of the project.

The term engineering hints at the possible parallels that can be drawn with the older engineering fields, for instance starting from an intelligible and complete design, the development of the product in a number of carefully planned phases or the continuous audit of the development process. On the other hand, software doesn't need to be maintained due to physical wear and tear but because of errors that manifest themselves late or due to changing requirements. As a relatively young discipline software engineering isn't without its detractors from inside the programming community. The controversy surrounding the so-called agile methods (c.f. Section 2.1.3) revived the old debate over whether the field may be more craft-like than engineering-like [42]. The proponents of the agile methods called for a bigger emphasis on the activity of coding [43], which detractors view as neglecting the business and social factors involved in creating software [44].

In 2004, the IEEE[1] computer society and ACM[2] first published their SWEBOK (software body of knowledge) guide [45]. Its aim was to characterise, structure, and harmonise the contents of software engineering and to establish a border between software engineering and related disciplines. In its newest edition it identifies the 15 knowledge areas presented in Table 2.1. The computing

[1] Institute of Electrical and Electronics Engineers
[2] Association for Computing Machinery

Knowledge Area	Role
Requirements	expresses the needs and constraints placed on the product
Design	establishes the internal structure of the software that will serve as the basis for its construction
Construction	creates working software through programming
Testing	verifies that a program provides expected behaviours
Maintenance	provides support to software
Configuration Management	idenfities and controls the configuration of the software
Engineering Management	implements and monitors software engineering processes
Engineering Process	describes the work activities performed by software engineers to develop software
Engineering Models and Methods	support a systematic approach to software development and modification
Quality	appraises the state of software quality during development
Engineering Professional Practice	touches on the knowledge, skills and attitudes needed by software engineers
Engineering Economics	places software engineering in a business context
Computing Foundations and Mathematical Foundations	provide background
Engineering Foundations	help complete the work efficiently and effectively

Table 2.1: Summary of the 15 knowledge areas identified by the software body of knowledge.

and mathematical foundations are included because software engineering builds on computer science and mathematics [46]. The related disciplines that intersect with software engineering according to the SWEBOK are computer engineering, general management, project management, quality management, and systems engineering.

2.1 The Software Development Process

Because complex software is difficult to create and maintain, software developers use a software development plan. This plan (the process model) divides the development process into manageable phases limited temporally and in form and content. The software is then completed step by step. The actual development process is accompanied by project management and quality assurance. Process models split individual activities into different phases in the development process and these are then traversed once, like for the waterfall model presented in the next section, or several times. Repeated iterations involve an iterative refinement of the individual software components.

There is disagreement about the optimal process model. As a rule, process models differentiate between at least two major activity groups in the development process: the domain analysis on one hand and the technical realisation (design and programming) on the other hand. Process models differ significantly in their level of detail. Some are detailed procedures that provide concrete work instructions to those involved in the development process.

Three processes will be described in the following sections. The waterfall model because it is the prototypical, sequential software development process, and maybe best understood as a rhetorical device. The discussion of the waterfall model is followed a review of the V-model. As we will see, it leans heavily on the waterfall model, yet with documented use in the public sector. The V-Model is both a principle which states that each stage of the development corresponds to a test

phase and a model of the development phases. Finally, to present the disagreement mentioned before, the Agile method is covered in Section 2.1.3. Agile software development is concerned with methods that remove administrative aspects and let the developer work creatively.

2.1.1 The Waterfall Model

The waterfall model is an early example of a strictly phased software development process. It is maybe the most straight-forward adaptation of a logical needs → plans → action → feedback hierarchy, which contributes to its exact history being unclear. It is generally attributed to a 1970 paper by Royce which doesn't include the term waterfall [47]. The name only appears in a publication a few years later [48] while the process itself was described as early as 1956 [49].

It is a linear (not iterative) top-down process model. The result of previous phases is used as binding inputs for the subsequent phase. In this model, every phase has predefined starting and stopping conditions with clearly defined results. Often the model also describes the activities needed to obtain these results. For certain milestones and after every phase the development documents produced are approved within the project management which is why the waterfall model is called a document-driven model. The name waterfall comes from the common presentation of the consecutive phases as a cascade, as Figure 2.1 shows. Many variants exist. Extensions to the simple model like the iterative waterfall allow a stepwise upward motion in the cascade. A deviation from the strict predecessor-successor principle is also required when need for action associated with an earlier phase becomes apparent, for instance adjustments in the system design due to findings during testing. The first phase is the requirements analysis, which results in a specification, followed by the system design which produces the software architecture. Then comes the programming and module testing after which the actual software is obtained. Finally, integration and system testing come before delivery, deployment and maintenance. The phases are presented in Figure 2.1. Validation answers the question if the product fulfills the requirements of the stakeholders while verification checks if the program does what the specification dictates. Royce described the model as in need of improvement and proposed the following changes:

- inserting a design phase that precedes the analysis phase, in which an early prototype of the final product is implemented and documented using the same model
- planning, measuring and monitoring of the tests, to make sure that the tests perform their task
- formal and ongoing involvement of the stakeholders in the form of reviews

Waterfall models are generally advantageous where requirements, services and processes can be described relatively precisely in the planning phase. However the model suffers from a number of significant drawbacks. For one, clearly separated phases are often unrealistic - the transition between them is fluid. Parts of a system may still be in planning while others are already being implemented or used. The individual phases run one after the other in theory, but in practice regression is often inevitable. Early freezing of the requirements is often problematic as it may lead to costly changes down the line, especially since the whole process ought to be repeated.

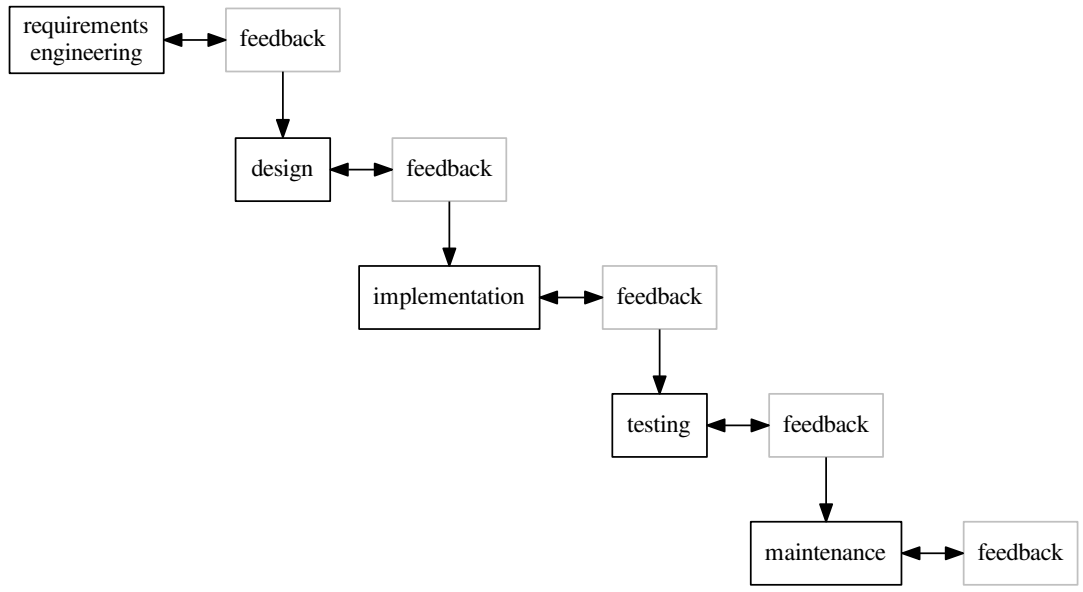


FIGURE 2.1: The waterfall model. Validation and verification make up the feedback step.

Errors are detected late in the development cycle[3] and must be removed with considerable effort. Since it is difficult to define everything definitively and in detail at the beginning of the project, there is the risk that the final result will not meet the actual requirements. In order to counter this, a disproportionate effort is often made in the analysis and design phase. In addition the waterfall model does not allow, or only to a limited extent, to record changes during the course of the project. Since larger software projects usually have a very long runtime, it may happen that the new software is already out of date at the time of its introduction. Because of these sometimes serious drawbacks of the waterfall model with sometimes significant economic consequences, the IT industry has developed a variety of alternative or complementary approaches, software technologies, proposals and tools. The next two sections will describe the V-model and agile software development.

2.1.2 The V-Model

The V-model is a process model for software development that is similar to the waterfall model in that it organises the software development process in phases. In addition to these development phases, the V-model also defines the procedure for quality assurance (testing) by juxtaposing the development phases with test phases. On the left-hand side, a functional specification is started, which is being expanded into a technical specification and a basis for the implementation. At the bottom is the implementation, which is then tested on the right side against the corresponding specification on the left. This is the epitome of the eponym V, which places the individual development levels with their respective test levels, as shown in Figure 2.2. This approach was first proposed by Boehm in 1979 and is based on the waterfall model: The phase results are binding specifications for the next lower project phase [50]. The left, downward leading branch for the specification phases concludes with the realisation phase. An extension to the waterfall

[3] referred to as the big bang

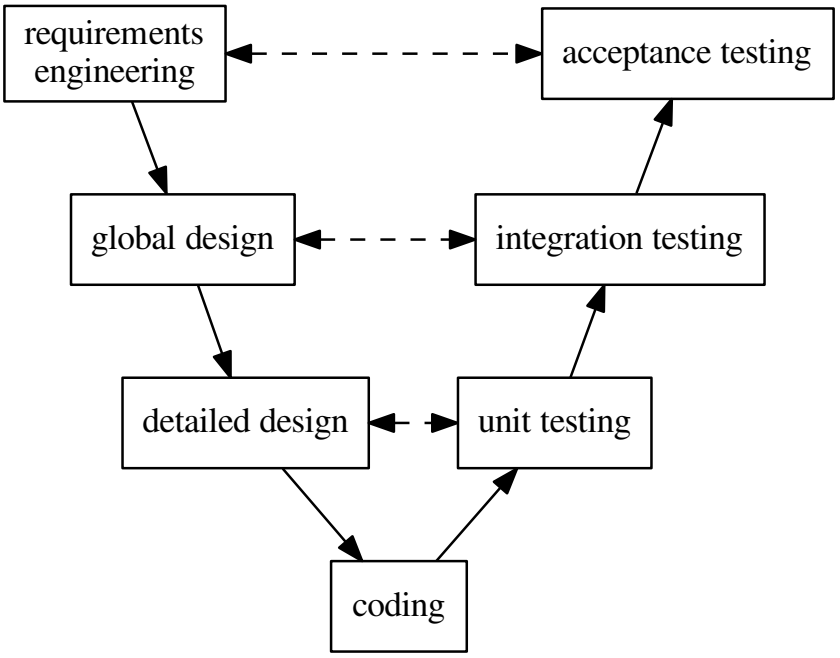


FIGURE 2.2: The V model.

model are the subsequent test phases, which are shown in the right, upward leading branch. The specifying phases each face testing phases. This comparison should lead to the highest possible test coverage, because the specifications of the respective development stages are the basis for the tests (test cases) in the corresponding test stages. For the V model in general, the number of phases and their designations vary in the literature, but always feature a 1 to 1 comparison of design and test levels.

The V-Modell XT [51], where XT stands for eXtreme tailoring, is the basis of development standards of the public sector in Germany. Not only does it guide the work of the project and quality assurance managers, but it defines roles and responsibilities over the complete project lifespan. It has a regulatory significance, meaning its use by the federal administration and its contractors is strongly recommended. The declared goals are improving the communication between the contracting entity and the contractors, minimising risks (both in time and cost), and ensuring quality. Newer editions also incorporate agile ideas, for example allowing scrum, and deal with information security.

2.1.3 Agile Methods

Agile software development refers to approaches in the software development process that should increase transparency and flexibility and lead to a faster deployment of the developed systems in order to minimise risks in the development process. Depending on the context, the term refers to areas of software development, as in the case of Agile Modeling, or to the entire software development process – examples include extreme programming or scrum.

Agile software development is characterised by self-organizing teams, as well as an iterative and incremental approach. It seeks to get by with little bureaucracy and rules and to adapt quickly to change without increasing the risk of error.

The goal of agile software development is to make the development process more flexible and leaner than is the case with classic process models. Agile software development is a countermovement to traditional software development processes, which are considered heavy and bureaucratic, such as the V-Modell discussed in the previous section.

The first software development practices, which are now known collectively as agile, originated in the early 1990s. Agile software development first became popular in 1999, when Kent Beck et al. released the book on extreme programming [52]. This paved the way for other agile processes and methods. The designation agile for this type of software development was introduced in the so-called agile manifesto published in 2001, which states the values and principles underlying the agile process [43].

An agile principle is a guiding principle for agile work. Agile principles are also referred to as methods. The technically incorrect use of the term method for principles is sometimes deliberately used by the representatives of agile methods. In heavy-weight processes, principles are overlaid by extensive method descriptions, often leaving the principles in oblivion. In addition, processes are usually defined primarily by methods, not principles. The listing of principles as methods should give the principles more weight over formal methods. The Agile Manifesto lists twelve principles:

1. Satisfying the customer through early and continuous delivery of valuable software
2. Agile processes make use of changes (even late in development) to the competitive advantage of the customer
3. Delivery of working software in regular, preferably short periods of time (a few weeks or months)
4. Almost daily collaboration between domain experts and developers during the project (collective code ownership)
5. Providing the environment and support needed by motivated individuals to accomplish the task
6. Information transfer, if possible, in conversation face to face
7. The most important measure of progress is the functionality of the software
8. Maintain a consistent workload of clients, developers and users for sustainable development
9. Constant attention to technical excellence and good design
10. Simplicity is essential (KISS principle)
11. Self-organisation of the teams during planning and implementation
12. Self-reflection of the teams on their own behaviour to adapt in order to increase efficiency

With regards to item 4, the collaboration between domain experts and developers, research software projects should find themselves at an advantage because collaborators fill both roles at the same time. The relevant domain knowledge enable more informed decisions during development. But collaboration and communication are still key to avoid single points of knowledge and foster collective code ownership.

Examples of actual agile methods include pair programming, test driven development, constant refactoring, the use of story cards and fast code reviews.

The aim of the approach is to make the software development process more efficient by reducing bureaucracy and by taking greater account of human aspects.

Most agile processes are based on trying to minimise the design-only phase and getting executable software as early as possible in the development process, which can then be submitted to the stakeholder for mutual approval at regular, short intervals. In this way, it should be possible at all times to respond flexibly to stakeholder requests in order to increase overall stakeholder satisfaction. There are now a variety of agile processes with the best known examples being extreme programming and scrum.

It is controversial whether the development process of software is so well understood that a planned production is possible [42]: It could be argued that software is nothing but "executable knowledge". However, knowledge cannot be created in engineering-like manner, such as a bridge or skyscraper, but is found in a creative process. One reason for this is that both the goals and the environment (people involved, requirements, technical environment / interfaces) are flexible and change over time.

The agile methods are therefore particularly well suited to respond to changing requirements, since the development cycles are usually not designed to be long from the outset. The requirements are usually recorded only with short descriptions and formulated shortly before the start of implementation and test preparation. Due to the short periods, the requirements are relatively free of subsequent changes. However, the V-model doesn't prohibit the use of agile elements, such as rapid prototyping; neither before nor during the stages of requirement definition or design. In general, an agile approach is necessary in projects, if only to take into account and integrate the changes that a project inevitably brings. Clearly defined process models help to include these changes in the project in an orderly fashion.

In a business context, the characteristics of the agile procedure must be sufficiently taken into account when concluding the contract. However, clear specifications are hardly possible, since by definition the requirements are developed only during the project.

In its first annual Agile Methodology Survey in 2007 with 4000 participants from various channels in the software development communities, VersionOne found that as many as 84% of the polled companies had adopted agile processes within their organisation. In the twelfth annual survey (2017), that number has increased to 95% [53]. The study Status Quo Agile 2016/2017 with more than 1000 participants showed an improved performance of agile methods compared to traditional project management, notably in team productivity, delivery speed and the ability to manage changing priorities. However, it also noted that the majority of respondents used agile methods in a hybrid approach, supplementing traditional processes [54].

2.2 Software Design and Architecture

Software design is the process for planning a software solution and is part of the overall software development process. Software design is usually required to make the complexity of most computer programs manageable for programmers and to reduce the risk of undesirable developments. In general, the specification results from the requirements survey prior to the software design. Subsequently, various procedures are used to develop a concept with which program structures, programming techniques, and algorithms the requirements will be met [55].

The software architecture is part of the software design. It is a structured arrangement of the system components and the description of their relationships. The architectural components form a decomposition of the entire system, which means that each software element is assigned to exactly one architectural component. The design of a software architecture is the process of creating a rough structure of a software system. Functional and non-functional requirements as well as technical and organisational influencing factors serve as input. The design process is usually iterative and incremental. The result of the software architecture design is a software architecture description that forms the basis for the detailed design.

Complex software systems are decomposed into modules by software architects following a set of fundamental design principles, such that the modules can be implemented relatively independently [56]. Using the principle of targeted abstraction of information, they make the complexity of a system manageable. The principle of separation of concerns ensures that each component of an architecture is responsible for a single task. The inner workings of components are encapsulated through interfaces, which goes back to the principle of information hiding. Ideally, the system is decomposed into a set of self-contained, loosely coupled, high-cohesion components making it easier to understand and adapt. Moreover, a software architecture is often hierarchical. Using Fidimag as an example, Figure 4.1 in Section 4.2 shows how these principles are reflected in the software architecture of the research softwares presented in this thesis.

Software architects use proven design solutions that are documented as so-called architectural patterns [57]. These provide templates for the basic organization and interaction of software components. One example of architectural patterns is the client-server pattern which is the foundation for HTTP and as such at the heart of the internet as we know it. Quality requirements (e.g. for performance, maintainability, reliability and security) are a major factor influencing the design of a software architecture, since functional requirements could in theory also be realised with unstructured software. Often it is the task of the software architect to clarify the technical feasibility and the cost of non-functional architectural design requirements. For this purpose, usage scenarios can be designed, similar systems can be examined and experimental prototypes can be created. Current practical topics are for instance software architectures for cloud computing, multi-core processors and mobile devices [58], as well as service-oriented architectures [59, 60].

As we saw in Section 2.1.3, agile software development projects are increasingly turning to evolutionary software architecture and emergent design, as opposed to the Big Design Up Front. Techniques such as behaviour driven development, test-driven development and, above all, refactoring should ensure that the technical design and architecture are constantly adapted to the requirements of a software development project. Refactoring is restructuring existing source code without changing the observable behaviour of the software. It is unlikely for an optimal

design to emerge on the first pass but through the process of refactoring an existing design can be clarified and simplified. The source code becomes more readable and less complex, thereby increasing maintainability while reducing the effort required for future changes. Interestingly, this made knowledge management for software architectures an active topic of current research in the field of software architectures [61]. As Borrego et al. note, an important challenge in the adoption of agile methods is architectural knowledge management [62], because in agile methods, the documentation of architectural designs, data models or deployment specifications is underemphasised with respect to "working software" [43].

2.3 Software Testing

A software test checks and evaluates software with regard to the requirements defined for its application and measures its quality. The findings are used to detect and correct software errors. Tests during software development serve to complete the software as error-free as possible.

The proof that there are no (more) errors cannot be provided by the software test. It can only fallibilistically determine that certain test cases were successful. Edsger W. Dijkstra wrote: "Program testing can be a very effective way to show the presence of bugs, but is hopelessly inadequate for showing their absence" [63]. The reason for this is that all program functions and all possible values in the input data would have to be tested in all their combinations - which is practically impossible (except for very simple test objects). For this reason, different test strategies and concepts deal with the question of how to achieve a large test coverage with as few test cases as possible.

Pol, Koomen, Spillner [64] explain testing as follows: "Testing is not the only measure of quality management in software development, but often the ultimate one. The later errors are discovered, the more their removal is complicated, from which the reverse conclusion derives: quality must be implemented throughout the project and cannot be 'tested into' a project". They also note significant difference to industry. In the quality control process section, mistakes are often expected only in extreme situations. In hardly any other discipline of software development, such a wide variety of terms has formed as for software testing. This is especially true for the names with which test types and variants are named. They are usually derived from the different situations in which they are performed and the test objectives to which they are directed. Accordingly, the terms of several test types may apply to a specific test. In literature and practice, these terms are usually only partially used, sometimes with different meanings.

According to the V model discussed in Section 2.1.2, the classification of the test stages or test cycles follows the state of development of the system. In each test stage the system is tested against the designs and specifications of the corresponding development stage. The test objectives and test cases are thus based on the respective development results. However, this principle is only applicable if changes made in later stages of development have been updated in the older specifications.

In practice, the test levels described below are often not sharply delimited from each other, but can, depending on the project situation, run fluently or via additional intermediate stages. The module test, also called component test or unit test, is a test at the level of the individual modules

of the software. The object of the test is the functionality within individual definable parts of the software (modules, programs or subprograms, units or classes). The test objective of these tests, which are often carried out by the software developer himself, is the proof of the technical ability to run and correct technical (partial) results.

The integration test or interaction test assesses the cooperation of interdependent components. The focus of the test is on the interfaces of the involved components and should prove correct results over complete processes. The system test is the test stage in which the entire system is tested against the entire requirements. In business settings, the system tests are followed by so-called acceptance tests performed by the customer.

Continuous integration is a term used in software development that describes the process of continuously merging components into an application [65, 66]. The goal of continuous integration is, again, to increase software quality. Usually, not only is the entire system rebuilt, but automated tests are also performed and software metrics are created to measure software quality. The entire process is automatically triggered by checking in code changes to version control. A simplified variant of continuous integration - and often its precursor - is the nightly build (nocturnal build process). In practice, we also encounter continuous integration, paired with a nightly build, to combine the benefits of both systems. Continuous delivery is a further development of continuous integration. A new version of the software is delivered at certain intervals or when a certain quality metric is reached [67].

Continuous integration has the benefit of issues constantly being discovered and fixed – not just shortly before a milestone. Further, the immediate reaction of the system to the check-in of a faulty or incomplete code “educates” the developers in a positive sense to a more responsible handling of check-ins and shorter check-in intervals. Continuous delivery has the added benefit of the constant availability of a recent and functioning executable.

Chapter 3

Simulations of Ferromagnetic Nanodevices

The methodological underpinnings for the simulations presented in this thesis are given by a theory called micromagnetics. Following a brief introduction to magnetism at the nanoscale, the central equation of micromagnetics, namely the Landau-Lifshitz-Gilbert equation will be presented in Section 3.2. We will cover its energy constituents as well as spin-transfer torque terms in Section 3.3 and Section 3.4. As we will see in this chapter, micromagnetics make the phenomena in ferromagnetic nanodevices amenable to computation. The corresponding numerical methods are presented in Section 3.5. The chapter closes with the review of the so-called standard problems in computational micromagnetics in Section 3.6.

3.1 Magnetism at the Nanoscale

In conductors, the valence and conduction bands overlap, so that a fraction of the valence electrons contribute to the number of conduction electrons [68]. The wave functions of the remaining valence electrons, which are bound to the ions in the lattice, extend to a distance of their respective nuclei and overlap with the wave functions of electrons of neighbouring ions. This overlap causes an additional energy term of a purely quantum mechanical nature, without classical equivalent.

The starting point for such a quantum mechanical treatment of ferromagnetic phenomena is the Heisenberg Hamiltonian. It accounts for the exchange interaction between electrons. When spin-orbit coupling is neglected, the Hamiltonian of a n-electron system doesn't contain spin-dependent terms. Still, the expectation values for the energy depend on the configuration of spins, so it makes sense to have a Hamiltonian that is spin-dependent and has eigenfunctions that describe the spin-configuration. If one is only interested in the magnetic properties of the studied system, this approach is warranted, because the magnetic arrangement is based on the tendency of the atomic spins to align in parallel or antiparallel. The Heisenberg Hamiltonian thus obtained is

$$\mathcal{H}_{\text{Heis}} = -\sum_{i,j} J_{ij}\, \mathbf{S}_i \cdot \mathbf{S}_j, \tag{3.1}$$

where the spin interaction which is of the form $\mathbf{S}_i \cdot \mathbf{S}_j$ is called an exchange interaction, the coefficients J_{ij} are called exchange constants, and i and j refer to sites on a lattice. The Hamiltonian contains the sum over the scalar products of all the spins localised at atoms. In good approximation only the coupling between neighbouring spins is to be considered, so J_{ij} is nonzero only if i and j are nearest-neighbour lattice sites. Further, we can assume $J_{ij} = J$, where J is a constant. In a ferromagnetic system $J > 0$ and all spins are aligned in the ground state of the operator. When $J < 0$ the spins will favour an antiparallel alignment and the system is called antiferromagnetic. As the operator does not depend on position, no particular direction in space is favoured. This hints at the model being incomplete, because observed magnetic structures do have favoured directions in space, so called easy axes (see Section 3.3.4). Also, except for very small structures, no perfect alignment of the spins can be observed. The reason for this is the magnetic dipole-dipole interaction between the spins. Typically, it is 10^3 times weaker than the exchange coupling but has a much longer reach. The demagnetising field exists as a consequence, and emanates from where the magnetisation is not solenoidal (see Section 3.3.2).

To date, it is not practically feasible to handle the exchange interaction, the spin-orbit coupling and the magnetic dipole-dipole interaction in a fully quantum mechanical way for systems of interest. In 1932 Bloch suggested to use a classical continuous theory instead of quantum mechanics, where only the expectation values of the spins at the atoms are considered and their spatial distribution is approximated with continuous functions [69]. For low temperatures this is valid, because the expectation value for the angle between neighbouring spins is low. We will see in the next section, when we cover the Landau-Lifshitz-Gilbert equation, that its description of magnetisation dynamics is strictly speaking only valid for 0 K. Most often, temperature effects are ignored. For temperature close to the Curie temperature, the material constants are adjusted. In particular, the saturation magnetisation is reduced. An alternative is including thermal fluctuations in the Landau-Lifshitz-Gilbert equation, as we will review in Section 3.3.6. Anyways, it is possible to group the spin magnetic moments of all electrons in a number of neighbouring unit cells to a resulting moment vector. This vector then has a spin quantum number so large, that its change of direction is quasi continuous and can be subjected to classical equations.

This is the basis for the micromagnetic description of ferromagnetism. The paradigm shifts from spins localised at atoms to a continuously spatially varying vector field $\mathbf{m}$, which shows the averaged spin direction in the grouped unit cells. In good approximation, the average spin magnetic moment in relation to the volume does not depend on position and reflects the maximally possible magnetisation. Landau and Lifshitz [70] were the first to give an equation of motion for $\mathbf{m}$. The dipolar coupling of the electron spins was added by W.F. Brown in Ref. [71]. The micromagnetic equations were thus coupled to the Maxwell equations and are adequate to describe phenomena in which the changes of spatially-dependent state variables on a length in the vicinity of the lattice constant are small.

More generally, besides the thermal effects mentioned, one of the challenges in representing “real” materials and devices in micromagnetics lies in simulating sufficiently large systems. As we will review in Section 3.5.2, there are upper bounds for the size of the mesh discretisation. Not

only can large systems then slow down the computation, an additional bottleneck could be the memory requirements imposed by the boundary element method used in the computation of the demagnetizing field in the finite element case (c.f. Sections 3.3.2 and 3.5.2). The size of the boundary element matrix grows quadratically with the number of surface nodes. Further, "real" devices present imperfections affecting their characteristics. In the finite element case, it is possible to deform the mesh to introduce imperfections, for example edge roughness in nanowires [72].

3.2 The Landau-Lifshitz-Gilbert (LLG) Equation

An equation that describes the behaviour of magnetic moments in a ferromagnet was proposed in 1935 by Landau and Lifshitz [70]. It describes the evolution in time of magnetic moments in an effective magnetic field. A phenomenological damping term was introduced to account for dissipative effects. The damping term was postulated to minimise the energy of the ferromagnetic material and cannot be derived from quantum mechanics. Because it is inadequate for strong damping [73], Gilbert proposed an equation which takes strong damping into account and which can be transformed into the Landau-Lifshitz equation [74]. That equation is called the Landau-Lifshitz-Gilbert (LLG) equation and its first term will be derived using quantum mechanics in this section.

Every electron has an intrinsic spin which is connected with a macroscopic magnetic moment $\mathbf{M}$. The elementary assumption is a small directional variation of adjacent magnetic moments. Then, the discrete atomistic states can be approximated with a continuous vector field $\mathbf{M}(\mathbf{r})$ with the norm

$$|\mathbf{M}(\mathbf{r})| = M_\mathrm{S}, \tag{3.2}$$

where M_S is the density of the magnetic moments in the ferromagnetic material, or saturation magnetisation. That way the dynamics of the magnetisation in the ferromagnet can be derived from the equation of motion of a single spin.

The equation of motion for the j-th component of a single spin $\mathbf{S}$ is given by the Heisenberg equation

$$\frac{\mathrm{d}\hat{S}_j}{\mathrm{d}t} = \frac{i}{\hbar}\left[\hat{H}, \hat{S}_j\right] = \frac{1}{i\hbar}\left[\hat{S}_j, \hat{H}\right]. \tag{3.3}$$

The Hamiltonian $\hat{H}$ of the system can be expanded in powers of $\hbar$. This yields [75, 76]

$$\left[\hat{S}_j, \hat{H}\right] = \sum_k \frac{\partial \hat{H}}{\partial \hat{S}_k}[\hat{S}_k, \hat{S}_j] + \mathcal{O}(\hbar^2)$$

and using the commutation rule for angular momenta $[\hat{S}_k, \hat{S}_j] = -i\hbar \sum_l \epsilon_{jkl}\hat{S}_l$

$$\left[\hat{S}_j, \hat{H}\right] = -i\hbar \sum_{k,l} \frac{\partial \hat{H}}{\partial \hat{S}_k}\epsilon_{jkl}\hat{S}_l + \mathcal{O}(\hbar^2).$$

The commutator $[\hat{S}_j, \hat{H}]$ can be inserted into Equation (3.3) leading to

$$\begin{aligned}\frac{\mathrm{d}\hat{S}_j}{\mathrm{d}t} &= \frac{1}{i\hbar}\left(-i\hbar\sum_{k,l}\frac{\partial\hat{H}}{\partial\hat{S}_k}\epsilon_{jkl}\hat{S}_l + \mathcal{O}(\hbar^2)\right)\\ &= -\sum_{k,l}\frac{\partial\hat{H}}{\partial\hat{S}_k}\epsilon_{jkl}\hat{S}_l + \mathcal{O}(\hbar)\end{aligned}$$

for the j-th component and finally

$$\frac{\mathrm{d}\hat{\mathbf{S}}}{\mathrm{d}t} = -\hat{\mathbf{S}}\times\frac{\partial\hat{H}}{\partial\hat{\mathbf{S}}} + \mathcal{O}(\hbar). \tag{3.4}$$

when the ϵ-tensor is rewritten as a cross product in three dimensions[1]. In the classical limit $\hbar \to 0$ the operators $\hat{\mathbf{S}}$ and $\hat{H}$ are replaced by their expectation values $\langle\hat{\mathbf{S}}\rangle$ and $\langle\hat{H}\rangle$, respectively. $\langle\hat{\mathbf{S}}\rangle$ is related to the macroscopic magnetic moment

$$\mathbf{M} = \gamma_e\langle\hat{\mathbf{S}}\rangle \tag{3.5}$$

by the gyromagnetic ratio $\gamma_e = g_e\mu_B/\hbar$ where g_e is the Landé factor and μ_B the Bohr magneton. Furthermore, replacing the expectation value of the Hamiltonian by the energy density w of the system in Equation (3.4) yields

$$\frac{\mathrm{d}\mathbf{M}}{\mathrm{d}t} = -\gamma_e\mathbf{M}\times\frac{\partial w}{\partial\mathbf{M}}. \tag{3.6}$$

In a magnetic field $\mathbf{H}_{\text{eff}}$ the energy density w of a magnetic moment $\mathbf{M}$ is defined by $w = -\mu_0\mathbf{M}\cdot\mathbf{H}_{\text{eff}}$ where μ_0 is the vacuum permeability. Partial derivation yields

$$\mathbf{H}_{\text{eff}} = -\frac{1}{\mu_0}\frac{\partial w}{\partial\mathbf{M}} \tag{3.7}$$

so that the evolution in time of $\mathbf{M}$ reads

$$\frac{\mathrm{d}\mathbf{M}}{\mathrm{d}t} = -\gamma\mathbf{M}\times\mathbf{H}_{\text{eff}} \tag{3.8}$$

with $\gamma = \mu_0\gamma_e$. Equation (3.8) describes the precession of the magnetic moment $\mathbf{M}$ around the field $\mathbf{H}_{\text{eff}}$ with a constant angle between $\mathbf{M}$ and $\mathbf{H}_{\text{eff}}$. This stationary oscillation preserves energy and does not account for losses due to eddy currents, interaction with lattice excitations or scattering at crystal defects.

Dissipation can be added to Equation (3.8) with a phenomenological damping term [73]. In analogy to classical mechanics, where the usual damping terms are proportional to the velocity, the damping term is chosen to be proportional to $\frac{\mathrm{d}\mathbf{M}}{\mathrm{d}t}$. Because in Equation (3.2) we prescribed the norm of the magnetisation to be constant, the damping term has to be perpendicular to $\mathbf{M}$. It points in the direction of the energetic minimum, i.e. perpendicular to the precession of $\mathbf{M}$ around the field:

$$\mathbf{D} = \frac{\alpha}{M_{\text{S}}}\mathbf{M}\times\frac{\mathrm{d}\mathbf{M}}{\mathrm{d}t} \tag{3.9}$$

The dimensionless damping constant α has to be determined by experiment and factor $1/M_{\text{S}}$ ensures that the damping does not depend on the saturation magnetisation. Adding $\mathbf{D}$ to

[1] Using $(\mathbf{a}\times\mathbf{b})_j = \sum_{k,l}\epsilon_{jkl}a_kb_l$.

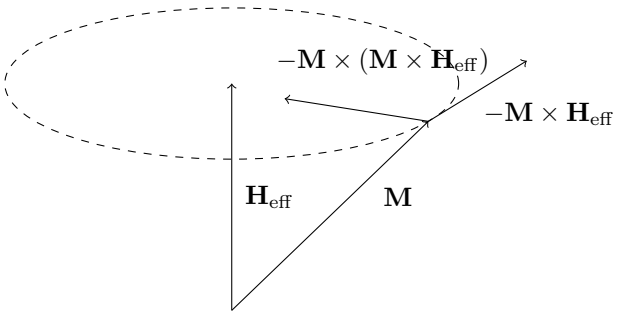


FIGURE 3.1: The precession and damping of the magnetic moment $\mathbf{M}$ in $\mathbf{H}_{\text{eff}}$.

Equation (3.8) leads to the Gilbert equation

$$\frac{\mathrm{d}\mathbf{M}}{\mathrm{d}t} = -\gamma \mathbf{M} \times \left(\mathbf{H}_{\text{eff}} - \frac{\alpha}{M_{\text{S}}} \frac{\mathrm{d}\mathbf{M}}{\mathrm{d}t} \right). \tag{3.10}$$

Equation (3.10) is an implicit equation which, following [34], can be converted into the explicit form

$$\frac{\mathrm{d}\mathbf{M}}{\mathrm{d}t} = -\gamma' \mathbf{M} \times \mathbf{H}_{\text{eff}} + \frac{\alpha'}{M_{\text{S}}} \mathbf{M} \times (\mathbf{M} \times \mathbf{H}_{\text{eff}}) \tag{3.11}$$

where $\gamma' = \gamma/(1+\alpha^2)$ and $\alpha' = \alpha\gamma'$. Equation (3.11) is the Landau-Lifshitz-Gilbert equation, commonly used in micromagnetic simulations. Figure 3.1 shows the forces acting on the magnetic moment $\mathbf{M}$ in the field $\mathbf{H}_{\text{eff}}$. They drive the magnetic moments to align with the field $\mathbf{H}_{\text{eff}}$ and minimise the energy of the ferromagnet.

The description of magnetisation dynamics given by the LLG equation is deterministic, and only strictly valid for zero-temperature. It is however experimentally known that the dynamic properties of small magnetic systems depend on temperature, and that the temperature dependence increases with decreasing sample volume (and decreasing anisotropy) [77]. To include thermal effects, the LLG equation can be augmented with a fluctuating magnetic field presented in Section 3.3.6. The equation thus obtained is known as the stochastic LLG (SLLG) equation, and attempting to integrate the SLLG with a solver that is appropriate for the deterministic LLG is "dangerous" [78]. Special care has to be taken to integrate the SLLG, and Section 3.5.4 will review one appropriate scheme.

3.3 Energy Contributions to the LLG Equation

The Landau-Lifshitz-Gilbert Equation (3.11) describes the dynamics of the magnetisation in the field $\mathbf{H}_{\text{eff}}$. The field $\mathbf{H}_{\text{eff}}$ is called the effective field and is the sum of internal and external field terms

$$\mathbf{H}_{\text{eff}}(\mathbf{r}) = \mathbf{H}_{\text{ex}}(\mathbf{r}) + \mathbf{H}_{\text{demag}}(\mathbf{r}) + \mathbf{H}_{\text{Z}}(\mathbf{r}) + \mathbf{H}_{\text{a}}(\mathbf{r}) + \mathbf{H}_{\text{dmi}}(\mathbf{r}) \tag{3.12}$$

with the exchange field $\mathbf{H}_{\text{ex}}$, the demagnetisation field $\mathbf{H}_{\text{demag}}$, the Zeeman field $\mathbf{H}_{\text{Z}}$, the anisotropy field $\mathbf{H}_{\text{a}}$, and the field induced by the Dzyaloshinskii-Moriya interaction $\mathbf{H}_{\text{dmi}}$. The different contributions to Equation (3.12) will be explained in the following sections.

3.3.1 Exchange Energy

The preference for neighbouring magnetic moments to align is an important property of ferromagnets. The reason for this lies in the exchange interaction between adjacent spins. The moments interact as spins and thus it is not correct to treat them independently [79, 80]. The energy density of two neighbouring magnetic moments is

$$w_{\mathrm{ex}} = -J\mathbf{m}_1 \cdot \mathbf{m}_2 = -J\cos\theta \tag{3.13}$$

with $\mathbf{m} = \mathbf{M}/M_{\mathrm{S}}$ and $|\mathbf{m}| = 1$ and with the angle θ between the magnetisation vectors. The exchange integral J results from the spatial overlap of the wave functions of the electrons which are connected with the magnetic moments [81]. Since the micromagnetic theory requires small angles between the magnetic moments, the cosine in Equation (3.13) can be expanded into a Taylor series up to the second order.

$$w_{\mathrm{ex}} \approx -J\left(1 - \frac{\theta^2}{2}\right) \tag{3.14}$$

The angle θ itself can be approximated by $|\theta| = |\mathbf{m}_i - \mathbf{m}_j| \sim \nabla\mathbf{m}$. Inserting this into Equation (3.14) and spatial integration leads to the following expression for the exchange energy

$$E_{\mathrm{ex}} = A\int_V (\nabla\mathbf{m}(\mathbf{r}))^2 \mathrm{d}V = \frac{A}{M_{\mathrm{S}}^2}\int_V (\nabla\mathbf{M}(\mathbf{r}))^2 \mathrm{d}V.$$

The equation above can be simplified by deriving the identity $2\,(\nabla\mathbf{M}) = \nabla^2\mathbf{M}^2 - 2\mathbf{M}\nabla^2\mathbf{M}$ from the product rule $\nabla^2(uv) = u\nabla^2 v + v\nabla^2 u + 2\nabla u\nabla v$ to obtain

$$E_{\mathrm{ex}} = -\frac{A}{M_{\mathrm{S}}^2}\int_V \mathbf{M}\cdot\nabla^2\mathbf{M}\ \mathrm{d}V. \tag{3.15}$$

The exchange constant A is a material constant that is obtained experimentally, but can be put into relation with J via

$$A = -\frac{JM_{\mathrm{S}}^2 n}{a}, \tag{3.16}$$

with n being the number of atoms in a unit cell with lattice constant a, so that there are $(n/a^3)\mathrm{d}V$ atoms contained in the volume $\mathrm{d}V$ [33].

To obtain the field generated by the exchange interaction, Equation (3.15) can be differentiated like in Equation (3.7) and this gives

$$\mathbf{H}_{\mathrm{ex}}(\mathbf{r}) = \frac{2A}{\mu_0 M_{\mathrm{S}}^2}\nabla^2\mathbf{M}(\mathbf{r}). \tag{3.17}$$

3.3.2 Demagnetisation Energy

As will be shown in this section, the demagnetisation field is generated by the magnetisation itself. Because of this, the demagnetisation energy is called a self energy term. For a ferromagnetic body with no electric currents, it will be derived from Maxwell's magnetostatic equations. The

differential form of Gauss's Law for magnetism is

$$\nabla \cdot \mathbf{B} = 0. \tag{3.18}$$

Since the magnetic flux density $\mathbf{B}$ is related to the magnetisation $\mathbf{M}$ and the magnetic field $\mathbf{H}$ by $\mathbf{B} = \mu_0(\mathbf{H} + \mathbf{M})$, Equation (3.18) can be restated as

$$\nabla \cdot \mathbf{H} = -\nabla \cdot \mathbf{M}. \tag{3.19}$$

The solution to the equation of Ampère's circuital law, which states

$$\nabla \times \mathbf{H} = 0 \tag{3.20}$$

in the absence of current, can be expressed as the gradient of a magnetic scalar potential $\phi(\mathbf{r})$:

$$\mathbf{H} = -\nabla\phi. \tag{3.21}$$

From Equations (3.19) and (3.21) it can be seen that the magnetic scalar potential ϕ satisfies the Poisson equation

$$\Delta\phi = -\nabla \cdot \mathbf{H} = \nabla \cdot \mathbf{M}(\mathbf{r}) \text{ with } \Delta = \nabla^2. \tag{3.22}$$

Outside of the body, the magnetisation $\mathbf{M}$ is zero, so that Equation (3.22) reduces to

$$\Delta\phi_{\text{out}} = 0. \tag{3.23}$$

Maxwell's equations require that the component of $\mathbf{H}$ parallel to the surface and the component of $\mathbf{B}$ perpendicular to the surface are continuous on the boundary. On the surface of the body, this leads to the boundary conditions [34]

$$\phi = \phi_{\text{out}}, \quad \frac{\partial\phi}{\partial n} - \frac{\partial\phi_{\text{out}}}{\partial n} = \mathbf{M} \cdot \mathbf{n}, \tag{3.24}$$

where $\mathbf{n}$ is the unit normal to the surface of the body. To ensure that the demagnetisation energy is finite, the potential ϕ is set to

$$\lim_{\mathbf{r}\to\infty} \phi(\mathbf{r}) = 0 \tag{3.25}$$

at infinity, which is an open boundary condition. Because of the uniqueness theorem of Poisson's equation, any potential ϕ that satisfies Equations (3.22) to (3.25) has a unique gradient, which as stated in Equation (3.21), is the field $\mathbf{H}$. The solution for the potential ϕ is [34, 82, 83]

$$\phi(\mathbf{r}) = \frac{1}{4\pi}\left(\int \frac{\nabla' \cdot \mathbf{M}(\mathbf{r}')}{|\mathbf{r} - \mathbf{r}'|}\mathrm{d}V' + \int \frac{\mathbf{n} \cdot \mathbf{M}(\mathbf{r}')}{|\mathbf{r} - \mathbf{r}'|}\mathrm{d}S'\right), \tag{3.26}$$

where the first integral is over the body and the second integral over its surface. By identifying the demagnetising field $\mathbf{H}_{\text{demag}}$ as $\mathbf{H}$, the energy of the magnetisation in the demagnetising field can be written as

$$E_d = -\frac{-\mu_0}{2}\int \mathbf{M}(\mathbf{r}) \cdot \mathbf{H}_{\text{demag}}(\mathbf{r})\mathrm{d}\mathbf{r} \tag{3.27}$$

$$= -\frac{\mu_0}{2}\int \mathbf{M}(\mathbf{r}) \cdot \nabla\phi(\mathbf{r})\,\mathrm{d}\mathbf{r}. \tag{3.28}$$

The factor 1/2 avoids counting the energy contribution of the points $\mathbf{r}$ and $\mathbf{r}'$ in Equation (3.26) two times.

3.3.3 Dzyaloshinskii-Moriya energy

The term skyrmion (named after British physicist Tony Skyrme) originated in particle physics and the mathematical area of topology to describe topologically stable field configurations with particle-like properties. But decades before the term skyrmion was coined, the original question was how to explain the weak ferromagnetism of mainly antiferromagnetic crystals like αFe_2O_3, $MnCO_3$ and $CoCO_3$. Dzyaloshinskii first asserted that the weak ferromagnetism was an intrinsic property of these crystals from purely symmetry grounds [35]. He showed that in αFe_2O_3, an antiferromagnetic spin arrangement perpendicular to the trigonal axis and a canted spin arrangement which has a net magnetic moment perpendicular to the trigonal axis have the same symmetry. Then he wrote down the free energy terms of the system allowed under the crystal symmetry in term of spin variables. The term

$$\mathbf{D}_{\mathrm{DMI}} \cdot [\mathbf{S}_1 \times \mathbf{S}_2] \,, \tag{3.29}$$

where $\mathbf{D}_{\mathrm{DMI}}$ is a constant vector, favours the canted spin arrangement rather than the antiferromagnetic one and gives rise to the weak ferromagnetism. The term is due to relativistic spin-lattice and magnetic dipole interactions. Dzyaloshinskii from this evolved a theory of helicoidal structures in antiferromagnets [84]. In the meantime, Moriya argued that Dzyaloshinskii's original explanation is phenomenological and derived Dzyaloshinskii's results by developing his theory of the anisotropic superexchange interaction [85].

Bak and Jensen explained the helical magnetic structures which had been reported in several neutron diffraction experiments on FeGe and MnSi [86]. They noted that both materials crystallise in the tetrahedral $P2_13$ structure in which there is no inversion symmetry. Starting with the expansion of the free energy in terms of a slow-varying spin density they linked the appearance of the helices to what was then called the ferromagnetic Dzyaloshinskii instability. Slowly, the name for it morphed to the Dzyaloshinskii-Moriya interaction, and Bogdanov et al. calculated three thermodynamically stable phases for ferromagnetic materials: a uniform state for high field values and a one-dimensionally modulated spiral state for low fields separated by a newly discovered vortex state [87, 88].

Until 2006, it was assumed that no skyrmions, meaning countable and stable particles in continuous fields, could form in magnetic materials. However, Rössler et al. theoretically proved that spontaneous skyrmion lattice ground states can exist for example at surfaces, in thin films and in bulk materials, where a lack of inversion symmetry leads to chiral interactions of the Dzyaloshinskii-Moriya type [89]. A year later, the observation of a magnetic order of specific chirality on single atomic layer of manganese was reported by Bode et al. [90]. Spin-polarised scanning tunneling microscopy revealed that adjacent spins were not aligned, but slightly canted resulting in a spin spiral structure. By showing that this was caused by the Dzyaloshinskii-Moriya interaction, they underlined the significance of this interaction for magnets in reduced dimensions.

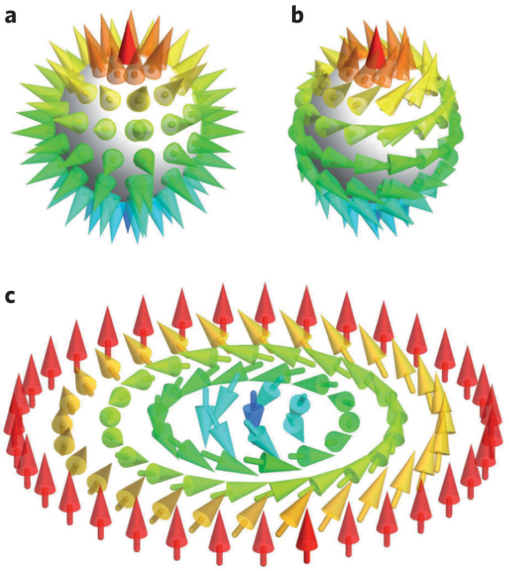


FIGURE 3.2: Rendering of skyrmions. Hairy sphere on the top and "vortex" on the bottom. The latter object being what we identify as a skyrmion in solid-state physics. Reproduced from [1].

The first proper skyrmion states were detected in the bulk magnet MnSi using neutron scattering [91]. In MnSi, the lack of inversion symmetry of the atomic crystal results in chiral spin-orbit interactions of the Dzyaloshinskii-Moriya type. Using Lorentz transmission electron microscopy, real-space imaging of a two-dimensional skyrmion lattice on a $Fe_{0.5}Co_{0.5}Si$ thin film was realised as well [92]. After further detection of skyrmion lattices in the bulk magnets FeCoSi and FeGe, Heinze et al. showed square lattices of skyrmions on a hexagonal iron monolayer using spin-polarised scanning tunneling microscopy. From this and first-principles calculations they developed a spin model on a discrete lattice which identified the interplay of Heisenberg exchange, four-spin and the DMI (Dzyaloshinskii-Moriya interaction) [93]. A good break-down of Heinze's article is given in Ref. [1]. It also shows the topological equivalence of a hairy sphere and a "vortex", the latter object being what we call a skyrmion, reproduced in Figure 3.2.

To enable the application of magnetic skyrmions for spintronics, challenges like low crystallisation temperature and low stability have to be overcome. In thin films of FeGe, Yu et al. reported skyrmion states near room temperature. When the films are thinner than the skyrmion lattice constant, the observed skyrmions are stable over a wide range of temperatures and magnetic fields. However, they become unstable when the films are thicker [94]. It was also calculated that skyrmion states could be spontaneous ground states on thin nanodisks with DMI and uniaxial anisotropy without the help of an external magnetic field or thermal fluctuations [95]. Rohart and Thiaville stress that the edges of the nanodisk are essential to the formation of skyrmions as they provide a confinement and limit skyrmion expansion [96]. Manipulation of the magnetic structures by currents is essential to spintronics as well. Calculations using the LLG-equation (c.f. Section 3.2) predict continuously rotating lattices when combining the currents with small gradients of either the magnetic field or the temperature [97]. The DMI also influences the formation of domain walls in ultrathin films with perpendicular anisotropy according to micromagnetic studies [98]. The domain walls assume a Néel configuration with fixed chirality, as was measured by Emori et al. [99].

With a diameter as small as a few nanometers and the ability of being moved by electrical currents, skyrmions are interesting candidates for future applications in information technology. It was shown that individual skyrmions can be written and deleted with a local spin-polarised current from a scanning tunneling microscope [100]. An external magnetic field was used to tune the energy landscape. Using numerical investigations, Sampaio et al. predict that individual skyrmions could not only be written and deleted with local spin-polarised currents, but that the current-induced spin torques could be used to displace the skyrmions even in presence of large defects [101]. Fert, Cros and Sampaio envision a device similar to the domain-wall racetrack memory, where information is stored by skyrmions on a nanoribbon instead of a domain-wall and magnetic domain pair [102]. The small size of a skyrmion is an advantage over the magnetic domain, which can hardly be reduced below 30-40 nm. Also, skyrmions can be manipulated using much lower spin-polarised current densities [103, 104].

Systems with Dzyaloshinskii-Moriya energy are challenging to model, because the energy is a new term that first needs to be implemented in simulators.

3.3.4 Magnetocrystalline Anisotropy Energy

Some ferromagnetic materials show preferred directions for the magnetisation. These directions are called easy axes and are due to inhomogeneous charge and spin distributions in the crystalline structures. Since the energy axes are not directed, the anisotropy energy is mirror symmetric. For the most common case of a single easy axis the anisotropy is called uniaxial and a simple phenomenological approach for the energy is given by

$$E_a = -\int \left(K_1 \sin^2(\vartheta(\mathbf{r})) + K_2 \sin^4(\vartheta(\mathbf{r}))\right) \mathrm{d}V, \tag{3.30}$$

where ϑ denotes the angle between the magnetisation and the easy axis. Only even powers are taken into account to ensure the required mirror symmetry. Terms of higher order than the fourth are omitted. The expansion factors K_1 and K_2 are determined experimentally. In all known cases $|K_2| \ll |K_1|$, and in most cases K_2 can be neglected. That justifies the power series expansion and stopping the expansion after the fourth order [34].

3.3.5 Zeeman Energy

The magnetisation $\mathbf{M}$ of a ferromagnet interacts with an external magnetic field. If that field is $\mathbf{H}_\mathrm{Z}$ the Zeeman energy will be

$$E_\mathrm{Z} = -\mu_0 \int \mathbf{H}_\mathrm{Z}(\mathbf{r}) \cdot \mathbf{M}(\mathbf{r}) \mathrm{d}r. \tag{3.31}$$

It is mimimised if the magnetisation is aligned parallel with the field. The field itself can directly be added to $\mathbf{H}_\mathrm{eff}$ in Equation (3.12).

3.3.6 Thermal Fluctuations

In Section 3.2, we saw that the LLG equation is deterministic and valid for the zero-temperature case, but could be extended to obtain the (stochastic) SLLG. The term to account for thermal fluctuations is an isotropic Gaussian-distributed random field with variance

$$\sigma^2 = n^2 \frac{2\alpha k_B T}{\gamma M_S V_i \left(1+\alpha^2\right) \Delta t} \tag{3.32}$$

given by Brown [105]. It can be added directly to the effective field $\mathbf{H}_{\text{eff}}$. It is delta-correlated, with V_i being the mesh element volume, and Δt the integration time step. The parameter n typically ranges from 1 to 3 and accounts for mesh size dependence of the simulation [78].

Describing the thermal effects by a multiplicative fluctuating term leads to the Ito-Stratonovich dilemma [106]. This entails having to specify which type of stochastic calculus to use, knowing that the choice of stochastic interpretation imposes the use of specific schemes for the integration. Brown, and subsequent works, use the Stratonovich calculus, and an appropriate method for integration will be presented in Section 3.5.4.

3.4 Spin-Transfer Torque

3.4.1 The Giant Magnetoresistance Effect

The giant magnetoresistance (GMR) was discovered by the groups of Albert Fert and Peter Grünberg at the end of the nineteen-eighties [107, 108]. The use of the GMR resulted in significant progress in the domain of data storage and magnetic sensors and Fert and Grünberg were awarded jointly with the Nobel Prize in Physics. Together with [109] the discovery led to the emergence of spintronics, which combines traditional electronics with the spin and magnetic moment of the electron. It has been used extensively in the read heads of modern hard drives.

The spin magnetic moment of the electrons inside a ferromagnetic conductor is preferably oriented in parallel to the magnetisation, rather than antiparallel. The electrons in the ferromagnet can be split up into two groups, one with the moment oriented along the magnetisation direction and the other with the moment oriented in the opposite direction. The two electron groups will have different energies in the band structure. Since only electrons above the Fermi level in the band structure participate in conduction, a different number of charge carriers for the two populations and thus a different conductivity can be expected. If in a metal the band for one of the two groups is completely below the Fermi level, the current will be carried exclusively by electrons with the other spin orientation. The spin polarisation is then 100 %, but will be lower for materials like Fe or Ni that have both bands half filled [110].

A trilayer system like shown in Figure 3.3 can illustrate the GMR. Two thicker ferromagnetic conductive layers are separated by a thin non-magnetic metallic layer which prevents exchange coupling between them. The current flows in the out of material-plane direction. This is the so-called *current perpendicular to plane* (CPP) geometry. Assuming the system is small enough, the magnetisation is homogeneous in the outer layers. In the absence of an external magnetic

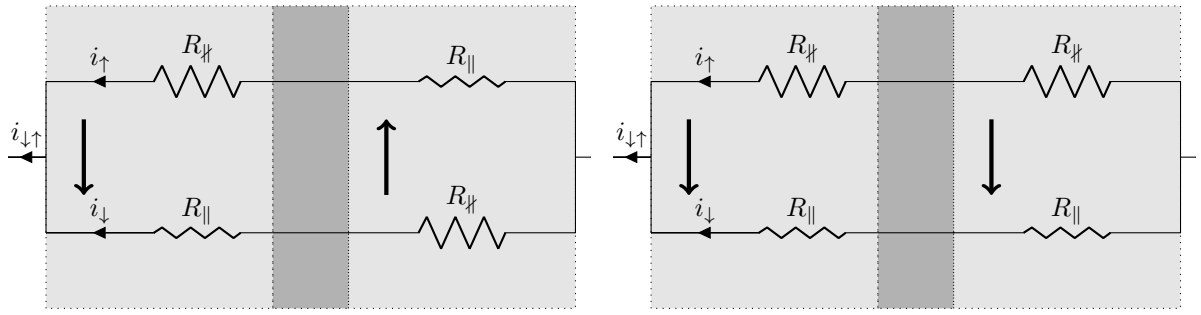


FIGURE 3.3: Trilayer system with a non-magnetic metal between two ferromagnetic layers in two possible configurations with different conductivity. (left) In the absence of an external field, the system relaxes into an antiparallel configuration. (right) The two magnetisations are parallel when an external field is applied.

field, the system relaxes into a state where the magnetisation in the outer layers is oriented in opposite directions, because it reduces the demagnetisation field in the system [111]. As can be seen on the left in Figure 3.3, each of the electron groups passes through a region with parallel spin (the resulting lower resistance is called $R_\parallel$) and a region with antiparallel spin (the resulting higher resistance is called $R_\nparallel$). The resistance for each group is then $R_\parallel + R_\nparallel$ and the total resistance for the circuit shown is $R = (R_\parallel + R_\nparallel)/2$. When an external field is applied, the magnetisation in the outer layers aligns with it, as can be seen on the right side in Figure 3.3. One of the two groups of electrons only experiences parallel spin and the other only antiparallel. The resistance for each group is $2R_\parallel$ and $2R_\nparallel$, respectively, and the total resistance becomes $R_{\text{aligned}} = 2R_\parallel R_\nparallel/(R_\parallel + R_\nparallel)$. The difference in resistance for the two possible configurations is [111]:

$$\Delta R = R + R_{\text{aligned}} = \frac{R_\nparallel - R_\parallel}{2(R_\nparallel + R_\parallel)^2}$$

If the direction of the magnetisation in one of the two layers is fixed, for instance due to magnetic anisotropy, a resistance measurement permits to determine the magnetisation direction in the other layer. This system has thus been successfully used as a magnetic sensor [112, 113]. Besides the CPP geometry, there is also the *current in plane* (CIP) geometry, where the current flows parallel to the plane of the layers. While the GMR effect is smaller relatively in the latter it is often used in GMR devices [110] because the absolute resistances are small and difficult to measure in the CPP geometry [110, 81].

3.4.2 Spin-Transfer Torque extension of the LLG Equation

The GMR effect shows that the magnetic configuration of a ferromagnet can have an influence on its electric conductivity. The inverse is true as well: the work of Berger and collaborators on current-induced domain wall motion in nineteen-seventies and nineteen-eighties [114, 115, 116, 117] investigated the possibility that torque may be applied to magnets by a spin-polarised current. In the event of non-collinear moments of the magnetic electrodes in a magnetic tunnel junction and a current traversing it, Slonczewski predicted an applied torque on the electrodes in 1989 [118]. At that time magnetic tunnel junctions could not achieve the current densities needed to provide a torque large enough to reorient the magnetic moments.

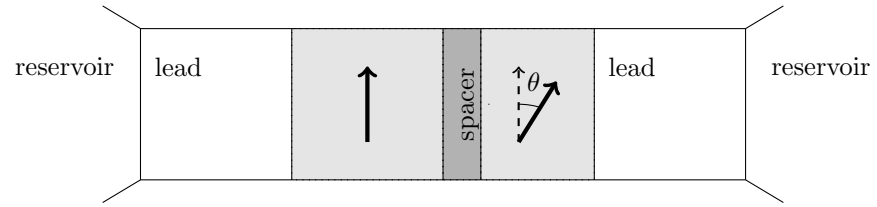


FIGURE 3.4: Schematics of a typical spin valve device. A thin non-ferromagnetic spacer is sandwiched between two ferromagnetic layers of different thickness. The magnetisation in the thinner free layer is inclined by an angle of θ in respect to the magnetisation in the thicker fixed layer.

Moving on to ferromagnet/normal-metal/ferromagnet multilayer spin-valve devices, Slonczewski and Berger predicted torque at achievable current densities in 1996 [119, 120]. The spin-valve geometry is a refinement of the general CPP trilayer system described in Section 3.4.1 and is shown in Figure 3.4. The first layer is called the fixed layer. It will be made less susceptible to the torque by making it thicker, or by using a material that gives it a larger total magnetic moment than the second layer, or, in the case of an etched pillar structure, by leaving it connected to an extended film rather than as part of the pillar structure. In contrast to this, the second ferromagnetic layer will be called free layer. The two layers are still separated by a thin nonmagnetic spacer. When unpolarised electrons pass the first layer, they will be subject to spin-filtering and arrive at the spacer with an average spin moment parallel to the moment of the fixed layer. While the filtering is not perfect, polarisations of around 40 % are possible in material systems like Co and Cu [121]. The thickness of the spacer must be inferior to the spin-diffusion length for some spin polarisation to remain when the current enters the free layer. Usually, the spacer layer is less than 100 nm thick [122]. The free layer will experience a torque turning its moment toward the orientation of the incoming spins and thus toward the direction of the fixed layer's moment, stabilising the parallel orientation of the two moments. The situation can be reversed when the current is incident from the right in Figure 3.4. The electrons reaching the spacer layer will have an average spin parallel to the free layer moment. At the normal-metal/fixed layer boundary a fraction of the electrons will be reflected back toward the free layer. Assuming that parallel spins are more easily transmitted, the reflected electrons will have an average spin antiparallel to the moment of the free layer and the torque the free layer experiences now will turn its magnetisation away from the fixed layer moment. A large enough current can then destabilise the parallel orientation of the two moments.

For a description of the dynamics of the free layer, an external magnetic field, anisotropies and damping have to be considered in addition to the spin-transfer torque. Figure 3.5 shows a simple case where an external applied field, the magnetic moment in the fixed layer and that the easy axis of uniaxial anisotropy in the free layer all point in the z-direction. Supposing that the magnetic moments in the fixed and free layer are misaligned by a small angle, the free layer moment will precess around the z-axis. In the absence of spin-transfer torque, the moment in the free layer will eventually end up in the lowest energy configuration, aligned with the z-axis due to damping. Depending on the sign of the current, the spin-torque can either reinforce the damping or weaken it. In the second case, when the spin-transfer torque is larger in magnitude than the damping, the system becomes instable. The spin-transfer torque adds energy to the free layer, so that it can spiral to higher-energy orientations away from $\theta = 0$. The dynamics for this situation have different outcomes, depending on the angular dependence of the spin-transfer and damping

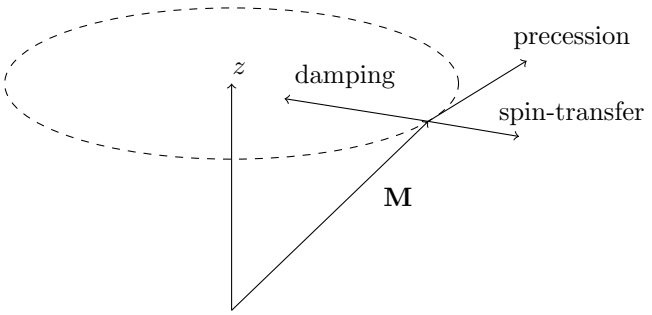


FIGURE 3.5: The precession and damping of the magnetic moment $\mathbf{M}$ in $\mathbf{H}_{\text{eff}}$ of Figure 3.1 with added spin-transfer torque. Depending on the sign of the current, the spin-transfer torque either reinforces the damping, or acts in an opposite direction (shown).

torques, and the magnitude of the external field. One outcome is magnetic reversal, where the free layer will spiral to a static state antiparallel to the fixed layer. The polarisation of the free layer can be switched back and forth with sufficiently large alternative pulsing currents [123, 124]. When the energy won due to spin-transfer torque is balanced with the energy dissipated due to damping over each cycle in the precession, it is also possible that the system reaches a dynamical equilibrium at an intermediate angle. In this case, a DC current can create steady-state magnetic precession oscilations with GHz or tens of GHz frequencies [121]. Such dynamical nodes have been observed experimentally and in different device geometries [125, 126, 127]. In Figure 3.5, the equilibrium will be at an constant angle to the z-axis, but depending on the magnitude of the current and direction and magnitude of the external magnetic field, together with more complicated magnetic anisotropies in real devices, several different types of dynamics modes are possible [127].

Slonczewski extended the Gilbert equation (Equation (3.10)) in [119] to account for spin-transfer torque in a single magnetic domain. In [128] he developed a theory of spin-transfer torque by combining quantum statistical mechanics for the description of the spacer layer with network analysis for rest of the structure. For a spin valve in which the two ferromagnets and leads are identical and the spacer layer is thin the torque L_S in function of the misalignment of the two moments θ is given as

$$L_S(\theta) = \frac{\hbar I}{2e} \frac{P\Lambda^2 \sin\theta}{(\Lambda^2+1) + (\Lambda^2-1)\cos\theta}, \tag{3.33}$$

where I is the total current flowing through the structure, the resistive polarisation factor is

$$P = \frac{\frac{1}{2}(R_\downarrow - R_\uparrow)}{\frac{1}{2}(R_\downarrow + R_\uparrow)} = \frac{r}{R} \tag{3.34}$$

and $\Lambda^2 = GR$. The conductance is proportional to the cross-sectional area of the device and $R_\uparrow$ and $R_\downarrow$ are effective resistances experienced by spin-up and spin-down electrons between the reservoir (shown in Figure 3.4) and the spacer layer.

Xiao et al. extended the work to the general asymmetric case [129], in which the ferromagnets are not identical. In [23] the Gilbert equation based on Xiao et al. is given as

$$\frac{\mathrm{d}\boldsymbol{m}}{\mathrm{d}t} = -|\gamma| \times \mathbf{H}_{\text{eff}} + \alpha \left(\boldsymbol{m} \times \frac{\mathrm{d}\boldsymbol{m}}{\mathrm{d}t} \right) + |\gamma| \beta \epsilon (\boldsymbol{m} \times \boldsymbol{m}_p \times \boldsymbol{m}), \tag{3.35}$$

where

$$\beta = \left|\frac{\hbar}{\mu_0 e}\right| \frac{J}{tM_\mathrm{s}} \text{ and}$$
$$\epsilon = \frac{P\Lambda^2}{(\Lambda^2+1)+(\Lambda^2-1)(\boldsymbol{m}\cdot\boldsymbol{m}_p)}$$

incorporate the torque L_S of Equation (3.33). J is the current density, e is the electron charge, $\boldsymbol{m}_p$ is the unit electron polarisation direction and t is the thickness of the free layer. An explicit form of the extended LLG equation is given by Sun in [130, 131]. With the notation used in Equation (3.11) it reads

$$\frac{\mathrm{d}\boldsymbol{m}}{\mathrm{d}t} = -\gamma'\boldsymbol{m}\times\mathbf{H}_\mathrm{eff} + \frac{\alpha'}{M_\mathrm{S}}\boldsymbol{m}\times\boldsymbol{m}\times(\mathbf{H}_\mathrm{eff}+\boldsymbol{H}_\mathrm{s}), \tag{3.36}$$

where

$$\boldsymbol{H}_\mathrm{s} = \frac{\eta\hbar I}{2eM_\mathrm{S}\alpha'}\boldsymbol{m}_p \tag{3.37}$$

is a spin-angular-momentum transfer term. I is the current, and $\eta = (I_\uparrow - I_\downarrow)/(I_\uparrow + I_\downarrow)$ is the spin-polarisation factor, where $I_\uparrow$ and $I_\downarrow$ are the majority and minority spin-polarised currents, with their polarisation axis defined by the fixed magnetic layer.

3.4.3 Spin-Torque Oscillators

Spin-torque oscillators (STO) are a class of devices in nanotechnology which have been subject to rapidly increasing interest due to their potential use for microwave generation over large frequency ranges [127]. They have also been studied for their potential applications as directionable steerable sources of spin waves, ultrasensitive microwave detectors and for technical applications within the paradigm of spin-wave logic and nano-optics [132].

There are three reasons which make it vital to fabricate devices in which current flow is confined to a small diameter in order to measure the effects of spin transfer torque. First, the necessary current needed to create magnetic excitations using spin transfer can be reduced because it scales with the total magnetic moment of the free layer. Then, the effect of spin transfer has to dominate the effects of the magnetic field that is generated by the current. For the circular nanopillar geometry which will be described in the next paragraph, a diameter of around 250 nm was determined as a maximum for Co layers a few nm thick [133]. At last, if the device is small enough that the moment in the free layer moves as a single domain the spin-transfer effects will be easier to interpret. This typically requires devices in the 100 nm scale.

In mechanical point contact devices, a magnetic multilayer is contacted by a sharp metal tip. The contact region has a size of tens of nanometers [126]. The point contact can also be fabricated by lithography [125]. In both cases the generated excitations of the magnetic layers are localised in a nanoscale region around the contact, but the required current densities are typically greater than $10^9\,\mathrm{A\,cm^{-2}}$ and thus large, causing the deterioration of the sample by Joule heating. In so-called nanopillars, which are thin, cylindric structures, the magnetic layers are not in contact with the rest of the macroscopic film. This reduces the needed currents significantly, to values around $10^7\,\mathrm{A\,cm^{-2}}$ [134]. The nanopillar geometry allows greater influence on the exact dimensions of the fixed and free magnetic layers. In experiments, nanopillar devices have been fabricated by

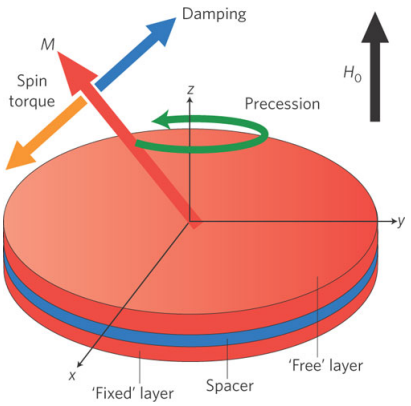


FIGURE 3.6: Schematics of a bilayer spin-torque oscillator (reproduced from [2]). When magnetic damping is compensated by the spin torque resulting from a spin-polarised current flowing from the fixed layer, the free layer magnetisation M precesses around the direction of an applied magnetic field H0.

stencil techniques [135], electrodeposition within cylindrical nanopores [136] and the combination of lithography and ion milling [137]. Most importantly, modeling is simplified by the nanopillar geometry due to the absence of exchange coupling with an extended film which facilitates quantitative tests of theoretical predictions.

In STOs, the transfer of spin angular momentum from a spin polarised current to a thin magnetic layer counteracts the damping inherent in the precession of the magnetisation around the effective field (see Section 3.4.2). This is illustrated in Figure 3.6 for a bilayer system. If the spin-torque term and the damping term are equal in magnitude, instead of spiraling towards the equilibrium direction, the magnetisation will stay on a stable cycle with a precession frequency in the GHz range, which can be tuned by changing the applied magnetic field, or the electric current [126, 127].

The mode of the precession can be a quasi-uniform mode [127], a spin-wave mode [126, 138, 139] or a magnetic gyrotropic mode [140, 141]. The last one describes the rotation of a magnetic vortex core about its equilibrium position in the center of the free layer. The precession of the magnetisation of the free layer is converted to an electrical signal by the means of giant magnetoresistance[2] (see Section 3.4.1) and will be detected as oscillations of the resistance across the device.

For practical use, the relatively low output power of less than 1 nW [138] and the large linewidth of the emitted signals [142] have to be improved. Mutual synchronisation of many weakly coupled oscillators is an approach to that challenge. That means the oscillators are forced to emit at a common frequency and in phase in spite of the intrinsic dispersion of their individual frequencies. It is impossible to achieve perfect uniformity in the geometric and magnetic characteristics of the STOs and this will impact their output frequency. In 2005, Mancoff et al. demonstrated synchronisation between a pair of 80 nm diameter point contacts when they were placed less than 200 nm apart [139]. The output power for this pair was approximately twice the total power from contact pairs which were placed farther apart. This, and the high-frequency output showing only one peak instead of two indicates that the closely spaced pair phase-locked with no phase-shift when the current reached a threshold value. Additionally, a slight decrease of around 10-30 % on

[2]for a magnetic spacer. In magnetic tunnel junctions, a dielectric spacer is used and the corresponding effect is the tunneling magnetoresistance.

average of the signal FWHM was observed. The same year, Kaka and collaborators reported similar behaviour for oscillators 500 nm apart for an electrical nano-contacts to thin-film magnetic bilayer geometry [138]. A year later, theory predicted how an array of potentially hundreds oscillators could be synchronised, not by placing them close together, but by coupling them electrically [143]. The coupling is due to the microwave components of the current induced in each oscillator by the oscillations in all the other oscillators. This raises complex problems related to the field of dynamics of nonlinear systems. In contrast to conventional auto-oscillators, the oscillation frequency and power output in STOs are strongly correlated. This increases the effective coupling to external signals and makes synchronisation possible in a wider frequency range [144]. The problem is that the same nonlinearity leads to a large[3] intrinsic phase shift between the oscillation of the magnetisation and external driving signals [145]. In ref. [145], phase shift jumps for a single oscillator of 180° occur, and both periodic and chaotic oscillations are observed. When every oscillator contributes to the driving signal, like for the imagined electrically coupled array in [143], this can completely destroy the mutual phase-locking. Space- and time-resolved images of steady-state oscillation of a magnetic vortex in a spin-valve nanostructure have been published [146].

The first demonstration of the synchronisation of more than two STOs was given in 2009, although using vortices and low frequencies [147]. In this work, four magnetic vortices achieved synchronisation through their interaction with antivortices in an array of nanocontacts, without a magnetic bias field. The Oersted field generated by the current flowing through a matrix of 2x2 nanocontacts nucleated a vortex under each of them. Because all four vortices had the same chirality, energy minimisation implied nucleation of antivortices on the matrix sides and of a vortex with opposite chirality in the matrix centre. As in the work of Mancoff et al. [139], the nanocontacts share the same unpatterned free layer. This approach is called parallel synchronisation. The record for spin-torque oscillators synchronised in the nanocontact geometry is currently three [148]. The predicted mutual phase locking of electrically connected nanopillar STOs that couple via a shared microwave current is known as serial synchronisation. It has not yet been confirmed experimentally, but analytically and numerically [149]. The transient synchronisation dynamics are being investigated to explore possible reasons behind this issue [150, 151].

Different approaches to synchronise spin-torque oscillators have emerged in the last two years. A parallel inductor-capacitor-resistor load synchronised an array of oscillators according to a numerical stydy published in 2013 [152]. However, this would be prohibitively difficult to realise experimentally. According to micromagnetic simulations, an applied periodic magnetic field can act as a medium to induce synchronisation even without coupling through spin current [153]. When synchronisation occurs via spin-wave coupling, it is characterised by the constructive interference pattern of two distinct wave sources as micromagnetic simulations show [154]. In a quasi one-dimensional system, possible long-range synchronisation via spin-wave coupling was confirmed by numerical studies which also found a minimum distance between oscillators [155]. That synchronisation of two oscillators in the nanopillar geometry occurs as a function of distance was also the conclusion of a numerical study done by Belanovsky et al., when the two nanopillars have two different diameters [156].

[3] around $\pi/2$

In summary, phase-locking phenomena occur only in limited conditions. Phase-drift among oscillators becomes a serious issue as the number of oscillators increases [157, 158]. In serial and parallel architectures, the oscillators interact via spin wave, spin vortex, and electric currents. To enhance these interactions, it is necessary to optimise the spatial and geometric parameters of the array layouts. Where analytical models and macrospin models don't agree with reality, full micromagnetic simulations can help understand experimental results.

3.5 Numerical Methods

When closed form analytical solutions to physical problems do not exist, approximate solutions can be obtained using numerical techniques such as the finite difference method (FDM) or the finite element method (FEM). This is generally the case for the micromagnetic equations [34].

3.5.1 The Finite Differences Method

The finite differences method is a simple approach to solve differential equations. It replaces the derivatives in the equations with appropriate differentials and thus turns the differential equations into algebraic ones. While the FDM predates the rise of information technology by far [159], the fact that it lends itself well to numerical treatment has made it a staple in engineering applications.

In calculus the derivative of a function $u(x)$ is defined as

$$\frac{\mathrm{d}u(x)}{\mathrm{d}x} = \lim_{\Delta x \to 0} \frac{u(x+\Delta x) - u(x)}{\Delta x}. \tag{3.38}$$

The function is then discretised, in other words defined on a finite, discrete set of points x_i instead of being continuous. The limit of $\Delta x \to 0$ can thus be replaced by the so-called forward Euler approximation

$$\frac{\mathrm{d}u(x_i)}{\mathrm{d}x} \approx \frac{u(x_i+\Delta x) - u(x_i)}{\Delta x} = \frac{u_{i+1} - u_i}{\Delta x} \tag{3.39}$$

with a finite $\Delta x = x_{i+1} - x_i$ and $u_i = u(x_i)$ representing the function value on the i-th point. Using the Taylor series of $u(x_{i+1})$ around x_i yields

$$u(x_i+\Delta x) = u(x_i) + \left.\frac{\mathrm{d}u}{\mathrm{d}x}\right|_{x_i} \cdot \Delta x + \left.\frac{\mathrm{d}^2u}{\mathrm{d}x^2}\right|_{x_i} \cdot \frac{\Delta x^2}{2!} + \left.\frac{\mathrm{d}^3u}{\mathrm{d}x^3}\right|_{x_i} \cdot \frac{\Delta x^3}{3!} + \dots \tag{3.40}$$

which after rearranging the terms allows quantifying the error of the approximation with

$$\frac{u(x_i+\Delta x) - u(x_i)}{\Delta x} - \left.\frac{\mathrm{d}u}{\mathrm{d}x}\right|_{x_i} = \left.\frac{\mathrm{d}^2u}{\mathrm{d}x^2}\right|_{x_i} \cdot \frac{\Delta x}{2!} + \left.\frac{\mathrm{d}^3u}{\mathrm{d}x^3}\right|_{x_i} \cdot \frac{\Delta x^2}{3!} + \dots \tag{3.41}$$

The difference between the derivative of u and its finite difference representation in Equation (3.41) corresponds to discarding everything but the first two terms of the Taylor series in Equation (3.40) and is therefore called the truncation error. It depends linearly on Δ_x, so we can write

$$\left.\frac{\mathrm{d}u}{\mathrm{d}x}\right|_{x_i} = \frac{u(x_i + \Delta x) - u(x_i)}{\Delta x} + O(\Delta x) \tag{3.42}$$

and call the forward Euler term in Equation (3.39) a first order approximant.

There is a simple way to obtain a higher order approximation. The backward difference reads

$$\frac{\mathrm{d}u}{\mathrm{d}x} \approx \frac{u(x_i) - u(x_i - \Delta x)}{\Delta x} = \frac{u_i - u_{i-1}}{\Delta x}. \tag{3.43}$$

Its truncation error can be determined using the Taylor series

$$u(x_i - \Delta x) = u(x_i) - \left.\frac{\mathrm{d}u}{\mathrm{d}x}\right|_{x_i} \cdot \Delta x + \left.\frac{\mathrm{d}^2u}{\mathrm{d}x^2}\right|_{x_i} \cdot \frac{\Delta x^2}{2!} - \left.\frac{\mathrm{d}^3u}{\mathrm{d}x^3}\right|_{x_i} \cdot \frac{\Delta x^3}{3!} + \dots \tag{3.44}$$

to be

$$\frac{u(x_i) - u(x_i - \Delta x)}{\Delta x} - \left.\frac{\mathrm{d}u}{\mathrm{d}x}\right|_{x_i} = -\left.\frac{\mathrm{d}^2u}{\mathrm{d}x^2}\right|_{x_i} \cdot \frac{\Delta x}{2!} + \left.\frac{\mathrm{d}^3u}{\mathrm{d}x^3}\right|_{x_i} \cdot \frac{\Delta x^2}{3!} + \dots \tag{3.45}$$

and thus

$$\left.\frac{\mathrm{d}u}{\mathrm{d}x}\right|_{x_i} = \frac{u_i - u_{i-1}}{\Delta x} + \mathcal{O}(\Delta x) \tag{3.46}$$

which is just as linear in Δx as the forward difference in Equation (3.42). But adding Equation (3.41) to Equation (3.45) and dividing by 2 yields

$$\frac{u(x_i + \Delta x) - u(x_i - \Delta x)}{2\Delta x} - \left.\frac{\mathrm{d}u}{\mathrm{d}x}\right|_{x_i} = \left.\frac{\mathrm{d}^3u}{\mathrm{d}x^3}\right|_{x_i} \cdot \frac{\Delta x^2}{3!} + \dots \tag{3.47}$$

where the leading error term $\left.\frac{\mathrm{d}^2u}{\mathrm{d}x^2}\right|_{x_i} \cdot \frac{\Delta x}{2!}$ was made to vanish. The result

$$\left.\frac{\mathrm{d}u}{\mathrm{d}x}\right|_{x_i} = \frac{u_{i+1} - u_{i-1}}{2\Delta x} + O(\Delta x^2) \tag{3.48}$$

is called the centered difference approximation and it converges quadratically for $\Delta x \to 0$ while still using information about the function u at two points, in this case x_{i+1} and x_{i-1}.

We assumed the points x_i to be equidistant. For two and three dimensions, the domain on which the function is defined is divided into rectangles and cuboids respectively. A unit of this partition is called cell and unlike the finite element method discussed in the next section, the function value is constant over the cell.

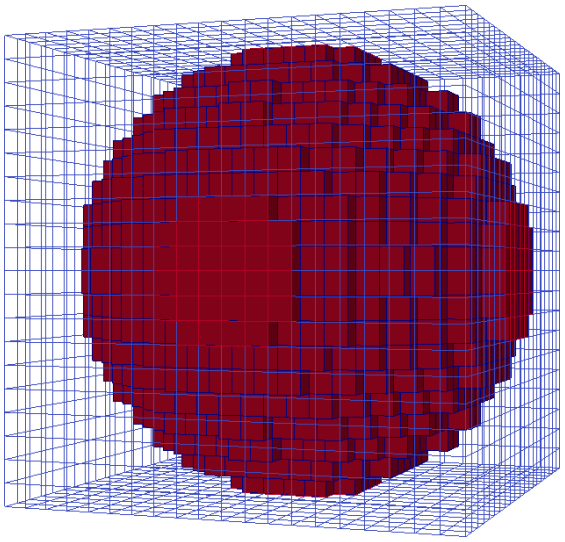


FIGURE 3.7: A sphere in a regular grid made of $20 \times 20 \times 20 = 8000$ cells.

The fact that grids are used to discretise the domain has some drawbacks. The regularity of the grid may manifest itself as anisotropy in the micromagnetic system [160]. For geometries more complex than rectangular prisms, as the example of a sphere discretised with cuboids in Figure 3.7 shows, some empty space may have to be modeled around the studied system, wasting both memory and computation time. In the context of micromagnetics it is common to set the saturation magnetisation $M_\mathrm{S} = 0$ in such regions. What remains is the problem of artifacts introduced by the discretisation of the surface of the structure.

3.5.2 The Finite Elements Method

In contrast to the FDM, the FEM deals with complex geometries better than the FDM. In two or three dimensions, the domain is generally discretised into triangles and tetrahedrons respectively instead of cuboids. This process is really an exercise in engineering judgment: A small element size and a large number of elements represent the domain as closely as possible and resolve the smallest features of the solution well. On the other hand, the incurred computational costs can be prohibitive. The sphere in Figure 3.8 is made of only 736 tetrahedrons and has a smoother surface compared to the FDM discretisation with 8000 cuboids of the sphere in Figure 3.7.

An upper bound to the element size is inherent to the micromagnetic model in which the angle between neighbouring magnetic moments needs to be small. In practice, the Bloch parameter

$$l_b = \sqrt{\frac{A}{K_1}} \tag{3.49}$$

and the exchange length

$$l_{ex} = \sqrt{\frac{2A}{\mu_0 M_\mathrm{S}^2}} \tag{3.50}$$

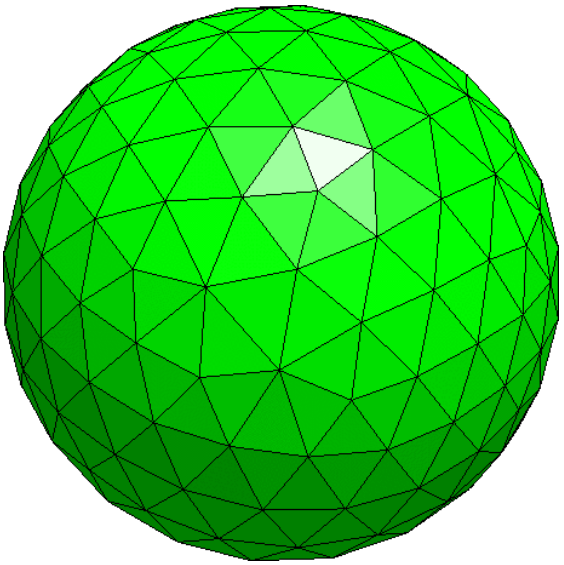


FIGURE 3.8: A triangulated sphere. The mesh is comprised of 736 volume elements.

provide such a limit [29]. For Cobalt, the Bloch parameter is $l_b = 7.6\,\mathrm{nm}$ and the exchange length is $l_{ex} = 4.9\,\mathrm{nm}$, using the exchange strength $A = 3 \times 10^{-11}\,\mathrm{J\,m^{-1}}$, the anisotropy constant $K_1 = 5.2 \times 10^5\,\mathrm{A\,m^{-1}}$ and the saturation magnetisation $M_\mathrm{S} = 1.4 \times 10^6\,\mathrm{A\,m^{-1}}$.

In the FDM, a function value is constant over the cell. Here the function values are defined on the nodal points of the finite element mesh. Inside an element, a value can be obtained by the polynomial interpolation of the nodal values using so-called basis or shape functions. A shape function of the i-th node, $\varphi_i(\mathbf{x})$ must satisfy the condition

$$\varphi_i(\mathbf{x}_j) = \delta_{ij} \tag{3.51}$$

so that it is 1 at the i-th node itself, but 0 at all other nodes. In Equation (3.51) $\mathbf{x}_j$ are the coordinates of the nodal point j. The shape functions are normalised.

$$\sum_i \varphi_i(\mathbf{x}) = 1 \tag{3.52}$$

It is then possible to think of the interpolation

$$u(\mathbf{x}) = \sum_i u_i \varphi_i(\mathbf{x}) \tag{3.53}$$

as a weighted average of the function values at the nodal points. Operations on the function work on the basis functions instead. For example, differentiation becomes

$$\nabla u(\mathbf{x}) = \sum_i u_i \nabla \varphi_i(\mathbf{x}) \tag{3.54}$$

with u_i being the value of u at the node i. Integrating over the volume is transformed into a sum over the cells.

Besides the element shape, a choice can be made with regards to the order of the polynomials used in the discretisation. Linear basis functions are common in micromagnetics [161].

After the discretisation with finite elements and the selection of interpolation functions the FEM stipulates finding the element properties. This is a matrix equation per finite element which mediates between the nodal values of the solution and other parameters, usually obtained by a variational approach or the Galerkin method. Then, using the element connectivities, the element equations would be *assembled* into a global equation system which covers the whole domain before imposing boundary conditions. This system would be sparse, symmetric and positive definite and would be solved using direct or iterative methods. The result being the nodal values.

In micromagnetics, this system of equations would have to be solved anew for every time step to obtain the effective field (c.f. Section 3.3) from the current magnetisation configuration. This can be avoided applying the box scheme used in magpar [29]

$$\mathbf{H}_{\mathrm{eff},i} = -\left(\frac{\delta E_{\mathrm{total}}}{\delta \mathbf{M}_{\mathrm{S}}}\right)_i \approx -\frac{1}{V_i}\frac{\partial E_{\mathrm{total}}}{\partial \mathbf{M}_{\mathrm{S},i}} \tag{3.55}$$

where V_i is the volume of a "box" around the node i.

The box scheme just described works well for the computation of the exchange and anisotropy fields, which are local in nature. For the computation of demagnetising field, the Fredkin-Koehler method is used [37]. It involves splitting the magnetic potential ϕ (c.f. Section 3.3.2) in two parts so that $\phi = \phi_1 + \phi_2$. Inside the magnetic region, ϕ_1 is the solution to the inhomogenous Neumann problem

$$\Delta\phi_1 = \nabla \cdot \mathbf{M}(\mathbf{r}) \tag{3.56}$$

with the boundary condition

$$\frac{\partial \phi_1}{\partial \mathbf{n}} = \mathbf{M} \cdot \mathbf{n}. \tag{3.57}$$

Outside the magnetic region $\phi_1 = 0$. This means that ϕ_2 satisfies the Laplace equation

$$\Delta\phi_2 = 0. \tag{3.58}$$

On the boundary

$$\phi_2 = \phi_1, \quad \frac{\partial \phi_2}{\partial \mathbf{n}} = 0. \tag{3.59}$$

The two equations Equations (3.56) and (3.58) are turned into two systems of linear equations using the boundary element method, a review of which can be found in Ref. [162].

3.5.3 Linear Systems of Equations

We saw in Section 3.5.2 how the FEM leads to a linear system of equations. There exist a multitude of direct and iterative methods to solve such a system numerically. Direct methods are an efficient approach when dealing with a small number of unknowns, but practical problems often lead to large and sparse systems. Storing such systems in memory is often only possible by disregarding the zero elements. Direct methods usually don't exploit this characteristic and store fully populated intermediate matrices that exceed the available memory or lead to unacceptable compute times. But since the system of equations is obtained by discretisation of the original problem in the first place, even an exact solution to the system of equations would only be an approximation of the solution to the given problem. Hence an approximate solution to the system of equations with an error of the same order of magnitude as the discretisation error is sufficient. Iterative methods suit this need well.

For a linear system of equations of the form

$$\mathbf{A}\mathbf{x} = \mathbf{b} \tag{3.60}$$

with a given right-hand side $\mathbf{b} \in \mathbb{R}^n$ and a regular matrix $\mathbf{A} \in \mathbb{R}^{n\times n}$, iterative methods determine successive approximations $\mathbf{x}_m$ to the exact solution $\mathbf{A}^{-1}\mathbf{b}$ by repeating a calculation

$$\mathbf{x}_{m+1} = \phi(\mathbf{x}_m, \mathbf{b}) \,\text{for}\, m = 0, 1, \ldots \tag{3.61}$$

and a starting vector $\mathbf{x}_0 \in \mathbb{R}^n$. An iterative method is then given by a function $\phi : \mathbb{R}^n \times \mathbb{R}^n \to \mathbb{R}^n$ and is called linear if matrices $\mathbf{M}, \mathbf{N} \in \mathbb{R}^{n\times n}$ exist, such that

$$\phi(\mathbf{x}, \mathbf{b}) = \mathbf{M}\mathbf{x} + \mathbf{N}\mathbf{b}. \tag{3.62}$$

The matrices $\mathbf{M}$ and $\mathbf{N}$ are given by the iteration procedure.

A vector $\tilde{\mathbf{x}}$ is a fixed point of the iteration procedure if

$$\tilde{\mathbf{x}} = \phi\left(\tilde{\mathbf{x}}, \mathbf{b}\right). \tag{3.63}$$

When the exact solution $\mathbf{A}^{-1}\mathbf{b}$ is a fixed point of ϕ, ϕ is called consistent to $\mathbf{A}$. The procedure is convergent when the limit

$$\tilde{\mathbf{x}} = \lim_{m\to\infty} \mathbf{x}_m = \lim_{m\to\infty} \phi(\mathbf{x}_{m-1}, \mathbf{b}) \tag{3.64}$$

exists.

We can consider a linear iteration procedure that is convergent, so $\mathbf{x}_m = \phi(\mathbf{x}_{m-1}, b)$ will converge to a fixed point $\tilde{\mathbf{x}} = \phi(\tilde{\mathbf{x}}, \mathbf{b})$, and also consistent, in which case the fixed point will satisfy Equation (3.60).

A simple iterative method is the Jacobi iteration, which transforms Equation (3.60) into

$$\mathbf{x} = \hat{\mathbf{B}}\mathbf{x} + \hat{\mathbf{b}} \tag{3.65}$$

with $\hat{\mathbf{B}} = \mathbf{I} - \mathbf{D}_{\text{diag}}^{-1}\mathbf{A}$ and $\mathbf{b} = \mathbf{D}_{\text{diag}}^{-1}\mathbf{b}$. The diagonal matrix $\mathbf{D}_{\text{diag}}$ with the diagonal of $\mathbf{A}$ needs to be nonsingular. The fixed point iteration to apply is

$$\mathbf{x}_{n+1} = \hat{\mathbf{B}}\mathbf{x}_n + \hat{\mathbf{b}}. \tag{3.66}$$

Without knowing the exact solution $\tilde{\mathbf{x}}$ it is impossible to compute the n-th error vector

$$\mathbf{d}_n = \mathbf{x}_n - \tilde{\mathbf{x}} \tag{3.67}$$

but convergence can be tested by using the n-th residual

$$\mathbf{r}_n = \mathbf{b} - \mathbf{A}\mathbf{x}_n \tag{3.68}$$

instead. We can relate the residual $\mathbf{r}_n$ and the error vector $\mathbf{d}_n$ via

$$\begin{aligned} \mathbf{r}_n &= \mathbf{b} - \mathbf{A}\mathbf{x}_n && (3.69)\\ &= \mathbf{A}\tilde{\mathbf{x}}_n - \mathbf{A}\mathbf{x}_n && (3.70)\\ &= -\mathbf{A}\mathbf{d}_n. && (3.71) \end{aligned}$$

In fact, we can rewrite the Jacobi iteration as

$$\mathbf{x}_{n+1} = \mathbf{x}_n + \mathbf{r}_n \tag{3.72}$$

using

$$\begin{aligned} \mathbf{r}_n &= \mathbf{b} - \mathbf{A}\mathbf{x}_n && (3.73)\\ &= \mathbf{B}\mathbf{x}_n + \mathbf{b} - \mathbf{x}_n && (3.74)\\ &= \mathbf{x}_{n+1} - \mathbf{x}_n && (3.75) \end{aligned}$$

assuming $\mathbf{D}_{\text{diag}} = \mathbf{I}$ and defining $\mathbf{B} = \mathbf{I} - \mathbf{A}$. The residual can also be computed via the recursion

$$\mathbf{r}_{n+1} = \mathbf{r}_n - \mathbf{A}\mathbf{r}_n = \mathbf{B}\mathbf{r}_n \tag{3.76}$$

obtained by multiplying Equation (3.72) by $-\mathbf{A}$. By induction we can conclude from Equation (3.76) that

$$\mathbf{r}_n = p_n(\mathbf{A})\,\mathbf{r}_0 \in \text{span}\{\mathbf{r}_0, \mathbf{A}\mathbf{r}_0, \ldots, \mathbf{A}^n\mathbf{r}_0\} \tag{3.77}$$

where $p_n(\xi) = (1-\xi)^n$ is a polynomial of degree n. With Equation (3.72) we realise that

$$\mathbf{x}_n = \mathbf{x}_0 + \mathbf{r}_0 + \ldots + \mathbf{r}_{n-1} = \mathbf{x}_0 + q_{n-1}(\mathbf{A})\,\mathbf{r}_0 \tag{3.78}$$

with a polynomial q_{n-1} of degree $n-1$. Comparing Equation (3.78) with Equation (3.77) we see that x_n lies in the affine space $\mathbf{x}_0 + \text{span}\{\mathbf{r}_0, \ldots, \mathbf{A}_{n-1}\mathbf{r}_0\}$ obtained when the subspace of $\mathbf{r}_{n-1}$ by $\mathbf{x}_0$. The subspace $K_n(\mathbf{A}, \mathbf{r}_0) = \text{span}\{\mathbf{r}_0, \mathbf{A}\mathbf{r}_0, \ldots, \mathbf{A}_{n-1}\mathbf{r}_0\}$ is called a Krylov subspace and it gives its name to a class of methods called Krylov subspace methods that share the idea of minimising the residual over the subspace formed by the sequence of increasing matrix powers multiplied by the initial residual. Note that the matrix $\mathbf{A}$ is only used to evaluate a matrix vector product. This means that for computation, $\mathbf{A}$ only needs to be given as an operator, or more precisely, a routine.

Possibly the most well-known Krylov subspace method, applied to symmetrical, positive definite matrices is the conjugate gradient (CG) method developed independently by Hestenes and Stiefel [163]. It introduces a quadratic function

$$\Phi(\mathbf{x}) = \frac{1}{2}\mathbf{x}^\intercal\mathbf{A}\mathbf{x} - \mathbf{b}^\intercal\mathbf{x} + \gamma \tag{3.79}$$

which is convex and has a unique minimum so that the gradient of Φ is the residual:

$$\nabla\Phi(\mathbf{x}) = \mathbf{A}\mathbf{x} - \mathbf{b} \tag{3.80}$$

$$= -\mathbf{r}. \tag{3.81}$$

Finding the $\mathbf{x}$ that minimises Φ makes the gradient and thus the residual disappear, which is akin to solving Equation (3.60).

Krylov subspace methods are still an active research field [164].

The convergence criteria of iterative methods, show the benefit of a small condition number of the matrix $\mathbf{A} \in \mathbb{R}^{n\times n}$ of the linear system $\mathbf{A}\mathbf{x} = \mathbf{b}$ [165]. This motivates an equivalent reformulation of the linear system with the goal of reducing the condition number of $\mathbf{A}$. The preconditioner is a matrix $\mathbf{P}$, such that $\mathbf{P}^{-1}\mathbf{A}$ has a smaller conditioner number than $\mathbf{A}$. As long as the matrix $\mathbf{P}$ is nonsingular, both

$$\mathbf{A}\mathbf{P}^{-1}\mathbf{P}\mathbf{x} = \mathbf{b} \tag{3.82}$$

and

$$\mathbf{P}^{-1}\left(\mathbf{A}\mathbf{x}-\mathbf{b}\right)=0 \tag{3.83}$$

give the same solution. The system in Equation (3.82) is called right preconditioned, while the system in Equation (3.83) is called left preconditioned. There is a performance trade-off when chosing the right preconditoner, since $\mathbf{P}^{-1}$ is applied at each step of the iterative solver. One extreme would be chosing the identity matrix as preconditioner, so $\mathbf{P}=\mathbf{I}$. This results in $\mathbf{P}^{-1}\mathbf{A}=\mathbf{I}\mathbf{A}=\mathbf{A}$, which corresponds to no preconditioning at all. On the other hand, chosing $\mathbf{P}=\mathbf{A}$ yields $\mathbf{P}^{-1}\mathbf{A}=\mathbf{I}$, which has condition number 1. While such a system would require only one iteration for convergence, the application of the preconditioner would be as difficut as solving the original system. The secret of a performant preconditioner is thus keeping $\mathbf{P}^{-1}$ simple, while reducing the number of iterations needed for the solution of the linear system.

A simple example of preconditioner is the Jacobi preconditioner. For Jacobi preconditioning, $\mathbf{P}$ is set to the diagonal of the matrix $\mathbf{A}$.

$$\mathbf{P}_{\text{Jacobi}}=\text{diag}\left(\mathbf{A}\right) \tag{3.84}$$

Modern, sophisticated codes work in a matrix-free method, i.e. the coefficient matrix $\mathbf{A}$ is not stored explicitly. Instead, matrix-vector products are evaluated. The Jacobi preconditioner would then read

$$\mathrm{P}_{ij}^{-1}=\frac{\delta_{ij}}{\mathrm{A}_{ij}}, \tag{3.85}$$

assuming $A_{ii}\neq 0\,\forall i$. For a diagonally dominant matrix $\mathbf{A}$, Jacobi preconditioning is very efficient.

Such techniques of reformulation constitute the proconditioning of a linear system and have been proven to be effective at accelerating and stabilising Krylov subspace methods both for model problems and practical applications. Besides the choice of iterative method, the choice of preconditioner is of vital importance for the resulting procedure. A comprehensive overview of preconditioning techniques is given in Ref. [166].

In the context of micromagnetics, the solving of linear systems of equations is involved both in the computation of the demagnetising field, as seen in Section 3.3.2, as well as for certain numerical integration schemes, as the next section will show. The advantage of proper preconditioning for the numerical integration of magnetic systems is known for the case of finite element discretisation [167]. For the computation of the demagnetising field, a recent paper proposes a preconditioned nonlinear conjugate gradient method to be used in energy minimisation simulations [168].

3.5.4 Time Integration of the LLG Equation

The explicit form of the LLG given by Equation (3.11) in Section 3.2 together with an initial magnetisation configuration poses an initial value problem of the form

$$m'\left(t\right)=f\left(t,m\left(t\right)\right),\, m\left(0\right)=m_0 \tag{3.86}$$

for every value of $\mathbf{m}$ after the discretisation. It can be integrated simply using an Euler step similar to the one given by Equation (3.40):

$$\mathbf{m}\left(\mathbf{r}, t+\Delta t\right) = \mathbf{m}\left(\mathbf{r}, t+\Delta t\right) + \frac{\mathrm{d}\mathbf{m}}{\mathrm{d}t}\Delta t \tag{3.87}$$

or for instance using an explicit Runge-Kutta method detailed in Ref. [169]. However, due to the comparatively high strength of the exchange interaction compared to the other energies in the system the LLG equation is quite stiff and explicit methods become computationally expensive. It is for this reason that micromagnetic softwares generally either use explicit methods with an adaptive time step or implicit methods, like the backward differentiation formula (BDF). The BDF exists with a variable order k of the polynomial q_k used to interpolate the last k approximations $m_{n+1-k}, \cdots, m_n$ to the solution and the unknown value m_{n+1}:

$$q_k\left(t\right) = \sum_{j=0}^{k} l_j\left(t\right) m_{n+1-j}, \tag{3.88}$$

where $l_j\left(x\right)$ are the Lagrange polynomials for the steps $t_{n+1-k}, \cdots, t_{n+1}$. A system of equations for m_{n+1} is obtained when requiring that the polynomial satisfies the differential equation for t_{n+1}:

$$q'\left(t_{n+1}\right) = f\left(t_{n+1}, m_{n+1}\right). \tag{3.89}$$

Assuming a constant step size $h = t_{i+1} - t_i$, the BDF needs the starting points $m_1, \cdots, m_{k-1}$ which can be determined with single step methods after which it can continue with the recursion

$$m_{n+1} = \sum_{j=0}^{k-1} \alpha_j m_{n-j} + h\beta f\left(t_{n+1}, m_{n+1}\right) \tag{3.90}$$

with the coefficients α_j, β for a given order k to be found in numerical analysis books.

The LLG equation conserves the magnitude $\|\mathbf{m}\| = 1$ of the magnetisation because it contains only terms perpendicular to $\mathbf{m}$. Due to the numerical inaccuracy which accrues during the time integration, this property is not conserved which has lead to the development of so-called geometrical integrators specific to the LLG equation [170, 171]. Alternatively, the problem can be remedied by including a longitudinal correction term called relaxation term proportional to

$$\sqrt{\left(\frac{\partial \mathbf{m}}{\partial t}\right)^2}\left(1 - \mathbf{m}^2\right)\mathbf{m} \tag{3.91}$$

to the LLG equation (Ref. [28] and Eq. 4 in Ref. [172]), which scales $\mathbf{m}$ back to its length of 1.

To integrate the stochastic LLG equation, a second-order Heun scheme can be used [78, 173]. New integration schemes have been proposed, that purport to allow for larger time steps [174, 175].

3.5.5 Nudged Elastic Band Method

It is also possible to run micromagnetic simulations without necesseraly integrating the magnetic configuration of a system over time. For instance, in the context of data storage, it may be preferable to be able to directly compute the energy barrier that separates two equilibrium states in a magnetic system, that together are used to represent one bit of information. The energy barrier indicates the energy of thermal fluctuations or excitations needed to drive the system from one state to the other, allowing the analysis of the stability of the system and its potential for application for data storage. A method allowing the calculation of minimum energy transitions between equilibrium states is the nudged elastic band (NEB) method [176].

The NEB introduces the notion of an image of a system. In a discrete spin model, each node on a lattice with P nodes has a three dimensional spin vector $\mathbf{s}_i$, $i \in \{0, 1, \cdots, P-1\}$ associated to it. An image $\mathbf{Y}$ is then the entirety of spin vectors, i.e. $(\mathbf{s}_0, \mathbf{s}_1, \cdots, \mathbf{s}_{P-1})$, and captures the magnetic configuration of the system. A band is a set of N images $\mathbf{Y}_i$, $i \in \{0, 1, \cdots, N-1\}$ which represent one system in different magnetic configurations. The first image $\mathbf{Y}_0$ and the last image $\mathbf{Y_{N-1}}$ of the band are fixed, and correspond to the two equilibrium states for which to find the energy barrier. The remaining images $(\mathbf{Y}_1, \mathbf{Y}_2, \cdots, \mathbf{Y}_{N-2})$ in-between are set to an initial sequence of magnetic configurations. The NEB method is an iterative process in which the images in the band are evolved to find the lowest energy path between $\mathbf{Y}_0$ and $\mathbf{Y}_{N-1}$. The process is driven by an effective force $\mathbf{G}_i = \mathbf{G}(\mathbf{Y}_i)$ with a component that is perpendicular to the energy band and a spring force $\mathbf{F}$ (parallel to the band) between images to keep them equally spaced in the phase space. It reads

$$\mathbf{G}_i = -\nabla_\mu E(\mathbf{Y}_i)|_\perp + \mathbf{F}(\mathbf{Y}_i)|_\parallel. \tag{3.92}$$

The gradient in Equation (3.92) is with respect to the spin moment $\mu = \mu_\mathrm{s}\mathbf{s}$, with the magnetic moment $\mu_\mathrm{s} = g\mu_B S$, where S is the total average spin per lattice site. The gradient can be evaluated using the effective field in a manner analogous to Equation (3.7):

$$\nabla_\mu E(\mathbf{Y}_i) = \frac{1}{\mu_\mathrm{s}} \frac{\partial E}{\partial \mathbf{s}} = -\mathbf{H}_\mathrm{eff} \tag{3.93}$$

Without the spring force, the images would cluster around the equlibrium states. It can be defined by

$$\mathbf{F}\left(\mathbf{Y}_i\right) = k\left(|\mathbf{Y}_{i+1} - \mathbf{Y}_i| - |\mathbf{Y}_i - \mathbf{Y}_{i-1}|\right)\mathbf{t}_i \tag{3.94}$$

where the vectors $\mathbf{t}_i$ are tangents to the energy band, and using an appropriate formula for the geodesic distance between images [176]. The constant k typically has values between 10^2 and 10^5 [177].

A minimum energy path is found when the perpendicular component of the effective force $\mathbf{G}$ vanishes. In this case, the images can only move along the band. When the NEB method converges, the resulting band crosses an energy maximum along a single direction in phase space. This energy maximum is known as saddle point and determines the energy barrier between the

equlibrium states. This path is not unique however. The minimum energy transition is given by the path with the smallest energy barrier.

Using an earlier version of this method [178], Dittrich et al. computed the minimum energy transitions in simple magnetic systems, such as small or elongated particles [179]. Corroborated with analytic theory, they showed the transitions in these particles is characterised by coherent rotation, respectively domain wall propagation.

3.6 Micromagnetic Standard Problems

We saw at the beginning of Section 3.5 that except for very simple cases, the micromagnetic equations cannot be solved analytically. Therefore, it is appropriate to solve them numerically using computer simulations and the discussion of the numerical methods involved followed. The field studying magnetic materials using simulations, and evolving the techniques, became known as computational micromagnetics.

The challenges of good software engineering facing them (c.f. Chapter 1) preoccupied researchers in the micromagnetic modeling activity group [4] at the NIST Center for Theoretical and Computational Materials Science. In 1995, they formed a working group to conceive of a set of "standard" problems for the verification of micromagnetic softwares [180, 181]. The selected problems should be simple and have experimentally accessible material parameters and applied fields. The provided solutions should contain the specific details of input parameters and solution methods. Further, it was agreed to share the problems and their solutions over the internet. Five standard problems are available [4].

The first standard problem models a $1 \times 2\,\mathrm{nm}$ Permalloy rectangle that is 20 nm thick and requires the computation of its hysteresis loop as an external magnetic field is stepped from 50 mT to $-50\,\mathrm{mT}$ and back. The hysteresis loop is the curve of $\mathbf{M}$ versus $\mathbf{H}$. The problem specification includes the saturation magnetisation, exchange interaction strength, unaxial anisotropy with the easy axis aligned with the long axis of the rectangle, as well as the demagnetising field. The simulation is carried out for two configurations of the external field: it is to be parallel once to the long axis and once to the short axis of the rectangle. The devil is already in the detail as the external field should be rotated 1° counterclockwise off the nominal axis because of the presence of meta-stable symmetric states.

The second standard problem models a system similar to the first to compute coercivity, that is the magnitude of the external field at which the projection of the magnetisation along the field is zero, and remanence, so the magnetisation at zero field.

Moving from thin films to a cubic geometry, the third micromagnetic standard problem is concerned with the energy of two different magnetic configurations. The task is computing the single domain limit, that is the size of the cube for which the possible flower (single-domain) and vortex state have the same energy. This problem is special insofar as the other problems keep the geometry fixed and modify the energies instead of the other way around like here. With a graphical software such as OOMMF, the user would have to manually configure meshes of different sizes and rerun the simulation twice for every mesh size (once for each state) to find the single domain limit. Alternatively, the user could write a program from scratch to drive OOMMF

using its limited batch mode and to analyse its outputs. Nmag was written as an extension to Python to gain its flexibility, but the creation of the meshes would be outside of the scope of the software as they are imported from separate files. For increased reproducibility, FinMag and Fidimag follow a code as configuration approach, and it includes meshing. We will see thus in Section 5.4 how by using FinMag the standard problem 3 can be expressed in a single program that delegates finding the single domain limit to a simple bisect algorithm.

The fourth standard problem returns to using thin Permalloy films and is interested in magnetisation reversal after an external field of sufficient magnitude is applied to an equilibrium state. By following the components of the spatially averaged magnetisation as a function of time, it is the first problem to investigate the dynamics of the LLG equation. The results should also be independent of the discretisation size. The solution to the fourth micromagnetic standard problem with Fidimag is presented in Section 4.4.

The fifth standard problem [182] introduces spin torque dynamics. The effects of a polarised current on a 10 nm thin 100×100 nm Permalloy square are simulated. A vortex is present in the simulated sample and its movement in response to the current applied in-plane are recorded, to compare its trajectory and final position.

More standard problems have been proposed. Ref. [183] describes spin wave propagation in a thin Permalloy strip by means of dispersion curves and presents simulations carried out with OOMMF and Nmag. We repeated the study with our finite element software FinMag and found the results to depend somewhat on discretisation size. Full replication was possible after obtaining the exact mesh files used by the authors of the study.

The finite differences code Fidimag that will be the subject of the next chapter was featured in a proposal for a micromagnetic standard problem for materials with the Dzyaloshinskii-Moriya interaction described in Section 3.3.3 [184]. FinMag was used in a proposal for a micromagnetic standard problem computing ferromagnetic resonance spectra under a microwave field, as will be described in Section 5.6.

Chapter 4

Fidimag – a finite difference software for atomistic and micromagnetic simulations

Fidimag is an open-source scientific code for the study of magnetic materials at the nano- or micro-scale using either atomistic or finite difference micromagnetic simulations, which are based on solving the Landau-Lifshitz-Gilbert equation. In addition, it implements simple procedures for calculating energy barriers in the magnetisation through variants of the nudged elastic band method. This computer software has been developed with the aim of creating a simple code structure that can be readily installed, tested, and extended. An agile development approach was adopted, with a strong emphasis on automated builds and tests, and reproducibility of results. The main code and interface to specify simulations are written in Python, which allows simple and readable simulation and analysis configuration scripts. Computationally costly calculations are written in C and exposed to the Python interface as Cython extensions. Docker containers are shipped for a convenient setup experience. The code is freely available on GitHub and includes documentation and examples in the form of Jupyter notebooks.

The simulation of magnetic materials falls into several paradigms, depending on the length scales, materials and phenomena of interest. For many materials, the spin of atoms can be assumed to be localised around the atom, and a classical approximation can be made in which the atomic spin is treated as a point dipole - this is the classical Heisenberg model. The continuum limit of this theory, known as micromagnetism, allows the computational treatment of much larger systems, though this excludes the study of materials which exhibit antiferromagnetism and ferrimagnetism.

Fidimag is a software library which allows researchers to model magnetic materials using both the classical Heisenberg and micromagnetic models. Users of the software provide the magnetic parameters of the material under study, the system geometry, and a set of initial conditions. The simulation can occur under different kind of dynamics, with Fidimag implementing the Landau-Lifshitz-Gilbert (LLG), the stochastic LLG (SLLG), and several spin-transfer torque variations of the LLG equation. From this initial setup, the user can then choose to evolve the system either through time, and hence study the magnetisation dynamics, or to “relax” the

system to find its metastable energy states. In addition, the software implements the nudged elastic band method [176] to find minimum energy paths and the size of energy barriers between different configuration states. The method is described in greater detail in Section 4.1.

Fidimag has been used to obtain the results in several scientific publications [185, 186, 187, 188]. The supplementary data to ref. [187] demonstrates that with Docker, the process of correctly reproducing the results can be simplified to a single, reliable Makefile instruction [189].

4.1 Finite Difference Simulations of Magnetic Materials

At the atomic level, the magnetic effects in magnetic materials originate from two angular momentum terms from electrons: their orbital motion around the nucleus and, mainly, from a quantum property of electrons called spin[1]. Within a semi-classical model, the sum of their contributions can be described by a three dimensional vector $\mathbf{S}_i = \mu_i \mathbf{s}_i$ representing a magnetic moment (a magnetic dipole) μ_i for every atom i, with a specific direction $\mathbf{s}_i$. This idea originates from Heisenberg's model Hamiltonian to describe magnetic interactions [190, 191]. In magnetic solids, atoms arrange in a lattice with a specific periodic crystal geometry. For example, a cubic lattice is made of repeated cells where atoms and their corresponding magnetic moments lie at the corners of this cubic cell.

An atom is at the scale of a few Angstrom, thus in large systems, which are at the nano-scale, it would be necessary to describe the system using thousands of spins, which is computationally expensive. Therefore, it is possible to approximate the material as a continuum, where the field from discrete magnetic moments turns into a continuum field called magnetisation that depends on the space coordinates $\mathbf{r}$, assuming that neighbouring spins do not change drastically in direction [190, 191]. This means the limit from the discrete atoms into the space dependent field $\mathbf{s}_i \to \mathbf{M}(\mathbf{r})/M_\mathrm{s} = \mathbf{m}(\mathbf{r})$. The magnitude of the field is measured through the saturation magnetisation M_s, which is defined as a magnetic moment per unit volume, *i.e.* $M_\mathrm{s} = \mu/\Delta V$, with ΔV as the volume of a unit cell of the discrete crystal lattice. This theory of magnetism in the continuum limit is known as micromagnetics. Numerically, the magnetisation field can be discretised into a mesh of magnetic moments $\mathbf{M}_i$.

To describe a magnetic system, we firstly specify its geometry. If we simulate this system using a discrete spin model, we have to generate a lattice of magnetic moments with a specific arrangement of atoms (according to the crystal symmetry) such that they describe this geometry. In the case of micromagnetics we use the finite differences numerical technique, thus the sample is discretised into cuboids and each one of them represents a volume with uniform magnetisation inside that volume.

Dynamics Magnetic phenomena emerge from the interactions between magnetic moments and their interaction with external and internal magnetic fields, the later including dipolar interactions and anisotropic interactions, among others. These interactions depend on the material and specify the total energy of the system. In general, the spins perform a precessional motion following directions set by the magnetic interaction. The dynamics of the magnetic moments is given

[1] For this reason when we mention spins we strictly refer to the total angular momentum or magnetic moments.

by the Landau-Lifshitz-Gilbert (LLG) equation [190, 191]. Within the discrete spin model this dynamical equation for a single spin $\mathbf{s}$, reads

$$\frac{\partial \mathbf{s}}{\partial t} = -\gamma\, \mathbf{s} \times \mathbf{H}_{\text{eff}} + \frac{\alpha_{\text{G}}\gamma}{\mu}\, \mathbf{s} \times \mathbf{s} \times \mathbf{H}_{\text{eff}}\,, \tag{4.1}$$

at zero temperature, where $\mathbf{H}_{\text{eff}}$ is an effective field that contains the sum of every magnetic interaction present in the system, γ is the Gilbert gyromagnetic ratio constant, which sets the time scale of the spin motion, and $0 \leq \alpha_{\text{G}} \leq 1$ is the Gilbert damping. The first term in equation 4.1 describes a precessional motion of spins around the effective field and the second term is a dissipative term that make spins follow the effective field direction.

In micromagnetics this equation has the same structure

$$\frac{\partial \mathbf{M}}{\partial t} = -\gamma \mathbf{M} \times \mathbf{H}_{\text{eff}} + \frac{\alpha_{\text{G}}\gamma}{M_{\text{s}}} \mathbf{M} \times \mathbf{M} \times \mathbf{H}_{\text{eff}}\,. \tag{4.2}$$

Fidimag can solve the non-linear differential equations 4.1 and 4.2, depending on the specified theoretical model. Accordingly, it is necessary to specify an initial magnetic configuration for the spin directions, and the magnetic interactions involved in the system. Currently, Fidimag has the following interactions implemented in the code,

Exchange Favours the parallel or anti-parallel alignment of neighbouring spins. The exchange can be computed using nearest neighbour approximations in the classical Heisenberg case. The continuum micromagnetic model only allows parallel alignment.

Dipolar interactions Product of the interaction of a spin with the field generated from the other spins in the system

Dzyaloshinskii-Moriya Favours a canted alignment between neighbouring spins

Anisotropy Sets directions along which the system energy decreases when spins align towards them

Zeeman Product of the interaction of spins with an external magnetic field

The underlying equations that were used to implement these interactions can be found in Ref. [192].

One method for finding energy minima in the system's energy landscape, which is implemented in the code, is to evolve the system with the LLG equation, since the spins will precess until a stable configuration is attained, *i.e.* a local or global minimum of energy, that depends on the initial magnetic configuration. We refer to this process as relaxation.

Dynamical effects such as the generation of spin waves, ferromagnetic resonance, domain wall motion, among other multiple magnetism related phenomena, are also given by the evolution of spins with the LLG equation. Variations of this equation are obtained when applying electric currents or temperature, which can also be specified in the code.

Constraint At zero temperature the length of the magnetic moments and the magnetisation vector is fixed. This condition must be specified in the equation of motion of spins explained in the last section, which sets a constraint to the spin length. Multiple magnetic phenomena can be explained at a zero-temperature formalism thus the majority of Fidimag's equations are implemented in this regime.

Although this constraint is implicit in the LLG equation, numerically the spin length varies when integrated: rounding may cause some drift. As reviewed in Section 3.5.4, a correction term can be added to the right hand side of equations 4.1 and 4.2 to address this problem. It is proportional to

$$\sqrt{\left(\frac{\partial \mathbf{s}}{\partial t}\right)^2}\left(1-\mathbf{s}^2\right)\mathbf{s}\,. \tag{4.3}$$

and similar in the micromagnetic model, by setting $\mathbf{s} \to \mathbf{m}$. This term is zero physically and makes the spin length increase when it is less than the unit and decrease otherwise. Another way to to fix the magnitude of $\mathbf{s}$ would be to renormalise the vector at every time step. This is the approach followed by Magpar. Because it introduces additional jitter into the time integration we opted for the relaxation term instead.

Finding the lowest energy cost to drive a magnetic system from one equilibrium state towards another, also known as energy barrier, has become a relevant problem for the analysis of the stability of magnetic structures. An energy barrier is then associated to the transition path, between two states, that requires the least energy. This is important, for example, for the potential application of a magnetic structure in a technological device since an energy barrier can be used to estimate the lifetime of the structure against energy fluctuations from excitations such as thermal noise, present at finite temperatures. As we saw in Section 3.5.5, the nudged elastic band method (NEBM) is a technique for the calculation of minimum energy paths, and hence energy barriers. It was first used in chemistry to study molecular transitions [178]. It is based on fixing two equilibrium states, which can be local or global minima, and making copies, called images, of the system in different configurations between these extrema. This sets a *band* of images and each one of the images will have a different energy in the energy landscape associated to the system. The algorithm will iteratively find a path in the energy landscape that decreases the energy cost of the *band*, trying to keep the images equally spaced in the energy landscape using an inter-image spring force, to avoid images move towards the minima at the extremes of the band. After relaxation, the minimum energy path will cross one or more maxima of energy, which set the energy barrier magnitude. An optimisation of the original algorithm has recently been published by Bessarab et al. [176], where geodesic distances are defined in the energy landscape.

The NEBM was implemented in Fidimag by Cortés-Ortuño [187], and it can be used both within micromagnetics and a discrete spin model. The optimised version, called Geodesic NEBM is the one that performs more efficiently, and combines Cartesian coordinates for the description of the spins and geodesic distances in the energy landscape. Original versions of the algorithm, which only use Cartesian or spherical coordinates, can present convergence issues for systems with more complex magnetic configurations, such as vortices. To run a NEBM simulation, it is necessary to specify two equilibrium states, the number of images in between, and an initial state for the internal images. This initial configuration can be set by either manually creating a series

of images in different magnetic states or by using a linear interpolation for the spin directions, which is implemented in the code to generate the configurations automatically.

A detailed review of the method can be found in Ref. [176]. A study of the thermal stability of skyrmions, which are vortex-like magnetic configurations, that uses the NEBM implemented in Fidimag, can be found in [187]. In addition, we tested our code with the skyrmions example from [176], making a repository with the simulation details in [193].

Fidimag provides both micromagnetic and atomistic simulation capabilities and is the only software that allows switching between the two modelling paradigms. The most notable micromagnetic code is OOMMF, which had its alpha release in 1998. OOOMMF is written in C++ and Tcl/Tk. Fidimag is mostly implemented in the Python programming language, with performance critical parts realised in C and linked in via Cython. As Python is well established in the scientific community, it presents a lower barrier to entry for programmers [194]. Further, Python has a reputation for being easy to learn for beginners [195, 196]. This has informed the decision to build Fidimag as a software library instead of a GUI-driven program. Indeed, Fidimag respects best practices for Python's module and namespacing system.

This library model of execution, where Fidimag is imported into the namespace of a Python program empowers the user to deal with the complexity of the batch processing of simulations in a more easily tested and reproducible way compared to ad-hoc shell commands and specialised batch modes in software with graphical user interfaces [197]. The standard problem #3 especially examplifies how naturally Fidimag allows higher order logic thanks to these choices. In it, two possible magnetic configurations in cubes of increasing sizes are to be studied to determine when they are equally energetically favourable.

```
1  """
2  Solution to mag standard problem #3 with fidimag.
3      http://www.ctcms.nist.gov/~rdm/mumag.org.html
4
5  """
6  from .cube_sim import energy_difference
7  from scipy.optimize import bisect
8
9  single_domain_limit = bisect(energy_difference, 8, 8.5, xtol=0.1)
10 print("L = {:.2} nm.".format(str(single_domain_limit))
```

LISTING 4.1: Solution to the standard problem #3. The code that computes the energy difference between the two possible magnetic configurations has been abstracted into the function `energy_difference` in the module `cube_sim` that is imported in line 6. The function `energy_difference` can then be handed off to a bisect method provided by SciPy to find the cube size for which both configurations have the same energy.

In Listing 4.1 the logic of setting up and running the simulations, as well as comparing the total energies of the magnetisation in the two possible configurations has been abstracted into a function called `energy_difference`. Finding the cube size of equal energies called `single_domain_limit` then involves nothing more than another function call, this time to a bisection method called `bisect` provided by SciPy [198].

A similar approach had been used in Nmag and FinMag, where it has proven to be successful [28]. Unlike Nmag however, Fidimag doesn't require bundling the software with a modified Python binary. The approach is now seen in other micromagnetic packages, for example micromagnum [25]. The Jupyter/OOMMF project follows a hybrid approach and harnesses the proven capabilities of the OOMMF binary with a Python interface optimised for Jupyter notebooks [199].

Vampire is a performant atomistic code that defines its own declarative syntax [200]. It comes in two versions, either a binary with an installation script or the code which needs to be compiled from source. In contrast, while the traditional installation methods are available, installing Fidimag with Docker is as easy as `docker pull fidimag/notebook`.

Best practices recommend using version control for any kind of computational endeavours [201]. Because there are no command line parameters to record or options set in a GUI program Fidimag offers a one-to-one mapping of the simulation code to its result. This is a boon for the goal of reproducibility that can't be overstated. For example, Fidimag plays well with Sumatra [202] which allows for the automatic tracking of the computations run. Jupyter Notebooks [203] are used for interactive tutorials and included in the extensive documentation [204]. There are tests that ensure that these notebooks are executing consistently, without errors, and that the execution of the stored inputs match the stored outputs. For this, a py.test plugin called nbval was developed and made available to the open source community [205]. Fidimag has an automated testing procedure that runs on a public continuous integration system [206] Fidimag's simple installation procedure via Docker, the extensive documentation including interactive tutorials stored in Jupyter notebooks and the accessible interface have made it not only a useful tool for research as stated before, but also for teaching [207].

4.2 Implementation

The code is stored using the distributed version control system git and new features are developed in branches. Automated testing and builds of the software were priorities from the start.

The code base consists of roughly 5000 lines of Python code and 4000 lines of C code and cython extensions[2]. The tests and examples add another 5500 lines of Python.

The source code is split into three major sections - 'atomistic', 'micro' and 'common' which contain code specifically for classical Heisenberg simulations, micromagnetic simulations, and for both respectively, as can be seen in Figure 4.1. There are many common components which can be shared between both types of simulation. The primary data in both atomistic and micromagnetics is stored in vector fields, and the mechanisms for passing this data through the simulation are kept in a common base 'Sim' class. This allows, for example, the saving of the magnetisation progressive steps in the simulations to be handled identically for both types of simulation. Fidimag comes with visualisation and data saving utilities out of the box.

Mesh and lattices As we mentioned earlier, we define a magnetic system by setting its geometry. How the geometry is approximated depends on the chosen model.

[2]Numbers obtained with cloc version 1.60

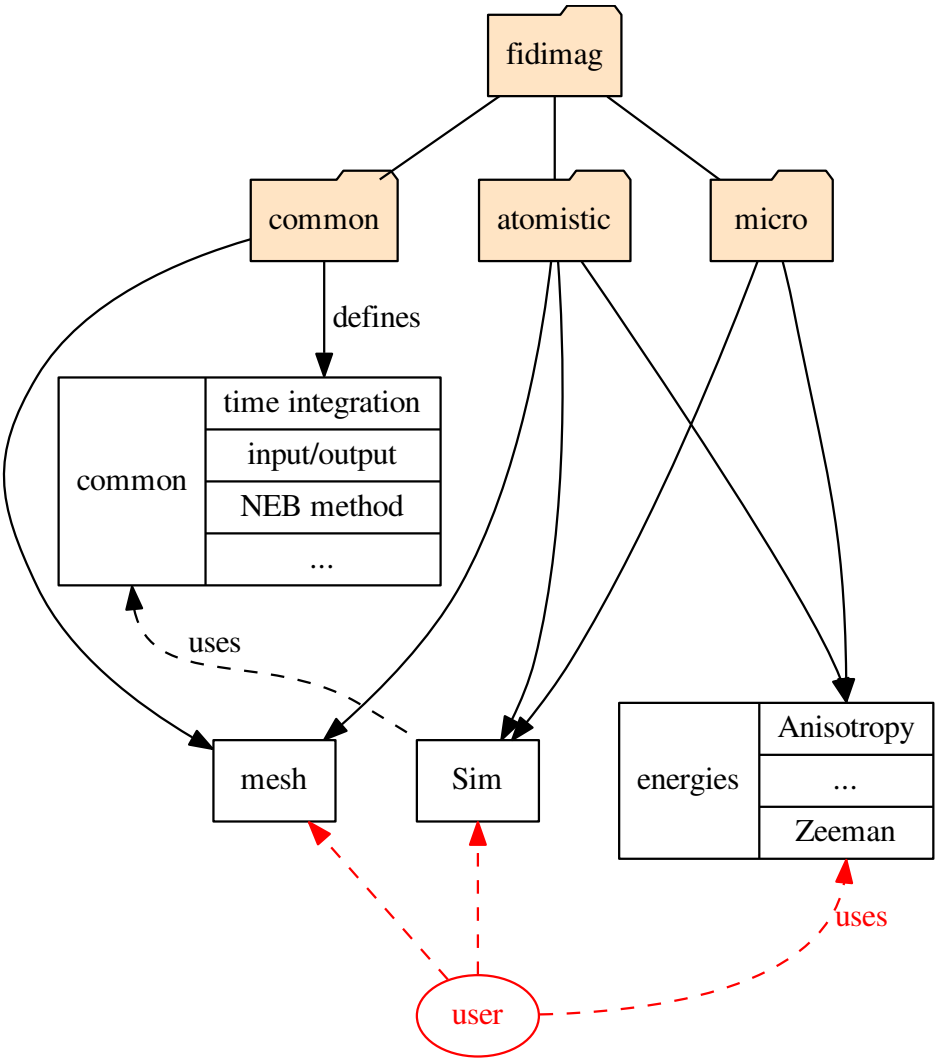


FIGURE 4.1: Architecture of Fidimag as reflected by the directory structure and Python modules. Code useful to micromagnetic and atomistic simulations was extracted into a `common` namespace. The red arrows point to user-facing parts of the system.

Within the continuum approximation, Fidimag uses finite differences which means dividing the sample into a mesh of cubes, as shown in Fig. 4.2a, where each cuboid centre represents the position of a magnetic moment vector. Accordingly, derivatives for the calculation of magnetic interactions and energies are discretised using differences between neighbouring mesh sites. The cuboid mesh is coded in the `CuboidMesh` class located in Fidimag's `common` directory. Distance between the centres and the number of cuboids in the three spatial directions are specified by the user. The indexes of every mesh site are labelled following the $x \to y \to z$ direction, as shown in Fig. 4.2a. To keep track of the neighbouring sites of every mesh site, the mesh class has a `neighbours` method defined as an $N \times 6$ array, where N is the total number of cuboids, that stores the indexes of nearest neighbours, with the value -1 for non-material sites. Correspondingly, every row represents the 6 nearest neighbours in the $(-x, +x, -y, +y, -z, +z)$ order. For instance,

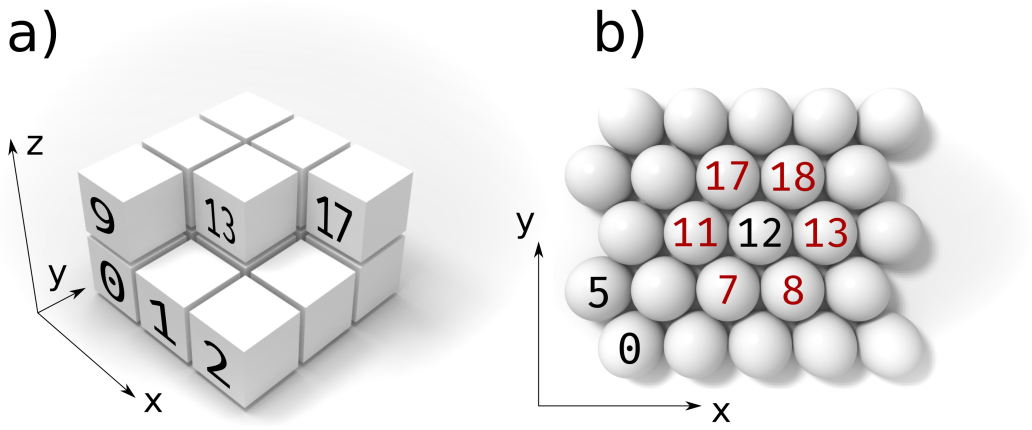


FIGURE 4.2: The cuboid mesh (a) is used in the continuum approximation. In the atomistic case, the ordering of the atoms or molecules is given by the crystal structure of the magnetic solid. The meshes then represent the 2D/3D square lattice (a) and the 2D hexagonal lattice (b).

for the mesh of Fig. 4.2a and site 0 the `neighbours` array is `[-1, 1, -1, 3, -1, 9]`. This helps to easily implement interactions in Fidimag's C functions.

For the calculation of the demagnetising field, Fidimag uses a fast-Fourier-transform method that requires a uniform grid, hence for complex geometries, such as curved samples, Fidimag still defines a cuboid mesh but boundaries or specific mesh sites without material are approximated by setting the magnetisation as zero, $M_s = 0$. Thus to obtain a better approximation it is necessary to define a large number of cuboids and decrease the distance between their centres. In order to set a null magnetisation, the user can specify a Python function that has as an input the position vector and that returns M_s according to the sample geometry [28].

In code snippet 4.2 we set a 2 nm diameter sphere using a $6 \times 6 \times 6$ cuboid mesh with elements of $1\,\text{nm} \times 1\,\text{nm} \times 1\,\text{nm}$ size. To simplify the dimensions scaling we set a unit length in the mesh definition.

```
import fidimag
import numpy as np

mesh = fidimag.common.CuboidMesh(nx=6, ny=6, nz=6,
                                 dx=1, dy=1, dz=1,
                                 unit_length=1e-9)
sim = fidimag.micro.Sim(mesh)
Ms = 1e6  # A / m

def sphere(pos, radius=2):
    """
    Returns `Ms` for a position `pos` inside the sphere
    of radius `radius` around the center of the mesh.

    """
    if np.sqrt(np.sum(pos ** 2)) < radius:
        return Ms
    else:
        return 0

sim.set_Ms(sphere)
```

LISTING 4.2: Definition of a 2 nm wide sphere geometry for a micromagnetic Fidimag simulation.

In the case of discrete spin simulations, atoms order in a lattice according to different crystallographic arrangements. The most simple ordering is the *simple cubic* lattice (SC), where atoms centres lie at the centre of cubes separated by the same distance in the three spatial dimensions. Since this follows the same principle as the finite differences cuboid mesh, Fidimag uses the same `CuboidMesh` class as the one for micromagnetic simulations, thus it is defined in Fidimag's `common` directory. As part of the work presented in this thesis, and in support of the work by Cortés-Ortuño et al. published in Ref. [187], we implemented a two dimensional hexagonal lattice that can be aligned along the x or y axis. A hexagonal lattice aligned with the x-axis is shown in Fig. 4.2b. The lattice is defined in the `HexagonalMesh` class in Fidimag's `atomistic` directory. As in the `CuboidMesh`, a `neighbours` array keeps track of the neighbours indexes, which are defined in the $x \rightarrow y$ order. In Fig. 4.2b we highlight, for example, the 6 neighbours of the lattice site with index 12.

Using as a base the class structure of the meshes currently implemented in Fidimag, it is possible to define and implement other crystal lattices such as body-centered-cubic or face-centered-cubic arrangements.

Besides scalars, most of the physical quantities involved in micromagnetic computations are fields and vector fields. In Fidimag, they are represented with numpy arrays [208]. There are utilities to instantiate them from point-wise expressions. These fields are the objects the physical interactions in the simulations deal with. A quantity central to the solution of the initial value problem given by the micromagnetic approach is the effective field, which is itself a sum of smaller fields.

Each of the smaller fields is the result of a separate computation which represents the influence of an interaction present in the system, like the exchange energy or the Dzyaloshinskii-Moriya interaction. For the dipolar field, we accelerate the calculation using a convolution done in Fourier space [209, 210, 211], accelerated using the library FFTW. It is in the computation of the interactions where the differences between the micromagnetic and the atomistic approach become the most apparent. Consequently, some energy terms are represented twice in the codebase - once for each model. With the effective field computed for a given state of the magnetisation the next step is solving the equation of motion for the system, which is the Landau-Lifshitz-Gilbert equation or one of its many variants. Fidimag supports the inclusion of stochastic terms and spin-polarised currents in the equation of motion. The terms and their corresponding equations are shown in Ref. [212].

For time integration, Fidimag can use the CVODE solver of the SUNDIALS suite [213] (wrapped via Cython) or the Fortran codes bundled in Scipy's [198] `integrate.ode`. Fidimag also implements Heun's method and the classic fourth-order Runge-Kutta method. Because the time integrators will need a varying amount of evaluations of the right hand side function, *i.e.* the equation of motion chosen in the last paragraph, it makes sense to partly relinquish runtime control from the user to the time integration. A so-called driver is in charge of coordinating time integration and user needs. Time integration runs uninterrupted for a specified amount of (simulated) time, after which arbitrary code can be executed.

Operating system

GNU/Linux, Mac OS X and on any platform supported by Docker, like Microsoft Azure and AWS.

Programming language

Python version 3, C, cython.

Dependencies

1. Numpy $\geq$ 1.10
2. SciPy $\geq$ 1.0.0
3. SUNDIALS $\geq$ 2.7.0
4. FFTW $\geq$ 3.3.4

Fidimag builds using the OpenMP versions of these libraries for multiprocessing purposes.

```
def test_save_scalar_field_hexagonal_mesh(tmpdir):
    mesh = HexagonalMesh(radius=1, nx=3, ny=3)
    s = scalar_field(mesh, lambda r: r[0] + r[1])
    vtk = VTK(mesh, directory=str(tmpdir), filename="scalar_hexagonal")
    vtk.save_scalar(s, name="s")
    assert same_as_ref(vtk.write_file(), REF_DIR)
```

LISTING 4.3: A unit test in which a scalar field `s` is created on a hexagonal mesh, saved to a VTK file and compared to a reference file.

Software location:

1. Name: Fidimag
2. Licence: BSD
3. Code Repository https://github.com/computationalmodelling/fidimag
4. Archive: https://zenodo.org/record/841113

4.3 Software Testing

Testing small chunks of code and preferably isolated pieces of the system is unit testing. In Fidimag, most of the unit tests compute and check the physical quantities involved in successfully running a simulation. Besides giving helpful feedback during development, the unit tests were a factor in increasing confidence to edit, enhance or refactor the code, resulting in the improved decoupling and maintainability of it. The coverage of Fidimag's code base with unit tests is monitored using codecov [214]. At the time of writing, it reports 73% coverage.

Functional testing then examines a slice of the system. For example, one of the test cases covering the saving of scalar fields on hexagonal meshes to a VTK file in Fidimag is shown in Listing 4.3.

Finally, system tests use Fidimag as a black box and compare simulation output to known good values. These values are obtained from problems that have analytical solutions like the macrospin or domain walls. As we saw earlier, the μmag standard problems are another good source of testing data. So are other finite difference codes, like the aforementioned OOMMF. In fact, because Fidimag and OOMMF both use the finite difference method, the tests used to evaluate Fidimag restrict the allowed difference in magnetisation vectors compared to OOMMF to much smaller values than the tests shown with FINMAG in Section 5.3: differences below 10^{-11} are commonplace. OOMMF is a good benchmark for accuracy for the testing of the common terms because of its wide circulation and long history of usage for many applications in micromagnetics.

Fidimag is also tested against the output of earlier versions of itself by storing some simulation results in the repository. This is a form of regression testing, and since no external software or potentially long-running simulations are involved the fastest way to check if a changeset has affected the computational parts in a tangible way.

The design choices discussed in the introduction and the focus on testing continue to provide tangible benefits to the collaborators working on the software, as well as the software itself, as

shown above. They also increase the confidence in Fidimag's results since unlike users, automated tests don't distinguish between new and old code and test potentially seldom-used parts of the software just as often as the commonly used functions.

For maximum value, the tests need to be run often and ideally without manual intervention. This is why a test cycle is triggered on the continuous integration platform Travis CI [206] on every push to the Github repository. First, Fidimag's C code and cython extensions are compiled. Then the tests are run. Finally the Jupyter notebooks are run and tested and the documentation is built.

To quickly check if Fidimag is installed and working on a machine, a user can launch an interactive Python shell and execute `import fidimag`. This command should not output any text. Afterwards, the user may chose to follow along with the tutorial called *A Basic Simulation* [215] or launch the provided examples. If Fidimag was installed from source, `make test-basic` will run a selection of tests that completes in under a minute, which is also helpful for smoke testing purposes during development on Fidimag.

The simulations are written in Python and use Fidimag as a library to be imported. Fidimag has been used to gather results for a number of scientific publications [185, 186, 187, 188] in the field of micromagnetics. It can be extended to account for other energy terms, to support more time integration methods or other variants of the equation of motion, as well as other drivers. Users can get in touch with the development team on Fidimag's issue page on GitHub [216].

4.4 Solution to Standard Problem #4

The Micromagnetic Modeling Activity Group μmag has defined a number of Standard Problems for the validation and comparison of micromagnetic simulation software [4]. The definitions of the problems and solutions from different research groups have been published on its website, and aim to show differences in approach between different computational micromagnetic software. To this end, we have compared Fidimag to the published results, and these problems are also given as examples in the Fidimag documentation.

For illustration, we show here the solution of Standard Problem #4. In this example, the magnetisation reversal dynamics of a bar of Permalloy with the dimensions $500 \times 125 \times 3\,\mathrm{nm}$ are computed. First, a relaxed 's-state' is obtained in the absence of an applied magnetic field. It is plotted in Figure 4.3. In a second simulation, a Zeeman field is applied that causes a reversal of the average magnetisation direction over the span of a few hundred picoseconds. Figure 4.4 shows the magnetisation configuration when the z-component of the spatially averaged magnetisation direction crosses 0. The system can be considered in equilibrium after a nanosecond and the corresponding magnetisation state is plotted in Figure 4.5. The code used is printed in Listings 4.4 and 4.5. To keep the listings reasonably short, comments were removed and the definitions of physical constants in Listing 4.4 were grouped in line 8. The plots were created with the code in the function `plot_quiver` defined in Listing 4.5.

```
1  import os
2  import matplotlib.pyplot as plt
3  import numpy as np
4  from fidimag.micro import Sim, UniformExchange, Demag, Zeeman
5  from fidimag.common import CuboidMesh
6  from fidimag.common.constant import mu_0
7
8  A = 1.3e-11; Ms = 8.0e5; alpha = 0.02; gamma = 2.211e5; mT = 0.001 / mu_0
9
10 mesh = CuboidMesh(nx=200, ny=50, nz=1, dx=2.5, dy=2.5, dz=3, unit_length=1e-9)
11
12 def compute_initial_magnetisation():
13     sim = Sim(mesh, name='problem4_init')
14     sim.driver.set_tols(rtol=1e-10, atol=1e-10)
15     sim.driver.alpha = 0.5
16     sim.driver.gamma = gamma
17     sim.Ms = Ms
18     sim.do_precession = False  # saves time - not interested in dynamics here
19     sim.set_m((1, 0.25, 0.1))
20     sim.add(UniformExchange(A))
21     sim.add(Demag())
22     sim.relax(dt=1e-13, stopping_dmdt=0.01, max_steps=5000,
23               save_m_steps=None, save_vtk_steps=None)
24     return sim.spin
25
26 def compute_dynamics(initial_magnetisation):
27     sim = Sim(mesh, name='problem4_dynamics')
28     sim.set_m(initial_magnetisation)
29     sim.driver.set_tols(rtol=1e-10, atol=1e-10)
30     sim.driver.alpha = alpha
31     sim.driver.gamma = gamma
32     sim.Ms = Ms
33     sim.add(UniformExchange(A))
34     sim.add(Demag())
35     sim.add(Zeeman([-24.6 * mT, 4.3 * mT, 0], name='H'), save_field=True)
36
37     crossed_zero = False
38     ts = np.linspace(0, 1e-9, 201)
39     for t in ts:
40         sim.driver.run_until(t)
41         mx, my, mz = sim.compute_average()
42
43         print("t = {:.3} ns\t mx = {:.3}".format(t*1e9, mx))
44         if mx <= 0 and not crossed_zero:
45             print("Crossed zero!")
46             np.save("problem4_m_when_mz_0.npy", sim.spin)
47             crossed_zero = True
48     return sim.spin
49
```

LISTING 4.4: Solution to the standard problem #4. The simulation defined in the function `compute_initial_magnetisation` returns the relaxed magnetisation configuration in absence of an applied magnetic field. In the function `compute_dynamics` that makes up the second simulation, the magnetic field that causes the magnetisation reversal is added on line 35.

```
50  def plot_quiver(m, filename):
51      m.shape = (50, 200, 3)
52      skip = 5
53      m = m[1::skip, 1::skip]
54
55      xyz = mesh.coordinates
56      xyz.shape = (50, 200, 3)
57      xyz = xyz[1::skip, 1::skip]
58
59      plt.figure(figsize=(12, 3))
60      plt.axes().set_aspect('equal')
61      plt.quiver(xyz[:,:,0], xyz[:,:,1],
62              m[:,:,0], m[:,:,1], m[:,:,1],
63              pivot='mid', scale=45, cmap=plt.get_cmap('jet'), edgecolors='None')
64      c = plt.colorbar()
65      plt.xlabel("x (nm)")
66      plt.ylabel("y (nm)")
67      c.set_label("$m_\mathrm{y}$ (1)")
68      plt.clim([0, 1])
69      plt.xlim([0, 500])
70      plt.ylim([0, 125])
71      plt.savefig(filename)
72
73  if __name__ == '__main__':
74      m0_file = "problem4_m0.npy"
75      if not os.path.exists(m0_file):
76          m0 = compute_initial_magnetisation()
77          np.save(m0_file, m0)
78      else:
79          m0 = np.load(m0_file)
80
81      mf_file = "problem4_mf.npy"
82      if not os.path.exists(mf_file):
83          mf = compute_dynamics(m0)
84          np.save(mf_file, mf)
85      else:
86          mf = np.load(mf_file)
87
88      plot_quiver(m0, "problem4_m0.pdf")
89      plot_quiver(np.load("problem4_m_when_mz_0.npy"), "problem4_m_when_mz_0.pdf")
90      plot_quiver(mf, "problem4_mf.pdf")
```

Listing 4.5: Solution to the standard problem #4 (cont.) The routines defined in Listing 4.4 are called to carry out the simulation. The snapshots in Figures 4.3 to 4.5 were made using the function `plot_quiver` defined on line 50.

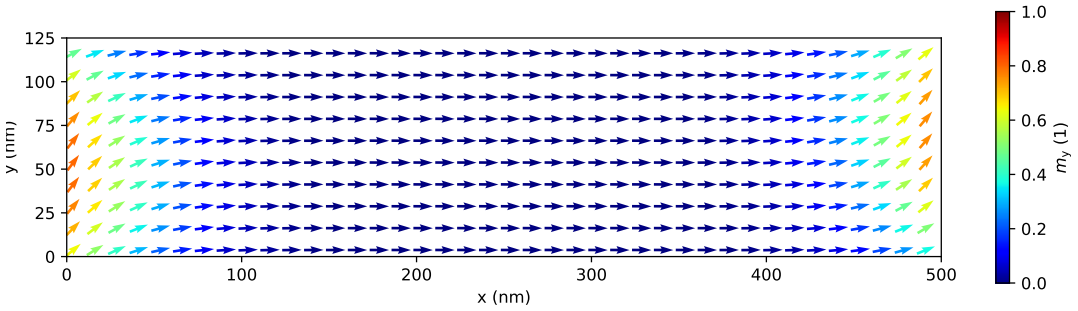


Figure 4.3: The initial unit magnetisation of the bar of Permalloy described in standard problem #4.

The evolution of the spatially averaged magnetisation is shown in Figure 4.6 with a comparison with the same simulation run with OOMMF. This comparison is performed regularly as part of Fidimag's automated testing.

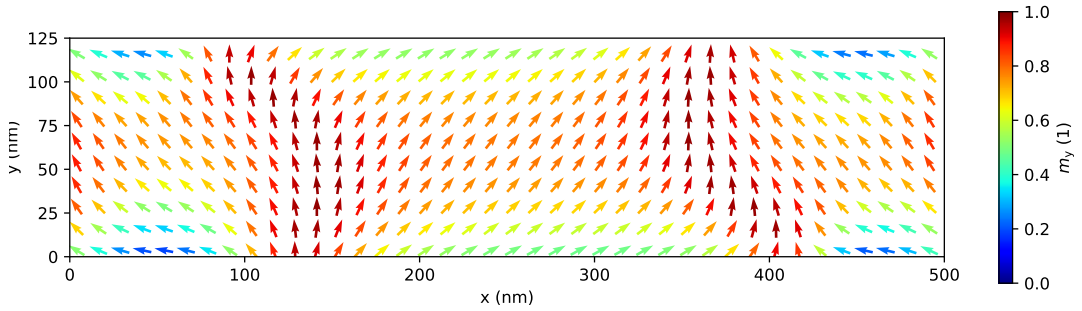


FIGURE 4.4: The magnetisation configuration at the moment of the reversal of the average magnetisation.

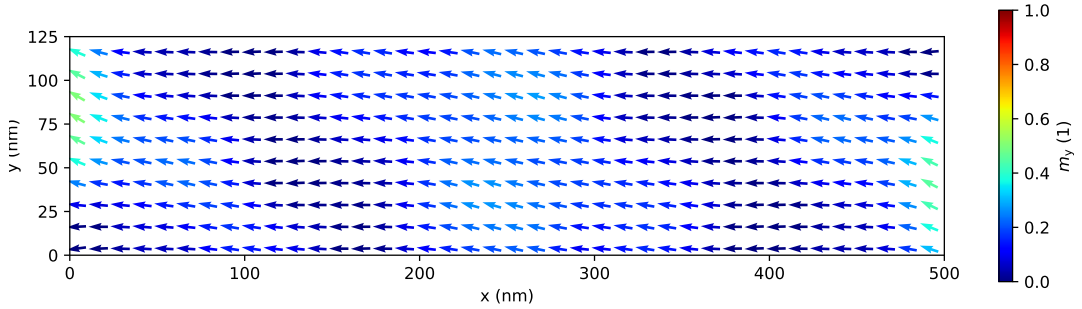


FIGURE 4.5: The magnetisation after a nanosecond of simulation time has elapsed.

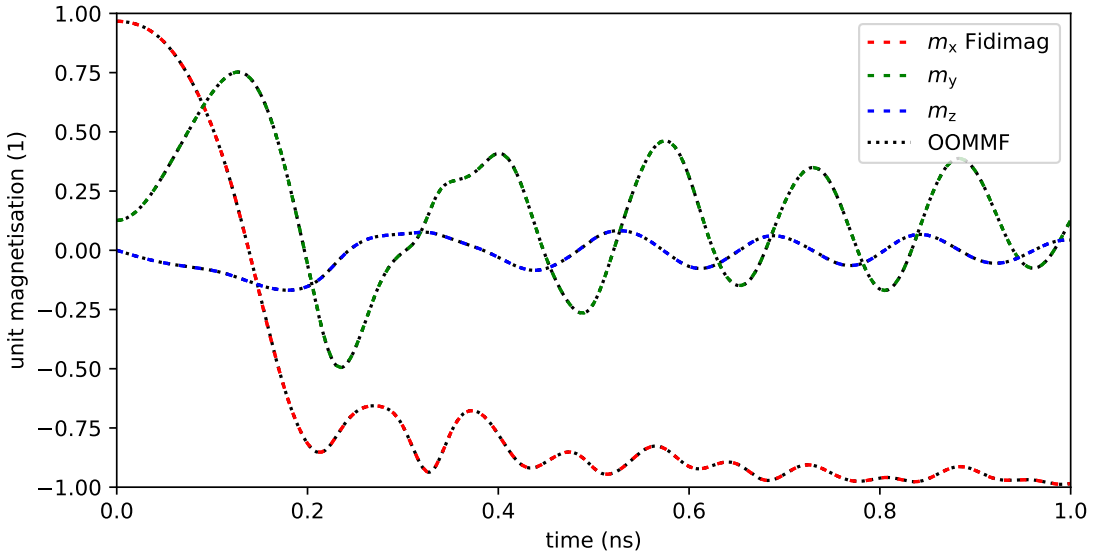


FIGURE 4.6: The components of the average magnetisation over time as computed with Fidimag and by OOMMF.

4.5 Time Integration of the LLG in Finite Difference Schemes

In Section 3.2, we saw that the LLG is an ordinary differential equation with $3n$ degrees of freedom, where n is the number of cells. As was reviewed in Section 3.5.4, standard methods for solving ordinary differential equations will integrate it. However, it is stiff enough to require very

short time steps when using explicit methods without stepsize control. If a longer time step is forced numerical instability ensues.

In Ref. [167], Suess et al. compare the performance of a backwards difference formula (BDF) with and without preconditioning, and an Adams formula in micromagnetic simulations as governed by the LLG equation. Both formulas are linear multistep methods of the family of implicit methods provided by the CVODE integrator from the SUNDIALS suite [213]. SUNDIALS is not a trivial piece of software to add on as a dependency, mainly because it needs the degrees of freedom data to be laid out in a certain way in memory. Suess et al. use magpar [29], a finite element code, to run the simulations involved in solving the standard problem #4 and record the CPU time. They conclude that the BDF speeds up the simulations compared to the Adams formula, and that the performance of the BDF can be further improved with preconditioning. Having integrated the SUNDIALS suite of integrators into our finite element code FinMag, discussed in the next chapter, testing confirmed that the preconditioned BDF is the best performing of the studied integrators in the finite element case.

In this section however, the study is extended to the finite differences case. Besides the integrators of the SUNDIALS suite, also integrated into Fidimag, we include explicit methods, like the Runge-Kutta method due to Dormand and Prince [217] in Scipy's `ode.integrate` Python module [198]. Using Fidimag, we will show that while the BDF is the most effective at reducing the number of evaluations of the LLG equation during the simulation, just as before, this comes at the cost of time spent solving the implicit problem. In the finite difference case this trade-off might not be optimal and an explanation for this will be given at the end of the section.

To be able to compare against a baseline, an implementation of the classic fourth-order Runge-Kutta method (RK4) is also included in the comparison. For the initial value problem $y'(t) = f\left(t, y(t)\right)$ with $y(t_0) = y_0$ and given function f, one integration step of size h in the Runge-Kutta method is given by

$$y_{n+1} = y_n + \frac{h}{6}\left(k_1 + 2k_2 + 2k_3 + k_4\right) \tag{4.4}$$

with the increments

$$\begin{aligned} k_1 &= f(t_n, y_n) \\ k_2 &= f(t_n + \frac{h}{2}, y_n + h\frac{k_1}{2}) \\ k_3 &= f(t_n + \frac{h}{2}, y_n + h\frac{k_2}{2}) \\ k_4 &= f(t_n + h, y_n + hk_3). \end{aligned}$$

The RK4 approximation of the next value y_{n+1} is determined by the value y_n and a weighted average of four increments, where the increments capture the slope at different points of the interval from t_n to $t_n + h$. In particular, k_2 and k_3 are based on the slope at the midpoint of the interval.

The increments can be represented in a visual aid called the Butcher tableau, which summarizes the coefficients. The simplest tableau corresponds to the forward Euler method, given by the formula

$$y_{n+1} = y_n + hf(t_n, y_n). \tag{4.5}$$

It is shown in Table 4.1.

0	
	1

TABLE 4.1: Butcher tableau of the forward Euler method.

In the Butcher tableaus there are as many rows above the horizontal line and as many columns to the right of the vertical line as there are increments. The numbers in the leftmost column yield the different time step sizes of the increments (found in the first argument of f) when multiplied by h. The numbers to the right of the vertical line in the n-th row of the tableau give the factors by which the other increments are multiplied in the second argument of f in the expression for the n-th increment. Explicit methods will exhibit a triangular form in the upper right quadrant as the n-th increment depends only on the previous increments. The numbers on the bottom correspond to the factors by which the increments are multiplied in the expression for y_{n+1}.

For the classic Runge-Kutta method, this process yields Table 4.2. The code implementing the classic Runge-Kutta method in Fidimag is shown in Listing 4.6.

0				
1/2	1/2			
1/2	0	1/2		
1	0	0	1	
	1/6	1/3	1/3	1/6

TABLE 4.2: Butcher tableau of the classic Runge-Kutta method. Using the third row of the tableau as an example, k_3 is obtained by evaluating f at $t_n + \frac{1}{2}h$ ($\frac{1}{2}$ in leftmost column), $y_n + \frac{1}{2}hk_2$ (0 indicating no contribution from k_1).

```
def runge_kutta_step(t, y, h, f):
    """
    Numerical integration using the classical Runge-Kutta method (RK4).
    Given the initial value problem y'(t) = f(t, y(t)), y(t_0) = y_0 one step
    of size h is y_{n+1} = y_n + h/6 * (k_1 + 2k_2 + 2k_3 + k4), where the
    weights are:
        k_1 = f(t_n,       y_n)
        k_2 = f(t_n + h/2, y_n + h/2 * k_1)
        k_3 = f(t_n + h/2, y_n + h/2 * k_2)
        k_4 = f(t_n + h,   y_n + h   * k_3).
    """
    k1 = f(t,           y)
    k2 = f(t + h / 2.0, y + h * k1 / 2.0)
    k3 = f(t + h / 2.0, y + h * k2 / 2.0)
    k4 = f(t + h,       y + h * k3)

    tp = t + h
    yp = y + h * (k1 + 2 * k2 + 2 * k3 + k4) / 6.0
    evals = 4
    return tp, yp, evals
```

LISTING 4.6: Python implemention of the classic Runge-Kutta method of fourth order in Fidimag.

The Runge-Kutta method due to Dormand and Prince uses six function evaluations per step, sharing the last evaluation of the function for one step with the first evaluation for the next. In

particular, it computes fourth- and fifth-order accurate solutions and then uses the difference between the two as error estimate for the fourth-order solution. This error estimate controls an adaptive stepsize. The Butcher tableau for the Runge-Kutta method due Dormand and Prince method is shown in Table 4.3.

0							
1/5	1/5						
3/10	3/40	9/40					
4/5	44/45	-56/15	32/9				
8/9	19372/6561	-25360/2187	64448/6561	-212/729			
1	9017/3168	-355/33	46732/5247	49/176	5103/18656		
1	35/384	0	500/1113	125/192	2187/6784	11/84	
	35/384	0	500/1113	125/192	2187/6784	11/84	0
	5179/57600	0	7571/16695	393/640	92097/339200	187/2100	1/40

TABLE 4.3: Butcher tableau of the Runge-Kutta method due to Dormand and Prince. The first row below the line yields the fifth-order solution. The error estimate is computed by subtracting an alternative solution given by the second row below the line.

The complete list of numerical integrators used in this study is given in Table 4.4. The dopri5 is the Dormand and Price integrator just reviewed. A higher-order variant of it is the dop853 with order 8. The lsoda integrator automatically switches between the Adams method and the BDF. One of the two formulas can be selected by the user in the vode integrator, and both variants are used in this study. SUNDIAL's cvode stands out as the only C code in the list, as the Scipy codes are written in Fortran 77.

Integrators with variable step sizes use the parameters `RTOL` and `ATOL` to determine the error control performed by the solver. The vector of estimated local errors

$$e = e_i \tag{4.6}$$

of the dependent variable y of length n is controlled by the solver according to an inequality of the form

$$\sqrt{\frac{1}{n}\sum\left(\frac{e_i}{\mathrm{EWT}_i}\right)^2} \leq 1 \tag{4.7}$$

where

Integrator	Package	Note
rk4	Fidimag	classic Runge-Kutta (RK) method
dopri5	Scipy	Dormand, Prince [217], 1993
dop853	Scipy	idem
lsoda	Scipy (ODEPACK)	Hindmarsh [218], 1982
vode/adams	Scipy (ODE)	Brown et al. [219], 1989
vode/bdf	Scipy (ODE)	idem
cvode	SUNDIALS	Hindmarsh et al. [213], 2005

TABLE 4.4: Complete list of numerical integrators used in study.

h

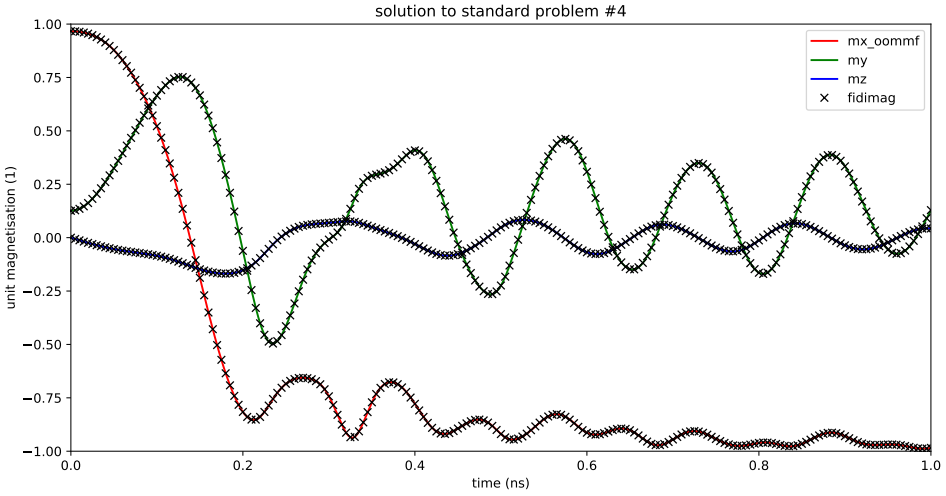


FIGURE 4.7: Comparison of the average magnetisation as computed by OOMMF and Fidimag for standard problem #4.

$$\mathrm{EWT}_i = |y_i| \cdot \texttt{RTOL} + \texttt{ATOL} \tag{4.8}$$

is a vector of weights that must always be positive. `RTOL` is the relative tolerance parameter and `ATOL` the absolute tolerance parameter. They should be non-negative. Setting `RTOL` to 0 results in pure absolute error control and vice versa.

The standard problem #4 is presented in Section 4.4. For every run of the simulation, the average magnetisation dynamics and the reversal time are automatically compared to a control simulation previously obtained with OOMMF. Figure 4.7 shows the time evolution of the average magnetisation as computed with Fidimag in comparison with OOMMF. The simulations presented in this section are controlled to within 0.1 % accuracy of the controls. The machine used to carry out the simulations had an Intel Xeon E3-1270 processor with 4 physical cores and a frequency of 3.4 GHz. It was equipped with 16 GB of RAM. The operating system in use was Ubuntu 14.04 LTS. The simulation was repeated 10 times for every integrator and an average of the walltime was taken.

Figure 4.8 and Figure 4.9 show how many times the LLG equation was evaluated during the simulations performed with the different time integrators. For $\texttt{RTOL} = \texttt{ATOL} = 10^{-8}$ and $\texttt{RTOL} = \texttt{ATOL} = 10^{-10}$, CVODE's BDF effectively reduces the number of evaluations. So do the other integrators.

For the classical Runge-Kutta method, the chosen time step of $\Delta t = 10^{-14}$s was the largest step still resulting in stable simulations. Out of the studied integrators, the Runge-Kutta method due to Dormand and Prince results in the fastest simulations. For $\texttt{RTOL} = \texttt{ATOL} = 10^{-8}$, it is an implementation of the method with order 5 called dopri5 as can be seen on Figure 4.10. Figure 4.11 shows that for smaller tolerances it can be beneficial to increase the order further. For $\texttt{RTOL} = \texttt{ATOL} = 10^{-10}$, the fastest method is called dop853 and has order 8.

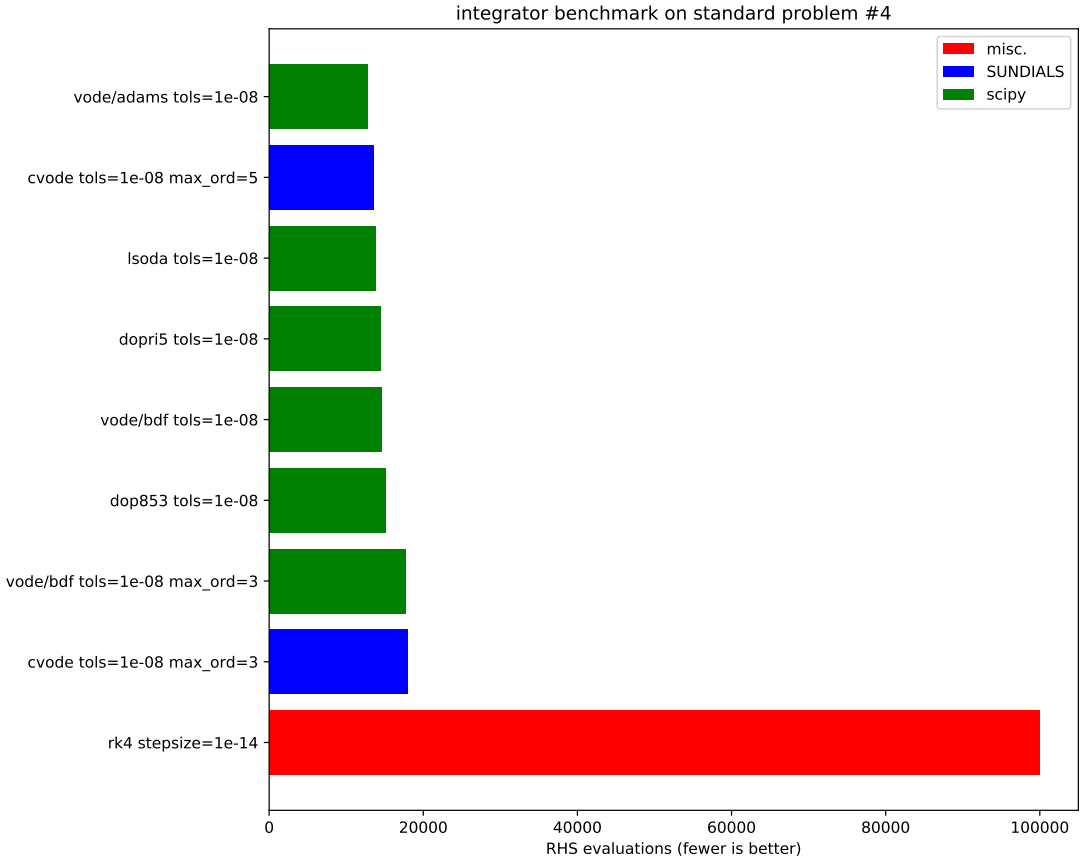


FIGURE 4.8: Number of evaluations of the LLG equation during the simulation of dynamical part of standard problem #4 for `RTOL` = `ATOL` = 10^{-8}. `max_ord` controls the maximum order of the BDF.

The number of evaluations of the right-hand side (RHS) of the LLG equation for the different numerical integrators shown in Figures 4.8 and 4.9 can be contrasted with the total runtime of the simulations shown in Figures 4.10 and 4.11. We see that for both set of tolerances the method dop853 resulted in shorter simulations than CVODE despite the fact that when setting a maximum order of 5, the use of CVODE resulted in fewer RHS evaluations. This effect is also pronounced when comparing the dop853 method to the Scipy ODE integrators vode/adams and vode/bdf. We conclude that for the finite difference case, the reduction in RHS evaluations does not compensate sufficiently for the time cost of solving the implicit problem.

The explanation as to why the findings of Suess et al. do not apply in the finite difference case follows. The meshes used in the finite element method are non-regular. As such, they will almost certainly include some cells that are smaller than others. In smaller cells, the strength of the exchange field increases due to the $\frac{1}{V_i}$ factor in Equation (3.55), where V_i is the volume of the cell. The exchange field strength increases relative to the demagnetising field, leading to a more marked stiffness of the LLG equation. Accordingly, implicit methods like the BDF show more of their advantage in the finite element case.

As a consequence of this study the default integrator choice for Fidimag is the Dormand & Prince scheme dop853 with the relative and absolute tolerances set to 10^{-8}. It offers the best compromise of speed and accuracy.

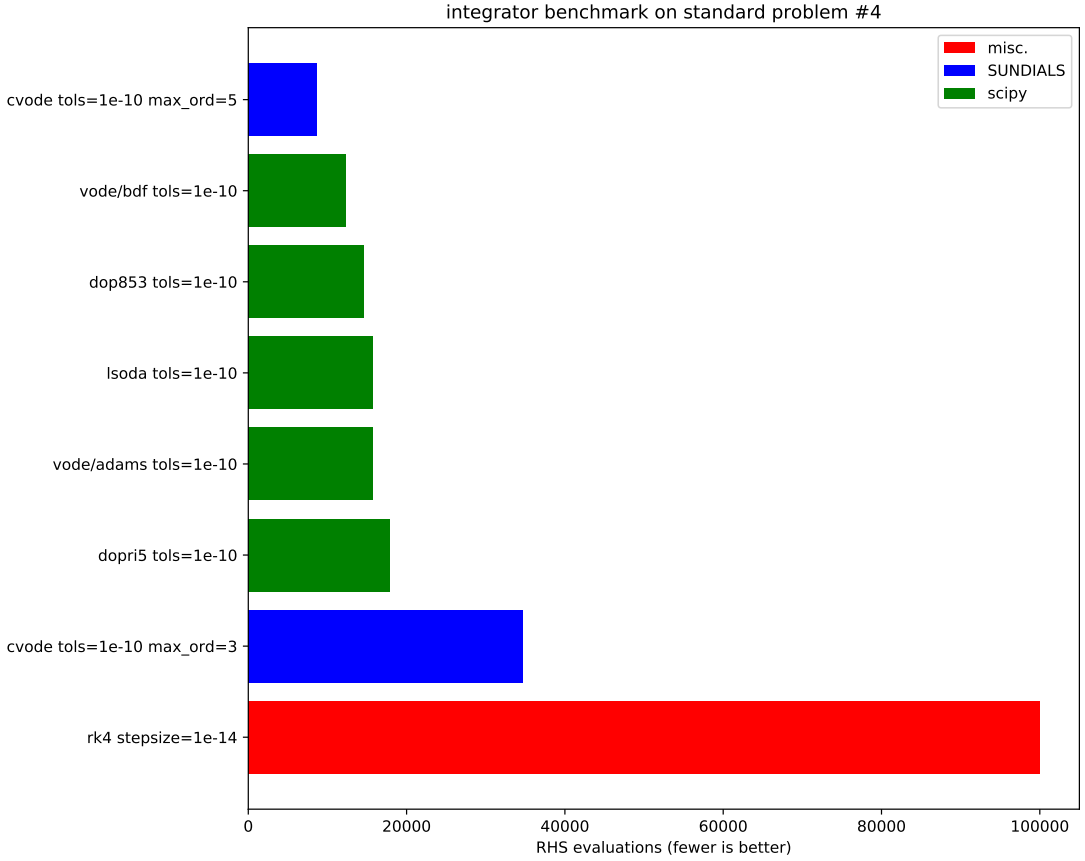


FIGURE 4.9: Number of evaluations of the LLG equation during the simulation of dynamical part of standard problem #4 for `RTOL` = `ATOL` = 10^{-10}.

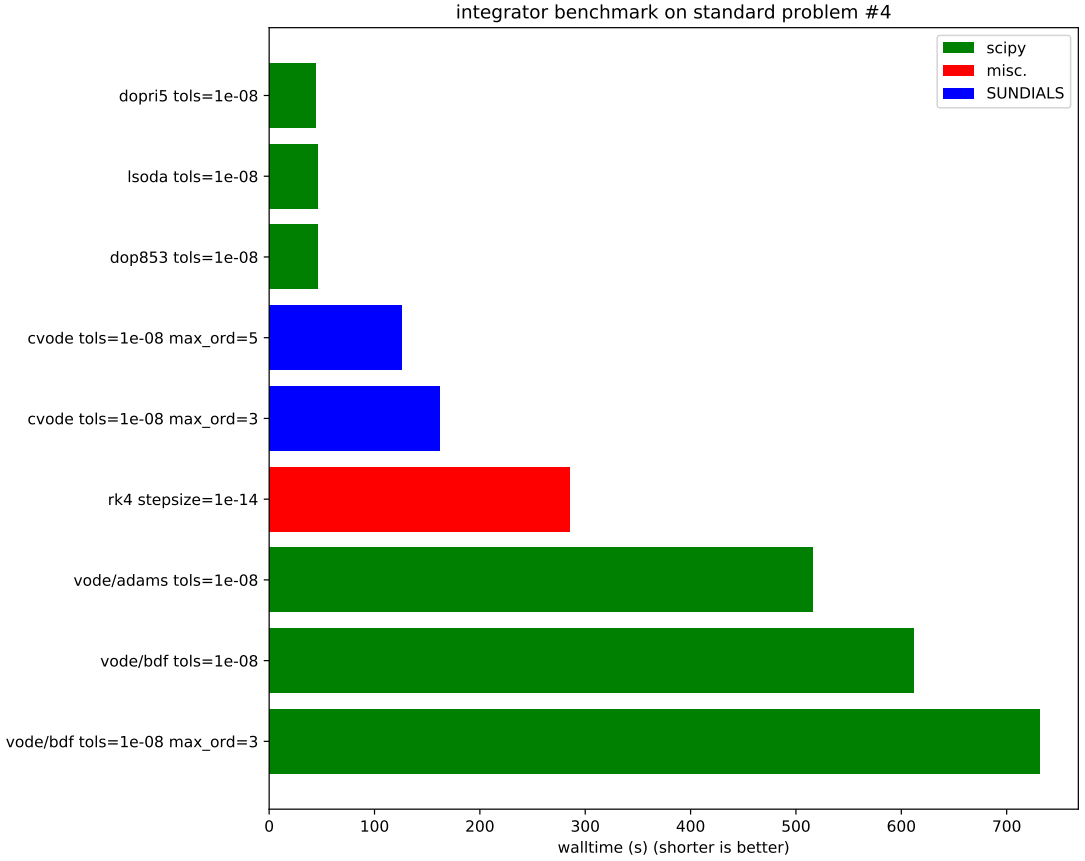


FIGURE 4.10: Walltime elapsed during the simulations for `RTOL` = `ATOL` = 10^{-8} for each of the integrators.

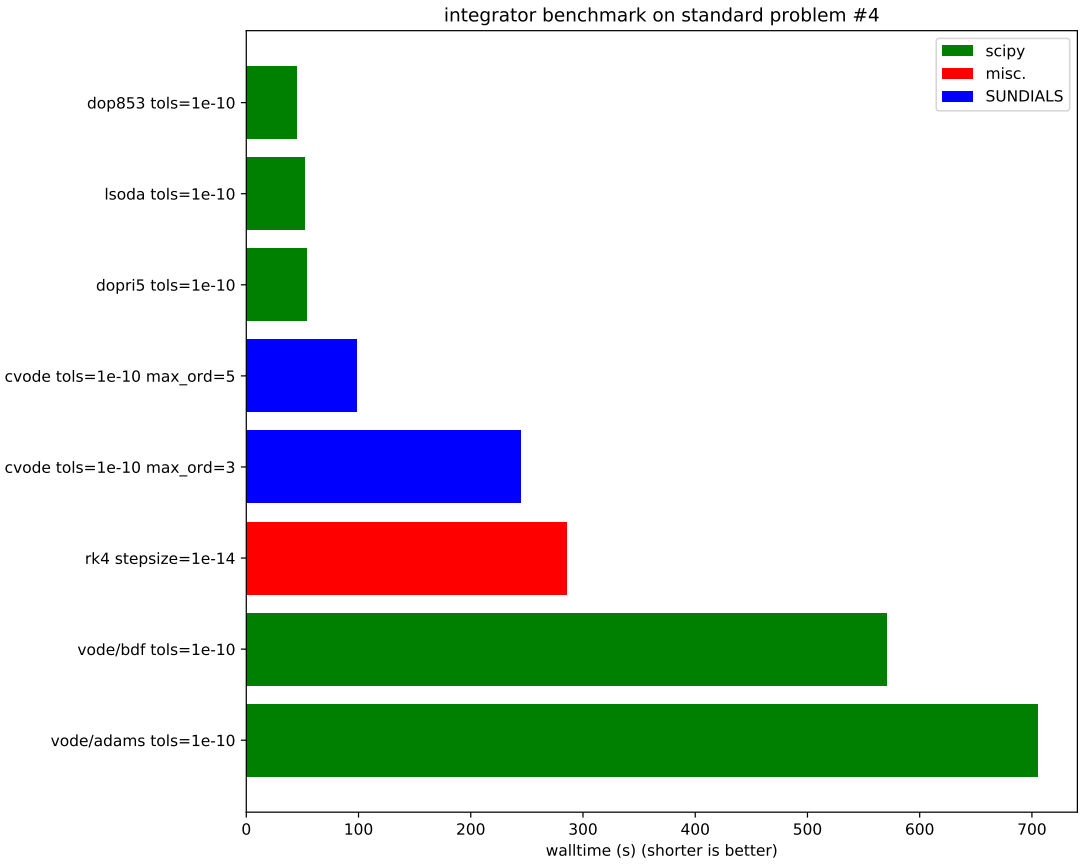


FIGURE 4.11: Walltime elapsed during the simulations for `RTOL` = `ATOL` = 10^{-10} for each of the integrators.

Chapter 5

FinMag – a finite element software for micromagnetic simulations

A micromagnetic simulation framework called FinMag was developed to not only fulfill the specific needs of the simulations carried out in the context of this project, but enable micromagnetic simulations for a wide range of purposes.

5.1 Introduction

A recent study on successful scientific software investigated the design goals of different projects and determined the following ranking of goals [220]:

1. Correctness
2. Performance
3. Portability
4. Maintainability

The appearance of portability and maintainability in the list of goals hints at a recent shift of paradigms in development. Unlike performance and correctness, both are not obvious design goals for scientific sofware and especially performance usually comes at the expense of maintainability. On the other hand the latter helps ensure the correctness of a program.

Currently, FinMag contains about 40 000 lines of code written in the Python programming language and about 7100 lines of code written in C++. While these counts do not contain blank lines and comments, but do contain simulation, example and test files, they nonetheless reflect how Python and C++ are used in this project. Python fosters a scientific community around it since many years — at the SciPy 2018 it will come together for its 17th annual conference — and

is conducive to rapid application development due to its high level nature. The programming interface and most of the internal code is written in Python. Indeed, simulations are described in that language. C++, being a faster compiled language instead of a dynamic language, is used for performance reasons where intensive computations are to be performed.

FinMag leverages the power of several open-source projects. Besides the use of the numpy extension to Python for the support of large, multi-dimensional arrays or part of the Boost libraries to interface Python and C++, FinMag most notably builds on the FEniCS project [221, 222]. FEniCS provides software tools for the efficient solution of differential equations using the finite element method. Instead of using FEniCS to pose the solution of the LLG equation as a variational problem, FinMag evaluates the LLG equation pointwise. A precise formulation of micromagnetics as a variational problem is a very recent addition to the literature "although it has long existed in the micromagnetics folklore" [223].

The finite difference method has the advantage of easier implementation and high computing power, but restricts the simulated geometries to those that can be described with cuboid cells. This leads to the creation of artifacts stemming from the discretisation when modelling non cuboid structures. The simplest shapes of simulation cells in the finite element method are tetrahedra, which makes it possible to approximate curved shapes. Another advantage is the variability of cell size. Regions of special interest can be partitioned using smaller cells and thereby be resolved with higher precision. With tetrahedra of varying sizes, even the most complex geometries can be described. The flexibility of the finite element method comes at the expense of high computational complexity [224]. The use of FEniCS harnesses the advantages of the finite element method and still allows more development time to be be spent on implementing the physical properties of the studied ferromagnetic systems.

FinMag computes the right-hand side of the LLG equation as shown in Equation (3.11) by default. It can also augment the LLG by spin-torque or stochastic terms. For time integration it can use an Adam-Moulton formula or a backward differentiation formula in so-called fixed-leading coefficient form called VODE [225]. Both are variable-order, variable-step multistep methods for solving ordinary differential equations and provided by the SUNDIALS library [213]. A Heun method is used in the stochastic case.

FinMag implements constant and time-dependent Zeeman fields, the exchange interaction and the Dzyaloshinskii-Moriya interaction. It offers uniaxial, cubic, easy-surface and easy-normal anisotropy terms. Two options are implemented for the computation of the demagnetising field, both based on a hybrid finite element method / boundary element method (FEM/BEM). One was introduced by Fredkin and Koehler [37] and is widely used in state-of-the-art micromagnetic finite element packages [83], the other was proposed later by Garcia-Cervera and Roma and allegedly has better numerical properties [226]. A review of FEM/BEM methods is given in [162]. It is possible for the user to implement new energy terms on his own and have them included in FinMag simulations. This can be achieved without knowing about FinMag's internals but by programming against a small interface.

The user can schedule pre-defined or custom events to be triggered during the simulation. For instance, FinMag can save its simulation data or export it to vtk.

5.2 Implementation of the Dzyaloshinskii-Moriya Energy

In Section 3.3 the definition of the Dzyaloshinskii-Moriya interaction was reviewed. As an example of how a new energy term might be added to FinMag, an implementation of the DMI is shown in Listing 5.1. The implementation defines a class named DMI which inherits from FinMag's EnergyBase class. It also implements the computation of the Dzyaloshinskii-Moriya energy density. FinMag will use this to compute the field contribution to the effective field and the total energy of the DMI. Behind the scenes, FinMag will for example also include the DMI in logging and auto-saving of simulation results to disk.

```
import dolfin as df
from aeon import mtimed
from finmag.energies.energy_base import EnergyBase
from finmag.util.helpers import times_curl

class DMI(EnergyBase):
    def __init__(self, D, method="box-matrix-petsc"):
        self.D = D
        super(DMI, self).__init__(method, in_jacobian=True)

    @mtimed
    def setup(self, S3, m, Ms, unit_length=1):
        # from implementation of finite element method:
        # S3 is a vector function space obtained from a given mesh
        # and a choice of basis
        self.DMI_factor = self.D * df.Constant(1.0 / unit_length)
        E_integrand = self.DMI_factor * times_curl(m, dim)
        # E_integrand
        super(DMI, self).setup(E_integrand, S3, m, Ms, unit_length)
```

Listing 5.1: Implementation of the Dzyaloshinskii-Moriya interaction. FinMag provides several numerical methods to compute the field from the definition of the energy density, one of which was chosen as default in the class's initialising method.

Listing 5.2 shows how the DMI would be added to as simulation as an object at runtime. Effectively making the energy terms in FinMag equivalent to plugins. Hence, a user of a packaged version of FinMag could extend it in this way without needing to obtain FinMag's sources or compile the parts of FinMag written in C++.

```
from finmag import Simulation
from finmag.energies import DMI
from finmag.util.meshes import box

Ms = 3.84e5  # A/m
A = 8.78e-12  # J/m

mesh = box(0, 0, 0, 20, 20, 1, maxh=2.5)
sim = Simulation(mesh, Ms, unit_length=1e-9)
sim.add(DMI(A))
```

Listing 5.2: Creation of a simulation and addition of the DMI to it.

FinMag was the first micromagnetic code in which the Dzyaloshinskii-Moriya energy was implemented. In fact, it motivated keeping FinMag closed source until the publication of Ref. [6].

5.3 Assessing FinMag's Reliability

FinMag is developed using agile development methods. The architecture is not developed upfront, but during the growth of the project. Borrowing some of the ideas of test-driven development [227], automated tests are written. At the time of writing, there are nearly 500 tests run after every commit to the version control repository. Additionally, some long-running simulations are run and verified in regular intervals. Refactoring[1] improves the codebase in terms of extensibility and maintainability [228] and the automated tests help detect if errors were introduced.

Additionally, a set of simulations that compute the effective field contributions for different configurations to compare them against other simulators exists. The same simulations have been written to run with Magpar [29], OOMMF [23], and Nmag [28] and the results are compared against each, or against analytical solutions where this is possible. The comparison with the other micromagnetic packages will be presented in this section.

Because Magpar and Nmag both use the finite element method, results defined on the same mesh can be compared easily to FinMag. Oommf uses the finite difference method and uses meshes built of cubic cells. To check a field computed with FinMag against its OOMMF equivalent, the former is probed at the locations of the vertices of the corresponding OOMMF mesh. When fields are compared beetween FinMag and Nmag, special care has to be taken: Instead of the field, the cross product of the magnetisation and the field $\boldsymbol{m} \times \boldsymbol{H}$ has to be evaluated and used for the comparison. This is because Nmag encapsulates it's data differently, using the fact that m and H don't enter the LLG equation on their own, but as a cross product.

The relative difference between FinMag's results and another package is computed with

$$\Delta = \frac{\vec{r}_{\text{finmag}} - \vec{r}_{\text{ref}}}{\max(\|\vec{r}_{\text{ref}}\|)}. \tag{5.1}$$

The absolute difference is divided by the maximum value of the euclidean norm of the vectors in the reference simulation to filter out big relative errors due to comparing values which are supposed to be zero. In the Tables 5.1 to 5.4, the maximum relative difference is denoted by $\Delta_{\max}$ and the mean is $\bar{\Delta}$. The standard deviation is the square root of the average of the squared deviations from the mean and denoted by σ. Usually, $\Delta_{\max}$ is rounded up to one decimal place and used as a safeguard against programming mistakes, called Δ_{test}. By comparing the hard-coded Δ_{test} to the computed $\Delta_{\max}$ and displaying an error message if $\Delta_{\max} > \Delta_{\text{test}}$, regressions can be identified.

It could be argued, that from an experimental point of view, testing could use the mean difference $\bar{\Delta}$ and the standard deviation σ instead of the maximum relative difference $\Delta_{\max}$. This would more closely resemble an averaged magnetisation such as one captured by a sensor. In our view however, it is best to approach the tests as highlighting numerical aberrations by default, even if they occur on a small region of the mesh. Non-physical singularities like a vortex core in an in-plane sample could still be rationalized post-facto.

[1] rewriting parts of the code without adding functionality

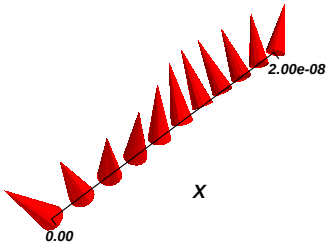


FIGURE 5.1: Starting magnetisation $\boldsymbol{m}$ for the exchange field comparison.

	Δ_{test}	$\Delta_{\max}$	$\bar{\Delta}$	σ
Nmag	2×10^{-14}	1.35×10^{-14}	3.37×10^{-15}	3.91×10^{-15}
OOMMF	8×10^{-2}	7.53×10^{-2}	2.34×10^{-2}	2.37×10^{-2}

TABLE 5.1: Comparison of the exchange field computed with FINMAG against Nmag and OOMMF.

Exchange For the comparison of the exchange field, it is computed on a one-dimensional mesh of length $L = 20\,\text{nm}$ with a starting magnetisation as shown in Figure 5.1 and described by

$$\begin{aligned} m_x &= (2x' - 1) \cdot \frac{2}{3} \\ m_y &= \sqrt{1 - m_x^2 - m_z^2} \\ m_z &= \sin(2\pi x') \cdot \frac{1}{2} \end{aligned}$$

where $x' = x/L$ and $x \in [0; L]$. The value of the exchange coupling constant is $1.3 \times 10^{-11}\,\text{J}\,\text{m}^{-1}$.

The values for the relative difference of the compute exchange field are listed in Table 5.1. While the agreement with Nmag is very good, there is a relative difference of around 8 % with OOMMF. In the limit of infinitely small meshes, we expect the finite element and the finite difference method to give approximately the same results. Indeed, as will be seen in the next section, FINMAG's and OOMMF's solution converge for progressively smaller meshes. Because this problem is defined on a one-dimensional mesh and this is not supported by Magpar, no comparison was possible for this particular test.

Uniaxial Anisotropy A cube with the edge length $L = 20\,\text{nm}$ is modelled. The initial magnetisation used for the computation of the anisotropy field is shown in Figure 5.2 and defined by

$$\begin{aligned} m_x &= (2 - y') \cdot (2x' - 1) \cdot \frac{1}{4} \\ m_y &= \sqrt{1 - m_x^2 - m_z^2} \\ m_x &= (2 - y') \cdot (2z' - 1) \cdot \frac{1}{4} \end{aligned}$$

where $x' = x/L$, $y' = y/L$, $z' = z/L$, and $x, y, z \in [0; L]$. The easy axis is in the $(1, 0, 0)$ direction with a parameter $K1 = 5.2 \times 10^5\,\text{J}\,\text{m}^{-3}$. The values for the relative difference are listed in Table 5.2.

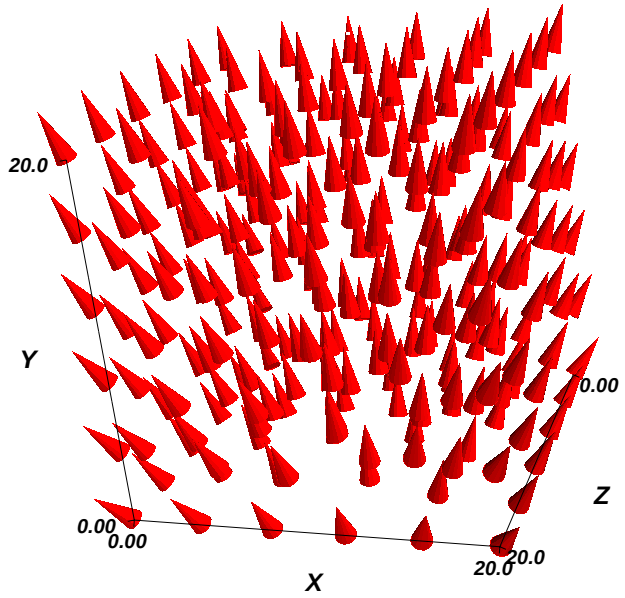


FIGURE 5.2: Starting magnetisation $\boldsymbol{m}$ for the anisotropy field comparison.

	Δ_{test}	$\Delta_{\max}$	$\bar{\Delta}$	σ
Nmag	6×10^{-2}	5.95×10^{-2}	3.62×10^{-3}	7.94×10^{-3}
Magpar	5×10^{-7}	4.10×10^{-7}	9.42×10^{-8}	9.77×10^{-8}
OOMMF	7×10^{-2}	6.97×10^{-2}	2.31×10^{-3}	6.70×10^{-3}

TABLE 5.2: Comparison of the anisotropy field computed with FINMAG against Nmag, Magpar and OOMMF.

The convergence of OOMMF's and FINMAG's results was investigated. A cube with the edge length $L = 100\,\text{nm}$ was modelled with increasingly finer meshes. The relative difference between the two results, when plotted with a double-logarithmic scale against the number of vertices as in Figure 5.3, shows a linear decrease with finer meshes. The maximum does not decrease at the same rate as the mean because in a finite difference code like OOMMF, m is constant inside every cuboid, whereas the magnetisation can change linearly over the tetrahedral cell in FINMAG. The anisotropy parameter was $K1 = 4.5 \times 10^5\,\text{J}\,\text{m}^{-3}$, the initial magnetisation was $\boldsymbol{m} = (x/L, \sqrt{1 - m_x^2}, 0)$ and the easy axis was in the $(1, 1, 1)$ direction.

Demagnetising field In the automated tests, the demagnetising field is computed for a uniformly magnetised sphere with radius 10 nm as shown in Figure 5.4. The starting magnetisation is $\boldsymbol{m} = (1, 0, 0)$ everywhere in the sphere. The exact result for this simple system is known analytically: The demagnetising field vector is equal to minus one-third of the magnetisation vector. Table 5.3 shows the comparison of the demagnetizing fields computed with FINMAG, Nmag, and Magpar against the analytical solution. The table also shows a comparison between FINMAG and Nmag and Magpar, to obtain values of Δ_{test} to be used for automatic testing. It is worth noting that the Nmag and FINMAG solution are close to each other, with Magpar further away both from them and the analytical solution.

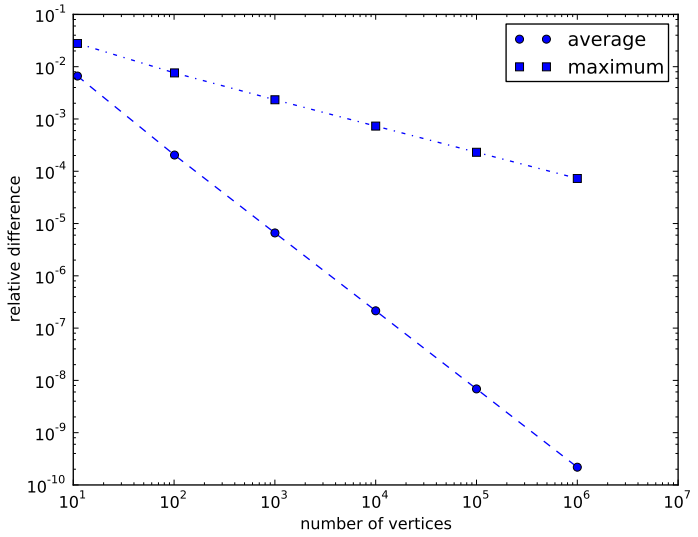


FIGURE 5.3: Convergence of FINMAG's and OOMMF's solution for fine meshes.

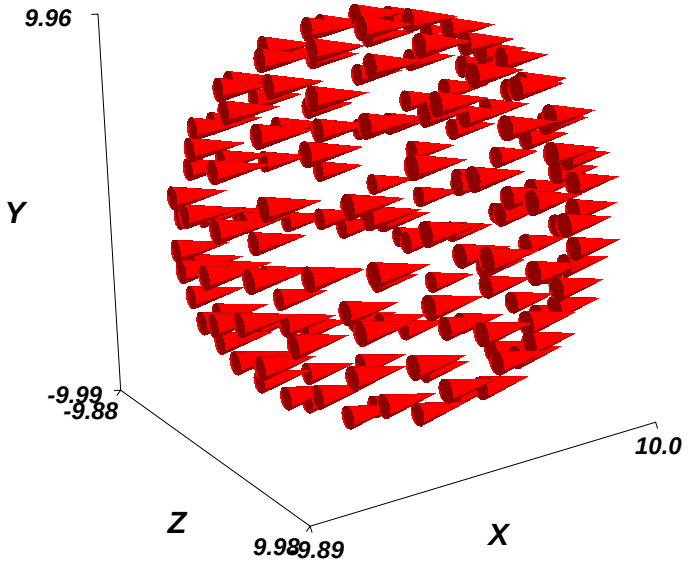


FIGURE 5.4: Starting magnetisation $\boldsymbol{m}$ for the demagnetising field comparison.

	Δ_{test}	$\Delta_{\max}$	$\bar{\Delta}$	σ
FINMAG	2×10^{-2}	1.26×10^{-2}	2.33×10^{-3}	2.49×10^{-3}
Nmag		1.26×10^{-2}	2.33×10^{-3}	2.49×10^{-3}
Magpar		2.14×10^{-1}	6.99×10^{-2}	9.80×10^{-2}
FINMAG $\leftrightarrow$ Nmag	5×10^{-5}	3.84×10^{-5}	7.65×10^{-6}	5.85×10^{-6}
FINMAG $\leftrightarrow$ Magpar	0.3	2.57×10^{-1}	8.51×10^{-2}	1.20×10^{-1}

TABLE 5.3: Comparison of the demagnetising field computed with FINMAG, Nmag, and Magpar against the analytical solution followed by a comparison between FINMAG and Nmag and Magpar.

Solution of the LLG equation A comparison of the solution of the LLG equation for a system with an analytical solution is shown in Section 5.3. For reasons of completeness, Table 5.4 shows the comparison with OOMMF. The modelled system consists of a rectangular hexahedron with the dimensions $20 \times 10 \times 1\,\text{nm}$, a homogeneous effective field $\mathbf{H}_{\text{eff}} = \frac{M_S}{2\sqrt{3}}(1, 1, 1)$ and the

	Δ_{test}	$\Delta_{\max}$	$\bar{\Delta}$	σ
OOMMF	3×10^{-16}	2.28×10^{-16}	1.12×10^{-16}	3.86×10^{-17}

TABLE 5.4: Comparison of the solution of the LLG equation of FINMAG with OOMMF.

initial magnetisation $\boldsymbol{m} = \frac{1}{\sqrt{14}}(-3, -2, 1)$.

Only the nodal values of $\mathbf{H}_{\text{eff}}$ and $\mathbf{M}$ are considered when solving the LLG equation, which for all practical purposes places the computation outside of the framework of the finite element or finite difference method. Because of this, the agreement between FINMAG and OOMMF is excellent.

Macrospin Example In the absence of demagnetisation, anisotropy and exchange fields the magnetisation behaves like a macrospin, i.e. the magnetisation is uniform and thus does not depend on the position: $\mathbf{M}(\vec{r}, t) = \mathbf{M}(t)$. By applying an external field, the motion of this macrospin in a static field can be computed and compared with an analytical solution to the LLG for this system, given in [111].

Starting with the magnetisation aligned with the x-direction, and an applied field acting in the z-direction, the time developments for the magnetisation for different values of the damping constant α are shown in Figure 5.5. The agreement with the analytical solution is very good. For the values of the damping parameter $\alpha = 0.02$, 0.10, 0.50, and 1.00 the highest value of the maximum relative difference (see Equation (5.1)) is $\Delta_{\max} = 9.92 \times 10^{-10}$.

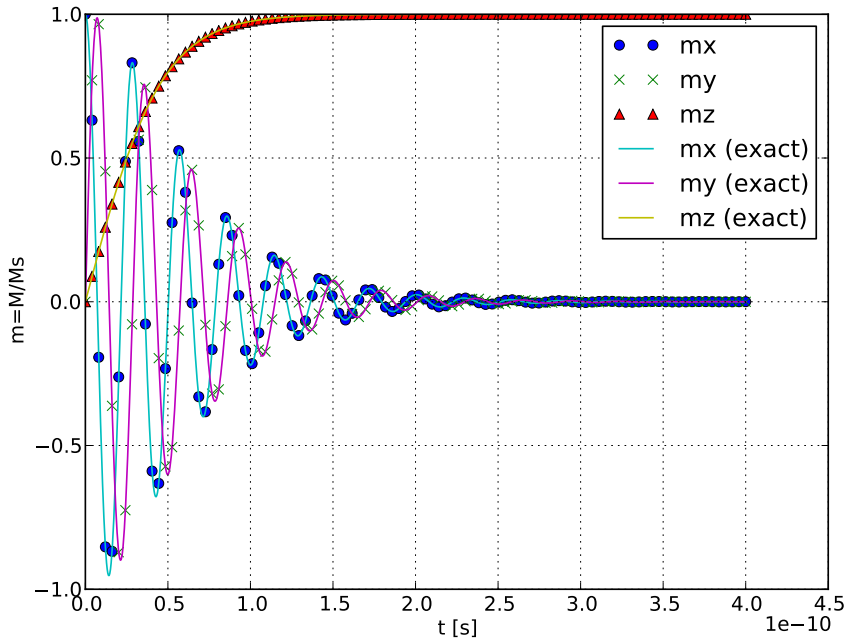
Example with Exchange and Demagnetisation In this example, the time development of a system in which the magnetisation initially points 45° away from the x-axis in the direction of the z-axis, that is the (1,0,1)-direction, is computed. Both the exchange and demagnetisation fields are considered and the results are compared with Nmag. The simulation is run for a total of 0.3 ns, with a data saving time step $\Delta t = 5 \times 10^{-12}$ s or 60 steps.

The geometry used is a bar of dimensions $30 \times 30 \times 100$ nm. The saturation magnetisation is $0.86 \times 10^6\,\mathrm{A\,m^{-1}}$ and the exchange coupling strength is $13.0 \times 10^{-12}\,\mathrm{J\,m^{-1}}$. The damping parameter is 0.5.

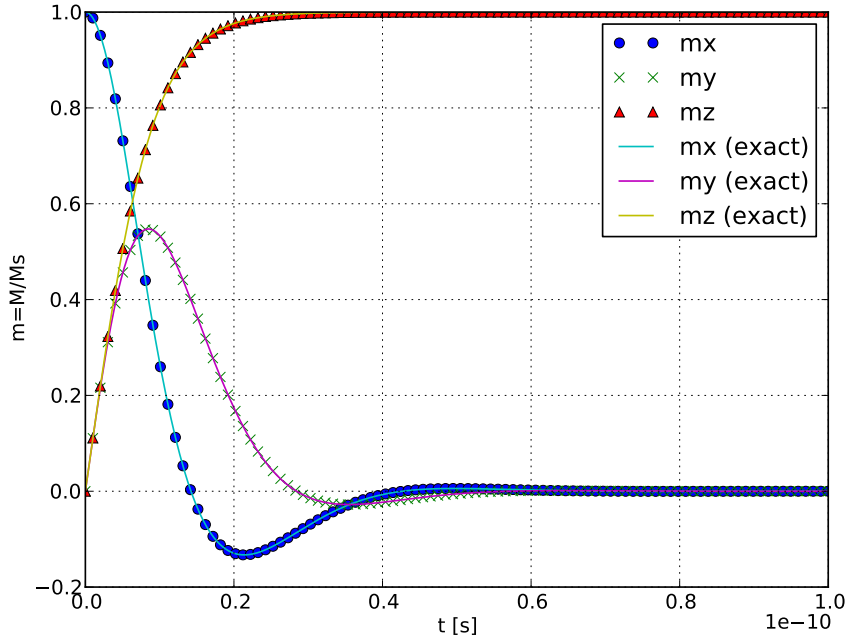
The relative difference of the normalised magnetisation to the Nmag implementation is below 1×10^{-4}. Figure 5.6 shows the comparison between the FINMAG and Nmag results for the average magnetisation over time. The relative difference of the exchange and demagnetisation energy density is below 1×10^{-4} and 1×10^{-3} respectively.

After ten time steps, the energy density through the longest axis of the bar is computed, as can be seen in Figures 5.7 and 5.8. Comparing the exchange energy density to both Nmag and OOMMf, shows that FINMAG's solution is closer to OOMMF than to Nmag, because the Nmag curve is noisier. Because of this, the relative error is a bit higher in this case, with $\Delta_{\max} = 2.8 \times 10^{-2}$. The results are closer for the demag energy density, with $\Delta_{\max} = 6 \times 10^{-3}$.

FINMAG needs 2.159 s to create all needed objects, while the simulation itself runs for 20.694 s. Compared to Nmag's 8.217 s and 23.804 s, the setup is roughly sped-up by a factor of 4.



(A) $\alpha = 0.10$



(B) $\alpha = 1.00$

FIGURE 5.5: Time development of the macrospin for two values of the damping parameter α. The plots show FINMAG's solution drawn with markers and the analytical solution as lines.

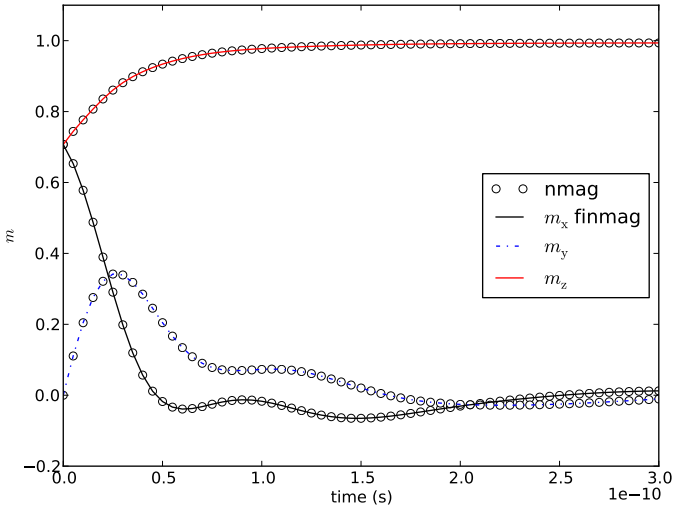


FIGURE 5.6: Example with exchange and demagnetising field. Average unit magnetisation $\boldsymbol{m}$ over time for FINMAG and Nmag.

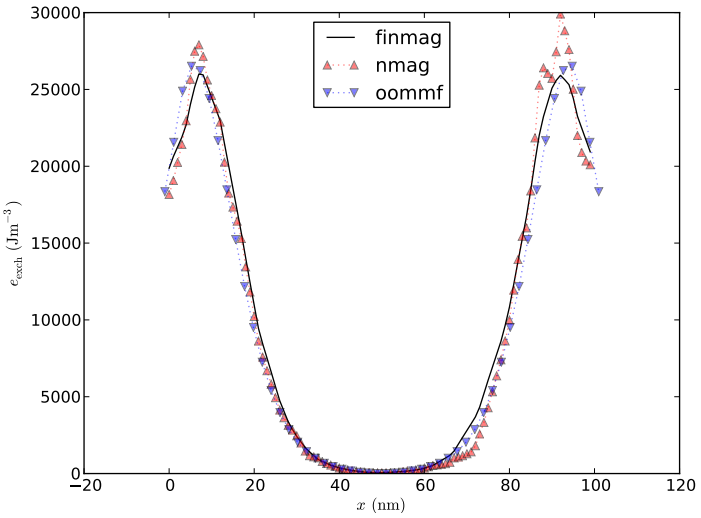


FIGURE 5.7: Example with exchange and demagnetising field. Exchange energy density through the x-axis of the bar.

5.4 Solution to Standard Problem #3

The Micromagnetic Modeling Activity Group (μmag) has defined a number of standard problems for the validation of micromagnetic simuliation software. The definitions of the problems and solutions from different research groups have been published on its website [4]. As an example, the standard problems #3 is now presented using FINMAG for its solution.

A cube with edge length L, expressed in units of the intrinsic length scale, $l_{ex} = \sqrt{A/K_m}$m is simulated, where A is the exchange coupling strength and $K_m = \frac{1}{2}\mu_0 M_S^2\,\mathrm{kg\,m^{-1}\,s^{-2}}$ is the magnetostatic energy density. The uniaxial anisotropy parameter is K_u with $K_u = K_m/10$, and the easy axis is directed parallel to a principal axis of the cube.

Depending on the edge length L, the total energy of the magnetisation will either be lower for a modified single domain state called flower state or a vortex state. The problem now consists in

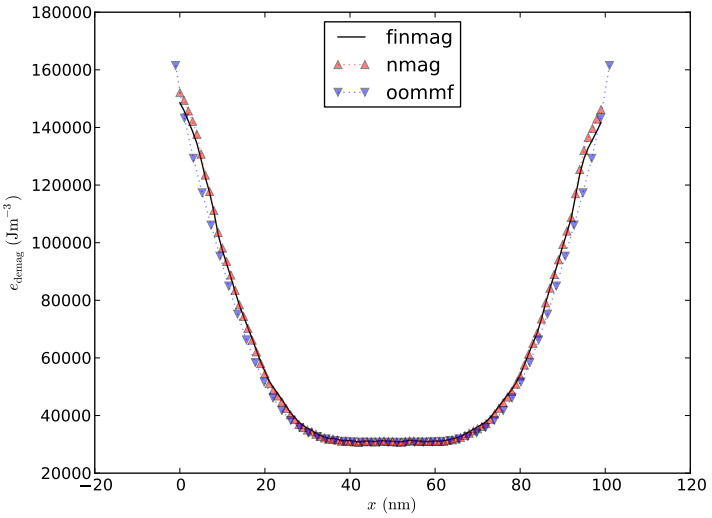


FIGURE 5.8: Example with exchange and demagnetising field. Demagnetisation energy density through the x-axis of the bar.

	FINMAG	Rave et al.	J. Martins	R. Hertel
L	8.4375	8.47	8.4073-8.4687	8.52

TABLE 5.5: Single domain limit L for standard problem 3. Contrary to what is said in the problem definition, J. Martins found that L does depend on the discretisation size [4].

	FINMAG		Rave et al.		R. Hertel	
	flower	vortex	flower	vortex	flower	vortex
Demag.	0.277	0.076	0.279	0.078	0.284	0.083
Exchange	0.018	0.173	0.018	0.172	0.016	0.170
Anisotropy	0.006	0.052	0.006	0.052	0.005	0.052
Total		0.301		0.302		0.305

TABLE 5.6: Energy contributions in the flower and vortex state in the standard problem 3. The energies are relative to K_m.

	FINMAG		Rave et al.		R. Hertel	
	flower	vortex	flower	vortex	flower	vortex
m_x						
m_y		0.347		0.352		0.344
m_z	0.970		0.971		0.874	

TABLE 5.7: Components of the spatially averaged unit magnetisation $\boldsymbol{m}$ in the single domain limit. Empty cells have the value 0. While the magnetisation points in the direction of the easy axis in the flower state, the vortex state has two possible configurations.

finding the value of L for which the two states have equal energy. That value of L is called the single domain limit. The individual energy contributions and the spatially averaged magnetisation need to be recorded as well.

The findings are presented in Tables 5.5 to 5.7 and compared with the solutions of Rave et al. [229], J. Martins [4], and R. Hertel [4]. Overall, FINMAG's results are in line with those published on the μmag website.

5.5 Performance of Krylov Methods in Demagnetising Field Computation

For the computation of the demagnetising field, both the method proposed by Fredkin and Koehler [37] and by Garcia-Cervera and Roma [226] have been implemented. Because computing the demagnetising field easily accounts for more than half of the total computation time in any given simulation, care was taken to avoid any easily preventable loss of performance. One area of special consideration was the choice of preconditioner and Krylov subspace method for the solution of the two parts of the magnetic potential as reviewed in Section 3.3.2 and Section 3.5.3.

To determine the fastest numerical method available to solve the two systems, two simulation setups were investigated. The first one is a homogeneously magnetised sphere of radius 20 nm with 13431 vertices. The second simulation is a thin magnetic film of dimensions $500 \times 50 \times 1$ nm with 14330 vertices. Because this system's volume is negligible in relation to its surface, we expect the solution of the inhomogeneous Neumann problem to bear most of the computational load.

Via the intermediary of FEniCS, PETSc [230] provides seven Krylov subspace methods along with 14 preconditioners and thus 97^2 different combinations of solver and preconditioner. An overview of the used linear solvers is given in Table 5.8, while the different preconditioners are shown in Table 5.9.

Abbreviation	Solver	Reference
cg	Conjugate gradient method	Hestenes, Stiefel [163], 1952
gmres	Generalized minimal residual method	Saad, Schultz [231], 1986
minres	Minimal residual method	Paige, Saunders [232], 1975
tfqmr	Transpose-free quasi-minimal residual method	Freund [233], 1993
richardson	Richardson method	Richardson [234], 1910
bicgstab	Biconjugate gradient stabilized method	van der Vorst [235], 1992

Table 5.8: List of linear solvers used in the numerical study of magnetised sphere and thin film.

Abbreviation	Preconditioner	Reference
ilu	Incomplete LU factorisation	Meijerink, van der Vorst [236], 1977
icc	Incomplete Cholesky factorisation	described in Ref. [237]
sor	Successive over-relaxation	Young [238], 1950
petsc_amg	PETSc algebraic multigrid	PETSc [230]
jacobi	Jacobi iteration	c.f. Section 3.5.3
bjacobi	Block-Jacobi	PETSc [230]
additive_schwarz	Additive Schwarz	described in Ref. [237]
amg	Algebraic multigrid	Ruge, Stüben [239], 1987
hypre_amg	Hypre algebraic multigrid (BoomerAMG)	Henson, Yang [240], 2002
hypre_euclid	Hypre parallel incomplete LU factorization	Hysom, Pothen [241], 2001
hypre_parasails	Hypre parallel sparse approximate inverse	Chow [242], 2000

Table 5.9: List of preconditioners used in the numerical study of magnetised sphere and thin film.

We covered the Jacobi preconditioning in Section 3.5.3.

[2] in FEniCS, behind the scenes, the combination default/default corresponds to the generalised minimal residual method (gmres) with incomplete LU factorisation (ilu) as a preconditioner.

Sparse linear systems can be solved using the factorisaton $\mathbf{A} = \mathbf{LU}$ of $\mathbf{Ax} = \mathbf{b}$, with the lower unitriangular matrix $\mathbf{L}$ and upper triangular matrix $\mathbf{U}$. The problem reduces to solving $\mathbf{Ly} = \mathbf{b}$ and $\mathbf{Ux} = \mathbf{y}$. Because the matrices are triangular, this can be done efficiently. However, $\mathbf{L}$ and $\mathbf{U}$ end up much less sparse than the original matrix [237], which introduces a bottleneck in the form of memory requirements. Instead, an incomplete factorisaton only seeks triangular matrices for an approximation $\mathbf{A} \approx \mathbf{LU}$. This approach gives the ilu preconditioner its name, which stands for incomplete $\mathbf{LU}$. However, solving the system $\mathbf{LUx} = \mathbf{b}$, while faster, does not give an exact solution to the original problem $\mathbf{Ax} = \mathbf{b}$. To obtain an accurate solution, the matrix $\mathbf{P} = \mathbf{LU}$ is used as a preconditioner for another iterative solver, such as the conjugate gradient method we reviewed in Section 3.5.3.

To weed out the combinations of solver and preconditioner that fail to converge, the results are compared to the known analytical expression for the homogeneously magnetised sphere, and to the default choices for the thin magnetic film. For each of the two systems, the demagnetising field is computed ten times and the average is taken.

The results are presented in Tables 5.10 to 5.13 and put into perspective in Figures 5.9 to 5.12.

	default	cg	gmres	minres	tfqmr	richardson	bicgstab
default	32.1	26.4	32.5	30.9	34.0	314.1	32.8
none	73.8	41.4	73.4	46.7	59.7	–	53.2
ilu	32.1	26.5	32.2	29.8	32.4	316.3	33.2
icc	32.9	27.3	33.0	30.9	33.5	334.5	35.3
sor	38.9	30.0	39.1	33.4	36.8	4681.1	40.0
petsc_amg	147.3	–	148.6	153.8	–	168.6	244.7
jacobi	64.7	35.0	65.3	42.4	48.8	867.1	43.9
bjacobi	35.0	27.0	33.5	30.0	32.9	315.5	34.0
additive_schwarz	35.7	30.3	35.5	33.3	37.1	350.8	36.9
amg	213.3	–	215.5	210.4	–	218.5	587.2
hypre_amg	205.9	–	205.9	206.5	–	217.6	587.7
hypre_euclid	60.8	54.6	60.9	58.0	60.7	362.7	61.6
hypre_parasails	477.5	–	477.2	–	481.0	1115.0	472.1

TABLE 5.10: Time spent solving the first linear system for the magnetised sphere as a function of preconditioner (rows) and Krylov subspace method (columns).

	default	cg	gmres	minres	tfqmr	richardson	bicgstab
default	22.6	18.4	22.4	–	24.0	89.7	26.8
none	43.4	2156.6	42.6	–	–	–	–
ilu	22.7	18.4	23.2	–	23.9	87.6	26.7
icc	24.2	–	24.7	–	–	93.7	–
sor	23.8	22.0	23.4	–	25.8	4732.2	28.3
petsc_amg	135.4	149.3	135.4	–	140.4	–	140.6
jacobi	28.9	–	28.8	–	–	152.4	–
bjacobi	22.4	18.4	22.3	–	24.2	86.0	26.2
additive_schwarz	24.7	20.3	24.6	–	26.9	91.5	28.9
amg	142.2	141.5	142.0	148.8	160.9	145.6	156.8
hypre_amg	144.3	143.1	143.8	149.8	157.0	149.4	156.1
hypre_euclid	48.7	44.7	48.3	–	50.3	111.2	52.9
hypre_parasails	408.0	–	402.4	–	399.9	519.8	400.4

TABLE 5.11: Time spent solving the second linear system for the magnetised sphere as a function of preconditioner (rows) and Krylov subspace method (columns).

	default	cg	gmres	minres	tfqmr	richardson	bicgstab
default	572.0	70.8	584.4	82.1	–	–	114.7
none	–	178.7	–	233.4	–	–	321.1
ilu	561.2	69.5	560.1	83.0	–	–	111.6
icc	572.2	71.7	571.3	85.0	–	–	120.5
sor	1728.4	134.7	1729.1	158.7	259.3	3461.6	220.2
petsc_amg	94.2	–	93.6	164.1	–	–	–
jacobi	3451.8	161.9	3418.9	215.6	–	–	251.3
bjacobi	569.0	71.5	567.6	83.8	–	–	114.8
additive_schwarz	608.7	76.2	599.7	90.6	–	–	125.1
amg	50.9	–	50.7	51.2	–	70.4	115.5
hypre_amg	50.7	–	50.7	51.3	–	70.4	115.7
hypre_euclid	625.9	100.2	627.9	114.0	–	–	152.6
hypre_parasails	1625.2	–	1617.9	–	–	–	293.0

Table 5.12: Time spent solving the first linear system for the thin film as a function of preconditioner (rows) and Krylov subspace method (columns).

	default	cg	gmres	minres	tfqmr	richardson	bicgstab
default	1.7	1.6	1.7	–	2.1	1.6	2.1
none	1.5	2.4	1.5	3.0	–	–	–
ilu	1.7	1.6	1.7	–	2.1	1.7	2.1
icc	2.8	5.7	2.8	6.8	2.9	2.6	2.7
sor	1.1	1.0	1.0	–	1.5	3450.8	1.5
petsc_amg	22.9	24.6	22.4	25.3	27.0	25.4	22.5
jacobi	0.8	2.2	0.8	3.3	0.9	0.7	0.8
bjacobi	1.7	1.7	1.7	–	2.1	1.6	2.1
additive_schwarz	2.7	2.6	2.7	–	3.1	2.6	3.2
amg	3.8	3.8	3.9	–	5.1	3.0	4.9
hypre_amg	3.9	3.8	3.9	–	4.9	2.8	4.8
hypre_euclid	25.6	25.6	25.4	–	25.7	25.4	25.7
hypre_parasails	20.5	20.6	20.7	–	20.6	20.1	20.3

Table 5.13: Time spent solving the second linear system for the thin film as a function of preconditioner (rows) and Krylov subspace method (columns).

For the magnetised sphere, the potential speed-ups of around 18 % for both linear solves are moderate. In the case of the thin film on the other hand, changing the default choice for the solution of the first potential to a combination of the generalised minimal residual method (gmres) with the hypre algebraic multigrid (hypre_amg) as a preconditioner cuts down the runtime by 91 %. For the solution of the second potential, a time saving of up to 55 % is possible. Because the conjugate gradient method (gc) with the incomplete LU factorisation (ilu) offers improved performance for both systems for the first linear solve, it was chosen as FinMag's new default. The new settings translate to a 70 % time saving when computing 0.1 ns of the magnetisation dynamics for the thin film (a Zeeman field and exchange interaction were added). The code used to generate the tables and figures shown is part of FinMag and can be used by the user to optimise the settings for his system.

There are benchmarking tools for Python programs, like cProfile. But benchmarking tools run at a low level of analysis and as in this case, present an unwieldy amount of data. In the approach presented here, specific units of code are annotated so that execution time is recorded during the simulations. This allows both for the precise examination of assumptions about the programs'

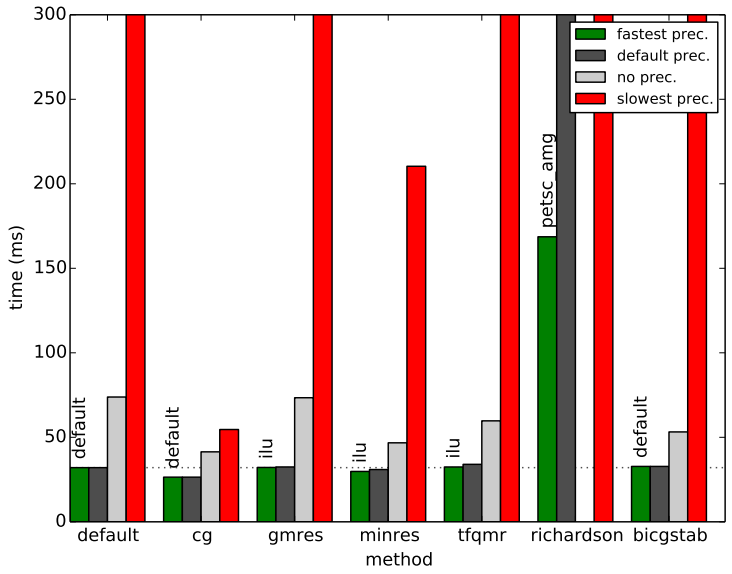


FIGURE 5.9: Time spent solving the first linear system for the magnetised sphere. For each solver, the runtime for the fastest, slowest, and default preconditioners are shown, along with a run without preconditioner.

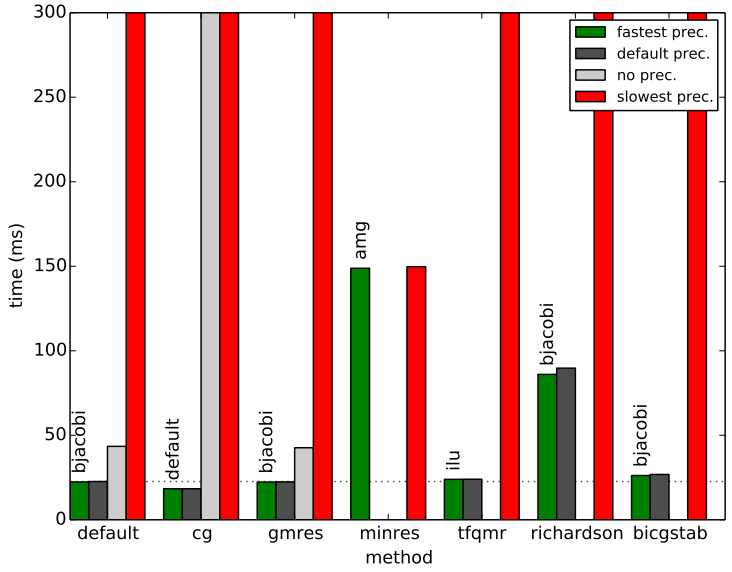


FIGURE 5.10: Time spent solving the second linear system for the magnetised sphere.

runtime behaviour and comparative tests of numerical computations. The profiling has been extracted into a python module published on the Python package index [243].

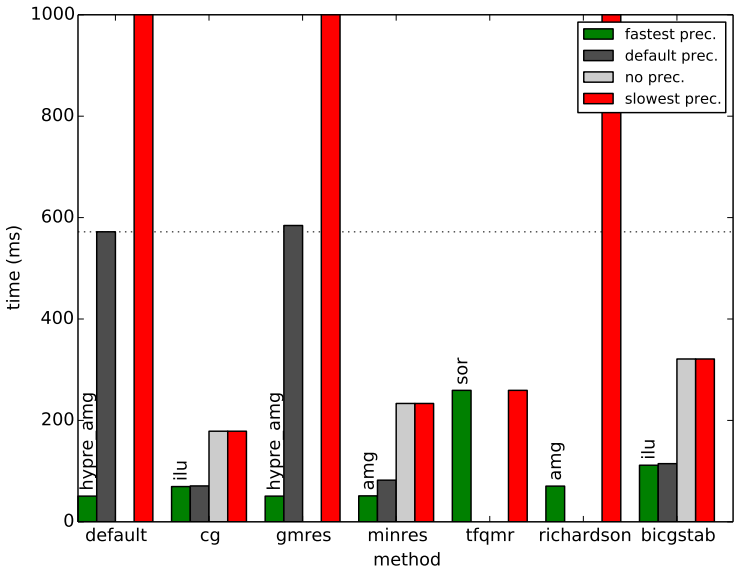


FIGURE 5.11: Time spent solving the first linear system for the thin film. For each solver, the runtime for the fastest, slowest, and default preconditioners are shown, along with a run without preconditioner.

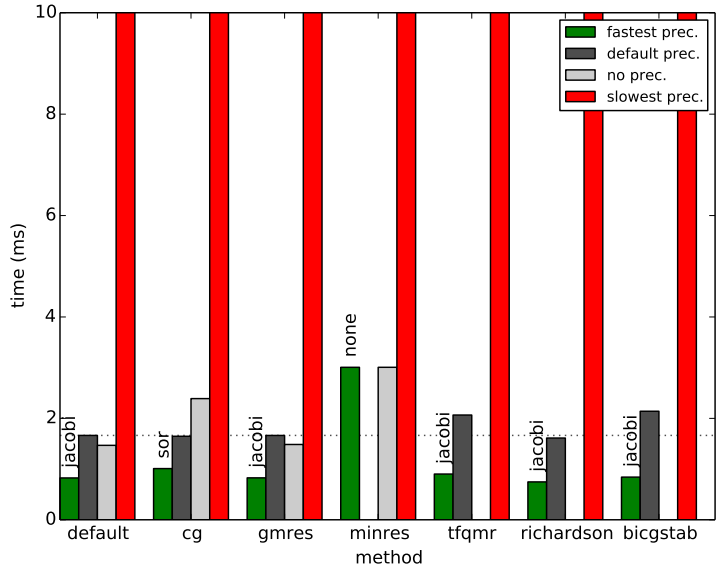


FIGURE 5.12: Time spent solving the second linear system for the thin film.

5.6 Use of FinMag in Research

In this section, studies to which the work presented contributed to both directly and indirectly through starting and continuing the development of FINMAG will be presented. The focus is on how FINMAG is well suited for academic research in the field of computational micromagnetics. Where specific contributions where made in preparation of this thesis, they will also be mentioned. For the first few papers described, this would be the implementation of the Dzyaloshinskii-Moriya

energy described in Section 5.2. FinMag was the first micromagnetic software in which this term was implemented.

In "Magnon-Driven Domain-Wall Motion with the Dzyaloshinskii-Moriya Interaction" [185], Wang et al. study the domain wall motion induced by magnons (spin wave excitations) in the presence of DMI. A head-to-head domain wall is put on a FeGe wire with exchange interaction, easy-plane uniaxial anisotropy, and DMI present. Spinwaves with a given frequency are excited on one end of the wire and the position and velocity of the domain wall is recorded as a function of time. It is known that when a magnon travels through a domain wall, the latter moves due to angular momentum of the magnon being transferred to the magnetisation structure. This happens without a change of the spinwave vector or frequency. The simulations show that the domain wall is driven with a velocity higher by an order of magnitude in presence of DMI. While the frequency of the magnon isn't significantly affected, this is not the case for the wave vector. The higher efficiency is attributed to linear momentum transfer. Unlike Nmag or magpar, FinMag also supports one-dimensional discretisations and two-dimensional meshes. This made it possible to carry out the simulations on a one-dimensional mesh, saving computational time compared with a three-dimensional mesh due to a reduced number of degrees of freedom. Also it produced results that were easier to compare to a one-dimensional analytical model for the control case.

In "Phenomenological description of the nonlocal magnetization relaxation in magnonics, spintronics, and domain-wall dynamics" [186], Wang et al. investigate a modified Landau-Lifshitz equation analytically and numerically. Baryakhtar derived a phenomenological magnetic relaxation term to explain the discrepancy between magnetic damping constants obtained from magnetic domain wall velocity measurements in dielectrics and ferromagnetic resonance experiments (as cited in Ref. [186]). Wang focuses on the manifestation of the Baryakhtar relaxation in problems specific for magnonics and domain wall dynamics. The simulations that were carried out with FinMag and OOMMF show that the Baryakhtar relaxation leads to an increased damping of short wavelength spin waves and a lower domain wall velocity. The LLG equation we reviewed in Section 3.2 assumes a magnetisation with constant norm $|\boldsymbol{M}(\boldsymbol{r})| = M_\mathrm{S}$. At each moment in time, the magnetisation relaxes towards the instantaneous direction of the effective magnetic field. This is not the case for the Landau-Lifshitz-Baryakhtar equation, where the norm of the magnetisation can deviate from its equilibrium value M_e and relaxes towards the value prescribed by the instantaneous longitudinal effective magnetic field. In comparison with the LLG equation, the Landau-Lifshitz-Baryakhtar (LLBar) equation brings an additional degree of freedom (the norm of the magnetisation) into the discussion. In FinMag, the code dealing with the computation of the LLG equation is decoupled from matters like time integration or the user interface. This made it possible to start with the LLG code as a self-contained template for the LLBar code without necessitating other changes to FinMag besides introducing it as an additional option during the creation of a simulation.

In "Ground state search, hysteretic behaviour, and reversal mechanism of skyrmionic textures in confined helimagnetic nanostructures" [6], Beg et al. identify the lowest energy states in helimagnetic thin film nanostructures at zero external magnetic field and in absence of magnetocrystalline anisotropy, but in presence of DMI and the demagnetising field. The simulations were run on the high performance computing facility Iridis of the University of Southampton. They show that the ground state in the studied system is a skyrmionic texture. Regions of metastability for the skyrmionic texture for disks of different sizes are also explored. The cyclindrical geometry

```
import os
from finmag import Simulation as Sim
from finmag.util.meshes import nanodisk

def reversal_sim(d, thickness, lmax, Ms=3.84e5, A=8.78e-12, dirname='./'):
    """ Convenience function for simulating reversal mechanism. """

    if not os.path.exists(dirname):
        os.makedirs(dirname)

    # Create a nanodisk mesh.
    mesh = nanodisk(d, thickness, lmax, save_result=False)

    # Create a simulation.
    sim = Sim(self.mesh, Ms, unit_length=1e-9)

    # Add energies.
    sim.add(Exchange(A))
    ...
```

LISTING 5.3: A snippet of the code used to create FINMAG simulations in Ref. [6] showing the programmatic mesh creation. FINMAG uses the NETGEN mesh generator internally [7]. The simulations are created in a function called `reversal_sim`. Some of its parameters have been ommited for brevity.

studied warranted the use of a finite element based simulation code, as we saw in Section 3.5.2. The list of energies that can be computed using Nmag doesn't include the DMI. Also, using FINMAG, it is possible to create a mesh of a cylinder with a given thickness and diameter from within the simulation script, as the code example in Listing 5.3 shows. This would not have been possible with Nmag, where handling the files specifying the geometry and the corresponding generated meshes as well as the loading of the mesh in the simulation present a source for potential confusion and error instead. The specification of the geometry in the simulation script occurs more commonly in finite differences based codes, due to the lower complexity of the discretisation inherent in the finite difference approach. OOMMF however wouldn't have supported expressing the meta or batch logic needed to conveniently and safely create and recreate simulations for a set of system geometries.

In "Skyrmions in thin films with easy-plane magnetocrystalline anisotropy" [244], Vousden et al. establish the presence of skyrmions as ground state on B20 thin film materials. The system was modeled after a $Fe_{0.7}Co_{0.3}Si$ thin film of 5 nm thickness. It included a Zeeman field, easy-plane uniaxial anisotropy, the DMI, and the demagnetising energy. The simulations carried out with FINMAG explored the phase space of the system with regards to the anisotropy and Zeeman field values, as well as different starting magnetic configurations. The results show that chiral skyrmionic magnetic configurations are the lowest energy states in this system. The size of the observed skyrmions increases with the anisotropy. The conclusion of the study is in counterpoint with earlier analytical research (Ref. [245, 246] as cited in Ref. [244]). However it gives credence to a numerical study (Ref. [247] as cited in Ref. [244]) that also reported the presence of skyrmions in the studied system, but modeled a two-dimensional infinite plane and didn't account for thickness and boundary effects like the full 3D model of FINMAG and the inclusion of the demagnetising energy could. This is all the more important as long as the role of layer thickness in skyrmion stability is not fully understood. Further, the results published in the letter by Vousden

et al. are compatible with experimental results (c.f. the references cited in Ref. [244]). As such, FinMag helped bridge the gap existing between analytical models, simplified computational models on one hand and the experimental data on the other.

The remarks in the last paragraph also apply to the work conducted in "Hysteresis of nanocylinders with Dzyaloshinskii-Moriya interaction" by Carey et al. [248], to which we contributed the code to compute hysteresis loops with FinMag. In this publication, Carey et al. investigate the question of existence of skyrmions in cylindrical nanostructures of variable thickness. The simulations carried out with FinMag show the existence of skyrmions but also the role the demagnetisation energy plays in stabilising the skyrmion in nanocylindrical samples. This further cements the use of a full micromagnetic code in skyrmion research. The study also uses the capability of FinMag to compute hysteresis loops. In a hysteresis loop simulation, the direction and magnitude of an external field is varied along a given axis. For every field value, the magnetisation is relaxed anew and the magnetisation pattern is recorded. In the letter, the loops are analysed and classified into accessible states. As Carey rightly remarks, hysteresis loop measurements are a standard laboratory magnetometry protocol – highlighting the practical aspect of the approach.

In "Frequency-based nanoparticle sensing over large field ranges using the ferromagnetic resonances of a magnetic nanodisc" [3], Albert et al. apply FinMag to a research question in the domain of sensing biomedical samples using magnetic nanoparticles. In the described biosensing setup depicted in Figure 5.13, a magnetic nanoparticle is suspended in a supposed non-magnetic bodily fluid. The fluid contains biological analytes of interest to which the nanoparticle is bound by antibodies. In conventional magnetoresistive sensors, the nanoparticle then induces a change in the static magnetisation configuration of the device's sensing layer, which can be picked up electronically through the change in the device resistance. In the sensor described by Albert et al., the presence of the particle in the vicinity of the sensing layer is made apparent by a shift in the resonant frequency of the disc's fundamental mode instead. Because the ferromagnetic resonance frequency of the device responds directly to the stray demagnetising field emanating from the particle, the detection works even if the underlying ground state is left unchanged. This unlocks frequency based detection schemes with better size scalability compared to static magnetoresistive sensing. Albert extended FinMag by an eigenvalue problem-based method to identify the resonant modes within the disc, while some work was necessary to make FinMag be able to correctly model the non-magnetic space between the magnetic nanoparticle and sensing layer. Like in the study described in the last paragraph, Jupyter notebooks are distributed in the electronic supplementary material associated with the publication to reproduce its plots from the also included raw data.

Together with OOMMF, Magpar, Nmag, and Fidimag, FinMag is one of the micromagnetic packages included in "Virtual Micromagnetics: A Framework for Accessible and Reproducible Micromagnetic Simulation" by Vousden et al. [249]. The Virtual Micromagnetics project provides virtual machine simulation environments to run open-source micromagnetic simulation packages. These environments allow easy access to simulation packages that are often difficult to compile and install, and enable simulations and their data to be shared and stored in a single virtual hard disk file, which encourages reproducible research. Virtual Micromagnetics can be extended to automate the installation of micromagnetic simulation packages on non-virtual machines, and to support closed-source and new open-source simulation packages, including packages from disciplines other than micromagnetics, encouraging reuse. The virtual machines are provisioned

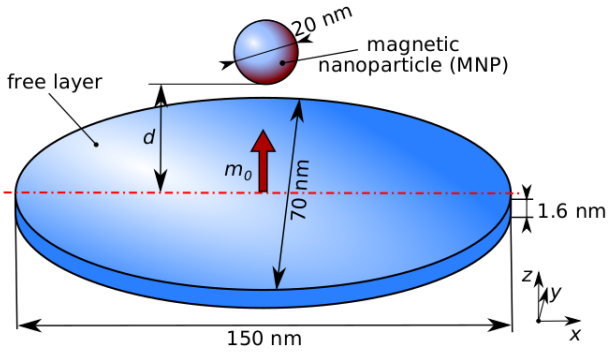


FIGURE 5.13: Geometry used for the simulations in Ref. [3], composed of a magnetic nanoparticle and an elliptical magnetic disc. Note the empty space between the particle and the disc, which still needs to be accounted for by the simulation code. Image from Ref. [3] reproduced with permission.

with Ansible. Ansible is software that automates software provisioning, configuration management, and application deployment [250]. The ansible scripts are used to install the micromagnetic packages in the virtual machines before the machines are packaged. The Ansible scripts were contributed as part of the work described in this thesis.

In "Proposal of a micromagnetic standard problem for ferromagnetic resonance simulations" (Ref. [251]), Baker et al. compute ferromagnetic resonance (FMR) spectra. FMR is a technique closely associated with many practical uses ranging from material characterization to the study of spin dynamics (Ref. [252] as cited in Ref. [251]). With FMR, magnetisation dynamics are probed in samples using microwave fields. Three different methods were used to simulate the FMR. Full micromagnetic simulations involving a time-dependent magnetic microwave field. The ringdown method, where a magnetic configuration in equilibrium is perturbed by a short-lived and weak excitation with a specific frequency. The simulation dynamics can be recorded and the resonance frequencies can be extrated via the Fourier transform. Finally, an eigenvalue method exists where the resonant frequencies are obtained as eigenvalues in an eigenvalue problem restated accordingly. Baker notes that the test constrains the simulated samples to cuboid so that a data point can be optained with a finite difference approach. Accordingly, Baker et al. present simulation results obtained with Nmag and OOMMF. Using full micromagnetic simulations and the ringdown method, we used FINMAG during the study to corroborate the findings obtained with the other codes.

In "Dynamics of skyrmionic states in confined helimagnetic nanostructures" the resonance frequencies and corresponding eigenmodes of skyrmionic states in thin film FeGe disk samples were studied using FINMAG. Beg et al. (Ref. [253]) use the ringdown method to predict the eigenmodes that are to be expected in experiment with FMR. Key differences in power spectral densities between different skyrmionic states are presented to contribute to the experimental identification of the state present in the sample. The study also emphasises how neglecting the demagnetisation energy contribution changes the obtained resonance frequencies substantially [253]. We contributed the code that computes the power spectral densities.

Pepper et al. show that skyrmionic states can form the ground state for a range of system sizes in both triangular and square-shaped FeGe nanostructures of 10 nm thickness in "Skyrmion states in thin confined polygonal nanostructures" (Ref. [254]). In doing so, Pepper et al. investigate

to which extent skyrmionic states are stable in geometries that do not match the cylindrical symmetry of the skyrmion.

FinMag has also found use in research publications not specifically related to the work presented in this thesis, namely:

1. Bin Zhang, Weiwei Wang, Marijan Beg, Hans Fangohr, and Wolfgang Kuch.
   Microwave-induced dynamic switching of magnetic skyrmion cores in nanodots.
   Applied Physics Letters 106, 102401 (2015) [255].

2. Weiwei Wang, Marijan Beg, Bin Zhang, Wolfgang Kuch, and Hans Fangohr.
   Driving magnetic skyrmions with microwave fields.
   Physical Review B (Rapid Communications) 92, 020403 (2015) [256].

Chapter 6

Research Software Engineering

As reviewed in Chapter 2, a gap between computational science and computer science has existed since the beginnings of modern computing. Computer science has historically been rooted in the disciplines of electrical engineering and applied mathematics. To distinguish itself from its origins computer science aimed for generality in its methods and techniques. Computational sciences engage in the "study of computer" only insofar as it is necessary for research in a particular field of science. In "Software Engineering for Computational Science", Johanson and Hasselbring identify recurring key characteristics of scientific software development that are the result of the nature of scientific challenges, the limitations of computers, and the cultural environment of scientific software development [19]. They point out the shortcomings of existing approaches between software engineering and computational science. They argue that modern practices of software engineering have not effectively been disseminated in computational science. Johanson's et al study was written from the perspective of computer science. Through the use of a literature survey they have created a theoretical framework to discuss the issues and the underlying causes for them in creating scientific software. However the issues occurring are of a practical matter and therefore can be studied by experiment.

In Chapter 4 and Chapter 5, two micromagnetic simulation softwares named Fidimag and FinMag are presented. They serve as case studies for the investigation of good software engineering practice and research software quality that follows. A number of modern software engineering techniques were used and assessed during the development of the two softwares. Some of the techniques used are possible specific solutions to the problems raised in Chapter 1. The techniques are presented in this chapter and discussed within the framework outlined by Johanson et al.

6.1 Research Software Engineering in an Academic Setting

In the computational sciences, researchers traditionally do not have their educational background in computer science, but from a discipline closer to the scientific problem being solved. For the University of Southampton, Table 6.2 shows the courses related to software engineering offered in the four-year Master programmes of physics, electrical engineering, and mathematics.

Course	Modules
Msc Physics	Physics Skills - Programming and Data Analysis, Computer Techniques in Physics (2)
MSc Electrical Enginering	Programming, Engineering Design (2)
MSc Mathematics	Operational Research I and Mathematical Computing (1) and elective modules Numerical Methods Mathematical Programming (2)

TABLE 6.1: Modules with software engineering content in selected programmes of the University of Southampton [5].

While some of the courses do include programming in C or Python, little or no formal training in software engineering and its practices can be assumed from new reseachers. In recent years, organisations such as Software Engineering for Science [257] and the Software Sustainability Institute [258] have been formed to identify key issues in scientific software. They organise workshops such as the software carpentry workshop, which is a two-day event teaching basic lab skills for research computing. Knowledge about good software engineering practice is acquired either independently, "on the job", and from colleagues. But lack of formal training aside, in the academic context there is also the issue of lacking incentives with regards to the parts of software engineering that are distinct from research. Policies requiring the data in results and publications to be shared have recently been adopted by some publishers and funders [259]. The situation with regards to code might change when the policies are extended to code generating scientific data.

The two projects presented in this thesis went from personal projects to codes worked on by several researchers [1]. The continuing development of FINMAG and Fidimag has been a collaborative effort with many developers. The researchers drew from the disciplines physics, electrical engineering, mechanical engineering, and mathematics.

6.1.1 Source Control

Source or version control encompasses the tracking and managing of changes to code. The codes presented in this thesis were first stored in Mercurial repositories [260]. Mercurial supports working in a truly distributed manner and in theory doesn't have a concept of server or master repository. In practice it is useful to being able to synchronise to a common reference repository. The repositories were first stored on a spare machine, then moved to the online offering of Bitbucket [261]. Bitbucket offered free private repositories. The repositories had to migrate to git [262] when Bitbucket dropped support for Mercurial in favour of git. The online offering GitHub introduced a workflow called pull requests, that made it easy to discuss, review, and modify the potential changes with collaborators [263]. When FINMAG and Fidimag were open sourced, the repositories moved to GitHub. Table 6.2 shows the number of lines of source code for Fidimag, FINMAG, Nmag, and OOMMF.

For the source code submission, no formal process was established. Long running developments or refactorings were made using feature branches. This lead to the occasional merge conflict when incorporating the changesets into the master branch. It was encouraged to keep the master branch building and correct. Two concepts that will be revisited in Section 6.2. Code submissions

[1] The initial implementation of the code that became Fidimag was created by Dr. W. Wang

	Lines of Code
Fidimag	12991 Python, 2.871 C, 9.851 C++, 1532 Cython, 244 make
FinMag	40153 Python, 4893 C, 4549 C++, 533 Cython, 794 Bash, 658 make
Nmag 0.2.1	30199 OCaml, 25835 Python, 9328 C, 8321 Bash, 397 make
OOMMF 2.0 alpha 2	152261 C++, 56614 Tcl/Tk, 1990 Perl

Table 6.2: Lines of source code for Fidimag, FinMag, Nmag, and OOMMF. Data generated with cloc version 1.74.

were handled by a combination of informal discussions, pull requests and the meetings discussed in the next section.

6.1.2 Agile Method

The researchers met in a weekly or biweekly fashion for mini-sprints called FinMag Fridays and later Fidimag Fridays. The meetings fostered a culture of learning and open participation. The Fridays replaced an online project management tool, shown in Figure 6.1, that was used to keep track of researchers activities and the shared backlog of ideas.

Johanson and Hasselbring raise the issue of the difficulty of creating a shared understanding of a code, especially in computational science, due to high personal turnover rates and lack of produced documentation [19]. According to them, scientists prefer more informal, collegial ways of knowledge transfer. Pair programming is an agile software development technique in which two developers work together at one computer. One developer, the driver, writes code while the other reviews each line of code as it is typed in. The two developers switch roles frequently. We found pair-programming to be effective at creating a shared understanding of the code and also at disseminating domain knowledge in the field of computational micromagnetics, where new scientists are for example not likely to be intimately familiar with both the finite element method and magnetism at the nanoscale.

6.2 Software Quality Assurance

The means used to monitor and ensure the quality of the softwares presented in this thesis will be described in the following sections.

6.2.1 Software Testing

Test-driven development (TDD) is a software development process with a very short development cycle. Test cases are written based on requirements, after which the software is extended to pass the new tests only. So in the strictest implementation tests are written before or alongside the tested units. It is however usually understood to mean a coding practice focusing on the testability of the code during development.

Following test-driven development, tests are produced as a natural consequence of programming. This not only prevents the testing phase in software development from being skipped, but also

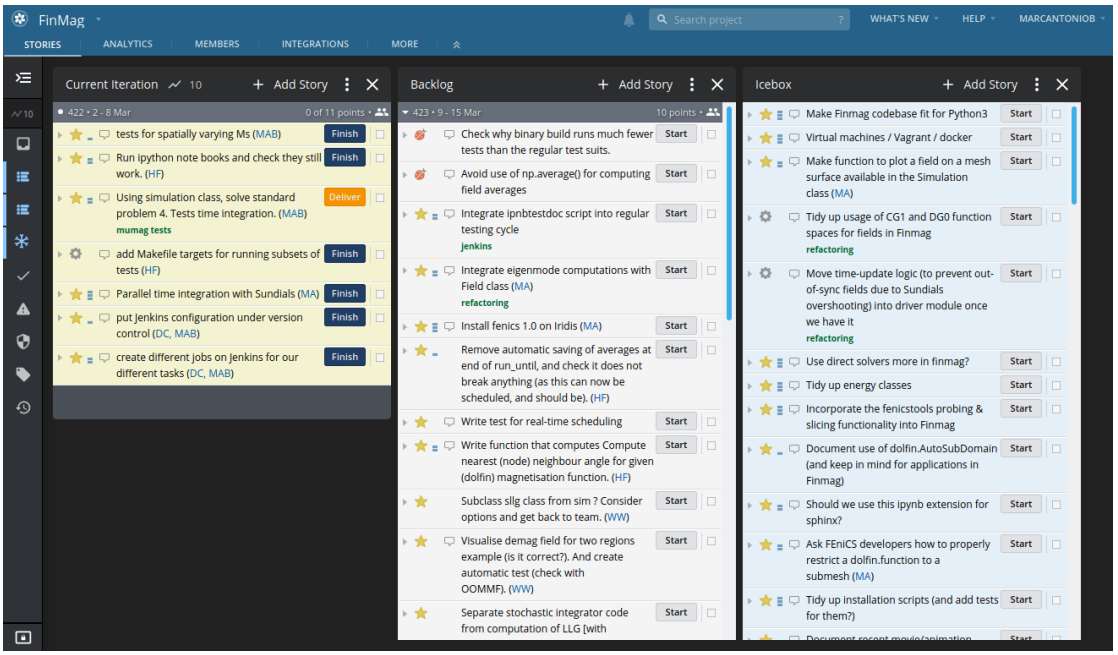


FIGURE 6.1: Screenshot from pivotaltracker.com project management tool.

affects the software design: In the waterfall or V-model software development processes, the software (including the boundaries between modules and their interfaces) are designed, then programming is carried out. After the programming the software is tested. It is during software testing that the interface exposed by the code is consumed for the first time and "user" code is written. Flaws in the software design from an outside's point of view thus only become apparent after the programming is done. With test-driven development, testing occurs alongside programming. It makes it possible for the insight from concretely using the interface to inform the programming. Thus test-driven development affects the software design positively. This is true in particular for softwares like Fidimag and FINMAG, that are meant to be used by means of their API from within other codes, because integration testing is indistinguishable from actual usage of the code.

Writing the tests reduces the complexity of the task as the focus is on the design without having to worry about the implementation. The tests also document the design, because each test is an example use case. It establishes that the code implements the design and encourages the design of testable code. It achieves a high test coverage of code automatically so that debugging can benefit from the tests. Finally, in the sense that TDD gives confidence in the code, it makes refactoring easier. Beck argues that TDD is a way of managing what he calls "fear" during programming [227]. Fear in the "this is a hard problem and I can't see the end from the beginning" sense, according to Beck, makes one person tentative, grumpy, want to communicate less, and avoid feedback. It also freezes creativity, which hinders the exploration of new ideas. Beck argues that confidence in the code reduces fear. We note that version control is another key part of the solution because it is always possible to return to a state of the source code that verifies.

TDD was used as a method to develop the micromagnetic softwares presented in this thesis. Table 6.3 shows the number of test cases in Fidimag, FINMAG, Nmag, and OOMMF.

	Test Cases
Fidimag	167
FinMag	552
Nmag 0.2.1	49
OOMMF 2.0 alpha 2	35

Table 6.3: Number of test cases in source codes of Fidimag, FinMag, Nmag, and OOMMF. Counted using `grep -R def 'test_' | wc -l` for Fidimag, FinMag, and Nmag, and by counting for OOMMF.

There are around 150 tests for Fidimag, and around 500 test cases in FinMag. Besides unit tests, there are integration tests that run simulations and compare the obtained energies and fields with known-good results from analytical solutions, previous versions of the simulator or other simulators. The tests include longer running simulations like the standard problems #3 and #4 that were discussed in Sections 4.4 and 5.4.

When the tests take too long to run, it becomes unpractical to run them on the developer's machine before code submission. Three tiers of testing were introduced as a result. A small number of simulations that run in under a minute in total were hand-picked to form of smoke testing before code submission. The bulk of tests ran unchanged. The slowest tests were marked to only run when specified on the command line. As a first measure slow tests were marked to only run when asked for on the command line. The next section will show how continuous integration ensures that all tests are run systematically.

6.2.2 Continuous Integration with Jenkins and Travis CI

To ensure all tests for FinMag could be run without slowing down the developer's machines, a separate machine was dedicated to running the tests. To automate the process, the free and open source automation server Jenkins [264] was installed on this machine. Via a webhook Jenkins was configured to automatically download the newest changesets from the online repository. It would compile and install the software from source, then run the tests and generate the manual. A Jenkins slave was added to build a compiled version of the software, when the tests succeeded. The results of the building and testing process could be accessed from the developers on a website served by the Jenkins server. Jenkins ensured that every code change submitted to the repository would be verified by running the complete test suite.

Collberg et al. [11] report that half of the codes they examined could not be made to compile or run. The approach with Jenkins just described does not address this issue completely. For one, Jenkins does not start every job with a clean slate and does not test the installation procedure. The machine was manually provisioned once beforehand with the required dependencies following FinMag's installation instructions. Worse, because Jenkins by default only updates a persistent local copy of the repository with the online changes, build and test artifacts from a previous version of FinMag could affect the assessment of a new one. As will be briefly described in Section 6.3, this lead to the unfortunate situation where errors in generating the manual for FinMag could remain undetected.

Online continuous integration (CI) services use virtualisation to run their user's codes and offer "clean" virtual machines for every build. FinMag's and Fidimag's continuous integration setup

was migrated to the online service CircleCI and when that rescinded its offerings, to Travis CI [206]. For the Travis CI service the developer writes a YAML configuration file that automates the installation of the software on a fresh virtual machine. The configuration file is specific to the Travis CI service but this approach already has the benefit of testing the installation prodecure as well as the software itself. On Travis CI the software testing for FinMag runs for a duration of just under an hour [265]. Section 6.4 will describe how containerisation was used to obtain a setup that is conducive to continuous delivery with Travis CI, but by decoupling the automation of the installation procedure from the specifics of the service also provides a mean of software distribution.

6.3 Software Documentation with Jupyter Notebooks

Early in the work, a documentation generator named Sphinx was used to generate a manual for users of FinMag [266]. The manual was successful in getting new users to familiarise with the software. This came at the price of a small software creation cost but a significant ongoing maintenance cost. The manual naturally showed code examples of the usage of FinMag. When the examples were not executed or tested, they went out of sync with regards to FinMag's rapidly evolving programming interface. Makefiles were used to generate figures for the manual. The make rule for a figure would have only the simulation script producing it listed as a prerequisite. Changes to the corresponding simulation script would then mark the target figure for an update. However changes to FinMag itself would not register with the make rule. This means that the process of documentation generation could complete successfully even though an error could have been introduced since the last update of the figure. This could happen on developer machines as well as on Jenkins, which also kept a persistent local copy of the repository.

Settings were introduced to force the recreation of the manual from scratch for every build along with tests that report on the successful creation of the manual. However, building the manual costs time that made iterative testing on developer machines not practical anymore. To sustain the documentation effort, additional infrastructure was dedicated to building the manual separately from testing. Using Jenkins this was achieved with separate pipelines for compiling FinMag, testing, and building the documentation.

There are two problems with the approach outline in the last paragraph. First, the testing of the documentation only indicates if the documentation was built successfully. This requires certain files and images to be generated by the example code for inclusion in the manual. It doesn't guarantee that the codes execute correctly and produce sensible output and plots, not without introducing more tests. Second, the needed effort in building and maintaining this infrastructure binds resources that are not allocated to the development of the research software. To avoid these problems, the use of Sphinx was abandoned in favour of a simpler setup using Jupyter notebooks [267] for Fidimag. Table 6.4 shows the amount of included documentation for Fidimag, FinMag, Nmag, and OOMMF.

The Jupyter project exists to develop open-source software, open-standards, and services for interactive computing across many programming languages. The Jupyter notebook software is an interactive computational environment with tools for parallel computing, delivered as a web-based text editor with good support for text, code, typeset mathematical expressions and inline plots. It

	Documentation
Fidimag	190 page manual incorporating 22 Jupyter notebooks
FinMag	60 page manual (deprecated) and 56 Jupyter notebooks
Nmag 0.2.1	189 page manual
OOMMF 2.0 alpha 2	255 page manual

Table 6.4: Documentation included with Fidimag, FinMag, Nmag, and OOMMF. The Fidimag manual, by incorporating Jupyter notebooks, also includes the standard output of its contained programs, somewhat inflating its page count.

has been positioned as a publishing format for reproducible computational workflows [203]. The use of Jupyter notebooks for disseminating a user manual has made the use of a documentation generator for FinMag obsolete, because Jupyter notebooks can be recomputed on-demand, and exported to PDF or HTML. Then only the issue of testing the notebooks remains.

Nbval is a Python test runner plugin for validating Jupyter notebooks that has been created by Cortès-Ortuño et al. [205] during their work on Fidimag. The intended purpose of the tests is to determine whether the execution of the code stored in the notebook matches its stored outputs. It also tests that the notebooks are running without errors. The testing of the notebooks occurs along with the testing of the codes and ensures they remain up to date. Further, as the code contained in the notebooks is run, it contributes to the software testing of the simulators. Together with the continuous integration service described in the last section, this allows the notebooks to perform the double duty of documenting and testing the simulators at the same time.

6.4 Software Distribution using Docker

Software distribution is the process of delivering software to the end user. The developers of a new research software are commonly also the first end users. It would also be desirable for a research software to be run during the review of a journal or conference submission using it. This is why it needs to be easy to install and run. The bigger hurdle typically consists of the installation procedure. The creators of Nmag obtained a robust installation setup by bundling the source code of the dependencies in the distributed tarball [268]. After downloading the tarball, Nmag could then be compiled and installed from source on a large variety of Linux distributions and releases. The downside is an increased developer time cost of updating the tarball. Alternate means of software distribution were explored and will be described in this section.

When the developers of the research software are also the sole end users, the software distribution can be handled by simply pointing the users to the source code repository. A combination of installation instructions and shell scripts for some Linux distributions and releases help the user through the installation. For continuous integration using an online service like described in Section 6.2.2 the installation procedure has to be automated. To develop the automation, the workflow of the CI service was implemented locally using Vagrant [269] for the automatic creation of virtual machines and Ansible [250] for their provisioning. The Anaconda [270] package manager was then used to compile and install a simulator both for the CI service or during a local installation. A framework for reproducible micromagnetic simulation based on this model is

presented by Vousden et al. [249], to which the scripts to provision the virtual machines and install Fidimag were contributed by the author of this thesis.

The downside of this approach is that the virtual images are comparatively large and thus challenging to distribute over the internet, since they include their own machine configuration and operating system. If Anaconda is used for distribution, the setup using virtual machines is essentially maintained only to support the continuous integration. Docker container images are standalone packages of software that include everything needed to run an application, but not more [31]. If virtualisation works on the level of the machine, containerisation corresponds to operating system level virtualisation. Containers represent a much leaner abstraction than virtual machines.

Listing 6.1 shows the so-called Dockerfile from which the container image for Fidimag is created. The Anaconda scripts were replaced by using Docker for both the continuous integration and software distribution of the two simulators presented in this thesis. Indeed, with Docker installed, an end user can simply issue the command `docker pull fidimag/notebook` to obtain a working installation of Fidimag. With the command `docker run -p 30000:8888 fidimag/notebook` the user is dropped into a Jupyter notebook and can start using Fidimag. Boettiger [271] describes how Docker can address some of the challenges for reproducible research. An example of reproducible research using FinMag will be given in the next section.

We reported in Section 5.6 how the simulations for the research described in "Ground state search, hysteretic behaviour, and reversal mechanism of skyrmionic textures in confined helimagnetic nanostructures" by Beg et al. [6] were run on the high performance computing (HPC) facility Iridis of the University of Southampton. Docker was not a part of the setup. Running Docker commands typically requires admin privileges. The fact that Docker allows directories to be shared between the Docker host and guest containers without limiting the access rights of containers allows users to obtain admin access trivially [272]. In short, there is no built-in way to limit the access rights of users running Docker and this is problematic in multi-user environments like HPC clusters. There are however efforts to safely enable Docker-based workflows on HPC resources and a continued convergence of high performance and cloud computing technologies can only further improve the situation [273].

6.5 Reproducible Science

Section 6.3 showed how Jupyter notebooks could be used for documentation and software testing. In Section 6.4 Docker was described as a practical solution for software distribution. A demonstration of reproducible science using FinMag that uses Docker and Jupyter notebooks will be given in this section.

In "Stable and manipulable Bloch point" Beg et al. [274] show a particle-like state occuring in a FeGe disk with two chiralities introduced by the Dzyaloshinskii-Moriya interaction. The data and codes are accessible in a GitHub repository [275]. If Docker is installed on the user machine, the command `make all` will automatically fetch a FinMag container image and rerun all the micromagnetic simulations used to obtain the results published in the report. Additionally one

```
FROM ubuntu:16.04

RUN apt -y update
RUN apt install -y git python3 python3-pip gcc psutils cmake wget make
RUN apt install -y gfortran libblas-dev liblapack-dev python3-tk sudo fonts-lato
RUN pip3 install cython matplotlib pytest scipy psutil pyvtk ipywidgets -U
RUN pip3 install --no-cache-dir notebook

RUN ln -s /usr/bin/python3 /usr/bin/python

WORKDIR /usr/local
RUN git clone https://github.com/computationalmodelling/fidimag.git
WORKDIR /usr/local/fidimag
# Work with stable release
RUN git checkout tags/v2.9
# Install CVODE and FFTW libraries
WORKDIR /usr/local/fidimag/bin
RUN bash install-fftw.sh
RUN bash install-sundials.sh

ENV PYTHONPATH="/usr/local/fidimag:$PYTHONPATH"
ENV LD_LIBRARY_PATH="/usr/local/fidimag/local/lib:$LD_LIBRARY_PATH"

WORKDIR /usr/local/fidimag
RUN python3 setup.py build_ext --inplace
RUN python3 -c "import matplotlib"
# Headless Matplotlib:
ENV MPLBACKEND=Agg

# Headless Matplotlib:
ENV MPLBACKEND=Agg

# Set threads for OpenMP:
ENV OMP_NUM_THREADS=2
# WORKDIR /io

# User to make Binder happy
ENV NB_USER magnetism
ENV NB_UID 1000
ENV HOME /home/${NB_USER}

RUN adduser --disabled-password \
    --gecos "Default user" \
    --uid ${NB_UID} \
    ${NB_USER}

# Make sure the contents of our repo are in £{HOME}
COPY . ${HOME}
USER root
RUN chown -R ${NB_UID} ${HOME}
USER ${NB_USER}

WORKDIR /home/${USER}/magnetism/doc/ipynb
```

Listing 6.1: Dockerfile for the creation of a container image of Fidimag.

Jupyter notebook exists for each of the central figures included in the report. In such a notebook the code used to generate the plots is shown and can be rerun.

Binder [276] is an online service that hosts computing environments for Jupyter notebooks published in online repositories. Pointing Binder towards the mentioned GitHub repository of Beg triggers the build of a Docker container that includes the code and the Jupyter notebooks from the repository. The Jupyter notebooks can then be opened with a click of the mouse, the plots can be edited and the data subjected to further analysis without needing to install anything on the user machine.

Chapter 7

Conclusion

Chapters 4 and 5 presented two micromagnetic simulators, namely the finite differences software Fidimag and the finite elements software FinMag. A report on Fidimag was published in Ref. [32]. The two simulators model the magnetisation dynamics based on the solution of the Landau-Lifshitz-Gilbert equation or one of its variants that includes spin-torque transfer terms. With the computation of the exchange, anisotropy, the demagnetising field, and the Zeeman field, the common energies affecting the effective field are included in the softwares. Stochastic terms can be added to simulations to model thermal fluctuations.

When a research software project is successful, the developed software enables original research. For this purpose, it must be tailored to the specific needs of a particular research community. The addition of the Dzyaloshinskii-Moriya energy term in FinMag is an example of software responding to such a need. Section 5.2 shows the implementation. Its addition had a multiplicative effect, as it enabled the exploration of novel physics in the research on magnetic skyrmions. Some of this research was reviewed in Section 5.6. We also extended FinMag to compute hysteresis loops for the simulations presented by Carey et al. [248]. Further, with the hexagonal lattice, we added a second geometry type to Fidimag that was used to represent the atomic arrangement of an FCC cobalt layer in the simulations of Cortés-Ortuño et al. [187].

For time integration in FinMag the cvode method from the SUNDIALS suite of numerical integrators is incorporated, as it is known to offer the best performance for the finite element case. The author contributed to an extension of Scipy offering access and a high level interface to the SUNDIALS integrators for Python [277]. The performance of numerical integrators in the finite differences case was studied with Fidimag in Section 4.5, and instead a fourth-order Runge-Kutta method due to Dormand and Prince was found to be the most effective.

The second numerical study was carried out with FinMag and presented in Section 5.5. Leveraging the FEniCS software, it quantified the impact of using different preconditioners and solvers on the performance of the computation of the demagnetising field in the finite elements case. Here, this computation is significantly more expensive than in the finite differences case due to the requirements of the FEM/BEM method. Large performance gains could be obtained by using incomplete LU factorisation to precondition the conjugate gradient method for the solution of the two linear systems involved in computing the magnetic scalar potential. These findings informed

new default settings in the corresponding code in FinMag. The new settings translate to time savings of up to 70 % for simulations of thin magnetic films, which make up the bulk of studied systems in micromagnetics.

The time integration in the finite differences case and the computation of the demagnetising field in the finite elements case were identified as performance bottlenecks by profiling. The code used to drive this profiling and generate the data in the two numerical studies just described is re-usable and was published as a package on the Python Package index [243]. Since its overhead is low, it monitors running FinMag and Fidimag simulations by default.

The two softwares presented in this thesis were developed in front of the backdrop of the scientific credibility crisis outlined in Chapter 1. Contributing to the body of methodology in the field of micromagnetics was not the only goal. It was also the intent to find and assess the software engineering practices suitable to an academic context that increase the quality of the software, and thus, its results.

The perceived "needs" of a software engineering project can be articulated as a set of requirements at different levels of abstraction. It is especially difficult however to anticipate the requirements of research software projects, since by definition they operate at the edge of scientific knowledge. Section 2.1.3 reviewed agile methods. The agile principles include continuous delivery of valuable software in short periods of time, making use of change, technical excellence and good design, and the self-organisation of the teams during planning and implementation.

Chapter 6 reports on the practices explored during the development of Fidimag and FinMag. We find that the principles inherent to test-driven development (not necessarily test-first development) are beneficial in the research scenario. Following test-driven development, tests are produced as a natural consequence of programming. Section 6.2.1 explained how test-driven development affected the software design positively. If agile methods tend to underemphasize requirements engineering and software design in favour of iteration and change, test-driven development offers a way to pay the required attention to design. Rigorous testing also undeniably increases the code quality. In computational micromagnetics, the standard problems provide a way to communicate over verification. Two of them were reviewed in Sections 4.4 and 5.4. The creators of Nmag note that embedding simulation into existing programming language provides unrivaled flexibility [268]. Here, they allow the solution to standard problems as could be performed by a user to easily be part of the automatic software verification process. This is another advantage over codes primarily designed around graphical user interfaces such as OOMMF. On the other hand, graphical user interfaces do have the advantage of being somewhat self-documenting, although Section 6.3 showed how Jupyter Notebooks could be made to kill two birds with one stone: software documentation and testing.

For maximum benefit the tests need to be executed often, and in regular intervals. Continuous integration presents this idea followed to its logical conclusion, when applied together with distributed version control: Automatic verification of the software for every change in code. Continuous integration was evaluated using an open source automation server and two free, hosted continuous integration services. Section 6.2.2 reported that CI could be implemented with all three, but the online offerings came without an increased demand for software maintenance (of the automation). This is mainly due to the fact that the online offerings purpose-built their software around virtualisation and later containerisation.

When the software aquires its first dedicated users, the resources available for software distribution and maintenance in an academic setting are low. The report "Virtual Micromagnetics: A Framework for Accessible and Reproducible Micromagnetic Simulation" by Vousden et al. [249], to which the author contributed the software provisioning scripts, shows that virtual environments allow easy access to simulation packages that are often difficult to compile and install. By offering a slimmer abstraction than virtual machines, Docker containers make this process especially painless, as was shown in Section 6.4. Publications with the codes distributed in the manner described in Section 6.5, using Docker and Jupyter notebooks, would improve upon the discouraging findings of Collberg et al. [11] regarding the large fraction (half) of codes that could not be made to compile or run, out of the fraction of scientific publications that include the used code at all. No single solution to the credibility crisis will suffice, but a higher level of code quality and reproducibility certainly are a necessary part of a solution using the definition of Begley and Ioannidis [12].

In Chapter 1 we argued that research software engineering stood to gain from adopting practices of modern software engineering. We summarised above the impact of the studied techniques on the software design, implementation, testing, maintenance, and distribution of the research softwares presented in this thesis. When reading software engineering literature, estimating how much benefit could be obtained by applying the contained techniques and how much cost would be incurred was a particular challenge. Indeed, based on a multi-vocal[1] review of software engineering literature, Garousi names including a cost-benefit analysis of techniques proposed in papers one of the most important areas of improvement [278]. They argue this makes it more difficult to "convince" practitioners to adopt the techniques, thus leading to poor relevance. The issue of practical relevance of research however is not specific to software engineering but widely discussed as the rigour-relevance dilemma [279]. Certainly, rigorous results can be obtained by an explanatory mode of science that focuses on describing, explaining, and predicting; but it is also reductionistic and difficult to apply. Action research [280] or design science [281] claim to offer a way out of the dilemma with similar approaches that are equally scientific as practically relevant. Action research aims at generating knowledge and improving situations at the same time. Contrasted with the explanatory mode of science, design science focuses more on diagnosing, designing, and improving. Design science produces design knowledge, that is knowledge that can be used to design solutions to a class of problems [281]. Research software engineering, when complimenting a scientific pursuit, is then also a way to capture this design knowledge and improve situations. Better yet, it should give both the explanatory and design approaches their proper place resulting in a multi-approach research agenda [282].

Finally, further inspiration may come from the social sciences because as Ko notes, "many of the problems in software engineering are not technical problems, but people problems" [283]. Predictive models and techniques such as social network analysis are being applied to a host of software development problems [284, 285, 286].

[1] A multi-vocal literature review includes both peer-reviewed and gray literature.

References

[1] Christian Pfleiderer. Magnetic order: Surfaces get hairy. *Nature Physics*, 7(9):673–674, September 2011.

[2] Andrei Slavin. Microwave sources: Spin-torque oscillators get in phase. *Nat Nano*, 4(8):479–480, Aug 2009.

[3] Maximilian Albert, Marijan Beg, Dmitri Chernyshenko, Marc-Antonio Bisotti, Rebecca L Carey, Hans Fangohr, and Peter J Metaxas. Frequency-based nanoparticle sensing over large field ranges using the ferromagnetic resonances of a magnetic nanodisc. *Nanotechnology*, 27(45):455502, 2016.

[4] Micromagnetic Modeling Activity Group. mumag micromagnetics website. `http://www.ctcms.nist.gov/~rdm/mumag.org.html`. Accessed: 2017-09-25.

[5] University of Southampton. Degree courses. `https://www.southampton.ac.uk/courses.page`. Accessed: 2018-11-11.

[6] M. Beg, R. Carey, W. Wang, D. Cortés-Ortuño, M. Vousden, M.-A. Bisotti, M. Albert, D. Chernyshenko, O. Hovorka, R. L. Stamps, and H. Fangohr. Ground state search, hysteretic behaviour, and reversal mechanism of skyrmionic textures in confined helimagnetic nanostructures. *Sci. Rep.*, 5(October):17137, 2015.

[7] Joachim Schöberl. Netgen an advancing front 2d/3d-mesh generator based on abstract rules. *Computing and Visualization in Science*, 1(1):41–52, Jul 1997.

[8] Victoria C. Stodden, Friedrich Leisch, and Roger D. Peng. *Implementing Reproducible Research*. CRC Press, 2014.

[9] Tom Feilden. *Most scientists 'can't replicate studies by their peers'*, February 2017 (accessed May 5, 2018). http://www.bbc.com/news/science-environment-39054778.

[10] Monya Baker. 1,500 scientists lift the lid on reproducibility. *Nature*, 533(7604):452–454, May 2016.

[11] Christian Collberg and Todd A. Proebsting. Repeatability in computer systems research. *Commun. ACM*, 59(3):62–69, February 2016.

[12] C. Glenn Begley and John P.A. Ioannidis. Reproducibility in science. *Circulation Research*, 116(1):116–126, 2015.

[13] Simon Hettrick. Uk research software survey 2014, 2014.

[14] Marvin Gechman. *Project Management of Large Software-Intensive Systems*. CRC Press, 2019.

[15] J. Clarke, J. J. Dolado, M. Harman, R. Hierons, B. Jones, M. Lumkin, B. Mitchell, S. Mancoridis, K. Rees, M. Roper, and M. Shepperd. Reformulating software engineering as a search problem. *IEE Proceedings - Software*, 150(3):161–175, June 2003.

[16] M. L. Vanter, S. Faulk, S. Squires, E. Loh, and L. G. Votta. Scientific computing's productivity gridlock: How software engineering can help. *Computing in Science & Engineering*, 11:30–39, 11 2009.

[17] L. Hatton and A. Roberts. How accurate is scientific software? *IEEE Transactions on Software Engineering*, 20(10):785–797, Oct 1994.

[18] ISO/IEC 25010. ISO/IEC 25010:2011, systems and software engineering systems and software quality requirements and evaluation (SQuaRE) system and software quality models. 2011.

[19] A. Johanson and W. Hasselbring. Software engineering for computational science: Past, present, future. *Computing in Science Engineering*, 20(2):90–109, Mar 2018.

[20] D. Hook and D. Kelly. Testing for trustworthiness in scientific software. In *2009 ICSE Workshop on Software Engineering for Computational Science and Engineering*, pages 59–64, May 2009.

[21] T. Vogel, S. Druskat, M. Scheidgen, C. Draxl, and L. Grunske. Challenges for verifying and validating scientific software in computational materials science. In *2019 IEEE/ACM 14th International Workshop on Software Engineering for Science (SE4Science)*, pages 25–32, May 2019.

[22] H. J. Richter. The transition from longitudinal to perpendicular recording. *Journal of Physics D: Applied Physics*, 40(9):R149, 2007.

[23] M. J. Donahue, D. G. Porter, National Institute of Standards, and Technology (U.S.). *OOMMF user's guide [microform] / M.J. Donahue, D.G. Porter* . U.S. Dept. of Commerce, Technology Administration, National Institute of Standards and Technology, Gaithersburg, MD :, version 1.0. edition, 1999.

[24] Arne Vansteenkiste and Ben Van de Wiele. Mumax: a new high-performance micromagnetic simulation tool. *JOURNAL OF MAGNETISM AND MAGNETIC MATERIALS*, 323(21):2585–2591, 2011.

[25] MicroMagnum. http://micromagnum.informatik.uni-hamburg.de. `http://micromagnum.informatik.uni-hamburg.de`. Accessed: 2017-10-12.

[26] Gunnar Selke. *Design and Development of a GPU-Accelerated Micromagnetic Simulator*. PhD thesis, Universität Hamburg, Von-Melle-Park 3, 20146 Hamburg, 2013.

[27] MicroMagus. Software package for micromagnetic simulations. `http://www.micromagus.de`. Accessed: 2018-04-12.

[28] T. Fischbacher, M. Franchin, G. Bordignon, and H. Fangohr. A systematic approach to multiphysics extensions of finite-element-based micromagnetic simulations: Nmag. *Magnetics, IEEE Transactions on*, 43(6):2896 –2898, june 2007.

[29] Werner Scholz, Josef Fidler, Thomas Schrefl, Dieter Suess, Rok Dittrich, Hermann Forster, and Vassilios Tsiantos. Scalable parallel micromagnetic solvers for magnetic nanostructures. *Computational Materials Science*, 28(2):366 – 383, 2003. Proceedings of the Symposium on Software Development for Process and Materials Design.

[30] The FEniCS Project. `http://www.fenicsproject.org/`.

[31] Dirk Merkel. Docker: Lightweight linux containers for consistent development and deployment. *Linux J.*, 2014(239), March 2014.

[32] Marc-Antonio Bisotti, David Cortés-Ortuño, Ryan A Pepper, Weiwei Wang, Marijan Beg, Thomas Kluyver, and Hans Fangohr. Fidimag – a finite difference atomistic and micromagnetic simulation package. *Journal of Open Research Software*, 6(1):22, 2018.

[33] Stephen Blundell. *Magnetism in Condensed Matter*. Oxford University Press, 2001.

[34] Amikam Aharoni. *Introduction to the Theory of Ferromagnetism*. Oxford University Press, 2000.

[35] I. Dzyaloshinskii. Thermodynamic theory of weak ferromagnetism in antiferromagnetic substances. *J. Phys. Chem. Solids*, 4:241–255, 1958.

[36] Albert Fert, Nicolas Reyren, and Vincent Cros. Magnetic skyrmions: Advances in physics and potential applications. 2, 06 2017.

[37] D. R. Fredkin and Koehler T. R. Hybrid method for computing demagnetizing fields. *IEEE Trans. Magn.*, 26:415–417, 1990.

[38] Peter Naur and Brian Randell, editors. *Software Engineering: Report of a Conference Sponsored by the NATO Science Committee, Garmisch, Germany, 7-11 Oct. 1968, Brussels, Scientific Affairs Division, NATO*. 1969.

[39] J. N. Buxton and B. Randell, editors. *Software Engineering Techniques: Report of a Conference Sponsored by the NATO Science Committee, Rome, Italy, 27-31 Oct. 1969, Brussels, Scientific Affairs Division, NATO*. 1970.

[40] B. Randell. Software engineering in 1968. In *Proceedings of the 4th International Conference on Software Engineering*, ICSE '79, pages 1–10, Piscataway, NJ, USA, 1979. IEEE Press.

[41] F. DeRemer and H. H. Kron. Programming-in-the-large versus programming-in-the-small. *IEEE Transactions on Software Engineering*, SE-2(2):80–86, June 1976.

[42] Kyle Eischen. Software development: An outsider's view. *Computer*, 35(5):36–44, May 2002.

[43] Kent Beck, Mike Beedle, Arie van Bennekum, Alistair Cockburn, Ward Cunningham, Martin Fowler, James Grenning, Jim Highsmith, Andrew Hunt, Ron Jeffries, Jon Kern, Brian Marick, Robert C. Martin, Steve Mellor, Ken Schwaber, Jeff Sutherland, and Dave Thomas. Manifesto for agile software development. `http://www.agilemanifesto.org`, 2001. Accessed: 2020-01-19.

[44] Steven R. Rakitin. Manifesto elicits cynicism. *IEEE Computer*, 34(12):4, 2001.

[45] Pierre Bourque and Richard E. Fairley, editors. *SWEBOK: Guide to the Software Engineering Body of Knowledge*. IEEE Computer Society, Los Alamitos, CA, version 3.0 edition, 2014.

[46] Rick Adcock, Edward Alef, Bruce Amato, Mark Ardis, Larry Bernstein, Barry Boehm, Pierre Bourque, John Brackett, Murray Cantor, Lillian Cassel, Robert Edson, Richard Fairley, Dennis Frailey, Gary Hafen, Thomas Hilburn, Greg Hislop, David Klappholz, Philippe Kruchten, Phil Laplante, Qiaoyun (Liz) Li, Scott Lucero, John McDermid, James McDonald, Ernest McDuffie, Bret Michael, William Milam, Ken Nidiffer, Art Pyster, Paul Robitaille, Mary Shaw, Sarah Sheard, Robert Suritis, Massood Towhidnejad, Richard Thayer, J. Barrie Thompson, Guilherme Travassos, Richard Turner, Joseph Urban, Ricardo Valerdi, Osmo Vikman, David Weiss, and Mary Jane Willshire. Curriculum guidelines for graduate degree programs in software engineering. Technical report, New York, NY, USA, 2009.

[47] Walker W. Royce. Managing the development of large software systems: concepts and techniques. *Proc. IEEE WESTCON, Los Angeles*, pages 1–9, August 1970. Reprinted in *Proceedings* of the Ninth International Conference on Software Engineering, March 1987, pp. 328–338.

[48] T. E. Bell and T. A. Thayer. Software requirements: Are they really a problem? In *Proceedings of the 2Nd International Conference on Software Engineering*, ICSE '76, pages 61–68, Los Alamitos, CA, USA, 1976. IEEE Computer Society Press.

[49] Herbert D. Benington. Production of large computer programs. *IEEE Ann. Hist. Comput.*, 5(4):350–361, October 1983.

[50] B. W. Boehm. Guidelines for verifying and validating software requirements and design specifications. In P. A. Samet, editor, *Euro IFIP 79*, pages 711–719. North Holland, 1979.

[51] Daniel Angermeier et al. *V-Modell XT. Das deutsche Referenzmodell für Systementwicklungsprojekte*. Verein zur Weiterentwicklung des V-Modell XT e.V., Version 2.3.

[52] Kent Beck and Cynthia Andres. *Extreme Programming Explained: Embrace Change (2nd Edition)*. Addison-Wesley Professional, 2004.

[53] LLC VersionOne. 12th annual state of agile report. `https://explore.versionone.com/state-of-agile/versionone-12th-annual-state-of-agile-report`, 2017. Accessed: 2020-01-19.

[54] Ayelt Komus and Moritz Kuberg. Status quo agile – third study on success and forms of usage of agile methods. Hochschule Koblenz University of Applied Sciences, 2017.

[55] Ian Sommerville. *Software Engineering*. Addison-Wesley Publishing Company, USA, 9th edition, 2010.

[56] John Ousterhout. *A Philosophy of Software Design*. Yaknyam, 2018.

[57] Martin Fowler. *Patterns of Enterprise Application Architecture*. Addison-Wesley Longman Publishing Co., Inc., USA, 2002.

[58] António Miguel Rosado da Cruz and Sara Paiva. *Modern software engineering methodologies for mobile and cloud environments*. 01 2016.

[59] Irakli Nadareishvili, Ronnie Mitra, Matt McLarty, and Mike Amundsen. *Microservice Architecture: Aligning Principles, Practices, and Culture*. OReilly Media, Inc., 1st edition, 2016.

[60] P. D. Francesco, I. Malavolta, and P. Lago. Research on architecting microservices: Trends, focus, and potential for industrial adoption. In *2017 IEEE International Conference on Software Architecture (ICSA)*, pages 21–30, April 2017.

[61] Antony Tang, Paris Avgeriou, Anton Jansen, Rafael Capilla, and Muhammad Ali Babar. A comparative study of architecture knowledge management tools. *Journal of Systems and Software*, 83(3):352 – 370, 2010.

[62] Gilberto Borrego, Alberto L. Morn, Ramn R. Palacio, Aurora Vizcano, and Flix O. Garca. Towards a reduction in architectural knowledge vaporization during agile global software development. *Information and Software Technology*, 112:68 – 82, 2019.

[63] Edsger W. Dijkstra. ACM turing award lectures. chapter The Humble Programmer. ACM, New York, NY, USA, 2007.

[64] Martin Pol, Tim Koomen, and Andreas Spillner. *Management und Optimierung des Testprozesses - ein praktischer Leitfaden fuer erfolgreiches Software-Testen mit TPI und TMap (2. Aufl.)*. dpunkt.verlag, 2002.

[65] Martin Fowler. Continuous integration. `https://www.martinfowler.com/articles/continuousIntegration.html`, 5 2006. Accessed: 2017-10-04.

[66] Paul Duvall, Stephen M. Matyas, and Andrew Glover. *Continuous Integration: Improving Software Quality and Reducing Risk (The Addison-Wesley Signature Series)*. Addison-Wesley Professional, 2007.

[67] Jez Humble and David Farley. *Continuous Delivery: Reliable Software Releases Through Build, Test, and Deployment Automation*. Addison-Wesley Professional, 1st edition, 2010.

[68] Charles Kittel. *Introduction to Solid State Physics*. John Wiley & Sons, Inc., 8th edition, 2004.

[69] F. Bloch. Zur Theorie des Austauschproblems und der Remanenzerscheinung der Ferromagnetika. *Zeitschrift für Physik*, 74(5):295–335, May 1932.

[70] L. D. Landau and E. M. Lifshitz. On the theory of the dispersion of magnetic permeability in ferromagnetic bodies. *Phys. Z. Sowjetunion*, 8:153–169, 1935.

[71] W. F. Jr. Brown. *Micromagnetics*. Interscience Publishers, Inc., New York, 1963.

[72] Maximilian Albert, Matteo Franchin, Thomas Fischbacher, Guido Meier, and H. Fangohr. Domain wall motion in perpendicular anisotropy nanowires with edge roughness. *Journal of physics. Condensed matter : an Institute of Physics journal*, 24:024219, 01 2012.

[73] T. L. Gilbert. A phenomenological theory of damping in ferromagnetic materials. *IEEE Trans. Magn.*, 40(6):3443–3449, 2004.

[74] T. L. Gilbert. A lagrangian formulation of the gyromagnetic equation of the magnetic field. *Phys. Rev. Lett.*, 100:1243, 1955.

[75] Benjamin Krüger. *Current-Driven Magnetization Dynamics: Analytical Modeling and Numerical Simulation*. PhD thesis, Universität Hamburg, Von-Melle-Park 3, 20146 Hamburg, 2011.

[76] Claas Abert. *Discrete Mathematical Concepts in Micromagnetic Computations*. PhD thesis, Universität Hamburg, Von-Melle-Park 3, 20146 Hamburg, 2013.

[77] B. D. Cullity and C. D. Graham. *Introduction to Magnetic Materials*. Wiley-IEEE Press, 2 edition, 2008.

[78] James Deak. Finite temperature micromagnetics and magnetic measurements of submicron patterned permalloy thin films. *Journal of Applied Physics*, 93:6814–6816, 05 2003.

[79] W. Heisenberg. Mehrkörperproblem und Resonanz in der Quantenmechanik. *Z. Phys. A*, 38:411, 1926.

[80] P. A. M. Dirac. On the theory of quantum mechanics. *Proc. R. Soc. A*, 112:661, 1926.

[81] R. C. O'Handley. *Modern Magnetic Materials*. John Wiley & Sons, Inc., 2000.

[82] J. D. Jackson. *Classical electrodynamics*. John Wiley & Sons, 1999.

[83] Andreas Knittel. *Micromagnetic simulations of three dimensional core-shell nanostructures*. PhD thesis, University of Southampton, Faculty of Engineering and the Environment, 2011.

[84] I. Dzyaloshinskii. The theory of helicoidal structures in antiferromagnets. *Soviet Physics JETP*, 20(1), 1965.

[85] Tôru Moriya. Anisotropic superexchange interaction and weak ferromagnetism. *Phys. Rev.*, 120:91–98, Oct 1960.

[86] P. Bak and Mh. Jensen. Theory of helical magnetic structures and phase transitions in mnsi and fege. *Journal of Physics C: Solid State Physics*, 881, 1980.

[87] A. Bogdanov and A. Hubert. Thermodynamically Stable Magnetic Vortex States in Magnetic Crystals. *Journal of Magnetism and Magnetic Materials*, 138(3):255–269, 1994.

[88] A. Bogdanov and U. Röß ler. Chiral Symmetry Breaking in Magnetic Thin Films and Multilayers. *Physical Review Letters*, 87(3):037203, 2001.

[89] U. Roessler, A. Bogdanov, and C. Pfleiderer. Spontaneous Skyrmion Ground States in Magnetic Metals. *Nature*, 442(7104):797–801, 2006.

[90] M. Bode, M. Heide, K. von Bergmann, P. Ferriani, S. Heinze, G. Bihlmayer, A. Kubetzka, O. Pietzsch, S. Blügel, and R. Wiesendanger. Chiral Magnetic Order at Surfaces Driven by Inversion Asymmetry. *Nature*, 447(7141):190–3, 2007.

[91] S. Mühlbauer, B. Binz, F. Jonietz, C. Pfleiderer, A. Rosch, A. Neubauer, R. Georgii, and P. Böni. Skyrmion Lattice in a Chiral Magnet. *Science*, 323(5916):915–9, 2009.

[92] X. Z. Yu, Y. Onose, N. Kanazawa, J. H. Park, J. H. Han, Y. Matsui, N. Nagaosa, and Y. Tokura. Real-Space Observation of a Two-Dimensional Skyrmion Crystal. *Nature*, 465(7300):901–4, 2010.

[93] Stefan Heinze, Kirsten von Bergmann, Matthias Menzel, Jens Brede, André Kubetzka, Roland Wiesendanger, Gustav Bihlmayer, and Stefan Blügel. Spontaneous Atomic-Scale Magnetic Skyrmion Lattice in Two Dimensions. *Nature Physics*, 7(9):713–718, July 2011.

[94] X. Z. Yu, N. Kanazawa, Y. Onose, K. Kimoto, W. Z. Zhang, S. Ishiwata, Y. Matsui, and Y. Tokura. Near Room-Temperature Formation of a Skyrmion Crystal in Thin-Films of the Helimagnet FeGe. *Nature materials*, 10(February):1–4, 2011.

[95] Haifeng Du, Wie Ning, Mingliang Tian, and Yuheng Zhang. Magnetic vortex with skyrmionic core in a thin nanodisk of chiral magnets. *EPL (Europhysics Letters)*, 101(3):37001, 2013.

[96] S. Rohart and A. Thiaville. Skyrmion Confinement in Ultrathin Film Nanostructures in the Presence of Dzyaloshinskii-Moriya Interaction. *Physical Review B*, 88(18):184422, November 2013.

[97] Karin Everschor, Markus Garst, Benedikt Binz, Florian Jonietz, Sebastian Mühlbauer, Christian Pfleiderer, and Achim Rosch. Rotating Skyrmion Lattices by Spin Torques and Field or Temperature Gradients. *Physical Review B*, 86(5):054432, August 2012.

[98] André Thiaville, Stanislas Rohart, Émilie Jué, Vincent Cros, and Albert Fert. Dynamics of Dzyaloshinskii Domain Walls in Ultrathin Magnetic Films. *EPL (Europhysics Letters)*, 100(5):57002, 2012.

[99] Satoru Emori, Uwe Bauer, Sung-Min Ahn, Eduardo Martinez, and Geoffrey S. D. Beach. Current-Driven Dynamics of Chiral Ferromagnetic Domain Walls. *Nature materials*, 12(7):611–6, July 2013.

[100] Niklas Romming, Christian Hanneken, Matthias Menzel, Jessica E Bickel, Boris Wolter, Kirsten von Bergmann, André Kubetzka, and Roland Wiesendanger. Writing and Deleting Single Magnetic Skyrmions. *Science (New York, N.Y.)*, 341(6146):636–9, August 2013.

[101] J. Sampaio, V. Cros, S. Rohart, A. Thiaville, and A. Fert. Nucleation, Stability and Current-Induced Motion of Isolated Magnetic Skyrmions in Nanostructures. *Nature nanotechnology*, 8(11):839–44, November 2013.

[102] Albert Fert, Vincent Cros, and João Sampaio. Skyrmions on the Track. *Nature nanotechnology*, 8(3):152–6, March 2013.

[103] F. Jonietz, S. Mühlbauer, C. Pfleiderer, A. Neubauer, W. Münzer, A. Bauer, T. Adams, R. Georgii, P. Böni, R. A. Duine, K. Everschor, M. Garst, and A. Rosch. Spin Transfer Torques in MnSi at Ultralow Current Densities. *Science (New York, N.Y.)*, 330(6011):1648–51, December 2010.

[104] X. Z. Yu, N. Kanazawa, W. Z. Zhang, T. Nagai, T. Hara, K. Kimoto, Y. Matsui, Y. Onose, and Y. Tokura. Skyrmion Flow Near Room Temperature in an Ultralow Current Density. *Nature communications*, 3:988, January 2012.

[105] William Fuller Brown. Thermal fluctuations of a single-domain particle. *Phys. Rev.*, 130:1677–1686, Jun 1963.

[106] C. W. Gardiner. *Handbook of stochastic methods for physics, chemistry and the natural sciences*, volume 13 of *Springer Series in Synergetics*. Springer-Verlag, Berlin, third edition, 2004.

[107] M. N. Baibich, J. M. Broto, A. Fert, F. Nguyen Van Dau, F. Petroff, P. Etienne, G. Creuzet, A. Friederich, and J. Chazelas. Giant Magnetoresistance of (001)Fe/(001)Cr Magnetic Superlattices. *Phys. Rev. Lett.*, 61:2472–2475, Nov 1988.

[108] G. Binasch, P. Grünberg, F. Saurenbach, and W. Zinn. Enhanced magnetoresistance in layered magnetic structures with antiferromagnetic interlayer exchange. *Phys. Rev. B*, 39:4828–4830, Mar 1989.

[109] Mark Johnson and R. H. Silsbee. Interfacial charge-spin coupling: Injection and detection of spin magnetization in metals. *Phys. Rev. Lett.*, 55:1790–1793, Oct 1985.

[110] Gary A. Prinz. Magnetoelectronics. *Science*, 282(5394):1660–1663, 1998.

[111] Matteo Franchin. *Multiphysics simulations of magnetic nanostructures*. PhD thesis, University of Southampton, Faculty of Engineering, Science and Mathematics, 2009.

[112] J. Daughton, J. Brown, E. Chen, R. Beech, A. Pohm, and W. Kude. Magnetic field sensors using gmr multilayer. *Magnetics, IEEE Transactions on*, 30(6):4608 –4610, nov 1994.

[113] Multilayer magnetoresistive sensor. United States Patent 5.452.163, 1995.

[114] L. Berger. Low field magnetoresistance and domain drag in ferromagnets. *Journal of Applied Physics*, 49(3):2156 –2161, 1978.

[115] L. Berger. Exchange interaction between ferromagnetic domain wall and electric current in very thin metallic films. *Journal of Applied Physics*, 55(6):1954–1956, 1984.

[116] P. P. Freitas and L. Berger. Observation of s-d exchange force between domain walls and electric current in very thin Permalloy films. *Journal of Applied Physics*, 57(4):1266–1269, 1985.

[117] C.-Y. Hung and L. Berger. Exchange forces between domain wall and electric current in permalloy films of variable thickness. *Journal of Applied Physics*, 63(8):4276–4278, 1988.

[118] J. C. Slonczewski. Conductance and exchange coupling of two ferromagnets separated by a tunneling barrier. *Phys. Rev. B*, 39:6995–7002, Apr 1989.

[119] J.C. Slonczewski. Current-driven excitation of magnetic multilayers. *Journal of Magnetism and Magnetic Materials*, 159(12):L1 – L7, 1996.

[120] L. Berger. Emission of spin waves by a magnetic multilayer traversed by a current. *Phys. Rev. B*, 54:9353–9358, Oct 1996.

[121] Sadamichi Maekawa. *Concepts in Spin Electronics*. Oxford University Press, 2010.

[122] Mark Johnson. The bipolar spin transistor. *Journal of Magnetism and Magnetic Materials*, 140144, Part 1(0):21 – 24, 1995. International Conference on Magnetism.

[123] S. Urazhdin, Norman O. Birge, W. P. Pratt, and J. Bass. Current-Driven Magnetic Excitations in Permalloy-Based Multilayer Nanopillars. *Phys. Rev. Lett.*, 91:146803, Oct 2003.

[124] A. Fert, V. Cros, J.-M. George, J. Grollier, H. Jaffrs, A. Hamzic, A. Vaurs, G. Faini, J. Ben Youssef, and H. Le Gall. Magnetization reversal by injection and transfer of spin: experiments and theory. *Journal of Magnetism and Magnetic Materials*, 272276, Part 3(0):1706 – 1711, 2004. Proceedings of the International Conference on Magnetism (ICM 2003).

[125] W. H. Rippard, M. R. Pufall, S. Kaka, S. E. Russek, and T. J. Silva. Direct-Current Induced Dynamics in $Co_{90}Fe_{10}/Ni_{80}Fe_{20}$ Point Contacts. *Phys. Rev. Lett.*, 92:027201, Jan 2004.

[126] M. Tsoi, A. G. M. Jansen, J. Bass, W.-C. Chiang, V. Tsoi, and P. Wyder. Generation and detection of phase-coherent current-driven magnons in magnetic multilayers. *Nature*, 406(6791):46–48, Jul 2000.

[127] S. I. Kiselev, J. C. Sankey, I. N. Krivorotov, N. C. Emley, R. J. Schoelkopf, R. A. Buhrman, and D. C. Ralph. Microwave oscillations of a nanomagnet driven by a spin-polarized current. *Nature*, 425(6956):380–383, Sep 2003.

[128] J.C Slonczewski. Currents and torques in metallic magnetic multilayers. *Journal of Magnetism and Magnetic Materials*, 247(3):324 – 338, 2002.

[129] Jiang Xiao, A. Zangwill, and M. D. Stiles. Boltzmann test of Slonczewski's theory of spin-transfer torque. *Phys. Rev. B*, 70:172405, Nov 2004.

[130] J. Z. Sun. Spin-current interaction with a monodomain magnetic body: A model study. *Phys. Rev. B*, 62:570–578, Jul 2000.

[131] J. Z. Sun. Spin angular momentum transfer in current-perpendicular nanomagnetic junctions. *IBM Journal of Research and Development*, 50(1):81 –100, jan. 2006.

[132] Vladislav E. Demidov, Sergei Urazhdin, Vasyl Tiberkevich, Andrei Slavin, and Sergej O. Demokritov. Control of spin-wave emission from spin-torque nano-oscillators by microwave pumping. *Phys. Rev. B*, 83:060406, Feb 2011.

[133] J. A. Katine, F. J. Albert, and R. A. Buhrman. Current-induced realignment of magnetic domains in nanostructured Cu/Co multilayer pillars. *Applied Physics Letters*, 76(3):354–356, 2000.

[134] I. N. Krivorotov, N. C. Emley, A. G. F. Garcia, J. C. Sankey, S. I. Kiselev, D. C. Ralph, and R. A. Buhrman. Temperature Dependence of Spin-Transfer-Induced Switching of Nanomagnets. *Phys. Rev. Lett.*, 93:166603, Oct 2004.

[135] J. Z. Sun, D. J. Monsma, D. W. Abraham, M. J. Rooks, and R. H. Koch. Batch-fabricated spin-injection magnetic switches. *Applied Physics Letters*, 81(12):2202–2204, 2002.

[136] J.-E. Wegrowe, A. Fábián, Ph. Guittienne, X. Hoffer, D. Kelly, J.-Ph. Ansermet, and E. Olive. Exchange torque and spin transfer between spin polarized current and ferromagnetic layers. *Applied Physics Letters*, 80(20):3775–3777, 2002.

[137] J. A. Katine, F. J. Albert, R. A. Buhrman, E. B. Myers, and D. C. Ralph. Current-Driven Magnetization Reversal and Spin-Wave Excitations in Co /Cu /Co Pillars. *Phys. Rev. Lett.*, 84:3149–3152, Apr 2000.

[138] Shehzaad Kaka, Matthew R. Pufall, William H. Rippard, Thomas J. Silva, Stephen E. Russek, and Jordan A. Katine. Mutual phase-locking of microwave spin torque nano-oscillators. *Nature*, 437(7057):389–392, Sep 2005.

[139] F. B. Mancoff, N. D. Rizzo, B. N. Engel, and S. Tehrani. Phase-locking in double-point-contact spin-transfer devices. *Nature*, 437(7057):393–395, Sep 2005.

[140] V. S. Pribiag, I. N. Krivorotov, G. D. Fuchs, P. M. Braganca, O. Ozatay, J. C. Sankey, D. C. Ralph, and R. A. Buhrman. Magnetic vortex oscillator driven by d.c. spin-polarized current. *Nat Phys*, 3(7):498–503, Jul 2007.

[141] Q. Mistral, M. van Kampen, G. Hrkac, Joo-Von Kim, T. Devolder, P. Crozat, C. Chappert, L. Lagae, and T. Schrefl. Current-Driven Vortex Oscillations in Metallic Nanocontacts. *Phys. Rev. Lett.*, 100:257201, Jun 2008.

[142] M. Quinsat, D. Gusakova, J. F. Sierra, J. P. Michel, D. Houssameddine, B. Delaet, M.-C. Cyrille, U. Ebels, B. Dieny, L. D. Buda-Prejbeanu, J. A. Katine, D. Mauri, A. Zeltser, M. Prigent, J.-C. Nallatamby, and R. Sommet. Amplitude and phase noise of magnetic tunnel junction oscillators. *Applied Physics Letters*, 97(18):182507, 2010.

[143] J. Grollier, V. Cros, and A. Fert. Synchronization of spin-transfer oscillators driven by stimulated microwave currents. *Phys. Rev. B*, 73:060409, Feb 2006.

[144] A. Slavin and V. Tiberkevich. Nonlinear auto-oscillator theory of microwave generation by spin-polarized current. *Magnetics, IEEE Transactions on*, 45(4):1875 –1918, april 2009.

[145] Yan Zhou, J. Persson, and Johan Åkerman. Intrinsic phase shift between a spin torque oscillator and an alternating current. *Journal of Applied Physics*, 101(9):09A510, 2007.

[146] X. W. Yu, V. S. Pribiag, Y. Acremann, A. A. Tulapurkar, T. Tyliszczak, K. W. Chou, B. Bräuer, Z.-P. Li, O. J. Lee, P. G. Gowtham, D. C. Ralph, R. A. Buhrman, and J. Stöhr. Images of a Spin-Torque-Driven Magnetic Nano-Oscillator. *Phys. Rev. Lett.*, 106:167202, Apr 2011.

[147] RuotoloA., CrosV., GeorgesB., DussauxA., GrollierJ., DeranlotC., GuillemetR., BouzehouaneK., FusilS., and FertA. Phase-locking of magnetic vortices mediated by antivortices. *Nat Nano*, 4(8):528–532, Aug 2009.

[148] S Sani, J Persson, S M Mohseni, Ye Pogoryelov, P K Muduli, A Eklund, G Malm, M Käll, A Dmitriev, and J Å kerman. Mutually synchronized bottom-up multi-nanocontact spin-torque oscillators. *Nature communications*, 4:2731, January 2013.

[149] Tui Zeng, Yan Zhou, Johan Akerman, P. T. Lai, and Philip W. T. Pong. Linear Phase Tuning of Spin Torque Oscillators Using In-Plane Microwave Fields. *IEEE Transactions on Magnetics*, 50(1):1–4, January 2014.

[150] Ezio Iacocca and Johan Åkerman. Destabilization of serially connected spin-torque oscillators via non-Adlerian dynamics. *Journal of Applied Physics*, 110(10):103910, 2011.

[151] Dong Li, Yan Zhou, Changsong Zhou, and Bambi Hu. Global attractors and the difficulty of synchronizing serial spin-torque oscillators. *Phys. Rev. B*, 82:140407, Oct 2010.

[152] Arkady Pikovsky. Robust synchronization of spin-torque oscillators with an LCR load. *Physical Review E*, 88(3):032812, September 2013.

[153] B. Subash, V. K. Chandrasekar, and M. Lakshmanan. Synchronization of an array of spin torque nano oscillators in periodic applied external magnetic field. *EPL (Europhysics Letters)*, 102(1):17010, April 2013.

[154] V. Puliafito, G. Consolo, L. Lopez-Diaz, and B. Azzerboni. Synchronization of propagating spin-wave modes in a double-contact spin-torque oscillator: A micromagnetic study. *Physica B: Condensed Matter*, 435:44–49, February 2014.

[155] D. V. Berkov. Synchronization of spin-torque-driven nano-oscillators for point contacts on a quasi-one-dimensional nanowire: Micromagnetic simulations. *Physical Review B*, 87(1):014406, January 2013.

[156] A. D. Belanovsky, N. Locatelli, P. N. Skirdkov, F. Abreu Araujo, K. A. Zvezdin, J. Grollier, V. Cros, and A. K. Zvezdin. Numerical and analytical investigation of the synchronization of dipolarly coupled vortex spin-torque nano-oscillators. *Applied Physics Letters*, 103(12):122405, September 2013.

[157] Dong Li, Yan Zhou, Changsong Zhou, and Bambi Hu. Fractional locking of spin-torque oscillator by injected ac current. *Phys. Rev. B*, 83:174424, May 2011.

[158] Dong Li, Yan Zhou, Bambi Hu, and Changsong Zhou. Coupled perturbed heteroclinic cycles: Synchronization and dynamical behaviors of spin-torque oscillators. *Phys. Rev. B*, 84:104414, Sep 2011.

[159] Vidar Thome. From finite differences to finite elements: A short history of numerical analysis of partial differential equations. *Journal of Computational and Applied Mathematics*, 128(1):1 – 54, 2001. Numerical Analysis 2000. Vol. VII: Partial Differential Equations.

[160] Carlos J. Garca-Cervera, Zydrunas Gimbutas, and Weinan E. Accurate numerical methods for micromagnetics simulations with general geometries. *Journal of Computational Physics*, 184(1):37 – 52, 2003.

[161] Josef Fidler and Thomas Schrefl. Micromagnetic modelling - the current state of the art. *Journal of Physics D: Applied Physics*, 33(15):R135, 2000.

[162] Qiushi Chen and A. Konrad. A review of finite element open boundary techniques for static and quasi-static electromagnetic field problems. *IEEE Trans. Magn.*, 33(1):663–676, 1997.

[163] M. R. Hestenes and E. Stiefel. Methods of conjugate gradients for solving linear systems. *Journal of research of the National Bureau of Standards*, 49:409–436, 1952.

[164] Valeria Simoncini and Daniel B. Szyld. Recent computational developments in krylov subspace methods for linear systems. *Numerical Linear Algebra with Applications*, 14(1):1–59, 2007.

[165] A. Meister and C. Vömel. *Numerik linearer Gleichungssysteme*. Vieweg+Teubner Verlag, 2011.

[166] Michele Benzi. Preconditioning techniques for large linear systems: A survey. *Journal of Computational Physics*, 182(2):418 – 477, 2002.

[167] D Suess, V Tsiantos, T Schrefl, J Fidler, W Scholz, H Forster, R Dittrich, and J.J Miles. Time resolved micromagnetics using a preconditioned time integration method. *Journal of Magnetism and Magnetic Materials*, 248(2):298 – 311, 2002.

[168] Lukas Exl, Johann Fischbacher, Alexander Kovacs, Harald Oezelt, Markus Gusenbauer, and Thomas Schrefl. Preconditioned nonlinear conjugate gradient method for micromagnetic energy minimization. *Computer Physics Communications*, 235:179 – 186, 2019.

[169] J. C. Butcher. *The Numerical Analysis of Ordinary Differential Equations: Runge-Kutta and General Linear Methods*. Wiley-Interscience, New York, NY, USA, 1987.

[170] Massimiliano dAquino, Claudio Serpico, and Giovanni Miano. Geometrical integration of landaulifshitzgilbert equation based on the mid-point rule. *Journal of Computational Physics*, 209(2):730 – 753, 2005.

[171] Arbab Rahim, Carlo Ragusa, Bilal Jan, and Omar Khan. A mixed mid-point runge-kutta like scheme for the integration of landau-lifshitz equation. *Journal of Applied Physics*, 115(17), 2014.

[172] T. Fischbacher and H. Fangohr. Continuum multi-physics modeling with scripting languages: the Nsim simulation compiler prototype for classical field theory. *ArXiv e-prints*, July 2009.

[173] José Luis García-Palacios and Francisco J. Lázaro. Langevin-dynamics study of the dynamical properties of small magnetic particles. *Phys. Rev. B*, 58:14937–14958, Dec 1998.

[174] M. dAquino, C. Serpico, G. Coppola, I. D. Mayergoyz, and G. Bertotti. Midpoint numerical technique for stochastic landau-lifshitz-gilbert dynamics. *Journal of Applied Physics*, 99(8):08B905, 2006.

[175] J H Mentink, M V Tretyakov, A Fasolino, M I Katsnelson, and Th Rasing. Stable and fast semi-implicit integration of the stochastic landau–lifshitz equation. *Journal of Physics: Condensed Matter*, 22(17):176001, apr 2010.

[176] Pavel F. Bessarab, Valery M. Uzdin, and Hannes Jónsson. Method for finding mechanism and activation energy of magnetic transitions, applied to skyrmion and antivortex annihilation. *Computer Physics Communications*, 196(Supplement C):335 – 347, 2015.

[177] David Cortés-Ortuño. *Computational Simulations of Complex Chiral Magnetic Structures*. PhD thesis, University of Southampton, Faculty of Engineering and the Environment, 2017.

[178] Graeme Henkelman, Blas P. Uberuaga, and Hannes Jnsson. A climbing image nudged elastic band method for finding saddle points and minimum energy paths. *The Journal of Chemical Physics*, 113(22):9901–9904, 2000.

[179] R Dittrich, T Schrefl, D Suess, W Scholz, H Forster, and J Fidler. A path method for finding energy barriers and minimum energy paths in complex micromagnetic systems. *Journal of Magnetism and Magnetic Materials*, 250:12 – 19, 2002.

[180] Micromagnetic Modeling Activity Group. mumag action plan april 1995. `https://www.ctcms.nist.gov/~rdm/mumag.pln0495.html`. Accessed: 2019-02-10.

[181] Micromagnetic Modeling Activity Group. mumag micromagnetics workshop november 1995. `https://www.ctcms.nist.gov/~rdm/mumag.ws1195.html`. Accessed: 2019-02-10.

[182] Massoud Najafi, Benjamin Krüger, Stellan Bohlens, Matteo Franchin, Hans Fangohr, Antoine Vanhaverbeke, Rolf Allenspach, Markus Bolte, Ulrich Merkt, Daniela Pfannkuche, Dietmar P. F. Mller, and Guido Meier. Proposal for a standard problem for micromagnetic simulations including spin-transfer torque. *Journal of Applied Physics*, 105(11):113914, 2009.

[183] G. Venkat, D. Kumar, M. Franchin, O. Dmytriiev, M. Mruczkiewicz, H. Fangohr, A. Barman, M. Krawczyk, and A. Prabhakar. Proposal for a standard micromagnetic problem: Spin wave dispersion in a magnonic waveguide. *Magnetics, IEEE Transactions on*, 49(1):524 –529, jan. 2013.

[184] David Cortés-Ortuño, Marijan Beg, Vanessa Nehruji, Leoni Breth, Ryan Pepper, Thomas Kluyver, Gary Downing, Thorsten Hesjedal, Peter Hatton, Tom Lancaster, Riccardo Hertel, Ondrej Hovorka, and Hans Fangohr. Proposal for a micromagnetic standard problem for materials with dzyaloshinskii–moriya interaction. *New Journal of Physics*, 20(11):113015, nov 2018.

[185] Weiwei Wang, Maximilian Albert, Marijan Beg, Marc-Antonio Bisotti, Dmitri Chernyshenko, David Cortés-Ortuño, Ian Hawke, and Hans Fangohr. Magnon-driven domain-wall motion with the Dzyaloshinskii-Moriya interaction. *Phys. Rev. Lett.*, 114:087203, Feb 2015.

[186] Weiwei Wang, Mykola Dvornik, Marc-Antonio Bisotti, Dmitri Chernyshenko, Marijan Beg, Maximilian Albert, Arne Vansteenkiste, Bartel V. Waeyenberge, Andriy N. Kuchko, Volodymyr V. Kruglyak, and Hans Fangohr. Phenomenological description of the nonlocal magnetization relaxation in magnonics, spintronics, and domain-wall dynamics. *Phys. Rev. B*, 92:054430, Aug 2015.

[187] D. Cortés-Ortuño, W. Wang, M. Beg, R. A. Pepper, M.-A. Bisotti, R. Carey, M. Vousden, T. Kluyver, O. Hovorka, and H. Fangohr. Thermal stability and topological protection of skyrmions in nanotracks. *Scientific Reports*, 7:4060, June 2017.

[188] Weiwei Wang, Chaoyang Zhang, Ryan Pepper, Congpu Mu, Yan Zhou, and Hans Fangohr. Current-induced instability of domain walls in cylindrical nanowires. *Journal of Physics: Condensed Matter*, 2017.

[189] David Cortés-Ortuño and Hans Fangohr. Thermal stability and topological protection of skyrmions: Supplementary data, November 2016.

[190] Amikam Aharoni. *Introduction to the Theory of Ferromagnetism*, volume 109. Oxford University Press, second edition, 2000.

[191] Soshin Chikazumi. *Physics of Ferromagnetism*, volume 94 of *International Series of Monographs on Physics*. Oxford University Press, second edition, 2009.

[192] Fidimag. Documentation - core equations. `https://fidimag.readthedocs.io/en/latest/core_eqs.html`. Accessed: 2018-02-06.

[193] David Corts-Ortuo and Hans Fangohr. Test system for nudged elastic band method in nanoscale magnetism, Nov 2016.

[194] D. M. Beazley. Scientific Computing with Python. In N. Manset, C. Veillet, and D. Crabtree, editors, *Astronomical Data Analysis Software and Systems IX*, volume 216 of *Astronomical Society of the Pacific Conference Series*, page 49, 2000.

[195] Hans Fangohr. A comparison of C, Matlab, and Python as teaching languages in engineering. In Marian Bubak, Geert Dick van Albada, Peter M. A. Sloot, and Jack Dongarra, editors, *Computational Science - ICCS 2004*, pages 1210–1217, Berlin, Heidelberg, 2004. Springer Berlin Heidelberg.

[196] Hans Fangohr, Maximilian Albert, and Matteo Franchin. Nmag micromagnetic simulation tool: Software engineering lessons learned. In *Proceedings of the International Workshop on Software Engineering for Science*, SE4Science '16, pages 1–7, New York, NY, USA, 2016. ACM.

[197] Marijan Beg, Ryan A. Pepper, and Hans Fangohr. User interfaces for computational science: A domain specific language for oommf embedded in python. *AIP Advances*, 7(5):056025, 2017.

[198] Eric Jones, Travis Oliphant, Pearu Peterson, et al. SciPy: Open source scientific tools for Python. `http://www.scipy.org`, 2001–. Accessed: 2017-01-22.

[199] Marijan Beg, Ryan A. Pepper, and Hans Fangohr. User interfaces for computational science: A domain specific language for oommf embedded in python. *AIP Advances*, 7(5):056025, 2017.

[200] R. F. L. Evans, W. J. Fan, P. Chureemart, T. A. Ostler, M. O. A. Ellis, and R. W. Chantrell. Atomistic spin model simulations of magnetic nanomaterials. *Journal of Physics: Condensed Matter*, 26(10):103202, 2014.

[201] Martin Fowler. *Refactoring: Improving the Design of Existing Code.* Addison-Wesley, Boston, MA, USA, 1999.

[202] Andrew Davison. Automated capture of experiment context for easier reproducibility in computational research. *Computing in Science & Engineering*, 14:48–56, 2012.

[203] Thomas Kluyver, Benjamin Ragan-Kelley, Fernando Pérez, Brian Granger, Matthias Bussonnier, Jonathan Frederic, Kyle Kelley, Jessica Hamrick, Jason Grout, Sylvain Corlay, Paul Ivanov, Damián Avila, Safia Abdalla, Carol Willing, and Jupyter development team [Unknown]. Jupyter notebooks – a publishing format for reproducible computational workflows. 2016.

[204] Fidimag. Documentation. `http://fidimag.readthedocs.io/en/latest/`. Accessed: 2017-10-12.

[205] David Cortes-Ortuno, Oliver Laslett, T. Kluyver, Vidar Fauske, Maximilian Albert, MinRK, Ondrej Hovorka, and Hans Fangohr. nbval: a py.test plugin for validating jupyter notebooks. `https://github.com/computationalmodelling/nbval`, 2014–. Accessed: 2017-09-25.

[206] Travis CI GmbH. computationalmodelling/fidimag - travis ci. `https://travis-ci.org/computationalmodelling/fidimag`. Accessed: 2017-10-04.

[207] Thomas Schrefl. private communication.

[208] S. van der Walt, S. C. Colbert, and G. Varoquaux. The numpy array: A structure for efficient numerical computation. *Computing in Science Engineering*, 13(2):22–30, March 2011.

[209] S. W. Yuan and H. N. Bertram. Fast adaptive algorithms for micromagnetics. *IEEE Transactions on Magnetics*, 28(5):2031–2036, Sept 1992.

[210] NobuoHayashi, KojiSaito, and YoshinobuNakatani. Calculation of demagnetizing field distribution based on fast fourier transform of convolution. *Japanese Journal of Applied Physics*, 35(12R):6065, 1996.

[211] D. Hinzke and U. Nowak. Magnetization switching in nanowires: Monte carlo study with fast fourier transformation for dipolar fields. *Journal of Magnetism and Magnetic Materials*, 221(3):365 – 372, 2000.

[212] Fidimag. Documentation - extended equations. `https://fidimag.readthedocs.io/en/latest/extended_eqs.html`. Accessed: 2018-02-06.

[213] Alan C Hindmarsh, Peter N Brown, Keith E Grant, Steven L Lee, Radu Serban, Dan E Shumaker, and Carol S Woodward. SUNDIALS: Suite of nonlinear and differential/algebraic equation solvers. *ACM Transactions on Mathematical Software (TOMS)*, 31(3):363–396, 2005.

[214] Codecov LLC. Dashboard computationalmodelling/fidimag. `https://codecov.io/gh/computationalmodelling/fidimag`. Accessed: 2017-10-04.

[215] Fidimag. Tutorial: A basic simulation. `http://fidimag.readthedocs.io/en/latest/ipynb/tutorial-basics.html`. Accessed: 2017-11-29.

[216] Fidimag. Issue page on github. `https://github.com/computationalmodelling/fidimag/issues`. Accessed: 2018-02-13.

[217] E. Hairer, S. P. Nørsett, and G. Wanner. *Solving Ordinary Differential Equations I (2nd Revised. Ed.): Nonstiff Problems*. Springer-Verlag, Berlin, Heidelberg, 1993.

[218] A.C. Hindmarsh and Lawrence Livermore Laboratory. *ODEPACK, a Systematized Collection of ODE Solvers*. Lawrence Livermore National Laboratory, 1982.

[219] Peter N Brown, George D Byrne, and Alan C Hindmarsh. Vode: A variable-coefficient ode solver. *SIAM journal on scientific and statistical computing*, 10(5):1038–1051, 1989.

[220] Jeffrey C. Carver, Richard P. Kendall, Susan E. Squires, and Douglass E. Post. Software Development Environments for Scientific and Engineering Software: A Series of Case Studies. In *Proceedings of the 29th international conference on Software Engineering*, ICSE '07, pages 550–559, Washington, DC, USA, 2007. IEEE Computer Society.

[221] Anders Logg, Kent-Andre Mardal, Garth N. Wells, et al. *Automated Solution of Differential Equations by the Finite Element Method*. Springer, 2012.

[222] Anders Logg and Garth N. Wells. DOLFIN: Automated Finite Element Computing. *ACM Transactions on Mathematical Software*, 37(2), 2010.

[223] Giovanni Di Fratta, Cyrill B. Muratov, Filipp N. Rybakov, and Valeriy V. Slastikov. Variational principles of micromagnetics revisited. *SIAM Journal on Mathematical Analysis*, 52(4):3580–3599, 2020.

[224] B. Van de Wiele, A. Manzin, O. Bottauscio, M. Chiampi, L. Dupre, and F. Olyslager. Finite-difference and edge finite-element approaches for dynamic micromagnetic modeling. *Magnetics, IEEE Transactions on*, 44(11):3137 –3140, nov. 2008.

[225] P. N. Brown, G. D. Byrne, and A. C. Hindmarsh. VODE: A Variable Coefficient ODE Solver. *SIAM J. Sci. Stat. Comput.*, 1989.

[226] C. J. Garcia-Cervera and A. M. Roma. Adaptive mesh refinement for micromagnetics simulations. *IEEE Trans. Magn.*, 42(6):1648–1654, 2006.

[227] K. Beck. *Test driven development: By example.* Addison-Wesley Professional, 2002.

[228] M. Fowler. *Refactoring - improving the design of existing code.* Addison-Wesley, 1999.

[229] W. Rave, K. Fabian, and A. Hubert. Magnetic states of small cubic particles with uniaxial anisotropy. *Journal of Magnetism and Magnetic Materials*, 190(3):332 – 348, 1998.

[230] Satish Balay, Mark F. Adams, Jed Brown, Peter Brune, Kris Buschelman, Victor Eijkhout, William D. Gropp, Dinesh Kaushik, Matthew G. Knepley, Lois Curfman McInnes, Karl Rupp, Barry F. Smith, and Hong Zhang. PETSc users manual. Technical Report ANL-95/11 - Revision 3.4, Argonne National Laboratory, 2013.

[231] Youcef Saad and Martin H Schultz. Gmres: A generalized minimal residual algorithm for solving nonsymmetric linear systems. *SIAM J. Sci. Stat. Comput.*, 7(3):856869, July 1986.

[232] Christopher C Paige and Michael A Saunders. Solution of sparse indefinite systems of linear equations. *SIAM journal on numerical analysis*, 12(4):617–629, 1975.

[233] Roland W Freund. A transpose-free quasi-minimal residual algorithm for non-hermitian linear systems. *SIAM journal on scientific computing*, 14(2):470–482, 1993.

[234] Lewis Fry Richardson. Ix. the approximate arithmetical solution by finite differences of physical problems involving differential equations, with an application to the stresses in a masonry dam. *Philosophical Transactions of the Royal Society of London. Series A, Containing Papers of a Mathematical or Physical Character*, 210(459-470):307–357, 1911.

[235] H. A. Van der Vorst. BI-CGSTAB: A fast and smoothly converging variant of Bi-CG for the simulation of nonsymmetrical linear systems. *SIAM J. Sci. Stat. Comput.*, 13:631–644, 1992.

[236] J Andvandervorst Meijerink and Henk A Van Der Vorst. An iterative solution method for linear systems of which the coefficient matrix is a symmetric -matrix. *Mathematics of computation*, 31(137):148–162, 1977.

[237] Y. Saad. *Iterative Methods for Sparse Linear Systems.* Society for Industrial and Applied Mathematics, USA, 2nd edition, 2003.

[238] David M. Young. *Iterative methods for solving partial difference equations of elliptical type.* PhD thesis, Harvard University, 1950.

[239] J Ruge and K Stüben. Algebraic multigrid (amg). *Front. Appl. Math*, 3:73–130, 1987.

[240] Van Emden Henson and Ulrike Meier Yang. Boomeramg: A parallel algebraic multigrid solver and preconditioner. *Applied Numerical Mathematics*, 41(1):155 – 177, 2002. Developments and Trends in Iterative Methods for Large Systems of Equations - in memorium Rudiger Weiss.

[241] David Hysom and Alex Pothen. A scalable parallel algorithm for incomplete factor preconditioning. *SIAM Journal on Scientific Computing*, 22(6):2194–2215, 2001.

[242] Edmond Chow. A priori sparsity patterns for parallel sparse approximate inverse preconditioners. *SIAM Journal on Scientific Computing*, 21(5):1804–1822, 2000.

[243] Marc-Antonio Bisotti. Aeon - a non-intrusive python profiler. `https://pypi.org/project/Aeon/2.0.2/`. Accessed: 2017-10-04.

[244] Mark Vousden, Maximilian Albert, Marijan Beg, Marc-Antonio Bisotti, Rebecca Carey, Dmitri Chernyshenko, David Corts-Ortuo, Weiwei Wang, Ondrej Hovorka, Christopher H. Marrows, and Hans Fangohr. Skyrmions in thin films with easy-plane magnetocrystalline anisotropy. *Applied Physics Letters*, 108(13):132406, 2016.

[245] A. Bogdanov and A. Hubert. Thermodynamically stable magnetic vortex states in magnetic crystals. *Journal of Magnetism and Magnetic Materials*, 138(3):255 – 269, 1994.

[246] M. N. Wilson, A. B. Butenko, A. N. Bogdanov, and T. L. Monchesky. Chiral skyrmions in cubic helimagnet films: The role of uniaxial anisotropy. *Phys. Rev. B*, 89:094411, Mar 2014.

[247] Shi-Zeng Lin, Avadh Saxena, and Cristian D. Batista. Skyrmion fractionalization and merons in chiral magnets with easy-plane anisotropy. *Phys. Rev. B*, 91:224407, Jun 2015.

[248] Rebecca Carey, Marijan Beg, Maximilian Albert, Marc-Antonio Bisotti, David Cortés-Ortuño, Mark Vousden, Weiwei Wang, Ondrej Hovorka, and Hans Fangohr. Hysteresis of nanocylinders with Dzyaloshinskii-Moriya interaction. *Applied Physics Letters*, 109(12):122401, 2016.

[249] Mark Vousden, Marc-Antonio Bisotti, Maximilian Albert, and Hans Fangohr. Virtual micromagnetics: A framework for accessible and reproducible micromagnetic simulation. *Journal of Open Research Software*, 4:e41, 2016.

[250] Lorin Hochstein. *Ansible: Up and Running*. O'Reilly Media, Inc., 1st edition, 2015.

[251] Alexander Baker, Marijan Beg, Gregory Ashton, Maximilian Albert, Dmitri Chernyshenko, Weiwei Wang, Shilei Zhang, Marc-Antonio Bisotti, Matteo Franchin, Chun Lian Hu, Robert Stamps, Thorsten Hesjedal, and Hans Fangohr. Proposal of a micromagnetic standard problem for ferromagnetic resonance simulations. *Journal of Magnetism and Magnetic Materials*, 421:428 – 439, 2017.

[252] Michael Farle. Ferromagnetic resonance of ultrathin metallic layers. *Reports on Progress in Physics*, 61(7):755, 1998.

[253] Marijan Beg, Maximilian Albert, Marc-Antonio Bisotti, David Cortés-Ortuño, Weiwei Wang, Rebecca Carey, Mark Vousden, Ondrej Hovorka, Chiara Ciccarelli, Charles S. Spencer, Christopher H. Marrows, and Hans Fangohr. Dynamics of skyrmionic states in confined helimagnetic nanostructures. *Phys. Rev. B*, 95:014433, Jan 2017.

[254] Ryan Alexander Pepper, Marijan Beg, David Cortés-Ortuño, Thomas Kluyver, Marc-Antonio Bisotti, Rebecca Carey, Mark Vousden, Maximilian Albert, Weiwei Wang, Ondrej Hovorka, and Hans Fangohr. Skyrmion states in thin confined polygonal nanostructures. *Journal of Applied Physics*, 123(9):093903, 2018.

[255] Bin Zhang, Weiwei Wang, Marijan Beg, Hans Fangohr, and Wolfgang Kuch. Microwave-induced dynamic switching of magnetic skyrmion cores in nanodots. *Applied Physics Letters*, 106(10):102401, 2015.

[256] Weiwei Wang, Marijan Beg, Bin Zhang, Wolfgang Kuch, and Hans Fangohr. Driving magnetic skyrmions with microwave fields. *Phys. Rev. B*, 92:020403, Jul 2015.

[257] Jeffrey C Carver, Neil P Chue Hong, and George K Thiruvathukal. *Software engineering for science*. CRC Press, 2016.

[258] Stephen Crouch, Neil Chue Hong, Simon Hettrick, Mike Jackson, Aleksandra Pawlik, Shoaib Sufi, Les Carr, David De Roure, Carole Goble, and Mark Parsons. The software sustainability institute: Changing research software attitudes and practices. *Computing in Science and Engineering*, 15(6):74–80, 2013.

[259] Lisa M. Federer, Christopher W. Belter, Douglas J. Joubert, Alicia Livinski, Ya-Ling Lu, Lissa N. Snyders, and Holly Thompson. Data sharing in plos one: An analysis of data availability statements. *PLOS ONE*, 13(5):1–12, 05 2018.

[260] Bryan OSullivan and OSullivan Bryan. *Mercurial: The Definitive Guide*. OReilly Media, Inc., 2009.

[261] Inc. Atlassian. Bitbucket. `https://bitbucket.org`. Accessed: 2019-02-10.

[262] Scott Chacon and Ben Straub. *Pro git: Everything you need to know about Git*. Apress, second edition, 2014.

[263] Inc. GitHub. About pull requests. `https://help.github.com/en/github/collaborating-with-issues-and-pull-requests/about-pull-requests`. Accessed: 2019-02-10.

[264] John Ferguson Smart. *Jenkins: The Definitive Guide*. O'Reilly Media, Inc., 2011.

[265] Travis CI GmbH. fangohr/finmag - travis ci. `https://travis-ci.org/fangohr/finmag`. Accessed: 2017-05-01.

[266] Sphinx. Python documentation generator. `http://www.sphinx-doc.org`. Accessed: 2017-11-20.

[267] The Jupyter Project. Home. `https://jupyter.org`. Accessed: 2017-11-20.

[268] Hans Fangohr, Maximilian Albert, and Matteo Franchin. Nmag micromagnetic simulation tool: Software engineering lessons learned. In *Proceedings of the International Workshop on Software Engineering for Science*, pages 1–7, 2016.

[269] Mitchell Hashimoto. *Vagrant: up and running: create and manage virtualized development environments*. " O'Reilly Media, Inc.", 2013.

[270] Inc. Anaconda. Anaconda software distribution. `htts://anaconda.com`. Accessed 2017-09-25.

[271] Carl Boettiger. An introduction to docker for reproducible research. *SIGOPS Oper. Syst. Rev.*, 49(1):71–79, January 2015.

[272] Docker Inc. Docker Security – Docker daemon attack surface. `https://docs.docker.com/engine/security/#docker-daemon-attack-surface`. Accessed: 2020-12-05.

[273] Lucas Benedicic, Felipe A. Cruz, Alberto Madonna, and Kean Mariotti. Sarus: Highly Scalable Docker Containers for HPC Systems. In Michèle Weiland, Guido Juckeland, Sadaf Alam, and Heike Jagode, editors, *High Performance Computing*, pages 46–60, Cham, 2019. Springer International Publishing.

[274] Marijan Beg, Ryan A Pepper, David Cortés-Ortuño, Bilal Atie, Marc-Antonio Bisotti, Gary Downing, Thomas Kluyver, Ondrej Hovorka, and Hans Fangohr. Stable and manipulable bloch point. *Scientific reports*, 9(1):1–8, 2019.

[275] Marijan Beg, Ryan A. Pepper, David Cortés-Ortuño, Bilal Atie, Marc-Antonio Bisotti, Gary Downing, Thomas Kluyver, Ondrej Hovorka, and Hans Fangohr. GitHub repository: Stable and manipulable Bloch point. (Version 1.0.0) [Data set]. *Zenodo http://doi.org/10.5281/zenodo.2938933*, 2019.

[276] P Jupyter, M Bussonnier, J Forde, J Freeman, B Granger, T Head, C Holdgraf, K Kelley, G Nalvarte, A Osheroff, et al. Binder 2.0 - reproducible, interactive, sharable environments for science at scale. In *Proceedings of the 17th python in science conference*, volume 113, page 120, 2018.

[277] Benny Malengier, Pavol Kišon, James Tocknell, Claas Abert, Florian Bruckner, and Marc-Antonio Bisotti. Odes : a high level interface to ode and dae solvers. *Journal of Open Source Software*, 3(22), 2018.

[278] Vahid Garousi, Markus Borg, and Markku Oivo. Practical relevance of software engineering research: Synthesizing the community's voice. *Empirical Software Engineering*, March 2020.

[279] Daniel Andriessen. Reconciling the rigor-relevance dilemma in intellectual capital research. *Learning Organization, The*, 11:393–401, 08 2004.

[280] Herbert Altrichter, Stephen Kemmis, Robin Mctaggart, and Ortrun Zuber-Skerritt. The concept of action research. *Learning Organization, The*, 9:125–131, 08 2002.

[281] Joan van Aken. Management research based on the paradigm of the design sciences: The quest for field-tested and grounded technological rules. *Journal of Management Studies*, 41, 02 2001.

[282] Daniel Andriessen. *Making Sense of Intellectual Capital: designing a method for the valuation of intangibles*. Butterworth-Heinemann, Boston, 2003.

[283] A. J. Ko. The black hole of software engineering research. `https://blogs.uw.edu/ajko/2015/10/05/the-black-hole-of-software-engineering-research/`, 2015. Accessed: 2020-12-12.

[284] Mariam Mezouar, Feng Zhang, and Ying Zou. An empirical study on the teams structures in social coding using github projects. *Empirical Software Engineering*, 24, 12 2019.

[285] Matthäus Zylka and Roland Schreiber. Social network analysis in software development projects: A systematic literature review. *International Journal of Software Engineering and Knowledge Engineering*, 30:321–362, 05 2020.

[286] Yanming Yang, Xin Xia, David Lo, Tingting Bi, John Grundy, and Xiaohu Yang. Predictive models in software engineering: Challenges and opportunities, 08 2020.