Micromagnetic Modeling by Computational Science Integrated Development Environments (CSIDE)

D i s s e r t a t i o n

zur Erlangung des Doktorgrades Dr. rer. nat.
des Fachbereichs Informatik
der Universität Hamburg

vorgelegt von

Najafi Maryam Negari, Massoud

Hamburg
2011

Prüfungsausschuss

Genehmigung der MIN-Fakultät, Fachbereich Informatik der Universität Hamburg auf Antrag von:

Erstgutachter	Prof. Dr. Dietmar P. F. Möller
Zweitgutachter	Dr. habil. Guido Meier
Drittgutachter	Prof. Dr. Thomas Ludwig
Vorsitzender des Prüfungsausschusses	Professor Dr. Jianwei Zhang
Datum der Disputation	Hamburg, den 07. Juni 2011

Abstract

Nowadays micromagnetic simulations are the third pillar for the investigation of micro and nanostructured ferromagnetic materials. Micromagnetic simulations are used, where analytical calculations are too complex or experimental measurements are not available. Recently the influence of electric currents and temperature on the local magnetization has become a research priority, as these two phenomena led to novel memory devices like the the STTRAM or the racetrack memory. Generally the research interest is changing to the simulation of experimental setups including more and more physical phenomena. Therefore micromagnetic simulators are required that allow conveniently to perform simulations and to include new phenomena. The present work deals with the design of the finite-difference-method based micromagnetic simulator M^3S. The computational science focus of this design is the evaluation of computational science integrated development environments (CSIDEs) as the development basis combined with advanced software engineering concepts like object-oriented programming (OOP) and test-driven design (TDD). Important requirements for a micromagnetic simulator are identified and their realization possibilities using CSIDEs are evaluated by comparing three different CSIDE based M^3S prototypes. The evaluation revealed that using actual CSIDEs reduces the software complexity of a simulator significantly compared to pure *C/C++* or *FORTRAN* solutions, while maintaining a competitive runtime performance. The physical focus of the design of M^3S is the investigation of ferromagnetic systems effected by a current flow. Therefore the spin-transfer torque and the anisotropic magnetoresistivity (AMR) effect as two important phenomena are integrated into M^3S. The validation of the former extension has been addressed by proposing a new standard problem. The high sensibility of the proposed problem to errors is shown on the basis of typical error cases. Further the simulation results of different micromagnetic simulators are compared with an experimentally validated analytical model. It turns out that the proposed problem can discriminate errors larger than 3 %. The simulation experiment used for the proposed standard problem further revealed good properties for the measurement of the degree of non-adiabaticity. As a result a robust measurement scheme for this value has been proposed. The measurement scheme is robust against typical falsifying uncertainties occuring in experimental measurements. The scheme thus allows an estimation of the degree of non-adiabaticity with an accuracy of 5 %.

Zusammenfassung

Heutzutage stellt die mikromagnetische Simulation die dritte Säule bei der Untersuchung mikro- und nanostrukturierter ferromagnetischer Materialien dar. Mikromagnetische Simulationen werden dort eingesetzt, wo analytische Berechnungen zu komplex und experimentelle Messungen nicht realisierbar sind. Die Einflüsse von elektrischem Strom und Temperatur auf die lokale Magnetisierung sind aktuelle Forschungsschwerpunkte, da diese beiden Phänomene erfolgreich zur Entwicklung neuartiger Speichermedien, wie z.B. dem STTRAM oder dem Racetrack Speicher führten. Generell lässt sich ein Wandel des Forschungsinteresses zur Simulation experimenteller Versuchsaufbauten unter Berücksichtigung von immer mehr physikalischen Phänomenen feststellen. Dies erfordert mikromagnetische Simulatoren, die sowohl das Durchführen von Simulationen als auch das Einarbeiten neuer Phänomene komfortabel ermöglichen. Die vorliegende Arbeit behandelt den Entwurf des mikromagnetischen Simulators M^3S auf Basis der finiten Differenzen Methode. Aus Sicht der rechnergestützten Naturwissenschaften wird bei diesem Entwurf der neue Ansatz der "computational science integrated development environments" (CSIDEs) kombiniert mit fortschrittlichen Software-Entwurfstechniken, wie objektorientierter Programmierung und testgetriebenem Entwurf, verfolgt. Zunächst werden hierzu wichtige Anforderungen an einen mikromagnetischen Simulator identifiziert und darauffolgend die Realisierungsmöglichkeiten anhand dreier M^3S Prototypen miteinander verglichen. Diese Analyse zeigt, dass der Einsatz aktueller CSIDEs die Softwarekomplexität eines Simulators im Vergleich zu reinen *C/C++* oder *FORTRAN* Lösungen signifikant reduziert und zeitgleich zu einer vergleichbaren Laufzeitperformanz führt. Aus Sicht der Physik steht beim Entwurf von M^3S die Untersuchung von stromgetriebenen ferromagnetischen Systemen im Fokus. Hierzu wurden das Spintransfermoment und der anisotropische Magnetowiderstandeffekt (als zwei wichtige Phänomene) in M^3S integriert. Zur Validierung der erstgenannten Erweiterung, wurde ein neues Standardproblem vorgeschlagen. Die hohe Fehlersensibilität des Vorschlags wird anhand typischer Fehler demonstriert. Weiterhin werden die Simulationsergebnisse verschiedener Simulatoren mit einem experimentell validierten, analytischen Modell verglichen. Es zeigt sich, dass das Problem Fehler größer als 3 % aufdecken kann. Das im Standardproblem genutzte Simulationsexperiment zeigte weiterhin gute Eigenschaften für die Messung des Grades der Nichtadiabatizität. Als Ergebnis wurde eine Messmethode zur Bestimmung dieser Größe vorgeschlagen, die robust gegen typische verfälschende Einflüsse, die bei bisherigen Experimenten auftraten, ist. Sie ermöglicht daher die Bestimmung des Grades der Nichtadiabatizität mit einer Genauigkeit von 5 %.

Contents

Chapter 1

Introduction

Ferromagnets are used in devices that require nonvolatile storage of information, like hard disks in which they are in use for more than 50 years.[1] Recently, the field of research has been extended to the development of nanometer-sized ferromagnetic nonvolatile storage devices that offer a high storage density accompanied by a high data rate.[2,3] The magneto-resistive random access memory (MRAM) has been developed as a novel nano-structured ferromagnetic memory module.[4] To write information in such an MRAM-cell an Oersted field is applied to switch the cell.[4,5] As explained by various authors there are different restrictions using an Oersted field.[6,7]These limit the storage density of the MRAM due to field leakages. To make the MRAM competitive with other memory technologies like the dynamic random-access memory (DRAM), the static random-access memory (SRAM), or the Flash memory, the storage density needs to be significantly increased.[2,3]

In 1996 it was predicted[8,9] that a spin-polarized current flowing through a ferromagnetic conductor can apply a torque to its magnetization. Since its discovery the so-called spin-transfer torque (STT) has been considered as a key mechanism to increase the storage density and has led to a new generation of storage devices.[10–13] Two promising proposals are the spin-transfer torque random-access memory (STTRAM)[10] and the racetrack memory.[11] The STTRAM is an MRAM which uses the spin-transfer torque instead of the Oersted field for the writing process. The racetrack memory stores bits along a single ferromagnetic wire by domain walls. To read and write information, a current is applied along the wire that moves the bits to a reading or writing unit. The lesson learned from other memory devices such as the DRAM or the SRAM is, that it is necessary to develop analytical descriptions, compact models, and powerful simulation tools[2,3] to optimize the properties of a memory device.

For the simulation of ferromagnetic structures that are influenced by magnetic fields like an Oersted field, micromagnetic simulators are well accepted. These simulators need to be extended by the interplay of the magnetization and the current flow to be suited for the simulation of current-carrying ferromagnetic structures. But to extend these simulators

by new physical phenomena the micromagnetic model[14] needs to be extended first. The micromagnetic model describes appropriately the dynamics of the magnetization in ferromagnetic micro- and nano-structures. In this model the magnetization is assumed to be a spatial- and time-dependent continuous function. The magnetization dynamics are described by the Landau-Lifshitz-Gilbert (LLG) equation[15] including the energy contributions of the anisotropy, the exchange interaction, the magnetostatic interaction, and the Zeeman energy.

As proposed by Slonczewski[8] and Berger,[9] the spin transfer torque can be included in the micromagnetic model by adding current dependent spin-transfer torque terms to the LLG equation. As the spin transfer torque is not fully understood, only descriptions for two special cases exist. The first description has been developed by Slonczewski;[8,16] it accurately describes the torque arising from currents traversing through interfaces between ferromagnets and non-magnets as can be found in the STTRAM. In such an STTRAM the influence of the magnetization on the current is considered to depend on the structure of the multilayer by the giant magneto-resistivity (GMR)[17] or the tunnel magneto-resistivity (TMR)[18] effect. The second was developed by Bazaliy et al.[19] and has been extended by Zhang and Li[20] and Thiaville et al.[21] It deals with the spin-transfer torque due to continuous changes in the magnetization, e.g. due to domain walls or magnetic vortices. Since in this case the current flow is influenced by the magnetization through the anisotropic magneto-resistance (AMR) effect. The AMR needs to be considered self-consistently, in order to cover the interplay of the current and the magnetization.

The investigation of the STTRAM is only one example for the importance micromagnetic simulations have gained during the last decade. The importance of micromagnetic simulations rests on the possibility to make predictions and interpretations of the dynamic behavior of complex ferromagnetic systems. Considering that the communities research interest is moving to the simulation of real experimental setups including more physical phenomena, two demands will face up.

1. The demands for computation performance is further increasing. Current trends in the computer hardware show that a further increase in computation performance is only possible by parallel computing.[22–24] Thus the only possibilities to comply the demands to the runtime of a micromagnetic simulation are to optimize the algorithms and to parallelize them on novel hardware architectures.

2. With the new possibilities micromagnetic simulation offers for the investigation of ferromagnetic systems its user community grows. Most of the new community members will be physicists with a restricted knowledge in numerical analysis and software development. These users are expected to concentrate on their subject and thus need tools that are convenient to use and to extend.

These demands are concurrent, as the performance optimization and parallelization of a large program increases its software complexity drastically. Thus the hurdle to perform

micromagnetic simulations and to change the simulation to the users needs is increased. The conflict between runtime performance optimizations and maintainable software can be found in most computational science areas.[25] The experiences of the last decades in the scientific computing community have shown that this conflict could only be handled by prioritizing the software quality criteria portability and maintainability equally to the runtime performance of a scientific program.[25,26]

As a consequence scientific software environments like *MATLAB*, *Mathematica*,[27–29] *Java*[30] and its built-in scripting engine,[31] and *Python*[32] combined with basic numerical *C/C++* and *FORTRAN* libraries[33,34] have been developed. As introduced by Hudak et al.[35,36] these so-called computational science integrated development environments (CSIDE) offer a better balance between the software quality criteria compared with a pure *C/C++* or *FORTRAN* solution.

In addition to these criteria, the correctness and trustworthiness of software applications have to be ensured. Checking the correctness of a scientific software application is a difficult task. Many well-established validation methods that are used in the software development of enterprise software applications are not directly applicable to the development of scientific software.[25,26,37–39] Especially complex simulators are used when the mathematical models cannot be solved analytically. In such a case, it is difficult to find simulation problems with a known behavior that can be used as system tests. The comparison with investigations on real systems is also difficult for several reasons:

- The simulation is consulted for systems that cannot be investigated in reality.
- Real experimental results can only be used for a qualitative comparison, as the experimental results are affected by parasitic influences.

Referring to the scientists intuition for the expected behavior of the real system is often the only way to identify system-test specifications. In the micromagnetic community this problem has been faced by the Micromagnetic Modeling Activity Group (μMag).[40] This group has collected system test specifications for micromagnetic simulations, so-called standard problems. Until now, four standard problems have been published by μMag including the anisotropy, the demagnetization, the exchange, and the Zeeman field. Since their publication, these problems were referred in many following research activities to investigate the accuracy of the applied mathematical and numerical algorithms.[40]

In summary this thesis deals with the design and development of a finite-difference-method based micromagnetic simulator that allows to investigate ferromagnetic systems effected by a current flow. Furthermore the validation of the numerical models is discussed.

Another aspect is to see, if the use of a CSIDE to develop a complex simulator really reduces the software complexity and thus increases its usability while resulting in a

reasonable runtime performance. This investigation is important, as commonly CSIDE are used to prototype a scientific software application which is later reimplemented in *C/C++* or *FORTRAN* to increase the runtime performance. The question is now, if the current state of CSIDE really necessiates a full reimplementation.

This thesis is organized as follows:

Chapter 2 gives an overview of the fundamentals this thesis is based on. Section 2.1 summarizes aspects to be considered when developing scientific software. The section gives an overview of actual trends in the scientific computing community, and introduces the new approach of scientific development environments. Section 2.2 introduces the micromagnetic model, the spin-transfer-torque extensions, and the AMR effect. Finally Sec. 2.3 reviews existing micromagnetic simulators and the possibilities for the validation of these simulators.

Chapter 3 presents a micromagnetic simulator prototype written in *MATLAB*.[41] Based on this prototype benefits and pitfalls of using a CSIDE in general and *MATLAB* in detail for the development of a complex simulator are discussed. The validity of the prototype is proved by results for standard problem No. 4 and the Larmor-precession test. Further algorithmic runtime optimizations and possibilities for parallelization are identified and their feasibility using *MATLAB* is determined. Finally it is evaluated, if the restrictions identified for *MATLAB* are general restrictions of CSIDEs or only *MATLAB* specific. Therefore the *MATLAB* prototype is compared with two other prototypes written in *Java/Java Scripting API (JSA)*[30,31] and *Python/SciTools*.[32,42]

Chapter 4 deals with the integration of the spin-transfer torque extensions and the AMR effect into the micromagnetic simulation. Section 4.1 presents discretized models and implementation details of the spin-transfer torque extension. The verification of the spin-transfer torque extension for a spin valve has been prooven by comparing simulation results with the results published by Berkov and Gorn.[43] The correctness of the spin-transfer torque extension for continuously variable magnetization textures has been verified by a system test developed during this work. This system test has been proposed as a new standard problem and is presented in Sec. 4.2. Section 4.3 at last discusses a proposal for a robust measurement scheme for the degree of non-adiabaticity with a great accuracy of 5 % that is one order better than previous measurements of this property.

Chapter 5 concludes the thesis and describes possible future work.

Chapter 2

Fundamentals

This chapter introduces common challenges for the development of scientific software and reviews current trends to handle these challenges, focusing on the approach of computational science integrated development environments (CSIDE)[35,36] and test driven design (TDD).[44]

2.1 Development of scientific software

Computer based numerical analysis replaced scientific assistants that performed the numerical analysis by hand. Numerical analysis, recently also called *scientific computing* nowadays is one of the three pillars of computational physics.[45,46] Landau[46] and Basili et al.[47] summarized that most of the computations are performed on desktop computers rather than on supercomputers. Since the hardware architecture also for desktop computers has changed to parallel computer architectures, the parallelization of the sequential algorithms has become an important method to increase the runtime performance.[24] On the other hand the development of scientific software is different to the development of commercial application.[38] The requirements for a scientific program are not clear; often they are a result of the develpment itself since intermediate solutions help the scientist to identify requirements. In this sense, scientific software development is following an experimental approach.

2.1.1 Trends in scientific software development

Scripting and opportunistic programming

Scripting has evolved to an important method in computational sciences and scientific computing. This is due to the simplification that scripting offers non-experienced users. As explained by Ousterhout et al.[48] the scope of *scripting languages* is more likely the connection of system functionality rather than offering the possibility to efficiently implement new system functionalities. That is why they are also called *glue languages* or *system integration*

languages. Scientists started to use scripting with the upcoming of *TCL* and *Perl* at the end of the 1970ś. At that time scripting was mainly used to write small programs for automated simulation runs or for the development of graphical user interfaces (GUIs). It was also used on supercomputers to organize and schedule distributed jobs. Recently many scripting languages exist that are used by scientists.

Brandt et al.[49] describe the way non-experienced software developers implement software as *opportunistic programming*. Their case study showed, that non-experienced users do not write a code from scratch. They try the *copy-and-paste programming* and develop the code in an experimental way. This means, they take code snippets from a knowledge base (for instance via the world wide web) and modify it to get the desired functionality. Concerning the software engineering knowledge of scientists and their experimental way to develop scientific software, opportunistic programming fits well to scientists with a low experience in software development.

Numerical libraries and domain specific frameworks

Most of the investigated problems in computational physics are described by models expressible in mathematical notations, which can be represented by discretized computational models and then solved by a computer. During the last decades numerical libraries[50–53] have been developed using mainly the programming languages *C/C++*, and *FORTRAN*, as well as recently *Java*. These libraries all together implement most of the basic numerical algorithms in an optimal way. An overview of existing libraries has been given for instance by the NetLib project[54,55] or the Java Numerics Group.[56]

Another trend is, that a variety of domain specific frameworks have been developed for many areas of scientific computing. An overview of these frameworks is given by Steinhaus.[57] As investigated by Carver et al.[26] many scientific software is written in pure *C/C++* or *FORTRAN*. For the development scientifists prefer Unix based editors more than integrated development environments (IDEs). Numerical libraries are often used, while domain specific frameworks do not have this acceptance. A review of existing micromagnetic simulation packages and the used approaches is given in Sec. 2.3.

The decision for the use of a standard numerical libraries/domain specific framework or an own implemented library is an example for a so-called *make or buy* decision. The *make or buy* decision is a general management concept, that applied to software products decribes the decision between the development of a piece of software inhouse or the purchase of a license for an externally provided piece of software that offers the desired functionality.[58] The benefits of an inhouse development is that the developer has the full control of the software and can change it to its needs. The benefit of a purchased software is that the software commonly is extended during the time by the external provider and often includes better optimized algorithms as an inhouse software. Hence often, with the purchased

software its included knowledge is purchased too. The new trend for open-source libraries offers here a new option, as in contrast to a purchased software an open-source software can be changed to the users needs.[59]

Parallel computing

Current trends in the computer hardware indicate that the future computer architecture also on desktop computers will be a parallel architecture.[22–24]

Here the community offers a variety of parallel hardware architectures. For example using field programmable gate arrays (FPGA) for *reconfigurable computing*,[60] or *CUDA*[61] on general purpose graphical processing units (GPGPU) has become competitive to well-established techniques like *symmetric multiprocessing* (SMP) and cluster-based parallelization using the *message passing interface* (MPI).[62,63] Looking at the top 500 list,[64] that lists the 500 fastest supercomputers of the world, shows that the next step is a combination of these techniques. This trend shows, that further advances of the hardware can be expected in future.

Parallelizing scientific software is a difficult task. Thus spending effort on the run-time performance optimization for a given hardware[25] can result in lost time, if the hardware renewing period is too short. At the same time scientific software has been developed without the view on the later parallelization of the code which complicates the parallelization process.

Several strategies may be applied on sequential algorithms to run them on parallel hardware:[65]

- Run the highly optimized sequential program tasks in parallel to perform parameter sweeps. This tactic is limitedly applicable as the simulation of micromagnetic problems has increased in complexity and the run time of a single simulation has exceeded a critical value.

- Use special compilers that are able to identify parallelization possibilities. This tactic is limited as parallelization possibilities can be located only on a high level beyond the scope of a compiler resulting in a worse performance gain compared to a manually performed parallelization.

- Use numerical libraries to express the algorithms and replace them by parallel versions. This tactic is the most promising tactic but its success depends on the existence of appropriate numerical libraries that offer the required functionality.

- For special cases, parallelized domain-specific frameworks have been developed. These frameworks can appropriately take the parallelism at different levels into ac-

count. In comparison to numerical libraries, adaptations concerning new hardware architectures need more time due to the smaller user community.

The aim of these strategies is to reduce the effort for using the parallel resources efficiently and to prevent scientists from reinventing the wheel. While the first two strategies necessiate no change of the source code to parallelize the code, the third and fourth strategy require sophisticated parallization knowledge that scientists usually do not have. The solution to consult an expert to perform the reconstruction * yields the problem that such an expert does not understand a software with a low intrinsic quality, so it is difficult to change the code. This can necessitate prior refactoring steps to increase first the intrinsic quality.

2.1.2 Scientific development environments

Scientific development environments or, as introduced by Hudak et al.[35,36] so-called *computational science integrated development environments (CSIDE)* like *MATLAB*, *Maple*,[68] *Mathematica*,[69] *O-Matrix*,[70] *Octave*,[71] *SciLab*,[72] or *Python/SciTools*[32,42] with the flexibility of a mathematically motivated scripting languages and integrated development environments (IDEs) as well as an extensive data analysis and visualization functionality have been developed based on numerical libraries.[27–29,35,36] These environments simplify the installation and access to the underlying libraries. Further they offer an interpreted scripting language that allows the interactive implementation of scientific models and user specific analysis functionality. Finally most of the environments handly and hide compatibility problems between different necessary libraries from the user, by offerening anconsistent superimposed application programming interface (API).

As summarized by Carver et al.[26] the acceptance of a CSIDE depends on:

> *" To be adopted by scientific and engineering programmers, a programming language has to be easy to learn, offer reasonably high performance, exhibit stability, and give developers confidence in the validity of the resulting machine instructions. "*

This work exemplary focuses on the CSIDEs *MATLAB*, *Java/JSA* and *Python/SciTools*. *MATLAB* and *Python/SciTools* have been chosen as they represent well-established CSIDEs within the scientific community and are based on *C/C++* or *FORTRAN*. *Java* is a young programming language and not established in the high performance and scientific computing community yet. Although itoffers unique possibility, like a just in time (JIT) compiler or a garbagecollector,[30] that cannot be found in *C/C++* or*FORTRAN*. In the following a short overview of the main propertiesfor all three CSIDEs is given. A more detailed review and comparison betweenthese three CSIDE is given in Sec. 3.3.

*A reconstruction is *"the transformation from one representation form to another at the same relative abstraction level, while preserving the subject system's external behavior (functionality and semantics)"*.[66,67]

MATLAB

MATLAB[41] became the leading scientific computing environment during the last decades.[73] Its success can be attributed to five of its main ideas:

- It offers a mathematically motivated scripting language that allows to formulate matrix operations in a clear and short way. The conformity to mathematical notations reduces the effort to learn the programming language, and to identify errors in the resulting implementation.[74]
- Established numerical frameworks like the basic linear algebra subroutines (BLAS),[51] LAPACK,[53] *FFTW*[50,75–78] are made available by the *MATLAB* runtime environment. Here compatibility and installation problems are hidden.[41]
- Compiled versions of the runtime environment are provided for many operating systems.
- A documentation, a knowledge base, and an interactive runtime environment are provided. Thus, the needed functionality can be developed by copying and pasting together code snippets from example code listings.[49]
- Through its license it is available at many universities. In addition, there exist many open-source tools that offer a translation of a *MATLAB* program into an open-source version or to compile it to a stand alone program.

From the software engineering point of view *MATLAB* offers a debugger, a code *Lint*-like tools for static analyzis of source code,[79] and a performance analyzing tool that includes code coverage metrics. Test packages have only been published as third-party projects.[80]

One drawback of *MATLAB* is, that all *C/C++* or *FORTRAN* based framework functions are not open source and thus not changeable if necessary. This circumstance leads to the development of several concurrent open-source or license-free alternatives that directly aim to provide a *MATLAB* derivative.[71,72,81] The optimization and parallelization of *MATLAB* script routines is a difficult task. *MATLAB* offered a solution for this problem in 2006 with the distributed computing toolbox and later with the parallel computing toolbox. The use of these toolboxes results in a parallelization with a poor speed-up. Several free solutions have been provided by the community.[82–85]

Python/SciTools

Python is a powerful scripting language that supports many software engineering concepts and is well-suited for large development projects.[86] It is widely used on *Unix* systems and recieved a large acceptance in the scientific computing community.

Python combined with the numerical libraries collection *SciTools*,[42] the interactive *Python* shell *IPython*[33] and the IDE *Eclipse* results in a powerful CSIDE comparable to *MATLAB*:[46]

- Eclipse[87] has been chosen, as it is a well-established software development environment, and combined with the *Pydev* project[88] it offers support for a wide range of software engineering concepts like refactoring and autocompletition for *Python*.
- *IPython*[33] has been chosen, as it is an extension of the interactive shell of *Python*, that offers syntax highlighting and autocompleting. It offers an exhaustive package for parallel computing in *Python*.
- *SciTools*[42] is a collection of different well-established numerical *Python* libraries. The collection includes the numerical capabilities of the *Python* libraries *NumPy*[89] and *SciPy*,[90] which themselves interface well-established numerical *C/C++* and *FORTRAN* libraries.[51,53,91] This combination results in a simple and clear similar to the API provided by *MATLAB*.[34]
- *Python* offers the automated testing frameworks *PyUnit*[92] and *py.test*[93] including test driver and test coverage tools. The main difference between the *PyUnit* and *py.test* tool is that *PyUnit* corresponds to the general xUnit specification[94] while the *py.test* package is less restrictive considering the structure of a test function.

Java/JSA

This approach is a step back compared to the previous prototypes. In this approach the physical core is implemented in *Java* similar to the architecture of *OOMMF*. Scripting is used here to implement the user script. This approach has been chosen to see if *Java* as upcoming programming language in the scientific computing community is competitive in this comparison.

In the newest version *Java 1.6.0_20* offers following important programming language elements by default:

- The *Java Runtime Environment (JRE)* makes *Java* a platform independent programming language. The basic idea is *"compile once, run everywhere"*. Technically this is realized in *Java* by splitting the compilation of a *Java* program into two steps. In the first step the *Java* program code, also called *source code*, is compiled to a system independent intermediate code called *byte code*. In the second step the byte code is executed on the users system calling the JRE. In the first versions of the JRE the byte code was interpreted at runtime resulting in a low runtime performance especially compared to *C/C++* or *FORTRAN*. In the current version of *Java* the JRE uses a *just in time* (JIT) compiler. In contrast to the interpreter, the JIT-compiler compiles the byte code to machine code, the first time the byte code is used. The JRE allows platform independency and the use of system-specific compilation settings as the JRE knows the system details when calling the JIT.

- In contrast to *C/C++* or *FORTRAN*, *Java* includes a so-called *garbage collector* that handles the freeing of the memory and thus releases the user from one of the most severe causes of programming errors. Especially in concurrent programs it is difficult to identify, if a memory is used by other threads or not. While the garbage collector in the first *Java* versions was a reason for a low runtime performance, in the newest *Java* version [†] special garbage collections for concurrent and parallel programs are included.
- Similar to the *Python* approach the IDE *Eclipse* is used, as *Java* combined with *Eclipse* offers extensive refactoring functionality.
- The *Java Scripting API* (JSA) is included in *Java* since the version 1.5. It offers by default engines to *Groovy*[95] or *Java Script*[96] but also allows to implement new engines for own scripting languages. *JSA* can either be used to provide a scripting API for the user script or to implement a domain-specific scripting API interfacing a domain-specific *Java* package. In the following, the *Java Script* engine is chosen as first approach. *Java Script* has been chosen due to its wide distribution in the computer science community. It is well known by many users and supports many structures like inheritance that are important to include user-specific code.
- A wide range of tools for runtime profiling, test coverage measurement, object-oriented analysis, and refactoring, that simplify the software development, exist for *Java*. In this project the test coverage tool *EMMA*[97] was used. In contrast to the *MATLAB* profiler *EMMA* offers to measure the (C_0) and (C_1) test coverage for all existing unit tests. The (C_0) and (C_1) test coverage measures will be explained in the following.

2.1.3 Validation and verification of scientific software

Several authors emphasize the importance of validation and verification of scientific software, since their subject is to proof, if the results appropriatly describe the reality.[26,37,47,98,99] The article of Hook and Kelly[37] points out, that no coherent definition exists accross the computational-science and engeneering communities due to the synonymos use of the terms validation and verification. In the following the definition of Hook and Kelly for the terms validation and verification are introduced:

- *"Validation for scientists primarily means checking the computer output against a reliable source, a benchmark that represents something in the real world. In the literature, validation is described by scientists as the comparison of computer output against various targets such as measurements (of either real world or bench test events), analytical solutions of mathematical models, simplified calculations using the computational models, or output from other computer software. Whether that target is another computer program, measurements taken in the field, or human knowledge, the goal of validation is the same: is the computer output a reasonable proximity to the real world?"*

[†] in this thesis this was version 1.6.0_20

- *"Verification is also described as a comparison of the computer output to the output of other computer software or to selected solutions of the computational model. Roache succinctly calls verification "solving the equations right".*[99] *This includes checking that expected values are returned and convergence happens within reasonable times. The goal of verification is the assessment of the suitability of the algorithms and the integrity of the implementation of the mathematics."*

Hook and Kelly identify validation and verification as software test goals and introduce a new model for testing scientific software as shown in Fig. 2.1 that sets these test goals in relation. In this model the new test goal code scrutinization is introduced as: *"Code scrutinization addresses code faults that arise in the realization of models using a computer programming language."*

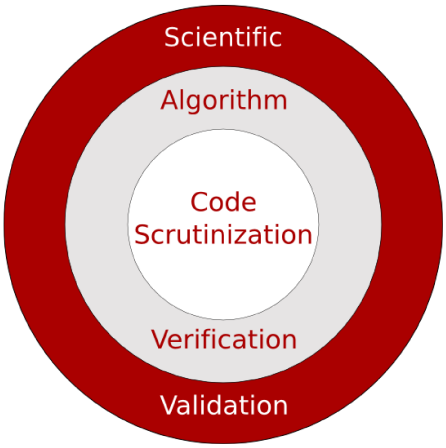


Figure 2.1: Model of Testing scientific software, modified from Hook and Kelly.[37]

This work covers these test goals by applying dynamic software test methods on the software. A *dynamic software test method* proves a program component by running the component with well-chosen input data and comparing the results with expected reference values.[44,100] This can be performed manually or by automated testing. Such an procedure tests the component randomly. It cannot prove the correctness of the component but allows to check its correctness for typical cases, i.e. wrong types or wrong number of arguments. Dynamical software test methods are further subdivided in *black-box* and *white-box tests*.[44]

A *black-box test* is a test that is derived from the specification for the component and is also called functional test. The challenge of such atest is to derive suitable test data from the specification. This can be donefor instance by using the equivalence partitioning and the boundary valueanalysis methods.[101,102]

A *white-box test* is a test that is derived from the source code of the component. Here the problem can occur that the developer is routine-blinded and does not test the component appropriately.

Since an automated dynamic test runs a component for distinct use cases and checks the results against expected results, the test simultaneously includes a description of the use case. In this way the test is simultaneously a documentation of the use case of the tested component and a working example for its usage. Since from the validation perspective each test has to run correctly, the developer spends time to keep the tests up-to-date. This means that in contrast to other documentation the dual nature of a dynamic test results in an up-to-date documentation of the code.[44]

System-/unit-tests

A test can be categorized by the object under test. One distinguishes between system and unit tests.

System tests use the whole software system or a distinct subpart of the system as a test object. A system test checks the expected behavior of the software system from the users (or specifications) point of view and is an example for a black-box test.[44] System tests cover large-grained functionality in order to help developers to find bugs. Therefore in addition so-called unit tests are used.

Unit tests use the smallest testable program units of the software system as test objects. Through the small size of the units, their behavior remains manageable and the test can be defined specifically, whereby in an error case the error cause can be located quickly. Unit tests are an example for white-box tests as they are commonly implemented and run by the software developers, who know the internal structure of the software.[44] They represent the knowledge of the developer about the specific unit and give an overview of the usage and functionality of the unit.

Automated testing

In the last decade, the use of automated testing has shown a huge potential to help developers handling complex software systems. This is based on the fact that in large software systems an error cause and its effect can be far away from each other. Finding these errors by debugging is a hard job.

Automated testing differs from dynamical testing as the test run and the check of the results are automated. Therefore expected reference values are needed that allow for a comparison with the current results of the test object. In this way automated testing offers a solution for identifying side effects of a change. When the developer makes changes to one unit, all tests can be run afterwards automatically to see if the tests for other units are effected by the change. A precondition for this way of using tests is a high test coverage of the code. Otherwise a change could effect untested parts of the software.

Many units are not runnable in a stand-alone mode because they depend on the environment they are integrated in. To be able to run them in a test, the original environment is replaced by the *test environment*. If components used in test environments have an internal state, they have to be instantiated to a well-defined state before a test run and reset to a default state after the test run. It is necessary to ensure that errors in the test are caused by the test object and not by the test environment due to an unexpected state. Another essential element of automated testing is the *test driver*. The test driver performs all automated tests and generates the test report.

Automated testing supports the opportunistic programming because of the documenting nature of tests.[49] A new user can run an automated test conveniently for instance in debug mode and learn, how different components are connected. Automated testing hence offers a knowledge base for the usage of the software systemand in this way supports opportunistic programming.[49]

Test coverage

The benefits of automated testing arise with a high *test coverage*. The test coverage measures the percentage of code covered by the execution of all tests. A low test coverage means that large parts of the program are not passed when running all tests and errors in these parts are not detected. The amount of test coverage therefore indicates how much the developer can trust in the existing test cases. The literature distinguishes between three methods to estimate the test coverage:

- The *statement coverage* (C_0) with $C_0 =$ executed SLOC/total SLOC, where SLOC is the source lines of code. The statement coverage is the simplest measure and its significance is debated in the community.[100]

- The *branch coverage* (C_1) with $C_1 =$ executed primitive branches/ total primitive branches. A primitive branch means here, that a conditional statement results in two optional parts of a program that are executed depending on the condition. The branch coverage is a much better measure as it allows to detect errors in branch conditions. Its limits are reached when dealing with loops.[44]

- The *path coverage* (C_2) with $C_2 =$ executed primitive paths / total primitive paths. A primitive path means here one possible combination of statements in a procedure. A procedure can have an unhandleable numbers of paths. Thus the path coverage is the most extensive measure, but it can handle loops. To make the path coverage resonable additional restrictions are needed to reduce the investigated numbers of paths.[44]

2.2 Micromagnetic modeling

In the following, the micromagnetic model as reviewed by Parkin et al. and Cimrák is introduced.[103,104] Then a review of the actual micromagnetic simulator landscape is given. Since the focus of this work is the development of a finite-difference-method (FDM) based micromagnetic simulator, in Sec. 2.2.2 the finite-difference-method as well as in Sec. 2.2.3 the FDM-based discretized micromagnetic model are introduced.

2.2.1 Micromagnetic model

For the description of the magnetic properties of ferromagnetic structures, the widely accepted model is the micromagnetic model. In 1935, Landau and Lifshitz[15] laid the foundation to this theory, with major contributions coming later from Gilbert, Néel, Bloch, Brown, and many others.[14,105–107] Several reviews and books[103,104,108–110] describe this theory in detail. Common to other physical systems, this model describes an energy minimization process, where the magnetization tries to reach the energy minimum. In the micromagnetic model,[107] the magnetization dynamics are described by an ordinary differential equation of the time evolution, the so-called Landau-Lifshitz-Gilbert (LLG) equation.[15] This equation describes the magnetization dynamics caused by an effective field.

Landau-Lifshitz-Gilbert (LLG) equation

The Landau-Lifshitz (LL) -equation describes the motion of the magnetization under the influence of an effective field. It was extended by Gilbert et al.[106,111] to the Landau-Lifshitz-Gilbert equation, where the phenomenological Gilbert-damping was added. This extension allowed to describe the experimentally observable damping in ferromagnetic structures.

The implicit Landau-Lifshitz-Gilbert equation is given by[104]

$$\frac{d\vec{M}}{dt} = -\gamma\vec{M}\times\vec{H}_{\text{eff}} + \frac{\alpha}{M_s}\vec{M}\times\frac{d\vec{M}}{dt} \tag{2.1}$$

with the magnetization $\vec{M}$, the gyromagnetic ratio γ, the Gilbert damping parameter $\alpha \geq 0$, the saturation magnetization M_s, and the effective field $\vec{H}_{\text{eff}}$. As shown in Fig. 2.2 the LLG describes a damped precession of the magnetization around the effective field. Equation (2.1) can be written in the explicit form

$$\frac{d\vec{M}}{dt} = -\gamma'\vec{M}\times\vec{H}_{\text{eff}} - \frac{\alpha\gamma'}{M_s}\vec{M}\times\left(\vec{M}\times\vec{H}_{\text{eff}}\right) \tag{2.2}$$

with the abbreviation $\gamma' = \gamma/(1+\alpha^2)$.

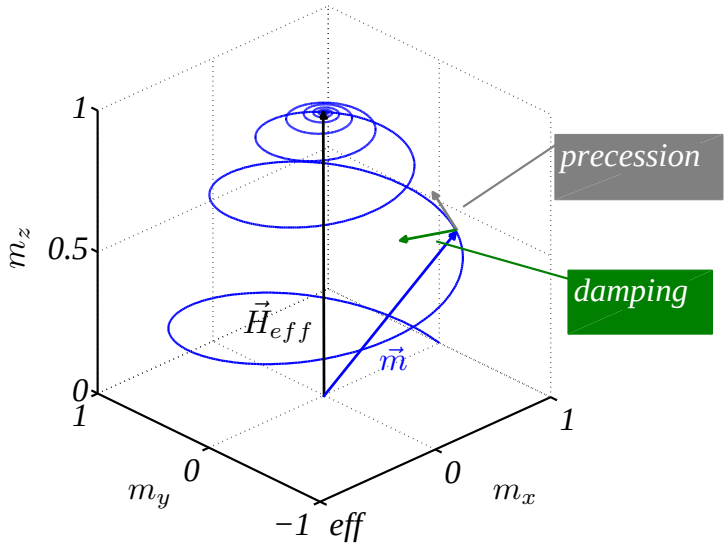


Figure 2.2: Trajectory of the normalized magnetization $\vec{m} = \vec{M}/M_s$ due to an effective field H_{eff}. The magnetization performs a damped precession around the effective field.

Effective field

In the micromagnetic model the effective field is a superposition of the external or Zeeman field and the intrinsic fields. The intrinsic fields consist of the crystalline anisotropy, the demagnetization, and the exchange field. These fields are material and geometry dependent. With these four contributions to the effective magnetic field, it is possible to describe most of the experimentally observed magnetic behavior.[109] As this thesis restricts its investigations to the material Permalloy which has no crystalline anisotropy, this field is excluded from the further introduction. Parkin et al.[103] explain that the effective field can be derived from the total magnetic energy according to

$$\vec{H}_{\text{eff}} = -\frac{1}{\mu_0}\frac{\delta E}{\delta \vec{M}}. \tag{2.3}$$

here μ_0 is the magnetic permeability of the vacuum.

Exchange field

The exchange field is of quantum mechanical origin and is usually described as[104,112–114]

$$E_{\text{ex}} = -\frac{A}{M_s^2}\int_V (\nabla \vec{M})^2 d^3r. \tag{2.4}$$

As given by Eq. (2.4), the exchange energy depends on the spatial change of the magnetization direction. The exchange-energy minimum is reached, when all magnetic moments are aligned parallel. Equation (2.3) results in the exchange field

$$\vec{H}_{\text{ex}} = \frac{2A}{\mu_0 M_s^2} \nabla^2 \vec{M}. \tag{2.5}$$

where A is the exchange coupling constant and M_s is the saturation magnetization. For this field the exchange length of $\Lambda = \sqrt{2A/\mu_0 M_s^2}$ defines the length scale.[103,104]

Demagnetization field

The demagnetization field represents the magnetostatic interaction of the elementary magnetic moments within the magnetic body over long distances.[103,104] This energy is given by

$$E_{\text{demag}} = -\frac{\mu_0}{2} \int_V \vec{M}(\vec{r}) \vec{H}_{\text{demag}}(\vec{r}) d^3 r. \tag{2.6}$$

The corresponding demagnetization field is given by

$$\vec{H}_{\text{demag}}(\vec{r}) = -\frac{1}{4\pi} \int_V (\nabla \vec{g}(\vec{r} - \vec{r}')) \vec{M}(\vec{r}') d^3 r'. \tag{2.7}$$

where $\vec{g} = \vec{r}/|\vec{r}|^3$, and V is the volume of the sample. The demagnetization field forces the magnetization to align parallel to the surface of the ferromagnetic sample to avoid surface charges. The exact calculation of $\vec{g}$ depends on the discretization method. Equation (2.7) describes a spatial convolution of $\nabla \vec{g}$ and $\vec{M}$.

Zeeman field

The Zeeman field is an external field and can have different sources. Thus its exact description depends on the concrete setup. In the general form it is given by

$$\vec{E}_{\text{Zeeman}} = -\frac{1}{\mu_0} \int_V \vec{H}_{\text{Zeeman}} \cdot \vec{M}. \tag{2.8}$$

where $\vec{H}_{\text{Zeeman}}$ is the Zeeman field.[103,104]

2.2.2 Discretization: finite difference method

Since the focus of this thesis lies on finite-difference-method based simulators, this subsection the finite difference method (FDM).

FDM is a method to solve partial differential equations numerically. The basic idea of FDM is to discretize the reference function $f(x)$ at discrete grid points x_i and to replace the spatial derivatives by finite differences[115,116] between the grid points. Many FDM-based micromagnetic simulators[117–122] use a regular grid as it allows to apply fast convolution methods for the calculation of the demagnetization field as explained in Sec. 2.2.3. For each grid point a corresponding volume is needed. Such a regular grid is shown in Fig. 2.3. Here the space is subdivided into a regular grid, where the function value is assumed to be located at the center of the cuboid.

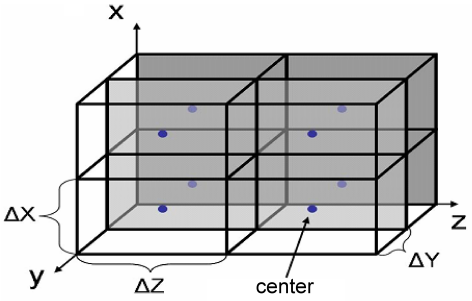


Figure 2.3: Resulting regular grid of cuboid using the finite difference method. The spatially resolved function value is assumed to be located at the center of each cuboid.

First- and second- order derivatives

In the micromagnetic model the calculation of the gradient and the Laplacian of the magnetization is necessary. The components of the gradient as well as the Laplacian are given by the spatial derivatives in each direction. In the following the calculation of the first and second derivative of the one dimensional function $f(x)$ is explained.
Based on the Taylor expansion[115] the first derivative from the two point central, forward, or backward approximation of the derivative is given by

$$f'(x) = \frac{f(x+\Delta x) - f(x-\Delta x)}{2\Delta x} + O(\Delta x^2), \tag{2.9}$$

$$f'(x) = \frac{f(x+\Delta x) - f(x)}{\Delta x} + O(\Delta x), \tag{2.10}$$

$$f'(x) = \frac{f(x) - f(x-\Delta x)}{\Delta x} + O(\Delta x). \tag{2.11}$$

here $O(\Delta x)$ indicates the error. The forward and backward cases are used, when x is at the border of the sample.

The second derivative can be derived from the Taylor expansion as well. The three point central, forward, and backward approximation are given by

$$f''(x) = \frac{f(x+\Delta x) - 2f(x) + f(x-\Delta x)}{\Delta x^2} + O(\Delta x^2), \tag{2.12}$$

$$f''(x) = \frac{f(x+2\Delta x) - 2f(x+\Delta x) + f(x)}{\Delta x^2} + O(\Delta x^2), \tag{2.13}$$

$$f''(x) = \frac{f(x-2\Delta x) - 2f(x-\Delta x) + f(x)}{\Delta x^2} + O(\Delta x^2). \tag{2.14}$$

A detailed discussion of more accurate approximations of the first and second derivative can be found in.[115,116,123,124] In the following the approximations of the gradient and the Laplacian using Eq. (2.9) - (2.11) as it is used in many micromagnetic simulators is presented.

Gradient and Laplacian on regular grids

The gradient of a scalar field for a specific grid point is given by

$$\nabla f(\vec{r}_{i,j,k}) = \begin{pmatrix} \partial_x f(\vec{r}_{i,j,k}) \\ \partial_y f(\vec{r}_{i,j,k}) \\ \partial_z f(\vec{r}_{i,j,k}) \end{pmatrix}. \tag{2.15}$$

The gradient can be calculated from Eq. (2.15) by replacing $\partial_d f(\vec{r}_{i,j,k})$ by the approximation given by Eq. (2.9), where $d \in \{x, y, z\}$ denotes the direction. This results in

$$\nabla f(\vec{r}_{i,j,k}) = \begin{pmatrix} \frac{f(\vec{r}_{i+1,j,k}) - f(\vec{r}_{i-1,j,k})}{2\Delta x} \\ \frac{f(\vec{r}_{i,j+1,k}) - f(\vec{r}_{i,j-1,k})}{2\Delta y} \\ \frac{f(\vec{r}_{i,j,k+1}) - f(\vec{r}_{i,j,k-1})}{2\Delta z} \end{pmatrix}, \tag{2.16}$$

where $\vec{r}_{i+1,j,k} = (x + \Delta x, y, z)$ is the next grid cell in x - direction, $\vec{r}_{i,j+1,k} = (x, y + \Delta y, z)$ is the next grid cell in y - direction, and $\vec{r}_{i,j,k+1} = (x, y, z + \Delta z)$ is the next grid cell in z - direction.

The Laplacian of a scalar field for a specific grid point is given by

$$\nabla^2 f(\vec{r}_{i,j,k}) = \partial_x^2 f(\vec{r}_{i,j,k}) + \partial_y^2 f(\vec{r}_{i,j,k}) + \partial_z^2 f(\vec{r}_{i,j,k}). \tag{2.17}$$

The Laplacian can be calculated from Eq. (2.17) in the same manner as shown in Eq. (2.16) by replacing $\partial_d^2 f(\vec{r}_{i,j,k})$ by Eq. (2.12) - (2.14).

Three-dimensional convolution on regular grids

For the implementation of the demagnetization field it is necessary to calculate a three-dimensional convolution. The three-dimensional convolution of the functions f and g is

given by:[125]

$$h(x,y,z) = \int\int\int f(x-x',y-y',z-z') \cdot g(x',y',z')dx'dy'dz'. \tag{2.18}$$

In discretized form this results in

$$h(x_i,y_j,z_k) = \sum_{k'}\sum_{j'}\sum_{i'} f(x_i - x_{i'}, y_j - y_{j'}, z_k - z_{k'}) \cdot g(x_{i'}, y_{j'}, z_{k'}). \tag{2.19}$$

If using a regular grid, the convolution can be calculated by the fast convolution method, which is based on the fast Fourier transformation (FFT) and is given by

$$h(x,y,z) = \mathscr{F}^{-1}\{\mathscr{F}\{f(x,y,z)\} * \mathscr{F}\{g(x,y,z)\}\}, \tag{2.20}$$

where $S = \mathscr{F}\{\}$ is the Fourier transformation of the function s and $\mathscr{F}^{-1}\{S\}$ is the inverse Fourier transformation of the function S.

2.2.3 Discretized model

In this sub-section the finite difference method is applied to the micromagnetic model. Fiedler and Schrefl[126] summarize the use of FDM as *"Replacing both space and time derivatives by their FD approximations ... is called an explicit-type marching process"*.

To understand the time and space complexity for the micromagnetic model calculation, the spatially dependence of the LLG and each field are listed. To increase the readability the following abbreviations are used if possible:

- $\vec{M}_{i,j} = \vec{M}(t_i, \vec{r}_j)$ is the magnetization at the i-th time step t_i and the position of the j-th cell $\vec{r}_j$. Here the cell index j is a linearization of the cell indices (j_x, j_y, j_z).
- $\vec{M}_{i,j+\Delta x} = \vec{M}(t_i, \vec{r}_j + \vec{\Delta x})$ is the magnetization at the i-th time step t_i and the position of cell $\vec{r} = \vec{r}_j + (\Delta x, 0, 0)$. This is defined in the same manner for Δy and Δz.
- For the spatial dependency *all* indicates all cells, and *nn* indicates the next neighbors to the j-th cell.

In the following the discretization of the LLG and the intrinsic fields are discussed in more detail.

Landau-Lifshitz-Gilbert (LLG) equation

As introduced in Sec. 2.2, the LLG describes a first order partial differential equation, that can be discretized using the finite-difference method by the separation of the time and spatial

dependency:

$$\frac{d\vec{M}_{i,j}}{dt} = -\gamma' \vec{M}_{i,j} \times \vec{H}_{\text{eff},i,j} - \frac{\gamma' \alpha}{M_s} \vec{M}_{i,j} \times \vec{M}_{i,j} \times \vec{H}_{\text{eff},i,j}. \tag{2.21}$$

The separation of the time and spatial dependence necessitates to choose time steps that are sufficiently small. Otherwise a stable solution cannot be obtained in all cases.[126]

Commonly explicit Runge-Kutta algorithms of forth- and fifth- order[127,128] as well as implicit Gauß-Seidel solvers[129] with an adaptive time-step control are used for solving this equation. Since standard ODE solver do not take the constraints of $\left|\vec{M}\right| = M_s$ into account, this aspect has been addressed by several works[130,131] and has been reviewed by Cimŕak.[104] In this thesis a renormalization of the magnetization after each timestep is used.

Effective field

The discretized effective field for a grid point $\vec{r}_j$ is given by

$$\begin{aligned} \vec{H}_{\text{eff},i,j}(\vec{r}_{all}, \vec{M}_{i,all}) =& \vec{H}_{\text{Zeeman},i,j} + \vec{H}_{\text{exch},i,j}(\vec{r}_{nn}, \vec{M}_{i,j}, \vec{M}_{i,nn}) \\ &+ \vec{H}_{\text{demag},i,j}(\vec{r}_{all}, \vec{M}_{i,all}). \end{aligned} \tag{2.22}$$

It is a superposition of all magnetic fields at that grid point. Due to the demagnetization field, the effective field depends on all cell positions $\vec{r}_{all}$ and all magnetization values $\vec{M}_{i,all}$.

Exchange field

The exchange field is approximated by the Laplacian of the magnetization as given by Eq. 2.5. As shown by Donahue et al.[113] the exchange field is accurately approximated using the three-point approximation of the Laplacian Eq. (2.17) resulting in

$$\begin{aligned} \vec{H}_{\text{exch},i,j}(\vec{r}_j, \vec{M}_{i,j}, \vec{M}_{i,nn}) =& \frac{2A}{{M_s}^2 \mu_0} \\ & \Big(\frac{\vec{M}_{i,j+\Delta x} - 2\vec{M}_{i,j} + \vec{M}_{i,j-\Delta x}}{\Delta x^2} \\ & + \frac{\vec{M}_{i,j+\Delta y} - 2\vec{M}_{i,j} + \vec{M}_{i,j-\Delta y}}{\Delta y^2} \\ & + \frac{\vec{M}_{i,j+\Delta z} - 2\vec{M}_{i,j} + \vec{M}_{i,j-\Delta z}}{\Delta z^2} \Big) \\ =& \frac{2A}{{M_s}^2 \mu_0} \sum_{k \in nn} \frac{\vec{M}_{i,k} - \vec{M}_{i,j}}{|\vec{r}_k - \vec{r}_j|^2} \end{aligned} \tag{2.23}$$

where μ_0 is the permeability of vacuum, A is the material dependent exchange constant, and $nn = \{\pm\Delta x, \pm\Delta y, \pm\Delta z\}$. Due to the physical origin of the exchange field, this approximation

is valid only if $|\vec{r}_k - \vec{r}_j|$ is significantly below the exchange length Λ and the angular change of the magnetization between two neighboring points is below a maximum angle.[114]

Donahue et al.[114] summarized the possible solutions in order to consider the boundary of the sample as it was used by others.[132,133] In this work the boundaries are taken into account by $\partial_i H_{\text{exch}} = 0$. The resulting three-point approximation of the Laplacian[114] for a boundary cell at $\vec{r}_j$, where the neighboring cell at $\vec{r}_{j+\Delta x}$ is outside, results in

$$\begin{aligned}\vec{H}_{\text{exch},i,j}(\vec{r}_j, \vec{M}_{i,j}, \vec{M}_{i,nn}) = &\frac{2A}{{M_s}^2 \mu_0} \\ &(-\frac{\vec{M}_{i,j} - \vec{M}_{i,j-\Delta x}}{\Delta x^2} \\ &+\frac{\vec{M}_{i,j+\Delta y} - 2\vec{M}_{i,j} + \vec{M}_{i,j-\Delta y}}{\Delta y^2} \\ &+\frac{\vec{M}_{i,j+\Delta z} - 2\vec{M}_{i,j} + \vec{M}_{i,j-\Delta z}}{\Delta z^2})\end{aligned} . \tag{2.24}$$

Demagnetization field

The demagnetization field is given by

$$\vec{H}_{\text{demag},i,j}(\vec{r}_{all}, \vec{M}_{i,all}) = \sum_{k \in all} \hat{N}(\vec{r}_j - \vec{r}_k, \tau_j, \tau_k) \cdot \vec{M}_{i,k}. \tag{2.25}$$

It describes a spatial convolution of the magnetization with the so-called demagnetization tensor. Here $\hat{N}(\vec{r}_j - \vec{r}_k, \tau_j, \tau_k)$ is the demagnetization tensor for two cuboidal ferromagnets separated by the distance vector $\vec{R} = \vec{r}_j - \vec{r}_k$, τ_j is the volume of the j-th cuboid, and τ_k is the volume of the k-th cuboid. The demagnetization tensor for the cuboid at point $\vec{r}_i$ is given by

$$\hat{N}_{jk}(\vec{r}_j - \vec{r}_k, \tau_j, \tau_k) = \frac{1}{4\pi\tau_j} \int_{\tau_j} \int_{\tau_k} \nabla_j \nabla_k (\frac{1}{|\vec{r}_j - \vec{r}_k|}) d\tau_j d\tau_k. \tag{2.26}$$

Newell et al.[134] showed how to solve Eq. (2.26) for such two separated cuboidal ferromagnets. Their solution can conveniently be applied to grids of cuboidal ferromagnets and thus can be used to calculate the demagnetization tensor for FDM-based discretizations of a ferromagnetic structure. In general this way of calculating the demagnetization field is expensive, as the demagnetization tensor has to be calculated for each possible distance vector between two cuboidal ferromagnets. For an irregular grid discretized by N cells this results in N^2 distance vectors. Using a regular grid reduces the number of distance vectors between all grid points significantly to $N_{DV} = (2p_x - 1) \cdot (2p_y - 1) \cdot (2p_z - 1)$, where p_d is the number of grid points in the x, y, z - direction. This allows to calculate an extended demagnetization tensor of dimensions $(P_x, P_y, P_z) = (2p_x - 1, 2p_y - 1, 2p_z - 1)$ for all distance vectors. The temporal independency of the tensor allows to reduce the number of calulations to one and so to reduce the time and space complexity.[103] The tensor for a grid point is then given

by the corresponding window of the extended tensor.

A regular grid also allows to calculate the convolution using fast convolution algorithms. The corresponding algorithm is exemplary depicted for the quasi 2D case (only one layer in z-direction) in Fig. 2.4. As explained in detail in the caption of this figure, this algorithm is given mainly by five steps. The previously mentioned windowing of the demagnetization tensor, which is necessary in the direct convolution algorithm, was not applicable for the fast calculation algorithm. Hence the expanded demagnetization field includes physically invalid regions, where the magnetization and the corresponding demagnetization tensor overlapped only partially. Thus in the final step of this algorithm the physically valid region of the expandend demagnetization field needs to be cut out.

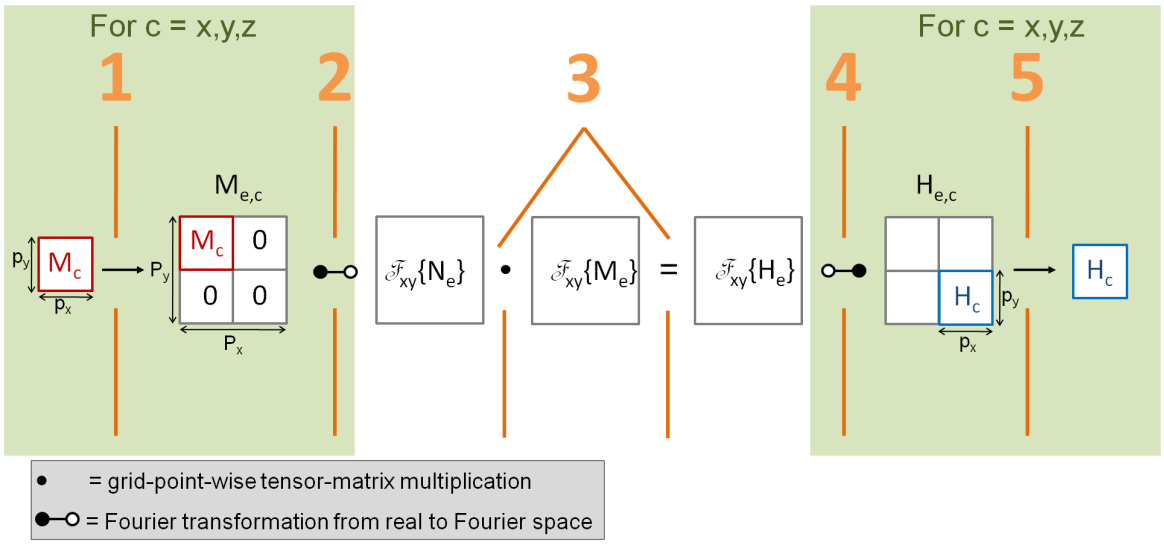


Figure 2.4: Scheme of the necessary steps to calculate the demagnetization field in the quasi 2D case (only one layer in z-direction). In step 1 each component of the magnetization is expanded to (P_x, P_y) to fit to the size of the expanded demagnetization tensor. In step 2 the expanded magnetization is transformed to the Fourier space. In step 3 the expanded demagnetization field is calculated in the Fourier space by multiplying the expanded magnetization and the expanded demagnetization tensor in the Fourier space. In step 4 the expanded demagnetization field is transformed back into the real space. In step 5 finally the physically valid region of the expandend demagnetization field is selected including the desired demagnetization field.

2.2.4 Extended models for current interaction

One goal of the present work is to integrate the spin-transfer torque and the anisotropic magnetoristivity (AMR) effect into M^3S. In this sub-section the physical models describing this two phenomena are introduced.

In 1996 it was predicted[8,9] that a spin-polarized current flowing through a ferromagnetic conductor can apply a significant torque to its magnetization. Theoretical extensions of the micromagnetic model have been proposed for two cases. The first case is given by a current flowing through a ferromagnetic sample, where the magnetization can change continuously. This case is called *spin-transfer torque in continuously variable magnetization*. The second case is given by a current flowing through a ferromagnetic multilayer system. The multilayer system is usually denoted as a spin valve. Here the magnetization changes discontinuously. This case is called *spin-transfer torque in a spin valve*.

The spin transfer torque describes only one direction of the interaction between a current flow and the magnetization. It has been shown that the difference in the relative orientation between the magnetization and the current, leads to local changes in the resistivity and so to changes in the current direction. In a system with continuously variable magnetization this leads to the AMR effect, while in a spin valve the giant magneto-resistivity (GMR) or tunnel magneto-resistivity (TMR). Concerning the effects of a current on the magnetization, this thesis focuses on the spin transfer torque in continuously variable magnetization and the AMR effect.

In the following all extensions of the LLG are given in implicit and explicit expression. The implicit expression is often more intuitive, while the explicit expression is implemented in the following.

Spin-transfer torque in a spin valve

A spin valve is a multilayer system, consisting basically of two ferromagnetic layers that are connected by a nonmagnetic spacer as shown in Fig. 2.5.
In such a magnetic multilayer system the magnetization changes abruptly at the connecting interfaces of the magnetic layers. For a spin valve where the currents flow perpendicular to the plane (CPP), Slonczewski[8,16] has introduced a spin-transfer torque extension to the original LLG, which then became the so called Landau-Lifshitz-Gilbert-Slonczewski equation (LLGS)[8,16,43,135,136] and is given by

$$\frac{d\vec{M}}{dt} = -\gamma\vec{M}\times\vec{H}_{\text{eff}} - \frac{\gamma a_j}{M_s}\vec{M}\times\left(\vec{M}\times\vec{p}\right) + \frac{\alpha}{M_s}\vec{M}\times\frac{d\vec{M}}{dt}. \tag{2.27}$$

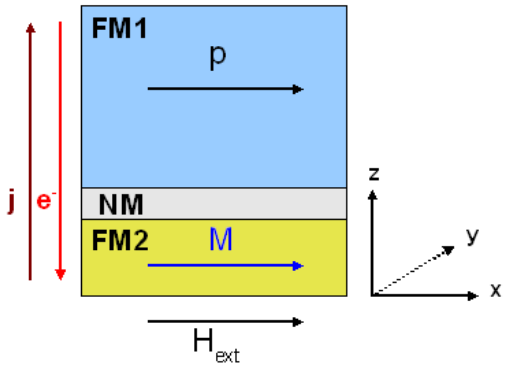


Figure 2.5: Sketch of a spin valve. The electrons flow in $-z$-direction and cross the fixed ferromagnetic layer FM1 first. FM1 polarizes the current in the direction of its magnetization called $\vec{p}$. The spin-polarized current influences the second ferromagnetic layer FM2 via the spin-transfer torque.

Here a_j is the current density dependent coupling constant between the current and the magnetization. Equation (2.27) is given in its explicit form as

$$\begin{aligned}\frac{d\vec{M}}{dt} = &-\gamma\vec{M}\times\vec{H}_{\text{eff}} - \frac{\gamma\alpha}{M_s}\vec{M}\times\left(\vec{M}\times\vec{H}_{\text{eff}}\right)\\ &-\frac{\gamma a_j}{M_s}\vec{M}\times\left(\vec{M}\times\vec{p}\right) + \gamma\alpha a_j\vec{M}\times\vec{p}.\end{aligned} \tag{2.28}$$

The spin-transfer torque in a spin valve originates from the interaction of the spin-polarized current with the local magnetic moments at the interface between the ferromagnet FM2 and the spacer. The ferromagnetic layer FM1, called the fixed layer, is designed to be unaffected by the spin-transfer torque. In reality, this is achieved by exchange-coupling of FM1 to additional layers, e.g. anti-ferromagnets. FM1 then serves as a source for the spin-polarized current. All electrons passing through this layer become polarized equal to its magnetization direction $\vec{p}$.

Spin-transfer torque in continuously variable magnetization

This type of spin-transfer torque describes magnetization dynamics within a ferromagnet with continuously variable magnetization[137] as shown in Fig. 2.6. The magnetization is excited by a spin-polarized current. The additional torque, called spin-transfer torque for such a system arises from the interaction of the spin-polarized current with the local magnetic moments within the ferromagnet. The itinerant electrons align their spin with the spins of the local electrons that constitute the magnetization. This torque on the moving electrons must be compensated by an opposite torque on the local magnetization to conserve the total momentum. The basic micromagnetic model was extended by the spin-transfer torque by

Bazaliy et al.[19] As proposed by Zhang and Li[20] it is given by

$$\begin{aligned}\frac{\partial \vec{M}}{dt} =& -\gamma \vec{M} \times \vec{H}_{\text{eff}} + \frac{\alpha}{M_s} \vec{M} \times \frac{d\vec{M}}{dt} \\ & - \frac{b_j}{M_s^2} \vec{M} \times \left(\vec{M} \times (\vec{j} \cdot \vec{\nabla}) \vec{M} \right) \\ & - \xi \frac{b_j}{M_s} \vec{M} \times (\vec{j} \cdot \vec{\nabla}) \vec{M}\end{aligned} \tag{2.29}$$

with the gyromagnetic ratio γ, the Gilbert damping parameter α, the saturation magnetization M_s, and the effective field $\vec{H}_{\text{eff}}$, as introduced in Sec. 2.2.3. The coupling constant between the current and the magnetization is $b_j = (P\mu_B)/(eM_s(1+\xi^2))$, where P denotes the spin polarization of the current density $\vec{j}$, μ_B the Bohr magneton, and $\xi = \tau_{\text{ex}}/\tau_{\text{sf}}$ the degree of non-adiabaticity, which is the ratio between the exchange relaxation time τ_{ex} and the spin-flip relaxation time τ_{sf}. The explicit form of Eq. (2.29) is given by

$$\begin{aligned}\frac{d\vec{M}}{dt} =& -\gamma' \vec{M} \times \vec{H}_{\text{eff}} - \frac{\alpha\gamma'}{M_s} \vec{M} \times \left(\vec{M} \times \vec{H}_{\text{eff}} \right) \\ & - \frac{b_j'}{M_s^2} (1 + \alpha\xi) \vec{M} \times \left(\vec{M} \times (\vec{j} \cdot \vec{\nabla}) \vec{M} \right) \\ & - \frac{b_j'}{M_s} (\xi - \alpha) \vec{M} \times (\vec{j} \cdot \vec{\nabla}) \vec{M}\end{aligned} \tag{2.30}$$

with the abbreviations $\gamma' = \gamma/(1+\alpha^2)$ and $b_j' = b_j/(1+\alpha^2)$ as introduced by Krüger et al.[138]

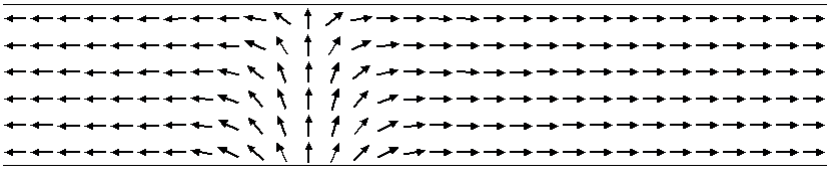


Figure 2.6: Magnetic wire as an example for a system with continuously variable magnetization. In the magnetic wire the magnetization changes continuously from the left to the right.

Magnetization dependent current distribution and AMR effect

In a ferromagnetic thin film element, the spin transfer torque is the effect of a current flow on the magnetization. But the magnetization also affects the current through the anisotropic magneto-resistance (AMR). The AMR effect leads to local resistance changes and to a local reduction of the current density. This causes a locally reduced spin-transfer torque acting on the magnetization dynamics. In turn, the magnetization influences the local resistivity. As a result, the mutual influence of current and magnetization causes non-linear effects in the linear regime of electron transport. The electronic transport can be treated classically and

calculated quasi-statically from a local version of Ohm's law

$$\vec{j}(\vec{r}) = \sigma(\vec{r})\vec{E}(\vec{r}), \tag{2.31}$$

while local charge neutrality is considered, $\nabla_{\vec{r}}\vec{j}(\vec{r}) = 0$

$$\nabla\vec{j}(\vec{r}) = \nabla\left[\sigma(\vec{r})\nabla\Phi(\vec{r})\right] = 0. \tag{2.32}$$

The influence of the magnetization on the current flow is incorporated in Eq. (2.32) via a magnetization-dependent conductivity tensor $\sigma(\vec{r}) = \sigma(\vec{M}(\vec{r}))$. The shape of the conductivity tensor accounts for the AMR, such that the resistivity locally obeys the relation

$$\rho = \rho_{\perp} + \Delta\rho\cos^2(\measuredangle(\vec{j},\vec{M})), \tag{2.33}$$

which reflects the dependence of the resistance on the angle between local current and magnetization. The AMR ratio in thin-film elements

$$\rho_{\mathrm{AMR}} = \frac{\rho_{||} - \rho_{\perp}}{\rho_{||} + \rho_{\perp}} \equiv \frac{\Delta\rho}{\rho_{||} + \rho_{\perp}} \tag{2.34}$$

characterizes the strength of the AMR effect. The material parameters $\rho_{||}$ ($\rho_{\perp}$) are the resistances for the sample being saturated due to an external magnetic field parallel (perpendicular) to the current flow. Thus, the anisotropic magneto-resistivity $\Delta\rho$ is the change in resistance between a parallel and a perpendicular magnetization with respect to the applied current.

2.3 Micromagnetic simulator landscape

There exist a variety of micromagnetic simulators that can be split in groups by the underlying discretization method and by the user license.[117–122,139–143] In the following an overview of existing micromagnetic simulators is given focussing on the tools *Magpar*, *OOMMF*, and *Nmag*. This section gives an overview of existing standard problems. A standard problem is a system test identified by the micromagnetic community and collected on the μMag webpage.[40]

2.3.1 Existing micromagnetic simulators

There exist open source and commercial as well as finite-difference-method (FDM) and finite-element-method (FEM) based simulators. Concerning the discretization, FDM-based simulators like the Object Oriented Micromagnetic Framework (*OOMMF*)[121] are in general faster and need less memory than FEM-based tools[103] like *Magpar*,[142] or *Nmag*.[143] But FDM-based simulators are used for samples, where the shape can be described by a regular grid of cuboid. As observed by several groups, surface roughness has a large effect on the dynamics,[144,145] which means the modeling of real experimental setups normally necessitates the use of FEM-based simulators. Hence, the choice of a simulator depends on the accuracy and runtime-performance requirements of the concrete problem.

Besides the discretization method and the user licence, many other requirements influence the choice of a simulator. Table 2.1 gives an overview of existing open-source and public-code tools focussing on the properties: necessary user licence, scripting support, used discretization method, support for parallel execution, used programming language, interfaced libraries. This table reveals that the use of numerical libraries and the support for parallel execution of simulations in the micromagnetic community is capable of improvement. Further scripting is only supported by half of the tools. To depict the current state of micromagnetic simulators in the following the well-established tools *Magpar*, *OOMMF*, and *Nmag* are reviewed in detail.

name	license	scripting	method	parallel	programming language	libraries
AlaMag[119]	GPL	-	FD	-	C++	-
JaMM[120]	PDC	-	FD	-	Java/XML	-
OOMMF[121]	open source	yes	FD	SMP	C++, *TCL/TK*	VODE
RKMAG[122]	open source	no	FD	no	FORTRAN	Intel MKL
MagFEM3D[141]	GPL	Unknown	FE	-	FORTRAN	-
Magpar[142]	GPL	TAO, Python	FE	MPI	C++	TAO, PVODE, Sundials PETSc
Nmag[143]	GPL	yes	FE	MPI	Python OCAML	PVODE, Sundials, PETSc, HLib

Table 2.1: List of existing license free micromagnetic simulators. For each simulator the type of license, the scripting language support, the used discretization method, the support for parallel computing, the basic programming language, and the used numerical libraries are listed.

Magpar

Magpar is a finite-element micromagnetics package which combines several unique features:[142,146,147]

- Applicability to a variety of static and dynamic micromagnetic problems including uniaxial anisotropy, exchange and magnetostatic interactions, and external fields.
- Flexibility of the finite-element method concerning the geometry and accuracy by using unstructured graded meshes.
- Availability due to its design based on free, open source software packages.
- Portability to different hardware platforms, which range from personal computers to massively parallel supercomputers.
- Scalability due to its highly optimized design and efficient libraries
- Versatility by including static energy minimization and dynamic time integration methods.

Magpar uses the well-established numerical finite element libraries PETSc[148] and *TAO*[149] as well as parallel ordinary equation solvers.[150,151] To specify the simulation problem several files have to be prepared.[146] In the file *allopt.txt* as shown in Code listing 2.1 all simulation parameters are specified. Each parameter is specified by a separate command option `-optionname optionvalue`. User specific scripts written in *TAO*[149] or *Python* can be specified for well defined options like for instance the exact calculation of the external magnetization. In addition the files *project.krn*, *project.inp*, and *project.0001.inp* including the material properties, the finite element mesh, and the initial magnetization distribution need to be prepared. 2.2 By contract all files have to be placed in the same directory. There also exists a graphical user interface that helps to prepare these files. A simulation finally can be started calling the command `magpar.exe`. As an example the necessary configuration files to run the Larmor-precession test specified (for details see Sec. 2.3.2) using *Magpar* are listed in the following in Code listing 2.1, 2.2, and 2.3.

```
-simName sphere
-meshtype 1
-size 10e-9
-init_mag 4
-mode 0
-demag 0
-hextini 1000
-ts_max_time 0.03
```

Code listing 2.1: allopt.txt

```

686 3237 3 0 0
1 1.0 0.0 0.0
2 0.0 1.0 0.0
...
...
685 1.0 0.0 0.0
686 1.0 0.0 0.0
```

Code listing 2.2: shere.inp

```

0.0   0.0   0.0      0.0     1.0  1e-11    0.0   uni
#
# theta phi   K1       K2      Js   A        alpha psi   # parameter
# (rad) (rad) (J/m^3)  (J/m^3) (T)  (J/m)    (1)   (rad) # units
```

Code listing 2.3: shere.krn

OOMMF

OOMMF (Object Oriented Micromagnetic Framework) was first released on January 15, 1998. This toolkit is written in *TCL/TK*[152,153] and *C++*.[154] In addition to the Oxs tool that performs the simulation, *OOMMF* offers control and visualization tools that communicate via TCP/IP. The Oxs tool is mainly written in *C++*. The other tools as well as the GUI are written in *TCL/TK*. To perform a simulation it is necessary to write a configuration file interfacing the *C++* based *OOMMF* core using *TCL/TK*. *OOMMF* offers three distinct levels to modify the code:[121]

- *"At the top level, individual programs interact via well-defined protocols across network sockets"*.
- *"The second level of modification is at the TCL/TK script level. Some modules allow TCL/TK scripts to be imported and executed at run time, and the top level scripts are relatively easy to modify or replace"*.
- *"At the lowest level, the C++ source is provided and can be modified"*. There are third party modules offering interfaces to *VODE*[91] and *VTK*.[155,156] But *OOMMF* originally interfaces to no numerical or scientific library.

The *OOMMF* user's guide[121] explains the reasons for this architecture as:

> *"The goal of the OOMMF project is to develop a portable, extensible public domain micromagnetic program and associated tools. This code will ... have a well documented, flexible programmer's interface so that people developing new code can swap their own code in and out as desired.*
> ...
> *In order to allow a programmer not familiar with the code as a whole to add modifications and new functionality, we feel that an object oriented approach is critical, and have settled on C++ as a good compromise with respect to availability, functionality, and portability. In order to allow the code to run on a wide variety of systems, we are writing the interface and glue code in TCL/TK."*

To specify the simulation problem a *.mif* file has to be prepared.[121] In this file all parameters are specified by creating the corresponding *C++*-Objects using *TCL/TK*. If necessary additional files can be addressed in the *.mif* file to include the material properties, and the initial magnetization distribution. To run a simulation, the Oxsii-tool is started either in bash mode or in a graphical user interface. By loading the *.mif* file the simulation is started. Code listing 2.4 shows the *.mif* file to run the Larmor-precession test explained in Sec. 2.3.2.

```
# MIF 2.1
set pi [expr 4*atan(1.0)]
set mu0 [expr 4*\$pi*1e-7]

Specify Oxs_BoxAtlas:atlas {
    xrange {0 3e-9}
    yrange {0 3e-9}
    zrange {0 3e-9}
}

Specify Oxs_RectangularMesh:mesh {
    cellsize {3e-9 3e-9 3e-9}
    atlas :atlas
}

Specify Oxs_UZeeman:field [subst {
    multiplier [expr {1e6}]
    Hrange {
    {1 1 0 1 1 0 0}
    }
}]

Specify Oxs_EulerEvolve {
    alpha 0.0
    do_precess 1
    start_dm 0.01
}

Specify Oxs_TimeDriver [subst{
    basename larmor
    evolver Oxs_EulerEvolve
    stopping_time  300e-12
    mesh :mesh
    stage_count 1
    stage_iteration_limit 0
    total_iteration_limit 0
    Ms { Oxs_UniformScalarField { value [expr[1e6/mu0] } }
    m0 { Oxs_UniformVectorField {
    norm 1
    vector {1 1 1}
    }}
}]
```

Code listing 2.4: *OOMMF-.mif* file that defines the Larmor-precession test simulation. As the Larmor-precession test is a boundary independent simulation, instead of a sphere one rectangular cells is used.

Nmag

Nmag is a micromagnetic software written in *Python* and *OCAML*. It uses the well established libraries *PETSc*[148] and *Sundials*[150] to implement the micromagnetic model based on the finite element method. Averaged-field results are stored in the *.ndt*[157] file format following the *.odt* file format esteblished by *OOMMF*.[121] Spatially resolved data are stored in the *HDF5* file format,[158] so that the results can be plotted using *VTK*[155] and three-dimensional visualization tools like *MayaVi*.[159] Further *Nmag* offers automated parallelization of user specific *Nmag* scripts.[160] The goal of the *Nmag* project is to develop a tool:[143]

> *"that handles specifications of micromagnetic systems at a sufficiently abstract level to enable users with little programming experience to automatically translate a description of a large class of dynamical multi-field equations plus a description of the system's geometry into a working simulation. Conceptually, this is a step toward a higher-level abstract notation for classical multi-field multi-physics simulations."*

Concerning the architecture of *Nmag*, the main advantages of this approach are:[160]

> *"first, we do not gradually evolve another ad-hoc (and potentially badly implemented) special purpose programming language. Second, by drawing upon the capabilities of a well supported existing framework for flexibility, we get a lot of additional power for free: the user can employ readily available and well supported Python libraries for tasks such as data post-processing and analysis, e.g. generating images for web pages etc. In addition to this, some users may benefit from the capability to use Nmag interactively from a command prompt, which can be very helpful during the development phase of an involved simulation script."*

To specify the simulation problem a *Python* script is prepared.[157] In this script the simulation problem is specified by creating the corresponding *Python* objects and starting simulation runs explicitly in the script. If necessary additional files also in *Nmag* can be addressed in the *Python* script to include the material properties, the finite element mesh, and the initial magnetization distribution; see Code listing 2.5 for an example. A simulation run is started by calling `nmag myproblem.py`.

```
import nmag

from nmag import SI, every, at, si

sim = nmag.Simulation(do_demag = False)

Py = nmag.MagMaterial(name="Py",
                      Ms=1.0*si.Tesla/si.mu0,
                      exchange_coupling=SI(13.0e-12, "J/m"),
                      llg_damping = SI(0.0))

sim.load_mesh("sphere1.nmesh.h5",
              [("sphere", Py)],
              unit_length=SI(1e-9,"m"))

sim.set_m([1,1,1])

Hs = nmag.vector_set(direction=[0.,0.,1.],
                     norm_list=[1.0],
                     units=1e6*SI('A/m'))

ps = SI(1e-12, "s")  # ps corresponds to one picosecond

sim.hysteresis(Hs,
               save=[('averages', every('time', 0.1*ps))],
               do=[('exit', at('time', 300*ps))])
```

Code listing 2.5: Example Script for the Larmor-precession test in *Nmag*.

Comparison

In addition to the differences listed in Tab. 2.1 the review of *Magpar*, *OOMMF*, and *Nmag* revealed further important differences in the support for scripting and the parallel execution of a simulation.

- *Magpar* supports MPI but does not offer scripting at all.
- *OOMMF* in contrast has just been parallelized for multicore systems and offers scripting for the creation of a configuration file. In the configuration file objects can be specified that are called during the defined simulation process of *OOMMF*. In this way *OOMMF* provides a more flexible configuration file as *Magpar*. A simulation run can be specified using the full functionality of *TCL/TK*. In contrast to *Magpar* and *Nmag*, *OOMMF* offers a graphical user interface to run a simulation.
- *Nmag* finally offers the parallel execution of complex simulation scripts using MPI. The user can define complex simulation runs like parameter sweeps and hysteresis loops on the basis of a well defined API. In contrast to *Magpar* and *OOMMF*, *Nmag* allows to implement pre- and post-processing steps for a simulation run like the preparation of the initial magnetization, the analyzis of the simulation results, or the automated execution of following simulation runs directly in the simulation script.

In summary *Nmag* offers a clear and flexible framework for the simulation of FEM-based micromagnetic simulations. The same flexibility cannot be found for FDM-based micromagnetic simulations. Allthough *OOMMF* is a well-established micromagnetic simulator, it lacks in the use of numerical standard libraries and the support for scripting.

2.3.2 Existing system tests

An important aspect for the choice of a simulator is its validity. As described above, a variety of micromagnetic simulation tools exist, which use different underlying algorithms. To ensure the correctness and to allow the comparison of these simulators, the Micromagnetic Modeling Activity Group (μMag) has collected system tests or so-called standard problems with a significant behavior.[40] Up to now there exist four standard problems, which have been published by μMag. These problems include the anisotropy, the demagnetization, the exchange, and the Zeeman field. In addition to these problems the Larmor-precession problem is a system test suitable to check the time integration method. From these five system tests standard problem No. 1 as the first standard problem was not appropriate for the comparison of different simulators.[40] Standard problem No. 2[40,161] deals with static micromagnetic simulations and standard problem No. 3 covers anisotropy effects, which are both not the subject of this work. Hence in the following, this work focuses on the Larmor-precession test[146,157] and standard problem No. 4.[40]

Larmor-precession test

This system test can be performed with only one cell as it includes no spatially dependent field. The only contribution to the effective magnetic field is the Zeeman field. As the damping is set to zero this simulation describes an undamped rotation of the magnetization around the Zeeman field. This setup can be used to check the correctness of the LLG implementation and in part the numerical time integration as for the resulting so-called Larmor-precession an analytical solution exists.

The test starts from a magnetization of $\vec{M} = (1,1,1)/\sqrt{3}M_s$, where $M_s = 1$ T $= 1/\mu_0$ A/m. Further simulation parameters are $\alpha = 0$ and $\gamma = 2.210173 \cdot 10^5$. A simulation is performed for 300 ps by applying an Zeeman field of $\vec{H} = (0,0,1 \cdot 10^6)A/m$. From the results the precession frequency is estimated by a sinusoidal fit. The estimated precession period is compared to the expected Larmor precession period of $T = 1/f_{Larmor} = 28.428477$ ps.[146,157]

Standard problem No. 4

Standard problem No. 4 describes a system, that was highly investigated by the micromagnetic community at the time it was published. It focuses on dynamic aspects of micromagnetic simulations.[162,163] The investigated sample is given by an Permalloy thin film of thickness $t = 3$ nm, length $L = 500$ nm, and width $d = 125$ nm as shown in Fig. 2.7.

The system test starts form an initial state that is an equilibrium s-state as shown in Fig. 2.8. Here the magnetization is assumed to be homogeneous in z-direction, meaning, that the z-direction is discretized by one cell. From the initial magnetization an instantaneously applied uniform and constant Zeeman field is applied on the sample causing a switching of the magnetization direction from positiv to negativ x-direction.[162,163]

The spatially averaged magnetization $\langle \vec{M} \rangle$ and the magnetization at the time, when the x-component of $\langle \vec{M} \rangle$ first crosses zero, are derived on the basis of simulations. For this

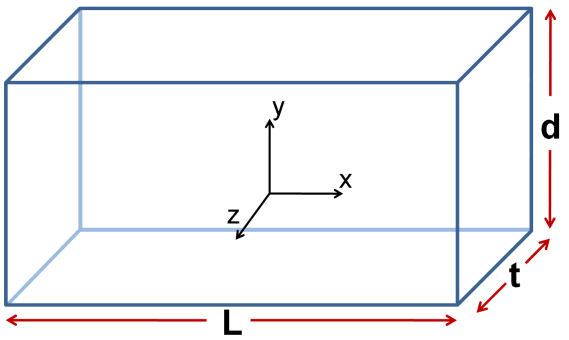


Figure 2.7: Ferromagnetic thin film investigated in standard problem No. 4.

simulation the exchange, the demagnetization, and the Zeeman field are included in the effective field. Additional simulation parameters are an exchange constant of $A = 1.3 \times 10^{-11}$ J/m, a saturation magnetization of $M_s = 8.0 \times 10^5$ A/m, a damping constant of $\alpha = 0.01$, and the gyromagnetic ratio of $\gamma' = 221$ km/As. This initial s-state for the system test can be

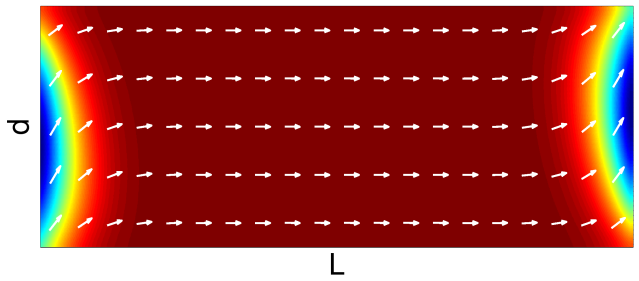


Figure 2.8: Initial s-state of the thin film of standard problem No. 4.

obtained by performing a separate simulation, where the sample is magnetized uniformly in $(1,1,1)$ direction and relaxes in absence of any Zeeman field to equilibrium resulting in the desired s-state.

The problem exists in two versions where the Zeeman field varies. In version 1 the Zeeman field is given by $\vec{H}_{Zeeman} = (\mu_0 H_x = -24.6$ mT, $\mu_0 H_y = 4.3$ mT, $\mu_0 H_z = 0.0$ mT) and in version 2 by $\vec{H}_{Zeeman} = (\mu_0 H_x = -35.5$ mT, $\mu_0 H_y = -6.3$ mT, $\mu_0 H_z = 0.0$ mT).

A review of standard problem No. 4 reveals, that the simulation problem describes a dynamic behaviour that is sensitive to wrong simulation parameters and thus allows the verification of a simulator. A closer look also reveals following weak points:

- The published results for the problem on μMag[40] substantiate, that the problem is highly sensitive to the chosen spatial discretization.
- It is not clear, if the chosen properties for the comparison are sensitive and unambiguous measures for the magnetization dynamics.
- In the case of a wrong simulation of the problem, the erroranous result is too complex to for tracing back the error cause.

Chapter 3

Micromagnetic simulator prototypes for (M^3S)

The use of a computational-science IDE (CSIDE) to develop scientific software is a promising new approach in the scientific-computing community. Whether if this approach really holds its promises or not depends on the concrete problem, which for this work is the simulation of micromagnetic problems.

This chapter deals with the conceptual approach of the micromagnetic modeling and simulation kit (M^3S). It begins with a detailed overview of the first prototype called *M^3S-MATLAB* developed using the CSIDE *MATLAB*. It explains important functional and technical requirements identified for a micromagnetic simulator based on *M^3S-MATLAB*. The chapter further introduces possibilities to comply the software quality criteria portability, maintainability, usability, and runtime performance of such a simulator.

In order to study the runtime performance, in a first step, an algorithmic-complexity analysis is performed for a simulation run. Possible performance optimizations are identified on the basis of this analysis. In a second step these optimizations are evaluated with respect to their performance gain and are compared to their impact on the architecture of*M^3S-MATLAB*.

Although the development of *M^3S-MATLAB* resulted in a promising micromagnetic simulator that is easier to extend as *OOMMF*, the use of *MATLAB* revealed several conceptual limitations. Consequently, the alternative prototypes *M^3S-Java* written in *Java* using the *Java Scripting API (JSA)*[31] and *Nmag-FD* written in *Python* using the SciTools package collection have been developed. The aim of the development of these two prototypes was to evaluate if the identified limitations are only *MATLAB* specific or hold in general forCSIDEs.

3.1 Conceptual considerations for M^{3}S

In this section important functional and technical requirements are identified for a micromagnetic simulator and possibilities are introduced to meet these requirements using *MATLAB*. A schematic overview of the architecture of the resulting tool *M^{3}S-MATLAB* is shown in Fig. 3.1 depicting the relationship between the main components `solver`, `configuration objects`, and `analysis kit`. The following article entitled "Simulating Magnetic Storage Elements: Implementation of the Micromagnetic Model into *MATLAB* - Case Study for Standardizing Simulation Environments" was presented at the 2007 Summer Computer Simulation Conference SCSC'07 (that took place between 15 and 18 July 2007 in San Diego, USA) and is reprinted in Sec. 3.1.1. This article identifies the requirements for the solver and describes its implementation using *MATLAB*. Therefore, first a desired architecture for the solver is introduced and in the second step its realization using *MATLAB-Script* and *MATLAB-Simulink* are compared. To forestall naming confusions, it is pointed out that the terms *SimState* and *calculateModel* in the article correspond to the terms *configuration* and *calculatedMdt* in this work.

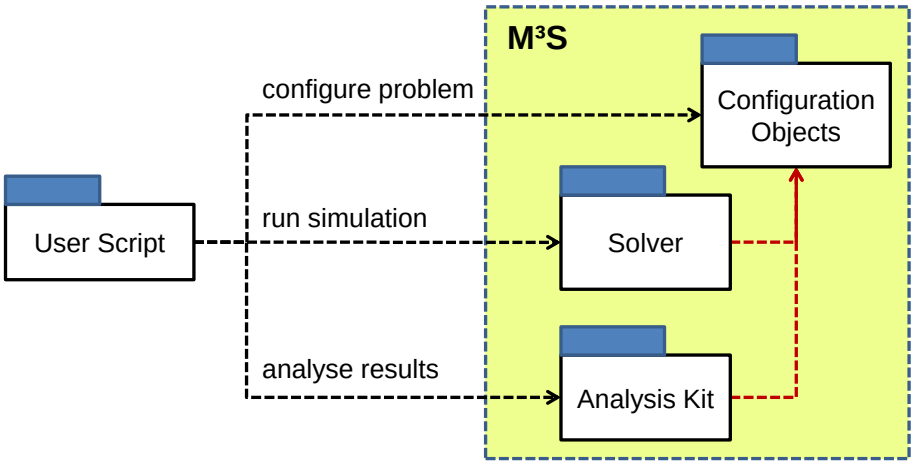


Figure 3.1: The schematic overview of the architecture of *M^{3}S-MATLAB*.

3.1.1 Publication SCSC'07

Simulating Magnetic Storage Elements: Implementation of the Micromagnetic Model into MATLAB - Case Study for Standardizing Simulation Environments

M. - A. B. W. Bolte, M. Najafi, G. Meier, and D. P. F. Möller

Proceedings of the Summer Computer Simulation Conference (SCSC'07), G. A. Wainer, Ed. San Diego, CA, USA: The Society for Modeling and Simulation, 2007, pp. 525-532

Simulating Magnetic Storage Elements: Implementation of the Micromagnetic Model into MATLAB - Case Study for Standardizing Simulation Environments

Markus-A. B. W. Bolte
Institut für Angewandte Physik
und Zentrum für Mikrostrukturforschung,
Jungiusstr. 11,
20355 Hamburg (Germany)
mbolte@physik.uni-hamburg.de

Massoud Najafi
Arbeitsbereich Technische Informatiksysteme,
Department of Informatics,
Vogt-Kölln-Str. 30,
22527 Hamburg (Germany)

Guido Meier
Institut für Angewandte Physik
und Zentrum für Mikrostrukturforschung,
Jungiusstr. 11,
20355 Hamburg (Germany)

Dietmar P. F. Möller
Arbeitsbereich Technische Informatiksysteme,
Department of Informatics,
Vogt-Kölln-Str. 30,
22527 Hamburg (Germany)

Keywords: Simulation of physical phenomena, micromagnetic model, multiphysics, multiscale simulations, MATLAB

Abstract:
Mass data storage devices are the backbone of today's world-wide connected society, and the development of magnetic storage devices has spurred technological advances in a number of fields. Therefore, continuing research on magnetic storage devices and the development of new non-volatile storage concepts with ever higher storage capacity are in great demand. Here we present an implementation of a micromagnetic model that describes the dynamics of magnetic structures on the nano- and micrometer scale into MATLAB for a future inclusion into a multiphysics framework. We explain the fundamentals of the micromagnetic model and the architecture of the implemented code along with its performance. We found that our code is two- to three-times faster in a correct computation of one of the standard problems of micromagnetism than other software.

1. INTRODUCTION

Moore's law describing the exponential increase of computing power over time has held true for the last decades. Both semiconductor technology and magnetic storage media, the two fundamental pillars of today's hardware architecture technology, are approaching fundamental limits, though for different reasons. Due to the downscaling of semiconductor devices the typical length scales are only a few atoms and quantum mechanical phenomena can no longer be neglected[1], while the miniaturization of the magnetic bits is impeded by the so-called superparamagnetic limit, in which the bits become thermally instable and lose their storing capability.

Several novel concepts for magnetic storage devices have been proposed that could potentially also allow for fundamentally different hardware architectures, among them the magnetic random access memory (MRAM), the racetrack memory[2], or domain wall logic devices[3]. With the discovery of the spin-torque transfer effect[4,5] a new field of research has erupted as the spin-transfer torque would allow for a local manipulation of the magnetization through electric currents. Storage devices using the spin-transfer torque effect can also be conveniently included in existing electronic circuits. This further promotes its application. Also it has been shown that magnetic storage devices can be included into standard CMOS fabrication processes to conform to the standard fabrication technique of semiconductor devices. All this makes spin-torque-driven magnetization dynamics a fascinating field of research with very promising applications in sight.

As for any new device, the physical processes must be understood on many levels. Starting from the atomic level to calculate the effects of different materials over the mesoscopic model of micromagnetism[6-8] to Maxwell's macroscopic equations, heat and electrical conductance as well as the magnetodynamics need to be fitted into one model. Some macroscopic equations have already been implemented into powerful simulation frameworks such as COMSOL[9], but a description of the magnetic behavior of ferromagnetic material is still missing.

We here show an implementation of a micromagnetic simulation tool in MATLAB[10] that would allow for an inclusion into the already existing multiphysics simulation environment of COMSOL. The simple mathematical notation of MATLAB makes the implementation much more straightforward than C++, Fortran, or Java. The core of the code with the implementation of the model is surrounded by a simulation framework with extensive test functions. Yet the total code is but a small fraction of a comparable C-code, because MATLAB allows to make use of many built-in functions. Existing codes such as OOMMF[11], LLG®[12],

 ISBN # 1-56555-316-0

MicroMagus®[13], and others are stand-alone packages that efficiently solve the LLG, but so far, the simulation model has been limited to micromagnetic interactions, i.e., the interaction of the magnetization with itself via exchange or demagnetization fields. The inclusion of electric currents or temperature so far has only been done according to simple approximations. For a realistic simulation of all physical properties, i.e., magnetization, current, and temperature, a multiphysics approach is needed.

In this work we describe the development of a micromagnetic simulation toolbox in MATLAB that could potentially be included into a multiphysics environment. First we review the micromagnetic model with its elementary equations and relations, followed by a description of the discretization of the model in space and time to allow for numerical computation. In Section three we explain the concrete implementation of the model into MATLAB code, validate it and evaluate its performance in comparison to an existing micromagnetic simulation tool. Finally, we give a summary and outlook of what we feel possible in the near future.

1. THE MICROMAGNETIC MODEL

The micromagnetic model describes the magnetic behavior of ferromagnetic systems on the nano- and micrometer scale. It can correctly model the static structure of ferromagnets, the formation of magnetic domains and their interfaces, called domain walls, but also the dynamics up to the THz-regime, the magnetic hysteresis, the switching of small magnetic grains, etc. In 1932, Landau and Lifshitz[6] laid the foundation to this theory, with major contributions coming later from Gilbert, Néel, Bloch, Brown, and many others[7,8,14,15]. Several excellent reviews and books describe this theory in great detail[16-18].

1.1. Equation of motion

The fundamental equation in the micromagnetic model is the equation of motion of the magnetization, Landau-Lifshitz-Gilbert-equation (LLG). The magnetization itself does not move, but its constituents, the spins of the localized electrons, can point in any direction in space. The magnetization $\mathbf{M}$ precesses around the local magnetic field $\mathbf{H}_{\mathbf{eff}}$ and is damped towards its equilibrium direction which is parallel to the effective field as described by the two terms on the right-hand side of Eqn. (1).

$$\frac{d\mathbf{M}(\mathbf{r},t)}{dt} = -\left|\bar{\gamma}\right|\mathbf{M}(\mathbf{r},t)\times\mathbf{H}_{\mathbf{eff}}(\mathbf{r},t) - \frac{\left|\bar{\gamma}\right|\alpha}{M_S}\mathbf{M}(\mathbf{r},t)\times(\mathbf{M}(\mathbf{r},t)\times\mathbf{H}_{\mathbf{eff}}(\mathbf{r},t)) \quad (1)$$

M_S is the saturation magnetization, the maximum magnetization a volume of a certain material can attain, i.e., when all micromagnetic moments are aligned parallel.

In turn the magnetization determines the effective field by a superposition of mainly two types of magnetic fields. These field types are caused by different interaction mechanisms that shall be explained in the following paragraphs: the exchange interaction and the demagnetization interaction, also called self-magnetization. Additionally, an external magnetic field can be applied which would then be added to the effective field.

1.2. Magnetic fields and energies

All magnetic interactions, including the exchange interaction which is of quantum-mechanical nature, can be written as a magnetic field interacting with local magnetic moments, even though the origins of the fields differ. The predominant interactions are revisited in the following section.

The relation between a magnetic field due to an interaction and its energy is generally given by

$$E = -\int \mu_0 \vec{\mathbf{H}}\cdot\vec{\mathbf{M}}\,dV\,. \quad (2)$$

This relation applies to all magnetic interactions, but in the simplest way the field can be seen as caused by an external magnetic field, also called Zeeman field, e.g., from a magnetic coil or from the write head of a magnetic hard drive. The field is potentially spatially inhomogeneous and can alter rapidly over time.

Sometimes it is easier to calculate the interaction energy. The corresponding field is then the total differential of the interaction energy density by the local magnetization

$$\mathbf{H} = -\frac{\delta E}{\mu_0 \delta \mathbf{M}}\,. \quad (3)$$

1.2.1. Exchange energy

The exchange interaction is of quantum-mechanical nature. Electrons have mass, energy, and angular momentum, like macroscopic objects, but they also hold a fourth quantity, the spin. The spins of the electrons of neighboring atoms interact in such a way that for ferromagnets the spins want to align parallel, thus increasing the overall magnetic moments of a material. In this way ferromagnets hold a magnetization even without an external field. The increase in energy due to the electron spin in a ferromagnet can be described by the equation

$$E_{exch} = -J\vec{S}_1\cdot\vec{S}_2\,, \quad (4)$$

where $\vec{S}_1$ and $\vec{S}_2$ are the spins and J is a material-dependent parameter, the exchange integral. Approximating the cosine of the scalar product up to second order and using $\mathbf{M}$ instead of the spin, one arrives at

$$E_{exch} = \int dV \frac{A}{M_S^2}\left(\left(\frac{\partial\vec{M}}{\partial x}\right)^2 + \left(\frac{\partial\vec{M}}{\partial y}\right)^2 + \left(\frac{\partial\vec{M}}{\partial z}\right)^2\right), \quad (5)$$

where A is the exchange constant which is proportional to J. Using Eqn. (3) and some vector analysis, we find that the interaction between the electron spins of neighboring atoms can be mimicked by a magnetic field of the strength

$$H_{exch} = \frac{2A}{\mu_0 M_S^2} \nabla^2 \vec{M} \ . \qquad (1)$$

Since the exchange interaction is the origin of ferromagnetism, this field is extremely strong, a thousand times stronger than the strongest external fields applicable in a lab, but it is also very short-ranged.

1.1.1. Demagnetization energy

Another magnetic interaction competes with the exchange interaction: Every electron can be seen as a little magnetic dipole. Each dipole "feels" the magnetic field originating from its neighbors

$$E_{demag} = -\frac{1}{2}\int_V \vec{M} \cdot \vec{H}_S dV. \qquad (2)$$

1.2. Finite-difference discretization of the model

These analytical expressions fully describe the micromagnetic model. The model must be spatially and temporally discretized to be numerically solvable. Care must be taken that the implementation of the discretization has the same behaviour as the analytical model and doesn't introduce artefacts into the simulation. We here describe how the model was spatially discretized via the finite difference method and the introduction of a topology. We limit our description to the LLG-equation as well the two dominant energy terms, the exchange and the demagnetization energy. We will then briefly touch upon disjoint simulations states and numerical integration methods to achieve the temporal discretization.

The magnetization is a function of both space and time. In the finite-difference method (FDM), the simulated volume $V = X \cdot Y \cdot Z$ is divided into n_X, n_Y, n_Z blocks called simulation cells of equal size $\Delta x \cdot \Delta y \cdot \Delta z$ (in Cartesian coordinates). As each dimension of the simulated volume is to be divided into an integer number of cells, only rectangular structures can be effectively and correctly simulated. For curved geometries, errors on the edges occur as the approximated solution can greatly differ from the correct one. For an alternative approach, cells of different shape and size, e.g., tetrahedrons, could be used to approximate the surface more correctly, as is done in the finite-element-method (FEM)[19]. However, for the micromagnetic model, the FDM is computationally less expensive as its regular grids allow the use of fast convolution algorithms for the computation of the demagnetization field, as will be shown below. We shall also limit ourselves to the Cartesian coordinate system.

After discretizing the simulation volume into cells, the material-dependent micromagnetic properties, i.e., saturation magnetization, exchange coefficient, and anisotropy constant, must be defined for every cell. The magnetization, the solution of the LLG-equation, is then determined as the product of a space and time dependent vector of norm 1 with a (spatially dependent) material parameter M_S. The LLG-equation can be solved for every cell individually so that solving the micromagnetic problem becomes finding the solution for $n = n_X \cdot n_Y \cdot n_Z$ ordinary differential equations written as

$$\begin{aligned} \frac{d\mathbf{M}(\mathbf{r}_j,t)}{dt} &= -\left|\bar{\gamma}\right|\mathbf{M}(\mathbf{r}_j,t)\times\mathbf{H}_{\mathbf{eff}}(\mathbf{r}_j,t) \\ &-\frac{\left|\bar{\gamma}\right|\alpha}{M_S(\mathbf{r}_j)}\mathbf{M}(\mathbf{r}_j,t)\times\left(\mathbf{M}(\mathbf{r}_j,t)\times\mathbf{H}_{\mathbf{eff}}(\mathbf{r}_j,t)\right) \end{aligned} \qquad (3)$$

The coupling between the individual cells only comes into play by the magnetic interactions that are summarized by the effective field. The discretization of the effective field is the great challenge when implementing the micromagnetic model.

$$\begin{aligned} &\mathbf{H}_{\mathbf{eff}}(\mathbf{r_j},t_i,\mathbf{M}(\mathbf{r_{all}},t_i),\mathbf{M}(\mathbf{r},t_i)) \\ &= \mathbf{H}_{\mathbf{Z}}(\mathbf{r_j},t_i) + \mathbf{H}_{\mathbf{E}}(\mathbf{r_j},t_i,\mathbf{M}(\mathbf{r_j},t_i),\mathbf{M}(\mathbf{r_{nn}},t_i)) \\ &+ \mathbf{H}_{\mathbf{D}}(\mathbf{r_j},t_i,\mathbf{M}(\mathbf{r_{all}},t_i)) + \mathbf{H}_{\mathbf{A}}(\mathbf{r_j},t_i,\mathbf{M}(\mathbf{r_j},t_i)) \end{aligned} \qquad (4)$$

where *all* are the indices of all cells, and *nn* the indices of the nearest neighbors to the j-th cell. To compute the exchange field one must find an efficient discretization of computing the partial differential in Eqn. (3). One uses a Taylor-polynomial to determine the change in magnetization between two cells. As the exchange interaction is of a very short-ranged nature it is generally sufficient to limit the computation to nearest neighbors. For the j-th cell ones gets

$$\mathbf{H}_{\mathbf{E}}(\mathbf{r_j},t_i) = \frac{A \cdot M_s}{\mu_0} \cdot \sum_{k=nn} \frac{\mathbf{M}(\mathbf{r_j},t_i)\cdot\mathbf{M}(\mathbf{r_k},t_i)}{(\mathbf{r_j}-\mathbf{r_k})^2}, \qquad (5)$$

where A is the exchange constant. There are several ways of selecting the nearest neighbors. For three-dimensional problems, one could either select either the six cells with common surfaces or the 26 cells with shared corners. In either way, the computational complexity is O(N)[20].

To discretize the demagnetization field one must first solve the demagnetization tensor **N** for every cell. The double sum, which would lead to an $O(N^2)$ complexity, can then be converted into an convolution integral. Newell and Dunlop [21] first delivered a solution for the demagnetization tensor of rectangular bodies, where before it had been determined for ellipsoidal bodies only.

By applying Gaussian's integral law, Newell reduced the demagnetization tensor elements to the surface integrals of a simulation cell.

$$\mathbf{H_D}(\mathbf{r_i}, t, \hat{\mathbf{N}}) = \sum_j \hat{\mathbf{N}}(\mathbf{r_i} - \mathbf{r_j}) \cdot \mathbf{M}(\mathbf{r_j}),$$

$$\hat{\mathbf{N}}(\mathbf{r_i} - \mathbf{r_j}) = \frac{1}{4\pi\tau_i} \int_{S_i} \int_{S_j} \left(\frac{1}{|\mathbf{r_i} - \mathbf{r_j}|}\right) d\mathbf{S_j} d\mathbf{S_i} \quad . \tag{1}$$

The diagonal elements, N_{ii} , describe the interaction of the magnetization at opposite surfaces of a cell, and the non-diagonal elements N_{ij} describe the interaction between the corresponding other surfaces. Applying some mathematics, Newell et al. derived a closed formula for the tensor elements (see Ref. 21 for details). For an implementation of the algorithm, one must also treat the cells on the surface of the simulated volume, i.e., where $n_X, n_Y, n_Z = 0$ or $n_{i,\max}$.

The convolution integral can then be substituted by a discrete convolution sum

$$\mathbf{H_D}(\mathbf{r_j}, t_i, \hat{\mathbf{N}}) = \sum_k \hat{\mathbf{N}}(\mathbf{r_j} - \mathbf{r_k}) \cdot \mathbf{M}(\mathbf{r_k}, t_i) \,. \tag{2}$$

The demagnetization field H_D needs to be computed for all j cells at every time step, making it by far the costliest part of the computation. To reduce the computing time, one can make use of the regular grid of the FDM and Fourier transform the tensor $\hat{\mathbf{N}}$ and the magnetization vectors. The computation then consists of nine multiplications (for each tensor element) and two Fourier transformations. Since the geometry of the simulated volume does not change in the course of the simulation, the demagnetization tensor elements need only be computed once, at the initialization phase and is stored as its Fourier transform. At every simulation step, **M** is Fourier transformed and the field is computed and then inversely transformed back into real space. By using the symmetry of the problem in Fourier space, one can even reduce the number of necessary operations to one-eighths[11].

1.1. Time discretization

To simulate the dynamics of the magnetic system from time t_0 to t_{end}, the time has to be discretized to a series of points $[t_i]$. The model is thereby changed as follows:

$$\frac{d\mathbf{M}(\mathbf{r}_j, t_i)}{dt_i} = -|\bar{\gamma}|\mathbf{M}(\mathbf{r}_j, t_i) \times \mathbf{H_{eff}}(\mathbf{r}_j, t_i) - \frac{|\bar{\gamma}|\alpha}{M_S(\mathbf{r}_j)} \mathbf{M}(\mathbf{r}_j, t_i) \times \left(\mathbf{M}(\mathbf{r}_j, t_i) \times \mathbf{H_{eff}}(\mathbf{r}_j, t_i)\right) \tag{3}$$

For the stepwise solution for the magnetization Eqn. (13) must be integrated numerically. The next values for the magnetization in each cell $\mathbf{M}(\mathbf{r}_j, t_i)$ are computed by multiplying the time derivative $d\mathbf{M}/dt$ with a discrete time step h_i and adding the result to the function values $\mathbf{M}(\mathbf{r}_j, t_{i-1})$, as depicted in Fig. (1). From the new values $\mathbf{M}(\mathbf{r}_j, t_i)$ the effective field components are then calculated and the integration is repeated until t_{end} is reached.

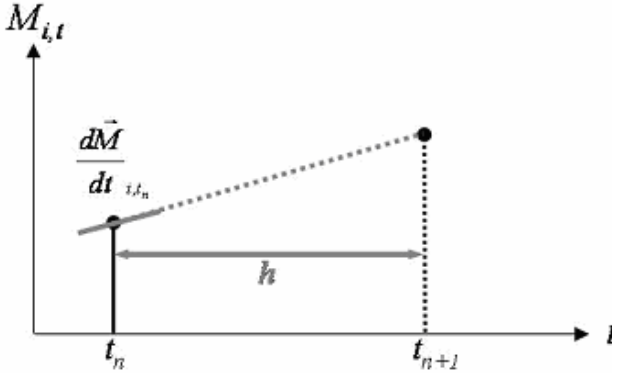


Figure 1: Numerical computation of the integral of the Landau-Lifshitz-equation.

For an efficient numerical integration it is feasible to use multi-step or Runge-Kutta methods. The multi-step method evaluates the function, in this case $d\mathbf{M}/dt$, at several time steps in the past (Adams-Bashforth methods), sometimes implicitly including the next time step for a predictor-corrector algorithm (Adams-Moulton methods) to increase the accuracy of the integration[22, 23]. For a given accuracy the time step can then be substantially enlarged so that the computation of the integration becomes much faster. Alternatively a time interval $[t_i, t_{i+1}]$ can be further divided by additional midpoints at which the function is evaluated (Runge-Kutta method).

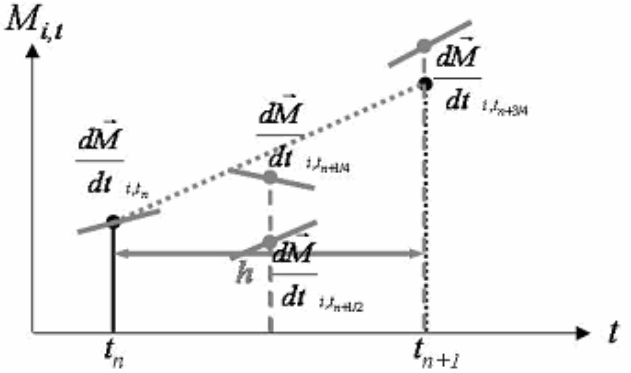


Figure 2: Numerical integration using midpoints following the Runge-Kutta method.

This leads to much larger time steps and faster simulation. Even though in both methods the integration requires multiple function evaluations, of which, as already mentioned, the computation of the demagnetization field is the most time consuming, the multi-step or Runge-Kutta integration schemes are often several orders of magnitude faster than

single step methods[24]. Today, efficient integration methods with adaptive time steps are available in standard literature[23].

1. IMPLEMENTATION INTO MATLAB

To implement the micromagnetic model with MATLAB, an abstract architecture for the tool was created, which serves for comparison of individual implementation variants with the aim to make the software easy to test and to expand. For this the architecture should be modular. Attention needs to be given to the dependencies between the modules: They should be tree-like and contain no cycles. The abstract architecture consists of two components, the "solver" representing the execution of a simulation run and the "driver" representing the automated execution of simulation runs. The driver is an optional component and serves only for the comfortable handling.

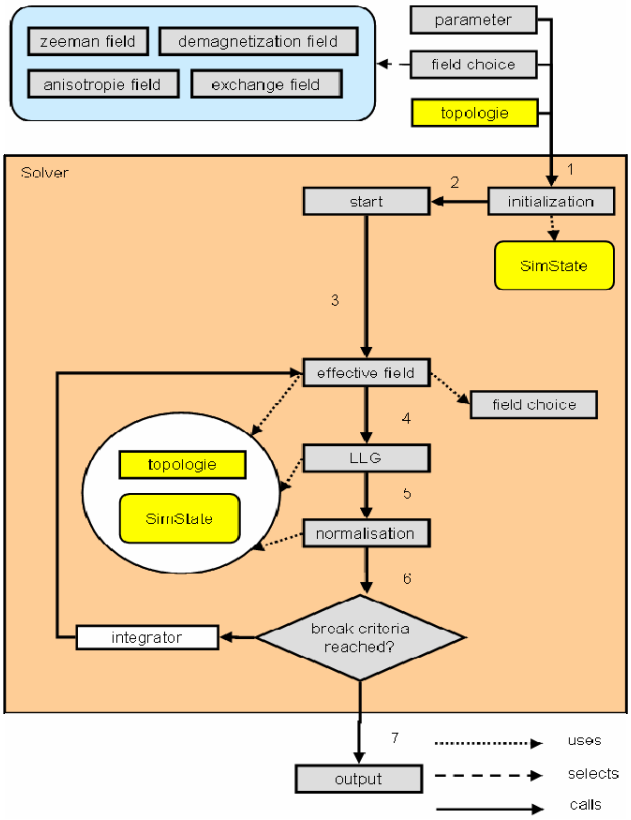


Figure 3: Abstract architecture of the simulation core (solver).

1.1. Solver

A simulation run from t_0 to t_{end} is executed by the solver. The solver is used to hide the implementation from the outer interfaces. It controls the timing and the initialization, and tests the input parameters for correctness. It can also make corrections for incorrect input parameters and thus simplifies the definition of a simulation problem. The solver uses the physical components, i.e., the topology, the different magnetic fields, the effective field, and the LLG, to accomplish the simulation. The topology represents the area, in which the problem is defined. Each field implementation depends on the topology; therefore, the selected field implementation must fit to the selected topology. The LLG and the effective field represent the micromagnetic model. A normalization is needed to adjust $|\mathbf{M}|$ due to integration errors. As the user can adapt the effective field and its components according to the desired problem definition without having to change the solver, the interface remains general enough for future extensions.

In a simulation run the topology must be initialized first. Then the fields are initialized based on the topology. The start of the numeric integration to t_{end} follows. In every step, the computation of the effective field, of the LLG, and of the normalization uses the topology and the field selection and **M** to compute d**M**/dt. The current state of the simulation can be stored at each time interval. Thus there is the possibility to continue the simulation run in the case of an abort. The "SimState" structure is used as that central state container.

1.2. Solver with MATLAB

MATLAB offers functions for the computation of numeric integrations[10]. The use of these functions simplifies the implementation of the solver substantially. In this implementation the solver has the task to initialize the simulation environment and to configure the output based on the user inputs and to start the simulation. The integrator of MATLAB then takes over the numeric integration from t_0 to t_{end}. For this the integrator needs a function, which computes d**M**/dt. In the following it is called "calculateModel". The integrator stops when t_{end} is reached. For alternative stopping criteria a corresponding function can be integrated. This is also the task of the solvers. Figure 4 shows the architecture adapted to the possibilities of MATLAB.

1.3. Solver with Simulink

Simulink has become a very extensive product of Mathworks[10]. Therefore we tried Simulink as another implementation of our micromagnetic simulation tool using the specific advantages it offers. The model in Simulink is arranged on a graphical user interface. It theoretically needs no line source code to realize a model. In this application, however, the functionality of Simulink is not sufficient to make it possible to realize the model, because, e.g., Simulink cannot compute a cross product between two matrices or deal with objects. Therefore the needed functionality was in part written as embedded MATLAB functions in Simulink blocks. The sequence of the simulation and the time integration was done by the integrator of Simulink. Overall, the advantages of Simulink do not outweigh its dis-

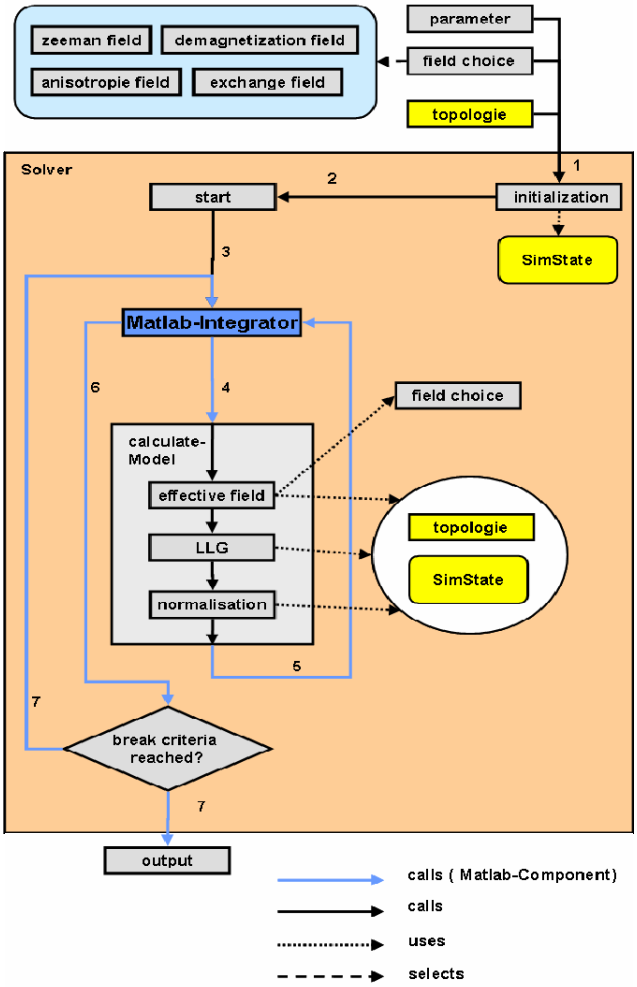


Figure 4: Architecture of the Solver for the implementation with MATLAB.

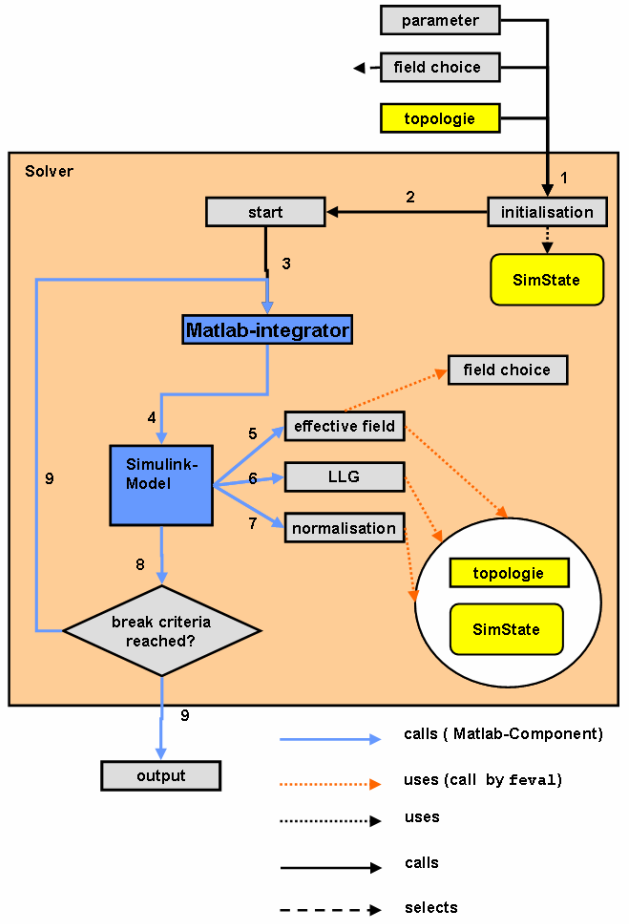


Figure 5: Architecture of the Solver for the implementation with Simulink. As shown by the arrows 5-7, Simulink calls embedded MATLAB functions to simulate the model.

advantages, i.e., worse performance and the loss of object-oriented programming, so that we decided against this implementation. Figure 5 shows the architecture adapted to the possibilities of Simulink.

1.4. Driver

The Solver alone is not sufficient for the needs of the user. Often a user would like to be able to simulate sequences of runs with varying parameters, e.g., hysteresis curves. A hysteresis curve describes the change of the magnetization as a function of a sequence of magnetic fields, which run from $-H_{max}$ to H_{max} and back. A hysteresis plot, i.e., M(H), yields the specific magnetic properties can be deduced, such as the coercive field or the saturation magnetization. The final state of the predecessor simulation run is used as starting condition for the next simulation run. Hysteresis loops are only one example. Generally the user needs a mechanism with which he can run a sequence of simulations with defined change (e.g. variation of a parameter) or assumption of previous simulation results (e.g. investigation of repeated shifting processes after the other). This function range is the task of the drivers. Figure 6 shows the basic architecture of such a driver.

1.5. Correctness

Best architecture is of no use if the simulation supplies wrong results. To test the correctness of individual parts of the simulation, i.e., the field computation or solving the LLG, a test framework was written that asserted the correctness of these parts by comparing the results for test values to analytical solutions[11] and checking the data interfaces. For the individual parts the accuracy we achieved with respect to analytical values reached the numerical accuracy. To validate the complete simulation code, correct reference values were needed, i.e., micromagnetic reference problems for which the solutions are known. The µMag group[25] has collected such standard problems. In the following the stan-

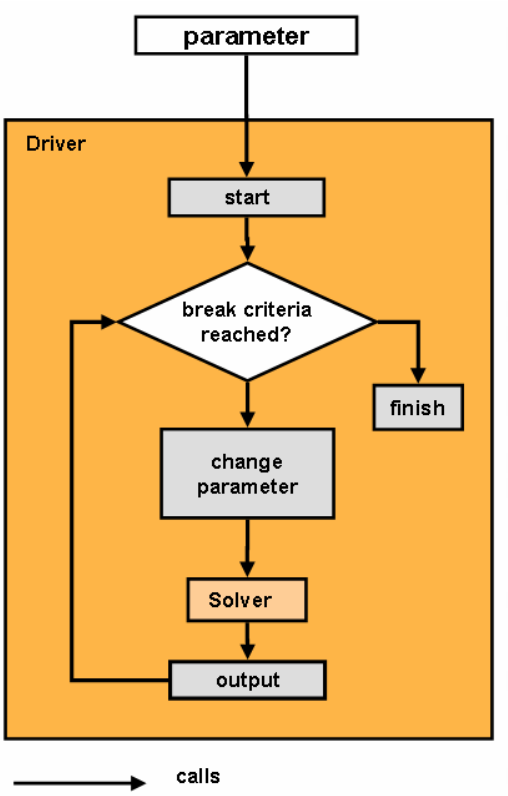


Figure 6: Architecture of the Driver

dard problem 4 and corresponding results were used to validate the simulation.

Standard problem 4 deals with the magnetization dynamics of a small ferromagnetic platelet due to an external field. It is an ideal test for the current version of this simulation tool. The test consists of two simulations with different external fields in opposite direction of the magnetization causing it to switch. Figure 7 show the comparison between the reference values from µMag contributors and the results of simulations with the present code. As can be seen the maximum deviation between the reference value and our simulation code is less than 6%. The results of the µMag group differ amongst each other by the same values so that one can safely say that within the accuracy of the model our results are correct.

1.6. Performance

The performance of our code was compared to one of the most popular open-source micromagnetic simulation codes, OOMMF[11]. The performance test consists of simulating standard problem 4 on the same computer and comparing computing times. It shows that our micromagnetic code in MATLAB is almost twice as fast as OOMMF (see Table 1), which uses the Euler integration procedure. The Simulink variant is around a factor 2 slower than the pure Matlab implementation, because of indirect calls. When comparing the number of lines of code it results that our program has only 4065 lines source code including tests, without tests only 2570 lines source code. At the same time this tool is more easily understood because of the mathematically motivated script language of MATLAB than pure C++ code. OOMMF for example doesn't contain tests and has more than 30000 lines of code. Thus the range of the software was reduced by the Factor 10 and the quality of the code was improved at the same time.

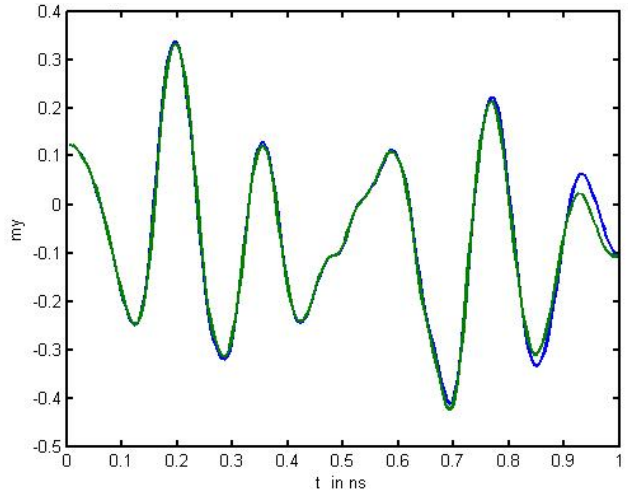


Figure 7: Comparative presentation of the y-component of the magnetization for standard problem 4 (field 1) as simulated with the present code (blue) and reference values (green). See Ref. 25.

	Cells	Simulated time	Needed time
OOMMF	10000	0-1ns	13873s
MATLAB	10000	0-1ns	6176 s

Table 1: Performance test results for standard problem 4.

2. SUMMARY AND OUTLOOK

We have shown the successful implementation of a micromagnetic simulation tool in MATLAB that is working correctly and efficiently. Its architecture and the MATLAB scripting language allows for a convenient expansion of the code to include other effects of physics and a possible connection to multi-physics platforms. A further optimization of the code through more efficient algorithms and parallelization of the code as well as an adaptation of the spin-torque effect are planned for the future.

3. REFERENCES

[1] Brennan, K. and Brown, A. 2002. *Theory of Modern Electronic Semiconductor Devices*, John Wiley & Sons, Inc., New York, NY.

[2] Parkin, S. S. P. 2004. US Patent 309, 6,834,005.

[3] Cowburn, R. P. 2004. Patent Application 309, WO00 2004077451A1.

[4] Berger, L. 1996. "Emission of Spin Waves by a Magnetic Multilayer Traversed by a Current", Phys. Rev. B **54**, 9353 (1996); J. C. Slonczewski. 1996. "Current-driven excitation of magnetic multilayers", J. Magn. Magn. Mater. 159, **L1**.

[5] Zhang, S. and Li, Z. 2004. "Roles of Nonequilibrium Conduction Electrons on the Magnetization Dynamics of Ferromagnets", Phys. Rev. Lett. **93**, 127204.

[6] Landau, L. and Lifshitz, E. 1935. "On the Theory of the Dispersion of Magnetic Permeability in Ferromagnetic Bodies", Physik. Z. Sowjetunion **8**, 153.

[7] Gilbert, T. L. 1955. "A Lagrangian Formulation of the Gyromagnetic Equation of Magnetization Field", Phys. Rev. **100**, 1243.

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

[9] http://www.femlab.com/

[10] http://www.mathworks.com/

[11] Donahue, M.J. and Porter, D.G. 1999. "Object oriented micromagnetic framework, OOMMF, User's Guide, Version 1.0", Interagency Report NISTIR 6376, NIST, Gaithersburg, MD.

[12] http://llgmicro.home.mindspring.com/

[13] http:// www.micromagus.de/

[14] Bloch, F. 1932. "Zur Theorie des Austauschproblems und der Remanenzerscheinung der Ferromagnetika", Z. Phys. 74, 295.

[15] Néel, L. 1955. C. R. Acad. Sci. 241, 533.

[16] Aharoni, A. 1996. *Introduction to the Theory of Ferromagnetism*. Clarendon, Oxford.

[17] Hubert, A. and Schäfer, R. 1998. *Magnetic Domains: The Analysis of Magnetic Microstructures*. Springer, Berlin, Germany.

[18] Kronmüller, H. and Fähnle, M. 2003. *Micromagnetism and the Microstructure of Ferromagnetic Solids*, Cambridge University Press, Oxford, UK.

[19] Fidler, J. and Schrefl, T. 2000. "Micromagnetic modelling - the current state of the art", Journal of Physics D: Applied Physics, 33 R135-R156.

[20] Donahue, M. J. and McMichael, R. D. 1997. "Exchange Energy Representations in Computational Micromagnetics", Physica B, 233, 272-278.

[21] Newell, A.J.; Williams, W.; Dunlop, D.J. 1993. "A Generalization of the Demagnetization Tensor for Nonuniform Magnetization". J. Geophys. Res. 98, No. B6. 9551-9555.

[22] Press, W.; Teukolsky, S.A.; Vetterling, W.T.; and Flannery, B.P. 2002. *Numerical Recipes in C, The Art of Scientific Computing*, 2nd Ed. Cambridge University Press, New York, NY.

[23] Deuflhard, P. and Bornemann, F. 2006. *Scientific Computing with Ordinary Differential Equations*, 1st Ed., Springer Verlag, Berlin, Germany.

[24] Bolte, M.; D. P. F. Möller; and Meier, G. 2004. "Simulation of Micromagnetic Phenomena". Proceedings of the 18th European Simulation Conference. SCS Publishing House. 407-412.

[25] http://www.ctcms.nist.gov/~rdm/mumag.org.html

Biographies

Markus-A. B. W. Bolte is finishing up his Ph.D. in Physics at the University of Hamburg, Germany, on the topic of micromagnetic simulation and X-ray microscopy of nanomagnets. In 2004, he received his masters in computer science and physics. His email address is mbolte (at)physik.uni-hamburg.de .

Guido Meier is a division leader in the research group of Prof. U. Merkt at the Institute of Applied Physics of the University of Hamburg, Germany. His research activities include ferromagnets, semiconductors, and hybrid devices of both materials on the micro- and nanometer scale. The group's homepage can be found at www.physnet.uni-hamburg.de/ institute/IAP/Group_N/ .

Massoud Najafi has recently developed the finite-difference based micromagnetic simulation tool described in this publication as part of his Masters's thesis in Informatics at the University of Hamburg, Germany. He has now started his PhD thesis. His email address is massoud.najafi@calja.org .

Dietmar P. F. Möller is a professor of computer science and engineering and chair of computer science at the University of Hamburg, Hamburg, Germany. He is currently working on embedded computing systems, virtual and augmented reality, modeling and simulation, mobile autonomous systems, soft computing, medical informatics and nanostructures. His homepage is www.informatik.uni-hamburg.de/TIS/ .

3.1.2 Supplementary

As described in Sec. 2.1.3, one important aspect of the development of M^{3}S was to use automated testing to ensure the correctness of its functionality. Since *MATLAB* does not offer an automated test framework, a simple framework has been developed for *M^{3}S-MATLAB*. This framework will be introduced in the following and possibilities to measure the test coverage in *MATLAB* are presented. In Sec. 3.1.1 the correctness of the solver has been verified by the simulation of standard problem No. 4. In addition this sub-section presents results for the Larmor-precession test.

Test driver

```
%testTempBaseDir - base directory to store intermediate results.
%testType - {``short'', ``long'', ``all''}.
function runAllTests(testTempBaseDir,testType)

% test - a function handle to the testfunction to run.
% testTempDir - the directory to store intermediate results.
function runTest(test, testTempDir)
```

Code listing 3.1: API of the `runTest` and `runAllTests` function.

The test driver as introduced in Sec. 2.1.3 is a component that performs all automated tests and generates the test report. In the test framework for *M^{3}S-MATLAB*, the test driver interface is given by the `runTest` and `runAllTests` function as shown in Code listing 3.1. These functions can be called to run either a single test function or all test functions within the current directory and all its subdirectories. To run all tests using `runAllTests` the parameter `testTempBaseDir` and `testType` need to be specified. `runAllTests` identifies a test function by its function name.

Here a distinction between test functions with a short and a long runtime has been included. By contract *shorttest* functions are tests that run quickly while *longtest* functions are either performance tests or whole simulation runs, which requires a longer runtime. The parameter `testType` allows to specify the desired test type to be run by `runAllTests`. For `testType` the value *short*, *long*, and *all* can be chosen, which correspond to the *shorttest* functionname prefix, *longtest* functionname prefix, and both functionname prefixes. The distinction between tests with a short and a long runtime is necessary. In micromagnetic simulation tests with a long runtime cannot be avoided and hence running all tests exceeds a critical runtime. Without this destinction developers run the tests only over night or over the weekend, as they do not want to wait too long for the completion of the test run. This strategy involves the risk that the longer the span is between two tests runs, the more untested changes might have been made to the code. Then it becomes difficult to attribute a fault to a specific change. The distinction in the test driver offers the developer to run all short-runtime tests with a high frequency and the long-runtime tests over night or weekend. Because the short-runtime tests are the majority of tests, this reduces the risk of

untested code crucially. The second aspect managed by the test driver is to provide destinct temporary directories, where the test function can save intermediate files. This is necessary, as simulation results can exceed the main memory space. The parameter `testTempBaseDir` specifies the overall directory for all test functions, from which the test driver generates destinct temporary directories for each test function. Additional the test report is stored in the log file *tests.log* in the base directory. It allows a later evaluation of the test run.

Finally the `runTest` function allows to run a specific test. This function is important as it can be used to rerun single test when analyzing the *tests.log* file. The results of the single test run are printed to the command window.

Test interface

To implement an automated test, it is necessary to implement a function that performs the test. In the test framework for M^{3}S a test function has to comply the test interface shown in Code listing 3.2 to be runnable by the test driver.

```
function testMYNAME(testTempDir,testReferenceDir,runClearResults)

function simtestMYNAME(testTempDir,testReferenceDir,runClearResults)
```

Code listing 3.2: API a test method has to fulfill.

The parameters `testTempDir` is a destinct directory provided by the test driver and is reserved for the intermediate files that are stored by a test. The parameters `testReferenceDir` is a directory provided by the test driver and is reserved for reference files that store to expected results for the test.

Test coverage

Combining automated unit-testing with test coverage measurements is essential as it allows to select more significant test cases and thus to increase the quality of the automated tests.

Test coverage of main solver functions.

Function	**Coverage (%)**
`calculateModel`	100
`LLG_DGL`	89
`calculateHeff`	100
`demagField`	81
`exchangeField`	100
`zeemanField`	100

Table 3.1: Test coverage of main solver functions in *M^{3}S-MATLAB*.

In *MATLAB* the measurement of the test coverage is limited as only the measurement of the C_0 metric is supported. The C_0 metric can be extracted from the *MATLAB* profiler which is mainly a runtime measurement tool. Table 3.1 shows the C_0-test coverage of the solver component of *M^{3}S-MATLAB* for which a nearly 97 % overall coverage has been reached. Table 3.1 also shows that the `LLG_DGL` and the `demagField` function are not sufficiently tested, as they are only covered by 89 % and 81 %.

Larmor-precession test

The Larmor-precession test is a system test that allows to validate the LLG implementation and the ordinary differential equation (ODE) solver. As explained in Sec. 2.3.2 this test is given by a micromagnetic simulation, where only a constant Zeeman field isincluded in the effective field. Since the damping constant α is set to zero, the magnetization starts to precess undampedly around the constant Zeeman field. The feature of this problem is, that the undamped-precession period, also called Larmor-precession period, can be estimated analytically and hence allows to estimate the error of the simulation run. To validate the LLG implementation and to compare different ODE solvers, the Larmor-precession test has been performed with varying error tolerances. As shown in Tab. 3.2, this evaluation reveals three relations between the choice of the ODE solver and the chosen error tolerances:

1. The error relates nonlinearly to the chosen error tolerances.
2. The necessary error tolerances to achieve one and the same error can vary depending on the chosen ODE solver and its adaptive time step algorithm by two orders of magnitude.
3. The number of necessary evaluations of $d\vec{M}/dt$ to achieve the same error can vary depending on the chosen ODE solver and its adaptive time step algorithm by one order of magnitude.

Solver	**rel. tol. (%)**	**abs. tol. (A/m)**	**precession period (ps)**	**error (%)**	**evaluations of $d\vec{M}/dt$ (#)**
ode45	$1\cdot10^{-2}$	$1\cdot10^{-2}$	28.425687	$9.8\cdot10^{-3}$	540
	$1\cdot10^{-5}$	$1\cdot10^{-5}$	28.428473	$1.4\cdot10^{-5}$	1608
	$1\cdot10^{-6}$	$1\cdot10^{-6}$	28.428477	$< 10^{-6}$	2832
ode23	$1\cdot10^{-2}$	$1\cdot10^{-2}$	28.379947	1.7	1293
ode23	$1\cdot10^{-5}$	$1\cdot10^{-5}$	28.428477	$< 10^{-6}$	12699
Cash/Karp	$1\cdot10^{-2}$	$1\cdot10^{-2}$	28.428477	$< 10^{-6}$	3984

Table 3.2: Results of the Larmor-precession test for three different adaptive-timestep ordinary-differential-equation solvers. For each solver the precession period is estimated for varying error tolerances and compared with the analytically determined precession period of 28.428477 ps. In addition the number of evaluations of $d\vec{M}/dt$ is listed.

The Larmor-precession test does not include the exchange and demagnetization field. That is too simple for a quantitative prediction of adequate error tolerances. Nevertheless the Larmor-precession test is a good means to check the correctness of the LLG implementation in combination with the used ODE solver.

The results shown in Tab. 3.2 further illustratethe difficulty for an user to choose the ODE solver and the error tolerances,such that the fastest simulation run with the accuracy desired by the user is achieved. This decision is even more difficult in general, since for most systems no analyical reference value is available. Assuming that an user investigates only one category of systems, one approach can be to estimate the error tolerances by a similar simulation scheme for an exemplary system. The estimation of the error for this approach differs from the method used in Tab. 3.2 in that way, that instead of the missing analytical reference value an extrapolated value is referred. Using this approach allows to perform the expensive simulation scheme only once per category of systems.

3.1.3 Configuration objects

In *M^{3}S-MATLAB* the simulation run is performed by the `Solver` as schematically shown in Sec. 3.1.1. The `Solver` is started by calling the `runSim` function from a user script. In this user script the simulation problem has to be specified and the simulation run has to be started calling the `runSim` function with the problem specification as argument. The `runSim` function internally creates and initializes the simulation state and starts the main simulation loop. Obviously the `Solver` and the problem specification API are the simulator components the user mainly gets in touch with. Thus the concrete realization of the corresponding API has large effects on the usability and acceptance of the tool and hence necessitates to use the most robust language elements for its implementation.

In the following three realization approaches are discussed:

1. passing all parameters as arguments in key-value pairs,
2. passing a struct containing the values in its named variables,
3. passing a configuration object that contains the problem definition.

The key-value pair approach is shown exemplary in Code listing 3.3. Here the parameters are passed as arguments, where each odd indexed argument is a string giving the key and each even indexed argument is its corresponding value. This approach is the most common pattern to realize parameter passing in *MATLAB*. It is used, when the number of parameters in general is small. But it has the drawback, that the validity of the parameters is checked within the `runSim` function so that the distance between the error message provided by the `runSim` function and its causing error can be hundreds of lines away from each other. This makes it difficult to trace back to the error cause from the error message.

```
runSim( 'key1', value1,...
        'key2', value2,...
        'key3', value3,...
        'key4', value4,...
        'key5', value5,...);
```

Code listing 3.3: Example for parameter passing using the key-value pairs approach.

For large numbers of parameters the parameter-struct approach as shown in Code listing 3.4 is more favorable in *MATLAB*. Here a struct, also called record in other programming languages, is a simple data structure. It is a data container consisting of variables stored under an unique identifier by which it can be accessed. In the struct approach, the parameter passing is realized by combining all parameters to a struct that is passed to the `runSim` function. In contrast to the key-value pairs approach, the struct approach allows debugging through the user script and to check the consistency of each parameter contained in the struct before passing it to the `runSim` function. Such a usage of the API can in principle be supported by defining a `isConsistent` function that can be called during the debugging to check, if the problem definition is in a valid state. Concerning that *MATLAB* is a dynamically typed language[164] and that the problem definition has a large complexity, the implementation of such an `isConsistent` function is a difficult task as a large number of possible sources of errors needs to be checked.

```
parameterStruct.key1 = value1;
parameterStruct.key2 = value2;
parameterStruct.key3 = value3;
parameterStruct.key4 = value4;
parameterStruct.key5 = value5;

runSim(parameterStruct);
```

Code listing 3.4: Example for parameter passing using the parameter struct approach.

In the configuration-object approach an object is used instead of a struct to hold the problem specification. As an object is a language element that encapsulates variables and functions to units, an object-oriented approach allows to control changes of the problem specification and thus to reject invalid changes. Further post-processing steps can be performed like changing dependent properties. The object-oriented approach in contrast to the struct approach allows to split the consistency check in the `set` method and the `isConsistent` method. As the `set` offers to reject type errors like wrong matrix shapes or the specification of strings instead of numbers, it reduces the complexity of the `isConsistent` method drastically compared to the `isConsistent` function used for the struct approach. Thus, the object oriented approach allows to provide clearer error messages and offers a better traceability of errors. Code listing 3.5 shows the parameter passing realized for using a configuration object. At first a configuration object is created calling the constructor. At second the values are set by calling the `set` method of the configuration object. The `set`

method generates error messages when the value is invalid for the simulation problem. For instance setting a magnetization with a wrong shape compared to the specified topology would lead to an error message. In the third step the configuration object is passed to the `runSim` function that internally calls the `isConsistent` function to check the consistency before starting the simulation.

When using configuration objects, one has to take care that *MATLAB* applies call-by-value to object-oriented programming. This means, the internal state of an object, which is passed as an argument to a procedure like the `set` method, is immutable. Changes applied to the object state by a procedure are only accessible, when the procedure returns the changed object.

```
conf = configuration();
conf = set(conf, 'key1', value1);
conf = set(conf, 'key2', value2);
conf = set(conf, 'key3', value3);
conf = set(conf, 'key4', value4);
conf = set(conf, 'key5', value5);

runSim(conf);
```

Code listing 3.5: Example for parameter passing using the configuration object approach.

An additional benefit of the configuration object is that it allows to design a modular program. With adding the `init` function to the configuration-object API, the configuration objects encapsulate the component-specific property specification, validation, initialization, andcalculation during the simulation run in methods. The increased modularitysimplifies the testability of the simulator drastically, as each component canbe tested independently from the other solver components and thus a simple testenvironment can be set up.

In the following an example script illustrates the structure of a simulation script and the basic elements to define a micromagnetic simulation problem in *M^{3}S-MATLAB*.

Simulation example

Code listing 3.6 shows the *M^{3}S-MATLAB* script to simulate and analyze the Larmor-precession test. The main script named `larmorPrecessionTest` configures the simulation problem (lines 2-32). It starts the simulation by calling `runSim` (line 34), and analyzes the results through calling the help function `analyzeResults` (line 35) listed below the main function.

```
function larmorPrecessionTest()
    basedir = 'MY_FOLDER';
    topo = topology(1e-9, 1e-9, 1e-9, 1e-9, 1e-9, 1e-9);

    Ms = 7.9577e5; %in A/m
    M0 = Ms * ones(topo.anzahl,3)/sqrt(3);

    fields = fieldMap();
    fieldParam = struct('Hext',[0 0 1e6]); %in A/m
    fields = set(fields,'zeemanField',@zeemanField, fieldParam);

    boundaries = boundaryMap(topo);
    boundaries = set(boundaries,'magneticToNonMagnetic',topo.XtoM);

    saveConf = saveConfiguration();
    saveConf = set(saveConf,'filePrefix','larmor');
    saveConf = set(saveConf,'directory',basedir);
    saveConf = set(saveConf,'save_M',true);
    saveConf = set(saveConf,'save_In_Delta_t',1e-12);

    % creating the configuration object and setting all parameters
    conf = configuration();
    conf = set(conf,'topology',topo);
    conf = set(conf,'alpha',0);
    conf = set(conf,'gamma',2.210173e5);
    conf = set(conf,'Ms',Ms);
    conf = set(conf,'M0',M0);
    conf = set(conf,'tStart',0);
    conf = set(conf,'tEnd', 300e-12);
    conf = set(conf,'maxStep',1e-12);
    conf = set(conf,'fields',fields);
    conf = set(conf,'saveConf',saveConf);

    runSim(conf,true,true);
    analyzeResults(basedir);
end

% fit function to derive the precession frequency from the simulation
% results
function analyzeResults(directory)
    data = loadState(directory,'topo,time,M');
    Mx = data.M(:,1);
    s = fitoptions('Method','NonlinearLeastSquares','Algorithm',...
    'Gauss-Newton','TolFun',1e-8,...
    'StartPoint',[0,30,0,0]);
    set(s,'Maxiter',1000000);
    f = fittype('A*sin(2*pi*x/B + C) + D');
    [curve, ~] = fit(data.time*1e12,Mx,f,s);
    fprintf('%2.8f\n',curve.B);
end
```

Code listing 3.6: *M^{3}S-MATLAB* script to run the Larmor-precession test.

In the example code listing the configuration objects `configuration`, `topology`, `boundaryMap`, `fieldMap`, and `saveConfiguration` have been used. The configuration object has been partitioned in sub-objects as a splitting of the problem definition reduces the complexity of each configuration object and simplifies checking their validity. Code listing 3.6 is structured as follows: First the configuration objects `fieldMap`, `boundaryMap`, `saveConfiguration`, and `configuration` are instantiated by calling the corresponding constructor. Further the problem definition is specified using the `set` and `put` methods of each object. The simulation is finally started through calling the `runSim` function passing the main problem definitionobject that is the `configuration` object as an argument to `runSim`.

The `runSim` function internally calls the methods `isConsistent` and `init` of the `configuration` object, that itself hierarchically calls the `isConsistent` and `init` method of the depending configuration objects. In this way an encapsulated dependency check and initialization is realized. Since *MATLAB* does not include the interface concept as language element all interfaces are only modeled by contract. By contract means that the framework expects that the used functions comply to a specific API. This expectation cannot be checked by the compiler. Thus, if the function does not comply to the API an error at runtime is caused. In the following the principle motivation and the differences in the `set` and `put` methods of each configuration object is discussed.

`configuration` object

```
conf = configuration();
conf = set(conf,propertyName,propertyValue);
```

Code listing 3.7: API of the constructor and the `set` method of the configuration object.

The `configuration` object includes all properties for a simulation run. The properties that can be set are: other configuration objects like the `field` object, the `topology` struct, and the solver-specific properties like the initial magnetization or the abort criteria.

`topology` struct

The `topology` struct holds the mesh information. It has been realized instead of an object as a struct since the number of parameters is small and there are no optional parameters provided. Error messages can be provided directly when calling the `topology` function that creates the `topology` struct. As shown in Code listing 3.8 the `topology` struct is initialized by calling the initialization function `topology`, where the parameters `sX,sY,sZ` and `dX,dY,dZ` are passed as arguments. Here the parameters `sX,sY,sZ` specify the sample size and the parameters `dX,dY,dZ` the grid point distance. The resulting struct includes for instance the variables `cellsX`, `cellsY`, and `cellsZ` that hold the number of grid points in each spatial direction.

```
sX = 200e-9; sY = 100e-9; sZ = 10e-9; % in [m]
dX = 2e-9; dY = 2e-9; dZ = 10e-9; % in [m]

topo = topology(sX,sY,sZ,dX,dY,dZ);
topo.cellsX
topo.cellsY
topo.cellsZ
```

Code listing 3.8: The initialization function for the `topology` struct and exemplary access to the struct fields `cellsX`, `cellsY`, and `cellsZ`. As arguments the initialization function expects the size of the problem and the cell size to be specified.

`boundaryMap` object

```
boundaries = boundaryMap();
boundaries = put(boundaries,boundaryKey,inside_indices);
```

Code listing 3.9: API of the constructor and the `put` method of the `boundaryMap` object. As arguments the `put` method expect the `boundaryMap` object itself, a unique key for the boundary, and a Boolean matrix identifying the cells inside the area.

To solve differential equations, it is important to know the sample boundaries. Examples for such boundaries in the micromagnetic simulation are the boundary between ferromagnetic and nonmagnetic materials, between different ferromagnetic materials, or between conducting and non-conducting areas. In *M^{3}S-MATLAB* the specification of the boundaries is handled by the `boundaryMap` object. This object is a map of possible boundaries included in the simulation run. As their arguments the `put` method expects the boundary map object itself, a unique key for the boundary, and a Boolean matrix identifying the cells inside the area. As in the micromagnetic simulation each component uses a subset of the boundaries, the data map structure has been chosen to simplify the access to a specific boundary. At the moment only the key *magneticToNonMagnetic* is supported, but the other boundaries described previously are planned to be included in later versions, as they are useful when multiphysical simulations with different overlapping materials or conductivity/non-conductivity areas need to be performed.

To specify a boundary the `inside_indices` argument has to be specified as a Boolean matrix. As indicated in Fig. 3.2 each position of the matrix corresponds to a grid point. In this matrix for each position inside the area the value `1` is assigned in the matrix. To allow a similar flexibility as *OOMMF* that offers a conversion of image data to a boundary definition, the function `boundariesFromData` has been added to the framework. This function converts a color matrix to a Boolean index matrix as it is necessary to specify a certain simulation boundary.

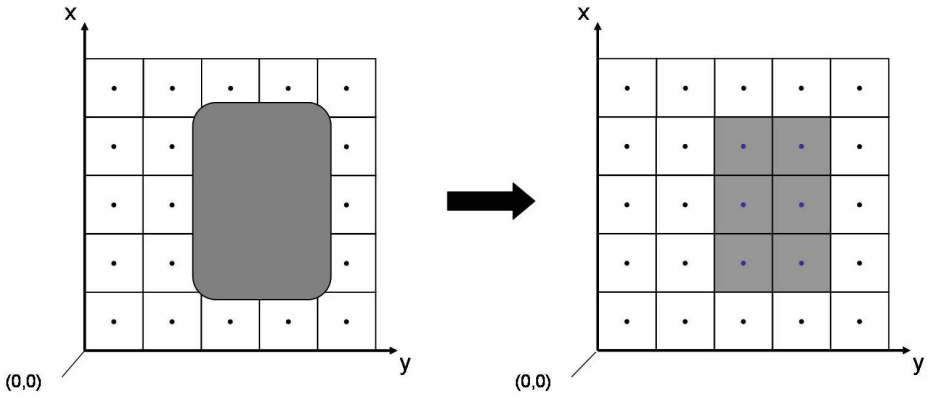


Figure 3.2: Example for a boundary definition. The gray cells are the cells inside the area and correspond to 1 in the Boolean matrix `inside_indices`.

`fieldMap` object

```
fields = fieldMap();
fields = put(fields,fieldname,field_constructor,field_parameters);
```

Code listing 3.10: API of the constructor and the `put` method of the `fieldMap` object. The `put` method expects as its arguments a unique key as identifier for the field, the field constructor as function handle, and the field specific parameter struct.

Various field terms can in principle be added to the effective field. For instance two Zeeman fields, one constant in x-direction, and one cosine-modulated in y-direction could be added to describe fields from two different sources. As another example it is important being able to compare different field implementations and thus to alternate between them. To offer a general way to specify the field terms included in the effective field calculation the `fieldMap` object has been added as configuration object. A field can be added to the `fieldMap` object by calling the `put` method as listed in Code listing 3.10. As arguments the `fieldMap` object itself, a unique key as identifier for the field, the field constructor as a function handle, and the field specific parameter struct have to be specified. The `fieldMap` object calls during its `init` method the field constructor of each field by passing the field specific parameter struct as argument corresponding to the field API shown in Code listing 3.11. Here the field constructor expects as its arguments the `topology` struct, the `boundaryMap` object, the saturation magnetization, and the field specific parameters as a struct.

```
field_obj = field_constructor(topo,boundaryMap,Ms,field_parameters);
H = field(field_obj,M);
E = energy(field_obj,H,M);
```

Code listing 3.11: A field object must be provided in order to be included in the `fieldMap` object.

`saveConfiguration` object

```
saveConf = saveConfiguration();
saveConf = set(saveConf,propertyName,propertyValue);
```

Code listing 3.12: Constructor and `set` method of the `saveConfiguration` object.

The `saveConfiguration` object covers all properties that are related to storing or printing the state. For instance the properties `filePrefix`, `directory`, `save_M`, `startNumber`, and `save_In_Delta_t` can be set. The dynamic state is stored in the time steps specified by `save_In_Delta_t` by storing each step in the corresponding file given by: `<directory>\<filePrefix>_<Property>_<stepNumber>.mat` resulting for instance in `C:\temp\myproblem_M_000002.mat`. Here the `stepNumber` is an ongoing number starting by `startNumber`; `Property` is a placeholder for *M*,*H*,*time*.

Performance issues

A discussion with a *MATLAB* developer on the MATLAB World Tour 2007 in Hannover, Germany, and a later performance test showed, that the object-oriented programming API offered by *MATLAB 2007* is about two times slower as a corresponding non-object-oriented program. Thus the object-oriented design of the configuration objects needs to be evaluated criticality due to its runtime performance.

In retrospective, the configuration objects are used for the specification of the simulation problem and as data containers for the static simulation state within the simulation run. Since the specification of the simulation problem and the initialization of the simulation run are no runtime-critical steps within the simulation run, a loss of about a factor of 2 is acceptable. For the main loop of the simulation run in contrast the object-oriented approach is too slow.

As a solution a recursive conversion of the configuration objects to simple structs has been implemented using *MATLAB* framework functions. This allows to perform the conversion within the solver and to hide this step from the user. To provide the same flexibility as before, therefore the function `toStruct` has been added to the configuration object API. The `runSim` function calls this function after the `init` function.

For the field objects a simple conversion to a struct was not applicable, as the methods and so the interfaces used by the effective field calculation would also be removed. Further runtime performance measurements of the *MATLAB* API revealed, that the use of so-called function handles combined with structs allows to construct an alternative API for a field with a similar flexibility as provided by API shown in Code listing 3.11. A function handle is a *MATLAB* value that points on a funtion definition and thus allows to call a function indirectly.[41]

The new API as shown in Code listing 3.13 returns an object instead of a struct. This struct by contract has to include function handles named `field` and `energy`.

```
field_struct = field_init_function(topo,boundaries,Ms,field_parameters);
H = field_struct.field(M);
E = field_struct.energy(H,M);
```

Code listing 3.13: New API for a field implementation that can be included in the `fieldMap` object. The `field_init_function` returns a struct that holds the field variables and in addition the function handles `field` and `energy` to calculate the field and to calculate the energy, respectively.

3.1.4 Analysis kit

In addition to the solver and the configuration objects, *M^{3}S-MATLAB* includes an analysis kit. The analysis kit provides a simple but powerful framework for the analysis of time-resolved micromagnetic simulation results. In the following the key functions of the analysis kit are introduced and their use is illustrated exemplary. The key functions are `loadState`, `analyzeState`, and `plot2DVectorField`.

`loadState` function

Time-resolved micromagnetic simulation results can easily exceed 40 GB and more storage space. Loading this as a whole in the main memory is not possible. As a solution the `loadState` function has been added to the analysis kit.

```
function result = loadState(directory,loadOptions,step_number)
```

Code listing 3.14: API of the `loadState` function. This function expects as its arguments the directory of the simulation results and a string with comma-separated identifiers. Optionally the time step number can be specified to load the dynamic simulation state only at the corresponding time step.

This function offers a simple API shown in Code listing 3.14 to load simulation results stored by *M^{3}S-MATLAB*. The key feature is that the API provides arguments to load only the necessary simulation properties at certain time steps and thus allows to reduce the main memory usage. To call the `loadState` function the simulation result directory, load options, and optionally a time step number need to be specified. As load options a string of comma-separated identifiers has to be specified corresponding to the different simulation properties, i.e. *conf* for the configuration, *M* for the magnetization, or *H* for the effective field.

By default `loadState` loads all selected dynamic state properties at all stored time steps. Specifying the optional argument `step_number` allows to load the dynamic state

properties only at the selected time step number and thus to load the results piecewise. With both parameters the `loadState` functions have the desired flexibility necessary to avoid running out of main memory.

`analyzeState` function

```
function result = analyzeState(directory, analyzeFunction, ...
                               analyzeOptions,loadOptions, ...
                               [startIndex, endIndex])
```

Code listing 3.15: API of the `analyzeState`-function. As its arguments the directory of the simulation data, the analysis function as a function handle, its specific arguments as a struct, and options for the internal `loadState` call have to be specified. Additionally the optional parameters `startIndex` and the `endIndex` can be specified.

It turned out, that the `loadState` function is often used to implement analysis functions, that load the dynamic state of the simulation time step by time step and derive single properties like the average magnetization or the total energy from the spatially resolved simulation data. To simplify the development of such analysis functions the `analyzeState`-function has been added to the analysis kit as shown in Code listing 3.15. The `analyzeState` function iterates over the dynamic state of each stored time step and calls the specified `analyzeFunction` to derive the desired properties.

As its argument this function expects the following arguments:

- `directory` - the directory where the simulation results are stored.
- `analyzeFunction` - the analysis function as a function handle.
- `analyzeOptions` - a struct of additional options that is passed through to the analysis function by every state.
- `loadOptions` - the output options for internal `loadState` calls.
- `startIndex` and `endIndex` - optional arguments to select a subset of the stored time steps to perform the analysis.

Similar to the `fieldMap` object, the `analyzeState` function expects the analysis function by contract to fulfill the API shown in Code listing 3.16.

```
function res = my_analysis_function(data,analyzeOptions)
    res.dataA = deriveAfromData(data);
    res.dataB = deriveBfromData(data);
end
```

Code listing 3.16: API that an analysis function has to fulfill. As arguments the spatially resolved data of a time step and analysis function specific parameters are passed.

The `analyzeState`-function calls internally the specified analysis function and passes the simulation data of one time step and the `analyzeOptions` as specified before to it. As a result the `analyzeFunction` returns a struct that includes a variable in the result struct for each derived property. These result structs are then combined by the `analyzeState`-function to a struct that includes an array with its value for each time step for each derived property.

In the following the function `analyzeSp4` is exemplary implemented as an example for the use of the `analyzeState` function. This function derives the spatially averaged magnetization $\langle \vec{M} \rangle$ for each time step and finally plots the data calling the user specific function `plotSp4`.

```
function res = analyzeSp4(directory)
    res = analyzeState(directory,@spatialMean,[],'M');
    plotSp4(res.time,res.mean(:,1),res.mean(:,2),res.mean(:,3));
end

function res = spatialMean(loadedData,options)
    res.mean = squeeze(mean(loadedData.M,2));
end
```

Code listing 3.17: Analysis function used for plotting the results of standard problem No. 4.

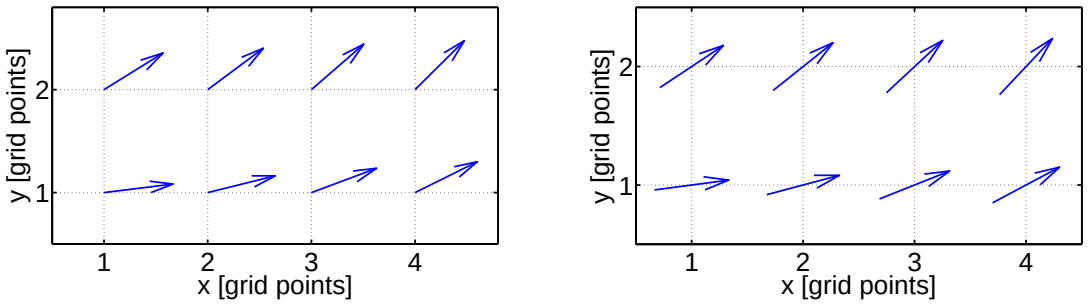
As shown in Code listing 3.17 the example function uses `analyzeState` to derive the spatially averaged magnetization. Therefore the analysis function `spatialMean` is defined implementing the API shown in Code listing 3.16. The `spatialMean` function stores the spatially averaged magnetization for one step in the result struct under the name `mean`. The field `mean` can also be found in the struct returned by the `analyzeState` call holding the mean values for all time steps. The struct also includes the field `time` holding the time step data. This example illustrates, how conveniently user-specific analysis functions can be implemented using the `analyzeState` function.

`plot2DVectorfield` function

Although a large plot functionality is included in *MATLAB*, it was necessary to develop a plot function for vector fields. This was necessary as all vector plots offered by *MATLAB* by default plot the end of an arrow aligned with the grid points as depicted in Fig. 3.3. But physicists imagine the magnetization vector to be aligned like a compass with its center at the grid point.

```
function plot2DVectorField(vec_field,topo,fieldname,plotOptions)
```

Code listing 3.18: API of the `plot2DVectorField` function. As its arguments the vector field data, the `topology`, the field name, and optional plot parameters need to be specified.

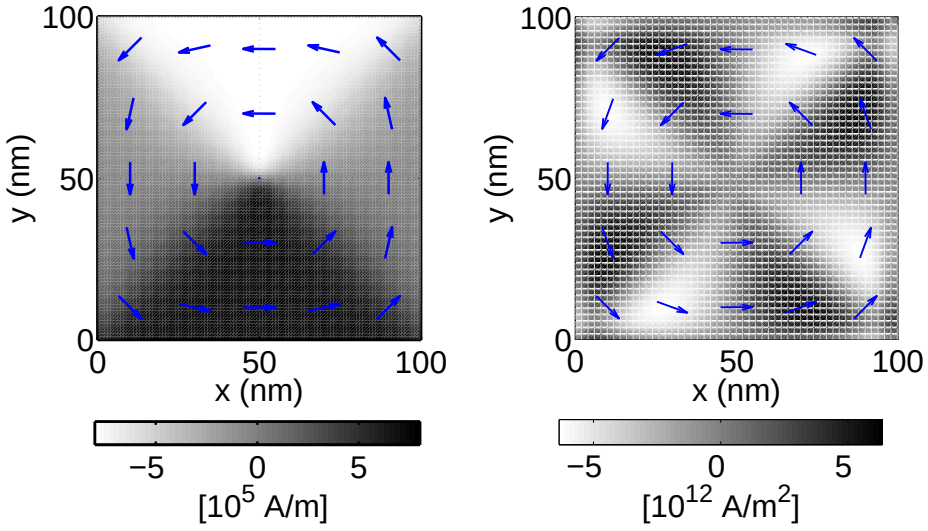


(a) (b)

Figure 3.3: Different plots of a vector field. The arrows are aligned in (a) with the end and in (b) with the center at the grid points.

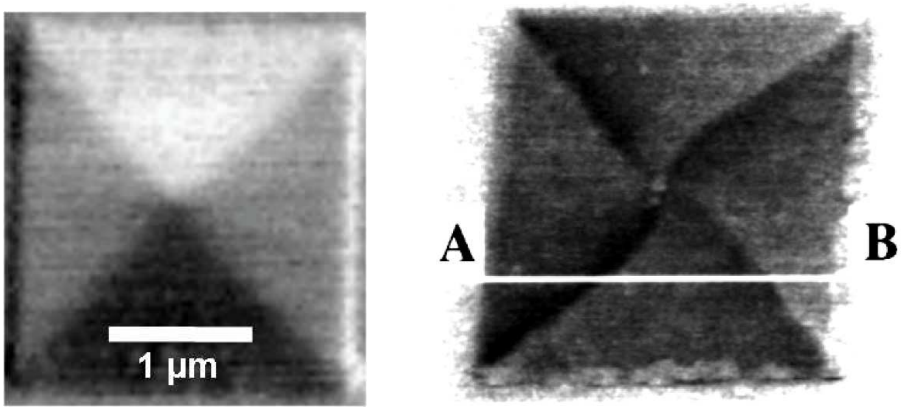
The function `plot2DVectorField` plots two-dimensional vector fields, but can also handle three-dimensional vector fields by offering a slicing option. This option allows to plot a specific z-slice of the three-dimensional data. As arguments `plot2DVectorField` expects the vector-field data, the `topology`, the field name, and optional parameters to configure the plot to the users needs. One important option is the `ground` option. This option offers to select predefined analysis functions that can be applied to the data to derive the ground color of the plot. Figure 3.4 shows plots of a vortex state with two different `ground` settings. The vortex state is a magnetization pattern where the in-plane magnetization curls around the center of the vortex. At the center of the vortex the magnetization turns out of plane. This part of the vortex is called vortex core.

In Fig. 3.4 the x-component and the divergence between x- and y- component of the magnetization have been chosen as `ground` settings. These analysis functions for instance have been included as predefined analysis functions, as they represent important views on the magnetization that correspond to images from experimental measurements. For the magnetization for instance the x-component is measured by x-ray microscopy and the divergence is measured by magnetic force microscopy (MFM).[165] Experimental example images of vortices are shown in Fig. 3.5.



(a) (b)

Figure 3.4: Two-dimensional plots of the magnetization of a vortex using the *M^{3}S-MATLAB* function `plot2DVectorfield`. The arrows show the *x*- and *y*- component of the magnetization. The ground color show (a) the x- component of the magnetization and (b) divergence between the *x*- and *y*- component of the magnetization.



(a) (b)

Figure 3.5: Two-dimensional images of a sample with a vortex as magnetization pattern. (a) shows an x-ray micrograph as depicted by Bolte et al.[166] and (b) shows a magnetic force micograph as depicted by Garcia et al.[167] In (b) the distance between the points A and B is 2 μm

3.1.5 Results and resumé of *M^{3}S-MATLAB*

The implementation of the prototype *M^{3}S-MATLAB* revealed that the use of a CSIDE significantly simplifies the implementation of a micromagnetic simulator. *M^{3}S-MATLAB* implements the core functionality of *OOMMF* and could be realized with a total source lines of code (SLOC) reduction to a tenth compared to *OOMMF*. The extensive documentation and the support of all necessary numerical operations allow for opportunistic programming.

Further a flexible API for the specifiation and starting of the simulation has been implemented, to support complex simulation structures like parameter sweeps or hysteresis loops. Finally, support for the analysis of the simulation results has been realized by the analysis kit, including help functions for the derivation of properties from spatially resolved simulation data and the application-specific plot functions based on the plot framework of *MATLAB*.

Despite these benefits the development of *M^{3}S-MATLAB* revealed two restrictions of *MATLAB* that reduce the maintainability of the simulator:

- *MATLAB* offers a poor support of encapsulation techniques that are essential for the development of large programs: advanced name space concepts with the possibility to control the visibility of components are not supported. Further object-oriented programming which would offer a better encapsulation as the use of a *MATLAB* function leads to performance losses as shown in Sec. 3.1.3.
- For test driven design only the C_0 metric can be estimated using the profiler of *MATLAB*. The profiler is made for performance measurements and not for the measurement of the C_0 metric, meaning that while an overview of the runtime is provided, this is missing for the C_0 results. Thus the profiler is only useful when calling one test by the `runTest`-function.

Concerning the runtime performance a first comparison to *OOMMF* revealed that *M^{3}S-MATLAB* results in a reasonable runtime performance. A detailed investigation of the runtime performance of *OOMMF* and *M^{3}S-MATLAB* as well as the analysis of optimization and parallelization possibilities is decribed in the following section.

3.2 Runtime performance optimization

The optimization of the runtime performance of a scientific program is a difficult task. It requires a deep understanding of the numerical algorithms, the numerical libraries, and the hardware architecture, to be able to identify the fastest implementation of the physical models. In many scientific programs most of the runtime is spend in only 10 % of the program. Hence it is important to identify these parts first to avoid the waste of time in non critical components.

Since *OOMMF*[121] is a highly optimized micromagnetic simulator, in a first step its runtime has been compared with *M*^{3}*S-MATLAB* to identify if there are optimization possibilities covered by *OOMMF* that can be included in *M*^{3}*S-MATLAB*. The first comparison as presented in the Article 3.1.1 showed, that the *M*^{3}*S-MATLAB* is about 2.25 times faster than *OOMMF* using the Euler solver. A more detailed analysis of the comparison revealed, that the increased runtime performance of *M*^{3}*S-MATLAB* compared to *OOMMF* is due to the difference in the number of evaluations of $d\vec{M}/dt$ needed by the Runge-Kutta 4-5 implementation offered by *MATLAB* and the Euler evolver used in *OOMMF*.
Consequently *M*^{3}*S-MATLAB* can be runtime optimized by the runtime of one evaluation of $d\vec{M}/dt$. Further replacing the Runge-Kutta 4-5 algorithm for solving the oridnary differential equation (ODE) by more complex algorithms could reduce the number of needed evaluations. As indicated in Sec. 3.1.2, investigations due to the used ODE would have exceeded the scope of this work and have been addressed in Ref.[104] The present work is restricted to investigations concerning the runtime performance of the calculation of $d\vec{M}/dt$ and does not cover ODE dependent optimizations. In the following different possibilities to optimize the runtime performance of one evaluation of $d\vec{M}/dt$ are considered.

The following article entitled “A Case Study for the Parallelization of a Complex MATLAB Program with Respect to Maintainability” was presented at the 2008 Huntsville Simulation Conference (HSC’08) (that took place between 22 and 23 October 2008 in Huntsville, USA). This article evaluates different sequential optimizations and parallelization possibilities of the demagnetization field calculation due to the runtime performance. Important aspects of this evaluation were to check if the optimizations used by *OOMMF* can be realized for *M*^{3}*S-MATLAB* and which drawbacks these implementations have on the usability and maintainability of the simulator.

3.2.1 Publication HSC'08

A Case Study for the Parallelization of a Complex MATLAB Program with Respect to Maintainability

M. Najafi, G. Selke, B. Krüger, B. Güde,
B. Krause-Kyora, M. Bolte, G. Meier, and D. P. F. Möller

Proceedings of the Huntsville Simulation Conference (HSC'08), J. Gauthier, Ed. San Diego, CA, USA: The Society for Modeling and Simulation, 2008, pp. 309–315

A Case Study for the Parallelization of a Complex MATLAB Program with Respect to Maintainability

Massoud Najafi [a,b*], Gunnar Selke [a,b], Benjamin Krüger [c], Bernd Güde [a,b], Bodo Krause-Kyora [c], Markus Bolte [a,b], Guido Meier [b], and Dietmar P. F. Möller [a]

[*] mailto://mnajafi@physnet.uni-hamburg.de,

[a] Arbeitsbereich Technische Informatik Systeme, Department Informatik, Universität Hamburg, Vogt-Kölln-Straße 30, 22527 Hamburg, Germany,

[b] Institut für Angewandte Physik und Zentrum für Mikrostrukturforschung, Universität Hamburg, Jungiusstraße 11, 20355 Hamburg, Germany,

[c] I. Institut für Theoretische Physik, Universität Hamburg, Jungiusstraße. 9, 20355 Hamburg, Germany

Keywords: simulation of physical phenomena, micromagnetic model, the micromagnetic modeling and simulation kit, parallelization of MATLAB, MATLAB

Abstract

Nowadays simulations are an indispensable part of scientific and technological research. In these fields simulation packages have already reached a high level of complexity. A way to control their complexity is to use intuitive high level programming languages as offered by development environments like MATLAB [1], Maple [2], or Mathematica [3]. Since the complexity of the simulations will certainly continue to grow to an even higher complexity, the demands in computing performance can only be fulfilled by parallelizing the sequential algorithms. The development of techniques to parallelize sequential algorithms is of general interest, since the future of computing could be the parallel processing paradigm [4, 5]. Here we list necessary considerations when parallelizing a complex modular MATLAB program using the example of the micromagnetic modeling and simulation kit M^{3}S [6]. M^{3}S implements the micromagnetic model by dividing the complex algorithm into modules on different algorithmic levels. In this article we investigate different parallelization approaches for the corresponding algorithmic levels and their impact on the maintainability and usability of the simulator.

1. INTRODUCTION

Computer simulation has become a important method for scientific and technological research. Its success is based on the increase of computing performance in the last decades and the development of higher programming languages. Both achievements make the computer simulation a powerful tool for the investigation of complex systems. To translate abstract models of complex systems into controllable computer simulators, it is important to choose adequate computer representations, e.g. programming languages, frameworks or environments. For the development of simulators for continuous systems, environments like MATLAB[1], Maple[2], or Mathematica [3] have become competitors to programming languages like C++ or Fortran. These environments offer intuitive high level programming languages, extensive optimized functionality and the integration of modules implemented in C++ or Fortran. The resulting simulators have a better balance between the maintainability and usability [7] than optimized low-level programs. As shown by Bolte et al. [8], the development of the micromagnetic modeling and simulation kit M^{3}S [6] in MATLAB increased the extensibility, in comparison to other sequential simulators like OOMMF [9] while retaining the performance.

Several examples of recently developed computer architectures indicate that the future of computing could be the parallel processing paradigm [4, 5]. Thus the parallelization of sequential simulators is of large interest to fulfill the growing demand to the computing performance. It is a challenge to parallelize MATLAB-code as the language offers no direct functionality for the parallelization of algorithms on shared memory machines (SMP). The parallelization of the MATLAB-code has an large impact on the maintainability, because it has to be exported to C++ or Fortran and integrated into the simulator again using the offered interface of MATLAB. In this article we list necessary considerations when parallelizing a complex modular MATLAB program using the example of the simulation toolkit M^{3}S [6]. M^{3}S implements the micromagnetic model by dividing it into modules on different algorithmic levels.

The outline of this article is as follows: Section 2 introduces M^{3}S and the micromagnetic model. Section 3 presents a performance analysis of the sequential code of M^{3}S on different algorithmic levels. Section 4 shows possible parallelizations on SMPs for different levels and means of realization as well as the impact on the maintainability and usability of the simulator. We found that the parallelization of the module that calculates the demagnetization field shows an optimal balance between the increase of performance and the maintainability of the simulator. Finally, section 5 presents performance measurements of the resulting parallelization of M^{3}S.

2. THE MICROMAGNETIC MODELING AND SIMULATION KIT M^{3}S

M^{3}S is a framework for the simulation of micromagnetic problems. It uses explicit time integration algorithms [10] and a finite-difference-method (FDM) based spatial discretization to solve the micromagnetic model numerically[8]. To reduce the overall complexity M^{3}S was developed in MATLAB [1]. It is written in the included high level programming language MATLAB-Script [1] using its extensive functionality.

2.1. Micromagnetic Model

In this section the micromagnetic model is introduced [11]. It is the appropriate model to describe ferromagnets on the nano- and micrometer scale. The micromagnetic model describes the magnetization dynamics by a non-linear partial differential equation, the so-called Landau-Lifshitz-Gilbert equation (LLG) and includes the interaction of the magnetization with the so-called effective magnetic field, which is a superposition of different magnetic field terms [12]. The interaction between the magnetization and the effective field leads to a complex dynamic behavior. Except for some analytically feasible systems, the magnetization dynamics can only be solved numerically. To calculate the magnetization dynamics of a ferromagnetic structure numerically, the continuous LLG and the effective fields must be discretized.

2.1.1. Equation of Motion

The LLG equation describes the spatially resolved magnetization dynamics influenced by the effective magnetic field. The FDM-based explicit LLG is given by

$$\begin{aligned}\frac{d\vec{M}_{i,j}}{dt} =& -\gamma\vec{M}_{i,j}\times\vec{H}_{\mathrm{eff},i,j}(\vec{r_{all}},\vec{M_{i,all}}) \\ &-\frac{\gamma\alpha}{M_s}\vec{M}_{i,j}\times[\vec{M}_{i,j}\times\vec{H}_{\mathrm{eff},i,j}(\vec{r_{all}},\vec{M_{i,all}})]\end{aligned} \tag{1}$$

Here M_s is the saturation magnetization, γ is the absolute value of the gyromagnetic ratio, α is the Gilbert damping constant, $\vec{M}_{i,j}=\vec{M}(t_i,\vec{r}_j)$ is the magnetization at at the i-th time step t_i and the position of the j-th cell $\vec{r}_j$, $\vec{H}_{\mathrm{eff},i,j}(\vec{r_{all}},\vec{M_{i,all}}) = \vec{H}_{\mathrm{eff}}(t_i,\vec{r}_j,\vec{r_{all}},\vec{M_{i,all}})$ is the effective field at time step t_i the position $\vec{r}_j$. It is a function of position $\vec{r}_{all}$ and the magnetization $\vec{M}_{i,all}$ of each cell at the time t_i, where *all* indicates all cells.

2.1.2. Effective Field

The micromagnetic model describes all magnetic interactions with the local magnetic moments as magnetic fields. The superposition of all magnetic fields at a point $\vec{r}$ give the effective field for $\vec{r}$. The basic model includes and the anisotropy field, the exchange field, the magnetostatic field, and the Zeeman field as shown in Eq. (2). To simplify our analysis, we focus on the last three magnetic fields and do not discuss the anisotropy field since they suffice to simulate the widely investigated ferromagnetic material Permalloy. Here,

$$\begin{aligned}\vec{H}_{\mathrm{eff},i,j}(\vec{r_{all}},\vec{M_{i,all}}) =& \vec{H}_{\mathrm{Zeeman},i,j} \\ &+\vec{H}_{\mathrm{exch},i,j}(\vec{M_{i,j}},\vec{M_{i,nn}}) \\ &+\vec{H}_{\mathrm{aniso},i,j}(\vec{M_{i,j}}) \\ &+\vec{H}_{\mathrm{demag},i,j}(\vec{r_{all}},\vec{M_{i,all}}),\end{aligned} \tag{2}$$

where $\vec{H}_{\mathrm{X},i,j}=\vec{H}_{\mathrm{X}}(t_i,\vec{r}_j)$ is the corresponding magnetic field at the i-th time step t_i and position of the j-th cell $\vec{r}_j$. *all* indicates all cells, and *nn* indicates the nearest neighbors to the j-th cell.

The demagnetization field, also called the stray field outside the ferromagnet, describes the magnetostatic interactions of the local magnetic moments over long distances within the body. This magnetic field is given by a spatial convolution of the magnetization with the so-called demagnetization tensor as given by

$$\vec{H}_{\mathrm{demag},i,j}(\vec{r_{all}},\vec{M_{i,all}}) = \sum_{j\in all}\hat{N}(\vec{r}_j-\vec{r}_k,\tau_j,\tau_k)\cdot\vec{M}_{i,k}. \tag{3}$$

Here $\hat{N}(\vec{r}_j-\vec{r}_k,\tau_j,\tau_k)$ is the demagnetization tensor for two cuboid ferromagnets in the distance $\vec{R}=\vec{r}_j-\vec{r}_k$, τ_j is the volume of the j-th cuboid, and τ_k is the volume of the k-th cuboid. The demagnetization tensor is given by

$$\hat{N_{jk}}(\vec{R},\tau_j,\tau_k) = \frac{1}{4\pi\tau_j}\int_{\tau_j}\int_{\tau_k}\nabla'_j\nabla'_k\left(\frac{1}{|\vec{R}|}\right)d\tau_j d\tau_k. \tag{4}$$

Newell et al. [13] showed how to solve this equation for two interacting cuboid ferromagnets in a distance $\vec{R}$. The exchange field describes the quantum mechanic interactions between the spins of neighboring atoms. The discretized exchange field is given by

$$\vec{H}_{\mathrm{exch},i,j}(\vec{r}_{nn},\vec{M}_{i,j},\vec{M}_{j,nn}) = \frac{2A}{{M_s}^2\mu_0}\sum_{k\in nn}\frac{\vec{M}_k-\vec{M}_j}{|\vec{r}_k-\vec{r}_j|^2}, \tag{5}$$

where μ_0 is the permeability of vacuum and A is the material dependent exchange constant.

The Zeeman field is the magnetic field from an external magnet and can be spatially homogeneous or inhomogeneous and is either static or dynamic.

2.2. Overview of M^{3}S

The core of M^{3}S consists of the configuration object, the solver and the integrator as shown in Fig. 1. To start a simulation, a configuration object (blue and orange rectangles at

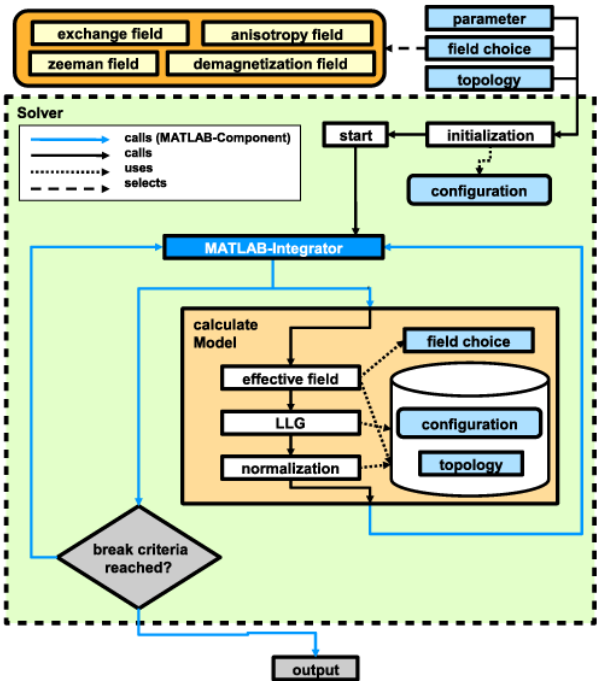


Figure 1. The architecture of M^{3}S showing the interaction of the basic components within a simulation run.

the top of Fig. 1) is created with the specific problem definition. Then the solver is called and the configuration is passed to it. The solver initializes all needed components, e.g., any included fields or the load-and-store functionality. Next the solver starts the time integration loop by calling the integrator. The integrator then uses the submodule 'calculateModel' for the calculation of the time derivative of the magnetization for a time t_i, which is needed to compute the magnetization at time t_{i+1}. This module itself is split up into submodules for the calculation of the effective field, the LLG, and the normalization of the magnetization to the absolute value of M_s. Figure 1 shows the main components of M^{3}S and the flow chart of a simulation run.

3. PERFORMANCE ANALYSIS OF THE SEQUENTIAL IMPLEMENTATION

Next we analyze the runtime performance of the solver. Aim of this analysis is to investigate the runtime distribution of a simulation experiment over the different modules of M^{3}S and to identify significant module. For the investigation of the runtime distribution, the asymptotic time complexity [14] is determined. The time complexity serves to select interesting modules for the runtime measurement.

3.1. Analysis of the Solver

The runtime of the solver during a simulation experiment is split up into the initialization phase and the computation of the time integration loop as shown in Fig. 2. Each simulation step within the loop uses the submodule for the calculation of the LLG, the effective field and the numerical time integration. The effective field in turn uses the concrete modules of the included magnetic fields. The runtime analysis focuses on

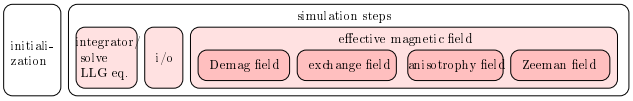


Figure 2. Scheme of the runtime of a simulation run. The simulation run consists of the initialization and a number of simulation steps. In each simulation step, results from the previous simulation step are stored and the LLG is solved. The solution of the LLG requires to calculate the effective field, which is the sum of different magnetic fields. The level of an abstraction is represented by the hue of an object in the figure. The more pale it is the higher its level of abstraction is.

the calculation of a simulation step, because the initialization is called once at the beginning of the simulation and thus influences the overall runtime only slightly. We shall describe the time complexity of all modules starting with the effective field and moving outward to the solver.

Component	time complexity
Zeeman field	$\mathcal{O}(1)$ - static field
	$\mathcal{O}(N)$ - dynamic field
demagnetisation field	$\mathcal{O}(N \cdot \log N)$
exchange field	$\mathcal{O}(N)$
anisotropie field	$\mathcal{O}(N)$
effective field	$\mathcal{O}(N \cdot \log N)$

Table 1. Time complexity of the submodules of the effective field for a system discretized by N cells.

The effective field calculates the superposition of all included magnetic fields as given by eq. (2). The implementation contains no computationally expensive algorithm in itself, hence its time complexity is composed of the complexities of all included magnetic fields. As shown in table 1, the time complexity of the effective field amounts to $\mathcal{O}(N \cdot \log N)$ and is governed by the demagnetization field.

Component	time complexity
effective field	$\mathcal{O}(N \cdot \log N)$
LLG	$\mathcal{O}(N)$
normalization	$\mathcal{O}(N)$
calculateModel	$\mathcal{O}(N \cdot \log N)$

Table 2. Time complexity of the submodules of the 'calculateModel' module for a system discretized by N cells.

As next we investigate the submodules of the 'calculateModel' module. This module also has no computationally expensive runtime, so that its time complexity results from the calculation of the effective field, the LLG and the normalization of the magnetization as listed in table 2 and amounts to $\mathcal{O}(N \cdot \log N)$.

Component	time complexity
calculateModel	$\mathcal{O}(N \cdot \log N)$
time integrator	$\mathcal{O}(N)$
I/O	$\mathcal{O}(N)$
solver	$\mathcal{O}(N \cdot \log N)$

Table 3. Time complexity of the submodules of the solver for a system discretized by N cells.

Finally we investigate the submodules of the solver. The solver uses the 'calculateModel' module, the time integrator, and the I/O module as submodules. As Table 3 shows, the time complexity of a simulation step amounts to $\mathcal{O}(N \cdot \log N)$ and is determined by the 'calculateModel'-module. In conclusion of this part of the performance analysis of the sequential implementation, it is clear that the runtime of a complete simulation run is dominated by the calculation of the demagnetization field. To verify this, we performed runtime measurements for various problem sizes. For simplicity, we only measured the runtime distribution of the exchange field, the demagnetization field and the remaining solver modules.

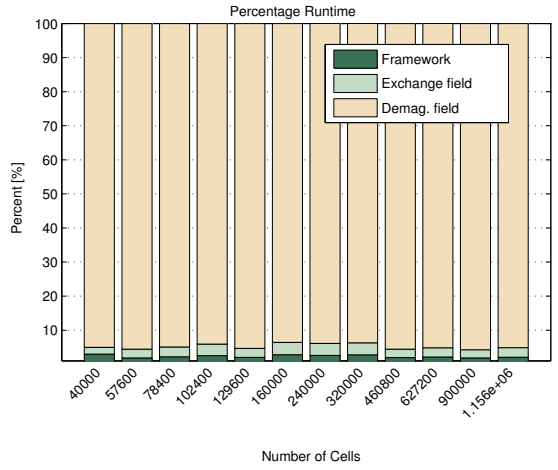


Figure 3. Runtime measurements of M^{3}S for different number of cells. The percentage runtime is split up into the runtime of the demagnetization field, the exchange field, and th framework.

3.2. Analysis of the Demagnetization Field

We performed three different simulations runs for each system size. The first simulation run includes only a static Zeeman field with a complexity of $\mathcal{O}(1)$. This determines the runtime of the solver alone since the complexity of computing the Zeeman field can be neglected. The second simulation includes a Zeeman field and the exchange field. The third simulation includes all three magnetic fields. The difference in runtime between these three simulation runs then determines the runtime of the solver, the exchange field, and the demagnetization field. Figure 3 shows the runtime measurements, which confirm the results of the asymptotic time complexity analysis. Intuitively, the demagnetization field as given by Eq. 3 can be computed using a loop-based convolution which has a time complexity of $\mathcal{O}(N^2)$. The use of the FDM allows to use the fast Fourier transform (FFTs) for the implementation of the convolution which reduces the time complexity of the convolution to $\mathcal{O}(N \cdot \log N)$. M^{3}S implements the FFT-based convolution, using its optimized 3D-FFT or 3D-IFFT implementation. The optimized 3D-FFT is a customization of the 3D-FFT of MATLAB, that considers the properties of the demagnetization tensor and the magnetization, i. e. real values that have a high symmetry in Fourier space which allows to reduce the number of convolutions to 7/12.

The time complexity of the demagnetization field is split up into the component-wise FFT of the magnetization, the cell-wise multiplication of the Fourier-transformed demagnetization tensor and Fourier-transformed magnetization, and the component-wise inverse FFT of the Fourier-transformed demagnetization field as listed in table 3.2.. The analysis of the time complexity shows that the runtime of the demagnetization field is equally spend on the calculation of the optimized 3D-FFTs and of the optimized 3D-IFFTs.

Component	time complexity
optimized 3D-FFTs	$\mathcal{O}(N \cdot \log N)$
$\hat{N}_{\mathsf{FFT}} \cdot \vec{M}_{\mathsf{FFT}}$	$\mathcal{O}(N)$
optimized 3D-IFFTs	$\mathcal{O}(N \cdot \log N)$
demagnetization field	$\mathsf{O}(N \cdot \log N)$

Table 4. Time complexity of the submodules of the demagnetization field for a system discretized by N cells.

We also performed runtime measurements of the submodules of the demagnetization for relevant problem sizes. Unlike the solver, the optimized 3D-FFT/3D-IFFT is not in the asymptotic range for these problem sizes. Figure 4 shows that the multiplication consumes 15-25% of the runtime. The difference between the optimized 3D-FFTs and the optimized 3D-IFFTs occurs, because some optimizations of the 3D-FFT can only be implemented with MATLAB efficiently for the optimized 3D-FFTs, not for the inverse FFT.

4. PARALLELIZATION

In the last sections we showed, which analysis is necessary, to identify the partition of the complex system into mod-

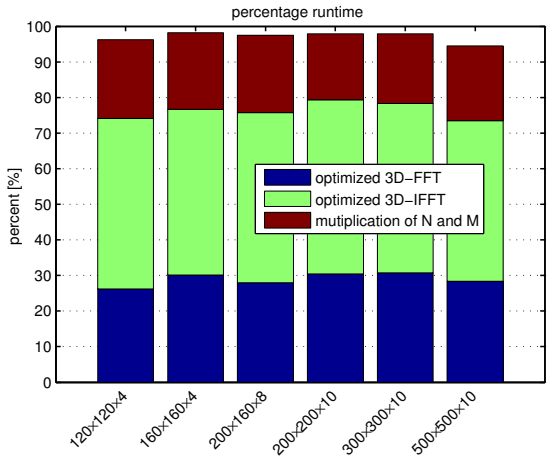


Figure 4. Runtime measurements of M^{3}S for different number of cells. The percentage runtime is split up into the runtime of the optimized 3D-FFT, the optimized 3D-IFFT, and the cell wise multiplication of $\hat{N}_{FFT}$ and $\vec{M}_{FFT}$.

ules within a modular MATLAB program. In this section we search for possible parallelizations for the modules of M^{3}S on SMPs, in particular for the different solutions that exist for the demagnetization field. We discuss the expected increase in performance for each solution by taking the results of the performance analysis into account. Besides that, we also depict the impact of the solutions on the maintainability of the simulator. The maintainability is decreased by the externalization of a modules, because domain specific aspects are formulated in C++. We determine the speedup, efficiency, and scalability of each solution and discuss the affect on the runtime of a simulation experiment.

4.1. Possible Parallelization of the Optimized 3D-FFT/3D-IFFT

The optimized 3D-FFT and the optimized 3D-IFFT consume between 75 and 80 percent of the runtime for the calculation of the demagnetization field as shown in Figure 4. Thus these functions are a good starting point for the search of optimal parallelization approaches. Generally a 3D-FFT can be computed by successive 1D-FFTs along each dimension as shows in Figure 5. During the computation of the 1D-FFTs along a dimension all 1D-FFTs are independent from each other and can be parallelized with an negligible overhead, as shown in Fig. 6. We expect a speedup of the 3D-FFT proportional to the number of processors [15]. The speedup of the parallelized 3D-FFT leads to a maximum speedup of the demagnetization field of 5, because the multiplication of the demagnetization tensor with the magnetization takes about 20 percent of the runtime.

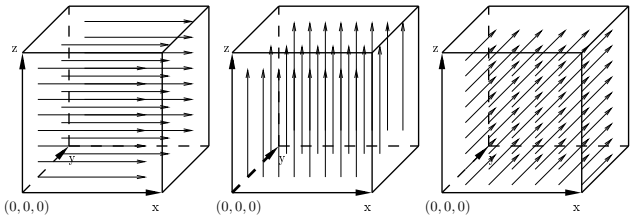


Figure 5. Schema of the representation of a 3D-FFT through 1D-FFTs along each dimension. For each dimension the arrows show the direction of the 1D-FFTs.

Hence the efficiency of this solution decreases from a number of processors rapidly and can only be increased by the parallelization of other modules [15]. Furthermore this approach shows a good scalability [16] for the 3D-FFT, since for a 3D-FFT many 1D-FFTs have to be computed. The impact of the parallelized 3D-FFT on the maintainability is small, because it encapsulates only few physical constraints.

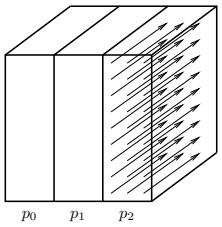


Figure 6. Distribution of the 1D-FFT onto the processors p_0, p_1, p_2.

4.2. Possible Parallelization of the Matrix Multiplication

For each discretization cell, the Fourier-transformed magnetization must be multiplied with the Fourier-transformed demagnetization tensor. This operation can also be parallelized easily, because the multiplications for two cells are independent from each other. We expect a speedup of the multiplication proportional to the number of processors. Since the multiplication takes 15 to 25 % of the runtime, the speedup of the parallelized multiplication leads only to a maximum speedup of the demagnetization field of 20 percent. This solution is only useful in combination with the parallelized 3D-FFT. The impact on the maintainability is small, because the parallelized operation just replaces the existing operation of MATLAB.

4.3. Possible Parallelization of the Solver

To parallelize M^{3}S on the level of the solver is not useful, because no great performance gain is expected. The parallelization on this level of abstraction has an great impact on

the maintainability, and leads to a externalized implementation of most parts of the simulator.

5. RESULTS

Section 4. described possible parallelization on different abstraction levels. To get an optimal solution for a multicore system with 4 to 8 cores, we decided to take the parallelized 3D-FFT. This solution shows the optimal balance between the increase of performance and the maintainability of the simulator for such a multicore system. Therefore we externalized the 3D-FFT form MATLAB and used the FFTW library [17] to calculate the 1D-FFTs. In this Section we finally present the performance measurements of the resulting parallelization of M^{3}S. As shown in Fig. 7 the parallelized 3D-FFT leads

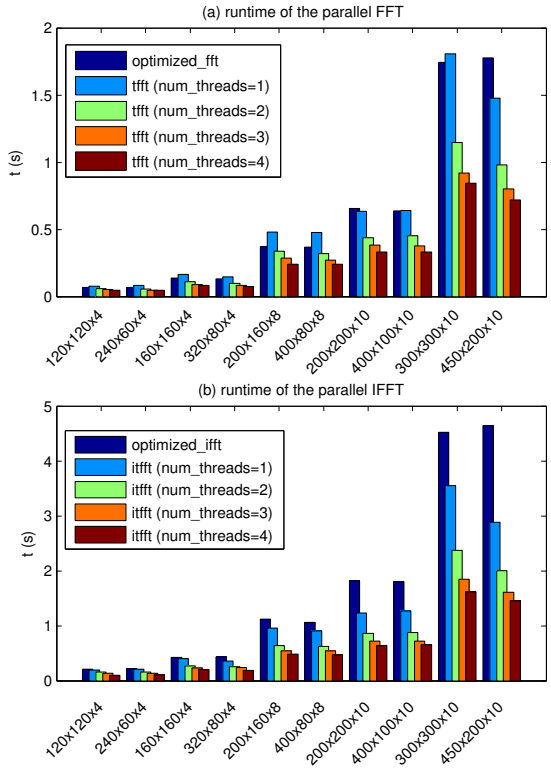


Figure 7. Comparison of the runtime of different 3D-FFT implementation for different number of cells. a.) shows the optimized 3D-FFT and the parallelized 3D-FFT (denoted as tfft) using up to 4 processors. b.) shows the optimized 3D-IFFT and the parallelized 3D-IFFT (denoted as itfft) using up to 4 processors.

to the predicted efficiency of the parallelized M^{3}S. The measurements also illustrate the maximum speedup of this solution of 5. The difference between both 3D-FFT implementation performed on one processor is based on optimizations within the optimized 3D-FFT that are not transferred to the parallelized 3D-FFT yet.

6. SUMMARY AND OUTLOOK

In this article we discussed different approaches for the parallelization of the complex MATLAB program M^{3}S. We showed , that such a complex modular program can be parallelized on different abstraction levels. For this aim it is necessary to consider software complexity besides time complexity for the choice of a parallelization. In the future we will investigate how MATLAB compilers like Star-P change this decision process.

7. ACKNOWLEDGMENTS

Financial support by the Deutsche Forschungsgemeinschaft via SFB 668 ”Magnetismus vom Einzelatom zur Nanostruktur” and via Graduiertenkolleg 1286 ”Functional metal-semiconductor hybrid systems” is gratefully acknowledged.

REFERENCES

[1] 2008. MATLAB,http://www.mathworks.co.uk/products/matlab/.

[2] 2008. Maple,http://www.maplesoft.com/.

[3] 2008. Mathematica,http://www.wolfram.com/.

[4] J. Held, J. Bautista, and S. Koehl. From a few cores to many: A tera-scale computing research overview, white paper, intel corporation, 2006.

[5] M. Gschwind, P. Hofstee, B. Flachs, M. Hopkins, Y. Watanabe, and T. Yamazaki. A novel simd architecture for the cell heterogeneous chip multiprocessor. *Hot Chips*, 17, 2005.

[6] M. Najafi, B. Krüger, S. Bohlens, G. Selke, B. Güde, M.-A. B. W. Bolte, and D. P. F. Möller. The micromagnetic modeling and simulation kit m^{3}s for the simulation of the dynamic response of ferromagnets to electric currents. *GCMS'08: Proceedings of the Conference on Grand Challenges in Modeling & Simulation*, pages 427–434, 2008.

[7] B. W. Boehm, J. R. Brown, and M. Lipow. Quantitative evaluation of software quality. *ICSE '76: Proceedings of the 2nd international conference on Software engineering*, pages 592–605, 1976.

[8] M.-A. B. W. Bolte and M. Najafi. Simulating magnetic storage elements: Implementation of the micromagnetic model into matlab - case study for standardizing simulation environments, 2007. SCSC 07:Proceedings of the 2007 Summer Computer Simulation Conference, 525.

[9] M. J. Donahue and D. G. Porter. Object oriented micromagnetic framework, OOMMF user's guide, version 1.0, 1999. Interagency Report NISTIR 6376, National Institute of Standards and Technology, Gaithersburg, MD.

[10] J. C. Butcher. *Numerical methods for ordinary differential equations*. John Wiley and Sons Inc., West Sussex, UK, 1963.

[11] W. F. Brown Jr. *Micromagnetics*. Interscience Publishers, New York, NY, 1963.

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

[13] A.J. Newell, W. Williams, and D.J. Dunlop. A generalization of the demagnetization tensor for nonuniform magnetization. *J. Geophys. Res. 98*, 1993.

[14] D. Knuth. *The Art of Computer Programming, Volume 1: Fundamental Algorithms, Third Edition.* Addison-Wesley, 1997.

[15] D. L. Eager, J. Zahorjan, and E. D. Lazowska. Speedup versus efficiency in parallel systems. *IEEE Trans. on Computers*, 38:408–423, 1989.

[16] M.D. Hill. What is scalability? *Comp. Arch. News*, 18: 18 – 21, 1990.

[17] M. Frig and S. G. Johnson. The design and implementation of fftw3. *Proceedings of the IEEE*, 93:216 – 231, 2005.

Demagnetization field optimizations

The calculation of the demagnetization field via the demagnetization tensor (see Sec. 2.2.3) as given by Eq. (2.26) can be sped up using the fast Fourier transformation (FFT). Figure 2.4 shows the necessary steps to calculate the demagnetization field using the FFT. Donahue et al.[168] identify four main performance optimizations for the demagnetization field calculation:

1. The optimization arises for the calculation of the three dimensional FFT and inverse FFT (IFFT). The magnetization as decribed in Sec. 2.2.3 is expanded by zeros (so called zero-padding) to match the size of the expanded demagnetization tensor. Hence for the three-dimensional-FFT calculation one-dimensional-FFT calculations along strips being fully zero are performed. Further for the three-dimensional inverse FFT (IFFT) one-dimensional IFFT calculations can be skipped along strips that have no contribution to the physically valid region. This optimization leads as discussed by Donahue et al.[168] to an increase in runtime performance of 41 % to > 50 % depending on the exact number of cells in each dimension.

2. The one-dimensional FFT along the first dimension can be performed as real FFT. The real FFT results in a data array[169] that is symmetric along the second dimension. Thus only half of the one-dimensional-FFT calculations along the second dimension need to be calculated. The other half of the data array can be transformed by a symmetric copy. This results in a further reduction of the runtime.

3. A special 1D-FFT implementation that takes the symmetries into account and thus reduces the necessary main memory usage, results in a reduction of load and store commands.

4. Donahue et al. showed in 2009 how cache optimizations arise when the instructions in the distinct steps 1, 2, 3 in Fig. 2.4 are reordered.[168] The reordering further reduces the necessary load and store access on the main memory by a factor of 3 and 6, respectively.

The previous article covers the first two optimizations as they can be found in the official *OOMMF* version 1.2a3. The third optimization was not covered as this necessitates to implement an own one-dimensional FFT and thus to loose the advantages of novel FFT packages. These advantages will be discussed in the following sub-section. The forth optimization has not been covered as it was published in the end of 2009 by Donahue et al..

Consequently it is possible to include sequential optimizations and parallelization possibilities in the *M*^{3}*S-MATLAB* efficiently by implementing the convolution in *C* and to include it into *MATLAB*. This solution has the drawback, that a compiler and a corresponding configuration file for *MATLAB* is necessary. As the 64-bit *MATLAB* version offers no default *C*-compiler for different operating systems the flexibility of the prototype gets lost by this solution in the long run.

3.2.2 Using the best zero-padding

The investigation related to the calculation of the demagnetization field shows, that the calculation depends significantly on the runtime performance of the three-dimensional-FFT that itself depends on the one-dimensional-FFT algorithm, as discussed in the previous sub-sections. The use of novel FFT libraries[78,170,171] reveals a different possibility for runtime optimizations that will be discussed in the following.

The FFT algorithm is a divide-and-conquer algorithm[123] with an asymptotic complexity of $O(Nlog(N)$. This means, that the algorithm needs $A_n nlog(n) + C(n)$ operations to calculate the one-dimensional FFT of an array of length n. A_n is a constant factor depending on n and the concrete implementation, while $C(n)$ is a function that is small for large n and thus can be ignored for the following investigations. The simplest FFT algorithm is the so-called radix-2 algorithm: each step of the divide-and-conquer algorithm halves the data array. A restriction in the radix-2 algorithm is that the data array has to have a length, which is a power of two in order to apply this algorithm.

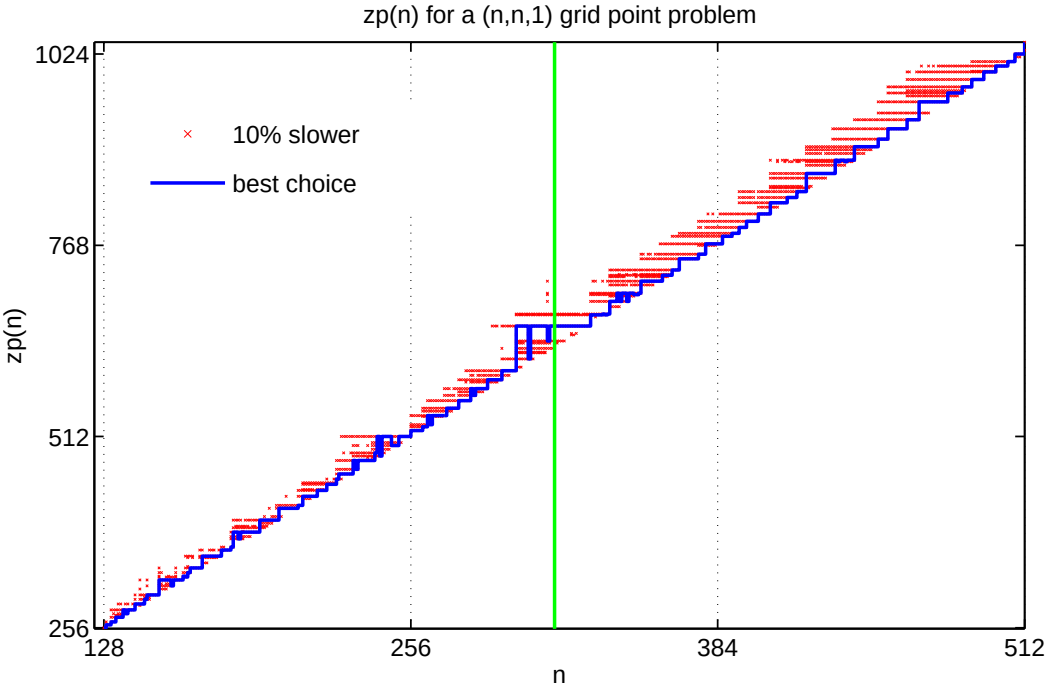
This problem can be solved in two ways:

1. Extend the array properly so that its length complies to the restriction.
2. Use a FFT algorithm that can handle the given array length.

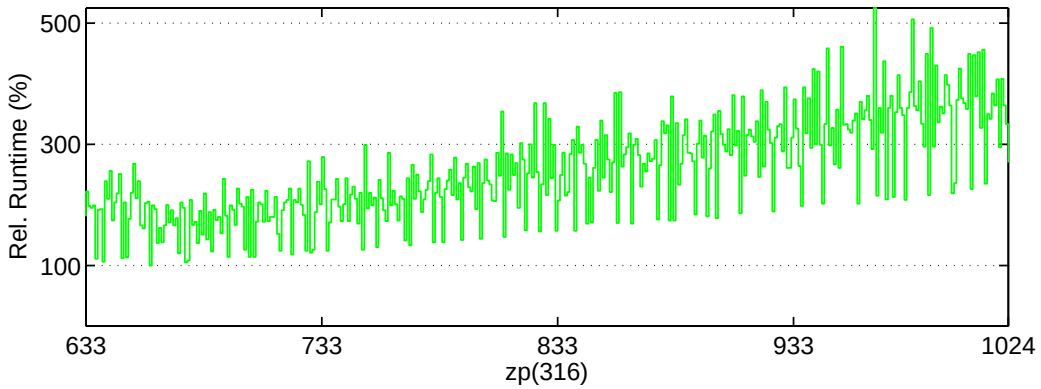
As reviewed by Duhamel et al.[169] there exist a variety of FFT algorithms that have no restriction like the simple radix-2 algorithm. These algorithms often differ in the prefactor A_n as well as in the accuracy. Novel FFT libraries like FFTW[50,78,172] or *SPIRAL*[170] include implementations of different FFT algorithms and adaptive selection techniques to choose the concrete algorithm applied to the data array.

An analysis of the demagnetization-field implementation in *OOMMF* revealed, that it similarly to *M^{3}S-MATLAB* does not cover periodic boundary conditions. This allows to expand the data array properly by adding zeros to it. This technique is also called *zero-padding*. As shown in Fig. 2.4 for the calculation of the demagnetization field, zero-padding has already been used in the basic algorithm discussed in Sec. 2.2.3 to resize the magnetization to the same number of grid points as the extended demagnetization tensor resulting in the number of elements $(P_x, P_y, P_z) = (2p_x - 1, 2p_y - 1, 2p_z - 1)$. Here p_i is the number of grid points in the i - direction with $i \in x, y, z$. *OOMMF* now zero-padds the zero-padded magnetization and the extended demagnetization tensor further to the number of grid points $(zp(p_x), zp(p_y), zp(p_z))$, where $zp(a) = np2(a)$, where $np2(a)$ is the next number larger $2a - 1$ that is a power of two. This means for instance, a sample discretized by (200,200,10) grid points corresponds to an extended tensor of dimensions (399,300,19). In *OOMMF* the zero-padded demagnetization tensor has then the dimensions (512,512,32).

The choice of $zp(a) = np2(a)$ as zero-padding in *OOMMF* is due to the radix-2 based FFT implementation used by *OOMMF* as indicated in Sec. 3.2. For zero-padding in principle however each number $zp(a) = 2a - 1 + z$ with $z \geq 0$ is valid.



(a)



(b)

Figure 3.6: (a) Runtime performance measurements for calculation of the demagnetization-field for all possible zero-padding choices $zp(n)$ for systems $(n,n,1)$ as the number of grid points. For each number n the optimal zero-padding choice is marked blue and the choices with maximally 10 % slower runtime as the optimal zero-padding choice are mearked red. The green line marks the results from the runtime measurement for $n = 316$. (b) shows the corresponding runtime performance measurement, where $zp(316)$ is varied from 633 to 1024. The measured runtime is depicted relatively to the fastest measured runtime.

For novel FFT implementations as the *FFTW* library[75,76,78,173] or *SPIRAL*[170] much better zero-padding strategies than $zp(a) = np2(a)$ exist. As shown in Fig. 3.6 the zero-padding strategy $p1(a) = 2a$ offers on the average a better performance as $np2(a)$. This is due to the fact that novel FFT libraries are on average faster for even numbers. But also the $p1(a)$ zero-padding strategy is up to two times slower as the optimal zero-padding $ozp(a)$. Novel FFT libraries support many FFT algorithms. Hence the optimal zero-padding $ozp(a)$ of both magnetization and demagnetization tensor cannot be anticipated. Thus it is necessary to derive adaptive measurement techniques to find a nearly optimal choice for the zero-padding.

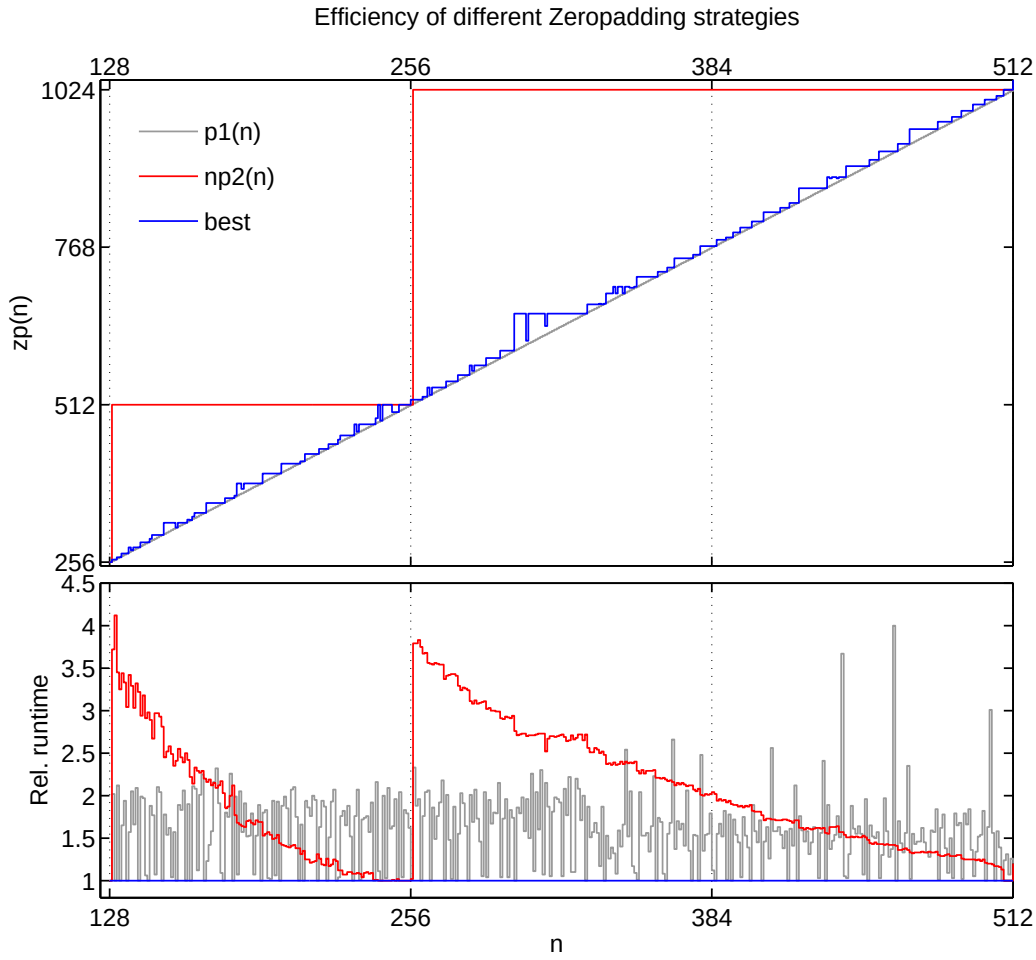


Figure 3.7: Measurement of the runtime performance for the calculation of the demagnetization field for the different zero-padding strategies $np2(n)$, $p1(n)$, and the optimal choice $ozp(n)$ for systems with a number of grid points of $(n,n,1)$. For each number n the zero-padding selected by the strategies and the resulting runtime of the demagnetization field calculation relative to the runtime of $ozp(n)$ are depicted. For this comparison n has been varied from 128 to 512.

For the following investigations the optimized implementation of thedemagnetization field that is fully written in *MATLAB* has beenused. However, the findings are in principle

adaptable to the C++module presented in Sec. 3.2.1. Fig. 3.6 shows the optimal measured zero-padding $ozp(n)$ for samples described by $(n,n,1)$ grid points, where n was varied from 128 to 512. A comparison with the zero-padding strategies $np2(n)$ and $p1(n)$ as shown in Fig. 3.7 revealed that the best choice of $zp(n)$ can lead to a performance increase of a factor of four compared to the use of $np2(n)$ and to a factor of two compared to the use of $p1(n)$.

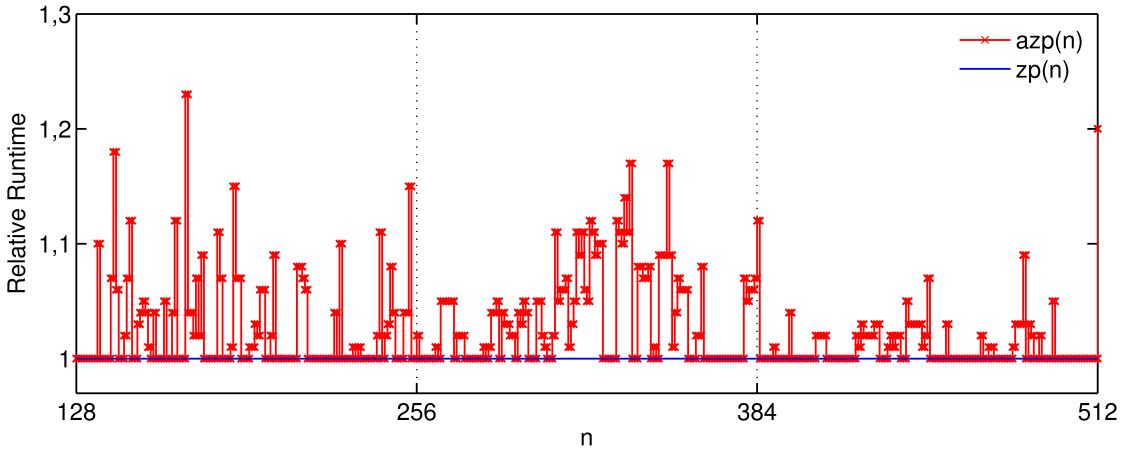


Figure 3.8: For systems with number of grid points $(n,n,1)$ the efficiency of the zero-padding strategy $azp(n)$ in relation to the optimal measured strategy $ozp(n)$ is depicted.

Based on these findings, the adaptive zero-padding strategy $azp(n)$ to select the number for $zp(n)$ has been developed. This strategy selects the zero-padding by the following algorithm that is partitioned in three steps:

1. For each spatial dimension runtime measurements of the one-dimensional-FFT for all numbers in the range of $p1(n)$ to $np2(n)$ are performed.
2. Runtime measurements of 10 demagnetization field calculations are performed for each combination $(zp(n_x),zp(n_y),zp(n_z))$ of the three best zero-padding choices for each dimension n_x,n_y,n_z found in the previous step.
3. The fastest combination is chosen as the desired zero-padding.

Figure 3.8 shows that the zero-padding chosen by $azp(n)$ results in a runtime on average 2.5 % and on rare occasions up to 23 % slower than the optimal measured zero-padding $ozp(n)$.

3.2.3 Landau-Lifshitz-Gilbert equation (LLG)

After having optimized and parallelized the calculation of the demagnetization field, the runtime of its calculation is reduced in maximum by a factor of four. Thus the relative runtime of the LLG becomes also important. In the worst case, the calculation

of the LLG takes about 28 % of the runtime of a `calculateModel` call. Further studies using the Profiler show that 89 % of the runtime of the `LLG` is spend in the `cross` function provided by *MATLAB* (listed in Tab. 3.3). Since this `cross` function is developed for the *MATLAB* framework, it includes checks of the inputs matrix transformations for a flexible matrix cross product that can handle multidimensional matrices.

LLG using `cross`

Function Call `LLG_DGL`	**Runtime (ms)**
`LLG_DGL()`	103.6
`MxH=cross(M,H);`	44.0
`MxMxH =cross(M,MxH);`	45.4

LLG using `myCross`

Function Call `LLG_DGL`	**Runtime(ms)**
`LLG_DGL()`	44.9
`MxH=myCross(M,H);`	14.4
`MxMxH =myCross(M,MxH);`	15.4

Table 3.3: Runtime of one `LLG_DGL` function call for a 256x256x4 cells system using the provided *MATLAB* function `cross` and the optimized function `myCorss`.

This flexibility is not necessary for the implementation of the LLG as the matrices passed to the `cross` function have always the shape of $3 \times n$, where n is the number of grid points. Implementing a new function `myCross` excluding the checks allows to reduce the runtime of the `LLG` function by a factor of 2.31 as shown in Tab. 3.3.

3.2.4 Result of the runtime performance optimization

The use of *MATLAB* as CSIDE to develop the M^{3}S prototypes offered a reasonable runtime performance compared to *OOMMF*. The remaining performance gap between *M^{3}S-MATLAB* and *OOMMF* could be associated to sequential optimizations that are not expressible by the built-in *MATLAB* functions.[82,84] The remainingcoptimizations can be realized by externalizing the runtime criticalccalculation to *C/C++* or *FORTRAN* and interfacing them as optional implementations. In this way the portability is increased, as a runnable software is always available. The externalization has to be performed with respect to all software quality criteria. Taking only the runtime performance into account would on long terms result in a reimplementation of the simulator.

On the other hand it turned out, that the use of novel libraries like the *FFTW* library offers new optimizations that cannot be expressed in the built-in FFT implementation used in *OOMMF*. The question is now, if the restrictions revealed in Sec. 3.1 and 3.2 are general for a CSIDE or only *MATLAB*-specific. This question is evaluated in the following section by comparing *M^{3}S-MATLAB* with two other CSIDE based prototypes.

3.3 Evaluation of different CSIDEs

In the following *Python/SciTools* and *Java/JSA* as CSIDE are evaluated focussing on the support for software engineering concepts and on the resulting performance. The development of *M³S-MATLAB* showed that the critical aspects mainly occur in the solver. Thus for the new prototypes *M³S-Java* and *Nmag-FD* only the solver including the exchange field, the demagnetization field, and the Zeeman field has been reimplemented.

3.3.1 Support for software engineering concepts

Support of high level programming language concepts

The basic idea to structure programs in *MATLAB* is to implement *MATLAB*-functions.[41] These functions can internally be modularized in private functions accessible only within a function file. A module concept is not provided. Instead the `MATLAB-PATH` is used to organize imports. It can be set by the user within the IDE or pragmatically by a setup function. If a statement is called, *MATLAB* searches the `MATLAB-PATH` for a variable or function that corresponds to the signature following a defined strategy. For small programs this is a simplification, as the user can put all files in one directory and does not need to care about imports. For large programs this strategy results in an unintuitive overloading of functions resulting in bugs that are difficult to find. As described in Sec. 3.1.3, *MATLAB* in principle supports object-oriented programming (OOP), but the language support is poor and results in a performance loss that restricts the reasonable use of OOP in *MATLAB*.

For *Python* equally to *MATLAB* a source path needs to be set. This path is specified by the system variable `PYTHONPATH`. Using *Pydev* as development environment takes on the management of the source path definition. As introduced in detail by Langtangen[34] *Python* supports a powerful module concept, OOP, and functional programming. Using these concepts has no effect on the runtime performance of the program. For *Python* it is important to use the provided progamming-language elements carefully. For instance module names can be renamed in the import statement to shorten statements. Using this progamming-language element includes the risk, that other developers reading the code overlook this renaming and misinterprete the new name.

For *Java* similar to *MATLAB* and *Python*, the source- and classpath need to be specified. Here the IDE *Eclipse* takes on the management of these two path definitions. *Java* is an OOP progamming language and thus supports namespaces and OOP. Functional programming is not supported but can be simulated. Investigations revealed, that as for other numerical calculations, OOP results in a reduced runtime performance due to OOP overhead. This means, realizing a multiplication of two arrays of complex numbers by representing each number in the array by objects results in a poor runtime performance. Implementing instead a complex array that calculates theoperation for the whole array non

object-oriented results in a similarruntime performance as the fully non object oriented approach. Thus OOP has tobe used either for non runtime-critical components or for the realization ofwhole array operations.[174]

Test coverage

As described in Sec. 3.1.2 the test coverage measurement support of *MATLAB* is limited to the C_0 test coverage metric. A measurement of the C_1 metric for *M^{3}S-MATLAB* hence would necessiate the development of additional tools. In contrast to *MATLAB*, *Python* and *Java* offer to measure the C_0 and C_1 test coverage.

For *Python* the tool *py.test* offers the needed functionality. This tool is both test driver and test coverage tool in one. *py.test* can be called for a directory to run all tests with a certain wildcard like `sim*` or by specifying the exact path of the test function to be run. For each performed test, a report is printed on the standard output. Part of the results are a measurement of the C_0 and C_1 metric.

For *Java* the tool *EMMA* has been used to measure the C_0 and C_1 test coverage for all existing unit tests. *EMMA* included in *Eclipse* offers a comfortable illustrated overview of the test coverage results for all tests as exemplary shown in Fig. 3.9 for *M^{3}S-Java*. With this support a user can conveniently identify components that are not properly tested. Furthermore as exemplary shown in Fig. 3.10 for each measured class the C_1 coverage of the lines of the code are marked by three colors. Lines of code that are not covered are marked by the color red, partially covered lines of code by yellow, and fully covered lines of code by green.

Software quality measurement

The measurement of the static software quality has become an important method for checking the quality of a software as it allows to identify quickly so called bad smells. Bad smells are programming structures that are known as fault-prone. Typically the code is checked by a *lint*[79] adaptation and by the use of measurement of standard metrics as listed in Balzert et al.[100]

lint was the first tool for checking the correct use of coding standards in *C*. *MATLAB*, *Python*, and *Java* all offer such tools. In *MATLAB* it is called *mlint*[41] and is integrated in the development environment by default. For *Python* and *Java* several open source tools exist for this purpose. In the following the tool *pylint*[175] for *Python* and the tool *lint4j*[176] for *Java*are used.

Furthermore *Python* and *Java* offer several tools for the measurement of the static quality metrics. In this project the open source packages *Metrics*[177] and *PyMetrics*[178] for *Java* and

Element	Coverage	Covered Instructions	Total Instructions
src - M3S_java_0.2	87,2 %	10922	12519
(default package)	0,0 %	0	134
de.unihamburg.informatik.m3s.environment	83,7 %	1008	1205
de.unihamburg.informatik.m3s.environment.basic	88,0 %	242	275
de.unihamburg.informatik.m3s.environment.fft	78,1 %	763	977
de.unihamburg.informatik.m3s.environment.fft.jnt	0,0 %	0	141
de.unihamburg.informatik.m3s.environment.fft.JTransforms	35,5 %	70	197
de.unihamburg.informatik.m3s.environment.io	64,5 %	363	563
de.unihamburg.informatik.m3s.magneticfields	100,0 %	9	9
de.unihamburg.informatik.m3s.magneticfields.demagnetization	93,6 %	4048	4326
de.unihamburg.informatik.m3s.magneticfields.demagnetization.filter	97,2 %	1460	1502
de.unihamburg.informatik.m3s.magneticfields.exchange	100,0 %	484	484
de.unihamburg.informatik.m3s.math.array	91,9 %	1128	1228
de.unihamburg.informatik.m3s.micromagneticmodel	95,1 %	234	246
de.unihamburg.informatik.m3s.solver	92,3 %	143	155
de.unihamburg.informatik.m3s.solver.integrator	95,0 %	800	842
de.unihamburg.informatik.m3s.testFramework	72,3 %	170	235

Figure 3.9: Tabular display of the test coverage results offered by the tool *EMMA*. The results are listed ordered by packages. A quick overview can be gained by looking at the first column, where the results are summarized by a color bar. More detailed information is given by the columns *Coverage*, *Covered Instuctions*, and *Total Instructions*, corresponding to the C_0 test coverage. For the C_1 test coverage *EMMA* offers to display additional columns.

```
public static void zeropadComplexToComplex(final double[] result,
        final double[] array, int sX, int sY, int sZ, int exp_sX,
        int exp_sY, int exp_sZ) {

    if (sX == exp_sX && sY == exp_sY && sZ == exp_sZ)
        System.arraycopy(array, 0, result, 0, array.length);
    else {
        // fill result with zeros
        for (int idx = 0; idx < result.length; idx++) {
            result[idx] = 0;
        }

        // copy array to result
        for (int z = 0; z < sZ; z++) {
            for (int y = 0; y < sY; y++) {
                System.arraycopy(array, 2 * (y * sX + z * sX * sY), result,
                        (2 * y * exp_sX + z * exp_sX * exp_sY), 2 * sX);
            }
        }
    }
}
```

Figure 3.10: Eclipse *Java* editor showing a class file including the results of a test coverage run with the tool *EMMA*. Each line of code is highlighted according to the coverage measurement in red, yellow, and green. The color red marks not covered lines of code, yellow partially covered lines of code, and green fully covered lines of code.

Python have been used. The tool *PyMetrics* is a commandline-based tool that offers the calculation of standard metrics as listed by Balzert et al.[100] The drawback of *PyMetrics* is its usability. It can only be called for a single *Python* module, and the report is similarly to *py.test* printed to the standard output. *Metrics* offers also the measurement of standard metrics as they can be found in the book of Balzert et al.[100] for *Java*. Incontrast to the tool *PyMetrics*, *Metrics* offers an *Eclipse*extension that illustrates the results as shown in Fig.3.11.

Metric	Total	Mean	Maximum	Method
⊞ Total Lines of Code	4168			
⊞ Method Lines of Code (avg/max per method)	2498	7,686	62	exchangeForComp
⊞ Nested Block Depth (avg/max per method)		1,354	4	calculateDistances
⊞ Number of Parameters (avg/max per method)		2,129	9	Topology
⊞ Number of Overridden Methods (avg/max per type)	0	0	0	

Figure 3.11: Results of *Metrics* for the prototype *M^{3}S-Java*. The results are shown in a tabular display. For each metric the total, the mean, and the maximum value are listed if possible. The method with the worst result is listed in the column "Method". If a metric result exceeds the safe range of a metric[100] it is highlighted red.

Here one can see that all metrics except for the numberof parameters are corresponding to Balzert et. al. in the safe range. Only thenumber of parameters is out of the safe range and thus marked red, since the class Topology expects nine parameters in the constructor. This problem could be solved for instance by replacing the directly passed arguments by a struct like data structures that holds the arguments. The static quality measurement shows, that all three prototypes result in a reduction of the total lines of code (TLOC) by a factor of 5-10 compared to *OOMMF*. This reduction can be attributed to the extensive use of libraries.

Support for numerical libraries

MATLAB offers the access to many established numerical *C/C++* and *FORTRAN* libraries. For the development of *M^{3}S-MATLAB* all necessary numerical algorithms, which were the FFT, ODE solvers, general marix operations, linear algebra solver, and sparce matricies were supported. As a commercial tool these libraries are not directly visible for the user. Only an API is provided that allows the access. The choice for a library hence is in the control of *MATLAB*. For instance a SMP based parallel three-dimensional-FFT implementation was not provided since *MATLAB 2010*, as for *MATLAB* this had a minor priority. As exemplary shown in Sec. 3.2.1 as solution other numerical libraries can be interfaced using so-called mex-fuctions. A mex-function is a special *MATLAB* functions with a defined API for the development of interfaces to *C/C++* and *FORTRAN*.

For *Python* as described previously *NumPy* and *SciPy* offer the needed numerical libraries. In principle *NumPy* and *SciPy* follow the same concept as *MATLAB*; a clear API is provided for many established numerical *C/C++* and *FORTRAN* libraries. The difference to *MATLAB* is here twofold. First, the public user license open allows the user to inspect the source code to understand the used algorithms. For scientists this is very important a open source code allows to proof the accuracy of the algorithms and thus increases the confidence in the used libraries.

Java is a young language and its user commuity for combutational sciences is small compared to *C/C++* and *FORTRAN*. A common opinion about *Java* is, that its runtime

performance for numerical operations is about three times slower compared to *C/C++* and *FORTRAN*. This opinion is based on the first concepts applied in the early JREs as explained before. Actual benchmarks proof that the runtime performance of *Java* has caught up with *C/C++* or *FORTRAN*. For instance the *Scimark 2.0* benchmark[179] shows that *Java* has nearly the same performance for numerical tasks as *C*. The remaining difference can be attributed to the existence of fast numerical libraries. Here the *JTransforms* library[180] offers runtime performance only two times slower than *FFTW* and is parallelized. Hence for the most important algorithm, the FFT, a fast numerical library exists. For the general matrix operations libraries like *Colt*[181] or *apache.math* can be used. *Colt* offers a selected number of operations highly optimized. *apache.math* in contrast is less optimized, but offers a extensive selection of numerical operations. In the prototype *M*^{3}*S-Java* these libraries have not been used, as the marix handling between *Colt* and *JTransforms* differs. Using *Colt* would have neccessiate the transformation of different matricies during the simulation. Since the focus of this evaluation was to see, if for *Java/JSA* as CSIDE the same restrictions as for *MATLAB* occur, solving this compatibility problems have been excluded from the evaluation and instead for only *JTransforms* as library have been used. Nevertheless all necessary libraries could be found for *Java*. An overview for the remaining libraries for instance for sparce matrices or linear algebra solvers is given by the Java Numerics Group.[56]

The comparison of the support for numerical libraries reveals, that *Python* offers the best support for numerical libraries. *MATLAB* as commerical software is more restrictive due to the user licence and for *Java* as new programming language for computational sciences less fast numerical libraries exist.

Call by value/reference

Another large difference between *MATLAB* and *Python* is the use of call by value or call by reference. *MATLAB* uses the so called *copy-on-write* strategy. Copy on write means, that a variable passed as an argument to a function is copied, when it is changed within the function. Further accessing a matrix by selecting a subset of indices always results in copying the sub-matrix. Code listing 3.19 shows a typical index operation. The *MATLAB Profiler* reveals that the indexingoperation in line 4 needs twice as long as the multiplication in line 5.

```
1 function selectionTest()
2     a = rand(3000,3000);
3     for i = 1:100
4         b = a(1:500,1:500);
5         c = 2  * b;
6     end
7 end
```

Code listing 3.19: Example code used for the runtime measurement of an index operation compared to a multiplication in *MATLAB*.

In contrast to this behavior, *NumPy* and *SciPy* are offering a more flexibly designed API allowing to reference parts of a matrix. Here the user can choose whether a reference or a copy of the subset of the matrix is desired. A profiling of the *Python* version of Code listing 3.19 shows that the indexing operation needs 100 times less than the multiplication. This flexibility allows to implement all sequential optimizations of the demagnetization field. Only the first optimization resulted in no performance gains, since *NumPy* provides a real one-dimensional FFT, which internally is mapped to a complex one-dimensional FFT.

In *Java* all instructions on non-primitive data types are performed through a call by reference. Call by value has to be implemented explicitly. This means, whether call by value or call by reference is used depends on the chosen library and the scripting engine. The chosen libraries and scripting engine for *M^{3}S-Java* all use call by reference, hence all identified sequential optimizations can be realized in *Java*.

Portability

M^{3}S-MATLAB is portable, as long as only functionality is used that is included in *MATLAB* or is written in *MATLAB-Script*. The portability comes from compiled versions of *MATLAB* for many operating systems provided by MathWorks.[182] As described in Sec. 3.2.1 the portability changes, when *C* or *FORTRAN* programs are interfaced due to performance optimizations. Doing so, the *C* or *FORTRAN* programs need to be compiled on the user's operation system which necessitates to bind a compiler to *MATLAB*. However, *M^{3}S-MATLAB* is more flexible than *OOMMF*, as it offers the non-optimized implementations based on *MATLAB-Script* and the optimized implementation interfacing *C*. *M^{3}S-MATLAB* uses the flexibility of *MATLAB* to estimate if the optimized version can be compiled. If not *M^{3}S-MATLAB* uses the lessoptimized implementation. In this way it is ensured that a running version is always available.

For *Nmag-FD* the same concepts as for *M^{3}S-MATLAB* can be applied. All libraries used for *Nmag-FD* can be found in the main *Python* repository[32] and can be installed using the package installation tool *easy_install*[183] provided for *Python*. Thus the installation of *SciTools* and *IPython* are especially simple because precompiled versions of the *NumPy* and *SciPy* package are offered by the community for many operation systems.

Since the *Java Runtime Engine (JRE)* is supported by the most operating systems *M^{3}S-Java* also runs on the most operation systems and hence is in this comparison the most flexible solution as long as only *Java*-based numerical libraries are used. The simulator can be compiled including the *Java Scripting API* allowing an extension of the simulator without the necessity to deploy the simulator. If the library support of *Java* does not fulfill the users demands, *C/C++* or *FORTRAN* libraries can be interfaced using the *Java Native Interface (JNI)*.[184] But the portability benefits of *Java* disappears when interfacing with *JNI*, because the portability of the prototype then depends on the used *C/C++* or *FORTRAN* libraries.

3.3.2 Runtime performance

An intensively discussed problem is the final runtime performance of scientific software. This topic is addressed by a runtime performance analysis and a comparison between all three prototypes.

Sequential optimization

Many factors have an effect on the runtime performance. In addition to the different algorithmic optimizations the compiler has a large effect on the runtime. Therefore it is necessary to build a basis for the comparison. First a runtime comparison between the official *OOMMF version (1.3.2a14)* further referenced as *OOMMF 2002*, the latest unoffical *OOMMF version (1.4a3 build 20091218)* further referenced as *OOMMF 2009*, *M³S-MATLAB*, *Nmag-FD*, and *M³S-Java* is performed*.

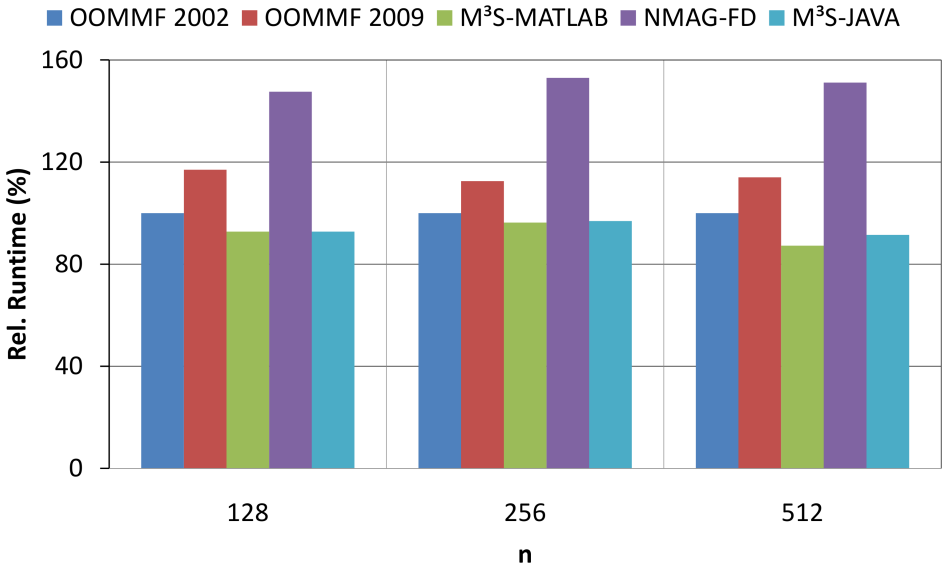


Figure 3.12: Comparison of the runtime of the official *OOMMF* version (*OOMMF 2002*), the latest unofficial *OOMMF* version (*OOMMF 2009*), *M³S-MATLAB*, *Nmag-FD*, and *M³S-Java*. For the comparison the runtime of a simulation loop is measured for a fixed number of evaluations using a micromagnetic problem including the demagnetization field, the exchange field, and the Zeeman field. For the calculation of the demagnetization field the simple algorithm provided by *OOMMF* in the class `Oxs_Simple_Demag` is used. For each tool the runtime of the simulation with a fixed number of evaluations is depicted relative to the runtime of *OOMMF 2002*. The measurements have been performed for systems of size $(n,n,1)$, where $n \in 128, 256, 512$.

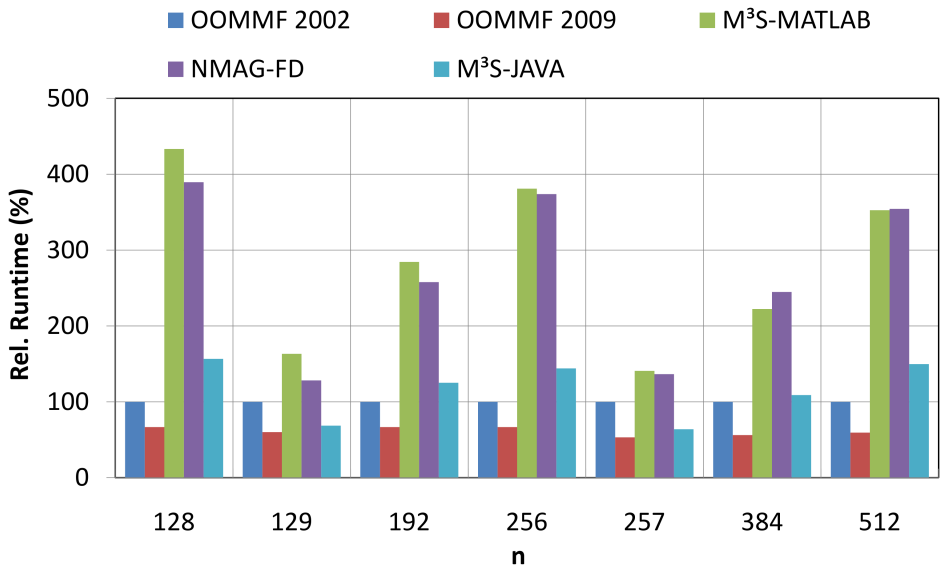
*The comparison has been performed on an Intel Core 2 6700 - 2.67 GHz and 3GB RAM. Due to installation problems with *OOMMF 2002* on Linux, the runtime comparison has been performed on the operation system Windows XP 64-bit SP3.

For the comparison the runtime of the simulation loop is measured for a micromagnetic problem including the demagnetization field, the exchange field, and the Zeeman field. In this way the runtime overhead produced by the ODE is included in the measurement, too. The demagnetization field is calculated by the algorithm implemented in *OOMMF* in the class `Oxs_Simple_Demag` (further called the simple demagnetization field algorithm). This algorithm uses the FFT with the $np2(n)$ zero-padding strategy. Figure 3.12 depicts the relative runtime of all tools in relation to the results of *OOMMF 2002*. The results reveal that:

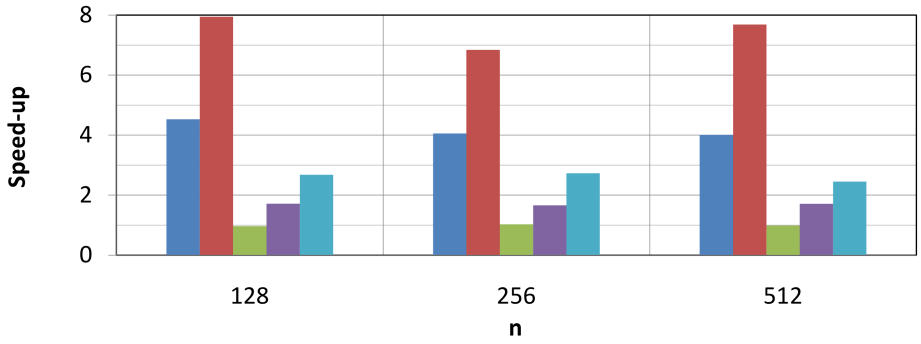
- *OOMMF 2009* is about 15 % slower than *OOMMF 2002*. This can be attributed to the overhead for the parallelization of the calculations included in *OOMMF 2009*.
- *M^{3}S-MATLAB* is about 8 % faster than *OOMMF 2002*. Considering that the one-dimensional-FFT implementation in *MATLAB* is faster than the *OOMMF 2002* implementation, the difference can be explained by the larger runtime of the remaining components, i.e. the LLG, the exchange field, and the ODE solver.
- In this comparison *Nmag-FD* is the slowest implementation; it is about 51 % slower as *OOMMF 2002*. The runtime difference can be attributed to the used FFT libraries. In the used version *SciPy* does not interface to *FFTW*. *FFTW* is only supported since version 0.7.0 and is not provided in the precompiled version for the system *Windows XP*. A runtime measurement on the operation system *Ubuntu*(version 10.04) including *FFTW3* showed a 30 % increase in runtime performance.
- *M^{3}S-Java* is about 7 % faster than *OOMMF 2002*, and thus nearly as fast as *M^{3}S-MATLAB*. Allthough *M^{3}S-Java* uses the slowest FFT library in contrast to *M^{3}S-MATLAB*, it offers nearly the same runtime performance. Hence the computation of the components excluding the demagnetization field is faster in *M^{3}S-MATLAB*.

All three CSIDE choices to implement a micromagnetic simulator resulted in a runtime comparable to the basic algorithms used in *OOMMF 2002*. Only *Nmag-FD* is slower when using the precompiled version of *NumPy* for *Windows XP*. From all three prototypes *M^{3}S-MATLAB* offers the best performance.

The question arises, which of the optimizations identified in Sec. 3.2 can be expressed in the CSIDEs and in which runtime compared to *OOMMF 2002* these optimizations result in. Therefore a second runtime measurement has been performed replacing the simple demagnetization field by the optimized version provided by each tool. The performance comparison depicted in Fig. 3.13 shows the relative runtime of all tools in relation to the results of *OOMMF 2002*. The figure also depicts the speed-up between a simulation including the simple and the optimized demagnetization field algorithm for each tool.



(a)



(b)

Figure 3.13: Runtime comparison of the official *OOMMF* version (*OOMMF 2002*), the latest unofficial *OOMMF* version (*OOMMF 2009*), *M³S-MATLAB*, *Nmag-FD*, and *M³S-Java*. For the comparison the runtime of the simulation loop is measured for a micromagnetic problem including the demagnetization field, the exchange field, and the Zeeman field. For the calculaton of the demagnetization field the best available algorithm provided by each tool is used. (a) depicts the runtime of the simulation for a fixed number of evaluations relatively to *OOMMF 2002* for each tool. The measurements have been performed for systems of size $(n,n,1)$, where $n \in 128, 129, 192, 256, 257, 384, 512$. The sizes have been chosen to demonstrate the effect of the zero-padding strategy. (b) depicts the speed-up estimated by comparing the simulations inculding the simple and including the optimal calculation of the demagnetization field for each tool.

Both results reveal the following conclusions:

- In *OOMMF 2002* the optimal implementation for the calculation of the demagnetization field results in a speed-up of about four compared to the simple demagnetization field calculation.
- *OOMMF 2009* needs only 62 % of the runtime of *OOMMF 2002*. This can be attributed to the additional cache optimizations included in the optimal demagnetization field calculation as mentioned in Sec. 3.2. Since the simple demagnetization field calculation in *OOMMF 2009* was slower than the calculation in *OOMMF 2002* the gained speed-up for *OOMMF 2009* results in an average factor of 7.2.
- For *M^{3}S-MATLAB* non of the optimizations could be efficiently implemented due to its call-by-value semantics. Only the adaptive zero-padding strategy results in a resonable performance gain. The optimized algorithm in *M^{3}S-MATLAB* results on the average in a 271 % slower runtime as *OOMMF 2002* and a negligible speed-up.
- For *Nmag-FD* optimizations one and two for the demagnetization field calculation could be implemented. Allthough the optimized algorithm in *Nmag-FD* results in average in a 255 % slower runtime as *OOMMF 2002* but in a speed-up of 1.7.
- In this comparison *M^{3}S-Java* is the fastest *M^{3}S* prototype and in average only 11 % slower than *OOMMF 2002*. The speed-up is smaller for *M^{3}S-Java* as for *OOMMF 2002* and results in an average factor of 2.7.

Consequently all three CSIDE choices to implement a micromagnetic simulator resulted in a runtime competivie to the basic algorithms used in *OOMMF 2002*. Only *Nmag-FD* is slower using the precompiled version of *NumPy* for *Windows XP*. *M^{3}S-Java* offers the best runtime performance of the three M^{3}S prototypes as it allows to implement all demagnetization field optimizations.

Parallelization

The parallelization of sequential software is as described in Sec. 2.1.1. On the long term parallelization is the only possibility to reduce the runtime of simulations significantly. Hence the support for parallelization techniques like the message passing interface (MPI) or the symmetric multiprocessing (SMP) are important for the choice of a CSIDE. In the following the focus is on SMP since due to the three-dimensonal FFT included in the calculation of the demagnetization field MPI has a too large communication cost.

MATLAB offers as an extension the *Parallel Computing Toolbox*. This toolbox parallelizes the code by starting a pool of *MATLAB* runtime environments as so called workers. Each worker runs in a distinct thread and can be called by the main runtime environment to perform tasks. The parallel execution of a loop is realized by splitting the loop into

independent sections and distributing these sections to the workers. Considering that each worker reserves about 200 MB main memory, the management of the pool takes a large amount of run-time; A performance gain of this solution is only given for long running tasks that need significantly more execution time than the distribution effort. Here several alternative open source and commercial solutions have been published[185,186] and reviewed by Sharma.[85]

While for *Python* different well-suited packages for the distributed calculation exist like *pyMPI*,[187] the parallelization using SMP is as difficult as for *MATLAB*. This is based on the *global interpreter lock (GIL)* that restricts the *Python* interpreter to execute only one command simultaneously. The current version *Python*, 2.6.3, includes the package *multiprocessing* by default. This package offers a similar solution as the *Parallel Distributing Toolbox*, with a less reserved main memory (about 10 MB per worker) in comparison to the case of *MATLAB*. Similar to *MATLAB* also for *Python* different open-source and commercial alternatives exist like the *IPython* project and *Star-P*.[186]

Java supports necessary software development techniques that are well-suited for the development of parallel applications on desktop computers. Here a concurrency package is provided in the newest version of *Java* by default offering standard solutions for the execution of parallel and concurrent programs. This allowed to implement the SMP based parallelization for the calculation of the demagnetization field in *M³S-Java* conveniently resulting in a reasonable speed-up as shown in Fig. 3.14.

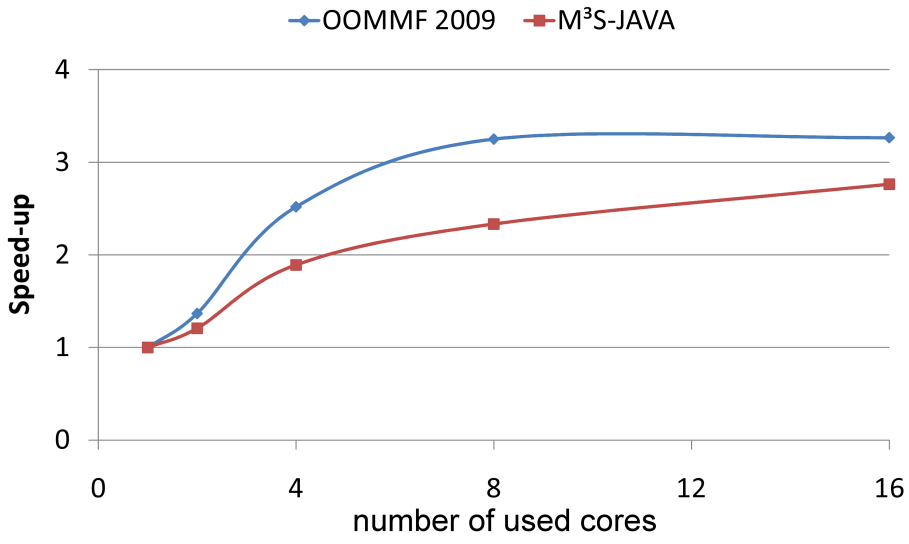


Figure 3.14: Parallelization speed-up of the simulations of a micromagnetic problem of size (256,256,4) grid points. The simulations have been performed with *OOMMF 2009* and *M³S-Java* on an Opteron with 16 cores and 128 GB RAM. The x-axis shows the number of cores enabled for the simulation run, while the y-axis shows the gained speed-up of the simulation.

Nevertheless, the default concurrency package of *Java* is restricted for the parallelization of numerical calculations as described by Taboada et al.[188] They further identify better solutions for *Java*.

3.3.3 Results of te evaluation of different CSIDEs

The evaluation of two other common CSIDEs revealed that the choice of *Python/SciTools* and *Java/JSA* are more promising than using *MATLAB*. A final decision for *Nmag-FD* or *M^{3}S-Java* could not be made with the prototypes at hand. *M^{3}S-Java* offers the better runtime performance, the better portability, and the better support for software quality measurements. Since *Java* is new in the numerical computing community, the support for numerical libraries is less extensive as for *MATLAB* or *Python*. *Nmag-FD* in contrast offers the support for *C/C++* and *FORTRAN* libraries and hence the better support for numerical libraries. Moreover, it offers with *NumPy* and *SciPy* a powerful scripting support for these numerical libraries that allowed to develop the whole software in a script language.

This evaluation revealed that the restrictions identified for *MATLAB* are mainly *MATLAB* specific. While other CSIDEs support all concepts that are necessary to develop large programs, the restrictions due to the runtime depend on the flexibility of the supported libraries and their integration into the scripting language.

Nevertheless this evaluation shows that the common approach to use CSIDEs to prototype scientific software and to reimplement the prototypes later in *C/C++* or *FORTAN* is not necessary. Instead if no proper library exists, only the runtime critical components need to be exported. In any case the simulator still remains portable as the unoptimized version of the components are always avaliable. Concerning *make or buy*, a scientist can start to develop a prototype using a CSIDE that bit by bit is extended to a complete simulator. In this approach the developer uses first the provided libraries to develop the algorithm. If the resulting component is not efficient enough, the developer can spend time in the optimization of the algorithm and use the library-based implementation as reference for tests.

The parallelization of the sequential algorithms could only be evaluated supersticially in this work. Here further analysis are necessary to substanciate the evaluation due to the parallelization possibilities.

Chapter 4

Current dependency

The last chapter presented the micromagnetic simulator *M*^{3}*S-MATLAB* and evaluated two alternative CSIDEs for the implementation of *M*^{3}*S*. In the first step a reverse engineering of the numerical model implemented in *OOMMF* has been performed. The resulting simulator offers a much better balance between the maintainability and usability compared to *OOMMF*, while the runtime is about two times slower.

This chapter uses these features to extend *M*^{3}*S-MATLAB* by the physical phenomena that occurs when a current flows through a ferromagnetic system. As motivated in Sec. 1 this topic has become essential in the focus of the research community as it promises novel storage concepts. Here the problem arises that the optimization of the properties of the nanostructured ferromagnets accompanies the understanding of the physical phenomena. Since the micromagnetic simulation has become an important method in the fundamental research of ferromagnetic nanostructures, it is important to extend a simulator by the known phenomena and to support their extension by new discoveries.

The following aspects concerning the current dependency have been addressed by this work and will be discussed further:

At first the prototype *M*^{3}*S-MATLAB* has been extended by the spin-transfer torque in continuously variable magnetization patterns and in spin valves (as introduced in Sec. 2.2.4). A detailed discussion of the development of both modules is given in the article entitled "The micromagnetic modeling and simulation kit M^{3}S for the simulation of the dynamic response of ferromagnets to electric currents", which was presented at the 2008 Grand Challenges in Modeling and Simulation Conference GCMS'08 (that took place between 16 and 19 June 2008 in Edinburgh, UK) reprinted in Sec. 4.1. This article emphasizes the simplicity that is offered by the *M*^{3}*S-MATLAB* and its modular architecture. It also presents the results for two system tests to verify the modules; these tests are based on results of Krüger et al.[189] as well as Berkov and Gorn.[43]

The extension of *M*^{3}*S-MATLAB* by a module for the spin-transfer torque in continuously variable magnetization patterns revealed the question, in which range the simulation of current-driven vortices or domain wall dynamics it is valid to approximate the current paths as homogeneous. To investigate this aspect a *MATLAB* module for the static calculation of the current paths and the AMR-effect have been interfaced to *M*^{3}*S-MATLAB* in cooperation with Stellan Bohlens. This cooperation resulted in a micromagnetic simulator that allows to investigate the mutual interplay. This aspect is discussed in detail in manuscript 1 in Ch. 6. The manuscript discusses the accuracy of the implementation for the simulation of current-driven vortexdynamics.

Studying existing tools[118,121,157] revealed, that many existing simulators have not been extended by the spin-transfer torque in continuously variable magnetization patterns. Considering the increase in importance of this phenomena in the last years, these simulators will likely be extended in the near future, too. A review of the system test for the spin-transfer torque in continuously variable magnetization pattern that was used in publication 4.1 showed that the test is suitable for the validation but not for the falsification of the module. A proposal for a new standard problem that allows for the falsification of the module has been developed. Details of the proposed problem have been discussed in the article "Proposal for a Standard Problem for Micromagnetic Simulations Including Spin-Transfer Torque", which has been published in the Journal of Applied Physics in 2009 and that is reprinted in Sec. 4.2. The article describes how the proposed problem is defined by applying selection criteria, which are in accordance to the quality criteria suggested on the μMag webpage.[40] A final comparison of the simulation results of different extended micromagnetic simulators illustrates the adequat properties of the problem.

During the development of the proposed standard problem, the question arose, which values were experimentally realistic for the degree of non-adiabaticity. A literature research revealed that the theoretically predicated as well as the experimentally measured values differ by one order of magnitude.[190–193] This circumstance could be explained by the small accuracy of existing measurement techniques. As the exact value of the degree of non-adiabaticity has a large influence on the current-driven dynamic of magnetic vortices and domain walls, in cooperation with Benjamin Krüger a robust measurement scheme has been suggested. Details of the proposed measurement scheme are discussed in detail in the article "Proposal of a Robust Measurement Scheme for the Nonadiabatic Spin Torque Using the Displacement of Magnetic Vortices", which has been published in Physical Review Letters in 2010 and that is reprinted in Sec. 4.3. The article illustrates the results of the measurement scheme by comparable simulations. As the simulations take into account typical perturbations like a Oersted field or the AMR effect, they substantiate the unique accuracy of the proposed measurement scheme.

4.1 Publication GCMS'08

The Micromagnetic Modeling and Simulation Kit M^3S For the Simulation of the Dynamic Response of Ferromagnets to Electric Currents

M. Najafi, B. Krüger, S. Bohlens, G. Selke, B. Güde, M. Bolte, and D. P. F. Möller

Proceedings of the 2008 Grand Challenges in Modeling and Simulation Conference (GCSM'08), H. Vakilzadian, R. Huntsinger, T. Ericson, and R. Crosbie, Eds. San Diego, CA, USA: The Society for Modeling and Simulation, 2008, pp. 427–434

The micromagnetic modeling and simulation kit M^{3}S for the simulation of the dynamic response of ferromagnets to electric currents

Massoud Najafi [a,b*], Benjamin Krüger [c], Stellan Bohlens [c], Gunnar Selke [a], Bernd Güde [a,b], Markus Bolte [a,b],and Dietmar P. F. Möller [a]

[*] mailto://mnajafi@physnet.uni-hamburg.de,

[a] Arbeitsbereich Technische Informatik Systeme, Department Informatik, Universität Hamburg, Vogt-Kölln-Straße 30, 22527 Hamburg, Germany,

[b] Institut für Angewandte Physik und Zentrum für Mikrostrukturforschung, Universität Hamburg, Jungiusstraße 11, 20355 Hamburg, Germany,

[c] I. Institut für Theoretische Physik, Universität Hamburg, Jungiusstraße. 9, 20355 Hamburg, Germany

Keywords: simulation of physical phenomena, micromagnetic modelling and simulation, spin valves, spin-transfer torque

Abstract

Micro- and nanostructured ferromagnetic materials are actively studied as they offer a variety of applications for microelectronics, hard disks and main memory devices. The widely accepted standard model to describe ferromagnetic systems in this regime is the micromagnetic model [1]. Recently, the interaction of electric currents with the local magnetization in a ferromagnet by transfer of spin momentum have become a focus in academic and industrial research. Hence it has become necessary to extend the micromagnetic model by current-dependent terms, known as the spin-transfer torque extensions. This work presents the micromagnetic modeling and simulation kit M^{3}S, which implements the basic micromagnetic model as well as the spin-transfer torque extensions for multilayer systems based on Slonczewski [2, 3] and the spin-transfer torque extension for continuously varying magnetization based on Zhang and Li [4]. The architecture of the M^{3}S is discussed and the validity of the implementation is proven by several test problems.

1. INTRODUCTION

The micromagnetic model [1] describes the magnetization dynamics by a time-dependent non-linear partial differential equation, the so-called Landau-Lifshitz-Gilbert equation (LLG) and includes the spatial interaction by different magnetic field terms [5]. In the beginning the micromagnetic model was used for analytical calculations of the widths of magnetic domain walls or the switching field in very small ferromagnetic particles. In recent years, through the rise of powerful computers, micromagnetic modeling and simulation have evolved into an important method for investigations in this field of research, because they enable the prediction and interpretation of the dynamic behavior of existing and virtual ferromagnetic systems. They also constitute a major factor in gaining a deeper understanding of the fundamental physical principles. Even more recently, the interaction of electric currents with the local magnetization in a ferromagnet have become a focus in this field of research. One example is discovery of the giant magnetoresistance effect [6, 7] for which P. Grünberg and A. Fert were awarded the Nobel Prize. New physical phenomena were integrated into the micromagnetic model by adding current-dependent torque terms, known as the spin-transfer torque terms, into the LLG equation [2, 4, 8]. Nowadays, two different current-dependent extensions of the LLG equation exist: The first, developed by Slonczewski [2], accurately describes currents traversing through interfaces between ferromagnets and non-magnets and the ensuing torque on the magnetization. The second was developed by Berger [8] and has since been extended by Zhang and Li [4] and Thiaville et al. [9]. It deals with the spin-transfer torque due to continuous changes in the magnetization, e.g., due to domain walls or magnetic vortices.

This work will present the micromagnetic modeling and simulation kit M^{3}S as an advancement of a micromagnetic simulation tool prototype presented at the Summer Computer Simulation Conference (SCSC) in San Diego in 2007[10]. It implements both versions of the spin-transfer torque term. The outline of this work is as follows: Section 2 describes the micromagnetic model and both spin-transfer torque extensions. Section 3 then presents M^{3}S with the spin-transfer torque module and discuss the benefits of its architecture. Section 4 validates M^{3}S by comparing the results of well defined structures with analytical and experimental results.

2. THEORETICAL BACKGROUND

In this section the micromagnetic model, which is the appropriate model to describe ferromagnets on the nano- and micrometer scale, as well as the spin-transfer torque extensions are described in more detail.

2.1. Micromagnetic Model

The micromagnetic model correctly predicts the static structure of nano- and micrometer-sized ferromagnets as well as the dynamics up to the THz-regime. In 1932, Landau and Lifshitz [5] laid the foundation to this theory, with major contributions coming later from Gilbert, Néel, Bloch, Brown, and many others [1, 11, 12]. Several excellent reviews and books describe this theory in great detail [13, 14, 15]. In this model the ferromagnet's magnetization wants to align itself to the magnetic fields that are present in each point of the volume. In turn the magnetization determines the effective magnetic field by a superposition of internal and external magnetic fields. The internal fields are caused by different magnetic interactions such as the quantummechanical exchange between neighboring spins or the magnetostatic interaction. The interaction between magnetization and effective field leads to a complex dynamic behavior. Except for some analytically feasible systems, the magnetization dynamics can only be solved numerically.

2.1.1. Equation of Motion

The LLG equation is the fundamental equation in the micromagnetic model and describes the motion of the magnetization. The magnetization $\vec{M}$ precesses around the local effective magnetic field $\vec{H}_{\text{eff}}$ and is damped towards its equilibrium direction, which is parallel to the effective field as shown in Fig. 1. It is described by the two terms on the right-hand side of Eqn. (1):

$$\frac{d\vec{M}}{dt} = -\gamma\vec{M} \times \vec{H}_{\text{eff}} + \frac{\alpha}{M_s}\vec{M} \times \frac{d\vec{M}}{dt}, \qquad (1)$$

Here M_s is the saturation magnetization, $\gamma = 2.21 \cdot 10^5$ m/C is the absolute value of the gyromagnetic ratio, and $\alpha > 0$ is the Gilbert damping constant.

2.1.2. Effective Field

The micromagnetic model includes all magnetic interactions as magnetic fields interacting with the local magnetic moments. The basic model includes the magnetostatic field, the exchange field, the anisotropy field, and the Zeeman field.

The magnetostatic field describes the magnetic interactions of the local magnetic moments over long distances within the body and favors the magnetization to be aligned to the surface. A magnetization perpendicular to a surface would lead to surface charges akin to electrical charges in a capacity and thus greatly increase the system's energy. The exchange field describes the interaction between the spins of neighboring atoms. In ferromagnets, the exchange interaction tends to align neighbor spins parallel to each other. The interplay between the exchange and magnetostatic interaction leads to the formation of magnetic domains in the ferromagnet. A domain is a region within the ferromagnet in which the magnetization is fully aligned. The boundaries of two domains in which the magnetization rotates from the direction in one domain to the direction in the other domain are called domain walls. The anisotropy field describes anisotropic effects that arise due to the structure of the lattice and to the particular symmetries that are present in certain crystals. It leads the ferromagnet to magnetize along specific directions, which in literature are referred to as easy axes. The Zeeman field is the field from an external magnet. The local summation of all these field types constitute the local effective field.

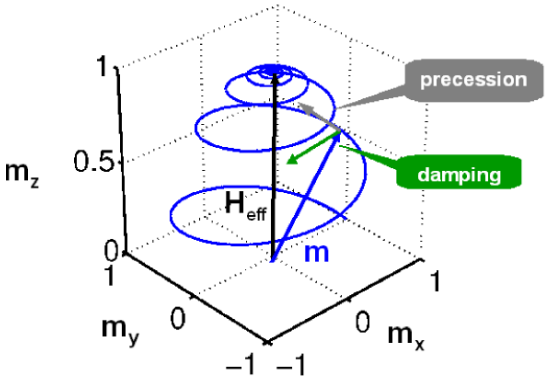


Figure 1. Trajectory of the magnetization due to an effective field. The magnetization performs a damped precession around the effective field.

2.2. Spin-transfer Torque for Media with Continuously Varying Magnetization

In addition to the standard micromagnetic model, an extension for the interaction of itinerant, i.e., moving electrons and the local magnetization in volumes with continuously changing magnetization have been introduced[4, 8]. It correctly describes magnetization dynamics within a ferromagnet with continuously varying magnetization as shown in Fig. 2, that is excited by a spin-polarized current. The additional torque, called *spin-transfer torque* for such a system arises from the interaction of the spin-polarized current with the local magnetic moments within the ferromagnet. The itinerant electrons align their spin with the spins of the local electrons that constitute the magnetization. This torque on the moving electrons must be compensated by an opposite torque on the local magnetization to conserve the total momentum. The extended LLG with two extra spin-transfer torque terms is[4, 16, 17]

$$\frac{d\vec{M}}{dt} = -\gamma\vec{M}\times\vec{H}_{\text{eff}} + \frac{\alpha}{M_s}\vec{M}\times\frac{d\vec{M}}{dt} - \frac{b_j}{M_s^2}\vec{M}\times\left(\vec{M}\times(\vec{j}\cdot\vec{\nabla})\vec{M}\right) - \xi\frac{b_j}{M_s}\vec{M}\times(\vec{j}\cdot\vec{\nabla})\vec{M}, \quad (2)$$

with the coupling constant $b_j = (P\mu_B)/(eM_s(1+\xi^2))$ between the current $\vec{j}$ and the magnetization $\vec{M}$, where μ_B is the Bohr magneton, e is the elemetary charge, $\xi = \tau_{\text{ex}}/\tau_{\text{sf}}$ is the degree of non-adiabacity, and P denotes the spin polarization of the current. Equation (2) can be written in explicit form

$$\frac{d\vec{M}}{dt} = -\gamma'\vec{M}\times\vec{H}_{\text{eff}} - \frac{\alpha\gamma'}{M_s}\vec{M}\times\left(\vec{M}\times\vec{H}_{\text{eff}}\right) - \frac{b_j'}{M_s^2}(1+\alpha\xi)\vec{M}\times\left(\vec{M}\times(\vec{j}\cdot\vec{\nabla})\vec{M}\right) - \frac{b_j'}{M_s}(\xi-\alpha)\vec{M}\times(\vec{j}\cdot\vec{\nabla})\vec{M}, \quad (3)$$

with the abbreviations $\gamma' = \gamma/(1+\alpha^2)$ and $b_j' = b_j/(1+\alpha^2)$ as shown by Krüger et al.[18].

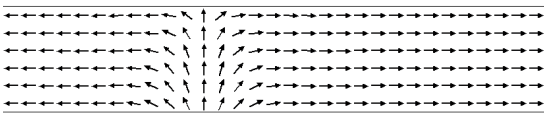


Figure 2. An example for a system with continuous varying magnetization that exhibit the spin-transfer torque effect. In the magnetic wire the magnetization changes continuously from the left to the right. The current flows along the wire direction and interacts with the spatially variation of the magnetization which leads to a motion and distortion of the domain wall.

2.3. Spin-transfer Torque in a Spin Valve

In magnetic multilayers the magnetization changes abruptly at the interfaces between the magnetic layers. The approximation made in the spin-transfer torque model for continuous media cannot be applied for these geometries. In the following section, the spin-transfer torque extension for the description of a spin valve with currents flowing perpendicular-to-plane (CPP) is introduced. A spin valve is a multilayer system, consisting of basically two ferromagnetic layers that are connected by a nonmagnetic metallic spacer as shown in Fig. 3.

In contrast to continuously varying magnetization, the spin-transfer torque in such a spin valve originates from the

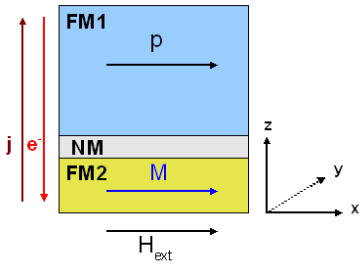


Figure 3. Simple sketch of a spin valve. The electrons flows in -z-direction and crosses first the fixed ferromagnetic layer FM1. FM1 polarizes the current in the direction of his magnetization $\vec{p}$. The spin-polarized current influences the second ferromagnetic layer FM2 via the spin-transfer torque

interaction of the spin-polarized current with the local magnetic moments at the interface between the ferromagnets and the spacer. The ferromagnetic layer FM1, called the fixed layer, is designed to be unaffected by the spin-transfer torque. In reality, this is achieved by exchange-coupling of FM1 to additional layers, e.g., antiferromagnets. FM1 then serves as a source for the spin-polarized current. All electrons passing through this layer becomes polarized equal to its magnetization direction $\vec{p}$. The dynamics of the other ferromagnetic layer, called free layer FM2, due to the spin-transfer torque is given by [2, 3, 17, 19]

$$\frac{d\vec{M}}{dt} = -\gamma\vec{M}\times\vec{H}_{\text{eff}} - \frac{\gamma a_j}{M_s}\vec{M}\times\left(\vec{M}\times\vec{p}\right) + \frac{\alpha}{M_s}\vec{M}\times\frac{d\vec{M}}{dt}. \quad (4)$$

Here $a_j = M_s\beta g(\theta)$ is the coupling constant between the current and the magnetization, with the angle θ between $\vec{M}$ and $\vec{p}$, $\beta = \hbar j/(\mu_0 M_S de)$ and $g(\theta) = \Lambda P/[2((\Lambda^2+1)+(\Lambda^2-1)cos\theta)]$. In these equations $\hbar$ is Planck's constant, μ_0 is the permeability of the vacuum. $\Lambda = G\cdot R$, the product of conductance and resistance, differs from unity if the layers have different thicknesses, P is the spin polarization of the current, and d is the thickness of the free layer, [3, 19, 20]. Employing the same abbreviations as in 2.3., equation (4) can be written in its explicit form

$$\frac{d\vec{M}}{dt} = -\gamma\vec{M}\times\vec{H}_{\text{eff}} - \frac{\gamma\alpha}{M_s}\vec{M}\times\left(\vec{M}\times\vec{H}_{\text{eff}}\right) - \frac{\gamma a_j}{M_s}\vec{M}\times\left(\vec{M}\times\vec{p}\right) + \gamma\alpha a_j\vec{M}\times\vec{p}. \quad (5)$$

3. M^{3}S

M^{3}S is a framework for the simulation of micromagnetic problems. It is the advanced version of the prototype of

the micromagnetic simulation tool presented at the Summer Computer Simulation Conference (SCSC) in San Diego in 2007[10]. From the developer's point of view, the purpose of the development of M^{3}S is to create a micromagnetic simulator with a high software quality [21], with a focus on the key attributes high modularity, easy testability, simple extensibility, and high efficiency. Since high modularity and high efficiency are in many cases opposing attributes, every development process must weigh up the possible solutions with respect to these attributes. The easy testability and simple extensibility directly correspond to modularity, because tests need the possibility to check components with manageable complexity. The actual way to deal with this decision is to follow the three steps in the advice of Kent Beck to *Make It Work, Make It Right, Make It Fast*[22].

In addition to the benefits mentioned in the previous publication [10], MATLAB offers a script language [23], providing a notation similar to the mathematic notation. It also provides simple interfaces to lower-level programming languages such as C,C++, or Fortran. For scientific applications the mathematical notation facilitates the first two steps and allows physicists with a moderate knowledge of MATLAB to quickly write code and to create automated tests for the code. An expert in MATLAB now can implement the third step, without the need to know the physics. This approach has proven invaluable in the development of the present framework for which programmers with backgrounds in computer science and as well as physics could contribute according to their area of expertise.

3.1. Basic Architecture

The core of M^{3}S consists of the configuration object, the solver and the integrator. To start a simulation, a configuration object must be filled with the specific problem definition. The configuration object is at this stage of the simulation responsible for the validation of the user inputs. Then the solver is called passing the configuration to it. If the configuration is consistent, it initializes all needed components, e.g., any included fields or the load-and-store functionality. Next the solver starts the time integration loop by calling the time integrator. The time integrator itself uses the function "calculateModel" for the calculation of the time derivative of the magnetization $d\vec{M}(t_i)/dt$ for a time t_i, which is needed to compute the magnetization at the time t_{i+1} via the LLG-equation. Figure 4 shows the main components of M^{3}S as well as the flow chart of a simulation run.

3.2. Spin-transfer Torque Module

As mentioned above, the action of a spin-polarized current on a ferromagnet is still under discussion. The spin-transfer torque for continuously varying magnetization and the spin-transfer torque for a spin valve are currently the accepted

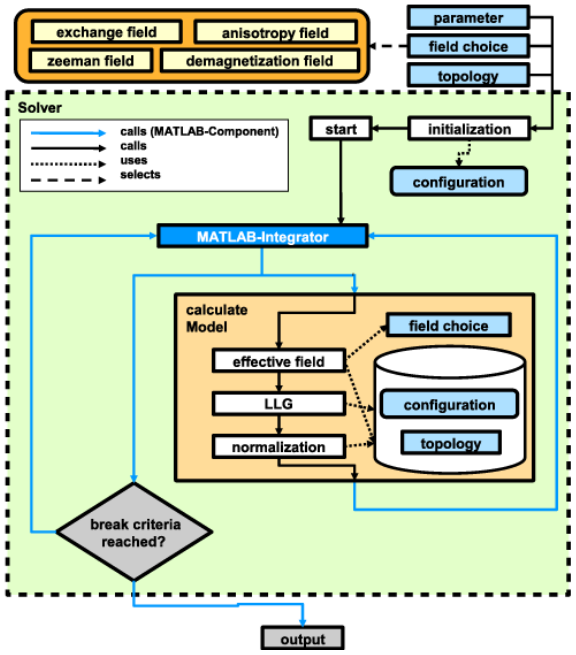


Figure 4. The architecture of M3S showing the interaction of the basic components within a simulation run.

physical descriptions for the respective problem domains. A general description of the spin-transfer torque of continuously and non continuously changing magnetization is still under investigation. Due to these circumstances, it is important to consider the architecture to be flexible for future extensions, without implementing functionality on stock.

The proposed architecture of the module consists of an interface (as shown in Fig.5), which is integrated into the LLG, and the two concrete realizations of spin-transfer torque extensions. To integrate a new spin-transfer torque extension into this architecture, the concrete realization must be implemented. It is important, that it is conformal to the interface. The new extension can then be chosen through the configuration.

4. VALIDATION

Testing the correctness of the simulation results is at least as important as ensuring a good architecture. Therefore, it is important to test individual parts of the simulation, e.g. the field computation or solving the LLG, by unit tests [22] as well as to validate the complete simulation code by integration tests. For the implementation of integration tests the initial parameters and the results of complex micromagnetic reference problems are needed. The μMag group[24] has collected such problems, known as standard problems.

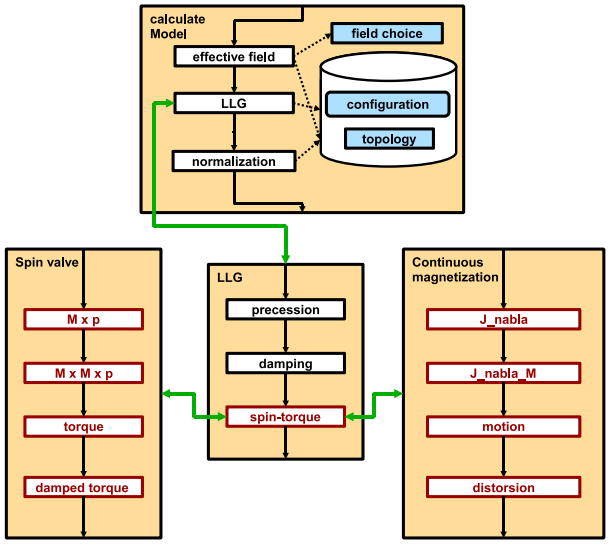


Figure 5. For the simulation, one of the spin-transfer torque extensions can be selected. The selected extension is called from the LLG through a general interface.

The standard problem #4 was used in the previous work to show the correct implementation of the basic micromagnetic model within the prototype [10], which is the basis of M^{3}S. In order to validate the correctness of M^{3}S the additional spin-transfer torque modules need to be validated. Since the spin-transfer torque is a new field of research, there are no standard problems, and the model itself is still a matter of active research and discussion. So this work uses approved analytical and computational results as basis of integration tests.

4.1. Spin-transfer Torque for Continuously Varying Magnetization

To confirm the spin-transfer torque module for spin-transfer torque in continuously varying magnetization, the test configuration as shown in Fig. 6 is used. The sample is a ferromagnetic square with a vortex core in the center. The magnetization is excited by a spin-polarized alternating current.

This structure is well suited for the validation, because it has already been investigated in detail [25, 26, 27, 28] and because there exists an analytical description of the magnetization dynamics[25]. The analytical model describes the vortex core dynamics due to a spin-polarized alternating current or an alternating magnetic field. The selected test configuration is a ferrromagnetic square with a sample size of $100\times100\times10$ nm^3. For the ferromagnetic material parameters, the values for permalloy were chosen, i.e., an exchange constant $A = 13\cdot10^{-12}$ J/m, a saturation magnetization $M_s = 8\cdot10^5$ A/m, a damping constant $\alpha = 0.1$, a degree of non-adiabaticity $\xi = 0.05$, and a gyromagnetic ratio $\gamma = 2.211\cdot10^5$ m/C. The effective field is given by the exchange and magnetostatic field. In addition to the effective field, a spatially homogeneous spin-polarized alternating current of $jP = \cos(\omega t)\cdot 2\cdot10^{11}$ A/m^2 with the frequency of ω = 4.4 GHz is applied in x-direction. The analytical model predicts that the vortex-core excited by such a current starts to gyrate around the center of the ferromagnetic square. Figure 7 shows the results from the simulation with M^{3}S and the analytical model. As can be seen the resulting trajectory fits excellently to the trajectory of the analytical calculation. This shows the validity of this part of the spin-transfer torque module.

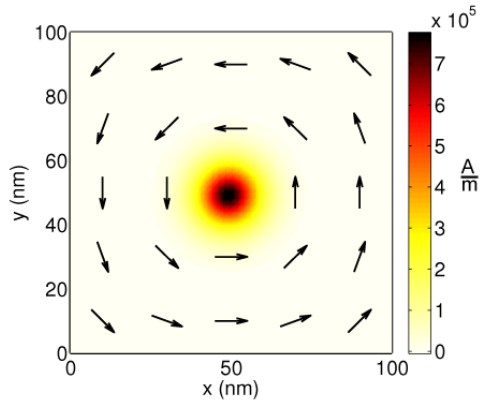


Figure 6. The initial magnetization pattern, a vortex, for the test configuration in this section. The color coding represents the out-of-plane magnetization component.

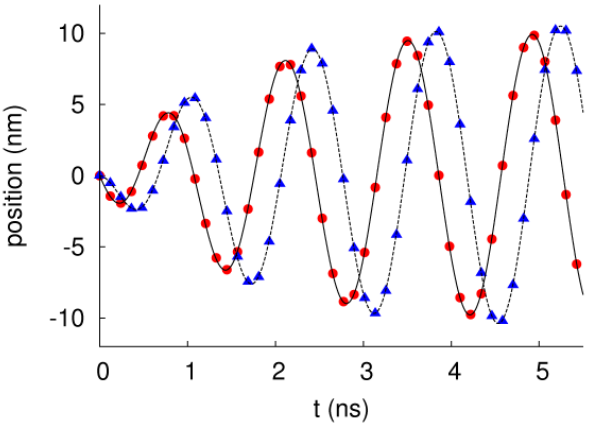


Figure 7. Calculated positions of a vortex that is excited with an alternating current versus simulation time. The circles and triangles denote the x- and y-positions of the vortex, respectively. The lines are fits with the analytical results of Krüger et al. [25].

4.2. Spin-transfer Torque in a Spin Valve

The correctness of the spin-transfer torque module for simulations of the spin-transfer torque in a spin valve is validated by using a rectangular spin valve consisting of two layers of cobalt connected by a copper spacer as a test configuration. This structure was chosen as it has previously been investigated by Li et al. [20] with a edge length b = 64nm and by Berkov et al. [29] for edge lengths between 16 and 120 nm. As integration test for this module, $b = 20$ nm and $b = 48$ nm was chosen, because the simulation results for these edge lengths lead to a distinct trajectory of the magnetization. The simulation parameters of the free layer are a saturation magnetization $M_s = 1.2 \cdot 10^7/(4 \cdot \pi)$ A/m, a damping constant $\alpha = 0.03$, a spin-transfer torque coupling constant $a_j = -4 \cdot 10^5/4 \cdot \pi$ and a gyromagnetic ratio $\gamma = 2.211 \cdot 10^5$ m/C.

The current flows in negative z-direction through the spin valve. The external field, the easy axis of the crystal anisotropy and the current polarization are aligned in x-direction as shown in Fig. 8. The effective field is given by the exchange field with the exchange constant of $A = 2 \cdot 10^{-11}$ J/m, the magnetostatic field, the uniaxial anisotropy field with $H_k = 5 \cdot 10^5/(4 \cdot \pi)$ A/m , and the uniform Zeeman field with $H_{ext} = 1.75 \cdot 10^6/(4 \cdot \pi)$ A/m.

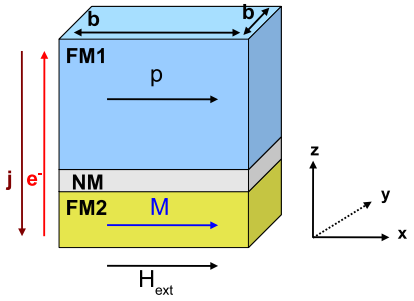


Figure 8. Scheme of the test system, which was used for the validation of the spin-transfer torque. For the validation this system was investigated with different edge lengths b.

At the beginning of the simulation the magnetization is aligned in y-direction. All simulations were computed according to the parameters given by Berkov et al. [29] with a cell size of $2 \times 2 \times 2.5$ nm^3 in (x,y,z)- direction. Figure 9 shows the results for $b = 20$ nm. The time resolved magnetization component m_x as well as the trajectory of the magnetization match well with the results of Berkov et al. Figure 10 shows the results for $b = 48$ nm. The results of this problem differ in the time resolved magnetization component, but the the trajectory of the magnetization match well with the results of Berkov et al. Since in their publication the magnetostatic field causes the difference between $b = 20$ nm and $b = 48$ nm, this error can be explained by the difference in the computation of the magnetostatic field by Berkov et al. In comparison to the results of standard problem #4, this difference has a magnitude of about 2%. We conclude that the spin-transfer torque modules are also valid for the simulation of spin valve systems.

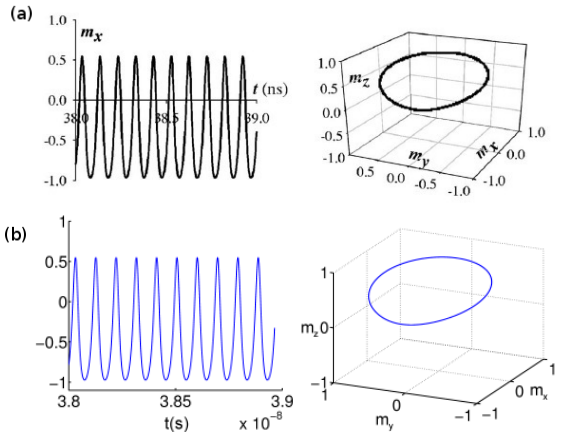


Figure 9. The simulation results for $b = 20$ nm. a.) shows the results from Berkov et al.[29], b.) shows the results of this work.

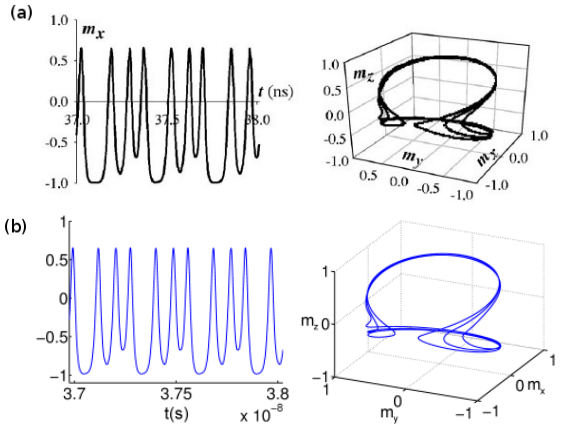


Figure 10. The simulation results for $b = 48$ nm. a.) shows the results from Berkov et al. [29], b.) shows the results of this work.

5. SUMMARY AND OUTLOOK

We have presented the new micromagnetic modelling and simulation kit M^{3}S with the spin-transfer torque module. The correctness of the implemented physic was proved by integration tests based on significant problem definitions. The main goal of this implementation is to ensure a high software quality and so to simplify future extensions. Future tasks will be

the expansion of our tool in view of multi threading and parallelization using the Message Passing Interface (MPI). This is necessary to achieve reasonable computation time for tasks such as the simulation of whole ferromagnetic wire or an array of ferromagnetic nano-particles.

6. ACKNOWLEDGMENTS

Financial support by the Deutsche Forschungsgemeinschaft via SFB 668 "Magnetismus vom Einzelatom zur Nanostruktur" and via Graduiertenkolleg 1286 "Functional metal-semiconductor hybrid systems" is gratefully acknowledged.

REFERENCES

[1] W. F. Brown Jr. *Micromagnetics*. Interscience Publishers, New York, NY, 1963.

[2] J.C. Slonczewski. Current-driven excitation of magnetic multilayers. *J. Mag. Mag. Mat.*, 159:1–7, 1996.

[3] J.C. Slonczewski. Currents and torques in metallic magnetic multilayers. *J. Mag. Mag. Mat.*, 247:324–338, 2002.

[4] S. Zhang and Z. Li. Roles of nonequilibrium conduction electrons on the magnetization dynamics of ferromagnets. *Phys. Rev. Lett.*, 93:127204, 2004.

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

[6] M. N. Baibich, A. Fert Broto, J. M., and F. Nguyen Van Dau. Giant magnetoresistance of (001)fe/(001)cr magnetic superlattices. *Phys. Rev. Lett.*, 61:2472 2475, 1988.

[7] 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, 1989.

[8] L. Berger. Emission of spin waves by a magnetic multilayer traversed by a current. *Phys. Rev. B*, 54:9353–9358, 1996.

[9] A. Thiaville, Y. Nakatani, J. Miltat, and Y. Suzuki. Micromagnetic understanding of current-driven domain wall motion in patterned nanowires. *Europhys. Lett.*, 69: 990, 2005.

[10] M.-A. B. W. Bolte and M. Najafi. Simulating magnetic storage elements: Implementation of the micromagnetic model into matlab - case study for standardizing simulation environments, 2007. SCSC 07:Proceedings of the 2007 Summer Computer Simulation Conference, 525.

[11] F. Bloch. Zur Theorie des Austauschproblems und der Remanenzerscheinung der Ferromagnetika. *Zeitschrift für Physik A Hadrons and Nuclei*, 74:295–335, 1932.

[12] L. Néel. Some theoretical aspects of rock-magnetism. *C. R. Acad. Sci.*, 241:533, 1955.

[13] A. Aharoni. *Introduction to the Theory of Ferromagnetism*. Oxford University Press,Oxford, Clarendon, 1963.

[14] A. Hubert and R. Schäfer. *Magnetic Domains: The Analysis of Magnetic Microstructures*. Springer, Berlin, Germany, 1998.

[15] H. Kronmüller and M. Fähnle. *Micromagnetism and the Microstructure of Ferromagnetic Solids*. Oxford University Press, Oxford, UK, 1963.

[16] S. Zhang and Z. Li. Spin-transfer torque for continuously variable magnetization. *Phys. Rev. Lett.*, 73: 054428, 2006.

[17] D.V. Berkov and J. Miltat. Spin-torque driven magnetization dynamics: Micromagnetic modeling. *J. Mag. Mag. Mat.*, 320:1238–1259, 2008.

[18] B. Krüger, D. Pfannkuche, M. Bolte, G. Meier, and U. Merkt. Current-driven domain-wall dynamics in curved ferromagnetic nanowires. *Phys. Rev. B*, 75: 054421, 2007.

[19] J. Xiao, A. Zangwill, and M. D. Stiles. Boltzmann test of slonczewski's theory of spin-transfer torque. *Phys. Rev. B*, 70:172405, 2004.

[20] Z. Li and S. Zhang. Magnetization dynamics with a spin-transfer torque. *Phys. Rev. B*, 68:024404, 2003.

[21] B. W. Boehm, J. R. Brown, and M. Lipow. Quantitative evaluation of software quality. *ICSE '76: Proceedings of the 2nd international conference on Software engineering*, pages 592–605, 1976.

[22] K. Beck. *Test Driven Development: By Example*. Addison-Wesley, 2003.

[23] http://www.mathworks.co.uk/products/matlab/,2008.

[24] http://www.ctcms.nist.gov/ rdm/mumag.org.html,2008.

[25] B. Krüger, A. Drews, M. Bolte, U. Merkt, D. Pfannkuche, and G. Meier. Vortices as harmonic oscillators. *Phys. Rev. B*, 76:224426, 2007.

[26] V. Novosad, F. Y. Fradin, P. E. Roy, K. S. Buchanan, K. Yu. Guslienko, and S. D. Bader. Magnetic vortex resonance in patterned ferromagnetic dots. *Phys. Rev. B*, 72:024455, 2005.

[27] A. Drews, B. Krüger, M. Bolte, and G. Meier. Current- and field-driven magnetic antivortices. *Phys. Rev. B*, 77: 094413, 2008.

[28] M. Bolte, G. Meier, B. Krüger, A. Drews, R. Eiselt, L. Bocklage, S. Bohlens, T. Tyliszczak, A. Vansteenkiste, B. Van Waeyenberge, K. W. Chou, A. Puzic, and H. Stoll. Time-resolved x-ray microscopy of spin-torque-induced magnetic vortex gyration. *Phys. Rev. Lett.*, 100:176601, 2008.

[29] D. Berkov and N. Gorn. Transition from the macrospin to chaotic behavior by a spin-torque driven magnetization precession of a square nanoelement. *Phys. Rev. B*, 71:052403, 2005.

4.2 Publication JAP'09

Proposal for a standard problem for micromagnetic simulations including spin-transfer torque

M. Najafi, B. Krüger, S. Bohlens, M. Franchin,
H. Fangohr, A. Vanhaverbeke, R. Allenspach, M. Bolte,
U. Merkt, D. Pfannkuche, D. P. F. Möller, and G. Meier

Jornal of Applied Physics **105**, 113914 (2009)

Proposal for a standard problem for micromagnetic simulations including spin-transfer torque

Massoud Najafi,[1,2,a)] Benjamin Krüger,[3,b)] Stellan Bohlens,[3] Matteo Franchin,[4] Hans Fangohr,[4] Antoine Vanhaverbeke,[5] Rolf Allenspach,[5] Markus Bolte,[1,2] Ulrich Merkt,[2] Daniela Pfannkuche,[3] Dietmar P. F. Möller,[1] and Guido Meier[2]
[1]*Arbeitsbereich Technische Informatiksysteme, Fachbereich Informatik, Universität Hamburg, Vogt-Kölln-Str. 30, 22527 Hamburg, Germany*
[2]*Institut für Angewandte Physik und Zentrum für Mikrostrukturforschung, Universität Hamburg, Jungiusstr. 11, 20355 Hamburg, Germany*
[3]*I. Institut für Theoretische Physik, Universität Hamburg, Jungiusstr. 9, 20355 Hamburg, Germany*
[4]*School of Engineering Sciences, University of Southampton, SO17 1BJ Southampton, United Kingdom*
[5]*IBM Zürich Research Laboratory, Säumerstrasse 4, CH-8803 Rüschlikon, Switzerland*

(Received 16 January 2009; accepted 23 March 2009; published online 5 June 2009)

The spin-transfer torque between itinerant electrons and the magnetization in a ferromagnet is of fundamental interest for the applied physics community. To investigate the spin-transfer torque, powerful simulation tools are mandatory. We propose a micromagnetic standard problem including the spin-transfer torque that can be used for the validation and falsification of micromagnetic simulation tools. The work is based on the micromagnetic model extended by the spin-transfer torque in continuously varying magnetizations as proposed by Zhang and Li. The standard problem geometry is a permalloy cuboid of 100 nm edge length and 10 nm thickness, which contains a Landau pattern with a vortex in the center of the structure. A spin-polarized dc current density of 10^{12} A/m^2 flows laterally through the cuboid and moves the vortex core to a new steady-state position. We show that the new vortex-core position is a sensitive measure for the correctness of micromagnetic simulators that include the spin-transfer torque. The suitability of the proposed problem as a standard problem is tested by numerical results from four different finite-difference and finite-element-based simulation tools. © *2009 American Institute of Physics*.
[DOI: 10.1063/1.3126702]

I. INTRODUCTION

Ferromagnets can be found in most devices that require nonvolatile storage of information. Ferromagnets have been successfully used in hard disks for more than 50 years.[1] Recently the field of research has been extended to the development of nanometer-sized ferromagnetic nonvolatile storage devices that offer a high storage density accompanied by a high data rate.[2] The magnetic random access memory (MRAM) has been developed as the first nanostructured ferromagnetic memory module.[3] An MRAM cell consists of a multilayer system with two ferromagnetic layers separated by a nonmagnetic layer. Information is stored in the orientation of the magnetization in the two ferromagnetic layers. Depending on the properties of the nonmagnetic layer, the information can be read with the help of the tunnel magnetoresistance effect[4] or the giant magnetoresistance effect.[5] For this, a current is applied to the multilayer. The resistance depends on the relative alignment of the magnetizations of the ferromagnetic layers. To write information in such a memory cell, a current is applied across two perpendicular wires. At the intersection of the two wires, the resulting Oersted field is strong enough to switch the magnetic orientation of the first magnetic layer, the so-called free layer. The magnetic orientation of the second ferromagnetic layer, the so-called pinned layer, should not change during this process.[3,6] The application of an Oersted field corresponds to the write process in a hard disk. As explained by Chappert *et al.*,[7] there are different restrictions using an Oersted field that limit the storage density of the MRAM. To increase the storage density, it is therefore necessary to find an alternative way to switch the magnetization.

Slonczewski[8,9] and Berger[10] predicted in 1996 that a spin-polarized current flowing through a ferromagnetic conductor can apply a relevant torque to its magnetization, owing to the exchange coupling between the spins of the itinerant electrons and those of the localized electrons. Since its discovery the so-called spin-transfer torque (STT) has been considered as a key to increase the storage density and lead to a new generation of storage devices, such as the STT random access memory (STTRAM) (Ref. 11) and the racetrack memory.[12] The STTRAM is an MRAM that uses the spin-transfer torque instead of the Oersted field for the switching process. The racetrack memory stores bits along a single ferromagnetic wire. To write and read information, a current is applied along the wire that moves the bits to a writing or reading unit.

Two theoretical descriptions of the spin-transfer torque exist: The first description has been developed by Slonczewski[8,9] and describes a current traversing an interface between a ferromagnet and a nonmagnetic metal and its concomitant torque on the magnetization. It can successfully de-

[a)]Electronic mail: mnajafi@physnet.uni-hamburg.de.
[b)]Electronic mail: bkrueger@physnet.uni-hamburg.de.

scribe a STTRAM. The second description has been developed by Berger[10] and was later refined by Zhang and Li[13] as well as by Thiaville *et al.*[14] It deals with the spin-transfer torque in the case of a continuously varying magnetization. In this case the spin-transfer torque acts on inhomogeneous magnetization patterns, such as domain walls or magnetic vortices. Thus, also the magnetic processes in a racetrack memory[12] and gyrating magnetic vortices driven by spin-transfer torque[15,16] can be described.

Other memory devices such as the dynamic random access memory[17] or the static random access memory[18] have shown that it is necessary to develop analytical descriptions and powerful simulation tools like SPICE (Ref. 19) to optimize their properties.[2] The theoretical descriptions of the spin-transfer torque[8–10,13,14] are the basis for devices that exploit the interaction between spin-polarized currents and magnetization. There exists a variety of simulation tools, such as the micromagnetic modeling and simulation kit M^3S,[20] NMAG,[21] the object-oriented micromagnetic framework OOMMF,[22] LLG,[23] and micromagus,[24] that implement the micromagnetic model[25] and include the spin-transfer torque model. To compare different simulation tools the micromagnetic modeling activity group (μMag) (Ref. 26) publishes standard problems for micromagnetism. These micromagnetic problems allow the results of a simulation tool to be verified. So far, there is no standard problem that includes the spin-transfer torque. Here we propose a problem that allows the validation of micromagnetic simulation tools that implement the spin-transfer torque of Berger[10] with the extension by Zhang and Li.[13] We further present numerical solutions to the proposed problem and analytical solutions of the problem given by Krüger *et al.*[27]

II. PROBLEM SELECTION

In this section, selection criteria for the standard problem are defined and possible adaptations of each criterion are given. The focus of our standard problem is the spin-transfer torque extension. Thus we chose criteria that ensure the traceability of errors in the implementation of this extension. A prerequisite is that the simulation tool derives correct results for the numerical time integration, the demagnetization field, the exchange field, and the Zeeman field.

A. Selection criteria

To select a standard problem that is appropriate to trace errors in the spin-transfer torque extension, we first define four general selection criteria. According to the strategy of μMag,[26] these criteria are:

(1) The problem has to be specified in such a way that different simulation tools are able to reproduce the initial magnetization configuration independent of their implementation.
(2) The problem has to ensure that the reaction of the magnetization depends significantly on the current and leads to an unambiguous time evolution of the magnetization.
(3) The problem has to be solvable in reasonable computation time. This is important to run the standard problem repeatedly, which is necessary to fix program errors.
(4) The problem has to offer an unambiguous and characteristic measure for the magnetization dynamics and thus enable verification or falsification of a simulation tool. This measure has to be computable conveniently and independently of the implementation of the tool.

B. Theoretical background

We use the micromagnetic model including the spin-transfer torque of Berger[10] with the extension by Zhang and Li.[13] The equation of motion of the magnetization is given by

$$\frac{\partial \vec{M}}{dt} = -\gamma \vec{M} \times \vec{H}_{\text{eff}} + \frac{\alpha}{M_s}\vec{M} \times \frac{d\vec{M}}{dt} - \frac{b_j}{M_s^2}\vec{M} \times (\vec{M} \times (\vec{j} \cdot \vec{\nabla})\vec{M}) - \xi \frac{b_j}{M_s}\vec{M} \times (\vec{j} \cdot \vec{\nabla})\vec{M}, \tag{1}$$

with the gyromagnetic ratio γ, the Gilbert damping parameter α, and the saturation magnetization M_s. The effective magnetic field $\vec{H}_{\text{eff}}$ includes the external as well as the internal fields. The coupling constant between the current and the magnetization is $b_j = (P\mu_B)/[eM_s(1+\xi^2)]$, where P denotes the spin polarization of the current density $\vec{j}$, μ_B the Bohr magneton, and $\xi = \tau_{\text{ex}}/\tau_{\text{sf}}$ the degree of nonadiabacity, which is the ratio between the exchange relaxation time τ_{ex} and the spin-flip relaxation time τ_{sf}. Equation (1) can be written in the explicit form

$$\frac{d\vec{M}}{dt} = -\gamma' \vec{M} \times \vec{H}_{\text{eff}} - \frac{\alpha\gamma'}{M_s}\vec{M} \times (\vec{M} \times \vec{H}_{\text{eff}}) - \frac{b_j'}{M_s^2}(1+\alpha\xi)\vec{M} \times (\vec{M} \times (\vec{j} \cdot \vec{\nabla})\vec{M}) - \frac{b_j'}{M_s}(\xi - \alpha)\vec{M} \times (\vec{j} \cdot \vec{\nabla})\vec{M}, \tag{2}$$

with the abbreviations $\gamma' = \gamma/(1+\alpha^2)$ and $b_j' = b_j/(1+\alpha^2)$ as written by Krüger *et al.*[28]

C. Adaptation of the criteria

On the basis of the physical model, we define the standard problem that complies with the criteria defined above. Criterion (1) is fulfilled by splitting the problem into two subproblems that are computed separately. Each subproblem is the computation of a separate simulation run. The first simulation is performed based on Eq. (2) in the absence of current $\vec{j}$. It starts from a magnetization pattern that has to be given by an equation. The resulting equilibrium magnetization is used as the initial magnetization for the second simulation with an applied current.

Criterion (2) can be fulfilled by the selection of an inhomogeneous magnetization pattern, e.g., a domain wall or a vortex, and the selection of a spatially and temporally homo-

geneous current. We decided to take a permalloy cuboid with a vortex pointing upwards for the initial equilibrium state of the second subproblem. The choice of a vortex and a spatially and temporally homogeneous current leads to an unambiguously distinguishable adiabatic and nonadiabatic reaction of the magnetization.[27,29,30] The equation of motion leads to a new steady state that provides a simple validation measure independent of the prior time evolution. In contrast, the choice of a resonant excitation of the vortex with alternating current is not suitable, because a small error in the simulated resonance frequency would drastically change the phase and amplitude of the result, which would complicate the falsification. A dc current reduces the complexity of the problem and enables to check the correctness of the results by the final steady state of the vortex core as a characteristic measure.

Criterion (3) can be met by a small number of discretization points and a magnetization pattern that exhibits significant changes within few time-integration steps. The number of discretization points is given by the size of the cuboid and the average distance between the discretization points. We use a small cuboid that still can relax to a vortex state. The discretization of the permalloy cuboid must be chosen such that the vortex core is resolved. The necessary resolution is achieved if the distance between the discretization points is significantly below the exchange length $l_{\mathrm{ex}}=\sqrt{2A/(\mu_0 M_s^2)}$, where A is the constant of the exchange interaction. To decrease the number of time-integration steps, we choose a large Gilbert damping parameter α, so that the magnetization rapidly reaches equilibrium.

Criterion (4) can be fulfilled by the calculation of the spatially averaged magnetization, which is proportional to the vortex-core position as shown in Appendix A. Thus the motion of the vortex core is an unambiguous and characteristic measure of the magnetization dynamics.[27]

III. PROBLEM DEFINITION

The problem is defined with the standard material parameters of permalloy,[31] with the exception of the Gilbert damping parameter α. These parameters are given by an exchange constant $A=13\times10^{-12}$ J/m, a saturation magnetization $M_s=8\times10^5$ A/m, which corresponds to an exchange length $l_{\mathrm{ex}}=5.7$ nm, and a gyromagnetic ratio $\gamma=2.211\times10^5$ m/C. According to criterion (3) we select a cuboid geometry with a sample size of $100\times100\times10$ nm^3 in the x-, y-, and z-directions, respectively. This allows the problem to be simulated with a spatial and temporal discretizations, which can be computed in a few hours on a standard personal computer.[32] In contrast with a circular film element, the cuboid geometry simplifies the comparison of simulation tools using finite-difference (FDM) and finite-element methods (FEM), because there are no irregular edges that are a possible source of errors in the FDM.

A. Computation of the starting condition without spin-transfer torque

In accordance with criterion (1), the first subproblem of the standard problem starts with an initial magnetization pattern as illustrated in Fig. 1(a). The initial vortex state relaxes into equilibrium as illustrated in Fig. 1(b). The initial magnetization pattern is chosen as

$$\vec{M}=M_s\cdot\frac{\vec{f}}{|\vec{f}|},\quad \vec{f}=\begin{pmatrix}-(y-y_0)\\ x-x_0\\ R\end{pmatrix}, \tag{3}$$

where $\vec{r}=(x,y,z)$ is the position of the cell and $x_0=y_0=50$ nm are the coordinates of the center of the cuboid. R is related to the radius of the vortex and is set to $R=10$ nm as this value leads to a short relaxation time. A Gilbert damping constant of $\alpha=1$ is chosen to obtain a fast relaxation and thus save computation time, but the relaxed equilibrium state is independent of α. The effective field is given by the exchange and the demagnetization field. The simulation stops when the magnetization has reached an equilibrium state. The stopping criterion is $\max_{\vec{r}\in\mathcal{V}}|1/M_s\cdot d\vec{M}/dt|\leq0.01$ rad/ns, where $\mathcal{V}$ is the volume of the cuboid. As shown in Fig. 1(b), the equilibrium state is a vortex as required by criterion (2). The vortex core points in the z-direction (positive polarization) and the in-plane magnetization curls counterclockwise (positive chirality).

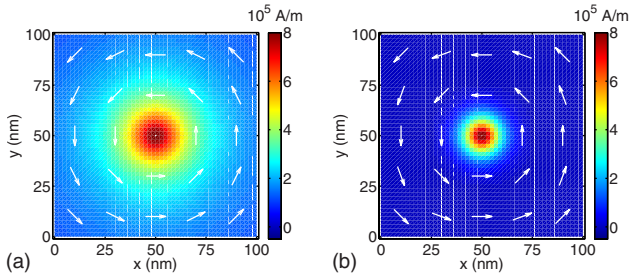


FIG. 1. (Color online) (a) Initial state of the magnetization for the first subproblem as given by Eq. (3). The magnetization is averaged along the z-direction. The color scale shows the z-component of the magnetization. (b) Relaxed vortex state as initial state for the second part of the computation including the spin-transfer torque. Simulations are computed with M^{3}S.

B. Computation including spin-transfer torque

The second subproblem, which includes the spin-transfer torque, starts with the equilibrium state of the first subproblem. The effective field is the same as in the first subproblem. As required in criterion (2) and illustrated in Fig. 2(a), a spatially homogeneous spin-polarized dc current of 10^{12} A/m^2 is instantaneously applied in the x-direction $[\vec{j}=(j,0,0)]$, i.e., the electrons flow from right to left. The damping constant $\alpha=0.1$ of this subproblem is chosen to obtain a reasonable fast relaxation on the one hand and enough oscillations to assist the comparison of results from different simulation packages on the other hand. The value also allows the detection of errors of the spin-transfer torque term that depend on the damping parameter α. The degree of nonadiabaticity $\xi=0.05$ is chosen to get a significant contribution of the nonadiabatic spin-transfer torque term to the final vortex-core position and to achieve a nonzero contribution of the fourth term in Eq. (2). The simulation stops when the stopping criterion $\max_{\vec{r}\in\mathcal{V}}|1/M_s\cdot d\vec{M}/dt|\leq0.01$ rad/ns has been reached. To compare different simulation packages,

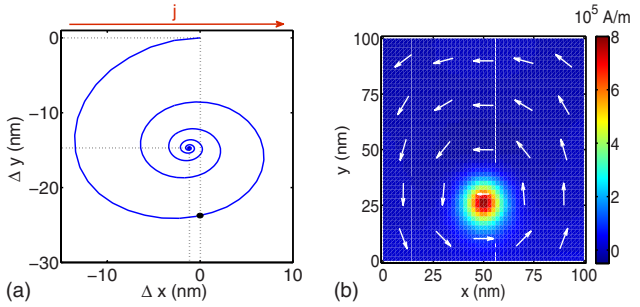


FIG. 2. (Color online) (a) Two-dimensional representation of the position of the vortex core as a function of time. The dot indicates the vortex-core position at the time $t=0.73$ ns. (b) Snapshot of the magnetization of the permalloy cuboid at $t=0.73$ ns when the vortex-core position crosses the line $\Delta x=0$ for the first time. The magnetization is excited by a homogeneous spin-polarized current density of 10^{12} A/m^2 in the x-direction, i.e., the electrons flow from right to left. The magnetization is averaged along the z-direction. The color scale is the same as in Fig. 1. Simulations are computed with M^{3}S.

one has to calculate the spatially averaged magnetization over time. The resulting trajectory of the simulation shows a damped rotation of the vortex core around a new steady-state position of $\Delta x=x-x_0=-1.2$ nm and $\Delta y=y-y_0=-14.7$ nm, as illustrated in Fig. 2. The vortex-core position Δx, Δy is related to the center of the cuboid. It is determined by averaging the magnetization along the z-direction and interpolating the out-of-plane magnetization in the x- and y-directions with a polynomial of second order. The position of the vortex core is then given by the maximum of this polynomial.

C. Falsification properties

Suitable falsification properties as demanded in criterion (4) are important for the development of a simulation tool. The influence of errors in the spin-transfer torque extension or an improper, i.e., too coarse, spatial discretization has been investigated for the proposed standard problem and is outlined in the following.

1. Sensitivity to errors in the spin-transfer torque extension

First we analyze the influence of errors in the spin-transfer torque extension. To show the sensitivity of the problem to those errors, we investigate changes in the spin-transfer torque given by a constant factor. This is emulated by a variation in the degree of nonadiabaticity ξ and the current density j. The analytical model explained in Appendix B predicts that a change in ξ will linearly affect the y-component of the spatially averaged magnetization $\langle M_y\rangle$, whereas a change in j will affect the x- and y-components of the spatially averaged magnetization $\langle M_x\rangle$ and $\langle M_y\rangle$ equally. Figure 3 shows three sets of parameters for ξ and j that illustrate the clearly distinguishable reactions of the magnetization to a change in the adiabatic, the nonadiabatic, and the entire spin-transfer torque. As a first set we chose an increased spin-transfer torque realized by an increased current density. It leads to a proportionally increased x- and y-component $\langle M_x\rangle$ and $\langle M_y\rangle$ of the spatially averaged magnetization during its time evolution. The second set is an

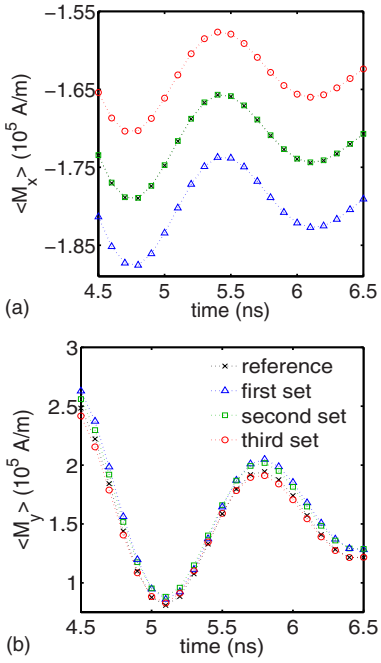


FIG. 3. (Color online) (a) Spatially averaged magnetization $\langle M_x\rangle$ and (b) $\langle M_y\rangle$ for different values of ξ and j. The crosses show the time evolution of the spatially averaged magnetization for the reference parameters $\xi=0.05$ and $j=10^{12}$ A/m^2. The triangles show the result for the first set of parameters, when the spin-transfer torque parameter j is increased by 5%. The squares show the result of the second set, when the nonadiabatic spin-transfer torque parameter ξ is increased by 5%. The circles show the results of the third set, when the adiabatic spin-transfer torque is changed by a simultaneous decrease in the current density and increase in ξ by 5% each. The maximum difference of the spatially averaged magnetization amounts to 14.40 kA/m (5.11%) and 8.40 kA/m (5.34%) (percentage values are related to the maximum values of $|\langle M_x\rangle|=281.61$ kA/m and $|\langle M_y\rangle|=157.43$ kA/m) for $\langle M_x\rangle$ and $\langle M_y\rangle$, respectively. Simulations are computed with M^{3}S.

increased nonadiabatic spin-transfer torque created by an increased degree of nonadiabaticity ξ. This configuration leads to a proportionally increased y-component $\langle M_y\rangle$ of the averaged magnetization during the time evolution of the magnetization. The third set describes a decreased influence of the adiabatic spin-transfer torque term obtained by simultaneously decreasing j and increasing ξ. This configuration induces a proportionally decreased x-component $\langle M_x\rangle$ of the spatially averaged magnetization during the time evolution of the magnetization. The results illustrate that a variation in ξ and j results in a clear change of the magnetization which, according to Appendix B, should be linear with the change in ξ and j. As illustrated in Fig. 3, a variation in the adiabatic spin-transfer torque by a constant factor linearly affects the x-component of the spatially averaged magnetization $\langle M_x\rangle$, whereas a variation in the nonadiabatic spin-transfer torque by a constant factor linearly affects the y-component of the spatially averaged magnetization $\langle M_y\rangle$. This enables one to distinguish between errors in the adiabatic and the nonadiabatic term. These linear changes are also in agreement with Eq. (B1).

2. Improper spatial discretization

To investigate the influence of the spatial discretization, we vary the number of discretization points of the FDM and

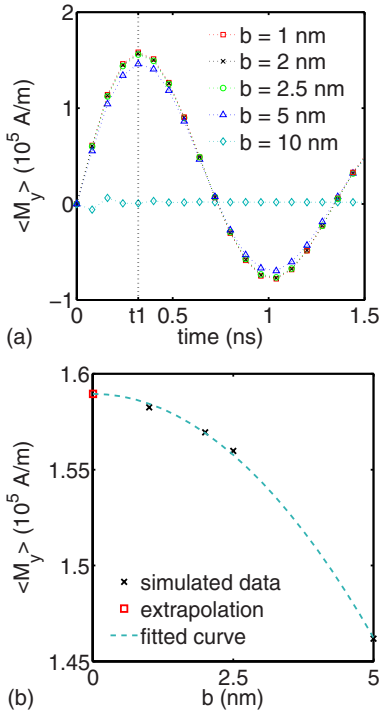


FIG. 4. (Color online) (a) Spatially averaged magnetization component $\langle M_y \rangle$ for different cell sizes b^3 computed with M^{3}S. (b) The y-component of the spatially averaged magnetization component $\langle M_y \rangle$ at time $t_1 = 0.32$ ns vs b.

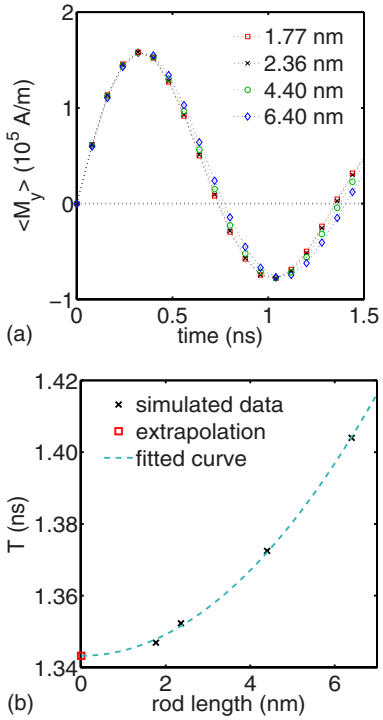


FIG. 5. (Color online) Results for different FEM meshes computed with NMAG (Ref. 21). As maximum rod lengths 1.77, 2.36, 4.40, and 6.40 nm are chosen, which corresponds to 355 488, 150 282, 25 560, and 8874 tetrahedra, respectively. (a) Spatially averaged magnetization $\langle M_y \rangle$. (b) Duration of the first gyration cycle vs rod length.

FEM meshes. A FDM mesh is a grid that consists of equally sized cuboids (so-called discretization cells). FEM meshes, in contrast, cannot be described that simply, because here the size of each finite element can vary. To investigate the influence of the spatial discretization, we simulated the problem for five different cell sizes using the FDM-based tool M^{3}S. The cell sizes used were $b \times b \times b$, for $b = 1$, 2, 2.5, 5, and 10 nm. Figure 4(a) shows the time evolution of the y-component of the spatially averaged magnetization for the different cell sizes. Results for cell sizes $b = 1$, 2, 2.5, and 5 nm show a slight decrease in the spatially averaged magnetization with increasing cell size. For a cell size of $b = 10$ nm, no vortex is formed, i.e., criterion (3) is not fulfilled. Figure 4(b) shows the y-component of the spatially averaged magnetization at time $t = 0.32$ ns versus cell size b fitted by a quadratic function. The extrapolation to $b = 0$ suggests that it is sufficient to take a FDM mesh with a cell size of $2 \times 2 \times 2$ nm^3.

We also simulated the problem for four FEM meshes using the FEM-based tool NMAG. Readers interested in FEM meshing can find a detailed description of the meshes used in the FEM simulations in Appendix C. In the following, we use the maximum rod length and the number of tetrahedra as characteristic measures for the fineness of a mesh. The simulations with NMAG are performed with maximum rod lengths of 1.77, 2.36, 4.40, and 6.40 nm, corresponding to 355488, 150282, 25560, and 8874 tetrahedra, respectively. Figure 5(a) shows the time evolution of the y-component of the spatially averaged magnetization for the different meshes. The results reveal a slight decrease in the precession frequency with increased rod length. Figure 5(b) shows the duration of the first gyration cycle for the rod length extrapolated to 0 nm by a quadratic function. The extrapolation suggests that it suffices to take a FEM mesh with a rod length of 2.36 nm. In accordance with the simulations of standard problem numbers 1–4 (Ref. 26), these results illustrate that to obtain reliable numerical results the distance between two discretization points should be significantly below the exchange length l_{ex}.

IV. COMPARISON OF EXISTING TOOLS

We compare the simulation results of OOMMF extended by Krüger *et al.*,[28] of OOMMF extended by Vanhaverbeke *et al.*,[33,34] of M^{3}S (Ref. 20) and of NMAG.[21] The results of both OOMMF-extensions and of M^{3}S have been computed using a cell size of $2 \times 2 \times 2$ nm^3, whereas the results of NMAG are computed using a mesh of type (1) as described in Appendix C with a maximum rod length of 1.77 nm. The corresponding regular mesh has 68211 mesh nodes, of which 17566 are surface nodes. The time evolution of the magnetization is performed by explicit or implicit numerical integration algorithms. Both tools, the spin-transfer torque extended OOMMF version of Krüger *et al.*[28] and M^{3}S,[20] use an implementation of a fifth-order Cash–Karp Runge–Kutta algorithm[35] with an absolute error tolerance of 10^{-3} A/m and a relative error tolerance of 10^{-4}. The spin-transfer torque extended OOMMF version of Vanhaverbeke *et al.*[33,34] uses a fifth-order Dormand–Prince Runge–Kutta algorithm[36] with the same error tolerances. NMAG uses the sundials libraries[37] with an absolute error tolerance of 8×10^{-2} A/m and a relative error tolerance of 10^{-7}. Figure 6 shows the time evolution of the magnetization for all tools, whereas in Table I the spatially averaged magnetization components for the relaxed state are

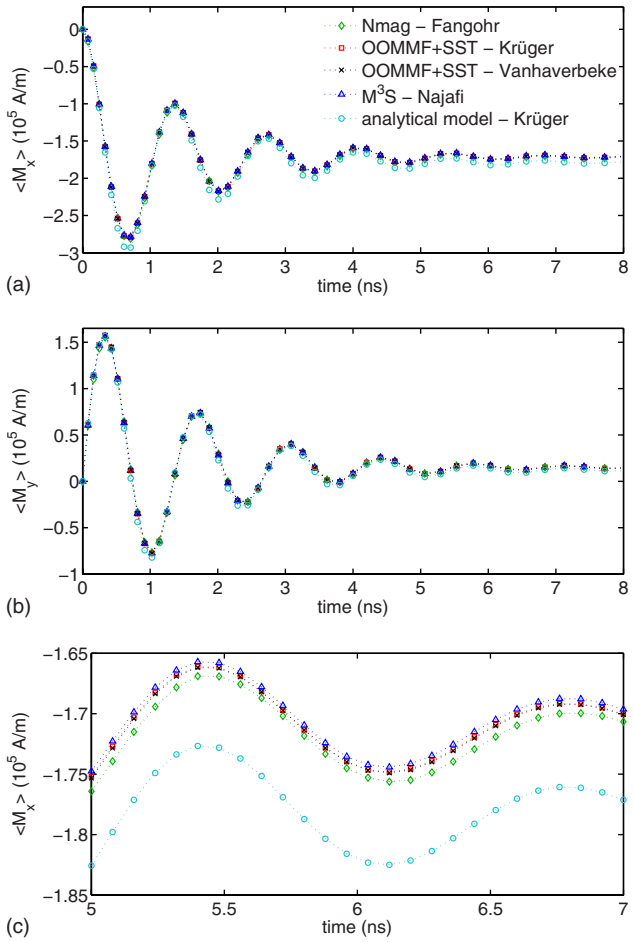


FIG. 6. (Color online) Solution of the proposed standard problem for a $100\times100\times10$ nm^3 permalloy cuboid calculated with four different simulation tools and the analytical model. A spatially and temporally homogeneous current density of 10^{12} A/m^2 is applied instantaneously in the x-direction. (a) The x-component of spatially averaged magnetization $\langle M_x\rangle$ and (b) $\langle M_y\rangle$. (c) Close-up of the x-component $\langle M_x\rangle$ for the time interval 5 ns$\leq t\leq$7 ns.

listed. For comparison we also plot the analytically calculated values according to Krüger *et al.*,[27] which is explained in more detail in Appendix B. The maximum difference of the spatially averaged magnetization between the simulation tools amounts to 5.41 kA/m (1.9%) (Ref. 38) (3.0%) (Ref. 38) for $\langle M_x\rangle$ and $\langle M_y\rangle$, respectively. In comparison with the analytical model, these differences are 16.14 kA/m (5.7%) (Ref. 38) and 11.27 kA/m (7.2%) (Ref. 38) for $\langle M_x\rangle$ and $\langle M_y\rangle$, respectively.

TABLE I. Spatially averaged magnetizations $\langle M_x\rangle$ and $\langle M_y\rangle$ for the simulation tools and the analytical model at t=14 ns when the vortex reached the new equilibrium position. All values in the table are rounded to two decimal places.

Tools	$\langle M_x\rangle$ $(1\times10^5$ A/m)	$\langle M_y\rangle$ $(1\times10^4$ A/m)
OOMMF+STT—Krüger	−1.71	1.51
OOMMF+STT—Vanhaverbeke	−1.71	1.50
M^{3}S—Najafi	−1.71	1.50
NMAG—Fangohr	−1.72	1.52
Analytical model—Krüger	−1.78	1.12

We believe that the differences between the results in Fig. 6 are due to the implementation of the demagnetization field. A comparison of the simulation results of OOMMF and M^{3}S for standard problem number 4 (Ref. 26) shows that they only differ in the calculation of the demagnetization field.[39] The spatially averaged magnetization of both OOMMF extensions are virtually identical but differ more significantly from M^{3}S. Both M^{3}S and the OOMMF extensions use a demagnetization field implementation based on Newel *et al.*[40] Unlike M^{3}S, OOMMF in addition uses an interpolation method to speed up the calculation of the demagnetization tensor. The FEM-based spatial discretization computes the demagnetization field with the hybrid finite element/boundary element method described by Fredkin and Köhler.[41] The difference between the numerical and the analytical results are a direct consequence of the approximations of the underlying analytical model, as explained in Appendix B. These results verify the suitability of the proposed standard problem, as the problem discriminates errors larger than about 3% (Ref. 38) and, in contrast with standard problem number 4, no point of discontinuity is identified.

V. EXPERIMENTAL FEASIBILITY

Although not required for the proof of the micromagnetic simulations, it is nevertheless important to choose a problem that can be proved by experiments. Permalloy cuboids that exhibit the simulated magnetization configuration shown in Figs. 1 and 2 including wires contacting their left and right edges can be fabricated by electron-beam lithography and liftoff processing.[15] Experimentally it is a challenge to apply current densities in the 10^{12} A/m^2 regime permanently because of the concomittant large Joule heating. However, recently this problem has been solved by the preparation of permalloy nanostructures on diamond substrates.[42] The diamond serves as a highly efficient heat sink and it has been demonstrated that current densities in excess of 10^{12} A/m^2 can be applied continuously to samples like the one required for the proposed standard problem. The detection of the vortex core at the shifted position could, for example, be performed by scanning electron microscopy with polarization analysis (SEMPA).[43,44] As SEMPA detects the final steady-state position of the vortex core, the value of the damping constant α=0.1 used in the simulation is not relevant. The degree of nonadiabaticity ξ=0.05 is a realistic experimental value.[45] As so far no experimental results of the proposed sample geometry are available, we validate the results of the micromagnetic simulations with the analytical model explained in detail in Appendix B. This model can serve as a reference because it has been already verified by experimental results on similar device geometries.[15]

VI. CONCLUSION

In this work we present a standard problem for micromagnetic simulation packages extended by the spin-transfer torque. For this standard problem, we defined the criteria necessary to ensure that the problem is suitable for the validation and falsification of micromagnetic simulation tools. These criteria have been applied to the underlying extended

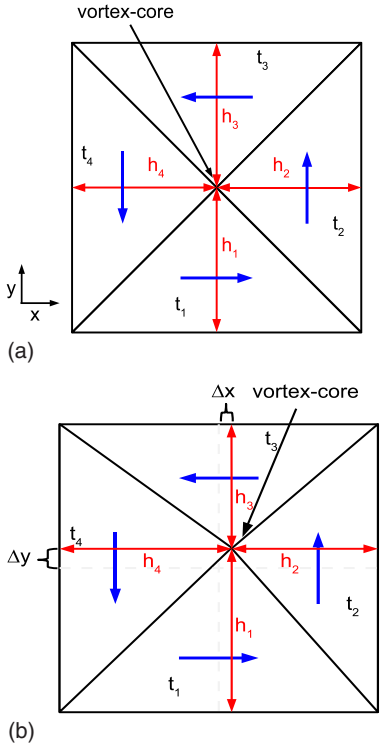


FIG. 7. (Color online) Model for the vortex motion as introduced by Krüger *et al.* (Ref. 27). The magnetization pattern is described by four triangles t_1 to t_4. The vortex core is at the center of the four triangles. (a) Magnetization pattern with the vortex core at the center of the sample. (b) Magnetization configuration with a vortex core displaced from the center by Δx and Δy.

micromagnetic model. We demonstrated that the standard problem has the required properties. To prove the good validation and falsification properties, we investigated the influence of typical errors, such as erroneous variations in the spin-transfer torque extension by a constant factor or an improper spatial discretization. The final comparison of the results for different tools substantiates these properties and shows that the problem discriminates errors larger than 5.41 kA/m (1.9%) (Ref. 38) and 4.80 kA/m (3.0%) (Ref. 38) for $\langle M_x \rangle$ and $\langle M_y \rangle$, respectively.

ACKNOWLEDGMENTS

Financial support by the Deutsche Forschungsgemeinschaft via the Graduiertenkolleg 1286 "Functional metal-semiconductor hybrid systems" and via the SFB 668 "Magnetism from single atoms to nanostructures," by the EPSRC (Grant Nos EP/E040063/1 and EP/E039944/1), and by the ESF EUROCORES collaborative research project SpinCurrent under the Fundamentals of Nanoelectronics program is gratefully acknowledged.

APPENDIX A: RELATION BETWEEN SPATIALLY AVERAGED MAGNETIZATION AND VORTEX-CORE POSITION

To show the correspondence of the vortex-core position and the spatially averaged magnetization, we use the model introduced by Krüger *et al.*,[27] where the vortex is described by four triangles t_1 to t_4 shown in Fig. 7. The magnetization in each triangle is assumed to be homogeneous. If the vortex core is in the center of the cuboid, all four triangles have the same volume.

As t_1 and t_3 as well as t_2 and t_4 have an antiparallel magnetization, the spatially averaged magnetization is zero. A deflection of the vortex core from the center of the cuboid changes the size of the triangles as illustrated in Fig. 7(b). The dependence of the spatially averaged magnetization on the volume differences and the deflection of the vortex core is given by

$$\begin{pmatrix} \langle M_x \rangle \\ \langle M_y \rangle \\ \langle M_z \rangle \end{pmatrix} = \begin{pmatrix} cM_s k \dfrac{V_1 - V_3}{V_{\text{cuboid}}} \\ cM_s k \dfrac{V_2 - V_4}{V_{\text{cuboid}}} \\ p \text{ const} \end{pmatrix} = \begin{pmatrix} cM_s k \dfrac{ld\Delta y}{l^2 d} \\ cM_s k \dfrac{ld(-\Delta x)}{l^2 d} \\ p \text{ const} \end{pmatrix} = \begin{pmatrix} cM_s k \dfrac{\Delta y}{l} \\ -cM_s k \dfrac{\Delta x}{l} \\ p \text{ const} \end{pmatrix}. \tag{A1}$$

Here V_i is the volume of triangle t_i, l is the edge length of the cuboid, d is its thickness, c is the chirality of the magnetization pattern, p is the polarization of the vortex, $\Delta x = (h_4 - h_2)/2$ is the deflection of the vortex core in the x-direction, $\Delta y = (h_1 - h_3)/2$ is the deflection in the y-direction, and h_i is the height of triangle t_i. The dimensionless fit parameter k is needed to convert the vortex-core position into the spatially averaged magnetization and takes into account that the domain walls between the triangles in Fig. 7 have a finite size and are not abrupt as treated in Eq. (A1). The value of k changes with the system size and is 1.4517 for the proposed geometry. Because of the cuboid geometry, the x-component of the spatially averaged magnetization $\langle M_x \rangle$ is proportional to the deflection Δy of the vortex core in the y-direction and the y-component of the spatially averaged magnetization $\langle M_y \rangle$ is proportional to the deflection Δx in the x-direction.

APPENDIX B: ANALYTICAL MODEL

The vortex-core position can be calculated by the analytical model described in Ref. 27. This model is in accordance with experimental results on the spin-transfer torque.[15] For a square, the model predicts that the final deflection of the vortex core in the x-direction depends only on the nonadiabatic spin-transfer torque term and that the final deflection in the y-direction depends only on the adiabatic spin-transfer torque term,

$$\begin{pmatrix} \Delta x_{\text{end}} \\ \Delta y_{\text{end}} \end{pmatrix} = - \begin{pmatrix} \dfrac{b_j j \Gamma \xi}{\alpha(\omega^2 + \Gamma^2)} \\ \dfrac{b_j j \omega}{\omega^2 + \Gamma^2} \end{pmatrix}. \tag{B1}$$

Here ω is the free frequency of the gyration of the vortex core, Γ is the damping constant of the vortex, α is the Gilbert

damping constant, and $(\Delta x_{\text{end}}, \Delta y_{\text{end}})$ is the final position of the vortex core related to the center of the cuboid. The time evolution of the core's position,

$$\begin{pmatrix} \Delta x(t) \\ \Delta y(t) \end{pmatrix} = \begin{pmatrix} Aie^{(-\Gamma+i\omega)t} - Bie^{(-\Gamma-i\omega)t} + \Delta x_{\text{end}} \\ Ae^{(-\Gamma+i\omega)t} + Be^{(-\Gamma-i\omega)t} + \Delta y_{\text{end}} \end{pmatrix}, \tag{B2}$$

depends on the coefficients $A=(-\Delta y_{\text{end}}+i\Delta x_{\text{end}})/2$ and $B=(-\Delta y_{\text{end}}-i\Delta x_{\text{end}})/2$. Owing to approximations within the analytical model concerning the detailed magnetization pattern a perfect agreement with the micromagnetic simulations cannot be expected.

APPENDIX C: USED FINITE-ELEMENT MESHES

We used two different types of finite-element meshes in the calculations with NMAG (Ref. 21):

(1) Meshes created by decomposing the cuboidal body into cubes,
(2) Meshes generated with the advancing front method using NETGEN.[46]

For method (1), each cube is subdivided into six tetrahedra consistently with the neighboring cubes. The cubes are then skewed to obtain nearly equilateral triangles on the surface of the mesh. We keep only those tetrahedra that lie within the ferromagnetic region and adjust those that intersect the meshing region surface (the points outside the meshing region are projected back onto its surface). The advantages of using this "regular mesh" are that all edge lengths are exactly known and that the mesh generation is very fast for the cuboidal geometry. For the unstructured tetrahedral mesh (2), we use the mesh generator NETGEN,[46] which is based on the advancing front method. The results of NMAG in Sec. IV have been computed using a mesh of type (1) with a maximum edge length of 1.77 nm that has 68211 mesh nodes, of which 17566 are surface nodes. This has been compared with an unstructured mesh generated with NETGEN with 25887 points and rod lengths varying from 1 to 3.8 nm, with an average rod length of 1.95 nm. The simulation results are virtually independent of the mesh types used.

[1] D. A. Thompson and J. S. Best, IBM J. Res. Dev. **44**, 311 (2000).
[2] International Technology Roadmap for Semiconductors 2007 Edition, Semiconductor Industry Association, 2007, http://www.itrs.net/Links/2007ITRS/Home2007.htm
[3] T. M. Maffitt, J. K. DeBrosse, J. A. Gabric, E. T. Gow, M. C. Lamorey, J. S. Parenteau, D. R. Willmott, M. A. Wood, and W. J. Gallagher, IBM J. Res. Dev. **50**, 25 (2006).
[4] M. Jullière, Phys. Lett. **54A**, 225 (1975).
[5] G. Binasch, P. Grünberg, F. Saurenbach, and W. Zinn, Phys. Rev. B **39**, 4828 (1989).
[6] E. C. Stoner and E. P. Wohlfarth, Philos. Trans. R. Soc. London, Ser. A **240**, 599 (1948).
[7] C. Chappert, A. Fert, and F. N. Van Dau, Nature Mater. **6**, 813 (2007).
[8] J. Slonczewski, J. Magn. Magn. Mater. **159**, L1 (1996).
[9] J. Slonczewski, J. Magn. Magn. Mater. **247**, 324 (2002).
[10] L. Berger, Phys. Rev. B **54**, 9353 (1996).
[11] M. Hosomi, H. Yamagishi, T. Yamamoto, K. Bessho, Y. Higo, K. Yamane, H. Yamada, M. Shoji, H. Hachino, C. Fukumoto, H. Nagao, and H. Kano Tech. Dig. - Int. Electron Devices Meet. **2005**, 459.
[12] S. S. P. Parkin, M. Hayashi, and L. Thomas, Science **320**, 190 (2008).
[13] S. Zhang and Z. Li, Phys. Rev. Lett. **93**, 127204 (2004).
[14] A. Thiaville, Y. Nakatani, J. Miltat, and Y. Suzuki, Europhys. Lett. **69**, 990 (2005).
[15] M. Bolte, G. Meier, B. Krüger, A. Drews, R. Eiselt, L. Bocklage, S. Bohlens, T. Tyliszczak, A. Vansteenkiste, B. Van Waeyenberge, L. W. Chou, A. Puzic, and H. Stoll, Phys. Rev. Lett. **100**, 176601 (2008).
[16] S. Bohlens, B. Krüger, A. Drews, M. Bolte, G. Meier, and D. Pfannkuche, Appl. Phys. Lett. **93**, 142508 (2008).
[17] J. A. Mandelman, R. H. Dennard, G. B. Bronner, J. K. DeBrosse, R. Divakaruni, Y. Li, and C. J. Radens, IBM J. Res. Dev. **46**, 187 (2002).
[18] R. W. Mann, W. W. Abadeer, M. J. Breitwisch, O. Bula, J. S. Brown, B. C. Colwill, P. E. Cottrell, W. G. Crocco, S. S. Furkay, M. J. Hauser *et al.*, IBM J. Res. Dev. **47**, 553 (2003).
[19] A. Vladimirescu, *The SPICE Book* (Wiley, New York, 1994).
[20] M. Najafi, B. Krüger, S. Bohlens, G. Selke, B. Güde, M. Bolte, and D. P. F. Möller, Proceedings of the 2008 Conference on Grand Challenges in Modeling and Simulation, 2008 (unpublished), p. 427.
[21] T. Fischbacher, M. Franchin, G. Bordignon, and H. Fangohr, IEEE Trans. Magn. **43**, 2896 (2007).
[22] M. J. Donahue and D. G. Porter, *OOMMF User's Guide*, Version 1.0 (National Institute of Standards and Technology, Gaithersburg, MD, 1999), Vol. 6376.
[23] M. R. Scheinfein, LLG, micromagnetics simulator, 2008, http://llgmicro.home.mindspring.com/
[24] D. V. Berkov, MICROMAGUS, software for micromagnetic simulation, 2008, http://www.micromagus.de
[25] W. F. Brown, Jr., *Micromagnetics* (Interscience, New York, 1963).
[26] Micromagnetic Modeling Activity Group, National Institute of Standards and Technology, Gaithersburg, MD, 2008, http://www.ctcms.nist.gov/rdm/mumag.org.html
[27] B. Krüger, A. Drews, M. Bolte, U. Merkt, D. Pfannkuche, and G. Meier, Phys. Rev. B **76**, 224426 (2007).
[28] B. Krüger, D. Pfannkuche, M. Bolte, G. Meier, and U. Merkt, Phys. Rev. B **75**, 054421 (2007).
[29] S.-K. Kim, K.-S. Lee, Y.-S. Yu, and Y.-S. Choi, Appl. Phys. Lett. **92**, 022509 (2008).
[30] K.-S. Lee, Y.-S. Yu, Y.-S. Choi, D.-E. Jeong, and S.-K. Kima, Appl. Phys. Lett. **92**, 192513 (2008).
[31] A. Hubert and R. Schäfer, *Magnetic Domains: The Analysis of Magnetic Microstructures* (Springer, Berlin, 1998).
[32] The computation time is measured with a simulation run on a computer containing a Intel-Core-Duo-E6600 microprocessor with a performance of 19.20 GFlops (Ref. 47). The simulation uses one core.
[33] A. Vanhaverbeke, OOMMF, extension of spin-transfer torque terms for current-induced domain wall motion, 2008, http://www.zurich.ibm.com/st/magnetism/spintevolve.html
[34] A. Vanhaverbeke, A. Bischof, and R. Allenspach, Phys. Rev. Lett. **101**, 107202 (2008).
[35] J. R. Cash and A. H. Karp, ACM Trans. Math. Softw. **16**, 201 (1990).
[36] J. R. Dormand and P. J. Prince, J. Comput. Appl. Math. **6**, 19 (1980).
[37] Sundials libraries, 2008, http://acts.nersc.gov/sundials/index.html
[38] Percentage values are related to the maximum values of $|\langle M_x \rangle| = 281.61$ kA/m and $|\langle M_y \rangle| = 157.43$ kA/m and 4.80 kA/m.
[39] M.-A. B. W. Bolte, M. Najafi, G. Meier, and D. P. F. Möller, Proceedings of the 2007 Summer Computer Simulation Conference, 2007 (unpublished).
[40] A. Newell, W. Williams, and D. Dunlop, J. Geophys. Res. **98**, 9551 (1993).
[41] D. R. Fredkin and T. R. Köhler, IEEE Trans. Magn. **26**, 415 (1990).
[42] S. Hankemeier, K. Sachse, Y. Stark, R. Frömter, and H. P. Oepen, Appl. Phys. Lett. **92**, 242503 (2008).
[43] R. Allenspach and P.-O. Jubert, MRS Bull. **31**, 395 (2006).
[44] H. Hopster and H. P. Oepen, *Magnetic Microscopy of Nanostructures* (Springer, Berlin, 2004).
[45] S. Lepadatu, M. C. Hickey, A. Potenza, H. Marchetto, T. R. Charlton, S. Langridge, S. S. Dhesi, and C. H. Marrows, Phys. Rev. B **79**, 094402 (2009).
[46] J. Schöberl, Comput. Visualization Sci. **1**, 41 (1997).
[47] Intel microprocessors, 2008, http://www.intel.com/support/processors/sb/CS-023143.htm

4.3 Publication PRL'10

Proposal of a Robust Measurement Scheme for the Nonadiabatic Spin Torque Using the Displacement of Magnetic Vortices

B. Krüger, M. Najafi, S. Bohlens,
R. Frömter, D. P. F. Möller, and D. Pfannkuche

Physical Review Letters **104**, 077201, 2010

Proposal of a Robust Measurement Scheme for the Nonadiabatic Spin Torque Using the Displacement of Magnetic Vortices

Benjamin Krüger,[1] Massoud Najafi,[2] Stellan Bohlens,[1] Robert Frömter,[3] Dietmar P. F. Möller,[2] and Daniela Pfannkuche[1]

[1]*I. Institut für Theoretische Physik, Universität Hamburg, Jungiusstr. 9, 20355 Hamburg, Germany*

[2]*Arbeitsbereich Technische Informatik Systeme, Universität Hamburg, Vogt-Kölln-Str. 30, 22527 Hamburg, Germany*

[3]*Institut für Angewandte Physik, Universität Hamburg, Jungiusstr. 11, 20355 Hamburg, Germany*

(Received 15 May 2009; revised manuscript received 26 November 2009; published 17 February 2010)

A spin-polarized current traversing a ferromagnet with continuously varying magnetization exerts a torque on the magnetization. The nonadiabatic contribution to this spin-transfer torque is currently under strong debate, as its value differs by orders of magnitude in theoretical predictions and in measurements. Here, a measurement scheme is presented that allows us to determine the strength of the nonadiabatic spin torque accurately and directly. Analytical and numerical calculations show that the scheme is robust against the uncertainties of the exact current direction and Oersted fields.

DOI: 10.1103/PhysRevLett.104.077201 PACS numbers: 75.60.Ch, 72.25.Ba, 75.70.Kw

A spin-polarized current flowing through a ferromagnetic sample interacts with the magnetization and exerts a torque on the local magnetic moments. This effect allows for direct and local manipulation of the magnetization in multidomain nanostructures and is a promising writing mechanism for new nonvolatile memory devices with high storage density. For conduction electron spins that follow the local magnetization adiabatically it has been shown that the interaction via spin transfer can be described by adding a current-dependent term to the Landau-Lifshitz-Gilbert equation [1]. This equation has been extended by an additional term that takes the nonadiabatic influence of the itinerant spins into account [2]. The strength of the nonadiabatic spin torque is quantified by the phenomenological parameter ξ. Theoretically, several mechanisms have been proposed as the origin of the nonadiabatic spin torque, leading to different orders of magnitude for ξ [2–6]. Thus a precise measurement of the nonadiabatic spin torque is necessary to give insight into its microscopic origin. The determination of ξ is further important for a reliable prediction of the current-driven domain-wall velocity [2] which is important for applications. Currently measured values of ξ for Permalloy differ by 1 order of magnitude [7–10]; thus the value of ξ is under strong debate. In these experiments, the observed motion of a domain wall was compared with micromagnetic simulations to determine ξ. However, this analysis is highly susceptible to surface roughness and Oersted fields.

Because of its high symmetry and spatial confinement, a vortex in a micro- or nanostructured magnetic thin-film element is a promising system for the investigation of the spin-torque effect [11–13]. Vortices are formed when the in-plane magnetization curls around a center region. In this few-nanometer-large center region, called the vortex core, the magnetization turns out of plane to minimize the exchange energy. There are four different ground states of a vortex. These states are labeled by the direction of the out-of-plane magnetization, called polarization p, and the sense of rotation of the in-plane magnetization, called chirality c. Polarizations of $p = 1$ and $p = -1$ denote a core that points parallel or antiparallel to the z axis, respectively. A chirality of $c = 1$ denotes a counterclockwise curling of the in-plane magnetization while $c = -1$ denotes a clockwise curling. It is known that vortices are displaced from their equilibrium position when excited by spin-polarized electric currents [12–20]. The spatial confinement of the vortex core within the film element yields an especially accessible system for measurements with scanning probe techniques, such as soft x-ray microscopy, x-ray photoemission electron microscopy, or scanning electron microscopy with polarization analysis. An analytical solution of the extended Landau-Lifshitz-Gilbert equation shows that for a current-driven vortex the forces due to the adiabatic and the nonadiabatic spin torque are perpendicular to each other [15].

In this Letter we present a scheme which allows us to measure the contributions due to the adiabatic spin torque, the nonadiabatic spin torque, and the Oersted field separately. It is based upon analytical calculations [15] and overcomes the two main difficulties that occur in an experiment. The first problem arises from an additional vortex displacement due to the Oersted field accompanying the current flow [12]. This displacement is comparable in size to the displacement due to the nonadiabatic spin torque and both displacements point in the same direction [15]. Thus, the unknown contribution of the Oersted field has to be separated from the measured signal. The second problem is the exact determination of the displacement angle. Since the displacement due to the adiabatic spin torque is about 1 order of magnitude larger than the displacement due to the nonadiabatic spin torque, a small uncertainty in the direction of the current through the sample would cause large errors in the determination of ξ. To test the applicability of our analytical findings, they

are applied to vortex displacements obtained from three-dimensional micromagnetic simulations.

For the analytical calculations we start from a modified version of the Thiele equation [21,22]

$$\vec{F} + G_0 \vec{e}_z \times (\vec{v}_c + b_j \vec{j}) + D_\Gamma \alpha \vec{v}_c + D_0 \xi b_j \vec{j} = 0 \quad (1)$$

that takes deformation of the vortex into account [23]. Here v_c is the velocity of the vortex core, α is the Gilbert damping, $\vec{F}$ the force on the vortex, $G_0 = -p|G_0|$ the z component of the gyrovector, and D_0 the diagonal element of the dissipation tensor. The coupling constant $b_j = P\mu_B/(eM_s)$ between the current and the magnetization depends on the saturation magnetization M_s and the spin polarization P of the current. The assumption of a magnetization pattern which rigidly gyrates holds true only for the small vortex core. Because of the spatial confinement, the remaining part of the vortex has to deform while the core is moving. D_Γ with $|D_\Gamma| < |D_0|$ is a phenomenological parameter that takes into account a reduced dissipation due to this deformation [23].

We will investigate a square thin-film element with a current flowing in x direction as shown in Fig. 1(a). This current is lateral homogeneous. The Oersted field accompanying the current consists of an in-plane component and an out-of-plane component. The out-of-plane component can be neglected as it does not change the equilibrium position of the vortex core. The in-plane field is negative at the top surface and positive at the bottom surface. It was verified by micromagnetic simulations that for a realistic strength this inhomogeneous Oersted field is not capable of significantly distorting the vortex. For a homogeneous current the average Oersted field vanishes and there will be no contribution of the Oersted field to the core displacement. However, such a contribution has been identified in experiment [12] and it is attributed to vertical inhomogeneities of the current density leading to an unbalanced in-plane Oersted field after taking the average over the thickness [23]. Here, we will approximate this unbalanced Oersted field by a homogenous field H in y direction while its precise shape and strength turned out to be of minor importance for the vortex dynamics. However, the force due to the Oersted field depends on the chirality. For small displacements of the vortex core from its equilibrium position, the demagnetization energy can be expanded up to second order in the core displacement $\vec{R} = (X, Y)$. The force on the vortex is then given by [15]

$$\vec{F} = -\begin{pmatrix} \mu_0 M_s H l d c + m\omega_r^2 X \\ m\omega_r^2 Y \end{pmatrix}, \quad (2)$$

with the lateral extension l, and thickness d of the system. The factor $m\omega_r^2$ parameterizes the confining potential [15].

For an excitation with a direct current [24], the core performs a damped gyration around a new equilibrium position [15,23]. By inserting Eq. (2) in Eq. (1) and setting $\vec{v}_c = 0$ we obtain the new equilibrium position

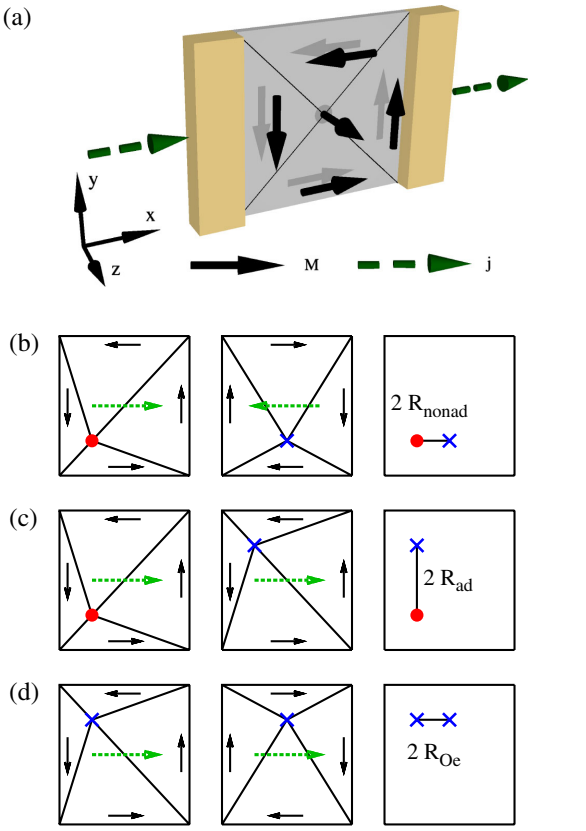


FIG. 1 (color online). (a) Sketch of the sample, including current contacts, for the proposed experiment for the determination of ξ. (b)–(d) Scheme for the determination of the three different contributions to the vortex displacement according to Eq. (4). By measuring the distance between the positions of two different vortices it is possible to separate the displacements (b) due to the nonadiabatic spin torque, (c) the adiabatic spin torque, and (d) the Oersted field. Points and crosses denote cores with positive and negative polarization, respectively. The in-plane magnetization is denoted by the solid arrows. The dashed arrows denote the current direction. For the sake of illustration the displacements are exaggerated.

$$\vec{R}_c^{\,p}(j) = -\frac{|G_0|}{m\omega_r^2}\begin{pmatrix} \tilde{H}c + |\frac{D_0}{G_0}|\xi\tilde{j} \\ \tilde{j}p \end{pmatrix} \quad (3)$$

with $\tilde{H} = \gamma H l/(2\pi)$, the gyromagnetic ratio γ, and $\tilde{j} = b_j j$.

From Eq. (3) it is obvious that an Oersted field has the same influence on the vortex as the nonadiabatic spin torque. Thus the presence of an Oersted field can disturb the measurement of ξ. In experiments the coordinate system is given by the sample axis. A small uncertainty of the direction of the current flow, e.g., due to a rotation or imperfections of the sample, yields a mixing of the displacement components, resulting from the adiabatic spin torque and the smaller nonadiabatic spin torque, relative to the sample axis. This mixing causes a large error in the measurement of the displacement originating from the nonadiabatic spin torque.

An excitation with a direct current causes a displacement of the vortex core to a new steady-state position. A benefit is that a direct current allows for a measurement with a non-time-resolving technique.

From Eq. (3) we find that the sign of the displacement induced by the Oersted field depends on the chirality of the vortex, while the displacement due to the adiabatic spin torque is determined by the polarization [20]. The nonadiabatic spin torque causes a displacement that is independent of the vortex properties p and c. Vortices with different p and c values can be achieved by demagnetizing the sample. Comparing the displacement of three vortices with different polarizations and chiralities it is therefore possible to separate the contributions of all three forces to the displacement of the vortex. From Eq. (3) we find

$$2R_{\mathrm{nonad}} = 2\left|\frac{G_0\xi\tilde{j}}{m\omega_r^2}\frac{D_0}{G_0}\right| = |\vec{R}_c^{\,p}(j) - \vec{R}_{-c}^{\,-p}(-j)| \quad (4a)$$

$$2R_{\mathrm{ad}} = 2\left|\frac{G_0\tilde{j}}{m\omega_r^2}\right| = |\vec{R}_c^{\,p}(j) - \vec{R}_c^{\,-p}(j)| \quad (4b)$$

$$2R_{\mathrm{Oe}} = 2\left|\frac{G_0\tilde{H}}{m\omega_r^2}\right| = |\vec{R}_c^{\,-p}(j) - \vec{R}_{-c}^{\,-p}(j)|. \quad (4c)$$

These equations are schematically illustrated in Fig. 1. From Eqs. (4a) and (4b) it is possible to determine ξ as

$$\xi = \frac{2R_{\mathrm{nonad}}}{2R_{\mathrm{ad}}}\left|\frac{G_0}{D_0}\right| = \frac{|\vec{R}_c^{\,p}(j) - \vec{R}_{-c}^{\,-p}(-j)|}{|\vec{R}_c^{\,p}(j) - \vec{R}_c^{\,-p}(j)|}\left|\frac{G_0}{D_0}\right|. \quad (5)$$

Since this equation is independent of the strength of the Oersted field, the angle of the sample, and the parameter D_Γ, it yields the sought measurement scheme. With this scheme a direct determination of ξ is accessible. Only one micromagnetic simulation for the determination of $|D_0/G_0|$ is necessary since $|D_0/G_0|$ is independent of ξ and j.

Micromagnetic simulations of the experimental setup allow us to determine the positions of the vortex core with a precise knowledge of the micromagnetic parameters of the system. The simulations therefore allow us to test the analytical results in Eqs. (3) and (5). For the simulations the material parameters of Permalloy, i.e., a saturation magnetization of $M_s = 8 \times 10^5$ A/m and an exchange constant of $A = 1.3 \times 10^{-11}$ J/m, are used. Since we are interested only in the steady final position of the vortex, we used a Gilbert damping of $\alpha = 0.5$ to ensure a fast damping of the transient states to reduce computation time. As a sample system we considered a square thin-film element of length $l = 500$ nm and thickness $d = 10$ nm with a cell size of 2 nm in the lateral directions and 10 nm perpendicular to the film. This system allows for a reasonable computation time. For the approximation of an infinitely large film we can estimate the in-plane Oersted field from Ampère's law $\vec{\nabla} \times \vec{H} = \vec{j}$ which yields $H(z) = (d - 2z)j/2$ with the aid of Stokes' theorem. Simulations with 1.25 nm cell size in z direction applying only the above in-plane field with j up to 2×10^{13} A/m^2 showed that it is a reasonable approximation that the magnetization is independent of the z coordinate. For the simulations, we used our extended version of the object oriented micromagnetic framework [25,26].

Figure 2 shows the displacement of the vortex core in simulations without the Oersted field. As predicted by Eq. (3) the displacement in the direction of the current flow is proportional to ξ and the displacement perpendicular to the current flow is independent of ξ. From these simulations the value $|D_0/G_0| = 2.26$ can be determined.

In experimental samples we are faced with an unbalanced Oersted field and possibly some uncertainty of the direction of the current flow. To mimic the unbalanced Oersted field in the simulations we applied an in-plane field perpendicular to the current. The strength of the field is proportional to the current density. We assume that a spin-polarized current density of 1×10^9 A/m^2 generates an unbalanced in-plane field of 1 A/m. For this field the ratio between the deflections due to the field and due to the current are in the regime found by experiments [12]. The uncertainty of the direction of the current flow was taken into account by rotating the sample by 5 degrees. Figure 3(a) shows the positions of the vortex core for both simulations. It becomes visible that the unbalanced Oersted field and the rotation of the sample strongly shift the core positions, complicating the determination of ξ.

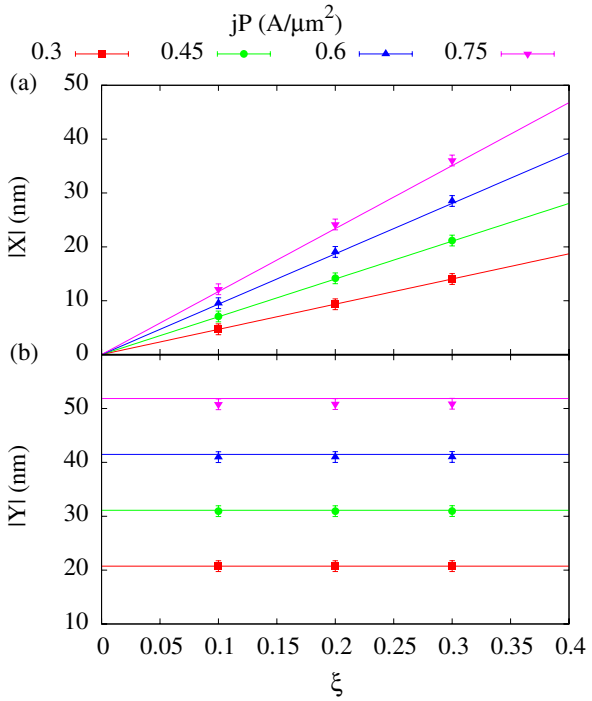


FIG. 2 (color online). Numerically calculated displacement of the vortex core due to a direct spin-polarized current of density jP in the absence of an Oersted field. (a) The displacement parallel to the current is proportional to ξ. (b) The displacement perpendicular to the current is independent of ξ. The lines are fits with the linear model in Eq. (3). For large current densities small nonlinear effects can be seen.

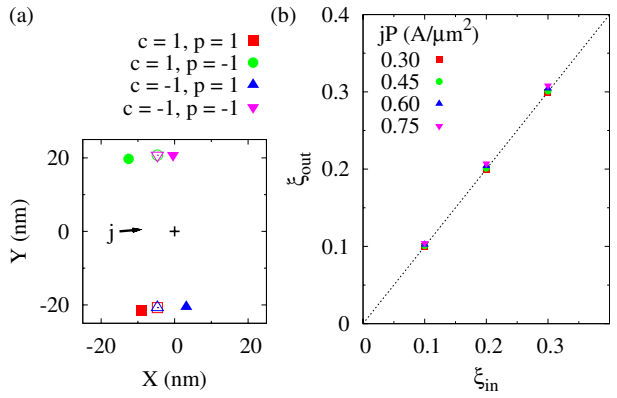


FIG. 3 (color online). (a) Position of the vortex core displaced by a spin-polarized direct current of density $jP = 3 \times 10^{11}$ A/m with $\xi = 0.1$. The overlapping open symbols denote the positions for a current in exact x direction without Oersted field. The closed symbols denote the positions with an applied Oersted field and a rotation of the sample by 5 degrees around its midpoint (plus). For the latter case the direction of the current is denoted by the arrow. (b) Results for ξ_{out} derived from the positions of the vortex with applied Oersted field, exemplarily shown in (a), using Eq. (5) for different current densities. ξ_{in} is the value of the nonadiabaticity parameter that was used for the simulations.

To test the analytical model we compared the nonadiabatic spin-torque parameter ξ_{in} that was inserted into the simulations with the value ξ_{out} that was calculated from Eq. (5) using the core positions. Here it is worth noting that the value of the Oersted field and the angle of the sample are not needed for the calculation of ξ_{out}. The results are shown in Fig. 3(b). It can be seen that all the perturbations that are inserted in the simulations can be effectively excluded by the analytical calculations.

In experimental samples we are also faced with the anisotropic magnetoresistance (AMR) effect that leads to inhomogeneous current paths, i.e., a higher current density in the vortex core. Simulations including these inhomogeneous current paths yield a small shift to lower values of ξ_{out}. This shift is up to 2% for an AMR ratio of 10%.

In the remaining part we will discuss the experimental accuracy in the determination of ξ that can be achieved with the presented scheme. In experiments direct currents of densities up to 1.5×10^{12} A/m^2 have been realized in Permalloy on a diamond substrate [27]. Assuming a spin polarization of 0.5 we get a spin-polarized current density of 0.75×10^{12} A/m^2, i.e., the maximum shown in Fig. 2. This yields values of up to $\tilde{j} = 55$ m/s.

The displacements of the vortex in the numerically investigated samples are small compared to the experimental resolutions available. A larger displacement of the vortex can be achieved by increasing the lateral size of the structure. For example, simulations of a square thin-film element of length $l = 5000$ nm and thickness $d = 10$ nm yielded values of $|D_0/G_0| = 3.8$ and $|G_0|/(m\omega_r^2) = 1 \times 10^{-8}$ s. With these values Eq. (4b) yields $2R_{\mathrm{ad}} = 1100$ nm. We assume that the core position can be measured with a resolution of $\delta(2R_{\mathrm{nonad}}) = 20$ nm. Equation (5) then yields that $\delta\xi = 0.005$ can be realized. This resolution ranges from 5% to 50% depending on the value of ξ [7–10]. The resolution can be further increased by using thin-film elements with still larger lateral sizes.

In conclusion we present a robust and direct measurement scheme for the nonadiabatic spin torque using the displacement of magnetic vortices. The scheme allows us to distinguish between the displacements of the vortex core due to the nonadiabatic spin torque, the adiabatic spin torque, and the Oersted field, independently of the exact direction of the current flow. We also showed that an inhomogeneous current due to the AMR effect can be neglected. The scheme thus allows a precise measurement of the nonadiabatic spin-torque parameter ξ.

Financial support by the Deutsche Forschungsgemeinschaft via SFB 668 "Magnetismus vom Einzelatom zur Nanostruktur" and via Graduiertenkolleg 1286 "Functional metal-semiconductor hybrid systems" is gratefully acknowledged.

[1] Y. B. Bazaliy *et al.*, Phys. Rev. B **57**, R3213 (1998).
[2] S. Zhang and Z. Li, Phys. Rev. Lett. **93**, 127204 (2004).
[3] H. Kohno *et al.*, J. Phys. Soc. Jpn. **75**, 113706 (2006).
[4] Y. Tserkovnyak *et al.*, Phys. Rev. B **74**, 144405 (2006).
[5] R. A. Duine *et al.*, Phys. Rev. B **75**, 214420 (2007).
[6] G. Tatara *et al.*, J. Phys. Soc. Jpn. **76**, 054707 (2007).
[7] M. Hayashi *et al.*, Phys. Rev. Lett. **96**, 197207 (2006).
[8] G. Meier *et al.*, Phys. Rev. Lett. **98**, 187202 (2007).
[9] L. Heyne *et al.*, Phys. Rev. Lett. **100**, 066603 (2008).
[10] L. Thomas *et al.*, Nature (London) **443**, 197 (2006).
[11] M. Najafi *et al.*, J. Appl. Phys. **105**, 113914 (2009).
[12] M. Bolte *et al.*, Phys. Rev. Lett. **100**, 176601 (2008).
[13] S. Kasai *et al.*, Phys. Rev. Lett. **101**, 237203 (2008).
[14] K. Yamada *et al.*, Nature Mater. **6**, 270 (2007).
[15] B. Krüger *et al.*, Phys. Rev. B **76**, 224426 (2007).
[16] B. Krüger *et al.*, J. Appl. Phys. **103**, 07A501 (2008).
[17] K. Y. Guslienko *et al.*, Phys. Rev. Lett. **96**, 067205 (2006).
[18] K.-S. Lee *et al.*, Appl. Phys. Lett. **92**, 192513 (2008).
[19] K.-S. Lee and S.-K. Kim, Phys. Rev. B **78**, 014405 (2008).
[20] J. Shibata *et al.*, Phys. Rev. B **73**, 020403(R) (2006).
[21] A. A. Thiele, Phys. Rev. Lett. **30**, 230 (1973).
[22] A. Thiaville *et al.*, Europhys. Lett. **69**, 990 (2005).
[23] See supplementary material at http://link.aps.org/supplemental/10.1103/PhysRevLett.104.077201.
[24] A measurement of the nonadiabatic spin torque with a resonant excitation using an alternating current is not suitable, as small deviations of the exciting frequency from the resonance frequency cause strong deviations in the trajectory of the vortex [23].
[25] OOMMF User's Guide, Version 1.0 M. J. Donahue and D. G. Porter Interagency Report NISTIR 6376, National Institute of Standards and Technology, Gaithersburg, MD (Sept. 1999) (http://math.nist.gov/oommf/).
[26] B. Krüger *et al.*, Phys. Rev. B **75**, 054421 (2007).
[27] S. Hankemeier *et al.*, Appl. Phys. Lett. **92**, 242503 (2008).

Chapter 5

Conclusion and Outlook

This work deals with the development of the finite-difference-method based micromagnetic simulator M^3S, that allows to investigate ferromagnetic systems effected by a current flow.

The first aspect that was in the focus for the development of M^3S was to see, if the use of a computational science integrated development environment (CSIDE) to develop a complex simulator really reduces the software complexity and thus increases its usability while maintaining a reasonable runtime performance. A reconstruction based on *OOMMF* led to the M^{3}S prototype *M^3S-MATLAB* that has been implemented using *MATLAB*-Script and test driven design (TDD). Using a well defined scripting language reduced the lines of code significantly and thus reduced the complexity of the simulator. Further several refactoring steps were necessary during the reconstruction and the later extension of the simulator, which would have been much more complicated to perform without the use of TDD. Hence, allthough not common for the development of scientific software, the use of TDD significanly simplified the reconstuction and the later extension of the simulator.

The development of *M^3S-MATLAB* revealed several restrictions of *MATLAB* as a CSIDE. Hence, the two additional prototypes *Nmag-FD* using *Python/SciTools* and *M^3S-Java* using *Java/JSA* have been developed. Comparing all three prototypes with the well-established micromagnetic simulator *OOMMF* revealed that using CSIDEs results in a more flexible solution. While *MATLAB* as a common CSIDE offered the best runtime performance for the simple algorithms, most of the runtime optimizations were not efficiently expressible. In contrast to *MATLAB*, *Python/SciTools* and *Java/JSA* offered the efficient implementation of the optimizations and due to their well designed languages and the range of tool support, these prototypes helped to achieve a better software quality. The use of novel libraries like the *FFTW* provided more flexibility as the simple FFT algorithm used in *OOMMF*. The new flexibility could be used to design an adaptive zero-padding strategy as new runtime optimization for the FFT based calculation of the demagnetization field. This new optimization results for the investigated range in a runtime performance increase of two at maximum.

The second aspect was the development of a micromagnetic simulator including current flow effects. For this investigation the micromagnetic prototype *M*^{3}*S-MATLAB* has been used. *M*^{3}*S-MATLAB* has been extended by theoretical descriptions of the corresponding physical phenomena, namely the spin-transfer torque and the magnetization-dependent current paths. Due to the use of scripting languages, opportunistic programming, and test driven design, the extensions could conveniently be included.

The validation of the extended tool has been addressed by proposing a new standard problem for the spin-transfer torque. The good properties of the proposed standard problem is shown by comparing the simulation results of different micromagnetic simulators with an experimentally validated analytical model. The extended simulator finally has also been used to design an experimental setup as a proposal for a robust measurement scheme for the degree of non-adiabaticity. This measurement scheme is robust against uncertainties of the exact current direction and Oersted fields.

In Conclusion the use of a CSIDE to develop scientific software is a competive approach to pure *C/C++* or *FORTRAN* solutions. Its efficiency depends significantly on the provided libraries and their support in the respective scripting language. For the development of a micromagnetic simulator this dependency was fulfilled and the development resulted in two simulator prototypes competitive to the performance of *OOMMF*, including the physical phenomena related to a current flow through a ferromagnetic system.

Outlook

Finite-difference-method (FDM) based micromagnetic simulators use the staircase method to represent the sample boundaries.[194] This method has a drawback in terms of the occurance of the so-called alias effect leading to nonphysical artifacts for non-rectangle sample geometries. Several works have addressed this problem and proposed corrections of the basic algorithms.[194,195] For a specific geometry however it is an open question how accurate a FDM-based simulation is compared to finite-element-method (FEM) based simulations. Including *Nmag-FD* into *Nmag* as an FDM extension provides the unique opportunity to design a tool that combines both discretization methods and allows to change the discretization method conveniently. This also allows the user to compare both methods and simplifies the reuse of the analysis functionality when switching between FDM and FEM.

Concerning the runtime performance different sequential optimizations can beincluded:

- Besides the fast-Fourier-transformation (FFT) implementation offered by *NumPy*, other *Python* interfaces for *FFTW* like *PyFFTW* could be included.
- Interfacing to novel ordinary differential equation (ODE) solvers, like *CVODE*[196] or *LSODA*[197] could further result in a reduction of the needed evaluations.

For the parallelization different approaches could be considered. The three-dimensional fast-convolution can in principle be parallelized efficently on symmetric multiprocessing architectures.[198] Thus advanced hardware architectures like graphical processing units (GPU)s[199] or field-progammable gate arrays (FPGA)s including digital-signal-processing (DSP) units are promising options.

A more conceptual approach is the use of the fast multipole method (FMM) for the calculation of the demagnetization field. In this method the demagnetization field is directly calculated in real space by an interpolation. The FMM has an asymptotic runtime complexity of $O(N)$,[200] which is comparable to the convolution-based calculation. The large benefit of FMM arises for the parallelization of the algorithm. The FFT needs the access to all other array elements within the data array to transform the data from real space to the Fourier space. Since the FMM is calculated in the real space its communication costs are sigificantly smaller as for the FFT. The calculation in real space in addition allows to combine the parallelization of the demagnetization field with the other field calculations and thus reduce the communication costs further.

Chapter 6

Appendix

Manuscript 1

Influence of Inhomogeneous Current Distributions on the Motion of Magnetic Vortices

S. Bohlens, B. Krüger, M. Najafi, and D. Pfannkuche

Influence of inhomogeneous current distributions on the motion of magnetic vortices

Stellan Bohlens,[1] Benjamin Krüger,[1] Massoud Najafi,[2] and Daniela Pfannkuche[1]

[1] *I. Institut für Theoretische Physik, Universität Hamburg, Jungiusstr. 9, 20355 Hamburg, Germany*
[2] *Arbeitsbereich Technische Informatik Systeme, Universität Hamburg, Vogt-Kölln-Str. 30, 22527 Hamburg, Germany*

(Dated: November 8, 2010)

The influence of inhomogeneous current paths on the gyroscopic motion of current-driven magnetic vortices in small thin-film elements is investigated by numerical simulations. It is found that the deflection of the gyrating vortex scales quadratically with the ratio of the anisotropic magnetoresistance. The enhancement of the gyration amplitude scales with the fundamental ratio between the dissipation tensor and the gyrovector and is determined by the lateral sample size and the sample thickness. The counteraction of the magnetization to the current manifests itself in a geometry-dependent renormalization of the spin transfer-torque coupling parameter.

PACS numbers: 75.60.Ch, 72.25.Ba

I. INTRODUCTION

Today's interest in spin-transfer torque phenomena can be traced back to its technological importance with the perspective of being the future magnetic technology. At the same time spin-transfer torque poses a theoretically appealing problem as it involves the interaction of non-equilibrium conduction electrons with the ferromagnetic order parameter, i.e., the magnetization. An understanding of the mutual interplay of both, current and magnetization, allows for a controlled manipulation of magnetization reversal and thus paves the path for current-controlled magnetic storage devices. Considering the mutual influence of electrical current and magnetization on equal footing provides the basis to a variety of fascinating non-linear spin-dependent phenomena. While the torque of a spin-polarized current influences the local magnetization[1,2], vice versa the magnetization influences the current flow via the anisotropic magnetoresistance (AMR).[3] The microscopic origin of the AMR is spin-orbit coupling.[4] Due to an asymmetric density of states the conduction electrons possess a larger scattering cross section for collinear alignment of conduction-electron spin and magnetization and consequently a smaller scattering cross section for transverse alignment. Classically spin-orbit coupling results in local resistance variations.[5] A transfer of spin-angular momentum from itinerant s-like conduction electrons to localized d electrons (spin-transfer torque) emerges in non-collinear magnetization patterns. It is accompanied by local resistance changes due to the AMR effect. An increase of the resistivity leads to a local reduction of the current density. This causes a locally reduced spin-transfer torque acting on the magnetization dynamics. In turn, the magnetization influences the local resistivity. As a result, the mutual influence of current and magnetization causes non-linear effects in the linear regime of electron transport.

Due to the non-collinearity, but high symmetry of its magnetization pattern and its quasiparticle-(soliton)-like behavior, the magnetic vortex in a micro- or nanostructured thin-film element is a prime example to study the interplay of electrical current and magnetization. Vortices are flux-closured states where the in-plane magnetization curls around a few nanometer large center region[6] to minimize the overall energy. Large angles between neighboring magnetic moments lead to a drastic increase of the exchange energy.[7] To overcome this situation the magnetization is forced out-of-plane forming the vortex core in the center of the thin-film element. In ferromagnetic square thin-film elements the vortex constitutes the energetic groundstate being fourfold degenerate due to the boolean vortex properties chirality and core polarization. Chirality and core polarization are topological quantities that characterize a vortex. A chirality of $+1$ (-1) denotes a counterclockwise (clockwise) curling of the magnetization around the vortex core while a polarization of $+1$ (-1) labels the out-of plane direction of the magnetization in the vortex core, up (down) respectively. Recent experiments showed that spin-polarized electric currents cause the vortex to precess.[8–11] Hitherto, analytical expressions as well as micromagnetic simulations confirming the elliptical gyration of vortex cores, take a homogeneous current flow into account neglecting the effect of inhomogeneous current paths occurring in real samples due to the AMR. The process of vortex-core switching is of fundamental interest and still an open question. Moreover it is of general interest, as vortex-core switching is the key ingredient in recent memory device proposals.[12,13] Thus for both, a detailed understanding of current-driven vortex dynamics and the purpose of technical utilization, it is crucial to consider realistic current paths.

In this article we investigate the current-driven gyroscopic motion of a magnetic vortex in square thin-film elements in the presence of an inhomogeneous current flow exemplarily depicted in Fig. 1. In the case of a homogeneous current the vortex gyration is topological in nature as the gyrotropic force that acts on the vortex and is responsible for its gyration solely depends on the vortex' polarization but is independent of the size of the vortex core.[14] We conclude that in the case of a vortex the non-linear effect of the counteraction of the magnetization on the current leads to an enhancement of the gyration

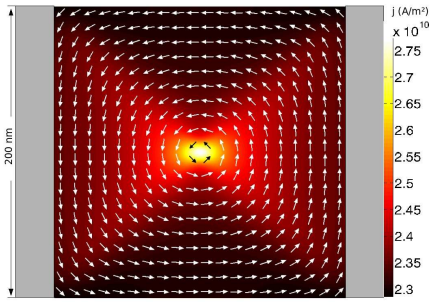


FIG. 1: (Color online) Inhomogeneous current distribution of a magnetic vortex in a $200 \times 200 \times 20\,\mathrm{nm}^3$ permalloy square. The arrows sketch the in-plane magnetization while the color (dark to bright) scales with the current density. The current flowing from left to right tends to flow through the vortex core. The gray areas indicate the non-magnetic ohmic contacts.

amplitude while it does not affect the quasiparticle like behavior of the vortex at all, e.g., no shape deformations are visible. As a consequence, the consideration of realistic current distributions leads to a geometry-dependent correction of the vortex' motion.

This article is organized as follows: In section II we explain how to consider inhomogeneous current paths due to non-collinear magnetization textures in the time-evolution of the magnetization. Section III investigates the gyroscopic motion of magnetic vortices and compares the homogeneous with the inhomogeneous case. Section IV yields a theoretical explanation of the simulated findings. Section V summarizes our findings of the amplitude enhancement in an analytical expression for the renormalized spin-transfer torque coupling parameter. Section VI attends to the highly non-linear regime of vortex-core switching. This article ends in Section VII with a conclusion.

II. NUMERICAL SIMULATIONS

In a continuous ferromagnet the influence of a spin-polarized current on the time-evolution of the magnetization $\vec{M}(\vec{r},t)$ is considered by the extended Landau-Lifshitz-Gilbert equation[15]

$$\begin{aligned}\frac{d\vec{M}(\vec{r},t)}{dt} =& -\gamma\vec{M}(\vec{r},t)\times\vec{H}_{\mathrm{eff}}(\vec{r},t)+\frac{\alpha}{M_s}\vec{M}(\vec{r},t)\times\frac{d\vec{M}(\vec{r},t)}{dt}\\ &-\frac{b_j}{M_s^2}\vec{M}(\vec{r},t)\times\Big(\vec{M}(\vec{r},t)\times(\vec{j}(\vec{r},t)\cdot\vec{\nabla}_{\vec{r}})\vec{M}(\vec{r},t)\Big)\\ &-\xi\frac{b_j}{M_s}\vec{M}(\vec{r},t)\times(\vec{j}(\vec{r},t)\cdot\vec{\nabla}_{\vec{r}})\vec{M}(\vec{r},t),\end{aligned} \tag{1}$$

where $b_j = P_j\mu_B/[eM_s(1+\xi^2)]$ is the coupling constant between current and magnetization, P is the absolute value of the spin polarization and M_S is the saturation magnetization. The terms containing the Gilbert-damping α and the degree of non-adiabaticity ξ are dissipative in the sense that they break the time-reversal symmetry of the LLG equation, i.e., they are odd under time-reversal transformation $t \to -t, \vec{H}_{\mathrm{eff}} \to -\vec{H}_{\mathrm{eff}}, \vec{j} \to -\vec{j}, \vec{M} \to -\vec{M}$.[16]

The electronic transport is treated classically and calculated quasi-statically from a local version of Ohm's law

$$\vec{j}(\vec{r}) = \sigma(\vec{r})\vec{E}(\vec{r}), \tag{2}$$

while local charge neutrality is considered, $\vec{\nabla}_{\vec{r}}\vec{j}(\vec{r}) = 0$. The influence of the magnetization on the current flow is incorporated in a magnetization-dependent conductivity tensor $\sigma(\vec{r}) = \sigma(\vec{M}(\vec{r}))$. The shape of the conductivity tensor accounts for the AMR, such that the resistivity locally obeys the relation

$$\rho = \rho_\perp + \Delta\rho\cos^2(\angle(\vec{j},\vec{M})), \tag{3}$$

which reflects the $\cos^2$-resistance dependence on the angle between local current and magnetization. The AMR ratio in thin-film elements

$$\rho_{\mathrm{AMR}} = \frac{\rho_{||}-\rho_\perp}{\rho_{||}+\rho_\perp} \equiv \frac{\Delta\rho}{\rho_{||}+\rho_\perp} \tag{4}$$

characterizes the strength of the AMR effect. The material parameters $\rho_{||}$ ($\rho_\perp$) are the resistances for the sample being saturated due to an external magnetic field parallel (perpendicular) to the current flow. Thus, the anisotropic magnetoresistivity $\Delta\rho$ is the change in resistance between a parallel and a perpendicular directed magnetization with respect to the applied current.

It follows from Eq. (3) that for non-collinear magnetization textures the magnetization influences the current via the anisotropic magnetoresistance by a spatially varying conductance. Figure 1 depicts the solution of the current density for a current passing a magnetic vortex structure in a permalloy square. The arrows sketch the in-plane magnetization of the vortex curling counterclockwise around the vortex core in the center. The sample dimensions are 200×200 nm^2 with a thickness of 20 nm. Dirichlet boundary conditions are imposed on the current biased probes (gray bars on the left and right hand side in Fig. 1) to fix the potential of the probes. Von Neumann boundary conditions ensure that no current leaves the sample through the upper or lower sample boundaries. Thus the current flows from left to right. The current favors the vortex core resulting in a higher local current density (bright color). In areas where the current is aligned perpendicular to the magnetization the conductivity is higher than in areas where the current is aligned parallel to the magnetization.

In the numerical simulations the mutual influence of current and magnetization is taken into account by gradually plugging the numerical result for the magnetization from Eq. (1) into the conductivity tensor of Eq. (2), calculating the current from Eq. (2) for the desired time-step Δt of Eq. (1), and iterating this procedure. The

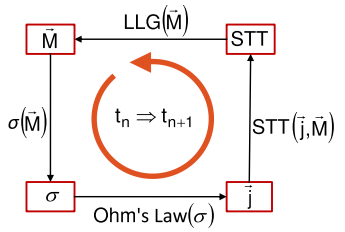


FIG. 2: (Color online) Self-consistency loop for the numerical computation of current-induced magnetization dynamics. The physical quantities in the boxes are solutions of the equations as denoted by the arrows. The anisotropic magnetoresistance is considered within a magnetization-dependent conductivity tensor $\sigma(\vec{M}(\vec{r}))$. The current paths $\vec{j}(\vec{r})$ are obtained from Ohm's law and are incorporated via the spin-transfer torque (STT) in the Landau-Lifshitz-Gilbert (LLG) equation.

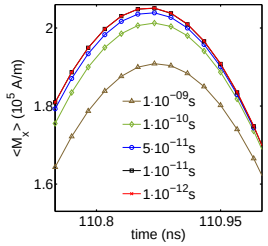


FIG. 3: (Color online) Mean x component of the magnetization of a magnetic vortex in a $200 \times 200 \times 20\,\mathrm{nm}^3$ permalloy square versus time. The different lines are the average x component of the magnetization belonging to the indicated timestep for the calculation of the current paths.

self-consistent calculation scheme for the counteraction of the magnetization on the current is illustrated in Fig. 2. The approach is justified because the band structure responsible for the electronic transport relaxes orders of magnitude faster ($\tau_{\mathrm{bs}} \approx 10^{-14}\,\mathrm{s}$) than the typical time scale of magnetization dynamics that is set by the Larmor frequency $\omega = \gamma M_s$ and is on the order of $\tau_{\vec{M}} \approx 10^{-11} - 10^{-12}\,\mathrm{s}$. There exist a separation of time scales in the fast electronic dynamics of the conduction electrons and the comparatively slow collective dynamics of the localized d electrons that constitute the magnetization.[17] From the viewpoint of the time-evolution of the magnetization the current flow is always in its steady state and can be computed quasi-statically by means of Eq. (2). The spin-transfer torque on the contrary is locally modulated by the inhomogeneous current density $\vec{j}(\vec{r})$ and acts on spatial inhomogeneities of the magnetization texture (cf. Eq. (1)). The local conductivity $\sigma(\vec{M}(\vec{r}))$ and thus the inhomogeneous current is determined by the magnetization itself and therefore varies on the time scale of magnetization dynamics. Thus, to capture the effect of the AMR on the vortex motion it is sufficient to compute the current paths on the time scale of vortex dynamics. Figure 3 depicts the mean x component of the magnetization of a gyrating vortex in its steady state. The sample dimensions are $200 \times 200\ \mathrm{nm}^2$ with a thickness of 20 nm and an AMR ratio $\rho_{\mathrm{AMR}} = 0.5$. As long as the time interval for a new current path calculation is below $\Delta t = 10^{-11}\,\mathrm{s}$ the result for the gyration amplitude is not affected and the physical results are independent of the unphysical time-interval for the current path calculation. This observation is in agreement with the Larmor frequency that takes for permalloy ($\mathrm{Py}=\mathrm{Ni}_{80}\mathrm{Fe}_{20}$) a value of $\omega_{\mathrm{Py}} = 1.77 \cdot 10^{11} s^{-1}$. Furthermore it is consistent with the adiabatic approximation that spin and charge currents are governed by the instantaneous magnetization that is implicitly assumed in the spin-transfer torque terms of Eq. (1).

In the case of harmonic excitations the vortex performs elliptical rotations.[18] At resonance the amplitude of the vortex core displacement in x and y direction is the same and the orbit is a circle. The ratio between the semi-axes is given by the ratio between the frequency of the excitation and the resonance frequency.[18] The sense of rotation of the vortex is controlled by its polarization, i.e., $p = +1$ ($p = -1$) causes a counterclockwise (clockwise) gyration of the vortex core around its equilibrium position. The analytic equation of motion for an applied homogeneous current in x direction reads for the quasiparticle coordinates of the vortex core[18]

$$\begin{pmatrix} \dot{X} \\ \dot{Y} \end{pmatrix} = \begin{pmatrix} -\Gamma & -p\omega \\ p\omega & -\Gamma \end{pmatrix} \begin{pmatrix} X \\ Y \end{pmatrix} + \begin{pmatrix} -b_j j - \frac{\Gamma^2}{\omega^2+\Gamma^2} \frac{\xi-\alpha}{\alpha} b_j j \\ \frac{p\omega\Gamma}{\omega^2+\Gamma^2} \frac{\xi-\alpha}{\alpha} b_j j \end{pmatrix}. \tag{5}$$

The free angular frequency $\omega = -pG_0 m\omega_r^2/(G_0^2 + D_0^2\alpha^2)$ and the damping constant $\Gamma = -D_0\alpha m\,\omega_r^2/(G_0^2 + D_0^2\alpha^2)$, as well as the constants G_0 of the gyrovector and D_0 of the dissipation tensor are defined in Ref. [18]. Figure 4 depicts the analytical steady-state trajectory of a vortex according to Eq. (5). The snapshots are the spatially resolved magnetization patterns and their corresponding current densities in the sample plane for four exemplary positions.

III. NUMERICAL RESULTS FOR COUPLED CURRENT AND MAGNETIZATION DYNAMICS

To investigate the influence of inhomogeneous current distributions on the magnetic vortex by means of the coupled Eq. (1) and (2), we conduct micromagnetic simulations. We perform simulations for magnetic thin-film elements with different lengths l and thicknesses t for various current densities and AMR values. In the following, the parameters of polarization and chirality are not varied. It follows from symmetry considerations that they do not influence the current flow in perfect square

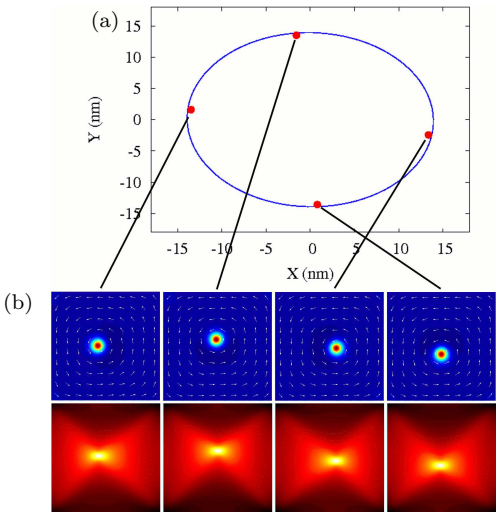


FIG. 4: (Color online) Steady-state trajectory of a current-driven magnetic vortex in a $200 \times 200 \times 20\,\mathrm{nm}^3$ permalloy square. (a) The line represents the analytical trajectory. The dots mark the positions of the vortex core that corresponds to the particular inset. (b) The insets depict the numerical results of the self-consistently calculated mutual current and magnetization dynamics. The upper row shows the spatially resolved magnetization where the arrows indicate the in-plane magnetization. The lower row displays the current density with the same scale as in Fig. 1.

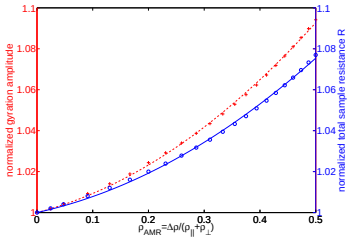


FIG. 5: (Color online) Enhancement of the gyration amplitude of a vortex due to the anisotropic magnetoresistance ratio (dashed red line) for a current density of $2.5 \cdot 10^{10}$ A/m^2 in a $200 \times 200 \times 20$ nm^3 permalloy square. Increase of the total sample resistance versus the AMR (solid blue line). The symbols denote the numerical results while the lines are quadratic fits.

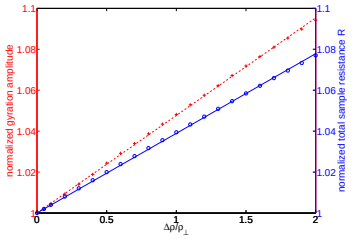


FIG. 6: (Color online) Enhancement of the gyration amplitude of a vortex due to the anisotropic magnetoresistivity (dashed red line) for a current density of $2.5 \cdot 10^{10}$ A/m^2 in a $200 \times 200 \times 20$ nm^3 permalloy square. Increase of the total sample resistance versus the normalized anisotropic magnetoresistivity (solid blue line). The symbols denote the numerical results while the lines are linear fits.

elements. We use the material parameters of permalloy, i.e., an exchange constant of $A = 13 \cdot 10^{-12}$ J/m and a saturation magnetization of $M_s = 8 \cdot 10^5$ A/m. For the Gilbert damping we assume a value of $\alpha = 0.01$, which is affirmed by recent experiments.[19–21] The degree of non-adiabaticity ξ is set to be equal to α.[22,23]

The simulation cells are chosen to be one cell of thickness t in z direction and 2 nm in x and y direction, which is well below the exchange length of permalloy. The position of the vortex is characterized by the maximum amplitude of the out-of-plane magnetization. It is determined by an interpolation with the Lagrange polynomial of second order of the respective simulation cell with maximum out-of-plane magnetization and its next neighbors.

To deduce the influence of inhomogeneous current paths on the vortex motion, alternating currents $P\vec{j}(\vec{r},t) = P\vec{j}(\vec{r})\cos\Omega t$ flowing spatially inhomogeneously in x direction are investigated. Even in simulations with idealized values of the AMR ratio ρ_{AMR} as high as 50% no deformation of the vortex structure is visible and no deviation from the quasiparticle behavior occurs. This suggests that the rigid particle model in Eq. (5) is sufficient to describe the vortex dynamics in the presence of inhomogeneous currents with a concomitant renormalization of the coupling parameters due to the counteraction of the magnetization by means of the AMR. To investigate the dependence of the gyration amplitude on the AMR ratio, we excite the magnetization in a $200 \times 200 \times 20$ nm^3 permalloy square for different current densities j at the vortex resonance frequency of 4.4 GHz in the vortex' gyrotropic mode. At about 100 ns the vortex gyration has reached its steady state and the amplitudes for different AMR ratios and current densities are compared. A variation of the AMR ratio is achieved by varying the parallel resistivity $\rho_{||}$ while fixing at the same time the perpendicular resistivity $\rho_\perp$.

The gyration amplitude depicted in Fig. 5 exhibits a quadratic amplitude enhancement with the AMR ratio and an offset of one (dashed red line). Similarly the total sample resistance R (solid blue line) increases quadratically. The mutual coupling of inhomogeneous current flow and magnetization dynamics leads to a non-linear response of the vortex motion and in terms of electron transport causes the vortex to act as a non-linear medium for the electric current. In the case of no AMR and a homogeneous current flow the gyration amplitude of the vortex scales with the current density.

However, instead of focusing on the AMR ratio, we decided to investigate the behavior of the gyration amplitude with the anisotropic resistivity $\Delta\rho$. Figure 6 depicts

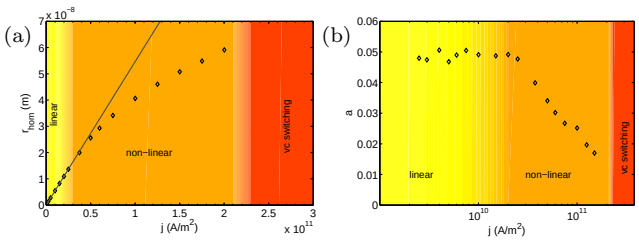


FIG. 7: (Color online) Enhancement of the gyration amplitude of the vortex in the steady state for a $200 \times 200 \times 20$ nm^3 permalloy square. (a) Radius enhancement versus current density for a homogeneous current flow. (b) The amplitude scaling a of Eq. (6) in dependence of the current density.

a linear increase of the gyration amplitude (dashed red line) as well as a concomitant linear increase of the total sample resistance R (solid blue line) with $\Delta\rho$

$$r_{\mathrm{AMR}} = \left(a\frac{\Delta\rho}{\rho_\perp} + 1\right) r_{\mathrm{hom}}, \tag{6}$$

where the free parameter a is the amplitude scaling and r_{hom} is the steady-state radius in the presence of a homogeneous current flow. Due to the inhomogeneous current flow an enhanced force acts on the vortex that causes a stronger deflection and an enhanced gyration amplitude compared to a homogeneous current.

Next, we investigate the enhancement of the gyration amplitude with respect to the applied current density. Figure 7 (a) depicts the steady-state radii for a homogeneous current flow in a $200 \times 200 \times 20$ nm^3 permalloy square. There exist three regimes of translational vortex motion. These regimes depend on the applied current density and thus on the deflection of the vortex core from its equilibrium position. The vortex can be regarded as a quasiparticle that moves in a restoring potential.[18] The restoring potential is caused by the demagnetization energy and the exchange energy due to the finite sample size and enhances with larger deflections of the vortex core from its equilibrium position. The linear regime with current densities of about $2.5 \cdot 10^9 - 2 \cdot 10^{10}$A/m^2 yields a linear increase of the steady-state amplitude with the applied current density. In the non-linear regime $2 \cdot 10^{10} - 2 \cdot 10^{11}$A/m^2 the amplitude increases in a sub-linear manner. Finally there exists the highly non-linear regime of vortex-core switching, which starts at approximately $2 \cdot 10^{11}$A/m^2 with no steady-state radius due to multiple vortex-core switching. Every regime is characterized by a different dependence of the vortex motion on the applied current density. In the linear regime of the vortex gyration, the vortex moves in a parabolic potential and the enhancement of the steady-state amplitude scales linearly with the applied current density (indicated by the line in Fig. 7 (a)). At higher current densities the enhancement flattens due to steeper non-linearities in the restoring potential.

Figure 7 (b) depicts the amplitude scaling a due to the AMR as determined by Eq. (6) with the applied current density in reference to a homogeneous current flow. A variation of the applied current density leaves the linear dependence of the anisotropic magnetoresistivity unaffected but alters its slope, the amplitude scaling a, as illustrated in Fig. 7 (b). In the linear regime of vortex motion we find an almost constant amplitude scaling independent of the applied current density. The harmonic potential does not affect the amplitude scaling and it attains a constant value. At about $2 \cdot 10^{10}$A/m^2 the vortex enters the non-linear regime of the vortex gyration and the amplitude scaling a decreases with increasing applied current density until the regime of vortex-core switching is reached (cf. Fig. 7 (b)). The decrease of the scaling is thus a direct consequence of the steeper confining potential: Due to a non-linear restoring force the amplitude scaling decreases along with the flattening of the amplitude enhancement in the non-linear regime of vortex motion. Besides the non-linear restoring force there is a second reason responsible for the decrease of the amplitude scaling. Micromagnetic simulations confirm a deformation of the vortex core in the non-linear regime of vortex motion due to the gyrotropic field[10,24]. More precisely the vortex core shrinks with increasing applied current density. A smaller vortex core in the presence of an inhomogeneous current flow results in a lower increase of the gyrotropic force on the vortex and thus in a lower scaling (cf. section IV for a detailed discussion). Note that the current dependence of $a(j)$ in the non-linear regime of the vortex gyration expresses directly the non-linear coupling of the current due to the counteraction of the AMR. These findings have an importance for experiments[25] and memory applications[13], since vortex-core switching depends critically on the radius of the vortex gyration.[26]

As with the current density, the geometry of the thin-film element affects the scaling of the gyration amplitude. To deduce the geometry dependence of a, we perform simulations on squares with various length l and thicknesses t. The value of the scaling a is the sole fit parameter and is thus a function of the applied current density and the sample geometry $a = a(j, l, t)$. Figure 8 (a) depicts a logarithmic geometry dependence of the scaling a for a current density of $2.5 \cdot 10^9$ A/m^2 and for sample lengths of $l = 200$, 300, 400 nm and thicknesses of $t = 10$, 20, 30 nm. Varying in turn the current density, the amplitude scaling always exhibits the functional behavior (cf. Fig. 8)

$$a(j, l, t) = \kappa(j, t) \log\left(\frac{\zeta(j, t)}{\sqrt[3]{L^2}} \frac{l}{\sqrt[3]{t}}\right), \tag{7}$$

where $\kappa(j, t)$ and $\zeta(j, t)$ are fit parameters and $L = \sqrt{2A/\mu_0 M_s^2}$ is the exchange length. The exchange length relates the exchange constant A to the saturation magnetization M_s and sets the relevant length scale in micromagnetism. While the parameter ζ is almost constant the run of κ with the current density is depicted in Fig. 9.

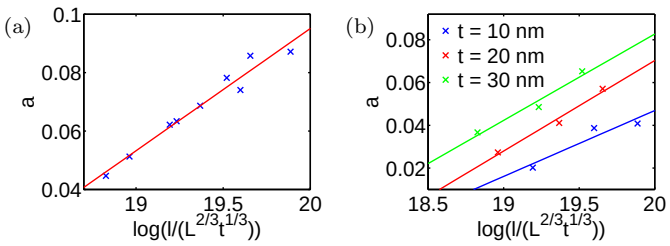


FIG. 8: (Color online) Geometry dependence of the amplitude scaling (a) in the linear regime of vortex motion for a current density of $2.5 \cdot 10^9$ A/m^2 and (b) in the non-linear regime for a current density of $7.5 \cdot 10^{10}$ A/m^2. In the non-linear regime the geometry dependence of the amplitude scaling holds for different sample thicknesses t individually.

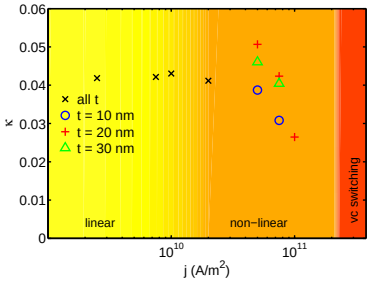


FIG. 9: (Color online) Dependence of the fit parameter κ defined in Eq. (7) on the applied current density for the linear and non-linear regime of vortex motion.

Analogously to the situation illustrated in Fig. 7 (b) we find two different reaction regimes. The linear regime of vortex motion yields a constant parameter κ that is independent of the applied current density and the sample geometry. In the non-linear regime of vortex motion $\kappa(j,t)$ is decreasing with the applied current density and according to Fig. 8 (b) depends moreover on the sample thickness t (cf. section IV for a detailed discussion).

In conclusion the transition in the vortex motion from the linear to the non-linear regime marks the transition from a linear transport regime with no explicit current dependence of $a(l,t)$ to a non-linear transport regime with $a(j,l,t)$ depending now explicitly on the current density. The logarithm of the ratio $l/\sqrt[3]{t}$ is proportional to the ratio of the constants belonging to the dissipation tensor and the gyrovector $D_0/G_0 \propto \log(l/\sqrt[3]{t})$ (cf. Ref. [26]). The ratio of dissipation tensor and gyrovector is in turn proportional to the ratio of damping Γ and the free frequency ω: $D_0/G_0 \propto \Gamma/\omega$.[18] Thus the geometric dependence in Eq. (7) is linked to characteristic quantities of the current-driven vortex.

IV. THEORETICAL EXPLANATION

In this section we give a theoretical explanation why inhomogeneous current paths affect the gyration amplitude of the current-driven vortex. As confirmed by micromagnetic simulations, the vortex keeps its static structure and no deviation from the particle-like behavior occurs when excited with a spin-polarized current. Therefore, the static motion still can be described by the Thiele equation[14] with the expansion by Nakatani et al.[27] to include the action of a spin-polarized current

$$\vec{F} + \vec{G} \times (\vec{v} + b_j \vec{j}) + D(\alpha \vec{v} + \xi b_j \vec{j}) = 0. \qquad (8)$$

Here, $\vec{F}$ is the restoring force due to the demagnetization and exchange fields that stems from the effective field, D is the diagonal dissipation tensor and $\vec{G}$ is the gyrovector. Besides the gyrotropic force the gyrovector constitutes the driving force due to the current of Eq. (8), while the dissipation tensor resembles the loss of energy occurring in magnetic systems, which is referred to damping of the electron system. Note the two distinct origins of dissipation, the first term in the expression of the dissipation tensor of Eq. (8) is the usual Gilbert damping of the localized d electrons, while the second term describes spin relaxation of the itinerant s electrons parametrized by the degree of non-adiabaticity ξ.[15] The magnetization is a vector field of uniform length that can be expressed in dependence of two coordinates: for the vortex the polar angle θ changes in radial direction and the azimuthal angle ϕ characterizes the curling in-plane magnetization. Equation (8) represents an already integrated version of the Thiele equation that assumes no spatial dependence either of the velocity $\vec{v}$ nor of the current $\vec{j}$. Considering realistic current paths this assumption clearly does not hold and we have to consider the full integral Thiele equation[18]

$$\begin{aligned} &-\mu_0 \int dV \left[(\vec{\nabla}\theta)\frac{\partial}{\partial \theta} + (\vec{\nabla}\phi)\frac{\partial}{\partial \phi} \right] (H_{\text{eff}} \cdot \vec{M}) \\ &-\frac{M_s \mu_0}{\gamma} \int dV \sin(\theta)(\vec{\nabla}\theta \times \vec{\nabla}\phi) \times (\vec{v} + b_j \vec{j}(\vec{r})) \\ &-\frac{M_s \mu_0}{\gamma} \int dV \, (\vec{\nabla}\theta\vec{\nabla}\theta + \sin^2(\theta)\vec{\nabla}\phi\vec{\nabla}\phi)(\alpha \vec{v} + \xi b_j \vec{j}(\vec{r})) \\ &= 0. \end{aligned} \qquad (9)$$

However, the simulations presented in section III indicate that a description of vortex motion in terms of collective coordinates by an integrated version of the Thiele equation still offers a good description for the case of inhomogeneous current paths. The employment of the integrated version of the Thiele equation is possible with a proper renormalization of one of the coupling parameters in Eq. (8). In a first approximation of homogeneous current paths, the vortex motion is independent of the size of the vortex core and thus considered to be of topological nature.[14] A spatial dependence of the current in the integrands of Eq. (9) requires corrections compared

with the homogeneous case. As addressed in Ref. [28] the velocity in Eq. (8) must be modified to match with detailed micromagnetic simulations. For the case of a vortex confined in a thin-film element the rigid particle approximation is only approximatively fulfilled as the velocity within the vortex core is different compared to the velocity in the domains. There is no general rule how to treat modifications of the quasiparticle picture. In order to modify Eq. (8) as little as possible and to maintain a quasi-linear structure of the Thiele equation with respect to the current density, we decide to attribute the renormalization to the spin-transfer torque coupling parameter b_j whose derivation has been performed for a homogeneous current flow.[15] This approach is motivated by the following considerations. The gyrotropic force that arises due to the adiabatic current term (cf. Eq. (9)) reads for the case of a magnetic vortex[18]

$$\begin{aligned}\vec{G}\times\tilde{b}_j\vec{j} &= -\frac{M_s\mu_0}{\gamma}\int dV\ \sin(\theta)(\vec{\nabla}\theta\times\vec{\nabla}\phi)\times b_j\vec{j}(\vec{r})\\ &= -\frac{2\pi M_s\mu_0 p}{\gamma}t\,\vec{e}_z\times\tilde{b}_j\vec{j}\\ &= \tilde{b}_j G_0\vec{e}_z\times\vec{j}. \end{aligned}\tag{10}$$

Except for the small area of the vortex core, θ is almost constant and thus $\vec{\nabla}\theta$ in the integrand of Eq. (10) vanishes. This restricts the integration to the region of the vortex core. Though defined as an integral over the whole sample the gyrovector is primarily located at the vortex core. Due to the spatial integration the renormalized spin-transfer torque coupling can be expected to depend on the set of all possible parameters $\tilde{b}_j = \tilde{b}_j(j,\rho_{||},\rho_\perp,l,t)$. If we rearrange the modified version of Eq. (8) as follows

$$\begin{aligned}G_0^2\vec{v} &\approx (G_0^2+D_0^2\alpha^2)\vec{v}\\ &= \vec{G}\times\vec{F} - D_0\alpha\vec{F} - (G_0^2+D_0^2\alpha\xi)\tilde{b}_j\vec{j}\\ &\quad +\tilde{b}_j D_0\vec{G}\times\vec{j}(\xi-\alpha)\\ &\approx \vec{G}\times\vec{F} - D_0\alpha\vec{F} - G_0^2\tilde{b}_j\vec{j}, \end{aligned}\tag{11}$$

we deduce that the driving part proportional to the current $\tilde{b}_j\vec{j}$ is primarily given by the square of the gyrovector, where, as usually, we have assumed $\alpha,\xi \ll 1$. The influence of the cross product term in Eq. (11) can be disregarded, since we employed $\alpha \approx \xi$ in the simulations.[22] Note that in contrast to the gyrovector the dissipation tensor

$$D = -\frac{M_s\mu_0}{\gamma}\int dV\,(\vec{\nabla}\theta\vec{\nabla}\theta + \sin^2(\theta)\vec{\nabla}\phi\vec{\nabla}\phi),\tag{12}$$

attains its contributions mainly in the domains due to the change in the second term by the in-plane angle ϕ, while the contribution from the vortex core is small. It is little affected by the current flow as it contributes to the driving force via the non-adiabatic spin-transfer torque and is thus suppressed by factors of $\alpha\xi$, α^2 and $D_0/G_0(\xi-\alpha)$ (cf. Eqns. (11)).

To summarize, in the case of current excitations the driving force acts on the vortex core, while the energy dissipation mainly takes place in the domains of the Landau pattern as expressed by the second term on the right hand side of Eq. (11). These circumstances can also be directly understood from the LLG Eq. (1). The spin-transfer torque is proportional to the spatial derivative of the magnetization, hence the spin transfer-torque contribution is located in the center region while its influence is negligible in the almost uniform domains. In discs the rotational symmetry does not allow internal domain walls and the vortex exhibits similar behavior.[29] Thus, the contribution to the spin-transfer torque of the four Néel walls is small. This reveals a striking difference between inhomogeneous current and magnetic field excitations. While inhomogeneous magnetic fields cause deformations of the vortex structure, the electrical current mainly affects the vortex core and the vortex structure is kept stable, even in the case of a strong inhomogeneous current flow. This contrasts with alternating, homogeneous field and current excitations that result for the vortex in similar magnetization dynamics.

Taking now the AMR effect into account the current tends to flow through the vortex core resulting in a locally higher current density compared with the homogeneous case. The occurrence of the locally higher current density in the vortex core coincides with the location of the gyrovector that constitutes according to Eq. (11) the driving force. An enhanced gyrotropic force acts on the vortex and a bigger amplitude results for the vortex gyration compared with a homogeneous current flow. The stability of the vortex during the motion must be addressed to the high symmetry of the vortex pattern, such that internal stresses compensate each other and the magnetic configuration as a whole is not affected.

As mentioned in the context of Eq. (10), in the case of inhomogeneous current paths the geometry of the thin-film element influences the coupling parameter $\tilde{b}_j$ and thus the amplitude scaling a. The numerical simulations in Fig. 8 exhibit for the amplitude scaling a a logarithmic geometry dependence proportional to the ratio of dissipation tensor and gyrovector: $D_0/G_0 \propto (\log l - const.\cdot\log t)$. Owing to the integration over the sample in the expression for the gyrovector (cf. Eq. (10)), the lateral size of the sample gains its importance for the vortex motion due to the inhomogeneity of the current flow. In the preceding section we have determined the exact geometry dependence from micromagnetic simulations. In samples with a larger sample length l the driving force is bigger resulting in an enhanced gyration amplitude ($\tilde{b}_j \propto \log l$). At the same time the amplitude scaling a increases with decreasing sample thickness t ($\tilde{b}_j \propto \log 1/t$). The connection of the increase in the gyration amplitude with decreasing sample thickness t is exemplarily depicted in Fig. 10 for a fixed sample length of $l = 300$ nm and a current density of $j = 2.5\cdot10^{10}\mathrm{A/m^2}$. For smaller t a higher gyrotropic force acts on the vortex caused by the AMR effect.

As discussed, it is the vortex core that controls the dynamic behavior of the vortex state in the case of excita-

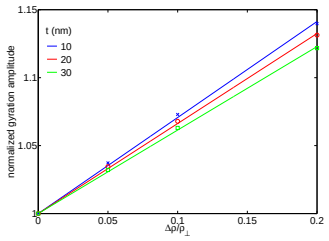


FIG. 10: (Color online) Comparison of the enhancement slope for a sample size of $l = 300$ nm and three different thicknesses $t = 10, 20, 30$ nm for a current density of $j = 2.5 \cdot 10^{10}\mathrm{A/m}^2$.

tion due to a spin-polarized electric current. With the particular role the vortex core takes in current-driven vortex dynamics, the origin of the decrease of the factor $\kappa(j,t)$ in the non-linear regime of vortex motion as depicted in Fig. 9 becomes comprehensible. The vortex core shrinks with increasing applied current density due to the non-linear restoring potential experienced by the vortex caused by larger displacements from the equilibrium position. To obtain the same amplitude scaling in the presence of the non-linear potential as compared to the linear case, the local current density within the core would have to become even more inhomogeneous than in the linear regime of vortex motion. As a consequence, the gyrotropic force on the vortex and thus $\kappa(j,t)$ decreases. In addition, the vortex reaches with smaller sample thickness t the non-linear regime for lower current densities or deflections from its equilibrium position. For small aspect ratios $t/l \ll 1$ the frequency of the vortex is approximately proportional to the aspect ratio itself $\omega \propto t/l$.[30] In turn, the vortex displacement is inversely proportional to the aspect ratio $r \propto l/t$. This means that the non-linearities set in earlier with lower sample thickness t due to a larger displacement of the vortex. A change in the sample thickness t affects the shape of the non-linear potential. The consequence is the occurring thickness dependence of $\kappa(j,t)$ in the non-linear regime, while the sample length l plays a minor role.

The observations of section III are a constant amplitude scaling κ in the linear regime of small deflections of the vortex core independent of the applied current density. In the non-linear regime $\kappa(j,t)$ decreases with higher current densities as a direct consequence of the non-linear potential felt by the vortex.

V. RENORMALIZATION OF THE SPIN-TRANSFER TORQUE COUPLING PARAMETER

The counteraction of the magnetization by means of the AMR results for the current-driven vortex in a geometry-dependent renormalization of the spin-transfer torque coupling parameter that can be interpreted as a correction to the entirely topological motion of vortices in the presence of a homogeneous current flow. As discussed in the preceding sections, considering the influence of inhomogeneous current paths on the gyrotropic motion of a magnetic vortex modifies the spin-transfer torque coupling parameter b_j. With respect to a description of vortex motion in terms of collective coordinates, $\tilde{b}_j j$ acts as a renormalized velocity due to the current in the equations of motion (5) according to

$$\tilde{b}_j(j,\rho_{||},\rho_\perp,l,t) = \left(a(j,l,t)\frac{\Delta\rho}{\rho_\perp}+1\right)b_j, \qquad (13)$$

$$a(j,l,t) = \kappa(j,t)\log(\frac{\zeta(j,t)}{\sqrt[3]{L^2}}\frac{l}{\sqrt[3]{t}}). \qquad (14)$$

The renormalization involves a dependence on the geometry, the electric current and on the parameters that characterize the AMR effect: $\tilde{b}_j(j,\rho_{||},\rho_\perp,l,t)$. Note that the explicit current dependence of $\tilde{b}_j$ in the non-linear regime of vortex motion expresses the non-linear coupling of current and magnetization.

For small deflections in the linear regime of vortex motion the correction due to the AMR effect is small and the quasiparticle approximation remains applicable. The equations of motion keep their shape and maintain their validity as effective equations of motion comprising the counteraction of the magnetic vortex on the electric current via the AMR effect. For higher deflections, in particular in the regime of vortex-core switching (cf. next section), the counteraction of the AMR leads to non-linear effects that have to be identified by detailed self-consistent micromagnetic simulations.

VI. INFLUENCE OF THE ANISOTROPIC MAGNETORESISTANCE ON THE HIGHLY NON-LINEAR REGIME OF VORTEX-CORE SWITCHING

If the vortex gyration exceeds a critical velocity (≈ 320 m/s for Py), the highly non-linear regime of vortex-core switching is entered.[10,24] The vortex-core switching is accompanied by a halo formation – a region with opposite oriented out-of-plane magnetization is formed close to the vortex – and subsequent vortex-antivortex nucleation and annihilation.[24] Due to the non-trivial topology of the combined vortex-antivortex state it is crucial to consider realistic current paths. The gyrotropic field responsible for the vortex-core distortion and the subsequent core-reversal at higher gyration amplitudes forms a dip with out-of-plane magnetization in the inside of the vortex' orbit.[24] An exemplary current density is depicted in Fig. 11 that reveals the complexity of the current paths in the regime of vortex-core switching as a direct consequence of the complex distorted magnetization texture. Thus far, we have considered the steady-state radius of the vortex. Let us now turn the attention to the time-domain. A question of experimental and applicational relevance is the time between excitation of the vortex and its switching. Figure 12 depicts the time required until the vortex

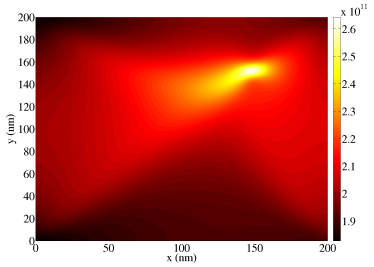


FIG. 11: (Color online) Current density of a magnetic vortex in a $200\times200\times20\,\mathrm{nm}^3$ permalloy square at the critical velocity 320 m/s for vortex-core switching.

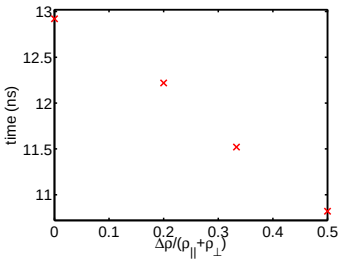


FIG. 12: (Color online) Time until a critical velocity of 320 m/s is reached for a vortex in a $200\times200\times20\,\mathrm{nm}^3$ permalloy square in dependence of the AMR ratio.

reached its critical velocity for switching with respect to the AMR ratio. The particular point in time in Fig. 12 corresponds to the critical velocity (320 m/s relates to a radius of 72.8 nm at a frequency of 4.4 GHz) that was found to be the universal criterion for vortex-core switching.[24] A higher AMR ratio linearly reduces the time until vortex-core switching sets in.

VII. CONCLUSION

In conclusion the counteraction of the magnetization on the current-driven magnetic vortex results in a geometry-dependent renormalization of the spin-transfer torque coupling parameter by means of the anisotropic magnetoresistivity. This can be interpreted as a correction to the topological motion of vortices in the presence of a homogeneous current flow. The renormalized coupling parameter depends on the ratio of the dissipation tensor and gyrovector that constitute intrinsic vortex' properties that are determined by the geometry of the thin-film element, namely its size and its thickness. In the non-linear regime of vortex motion the change in the shape of the vortex core introduces explicitly a non-linear dependence of the renormalized spin-transfer coupling parameter on the current density. The results are obtained by micromagnetic simulations taken the spin-transfer torque as well as the inhomogeneity of the current flow into account. Incorporating the counteraction of the magnetization onto the current flow provides a non-linear coupling of mutual current and magnetization dynamics. For experimental and technical implications we identified the AMR as a candidate to reduce the time until the critical velocity for vortex-core switching is reached.

Acknowledgments

We thank Ulrich Merkt and André Drews for valuable discussions. Financial support by the Deutsche Forschungsgemeinschaft via SFB 668 "Magnetismus vom Einzelatom zur Nanostruktur" and via Graduiertenkolleg 1286 "Functional metal-semiconductor hybrid systems" as well as from the Free and Hanseatic City of Hamburg in the context of the Landesexzellenzinitiative Hamburg "Exzellenzcluster NANO-SPINTRONICS" is gratefully acknowledged

[1] L. Berger, Phys. Rev. B **54**, 9353 (1996).
[2] J. Slonczewski, J. Magn. Magn. Mater. **159**, L1 (1996).
[3] W. Thomson, Proc. R. Soc. London **8**, 546 (1857).
[4] R. I. Potter, Phys. Rev. B **10**, 4626 (1974).
[5] T. R. McGuire and R. I. Potter, IEEE Trans. Magn. **11**, 1018 (1975).
[6] A. Wachowiak, J. Wiebe, M. Bode, O. Pietzsch, M. Morgenstern, and R. Wiesendanger, Science **298**, 577 (2002).
[7] T. Shinjo, T. Okuno, R. Hassdorf, K. Shigeto, and T. Ono, Science **289**, 930 (2000).
[8] J. Shibata, Y. Nakatani, G. Tatara, H. Kohno, and Y. Otani, Phys. Rev. B **73**, 020403(R) (2006).
[9] S. Kasai, Y. Nakatani, K. Kobayashi, H. Kohno, and T. Ono, Phys. Rev. Lett. **97**, 107204 (2006).
[10] K. Yamada, S. Kasai, Y. Nakatani, K. Kobayashi, H. Kohno, A. Thiaville, and T. Ono, Nature Materials **6**, 270 (2007).
[11] S. K. Kim, Y. S. Choi, K. S. Lee, K. Y. Guslienko, and D. E. Jeong, Appl. Phys. Lett. **91**, 082506 (2007).
[12] S.-K. Kim, K.-S. Lee, Y.-S. Yu, and Y.-S. Choi, Appl. Phys. Lett. **92**, 022509 (2008).
[13] S. Bohlens, B. Krüger, A. Drews, M. Bolte, G. Meier, and D. Pfannkuche, Applied Physics Letters **93**, 142508 (2008).
[14] A. A. Thiele, J. Appl. Phys. **45**, 377 (1974).
[15] S. Zhang and Z. Li, Phys. Rev. Lett. **93**, 127204 (2004).
[16] Y. Tserkovnyak, A. Brataas, and G. E. W. Bauer, J. Magn.

Magn. Mater. **320**, 1282 (2008).
[17] J.-i. Ohe and B. Kramer, Phys. Rev. Lett. **96**, 027204 (2006).
[18] B. Krüger, A. Drews, M. Bolte, U. Merkt, D. Pfannkuche, and G. Meier, Phys. Rev. B **76**, 224426 (2007).
[19] J. Nibarger, R. Lopusnik, and T. Silva, Appl. Phys. Lett. **82**, 2112 (2003).
[20] M. Schneider, T. Gerrits, A. Kos, and T. Silva, Appl. Phys. Lett. **87**, 072509 (2005).
[21] Z. Liu, F. Giesen, X. Zhu, R. D. Sydora, and M. R. Freeman, Phys. Rev. Lett. **98**, 087201 (2007).
[22] G. Meier, M. Bolte, R. Eiselt, B. Krüger, D.-H. Kim, and P. Fischer, Phys. Rev. Lett. **98**, 187202 (2007).
[23] M. Hayashi, L. Thomas, Y. B. Bazaliy, C. Rettner, R. Moriya, X. Jiang, and S. S. P. Parkin, Phys. Rev. Lett. **96**, 197207 (2006).
[24] K. Y. Guslienko, K.-S. Lee, and S.-K. Kim, Phys. Rev. Lett. **100**, 027203 (2008).
[25] M. Bolte, G. Meier, B. Krüger, A. Drews, R. Eiselt, L. Bocklage, S. Bohlens, T. Tyliszczak, A. Vansteenkiste, B. Van Waeyenberge, et al., Phys. Rev. Lett. **100**, 176601 (2008).
[26] K. Y. Guslienko, Appl. Phys. Lett. **89**, 022510 (2006).
[27] A. Thiaville, Y. Nakatani, J. Miltat, and Y. Suzuki, Europhys. Lett. **69**, 990 (2005).
[28] B. Krüger, M. Najafi, S. Bohlens, R. Frömter, D. P. F. Möller, and D. Pfannkuche, Phys. Rev. Lett. **104**, 077201 (2010).
[29] B. Krüger, A. Drews, M. Bolte, U. Merkt, D. Pfannkuche, and G. Meier, J. Appl. Phys. **103**, 07A501 (2008).
[30] K. Y. Guslienko, B. A. Ivanov, V. Novosad, Y. Otani, H. Shima, and K. Fukamichi, J. Appl. Phys. **91**, 8037 (2002).

Supporting material for publication PRL' 10

Supporting Material for: Proposal of a Robust Measurement Scheme for the Nonadiabatic Spin Torque Using the Displacement of Magnetic Vortices

B. Krüger, M. Najafi, S. Bohlens,
R. Frömter, D. P. F. Möller, and D. Pfannkuche

Phys. Rev. Lett. **104**, 077201, 2010

Supporting Material for: Proposal of a Robust Measurement Scheme for the Nonadiabatic Spin Torque Using the Displacement of Magnetic Vortices

Benjamin Krüger,[1] Massoud Najafi,[2] Stellan Bohlens,[1] Robert Frömter,[3] Dietmar P. F. Möller,[2] and Daniela Pfannkuche[1]

[1]*I. Institut für Theoretische Physik, Universität Hamburg, Jungiusstr. 9, 20355 Hamburg, Germany*
[2]*Arbeitsbereich Technische Informatik Systeme, Universität Hamburg, Vogt-Kölln-Str. 30, 22527 Hamburg, Germany,*
[3]*Institut für Angewandte Physik, Universität Hamburg, Jungiusstr. 11, 20355 Hamburg, Germany*

MODIFIED THIELE EQUATION

For an analytical investigation the motion of the vortex is commonly described employing the Thiele equation.[1–8] This equation is exact for the steady state motion of a non-deformable magnetization pattern. However, this assumption holds true only for the small vortex core. Due to the spacial restriction the magnetization pattern outside the core has to deform while the core is moving, as illustrated in Fig. 1. This yields a small modification of the Thiele equation that is especially important for current-driven vortex motion in view of the nonadiabatic spin torque.

Here we present a modified Thiele equation that takes a deformation of the outer part of the vortex into account.

With the magnetization $\vec{M}$ and the magnetic field $\vec{H}$ a general version of the Thiele equation reads [1]

$$\begin{aligned} 0 = &-\mu_0 \int dV \left[(\vec{\nabla}\theta)\frac{\partial}{\partial\theta} + (\vec{\nabla}\phi)\frac{\partial}{\partial\phi}\right](\vec{H}\cdot\vec{M}) \\ &- \frac{M_s\mu_0}{\gamma}\int dV\ \sin(\theta)(\vec{\nabla}\theta\times\vec{\nabla}\phi)\times(\vec{v}+b_j\vec{j}) \\ &- \frac{M_s\mu_0}{\gamma}\int dV\,(\vec{\nabla}\theta\vec{\nabla}\theta+\sin^2(\theta)\vec{\nabla}\phi\vec{\nabla}\phi)(\alpha\vec{v}+\xi b_j\vec{j}), \end{aligned} \tag{1}$$

with the saturation magnetization M_s, the gyromagnetic ratio γ, the current density $\vec{j}$, the Gilbert damping α, the nonadiabaticity parameter ξ, and the coupling constant b_j between current and magnetization. θ and ϕ are the out-of-plane and in-plane angle of the magnetization, respectively. The velocity $\vec{v} = \vec{v}(r)$ of the magnetization pattern may depend on the position. Assuming a rigid magnetization pattern the velocity is independent of the position. Then Eq. (1) can be written in its well known form [9]

$$\vec{F} + \vec{G}\times(\vec{v}_c + b_j\vec{j}) + D(\alpha\vec{v}_c + \xi b_j\vec{j}) = 0, \tag{2}$$

with the velocity $\vec{v}_c$ of the vortex core. Here

$$\vec{F} = -\mu_0\int dV\left[(\vec{\nabla}\theta)\frac{\partial}{\partial\theta} + (\vec{\nabla}\phi)\frac{\partial}{\partial\phi}\right](\vec{H}\cdot\vec{M}) \tag{3}$$

denotes the force on the magnetization pattern.

$$\vec{G} = -\frac{M_s\mu_0}{\gamma}\int dV\ \sin(\theta)(\vec{\nabla}\theta\times\vec{\nabla}\phi) = G_0\vec{e}_z \tag{4}$$

is the gyrovector and

$$D = -\frac{M_s\mu_0}{\gamma}\int dV\,(\vec{\nabla}\theta\vec{\nabla}\theta+\sin^2(\theta)\vec{\nabla}\phi\vec{\nabla}\phi) \tag{5}$$

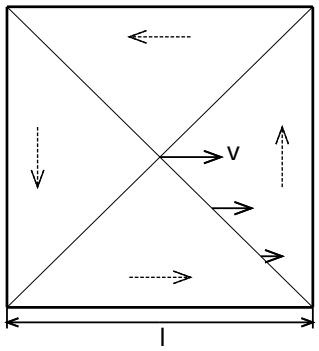


FIG. 1: Scheme of the magnetization (dashed arrows) in a square magnetic thin-film element with a vortex. The solid arrows denotes the velocity v of the vortex core and of different points within the domain wall.

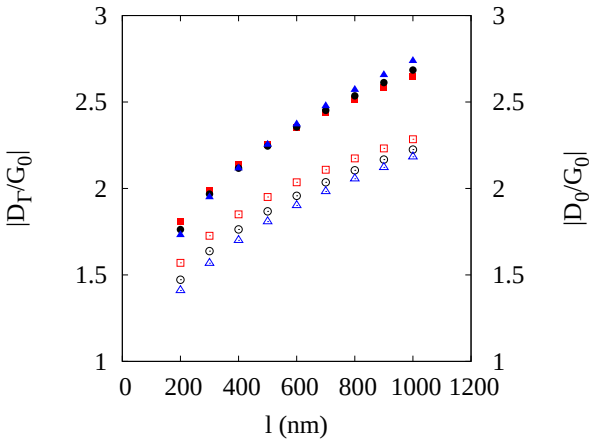


FIG. 2: Values of the strength D_Γ of the dissipation (open symbols) and the strength D_0 of the nonadiabatic spin torque (closed symbols). The data for films of 10 nm, 20 nm, and 30 nm is denoted by squares, circles, and triangles, respectively.

is the diagonal dissipation tensor with $D_{xx} = D_{yy} = D_0$ and $D_{zz} = 0$. The term $D\alpha\vec{v}_c$ in Eq. (2) describes the dissipation of energy due to the changing magnetization.

The integrand in the gyrovector is nonzero only in the small vortex core where the out-of-plane angle θ varies while the integrand in the dissipation tensor is also nonzero outside the core. Close to the boundaries of the sample the magnetization pattern moves slower compared to the center as it can be seen in Fig. 1. Thus the velocity in the third term of Eq. (1) depends on space. Aiming at a similar form as in Eq. (2) we replace the

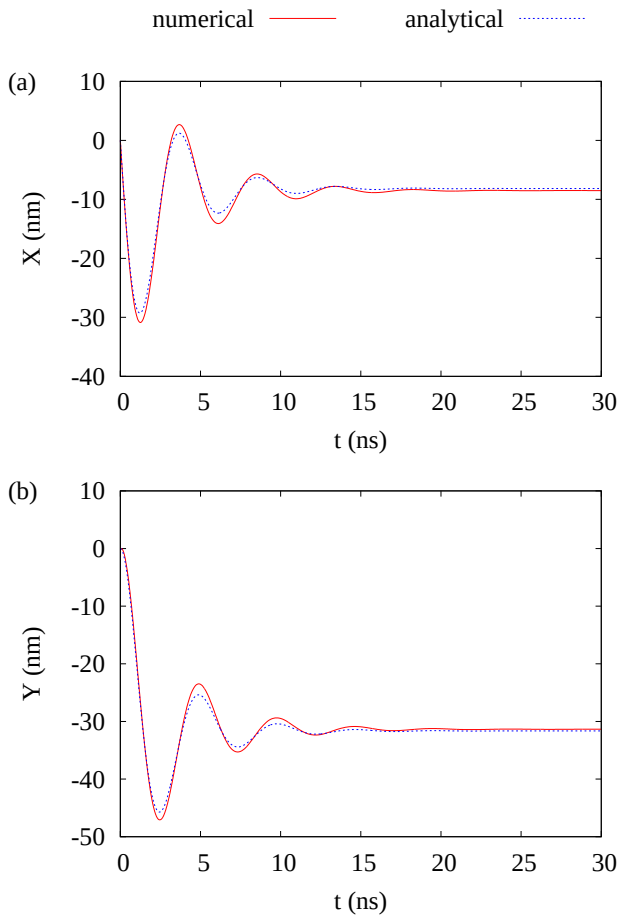


FIG. 3: Calculated position of a vortex core excited with a spin-polarized direct current of density $jP = 6 \cdot 10^{11}$ A/m^2 in a 1000 nm x 1000 nm square thin-film element. Shown is the x position (a) and the y position (b) versus time. A film thickness of 30 nm and $\xi = \alpha = 0.1$ was used. The solid (red) line is the vortex core position extracted from simulations. The dashed (blue) line is a fit with the theory based on the original Thiele equation.

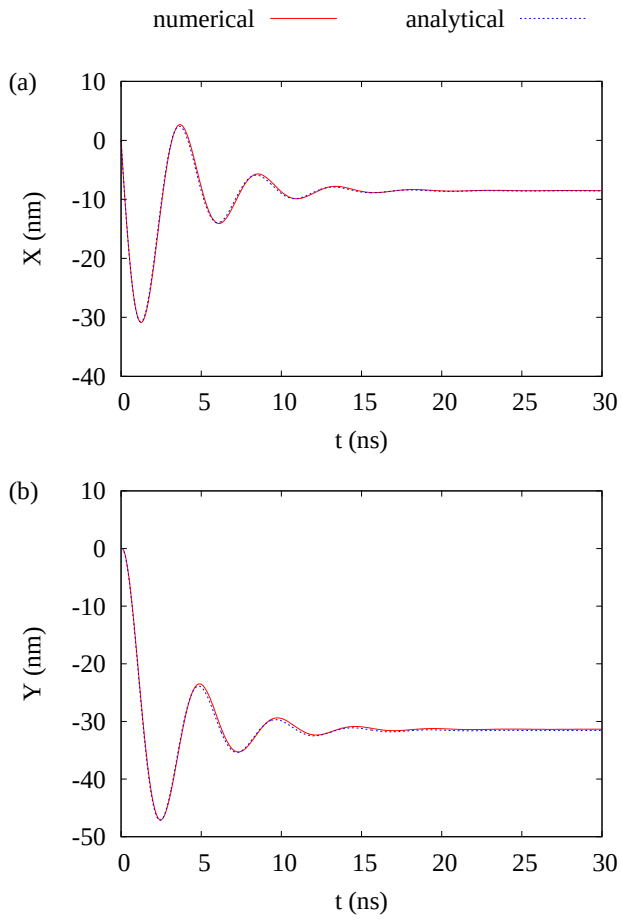


FIG. 4: Calculated position of a vortex core. The solid (red) line is the same as shown in Fig. 3. The dashed (blue) line is a fit with the theory based on the modified Thiele equation.

spatially dependent velocity $\vec{v}$ in the third term of Eq. (1) by an effective value $\vec{v}_e$ which is independent of the position. This effective velocity occurs only in the third term as the second term is located at the vortex core. For a homogeneous current flow $b_j\vec{j}$ is constant over the sample. Thus we do not replace the current by an effective value. The equation then reads

$$\vec{F} + \vec{G} \times (\vec{v}_c + b_j\vec{j}) + D_0\alpha\vec{v}_e + D_0\xi b_j\vec{j} = 0. \qquad (6)$$

The effective velocity $\vec{v}_e$ depends on the core position $\vec{R} = (X, Y)$ and the core velocity $\vec{v}_c$. For small deflections of the vortex core, i.e., small deformations of the vortex, $\vec{v}_e$ can be expanded in $\vec{R}$ and $\vec{v}_c$. For $\vec{v}_c = 0$ the magnetization is static and $\vec{v}_e = 0$. Thus the first nonvanishing term in the expansion is proportional to $\vec{v}_c$. Here and hereafter we write

$$\vec{v}_e = \frac{D_\Gamma}{D_0}\vec{v}_c. \qquad (7)$$

Since the effective velocity $\vec{v}_e$ is always smaller than the velocity $\vec{v}_c$ of the vortex core $D_\Gamma/D_0 < 1$. Inserting Eq. (7) in Eq. (6) yields a modified Thiele equation

$$\vec{F} + \vec{G} \times (\vec{v}_c + b_j\vec{j}) + D_\Gamma\alpha\vec{v}_c + D_0\xi b_j\vec{j} = 0. \qquad (8)$$

Employing the same conversions as used for the original Thiele equation [5] we find an expression for the velocity of the vortex core

$$(G_0^2 + D_\Gamma^2\alpha^2)\vec{v}_c = \vec{G} \times \vec{F} - D_\Gamma\alpha\vec{F} - (G_0^2 + D_\Gamma D_0\alpha\xi)b_j\vec{j} + b_j\xi D_0\vec{G} \times \vec{j} - b_j\alpha D_\Gamma\vec{G} \times \vec{j}. \qquad (9)$$

We investigate a square thin-film element with a current in x and a magnetic field in y direction. With the stray-field energy for small deflections [5]

$$E_s = \frac{1}{2}m\omega_r^2(X^2 + Y^2) \qquad (10)$$

and the total Zeeman energy [5]

$$E_z = \mu_0 M_s H l d c X \qquad (11)$$

we get a force of

$$\vec{F} = -\begin{pmatrix} \mu_0 M_s H l d c + m\omega_r^2 X \\ m\omega_r^2 Y \end{pmatrix}. \qquad (12)$$

Here l and d are the lateral extension and the thickness of the square, respectively. c is the chirality of the vortex. As for the original Thiele equation, in the absence of current and field the excited vortex performs an exponentially damped spiral rotation around its equilibrium position. The free frequency

$$\omega = -\frac{pG_0 m\omega_r^2}{G_0^2 + D_\Gamma^2\alpha^2} \tag{13}$$

and the damping constant

$$\Gamma = -\frac{D_\Gamma \alpha m\omega_r^2}{G_0^2 + D_\Gamma^2\alpha^2} \tag{14}$$

are slightly changed compared to their values derived from the homogeneous Thiele equation. Here p denotes the polarization of the vortex. In the following we express

$$D_\Gamma = \frac{\Gamma p G_0}{\omega\alpha} \tag{15}$$

by the frequency and the damping constant. The velocity of the vortex then reads

$$\begin{pmatrix}\dot{X}\\ \dot{Y}\end{pmatrix} = \begin{pmatrix}-\Gamma & -p\omega\\ p\omega & -\Gamma\end{pmatrix}\begin{pmatrix}X\\ Y\end{pmatrix} + \begin{pmatrix} \frac{p\omega\Gamma}{\omega^2+\Gamma^2}\frac{\mu_0 M_s H l d c}{G_0} - \frac{\omega^2}{\omega^2+\Gamma^2} b_j j - \frac{\Gamma\omega}{\omega^2+\Gamma^2}\left|\frac{D_0}{G_0}\right|\xi b_j j \\ -\frac{\omega^2}{\omega^2+\Gamma^2}\frac{\mu_0 M_s H l d c}{G_0} - \frac{p\omega\Gamma}{\omega^2+\Gamma^2} b_j j + \frac{p\omega^2}{\omega^2+\Gamma^2}\left|\frac{D_0}{G_0}\right|\xi b_j j \end{pmatrix}. \tag{16}$$

This equation can be solved for harmonic excitations of the form $\vec{H}(t) = H_0 e^{i\Omega t}\vec{e}_y$ and $\vec{j}(t) = j_0 e^{i\Omega t}\vec{e}_x$. The solution for the vortex motion is then given by [5]

$$\begin{pmatrix}X\\ Y\end{pmatrix} = A\begin{pmatrix}i\\ p\end{pmatrix}e^{-\Gamma t + i\omega t} + B\begin{pmatrix}-i\\ p\end{pmatrix}e^{-\Gamma t - i\omega t} - \frac{e^{i\Omega t}}{\omega^2 + (i\Omega+\Gamma)^2}\begin{pmatrix}\tilde{j} & \tilde{H}cp + \left|\frac{D_0}{G_0}\right| p\xi\tilde{j}\\ -\tilde{H}cp - \left|\frac{D_0}{G_0}\right| p\xi\tilde{j} & \tilde{j}\end{pmatrix}\begin{pmatrix}\frac{\omega^2}{\omega^2+\Gamma^2} i\Omega\\ \omega p + \frac{\omega\Gamma}{\omega^2+\Gamma^2} i\Omega p\end{pmatrix}, \tag{17}$$

with $\tilde{H} = \gamma H_0 l/(2\pi)$ and $\tilde{j} = b_j j_0$. The first two terms with prefactors A and B are exponentially damped and depend on the starting configuration.

The values of D_Γ and D_0 can be determined by micromagnetic simulations. For these simulations we used our extended version of the Object Oriented Micromagnetic Framework (OOMMF) that includes the adiabatic and nonadiabatic spin torque.[10–12] The position of the vortex core was defined as the point with the maximum out-of-plane magnetization. To determine this maximum, the simulation cell with maximum out-of-plane magnetization and its next neighbors are interpolated with a polynomial of second order. For the simulations the material parameters of Permalloy, i.e., a saturation magnetization of $M_s = 8\cdot 10^5$ A/m and an exchange constant of $A = 1.3\cdot 10^{-11}$ J/m were used.

For the determination of D_Γ the vortex was excited by a magnetic field pulse. The subsequent oscillation was then fitted with the first two terms in Eq. (17). D_Γ can then be determined from Eq. (15). Finally the value of D_0 was determined by fitting an excitation with a direct current. The results are shown in Fig. 2 for different edge lengths l and different thicknesses of the sample. It can be clearly seen that $|D_\Gamma|$ is smaller than $|D_0|$.

Figures 3 and 4 show an example for the fit of a numerically calculated vortex-core trajectory using both theories. The theory based on the modified Thiele equation shows better accordance than the theory based on the original Thiele equation. It can be seen that the Thiele equation has to be modified for a sufficient description of the dynamics of current-driven magnetic vortices in the presence of a nonadiabatic spin torque. This modification takes the deformation of the outer part of the vortex into account.

UNBALANCED OERSTED FIELD

In real samples we have to consider several mechanisms that lead to an inhomogeneous current flow and concomitantly to an unbalanced Oersted field. Here we will discuss four examples.

A first mechanism leading to an inhomogeneous current flow is that to ensure a sufficient electric contact the sample and the contacts have to overlap each other. If the specific resistivity of the sample is large compared to the contacts the current tends to flow in the sample from its top surface.[13, 14] Thus the way through the high-ohmic sample is shortest for a current flowing along the top surface. This leads to an inhomogeneous current flow with a higher current density in the upper part of the sample.

For high current densities Joule heating has to be taken into account. Theoretical considerations [15] as well as experimental results [16, 17] show that a major part of the heat is dissipated through the substrate. Consequently there is a temperature gradient in the sample where the top surface is hotter

than the bottom surface. Thus the specific resistivity deviates. This results in an inhomogeneous current flow depending on the temperature coefficient of the specific resistivity of the sample material.

Furthermore, finite-size effects are important. For thin-film elements with a thickness that is comparable with the mean free path of the conduction electrons, scattering at the surfaces becomes important. In the Fuchs-Sondheimer theory the surfaces of the film are described by a parameter p_s that denotes the probability that an electron is reflected specularly at the surface.[18] This theory was expanded by Mayadas and Shatzkes for polycrystalline films.[19] For a value of $p_s = 1$ the current is the same as for a bulk material. For $p_s < 1$ the current becomes smaller at the surfaces. If the value p_s is the same for both surfaces the suppression is symmetric and thus do not lead to an unbalanced Oersted field. For experimental samples the bottom surface is a border between two solids while the upper surface is normally a boundary to a gas or vacuum. This gives rise to the assumption that the probability of specularly reflection is different for both surfaces leading to an asymmetric current flow and therefore to an unbalanced Oersted field.

Finally the current flow can also be influenced by an inhomogeneous growth of the sample material and a concomitant inhomogeneity in the resistance.

In experiments a gyration driven by an unbalanced Oersted field has been observed for vortices that are excited with an alternating current.[13] The experimental results can be explained by a homogeneous field in y direction that is proportional to the current flowing in x direction.

[1] A. A. Thiele, Phys. Rev. Lett. **30**, 230 (1973).

[2] K. Y. Guslienko, B. A. Ivanov, V. Novosad, Y. Otani, H. Shima, and K. Fukamichi, J. Appl. Phys. **91**, 8037 (2002).

[3] K. Y. Guslienko, X. F. Han, D. J. Keavney, R. Divan, and S. D. Bader, Phys. Rev. Lett. **96**, 067205 (2006).

[4] K. Y. Guslienko, Appl. Phys. Lett. **89**, 022510 (2006).

[5] B. Krüger, A. Drews, M. Bolte, U. Merkt, D. Pfannkuche, and G. Meier, Phys. Rev. B **76**, 224426 (2007).

[6] K.-S. Lee, Y.-S. Yu, Y.-S. Choi, D.-E. Jeong, and S.-K. Kim, Appl. Phys. Lett. **92**, 192513 (2008).

[7] K.-S. Lee and S.-K. Kim, Phys. Rev. B **78**, 014405 (2008).

[8] S. Kasai, P. Fischer, M.-Y. Im, K. Yamada, Y. Nakatani, K. Kobayashi, H. Kohno, and T. Ono, Phys. Rev. Lett. **101**, 237203 (2008).

[9] A. Thiaville, Y. Nakatani, J. Miltat, and Y. Suzuki, Europhys. Lett. **69**, 990 (2005).

[10] S. Zhang and Z. Li, Phys. Rev. Lett. **93**, 127204 (2004).

[11] **OOMMF User's Guide, Version 1.0** M.J. Donahue and D.G. Porter Interagency Report **NISTIR 6376**, National Institute of Standards and Technology, Gaithersburg, MD (Sept 1999) (http://math.nist.gov/oommf/).

[12] B. Krüger, D. Pfannkuche, M. Bolte, G. Meier, and U. Merkt, Phys. Rev. B **75**, 054421 (2007).

[13] M. Bolte, G. Meier, B. Krüger, A. Drews, R. Eiselt, L. Bocklage, S. Bohlens, T. Tyliszczak, A. Vansteenkiste, B. Van Waeyenberge, K. W. Chou, A. Puzic, and H. Stoll, Phys. Rev. Lett. **100**, 176601 (2008).

[14] L. Bocklage, B. Krüger, R. Eiselt, M. Bolte, P. Fischer, and G. Meier, Phys. Rev. B **78**, 180405 (2008).

[15] C.-Y. You, I. M. Sung, and B.-K. Joe, Appl. Phys. Lett. **89**, 222513 (2006).

[16] S. Hankemeier, K. Sachse, Y. Stark, R. Frömter, and H. P. Oepen, Appl. Phys. Lett. **92**, 242503 (2008).

[17] F. Junginger, M. Kläui, D. Backes, U. Rüdiger, T. Kasama, R. E. Dunin-Borkowski, L. J. Heyderman, C. A. F. Vaz, and J. A. C. Bland, Appl. Phys. Lett. **90**, 132506 (2007).

[18] E. H. Sondheimer, Advances in Phys., **1**, 1 (1952).

[19] A. F. Mayadas and M. Shatzkes, Phys. Rev. B **1**, 1382 (1970).

Acknowledgement

None of my teachers at my old grammar school would have expected me to finish a Ph. D. thesis. At this point I thank all people that helped me on this way.

I gratefully thank especially

- Professor Dr.-Ing. Dietmar Möller for his optimism and for encouraging me to this Ph. D. project. Your guidance as my supervisor helped me a lot to stay on track. It was always an event visiting a conference with you. Thank you further for supplying Bernd and me with lots of sweets during our meetings.
- Privatdozent Dr. Guido Meier for supervising me in my Ph. D. project. Your comments helped me a lot to improve the quality of my thesis. Thanks also for reviewing this thesis and for pushing me on during the last year.
- Professor Dr. Ulrich Merkt and Dr. Katrin Buth for their support as the speaker and the coordinator of the Graduiertenkolleg "Functional Metal-Semiconductor Hybrid Systems". The lessons I learned from the Monthly Highlights helped me a lot to write this thesis. I am sure that the structures provided in this Graduiertenkolleg are well-wrought.
- Dr. Hans Fangohr, Dr. Matteo Franchin, Andreas Knittel, and Dr. Thomas Fischbacher from the Computational Pyhsics Group at the University of Southampton, United Kingdom. Thanks for the fruitful cooperation for the proposal of the standard problem and for facilitating a great reasearch stay during the summer of 2009 for me.
- Bernd Güde for being my colleague in the Graduiertenkolleg during the last three years and one of my dearest friends. You held the fort when I left and you always had a sympathetic ear, whenever I needed it. Thanks for the nice time during the study and the Ph. D. project and for forming such a good "tag team" with me.
- Stellan Bohlens for being a colleague and a friend. Thank you for proofreading my thesis. It was a pleasure to program and to play table soccer together with you. I hope you will never have to handle broken simulation runs in the future again.
- Benjamin Krüger for explaining the physics of ferromagnets, and for the patience, when I needed longer to understand it.

- Jeanette Wulfhorst for being a colleague and a friend. Thanks for organizing so many events of the group N during the last five years. Thanks also for the discussions about the GMR effect and for proofreading my thesis. I also congratulate you and Andreas to Julian-Max.

- Bodo Krause-Kyora for maintaining the computer system. Thank you for setting up the software-tool infrastucture that we needed in the simulation group. It was a pleasure to discuss with you the current trends in the high performance computing community.

- Dr. Markus Bolte for his support, especially in the beginning of my Ph. D. project. Thank you for discussing the micromagnetic simulation in detail during my diploma thesis and for the conceptual discussions about possible extensions of M^3S.

- the members of the group „Quantentheorie der kondensierten Materie“ especially Prof. Dr. Daniela Pfannkuche, Daniel Becker, Evi Richter, Dr. Philipp Knake, Dr. Jacek Swiebodzinski, Dr. Peter Moraczewski, and Dr. Dirk-Sören Lühmann for affiliating me in your group. It was always a pleasure to watch the “largest salad competition” at lunch time.

- Gunnar Selke for working together with me during his diploma thesis.

- Claas Abert for developing the simulator prototype “yamms” during his diploma thesis. Thank you for the discussions about software engineering concepts.

- Carola Tenge for being the kind soul of “TIS”. Thank you for the endless organizational support, especially during the last year.

- Petra Roth for supporting me in many organizational matters at the department of physics. Thanks also for inviting to pleasent tea breaks.

- Jan Michels for proofreading my thesis and for improving the atmosphere in Bernd’s and my office whenever he came over for a tea.

- the Deutsche Forschungsgemeinschaft for funding my scholarship in the Graduiertenkolleg.

- my friends. Thank you all for sustaining me during the last three years. Especially I thank Andreas Soltau, Bernd Eggink, Marcus Mesch, and Oliver Flöreke for many discussions about the spirit and purpose of a Ph. D. thesis.

- my girlfriend Birte Reichow for many things: backing me up and bearing me during this project, sustaining almost endless discussions on technical details of the thesis, and persistently repeating the sentence “Einfach mal fertig machen”.

- the members of Birte’s family, especially Hilde and Hanne for “spuring” me to finish the thesis.

- my family for believing in me. Thank you for inspiring me for natural sciences and for reinforcing my decision to start a Ph. D. project. Without your trust in my capabilities and your support I had never reached this point.

List of own publications

Journal article

- B. Krüger, M. Najafi, S. Bohlens, R. Frömter, D. P. F. Möller, and D. Pfannkuche, "Proposal of a Robust Measurement Scheme for the Nonadiabatic Spin Torque Using the Displacement of Magnetic Vortices", in *Physical Review Letter*, **104**, 2010, pp. 077201.

- M. Najafi, B. Krüger, S. Bohlens, M. Franchin, H. Fangohr, A. Vanhaverbeke, R. Allenspach, B. Güde, M. Bolte, U. Merkt, D. Pfannkuche, D. P. F. Möller, and G. Meier "Proposal for a Standard Problem for Micromagnetic Simulations Including Spin-Transfer Torque", in *Journal of Applied Physics*, **105**, 2009, pp. 113914.

Conference proceedings

- M. Najafi, B. Krüger, S. Bohlens, G. Selke, B. Güde, M. Bolte, and D. P. F. Möller, "The Micromagnetic Modeling and Simulation Kit M^3S for the Simulation of the Dynamic Response of Ferromagnets to Electric Currents", in *Proceedings of the 2008 Grand Challenges in Modeling and Simulation Conference (GCSM'08)*, H. Vakilzadian, R. Huntsinger, T. Ericson, and R. Crosbie, Eds. San Diego, CA, USA: The Society for Modeling and Simulation, 2008, pp. 427-434.

- B. Güde, M. A. B. W. Bolte, B. Krüger, M. Najafi, and D. P. F. Möller, "Spin Valves for Innovative Computing Devices and Architectures", in *Proceedings of the 2008 Summer Computer Simulation Conference (SCSC'08)*, D. Cook and K. Taylor, Eds. San Diago, CA, USA: The Society for Modeling and Simulation, 2008, pp. 279-285.

- M. Najafi, G. Selke, B. Krüger, B. Güde, B. Krause-Kyora, M. Bolte, G. Meier, and D. P. F. Möller, "A Case Study for the Parallelization of a Complex MATLAB Program with Respect to Maintainability", in *Proceedings of the Huntsville Simulation Conference (HSC'08)*, J. Gauthier, Ed. San Diego, CA, USA: The Society for Modeling and Simulation, 2008, pp. 309-315.

- M. A. B. W. Bolte, M. Najafi, G. Meier, and D. P. F. Möller, "Simulating Magnetic Storage Elements: Implementation of the Micromagnetic Model into MATLAB - Case Study for

Standardizing Simulation Environments", in *Proceedings of the 2007 summer computer simulation conference(SCSC'07)*, 525-532, G. A. Wainer, Eds. San Diego, CA, USA: The Society for Modeling and Simulation, 2007, pp. 525-532.

- M. A. B. W. Bolte, G. Meier, M. Najafi, and D. P. F. Möller, "Computation of Spin-Wave Spectra of Magnetic Nanostructures for Information Storage Systems", in *Proceedings of the 20th european conference on modeling and simulation (ECMS'06)*, 2006, pp. SE-136-1 - SE-136-6.

Invited talks

- M. Najafi, Vortex Symposium, 04.02.2010, University of Hamburg, Standard Problems in Micromagnetic Simulations

Research stays

- M. Najafi, University of Southampton, 3 Juli 2009–29 September 2009, Southampton, UK, Research in the computational physics group of Hans Fangohr with the topic: "Reimplementation of *M*^{3}*S-MATLAB* in *Python* and Integration into *Nmag* as *Nmag-FD* Module"

Bibliography

[1] D. A. Thompson and J. S. Best, "The Future of Magnetic Data Storage Techology," *IBM Journal of Research & Development*, vol. 44, p. 311, 2000.

[2] "International Technology Roadmap for Semiconductors 2007 Edition," November 2010. [Online]. Available: http://www.itrs.net/Links/2007ITRS/Home2007.htm

[3] "International Technology Roadmap for Semiconductors 2009 Edition," November 2010. [Online]. Available: http://www.itrs.net/links/2009ITRS/Home2009.htm

[4] T. M. Maffitt, J. K. DeBrosse, J. A. Gabric, E. T. Gow, M. C. Lamorey, J. S. Parenteau, D. R. Willmott, M. A. Wood, and W. J. Gallagher, "Design Considerations for MRAM," *IBM Journal of Research & Development*, vol. 50, p. 25, 2006.

[5] E. C. Stoner and E. P. Wohlfarth, "Mechanism of Magnetic Hysteresis in Heterogeneous Alloys," *Philosophical Transation of the Royal Society of London*, vol. A240, p. 599, 1948.

[6] C. Chappert, A. Fert, and F. N. Van Dau, "The Emergence of Spin Electronics in Data Storage," *Nature Materials*, vol. 6, p. 813, 2007.

[7] T. M. Maffitt, J. K. DeBrosse, J. A. Gabric, E. T. Gow, M. C. Lamorey, J. S. Parenteau, D. R. Willmott, M. A. Wood, and W. J. Gallagher, "Design Considerations for MRAM," *IBM Journal of Research & Development*, vol. 50, no. 1, p. 25, 2006.

[8] J. Slonczewski, "Current-Driven Excitation of Magnetic Multilayers," *"Journal of Magnetism and Magnetic Materials"*, vol. 159, p. 1, 1996.

[9] L. Berger, "Emission of Spin Waves by a Magnetic Multilayer Traversed by a Current," *"Pyhsical Review B"*, vol. 54, p. 9353, 1996.

[10] M. Hosomi, H. Yamagishi, T. Yamamoto, K. Bessho, Y. Higo, K. Yamane, H. Yamada, M. Shoji, H. Hachino, C. Fukumoto, H. Nagao, and H. Kano, "A Novel Nonvolatile Memory with Spin Torque Transfer Magnetization Switching: Spin-RAM," in *Electron Devices Meeting, 2005. IEDM Technical Digest. IEEE International*, 2005, p. 459.

[11] S. S. P. Parkin, M. Hayashi, and L. Thomas, "Magnetic Domain-Wall Racetrack Memory," *Science*, vol. 320, p. 190, 2008.

[12] S. Bohlens, B. Krüger, A. Drews, M. Bolte, G. Meier, and D. Pfannkuche, "Current Controlled Random-Access Memory Based on Magnetic Vortex Handedness," *"Applied Pyhsical Letters"*, vol. 93, p. 142508, 2008.

[13] A. Drews, B. Krüger, M. Bolte, and G. Meier, "Current- and Field-Driven Magnetic Antivortices," *"Pyhsical Review B"*, vol. 77, p. 094413, 2008.

[14] F. Bloch, "Zur Theorie des Austauschproblems und der Remanenzerscheinung der Ferromagnetika," *Zeitschrift für Physik A Hadrons and Nuclei*, vol. 74, p. 295, 1932.

[15] L. Landau and E. Lifshitz, "On the Theory of the Dispersion of Magnetic Permeability in Ferromagnetic Bodies," *Physikalische Zeitschrift der Sowjetunion*, vol. 8, p. 153, 1935.

[16] J. Slonczewski, "Currents and Torques in Metallic Magnetic Multilayers," *"Journal of Magnetism and Magnetic Materials"*, vol. 247, p. 324, 2002.

[17] G. Binasch, P. Grünberg, F. Saurenbach, and W. Zinn, "Enhanced Magnetoresistance in Layered Magnetic Structures with Antiferromagnetic Interlayer Exchange," *"Pyhsical Review B"*, vol. 39, p. 4828, 1989.

[18] M. Julliére, "Tunneling Between Ferromagnetic Films," *"Pyhsical Letters A"*, vol. 54, p. 225, 1975.

[19] Y. B. Bazaliy, B. A. Jones, and S.-C. Zhang, "Modification of the Landau-Lifshitz Equation in the Presence of a Spin-polarized Current in Colossal- and Giant-Magnetoresistive Materials," *"Pyhsical Review B"*, vol. 57, p. R3213, 1998.

[20] S. Zhang and Z. Li, "Roles of Nonequilibrium Conduction Electrons on the Magnetization Dynamics of Ferromagnets," *"Pyhsical Review Letters"*, vol. 93, p. 127204, 2004.

[21] A. Thiaville, Y. Nakatani, J. Miltat, and Y. Suzuki, "Micromagnetic Understanding of Current-Driven Domain Wall Motion in Patterned Nanowires," *Europhysics Letter*, vol. 69, p. 990, 2005.

[22] J. Held, J. Bautista, and S. Koehl, "From a Few Cores to Many: A Tera-Scale Computing Research Overview," *White paper*, 2006.

[23] M. Gschwind, P. Hofstee, B. Flachs, M. Hopkins, Y. Watanabe, and T. Yamazaki, "A Novel SIMD Architecture for the Cell Heterogeneous Chip Multiprocessor," *Hot Chips*, vol. 17, 2005.

[24] K. Asanovic, R. Bodik, B. C. Catanzaro, J. J. Gebis, P. Husbands, K. Keutzer, D. A. Patterson, W. L. Plishker, J. Shalf, S. W. Williams, and K. A. Yelick, "The Landscape of Parallel Computing Research: A View from Berkeley," Electrical Engineering and Computer Sciences, University of California at Berkeley, Tech. Rep. UCB/EECS-2006-183, 2006.

[25] J. E. Hannay, C. MacLeod, J. Singer, H. P. Langtangen, D. Pfahl, and G. Wilson, "How Do Scientists Develop and Use Scientific Software?" in *Proceedings of the 2009 ICSE Workshop on Software Engineering for Computational Science and Engineering*, ser. SECSE '09. "IEEE Computer Society", 2009, p. 1.

[26] J. C. Carver, R. P. Kendall, S. E. Squires, and D. 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*, ser. ICSE '07. "IEEE Computer Society", 2007, p. 550.

[27] B. Ge, "Programming in a High Level Approach for Scientific Computing," in *Computational Science and Its Applications - ICCSA 2003*, ser. Lecture Notes in Computer Science, V. Kumar, M. Gavrilova, C. Tan, and P. L'Ecuyer, Eds. Springer, 2003, vol. 2667, p. 962.

[28] S. Steinhaus, "Comparison of Mathematical Programs for Data Analysis," November 2010. [Online]. Available: http://www.scientificweb.com/ncrunch/ncrunch4.pdf

[29] A. Soni, "Analysis of Scientific Computing Environments: A Customer's View," Master's thesis, School of Engeneering, Massachusetts Insitute of Technology, 2008.

[30] K. Arnold, J. Gosling, and D. Holmes, *Java ®Programming Language, The (4th Edition)*. Addison-Wesley Professional, 2005.

[31] M. Grogan, "Scripting for the Java Platform," *Technical Report, Java Community Process (JSR-223)*, 2006.

[32] "Python Programming Language," January 2009. [Online]. Available: http://www.python.org/

[33] F. Perez and B. E. Granger, "IPython: A System for Interactive Scientific Computing," *"Computing in Science and Engineering"*, vol. 9, no. 3, p. 21, 2007.

[34] P. H. Langtangen, *Python - Scripting for Computational Science*. Springer, 2008.

[35] D. E. Hudak, N. Ludban, V. Gadepally, and A. Krishnamurthy, "Developing a Computational Science IDE for HPC Systems," in *Proceedings of the 3rd International Workshop on Software Engineering for High Performance Computing Applications*, ser. SE-HPC '07. "IEEE Computer Society", 2007, p. 5.

[36] D. E. Hudak, N. Ludban, A. Krishnamurthy, V. Gadepally, S. Samsi, and J. Nehrbass, "A Computational Science IDE for HPC Systems: Design and Applications," *International Journal of Parallel Programming*, vol. 37, p. 91, 2009.

[37] D. Hook and D. Kelly, "Testing for Trustworthiness in Scientific Software," in *ICSE Workshop on Software Engineering for Computational Science and Engineering, 2009*, ser. SECSE '09, 2009, p. 59.

[38] J. Segal, “Some Challenges Facing Software Engineers Developing Software for Scientists,” in *Proceedings of the 2009 ICSE Workshop on Software Engineering for Computational Science and Engineering*, ser. SECSE ’09. "IEEE Computer Society", 2009, p. 9.

[39] V. Maxville, “Preparing Scientists for Scalable Software Development,” in *Proceedings of the 2009 ICSE Workshop on Software Engineering for Computational Science and Engineering*, ser. SECSE ’09. Washington, DC, USA: "IEEE Computer Society", 2009, p. 80.

[40] “Micromagnetic Modeling Activity Group (μMag), NIST,” July 2008. [Online]. Available: http://www.ctcms.nist.gov/$~$rdm/mumag.org.html

[41] “MATLAB - The Language Of Technical Computing,” July 2008. [Online]. Available: http://www.mathworks.co.uk/products/matlab/

[42] “SciTools - Python Library for Scientific Computing,” November 2010. [Online]. Available: http://code.google.com/p/scitools/

[43] D. Berkov and J. Miltat, “Spin-Torque Driven Magnetization Dynamics: Micromagnetic Modeling,” *"Journal of Magnetism and Magnetic Materials"*, vol. 320, p. 1238, 2008.

[44] S. A. and T. Linz, *Basiswissen Softwaretest*. Dpunkt Verlag, 2005.

[45] K. Binder, “Computersimulationen,” *Physik Journal*, vol. 3, p. 25, 2004.

[46] R. H. Landau, “Resource Letter CP-2: Computational Physics,” *American Journal of Physics*, vol. 76, p. 296, 2008.

[47] V. R. Basili and M. V. Zelkowitz, “Empirical Studies to Build a Science of Computer Science,” *Communications of the ACM*, vol. 50, p. 33, 2007.

[48] J. K. Ousterhout, “Scripting: Higher-Level Programming for the 21st Century,” *"IEEE Computer"*, vol. 31, no. 3, p. 23, 1998.

[49] J. Brandt, P. J. Guo, J. Lewenstein, S. R. Klemmer, and M. Dontcheva, “Opportunistic Programming: Writing Code to Prototype, Ideate, and Discover,” *"IEEE Software"*, vol. 26, no. 5, p. 18, 2009.

[50] S. G. Johnson and M. Frigo, “A Modified Split-Radix FFT with Fewer Arithmetic Operations,” *"IEEE Transaction on Signal Processing"*, vol. 55, no. 1, p. 111, 2007.

[51] C. L. Lawson, R. J. Hanson, D. R. Kincaid, and F. T. Krogh, “Basic Linear Algebra Subprograms for Fortran Usage,” *ACM Transation on Mathematic Software*, vol. 5, no. 3, p. 324, 1979.

[52] C. W. Antoine, A. Petitet, and J. J. Dongarra, “Automated Empirical Optimization of Software and the ATLAS Project,” *Parallel Computing*, vol. 27, p. 2001, 2000.

[53] E. Anderson, Z. Bai, C. Bischof, J. Demmel, J. Dongarra, J. Du Croz, A. Greenbaum, S. Hammarling, A. McKenney, S. Ostrouchov, and D. Sorensen, *LAPACK’s User’s Guide*. "Society of Industrial and Applied Mathematics (SIAM)", 1992.

[54] J. Dongarra, G. H. Golub, E. Grosse, C. Moler, and K. Moore, "Netlib and NA-Net: Building a Scientific Computing Community," *IEEE Annals of the History of Computing*, vol. 30, p. 30, 2008.

[55] "Netlib Repository," November 2010. [Online]. Available: http://www.netlib.org/

[56] "Java Numerics," November 2010. [Online]. Available: http://math.nist.gov/javanumerics/

[57] S. Steinhaus, "The Scientific Web," January 2010. [Online]. Available: http://www.scientificweb.com/

[58] J. Kuan, "The Phantom Profits of the Opera: Nonprofit Ownership in the Arts as a Make-Buy Decision," *Journal of Law, Economics, and Organization*, vol. 17, p. 507, 2001.

[59] S. Jansen, A. Finkelstein, and S. Brinkkemper, "A Sense of Community: A Research Agenda for Software Ecosystems," in *31st International Conference on Software Engineering - Companion Volume, 2009. ICSE-Companion 2009.*, 2009, p. 187.

[60] S. Hauck and A. DeHon, Eds., *Reconfigurable Computing The Theory and Practice of FPGA-Based Computation*. Morgan Kaufmann Publishers, 2008.

[61] "CUDA," January 2011. [Online]. Available: http://www.nvidia.com/object/cuda$_$home$_$new.html

[62] W. Gropp, E. Lusk, and A. Skjellum, *MPI - eine Einfu̇hrung: Portable Parallele Programmierung mit dem Message-Passing Interface*. Oldenbourg Wissenschaftsverlag, 2007.

[63] V. Kindratenko, G. K. Thiruvathukal, and S. Gottlieb, "High-Performance Computing Applications on Novel Architectures," *"Computing in Science and Engineering"*, vol. 10, p. 13, 2008.

[64] "Top 500 List," November 2010. [Online]. Available: seehttp://www.top500.org/

[65] W. D. Gropp, "Software for Petascale Computing Systems," *"Computing in Science and Engineering"*, vol. 11, no. 5, p. 17, 2009.

[66] E. J. Chikofsky and J. H. Cross II, "Reverse Engineering and Design Recovery: A Taxonomy," *"IEEE Software"*, vol. 7, p. 13, 1990.

[67] T. Mens and T. Tourwe, "A Survey of Software Refactoring," *"IEEE Transaction on Software Engineering"*, vol. 30, p. 126, 2004.

[68] "Maple," January 2011. [Online]. Available: http://www.maplesoft.com/products/maple/

[69] "Mathematica," January 2011. [Online]. Available: http://www.wolfram.com/mathematica/

[70] "O-Matrix," January 2011. [Online]. Available: http://www.omatrix.com/

[71] "GNU Octave," November 2010. [Online]. Available: http://www.gnu.org/software/octave/

[72] "Scilab - The Free Platform for Numerical Computation," November 2010. [Online]. Available: http://www.scilab.org/

[73] C. Moler, "The Growth of MATLAB and The MathWorks over Two Decades," *The MathWorks News & Notes*, 2006.

[74] M. C. Lehn, "FLENS - A Flexible Library for Efficient Numerical Solutions," Ph.D. dissertation, Institut für Numerische Mathematik, Universität Ulm, 2008.

[75] M. Frigo and S. G. Johnson, "The Fastest Fourier Transform in the West," Massachusetts Institute of Technology, Tech. Rep. MIT-LCS-TR-728, September 1997.

[76] ——, "FFTW: An Adaptive Software Architecture for the FFT," in *Proceedings of the 1998 IEEE Intnational Conference on Acoustics Speech and Signal Processing*, vol. 3. IEEE, 1998, p. 1381.

[77] M. Frigo, C. E. Leiserson, H. Prokop, and S. Ramachandran, "Cache-Oblivious Algorithms," *Foundations of Computer Science, Annual IEEE Symposium on*, vol. 0, p. 285, 1999.

[78] M. Frigo and S. G. Johnson, "The Design and Implementation of FFTW3," *"Proceedings of the IEEE"*, vol. 93, no. 2, p. 216, 2005, special Issue on "Program Generation, Optimization, and Platform Adaptation".

[79] I. F. Darwin, *Checking C Programs with lint*. O'Reilly & Associates, Inc., 1986.

[80] G. Lombardi, "MUnit: A Unit Testing Framework in Matlab," January 2011. [Online]. Available: http://www.mathworks.com/matlabcentral/fileexchange/11306-munit-a-unit-testing-framework-in-matlab,~(09.01.2011)

[81] S. Papadimitriou, K. Terzidis, S. Mavroudi, and S. Likothanassis, "Scientific Scripting for the Java Platform with jLab," *"Computing in Science and Engineering"*, vol. 11, p. 50, 2009.

[82] M. Chevalier-Boisvert, L. Hendren, and C. Verbrugge, "Optimizing Matlab Through Just-In-Time Specialization," in *Compiler Construction*, ser. ICCS 2004, vol. 3039/2004. Springer, 2010, p. 46.

[83] J. Kepner, *Parallel MATLAB for Multicore and Multinode Computers*. "Society of Industrial and Applied Mathematics (SIAM)", 2009.

[84] B. Norris, A. Hartono, E. Jessup, and J. Siek, "Generating Empirically Optimized Composed Matrix Kernels from MATLAB Prototypes," in *Proceedings of the 9th International Conference on Computational Science*, ser. ICCS '09. Springer, 2009, p. 248.

[85] G. Sharma and J. Martin, "MATLAB ®: A Language for Parallel Computing," *International Journal of Parallel Progamming*, vol. 37, p. 3, 2009.

[86] A. Logg, H. P. Langtangen, and X. Cai, *Simula Research Laboratory*. Springer, 2009, ch. Past and Future Perspectives on Scientific Software, p. 321.

[87] "Eclipse IDE," January 2009. [Online]. Available: http://www.eclipse.org/

[88] "Pydev IDE," November 2010. [Online]. Available: http://www.pydev.org/

[89] T. Oliphant, *Guide to NumPy*. Trelgol Publishing, 2006.

[90] E. Jones, T. Oliphant, and P. Peterson, "SciPy: Open-Source Scientific Tools for Python," January 2009. [Online]. Available: http://www.scipy.org/

[91] P. N. Brown, G. D. Byrne, and A. C. Hindmarsh, "VODE: A Variable Coefficient ODE Solver," *SIAM Journal of Scientific and Statistical Computing*, vol. 10, no. 5, p. 1038, 1989.

[92] S. Purcell, "PyUnit," November 2010. [Online]. Available: http://pyunit.sourceforge.net/

[93] "py.test," November 2010. [Online]. Available: seehttp://pytest.org,~(09.01.2011)

[94] K. Beck, *Test Driven Development: By Example*. Addison-Wesley, 2003.

[95] D. Koening, A. Glover, P. King, G. Laforge, and J. Skeet, *Groovy in Action*. Manning Publications, 2007.

[96] D. Flanagan, *JavaScript: The Definitive Guide*. O'Reilly & Associates, Inc., 2006.

[97] "EMMA - A Free Java Code Coverage Tool," November 2010. [Online]. Available: http://emma.sourceforge.net/

[98] D. P. F. Möller, *Mathematical and Computational Modeling and Simulation*. Springer, 2004.

[99] P. J. Roache, *Verification and Validation in Computational Science and Engineering*. Hermosa Publishers, 1998.

[100] H. Balzert, *Lehrbuch der Softwaretechnik: Software-Management, Software-Qualitaẗssicherung, Unternehmensmodellierung*. Spektrum Akad. Verl., 1998.

[101] S. C. Reid, "An Empirical Analysis of Equivalence Partitioning, Boundary Value Analysis and Random Testing," in *Proceedings of the 4th International Symposium on Software Metrics*, ser. METRICS '97. "IEEE Computer Society", 1997, p. 64.

[102] G. J. Myers, T. Badgett, T. M. Thomas, and C. Sandler, *The Art of Software Testing*. Wiley & Sons, 2004.

[103] H. Kronmuẗller and S. S. P. Parkin, *Handbook of Magnetism and Advanced Magnetic Materials, Volume 4*. Wiley & Sons, 2007.

[104] I. Cimrák, "A Survey on the Numerics and Computations for the Landau-Lifshitz Equation of Micromagnetism," *Archives of Computational Methods in Engineering*, vol. 15, p. 277, 2008.

[105] L. Néel, "Some Theoretical Aspects of Rock-Magnetism," *Advances in Physics*, vol. 4, p. 191, 1955.

[106] T. L. Gilbert, "A Lagrangian Formulation of Gyromagnetic Equation of the Magnetization Field," *Physical Review*, vol. 100, p. 1243, 1955.

[107] W. F. Brown Jr., *Micromagnetics*. Interscience Publishers, 1963.

[108] A. Aharoni, *Introduction to the Theory of Ferromagnetism*. Oxford University Press, 1963.

[109] A. Hubert and R. Schäfer, *Magnetic Domains: The Analysis of Magnetic Microstructures*. Springer, 1998.

[110] H. Kronmüller and M. Fähnle, *Micromagnetism and the Microstructure of Ferromagnetic Solids*. Oxford University Press, 2003.

[111] T. L. Gilbert, "A Phenomenological Theory of Damping in Ferromagnetic Materials," *"IEEE Transaction on Magnetism"*, vol. 40, p. 3443, 2004.

[112] A. Drews, "Dynamics of Magnetic Vortices and Antivortices," Ph.D. dissertation, Institut für Angewandte Physik, Universität Hamburg, 2009.

[113] M. J. Donahue and R. D. McMichael, "Exchange Energy Representations in Computational Micromagnetics," *Physica B*, vol. 233, p. 272, 1997.

[114] M. J. Donahue and D. G. Porter, "Exchange Energy Formulations for 3D Micromagnetics," *Physica B - Condensed Matter*, vol. 343, p. 177, 2004, 4th International Symposium on Hysteresis and Micromagnetic Modeling (HMM 2003), Salamanca, Spain, May 28-30, 2003.

[115] P. Knabner and L. Angermann, *Numerik Partieller Differentialgleichungen: eine Anwendungsorientierte Einführung*. Springer, 2000.

[116] D. Braess, *Finite Elemente, Theorie, Schnelle Löser und Anwendungen in der Elastizitätstheorie*. Springer, 2007.

[117] M. R. Scheinfein, "LLG - Micromagnetics Simulator," July 2008. [Online]. Available: http://llgmicro.home.mindspring.com/

[118] D. V. Berkov, "MicroMagus - Software for Micromagnetic Simulation," July 2008. [Online]. Available: http://www.micromagus.de

[119] "AlaMag Micromagnetics Simulator," November 2010. [Online]. Available: http://faculty.mint.ua.edu/~visscher/AlaMag/

[120] "JaMM - Java MicroMagnetics," November 2010. [Online]. Available: http://jamm.uno.edu/

[121] M. J. Donahue and D. G. Porter, "Object Oriented Micromagnetic Framework, OOMMF User's Guide, Version 1.0," *National Institute of Standards and Technology, Gaithersburg, MD*, vol. Interagency Report NISTIR 6376, 1999.

[122] "RKMAG - User's manual," November 2010. [Online]. Available: http://www.rkmag.com/

[123] M. Abramowitz and I. Stegun, *Handbook of Mathematical Functions*. Dover, 1965.

[124] B. D'Acunto, *Computational Methods for PDE in Mechanics*. World Scientific Publ., 2004.

[125] D. Knuth, *The Art of Computer Programming, Volume 1: Fundamental Algorithms, Third Edition.* Addison-Wesley, 1997.

[126] J. Fielder and T. Schrefl, "Micromagnetic Modelling - The Current State of the Art," *Journal of Physics D: Applied Physics*, vol. 33, p. R135, 2000.

[127] J. R. Dormand and P. J. Prince, "A Family of Embedded Runge-Kutta Formulae," *"Journal of Computational and Applied Mathematics"*, vol. 6, p. 19, 1980.

[128] J. R. Cash and A. H. Karp, "A Variable Order Runge-Kutta Method for Initial Value Problems with Rapidly Varying Right-Hand Sides," *ACM Transaction on Mathematical Software*, vol. 16, p. 201, 1990.

[129] J. C. Butcher, *Numerical Methods for Ordinary Differential Equations*. Wiley and Sons Inc., West Sussex, UK, 1963.

[130] L. Baňas, "Numerical Methods for the Landau-Lifshitz-Gilbert Equation," in *Numerical Analysis and Its Applications*, ser. Lecture Notes in Computer Science, Z. Li, L. Vulkov, and J. Wasniewski, Eds. Springer, 2005, vol. 3401, p. 158.

[131] P. B. Monk and O. Vacus, "Accurate Discretization of a Non-Linear Micromagnetic Problem," *Computer Methods in Applied Mechanics and Engineering*, vol. 190, p. 5243, 2001.

[132] M. Labrune and J. Miltat, "Wall Structures in Ferro/Antiferromagnetic Exchange-Coupled Bilayers: a Numerical Micromagnetic Approach," *"Journal of Magnetism and Magnetic Materials"*, vol. 151, p. 231, 2008.

[133] L. Lopez-Diaz, O. Alejos, L. Torres, and J. I. Iniguez, "Solutions to Micromagnetic Standard Problem No. 2 Using Square Grids," *"Journal of Applied Physics"*, vol. 85, no. 8, p. 5813, 1999.

[134] A. Newell, W. Williams, and D. Dunlop, "A Generalization of the Demagnetization Tensor for Nonuniform Magnetization," *Journal of Geophysical Research*, vol. 98, p. 9551, 1993.

[135] J. Xiao, A. Zangwill, and M. D. Stiles, "Boltzmann Test of Slonczewskiś Theory of Spin-Transfer Torque," *"Pyhsical Review B"*, vol. 70, p. 172405, 2004.

[136] ——, "Macrospin Models of Spin Transfer Dynamics," *"Pyhsical Review B"*, vol. 72, p. 014446, 2005.

[137] S. Zhang and Z. Li, "Spin-Transfer Torque for Continuously Variable Magnetization," *"Pyhsical Review Letters"*, vol. 73, p. 054428, 2006.

[138] B. Krüger, D. Pfannkuche, M. Bolte, G. Meier, and U. Merkt, "Current-Driven Domain-Wall Dynamics in Curved Ferromagnetic Nanowires," *"Pyhsical Review B"*, vol. 75, p. 054421, 2007.

[139] "MagOasis," November 2010. [Online]. Available: http://www.magoasis.com/

[140] "FEMME," November 2010. [Online]. Available: http://www.firmasuess.at/?Products:FEMME

[141] K. Ramstöck, "MagFEM3D," July 2008. [Online]. Available: http://magfem3d.sourceforge.net/

[142] W. Scholz, "magpar - Parallel Finite Element Micromagnetics Package," January 2009. [Online]. Available: http://www.cwscholz.net/Main/MagparProject

[143] T. Fischbacher, M. Franchin, G. Bordignon, and H. Fangohr, "A Systematic Approach to Multiphysics Extensions of Finite-Element-Based Micromagnetic Simulations: Nmag," *"IEEE Transaction on Magnetism"*, vol. 43, p. 2896, 2007.

[144] M. Curcic, B. Van Waeyenberge, A. Vansteenkiste, M. Weigand, V. Sackmann, H. Stoll, H. Fahnle, T. Tyliszczak, G. Woltersdorf, C. H. Back, and G. Schütz, "Polarization Selective Magnetic Vortex Dynamics and Core Reversal in Rotating Magnetic Fields," *"Pyhsical Review Letters"*, vol. 101, p. 197204, 2008.

[145] A. Vansteenkiste, M. Weigand, M. Curcic, H. Stoll, G. Schütz, and B. Van Waeyenberge, "Chiral Symmetry Breaking of Magnetic Vortices by Sample Roughness," *New Journal of Physics*, vol. 11, p. 063006, 2009.

[146] W. Scholz, *Manual: magpar - version 0.9rc2 build 2916M*, 2009.

[147] W. Scholz, J. Fidler, T. Schrefl, D. Suess, R. Dittrich, H. Forster, and V. Tsiantos, "Scalable Parallel Micromagnetic Solvers for Magnetic Nanostructures," in *Proceedings of the Symposium on Software Development for Process and Materials Design*, vol. 28. Computational Materials Science, 2003, p. 366.

[148] S. Balay, K. Buschelman, V. Eijkhout, W. D. Gropp, D. Kaushik, M. G. Knepley, L. C. McInnes, B. F. Smith, and H. Zhang, "PETSc Users Manual," Argonne National Laboratory, Tech. Rep. ANL-95/11 - Revision 3.0.0, 2008.

[149] "TAO - Toolkit for Advanced Optimization," November 2010. [Online]. Available: http://www.mcs.anl.gov/research/projects/tao/

[150] "SUNDIALS - Suite of Nonlinear and Differential/Algebraic equation Solvers," November 2010. [Online]. Available: http://acts.nersc.gov/sundials/index.html

[151] "PVODE - Parallel VODE," November 2010. [Online]. Available: http://acts.nersc.gov/pvode/

[152] J. K. Ousterhout and K. Jones, *Tcl and the Tk Toolkit*. Addison-Wesley Professional, 2009.

[153] "Tcl Developer Exchange," November 2010. [Online]. Available: http://www.tcl.tk/

[154] M. Auperle, *Die Kunst der Programmierung mit C++: Exakte Grundlagen fuŕ die Professionelle Softwareentwicklung*. Vieweg & Sohn, 2002.

[155] "Visualization Toolkit (VTK)," January 2011. [Online]. Available: http://www.vtk.org/

[156] H. Fangohr and R. Boardman, "OVF2VTK: Tool for Conversion of OOMMF to VTK Files," January 2011. [Online]. Available: http://www.soton.ac.uk/~fangohr/software/ovf2vtk/index.html

[157] "Nmag Manual," November 2010. [Online]. Available: http://nmag.soton.ac.uk/nmag/

[158] "HDF5," January 2011. [Online]. Available: http://www.hdfgroup.org/HDF5/

[159] "MayaVi Project: 3D Scientific Data Visualization and Plotting," January 2011. [Online]. Available: http://code.enthought.com/projects/mayavi/

[160] T. Fischbacher, M. Franchin, G. Bordignon, A. Knittel, and H. Fangohr, "Parallel Execution and Scriptability in Micromagnetic Simulations," *"Journal of Applied Physics"*, vol. 105, p. 07D527, 2009.

[161] M. J. Donahue, D. G. Porter, R. D. McMichael, and J. Eicke, "Behavior of μ MAG Standard Problem No. 2 in the Small Particle Limit," *"Journal of Applied Physics"*, vol. 87, p. 5520, 2000.

[162] V. D. Tsiantos, D. Suess, T. Schrefl, and J. Fidler, "Stiffness Analysis for the Micromagnetic Standard Problem No. 4," *"Journal of Applied Physics"*, vol. 89, no. 11, p. 7600, 2001.

[163] R. D. McMichael, M. J. Donahue, D. G. Porter, and J. Eicke, "Switching Dynamics and Critical Behavior of Standard Problem No. 4," *"Journal of Applied Physics"*, vol. 89, p. 7603, 2001.

[164] T. L. and R. Wuyts, "Guest Editors' Introduction: Dynamically Typed Languages," *"IEEE Software"*, vol. 24, p. 28, 2007.

[165] J. Lohau, S. Kirsch, A. Carl, G. Dumpich, and E. F. Wassermann, "Quantitative Determination of Effective Dipole and Monopole Moments of Magnetic Force Microscopy Tips," *"Journal of Applied Physics"*, vol. 86, no. 6, p. 3410, 1999.

[166] M. Bolte, G. Meier, B. Krüger, A. Drews, R. Eiselt, L. Bocklage, S. Bohlens, T. Tyliszczak, A. Vansteenkiste, B. Van Waeyenberge, K. W. Chou, A. Puzic, and H. Stoll, "Time-Resolved X-Ray Microscopy of Spin-Torque-Induced Magnetic Vortex Gyration," *"Pyhsical Review Letters"*, vol. 100, no. 17, p. 176601, 2008.

[167] J. M. Garcia, A. Thiaville, J. Miltat, K. J. Kirk, J. N. Chapman, and F. Alouges, "Quantitative Interpretation of Magnetic Force Microscopy Images from Soft Patterned Elements," *"Applied Pyhsical Letters"*, vol. 79, no. 5, p. 656, 2001.

[168] M. J. Donahue, "Parallelizing a Micromagnetic Program for Use on Multiprocessor Shared Memory Computers," *"IEEE Transaction on Magnetism"*, vol. 45, p. 3923, 2009.

[169] P. Duhamel and M. Vetterli, "Fast Fourier-Transforms - A Tutorial Review and a State-of-the-Art," *"Signal Processing"*, vol. 19, no. 4, p. 259, 1990.

[170] M. Puschel, J. Moura, J. Johnson, D. Padua, M. Veloso, B. Singer, J. Xiong, F. Franchetti, A. Gacic, Y. Voronenko, K. Chen, R. Johnson, and N. Rizzolo, "SPIRAL: Code Generation for DSP Transforms," *"Proceedings of the IEEE"*, vol. 93, p. 232, 2005.

[171] "Intel ®Math Kernel Library 10.3," January 2011. [Online]. Available: http://software.intel.com/en-us/articles/intel-mkl/

[172] S. G. Johnson and M. Frigo, "Implementing FFTs in Practice," in *Fast Fourier Transforms*, C. S. Burrus, Ed. Connexions, 2008, ch. 11.

[173] M. Frigo, "A Fast Fourier Transform Compiler," in *Proceedings of the 1999 ACM SIGPLAN Conference on Programming Language Design and Implementation*, vol. 34, no. 5. Association for Computing Machinery (ACM), 1999, p. 169.

[174] C. Blilie, "Patterns in Scientific Software: An Introduction," *"Computing in Science and Engineering"*, vol. 4, p. 48, 2002.

[175] "pylint - Python Code Static Checker," November 2010. [Online]. Available: http://www.logilab.org/project/pylint

[176] "lint4j - lint for Java," November 2010. [Online]. Available: http://www.jutils.com/

[177] "Metrics," November 2010. [Online]. Available: http://metrics.sourceforge.net/

[178] "PyMetrics," January 2011. [Online]. Available: http://pymetrics.sourceforge.net/

[179] "Java SciMark 2.0," January 2009. [Online]. Available: http://math.nist.gov/scimark2/

[180] P. Wendykier and J. G. Nagy, "Large-Scale Image Deblurring in Java," in *Lecture Notes In Computer Science*, vol. 5101. Springer, 2008, p. 721.

[181] W. Hoschek, "The Colt Distribution: Open Source Libraries for High Performance Scientific and Technical Computing in Java," January 2009. [Online]. Available: http://dsd.lbl.gov/~hoschek/colt/

[182] "Mathworks - MATLAB and Simulink for Technical Computing," July 2010. [Online]. Available: http://www.mathworks.com/

[183] "easy_install," January 2011. [Online]. Available: http://peak.telecommunity.com/DevCenter/EasyInstall

[184] S. Liang, *The Java Native Interface: Programmerś Guide and Specification.* Addison-Wesley, 1999.

[185] P. Luszczek, "Parallel Programming in MATLAB," *International Journal of High Performance Computing Applications*, vol. 23, no. 3, p. 277, 2009.

[186] "Star-P," January 2011. [Online]. Available: http://www.microsoft.com/pathways/star-p/

[187] "pyMPI - Putting the py in MPI," November 2010. [Online]. Available: http://pympi.sourceforge.net/

[188] G. L. Taboada, J. Touriño, and R. Doallo, "Java for High Performance Computing: Assessment of Current Research and Practice," in *Proceedings of the 7th International Conference on Principles and Practice of Programming in Java*, ser. PPPJ '09. New York, NY, USA: Association for Computing Machinery (ACM), 2009, p. 30.

[189] B. Krüger, A. Drews, M. Bolte, U. Merkt, D. Pfannkuche, and G. Meier, "Harmonic Oscillator Model for Current- and Field-Driven Magnetic Vortices," *"Pyhsical Review B"*, vol. 76, p. 224426, 2007.

[190] Y. Tserkovnyak, H. J. Skadsem, A. Brataas, and G. E. W. Bauer, "Current-Induced Magnetization Dynamics in Disordered Itinerant Ferromagnets," *"Pyhsical Review B"*, vol. 74, p. 144405, 2006.

[191] R. A. Duine, A. S. Núñez, J. Sinova, and A. H. MacDonald, "Functional Keldysh Theory of Spin Torques," *"Pyhsical Review B"*, vol. 75, p. 214420, 2007.

[192] M. Hayashi, L. Thomas, Y. B. Bazaliy, C. Rettner, R. Moriya, X. Jiang, and S. S. P. Parkin, "Influence of Current on Field-Driven Domain Wall Motion in Permalloy Nanowires from Time Resolved Measurements of Anisotropic Magnetoresistance," *"Pyhsical Review Letters"*, vol. 96, p. 197207, 2006.

[193] G. Meier, M. Bolte, R. Eiselt, B. Krüger, D. Kim, and P. Fischer, "Direct Imaging of Stochastic Domain-Wall Motion Driven by Nanosecond Current Pulses," *"Pyhsical Review Letters"*, vol. 98, p. 187202, 2007.

[194] C. J. García-Cervera, Z. Gimbutas, and W. E, "Accurate Numerical Methods for Micromagnetics Simulations with General Geometries," *"Journal of Computational Physics"*, vol. 184, p. 37, 2003.

[195] M. J. Donahue and R. D. McMichael, "Micromagnetics on Curved Geometries Using Rectangular Cells: Error Correction and Analysis," *"IEEE Transaction on Magnetism"*, vol. 43, p. 2078, 2007.

[196] S. Cohen and C. Hindmarsh, "CVODE, A Stiff/Nonstiff ODE Solver in C," *Computers in Physics*, vol. 10, p. 138, 1996.

[197] L. Petzold and A. Hindmarsh, *LSODA (Livermore Solver of Ordinary Differential Equations)*. Computing and Mathematics Research Division, Lawrence Livermore National Laboratory, 1997.

[198] F. Franchetti, M. Puschel, Y. Voronenko, S. Chellappa, and J. Moura, "Discrete Fourier Transform on Multicore," *"IEEE Signal Processing Magazine"*, vol. 26, p. 90, 2009.

[199] A. Nukada and S. Matsuoka, "Auto-Tuning 3-D FFT Library for CUDA GPUs," in *Proceedings of the Conference on High Performance Computing Networking, Storage and Analysis*, ser. SC '09. New York, NY, USA: Association for Computing Machinery (ACM), 2009, p. 30:1.

[200] C. Seberino and H. Bertram, "Concise, Efficient Three-Dimensional Fast Multipole Method for Micromagnetics," *Magnetics, IEEE Transactions on*, vol. 37, p. 1078, 2001.

Contributions to the publications

Publication SCSC'07

The conference proceedings article entitled "Simulating Magnetic Storage Elements: Implementation of the Micromagnetic Model into *MATLAB* - Case Study for Standardizing Simulation Environments" was presented at the 2007 Summer Computer Simulation Conference SCSC'07 (that took place between 15 and 18 July 2007 in San Diego, USA) and is reprinted in Sec. 3.1.1.

The chapter "Implementation into MATLAB" depicts the main results. This chapter is written by myself and the depicted results are derived mainly by myself. Further I contributed to the summary and outlook in discussions.

All other chapters are written by myself with contibutions of my coauthors in discussions.

Publication HSC'08

The conference proceedings article entitled "A Case Study for the Parallelization of a Complex MATLAB Program with Respect to Maintainability" was presented at the 2008 Huntsville Simulation Conference (HSC'08) (that took place between 22 and 23 October 2008 in Huntsville, USA).

The chapters "The Micromagnetic Modeling and Simulation Kit M^3S" and "Performance Analysis of the Sequential Omplementation" showing a performance analysis of M^3S are derived by myself.

The chapers "Parallelization" and "Results" are written by myself on basis of the results derived by Gunnar Selke.

All other chapters are written by myself with contibutions of my coauthors in discussions.

Publication GCMS'08

The conference proceedings article entitled "The Micromagnetic Modeling and Simulation Kit M^3S for the Simulation of the Dynamic Response of Ferromagnets to Electric Currents" was presented at the 2008 Grand Challenges in Modeling and Simulation Conference GCMS'08 (that took place between 16 and 19 June 2008 in Edinburgh, UK).

The chapter "Theoretical Background" and "M^3S" were written by myself. Here Benjamin Krüger gave support with his theoretical physics knowledge. Benjamin also contributed equally to me to the section entitled "Spin-transfer Torque for Continuously Varying Magnetization" within the chapter "Validation".

All other chapters are written by myself with contibutions of my coauthors in discussions.

Publication JAP'09

The journal article entitled "Proposal for a Standard Problem for Micromagnetic Simulations Including Spin-Transfer Torque" has been published in the Journal of Applied Physics in 2009.

The chapter "Problem Selection" was derived mainly by myself. My coauthors Benjamin Krüger, Stellan Bohlens, and Guido Meier contributed equally to this chapter in discussions about the integrity and quality of the chosen criteria.

The chapter "Problem Definition" was derived mainly by myself. Here Benjamin Krüger contributed equally in the selection of the concrete simulation parameters for the final problem definition. Further Matteo Franchin suggested the formula 3.

The chapter "Experimental Feasibility" was derived by myself, Benjamin Krüger, and Guido Meier contributing equally.

The appendices were derived by myself, Benjamin Krüger, and Hans Fangohr:

- "A" was written by myself.
- "B" was written by myself on basis of a formula provided by Benjamin Krüger
- "C" was written by Matteo Franchin

All other chapters are written by myself with contibutions of my coauthors in discussions.

Publication PRL'10

The article "Proposal of a Robust Measurement Scheme for the Nonadiabatic Spin Torque Using the Displacement of Magnetic Vortices" has been published in Physical Review Letters in 2010. The article presents a measurement scheme that is robust against typical experimental perturbations and hence allow the measurement of the degree of non-adiabaticity with a unique accuracy.

This article is mainly written by Benjamin Krüger. My contributions to this article are as follows:

Benjamin Krüger and I together developed the idea of a proposal for a measurement scheme for the degree of non-adiabaticity. I further contributed conceptually to the figures 2 and 3 and the relating simulation experiments.

Manuscript 1

The extensions of M^3S-*MATLAB* by the static calculation of the current paths and the AMR-effect have been performed in cooperation with Stellan Bohlens. This manuscript is written by Stellan Bohlens. Stellan Bohlens and I contributed equally to the results depicted in the chapter entitled "Numerical Simulations" as this chapter discusses details of these M^3S-*MATLAB* extensions.

Supporting material for publication PRL' 10

This article has been published as supporting material for the article reprinted in Sec. 4.3. It is mainly written by Benjamin Krüger and has been added for reasons of completeness to the appendix. I contributed in discussions to this supporting material.

Eidesstattliche Erklärungen Massoud Najafi Maryam Negari

Ich versichere an Eides statt, dass ich bisher an keinem anderen Fachbereich einen Antrag auf Eröffnung eines Promotionsprüfungsverfahrens gestellt habe.

Hamburg, den 21.01.2011

Massoud Najafi Maryam Negari

Ich versichere an Eides statt, dass ich meine Dissertation "Micromagnetic Modeling by Computational Science Integrated Development Environments (CSIDE)" selbst verfasst habe und keine anderen als die angegebenen Hilfsmittel benutzt habe.

Hamburg, den 21.01.2011

Massoud Najafi Maryam Negari