Micromagnetic Device Simulation

João Pedro Gomes Moutinho

Thesis to obtain the Master of Science Degree in

Engineering Physics

Supervisors:  Prof. Dr. José Luís Rodrigues Júlio Martins
              Prof. Dr. Susana Isabel Pinheiro Cardoso de Freitas

Examination Committee

Chairperson:  Prof. Dr. Pedro Domingos Santos do Sacramento
Supervisor:   Prof. Dr. José Luís Rodrigues Júlio Martins
Member of the Committee:  Dr. Bertrand Lacoste

April 2017
To Mike and Gaspar.
Acknowledgments

First and foremost I would like to express my sincerest gratitude to my supervisors Professor José Luís Martins and Professor Susana Cardoso de Freitas for their guidance and support during the course of this thesis. I would like to highlight the support of Professor José, with whom I worked more directly, whose vast experience in computational physics provided me a rich learning environment to develop this thesis and tackle all of the problems that came with it. I also thank Professor Susana for introducing me to the field of Micromagnetics and suggesting me to pursue computational work with Professor José.

I would like to take a moment to acknowledge all my friends and colleagues who accompanied me on my academic career during this last couple of years. I have to mention Guilherme Raposo, Filipe Richheimer, João Ferreira and José Carvalho who I worked with side by side right from the start of the course up to this very conclusion. Not only without them, but also Rita Franco, Mariana Fernandes, Martim Pardal, and all the other great people I crossed paths with, the memories would all just be hard work. I would also like to acknowledge Francisco Pesquita, who I have been close friends with for too many years to count, and who always encouraged me to “bi teh best person mi can bi”.

Last but not least I would like to thank my family for their unconditional support. Without the help of my parents, João Moutinho and Ana Ribas, who always supported my decisions and encouraged me to prioritize my education I would have never been able to reach this point.
Resumo

Dispositivos micromagnéticos têm um grande número de aplicações no desenvolvimento de novas tecnologias, desde soluções compactas e de alta capacidade para o armazenamento de informação, RAMs magnetoresistivas rápidas e não voláteis, e também sensores com aplicações em biomedicina e na indústria. Simulações computacionais permitem estudar este tipo de dispositivos de uma forma eficiente e versátil.

A implementação numérica do modelo micromagnético é estudada usando uma discretização de diferenças-finitas. Os dois problemas principais nas simulações micromagnéticas são o cálculo do campo desmagnetizante entre os momentos magnéticos e a integração temporal da equação de Landau-Lifshitz-Gilbert.

São estudados métodos numéricos para a solução do campo desmagnetizante, onde métodos baseados em FFTs têm uma vantagem de eficiência. Um novo método baseado em FFTs é desenvolvido utilizando um kernel de convolução calculado através de uma discretização em sistema linear da equação de Poisson. É demonstrado que este método tem vantagens sobre outros métodos de FFT em termos de eficiência e precisão quando se usa uma discretização de Yee para as equações de Maxwell.

A integração temporal da equação LLG é estudada. Métodos de Runge-Kutta não conservativos foram implementados juntamente com um método de Gauss-Seidel semi-conservativo e um método mid-point implícito conservativo, descritos na literatura. Estes métodos demonstram ser opções viáveis para resolver a integração temporal, sacrificando alguma eficiência computacional a favor de precisão.

Por fim apresentamos o nosso trabalho agregado numa ferramenta em Mathematica para a simulação de válvulas de spin, fácil de adaptar e modificar devido à natureza compacta da linguagem.

Abstract

Micromagnetic devices have a large number of applications in the development of new technologies, from compact and large capacity information storage solutions, fast and non volatile magnetoresistive RAMs, and also sensory devices used in biomedical and industrial applications. Computational simulations provide an efficient and versatile tool to study the behaviour of these devices and aid their experimental development.

The numerical implementation of the micromagnetic model is studied using a finite-difference discretization. The two main problems in micromagnetic simulations are the solution of the global demagnetizing interaction between magnetic dipoles and the time integration of the Landau-Lifshitz-Gilbert equation to describe magnetization dynamics.

Numerical methods for the demagnetizing interaction are overviewed, where FFT based methods have a clear efficiency advantage. A new FFT based method is developed using a convolution kernel computed from a linear system discretization of the magnetostatic Poisson equation. We show this method has advantages over other FFT methods both in terms of speed and accuracy when a Yee cell discretization of Maxwell’s equations is used.

The time integration of the LLG equation is studied, where nonconservative Runge-Kutta methods were implemented alongside a semi-conservative Gauss-Seidel method and a conservative implicit mid-point method, found in literature. These methods prove to be promising options to deal with the time integration aspects of the model, sacrificing some efficiency in favour of accuracy.

Finally, our work was implemented in a Mathematica tool prepared to study spin valve devices, very easy to adapt and modify due to the compact nature of the language.

Keywords: Micromagnetic Device Simulation, Numerical Methods, FFT Poisson Solver, Conservative LLG Integration, Open Source Simulation Tool.
# Contents

Acknowledgments ................................................................. v
Resumo ...................................................................................... vii
Abstract ................................................................................... ix
List of Tables ................................................................. xv
List of Figures .............................................................. xvii
Nomenclature ............................................................... xix
Glossary .................................................................................... xxi

Introduction

1 The Micromagnetic Model .................................................. 5
   1.1 Micromagnetic Free Energy ........................................... 5
       1.1.1 Thermodynamics of Magnetic Media ..................... 6
       1.1.2 The Exchange Interaction ............................... 8
       1.1.3 The Anisotropy Interaction ............................ 9
       1.1.4 The Demagnetizing Interaction ....................... 10
       1.1.5 The Zeeman Interaction .................................. 12
       1.1.6 The Free Energy Functional ......................... 12
   1.2 Micromagnetic Equilibrium ......................................... 12
       1.2.1 Brown’s Equations and the Effective Field ...... 14
   1.3 Dynamic Equations .................................................... 15
       1.3.1 Landau-Lifshictz Equation ............................ 15
       1.3.2 The Gilbert Damping .................................. 16

2 Numerical Implementation .............................................. 19
   2.1 Discretization .......................................................... 19
       2.1.1 Finite-Differences ....................................... 20
   2.2 The Discrete Micromagnetic Model ......................... 21
       2.2.1 Landau-Lifshitz-Gilbert Equation .................. 21
       2.2.2 Exchange Interaction ................................. 22
       2.2.3 Anisotropy Interaction ............................... 23
       2.2.4 Demagnetizing Interaction ......................... 23
<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>2.2.5 Zeeman Interaction</td>
<td>26</td>
</tr>
<tr>
<td>2.3 Natural Units</td>
<td>26</td>
</tr>
<tr>
<td>3 Demagnetizing Field</td>
<td>27</td>
</tr>
<tr>
<td>3.1 Underlining the Problem</td>
<td>27</td>
</tr>
<tr>
<td>3.2 Direct Poisson Discretization</td>
<td>29</td>
</tr>
<tr>
<td>3.2.1 Boundary Conditions</td>
<td>29</td>
</tr>
<tr>
<td>3.2.2 Constructing the Linear System</td>
<td>30</td>
</tr>
<tr>
<td>3.2.3 Solving the Linear System</td>
<td>32</td>
</tr>
<tr>
<td>3.2.4 Linear System Accuracy</td>
<td>35</td>
</tr>
<tr>
<td>3.2.5 Shielded Periodic FFTs</td>
<td>37</td>
</tr>
<tr>
<td>3.3 FFT Convolution Method</td>
<td>37</td>
</tr>
<tr>
<td>3.3.1 Analytical Kernels</td>
<td>38</td>
</tr>
<tr>
<td>3.3.2 A Numerical Approach to the Scalar Kernel</td>
<td>41</td>
</tr>
<tr>
<td>3.3.3 Kernel Accuracy and Efficiency</td>
<td>43</td>
</tr>
<tr>
<td>4 Magnetization Dynamics</td>
<td>49</td>
</tr>
<tr>
<td>4.1 Landau-Lifshitz-Gilbert Equation</td>
<td>49</td>
</tr>
<tr>
<td>4.1.1 Structural Properties</td>
<td>50</td>
</tr>
<tr>
<td>4.2 Numerical Methods for the LLG Equation</td>
<td>51</td>
</tr>
<tr>
<td>4.2.1 Runge-Kutta Methods (RK)</td>
<td>51</td>
</tr>
<tr>
<td>4.2.2 Gauss-Seidel Projection Method (GS)</td>
<td>54</td>
</tr>
<tr>
<td>4.2.3 Mid-Point Geometrical Integration (MP)</td>
<td>56</td>
</tr>
<tr>
<td>4.3 Method Comparison</td>
<td>58</td>
</tr>
<tr>
<td>5 Spin Valve Simulation</td>
<td>61</td>
</tr>
<tr>
<td>5.1 Introduction to Spin Valves</td>
<td>62</td>
</tr>
<tr>
<td>5.1.1 Antiferromagnetic Pinning</td>
<td>62</td>
</tr>
<tr>
<td>5.1.2 Interlayer Coupling</td>
<td>63</td>
</tr>
<tr>
<td>5.2 Energy Minimization</td>
<td>64</td>
</tr>
<tr>
<td>5.3 Basic Spin Valve Simulation</td>
<td>65</td>
</tr>
<tr>
<td>6 Concluding Remarks</td>
<td>69</td>
</tr>
<tr>
<td>6.1 Thesis Overview</td>
<td>69</td>
</tr>
<tr>
<td>6.2 Comments and Future Work</td>
<td>70</td>
</tr>
<tr>
<td>Bibliography</td>
<td>73</td>
</tr>
<tr>
<td>A Open Source Codes</td>
<td>79</td>
</tr>
<tr>
<td>A.1 Tool Comparison</td>
<td>79</td>
</tr>
<tr>
<td>A.1.1 Installation</td>
<td>80</td>
</tr>
<tr>
<td>A.1.2 User Interface</td>
<td>80</td>
</tr>
</tbody>
</table>
A.1.3 Final Remarks ................................................................. 82

B Spin Valve Simulation Tool .................................................. 83
   B.1 User Interface .............................................................. 83
   B.2 Source Code ............................................................... 87

C Scientific Poster .............................................................. 91
List of Tables

<table>
<thead>
<tr>
<th>Table</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>2.1</td>
<td>Physical quantities normalized during implementation. The 1/2 factor in $\mu_0$ is simply to keep the natural unit of length consistent with the conventional exchange length definition.</td>
<td>26</td>
</tr>
<tr>
<td>2.2</td>
<td>Natural units used during the simulation.</td>
<td>26</td>
</tr>
<tr>
<td>3.1</td>
<td>Fitting parameters from function 3.25 to the data in figure 3.2.</td>
<td>35</td>
</tr>
<tr>
<td>3.2</td>
<td>Efficiency comparison between the four different methods discussed: linear system (LS), analytical kernel $G_a$, analytical kernel $S$ and numerical kernel $G_n$ for $\delta = {32, 64}$. Calculated on an Intel Core i5-3570k processor.</td>
<td>45</td>
</tr>
<tr>
<td>3.3</td>
<td>Reliability comparison between four methods to calculate the demagnetizing field: linear system method (LS) and three convolution methods with the analytical kernel $G_a$, analytical kernel $S$ and numerical kernel $G_n$ for $\delta = {32, 64}$. A cube with $L = 50$ nm was simulated with three different grids for the case of random magnetization and flower state magnetization.</td>
<td>45</td>
</tr>
<tr>
<td>5.1</td>
<td>Conversion from SI units to Gaussian-cgs units. Note that $emu$ is short for &quot;electromagnetic unit&quot; is not an actual unit in the traditional sense as it can be used to represent different physical quantities. The main difference in this system is that $M$ and $H$ have different units, with an extra factor of $4\pi$ in the conversion factor, which is equal to a factor of $\mu_0$. Equations relating $M$ and $H$ in SI units being transformed into Gaussian-cgs should have $H$ multiplied by a factor of $1/\mu_0$ to compensate.</td>
<td>61</td>
</tr>
<tr>
<td>B.1</td>
<td>Main arrays defined in the spin valve simulation tool.</td>
<td>89</td>
</tr>
<tr>
<td>B.2</td>
<td>Main functions defined in the spin valve simulation tool.</td>
<td>89</td>
</tr>
</tbody>
</table>
Initially tested in 1952 with the IBM 405 Alphabetical Accounting Machine, the ferrite core memory was the main form of random-access memory for the next 20 years, before semiconductor memories were introduced in the 1970s. In the image we see a 4 kb chip (64 × 64 array of cores) as used in the CDC6600, a supercomputer delivered to CERN in 1965. One bit of information, a "0" or "1", can be stored in each ring through the direction of the magnetization, which is controlled by the flow of currents. By inputing a current in a given $x$ and $y$ coordinate a single core out of the whole array can be selected to read/write, since cores with current flowing in only one direction are unaffected.

Spin valve type structure with two ferromagnets $F_1, F_2$, separated by a non-magnetic spacer, in both antiparallel and parallel states, corresponding to high and low resistance states. Changes between these two states can be obtained through the manipulation of external magnetic fields. The produced resistance changes can be measured with electrical currents flowing through the device. The layers usually have width and length in the order of micrometers, with nanometer thickness.

1. Separation of the dynamical motion of a single magnetization vector in its two main components: precession and damping, respectively.

2. Center-vector grid discretization. Every cell has the three components of each vectorial quantity discretized at the center.

2.1 Conjugate-dual grid discretization, or Yee’s Grid [32], used in the FDTD method. Grid dimensions are as shown in figure 2.1. Each spatial component of the field vectors is discretely organized along the respective borders of the cells in the grid.

3. Scheme of the considerations used for the multipole expansion of the boundary values. Each chunk, in yellow, has its own monopole, dipole and quadrupole. The boundaries are at a distance $L$ from the material, and the multipole expansion is only calculated for the red cells. We use the yellow color for charge grids, or conjugate grids.

3.1 Performance scaling of various linear system methods and a FFT method for the solution of Poisson’s equation in a charged cube made up of $N$ cells. Calculated on an Intel Core i5-3570k processor.
3.3 Horizontal cut through the center of a uniformly magnetized cube inserted in a $16 \times 16 \times 16$ grid, with the respective demagnetizing field. We use the green color for magnetization grids, or center-vector grids.

3.4 Ferromagnetic slab similar to Permalloy magnetized in an equilibrium s-state.

3.5 Scalar potential in natural units along the left border marked in red in figure 3.4 in terms of the cell index of the conjugate grid.

3.6 Scheme of the considerations used to calculate the numerical scalar kernel $G_n$.

3.7 Scalar potential along the border marked in red in figure 3.4 in terms of the cell index, including the results from the numerical approximation of kernel $G_n$.

3.8 Vertical cut through the center magnetized cube discretized in a $16 \times 16 \times 16$ grid with both a random magnetization distribution and a flower state distribution.

3.9 Time evolution of the average magnetization components according to the Standard Problem #4 [51], for the applied field 3.39. Results obtained with the scalar kernel $G_n$ were compared with the ones computed with the simulation codes MuMax [13, 14] and MicroMagnum [16], as well as the previous data submitted by Rocha [11] to the $\mu$Mag site [52].

3.10 Magnetization distribution over the slab of Standard Problem #4 [51] when $\langle m_x \rangle$ crosses 0. The z component is color coded. Each vector point is an average of the magnetization of $3 \times 3 \times 2$ cells.

4.1 Solution of the Standard Problem #4 [51] using three different time integration methods for the LLG equation: 4th order Runge Kutta, Gauss-Seidel Projection and Implicit Mid-Point. A spatial discretization of $128 \times 32 \times 1$ cells was used with a time step of $\delta t = 0.002 (\gamma M_s)^{-1}$, roughly $10^{-14}$ s. Average computation time per iteration — RK4: 36.0 ms, GS: 114.6 ms, MP: 176.4 ms.

5.1 Simple spin valve schematic in both parallel and antiparallel states.

5.2 Basic spin valve with both free and pinned layers in permalloy separated by a copper spacer. We consider a pinning field of $H_{pin} = 200 e_y$ Oe. The easy axis is also along the $y$ direction for both layers, however the anisotropy constant of the pinned layer is 20 times greater than the free layer.

5.3 Transfer curve of the average magnetization along the $y$ axis for the spin valve of figure 5.2, simulated both in our Mathematica tool as well as MuMax.

5.4 Relative variation between maximum (1.0) and minimum (-1.0) values of resistance for the spin valve in figure 5.2, according to equation 5.7.
Nomenclature

Mathematical Operations

\[ \delta_{\alpha\beta} \quad \text{Kronecker delta.} \]

\[ \langle f \rangle \quad \text{Spatial average.} \]

\[ \mathcal{F}(g) \quad \text{Fourier Transform.} \]

\[ \| v \| \quad \text{Euclidean norm.} \]

\[ f * g \quad \text{Convolution operation.} \]

Physical Constants

\[ \gamma \quad \text{Electron gyromagnetic ratio,} \, (2.21 \times 10^5 \, \text{m} \, \text{A}^{-1} \, \text{s}^{-1}). \]

\[ \hbar \quad \text{Reduced Planck constant,} \, (1.0545718 \times 10^{-34} \, \text{m}^2 \, \text{kg} \, \text{s}^{-1}). \]

\[ \mu_0 \quad \text{Vacuum permeability,} \, (4\pi \times 10^{-7} \, \text{N} \, \text{A}^{-2}). \]

\[ A_{\text{ex}} \quad \text{Exchange constant, material dependent,} \, (\text{J} \, \text{m}^{-1}). \]

\[ K_1 \quad \text{First order anisotropy constant, material dependent,} \, (\text{J} \, \text{m}^{-3}). \]

\[ M_s \quad \text{Saturation magnetization, material dependent,} \, (\text{A} \, \text{m}^{-1}). \]

Subscripts

\[ i, j, k \quad \text{Computational indices for 3D spatial discretization.} \]

\[ x, y, z \quad \text{Vector components in each cartesian coordinate direction.} \]

Superscripts

\[ n \quad \text{Time discretization index.} \]
## Glossary

<table>
<thead>
<tr>
<th>Abbreviation</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>FFT</td>
<td>Fast Fourier Transform — computational algorithm to compute the Discrete Fourier Transform of a sequence of data, or its inverse.</td>
</tr>
<tr>
<td>INESC-MN</td>
<td>Instituto de Engenharia de Sistemas e Computadores - Microsistemas e Nanotecnologias is a private, non-profit research and development institute in the areas of micro- and nanotechnologies and their application to electronic, biological and biomedical devices.</td>
</tr>
<tr>
<td>LLG</td>
<td>The Landau-Lifshitz-Gilbert equation is the main partial differential equation to describe magnetization dynamics.</td>
</tr>
<tr>
<td>MKL</td>
<td>Math Kernel Library — Library of computational mathematical routines such as Linear System methods and FFTs specifically optimized for Intel processors.</td>
</tr>
<tr>
<td>SOR</td>
<td>Successive Over-Relaxation is a numerical method in linear algebra to solve a linear system of equations.</td>
</tr>
</tbody>
</table>
Introduction

Motivation and Background

Over the past few decades the transition to the digital world has lead to an exponential growth in the requirements of information storage, both in terms of capacity and access speed. Magnetic materials have long been the focus of considerable research due to their wide range of technological applications. Not only are they found in mass magnetic storage, but also in sensing devices, magnetic stripes on credit cards and magnetic ink for character recognition used in the banking industry to process and clear documents.

Certain magnetic materials, referred to as ferromagnets, present spontaneous magnetization at room temperature resulting from the alignment of elementary magnetic moments. The formed magnetization distributions can then be manipulated with appropriate external magnetic fields. The fact that a given material can be found to have more than one stable state of magnetization opens up a vast array of possibilities when it comes to physically encoding information. One of the first applications of these principles in the area of magnetic storage was the Ferrite Core Memory, as shown in figure 1.

Figure 1: Initially tested in 1952 with the IBM 405 Alphabetical Accounting Machine, the ferrite core memory was the main form of random-access memory for the next 20 years, before semiconductor memories were introduced in the 1970s. In the image we see a 4kb chip (64 × 64 array of cores) as used in the CDC6600, a supercomputer delivered to CERN in 1965. One bit of information, a "0" or "1", can be stored in each ring through the direction of the magnetization, which is controlled by the flow of currents. By inputing a current in a given $x$ and $y$ coordinate a single core out of the whole array can be selected to read/write, since cores with current flowing in only one direction are unaffected.
Figure 2: Spin valve type structure with two ferromagnets $F_1, F_2$, separated by a non-magnetic spacer, in both antiparallel and parallel states, corresponding to high and low resistance states. Changes between these two states can be obtained through the manipulation of external magnetic fields. The produced resistance changes can be measured with electrical currents flowing through the device. The layers usually have width and length in the order of micrometers, with nanometer thickness.

Magnetic storage, of course, did not end with the ferrite core memory. The now widely used hard-drives of desktop computers are a very well-known example of high capacity magnetic storage. The chip shown in figure 1 has dimensions in the order of centimeters with a storage capacity of only $4 \times 10^3$ bits, and earlier versions were even larger. A common hard disk found today in a laptop computer has about the same size and can store more than 1 terabyte of data, or roughly $8 \times 10^{12}$ bits, an increase of nine orders of magnitude in the span of 50 years! From bits with dimensions of about 1 mm$^2$ as shown in figure 1 we have now reached the point where modern recording and sensing technology treats magnetic media with dimensions in the order of micro to nanometers — often referred to as micromagnetic devices.

More recent research has lead to the development of Magnetoresistive Random Access Memories (MRAM), which can match or even surpass common RAMs in terms of access speed, but still maintain the non-volatility characteristic of other magnetic storage devices (versus the electrical dependent and thus volatile RAMs commonly used today in desktop computers). An efficient and cost effective Non-Volatile RAM could serve as a universal memory used for all types of storage, and MRAMs are only one of the contenders. The available options, however, still have very limited commercialization.

The term “magnetoresistive”, as seen in MRAM, refers to the possibility of some magnetic materials or devices to produce changes in electrical resistance that are coupled with changes in the magnetization distribution. A common micromagnetic device with magnetoresistive properties, widely used both in the development of magnetic sensors and binary structures for MRAM cells, is the Spin Valve. This is a thin layered structure composed of both magnetic and non magnetic conducting materials, as exemplified in figure 2. The interest in these structures arises from the presence of the Giant Magnetoresistance (GMR). While not the only magnetoresistive effect, GMR is commonly observed in thin-film structures. It was discovered in 1988 independently by Albert Fert [1] and Peter Grünberg [2], granting them the 2007 Nobel Prize in Physics. The effect observed was that the electrical resistance of the layered device would change with the orientation of the magnetization in the various ferromagnetic layers. Given that electrical resistance changes are very easy to read, this type of effect has paved the way for numerous technological applications.

The micromagnetic device development group at INESC-MN\(^1\) specializes in magnetoresistive devices with biomedical and industrial sensory applications [3, 4, 5, 6], having also worked with magnetic storage focused devices [7, 8]. The process of developing these devices has a lot of stages and decisions that must be made in terms of layout, design rules, and nanofabrication. It is thus very common to resort to

---

\(^1\)Instituto de Engenharia de Sistemas e Computadores - Microsistemas e Nanotecnologias — http://www.inesc-mn.pt/
computer simulations to assist this process. These simulations must be reliable so that underneath all the complex interactions in the device the results can be trusted to match the physical reality of the system, and must be efficient not only to allow more device configurations to be tested in less time, but also to allow more complex systems to be simulated with the available computer power.

There has been constant interest in developing faster and more robust numerical methods for the computational implementation of the micromagnetic model. Previous codes have been developed at INESC-MN with this purpose [9, 10, 11], although they are now mostly outdated. Currently the experimental group uses the simulation software SpinFlow3D, however this tool has already stopped receiving support from its creators and online resources have been shut down. User made changes to the code are also very limited due to it being a closed-source software bound by a commercial license.

Similar motivations have lead other researchers to create fully-featured open-source micromagnetic simulation tools such as OOMMF [12], MuMax3 [13, 14], magnum.fe [15] and MicroMagnum [16], ready to be used by experimentalists. Some of these were also tested during the course of this thesis and considered as an option to be used by INESC-MN. Given the complexity of the micromagnetic model, however, it is not ideal to start working on these tools as without having a better understanding of how they work. For that reason, we started this thesis with the intent to go deeper and study some of the specific numerical problems that arise during the creation of such a tool from the ground up. In this context it is very important to create a tool that is easy to read and modify so that different students and researchers can learn to use it in an efficient manner and have complete control over the implemented interactions, having also the possibility to easily adjust the code should the need arise.

Objectives and Outline

The micromagnetic model can include various interactions depending on the device it is being applied to. In this thesis we will introduce the exchange interaction, a quantum effect related to the subatomic alignment of spins, the anisotropy interaction, related to the geometrical influence of lattices in magnetization distributions, the demagnetizing interaction, related to the long-range interaction of magnetic dipoles, and the Zeeman interaction, which accounts for externally applied fields in the system. Furthermore, the time evolution of the magnetization will be treated as described by the Landau-Lifshitz equation, and we will also introduce interlayer interactions for spin valve structures.

The demagnetizing interaction and the time integration of the Landau-Lifshitz equation are the two most resource intensive calculations in computational micromagnetics. The treatment of these two problems will be our main focus, while also working towards the improvement of the current situation with regards to micromagnetic simulations at INESC-MN.

The mathematical equations that describe both the demagnetizing field and the dynamical evolution of the magnetization have important physical properties rooted within. It is imperative that a discretized and numerically implemented physical model conserves its analytical properties in order to maximize the reliability of the numerical results. As we will see, however, this proves to be an intricate topic in the field of micromagnetic simulations. We will overview both the common implementations of these two problems.
in well known simulation tools and also other numerical methods found in literature. Furthermore, for the
demagnetizing field problem, we will present a new approach to the FFT convolution method which proves
to be faster than other methods found in literature by reducing the number of needed FFT operations
and conserves Maxwell’s equations in the discretization scheme that we use.

The development of a complete and general purpose micromagnetic simulation tool is a continuous
process and requires a lot of work, certainly more than the scope of this master thesis. The research groups
at INESC-MN have a great history of mentoring students from IST\textsuperscript{2} during internships and dissertations.
We aim not only to study the numerical implementation of the basic micromagnetic model, but also to
present both the theoretical and numerical aspects in a comprehensive way. Our objective is that this
work can serve as a reference to introduce future students to computational micromagnetics and provide
a solid basis for future work to be developed.

With the aforementioned considerations, we will outline this thesis in the following manner:

• In the first chapter we will introduce the micromagnetic model and explain the theoretical back-
ground of each of the interactions considered as well as the dynamical equations.

• In the second chapter we will present the discretization of the model and discuss the implications
that the conservation of certain fundamental physical and mathematical properties have in the
choice of discretization and numerical treatment of the model.

• The third chapter will be dedicated to the solution of the demagnetizing field problem where we
will start with an introductory linear system approach to the problem and work our way to the
more often used methods in simulation tools based on FFTs, also presenting a new approach to
these methods that proves to have advantages both in terms of efficiency and reliability in the
discretization scheme that we use.

• In the fourth chapter we will treat the implementation of magnetization dynamics with the time
integration of the Landau-Lifshitz equation and the important physical properties that must be
conserved. Most simulation tools use non-conservative Runge-Kutta methods due to their efficiency,
however both partial and fully conservative methods have already been developed in literature.

• In the fifth chapter we will discuss the simulation of spin valve structures by using the numerical
methods studied in our work to create a simple and compact simulation tool to be used at INESC-
MN.

• Finally we will present some concluding remarks on the work developed, its applicability and the
future work that could be pursued.

A more detailed introduction of these topics will be presented at the beginning of each chapter.

\textsuperscript{2}Técnico Lisboa — https://tecnico.ulisboa.pt/en/
The Micromagnetic Model

One of the first publications to present a detailed and unified description of the micromagnetic model was the work of Brown [17]. In this chapter we will overview the model and introduce its main aspects and mathematical treatment. Micromagnetic concepts have since been covered in other literature such as Landau et al. [18], Jackson [19] and Aharoni [20].

We will start with a discussion on some thermodynamic considerations for magnetic media, from which we will present the various interactions that occur on ferromagnetic materials. The micromagnetic equilibrium is found by minimizing the energy of these interactions. This minimization leads us to introduce an effective magnetic field which takes into account all the interactions, and is an important tool to solve problems within the micromagnetic model. Moreover, we will discuss the Landau-Lifshitz-Gilbert equation which describes the time evolution of the magnetization towards equilibrium solutions.

1.1 Micromagnetic Free Energy

The theory of micromagnetics stands upon a continuous magnetization framework. For a given magnetic body occupying a region $\Omega \subseteq \mathbb{R}^3$ we consider a “small” portion of it with volume $\Delta V_r$, defined by the position vector $r \in \Omega$. This volume is small when compared to the size of the whole body, but still large enough to contain a group of elementary magnetic moments $\mu_i$, $i = 1, ..., N$ for a statistically large $N$.

We now define the magnetization\(^1\) as a vector field $M(r)$ where the product $M(r) \Delta V_r$ represents the net magnetic moment in this small volume,

$$M(r) = \frac{1}{\Delta V_r} \sum_{i=1}^{N} \mu_i.$$  \hspace{1cm} (1.1)

Furthermore, for a dynamical situation, this vector field can also be a function of time,

$$M = M(r, t).$$  \hspace{1cm} (1.2)

The micromagnetic model includes both short-range interactions and long-range interactions described by Maxwell-type fields. This framework allows the treatment of both in terms of the free energy of the magnetic body, which we will introduce starting from the thermodynamics of magnetic media.

\(^1\)In the context of Maxwell’s equations we will refer a modern and more precise definition of magnetization.
1.1.1 Thermodynamics of Magnetic Media

We consider an infinitesimal volume $dV$ of a magnetic material in contact with a thermal bath at a constant temperature $T$. Volume expansions due to thermal and magnetic effects are neglected. We represent the net magnetic moment as the quantity $\mu_0 M = \mu_0 M dV$, where $\mu_0$ is the vacuum permeability.

Following the First Law of Thermodynamics, the energy conservation for an arbitrary infinitesimal transformation between two equilibrium states has a contribution of the heat absorbed and the work performed in the system,

$$dU = \delta Q + \delta W,$$ (1.3)

where $U$ is the state function that represents the internal energy of the system. For a constant external magnetic field $H$ felt by the infinitesimal magnetic volume $dV$ the magnetic work performed by a change of magnetization, as seen in equation 1.31 of Mandl [21], is

$$\delta W = \mu_0 H \cdot \delta M.$$ (1.4)

The Second Law of Thermodynamics can be stated as the existence of entropy, the state function $S$. As such, for isolated and non-isolated systems, the net change in entropy $dS$ in the system during an arbitrary thermodynamic transformation can be written as an inequality

Isolated: $dS \geq 0, \quad$ Non-Isolated: $dS \geq \frac{\delta Q}{T}$.

(1.5)

In both cases the equality is satisfied for reversible transformations.

Depending on the conditions under which the system is being studied, it is useful to introduce appropriate thermodynamic potentials, other than the internal energy $U$, by means of Legendre transformations. For constant temperature transformations we introduce the Helmholtz free energy, $F(M, T)$,

$$F = U - TS.$$ (1.6)

For an infinitesimal transformation under a fixed temperature $T$ we then have

$$dF = dU - TdS.$$ (1.7)

Using both the first and second law of thermodynamics for a non-isolated system we arrive at

$$dF \leq \delta W.$$ (1.8)

For the case where no work is done by the system the previous inequality yields

$$dF \leq 0.$$ (1.9)

As such, given an infinitesimal transformation with fixed temperature, the Helmholtz free energy of the
system evolves towards a minimum.

In addition to a fixed temperature $T$ we can now also consider a fixed external magnetic field $\mathbf{H}$, at which point it is useful to introduce the Gibbs free energy, $G(\mathbf{H}, T)$,

$$G = F - \mu_0 \mathbf{M} \cdot \mathbf{H}. \quad (1.10)$$

The same reasoning applied to the Helmholtz free energy now yields the inequality

$$dG \leq 0. \quad (1.11)$$

Much like the Helmholtz free energy, the Gibbs free energy also evolves towards a minimum under an infinitesimal transformation with fixed temperature and external magnetic field.

In the reversible transformation case of the free energy inequalities presented, it is easy to find that

$$dF = \delta W = \mu_0 \mathbf{H} \cdot \delta \mathbf{M}, \quad (1.12)$$
$$dG = -\mu_0 \mathbf{M} \cdot \delta \mathbf{H}. \quad (1.13)$$

These two equalities lead, respectively, to the following equations of state

$$\left. \frac{1}{\mu_0} \frac{\partial F}{\partial \mathbf{M}} \right|_T = \mathbf{H}, \quad \left. \frac{\partial G}{\partial \mathbf{H}} \right|_T = -\mu_0 \mathbf{M}. \quad (1.14)$$

The Gibbs free energy, as was defined in 1.10, only depends on $\mathbf{H}$ and $T$, and so the net magnetic moment must be written as an equation of state depending on the same variables,

$$\mathbf{M} = \mathbf{M}(\mathbf{H}, T), \quad (1.15)$$

which means that, for a given external field $\mathbf{H}$ and temperature $T$, at thermodynamic equilibrium, $\mathbf{M}$ is a uniquely determined state variable.

This result, however, is not valid for the case of a non-homogeneous magnetic system, such as a ferromagnetic body. To extend the results to this case we must consider the state variables as space-dependent state functions, assuming the body is in local thermodynamic equilibrium, and thus promote the free energies to functionals. So, for a ferromagnetic body occupying a region $\Omega$ with a magnetization vector field distribution $\mathbf{M}(\mathbf{r})$, which we consider to play the role of the state function, the free energies are written as integrals over the whole magnetic domain,

$$F[\mathbf{M}] = \int_{\Omega} f(\mathbf{M}, \hat{n}\mathbf{M}) d^3r, \quad \left. \frac{1}{\mu_0} \frac{\delta F}{\delta \mathbf{M}(\mathbf{r})} = \mathbf{H}(\mathbf{r}) \right|_{\Omega} \quad (1.16)$$

where $f$ is a function of the magnetization distribution and a local approximation of its first spatial derivatives. The derivatives in the state functions become functional derivatives. We can now focus our discussion on the various interactions that contribute to the free energy functional for ferromagnetic bodies.
1.1.2 The Exchange Interaction

The key interaction in ferromagnetic materials is the exchange interaction. These materials are known present a very strong spontaneous magnetization of the order of the saturation magnetization, even in the absence of an external field. This characteristic of ferromagnetic bodies arises from the effective spin-spin interactions of electrons on the atomic scale, which would require a much more complicated description than the Heisenberg model we will present. The alignment of neighbouring spins suggests ferromagnets should have small uniformly magnetized regions. These regions, called magnetic domains, were postulated by Weiss [22] in 1907. His theory included a phenomenological molecular field that produced these alignments. Later, in 1931, Heisenberg theoretically justified this approach describing the exchange interaction on the basis of quantum theory [23]. These magnetic domains have since been observed experimentally [24].

Weiss’s theory defined the behaviour of the magnetization magnitude along the ferromagnetic material as being simply a function of temperature and equal to the saturation magnetization, \( M_s = M_s(T) \). In order to extract the information about the direction of the magnetization at every location inside the magnetic body we must resort to micromagnetics. Given the continuum approximation we introduced, for constant temperature, Weiss’s theory separates the magnitude and direction of the magnetization vector field,

\[
M(\mathbf{r}, t) = M_s \mathbf{m}(\mathbf{r}, t),
\]

where we have introduced the magnetization unit-vector field \( \mathbf{m}(\mathbf{r}, t) \). The continuum approximation also implies we are not interested in dealing with the exchange of single atomic spins. Instead, we want to describe the exchange interaction in terms of the phenomena occurring on a larger spatial scale, such as the way the magnetic moments \( \mathbf{M} \Delta V_r \) interact with each other.

In 1935 Landau and Lifshitz [25] proposed the introduction of a term in the free energy to penalize magnetization disuniformities in function of the gradients of the magnetization components,

\[
f_{\text{ex}}(\mathbf{m}) = A_{\text{ex}} \left[ (\nabla m_x)^2 + (\nabla m_y)^2 + (\nabla m_z)^2 \right],
\]

with the constant \( A_{\text{ex}} \) having dimensions of \([J/m]\). While this exchange constant can be experimentally determined for each material, it is also possible to provide a theoretical approximation from a semi-classical continuum approach to the Heisenberg exchange interaction. Let us consider a cubic lattice of spins, with the interaction energy represented by the Heisenberg Hamiltonian summed only over nearest neighbours,

\[
\mathcal{H} = -2J \sum \mathbf{S}_i \cdot \mathbf{S}_j \tag{1.19}
\]

\[
= -2JS^2 \sum \cos(\theta_{i,j}). \tag{1.20}
\]

The spin angular momenta in lattice sites \( i \) and \( j \) are represented by \( \mathbf{S}_i \) and \( \mathbf{S}_j \), in units of \( \hbar \), and \( J \) is the average exchange strength. The spins have magnitude \( S \) and directions given by the unit vectors \( \mathbf{m}_i \).
and $\mathbf{m}_j$, where $\theta_{i,j}$ is the angle between them. We assume the interaction sufficiently strong such that neighbouring spins become almost parallel. This justifies a small angle approximation of 1.20,

$$\mathcal{H} \simeq -2JS^2 \sum \left( 1 - \frac{1}{2} \theta_{i,j}^2 \right) = E_{\text{const.}} + JS^2 \sum \theta_{i,j}^2.$$  

(1.21)

$$\simeq E_{\text{const.}} + JS^2 \sum (\mathbf{m}_j - \mathbf{m}_i)^2.$$  

(1.22)

We have also used the approximation that for small $\theta_{i,j}$, $|\theta_{i,j}| = \|\mathbf{m}_j - \mathbf{m}_i\|$. The first term is simply a constant shift to the total energy. Let us now consider $r_i$ and $r_j$ as the position vectors of the lattice sites $i$ and $j$, such that $\Delta r_j = r_j - r_i$ defines the position of neighbour $j$ in respect to site $i$. In the limit where the lattice becomes a continuous media there exists a continuous function $\mathbf{m}$ such that

$$\mathbf{m}_j - \mathbf{m}_i = \Delta r_j \cdot \nabla \mathbf{m}.$$  

(1.23)

Plugging the continuous limit into 1.22, where we neglect the constant term,

$$\mathcal{H} = JS^2 \sum (\Delta r_j \cdot \nabla \mathbf{m})^2$$  

(1.24)

$$= JS^2 \sum \left[ (\Delta r_j \cdot \nabla m_x)^2 + (\Delta r_j \cdot \nabla m_y)^2 + (\Delta r_j \cdot \nabla m_z)^2 \right]$$  

(1.25)

If we consider $n$ to be the number of spins per unit volume, we must now sum 1.25 over $j$ and multiply by $n$ to obtain the energy per unit volume $f_{\text{ex}}$. When summing over $j$ some geometrical considerations must be taken into account. The symmetry of a cubic lattice implies that, for $\Delta r_j = (x_j, y_j, z_j)$, cross terms sum to zero, $\sum_j x_j y_j = 0$, and the diagonal terms are all equal, $\sum_j x_j^2 = \sum_j y_j^2 = \sum_j z_j^2 = \frac{1}{3} \sum_j \Delta r_j^2$. All of these considerations lead to

$$f_{\text{ex}}(\mathbf{m}) = \left( \frac{1}{6} nJS^2 \sum_{\Delta \mathbf{r}_j} \nabla \mathbf{m} \right) \left[ (\nabla m_x)^2 + (\nabla m_y)^2 + (\nabla m_z)^2 \right],$$  

(1.26)

which is expression 1.18, as was proposed by Landau and Lifshitz. The exchange constant $A_{\text{ex}}$ can be particularized for other lattice geometries.

Integrating 1.26 over the whole $\Omega$ region gives us the contribution of the exchange interaction to the free energy of the magnetic body,

$$F_{\text{ex}}[\mathbf{M}] = \int_{\Omega} A_{\text{ex}} \left[ (\nabla m_x)^2 + (\nabla m_y)^2 + (\nabla m_z)^2 \right] d^3r.$$  

(1.27)

### 1.1.3 The Anisotropy Interaction

Anisotropic effects are a common occurrence in ferromagnetic bodies. The lattice structure of each material and certain crystal symmetries give rise to energy-favored directions for the magnetization. Even in the absence of an applied magnetic field, certain ferromagnetic materials tend to be magnetized along these “easy” directions. The micromagnetic model accounts for anisotropic effects by adding a
phenomenological term to the free energy functional that favours the easy direction alignment of the magnetization.

Much like the exchange interaction, the anisotropy effect concerns the direction of the magnetization, and so we write the energy per unit volume $f_{an}(\mathbf{m})$ as a function of the magnetization unit vector $\mathbf{m}$. Integrating $f_{an}(\mathbf{m})$ over the whole $\Omega$ region we obtain the anisotropy contribution to the free energy functional,

$$F_{an}[\mathbf{M}] = \int_{\Omega} f_{an}(\mathbf{m}) \, d^3r. \quad (1.28)$$

The phenomenological density $f_{an}(\mathbf{m})$ is always defined in such a way that the direction of $\mathbf{m}$ parallel to the easy directions minimize this contribution to the free energy of the system. It is physically reasonable to consider the antiparallel case to be energetically equivalent.

The most common case is for ferromagnetic materials to have uniaxial anisotropy, meaning there exists only one easy direction, or easy axis. As we will see, uniaxial anisotropy also accounts for the case where there are two easy directions. Anisotropic effects with cubic symmetry can also occur, where the material has three privileged directions, and while these will not be discussed in this work, they are very simple to implement should future need arise. Going back to the discussion of uniaxial anisotropy, it is considered that the energy density $f_{an}(\mathbf{m})$ is rotationally-symmetric with respect to the easy axis. Let us now consider the unit vector field $\mathbf{u}_{an}(\mathbf{r})$ which represents, for each position $\mathbf{r}$, the easy axis. It is very common for $\mathbf{u}_{an}$ to simply be a constant direction over all $\mathbf{r} \in \Omega$, but here we introduce the general case.

In order to verify the aforementioned considerations the following series can be written:

$$f_{an}(\mathbf{m}) = - \sum_{l=0}^{\infty} K_l \left( \mathbf{u}_{an} \cdot \mathbf{m} \right)^{2l} \quad (1.29)$$

$$\simeq - K_0 - K_1 \left( \mathbf{u}_{an} \cdot \mathbf{m} \right)^2. \quad (1.30)$$

The expansion is usually truncated at $l = 1$, as we have done in 1.30. The $K_0$ term is a constant shift to the energy of the system and can be neglected. In the second term, the important behaviour comes from the anisotropy constant $K_1$. If $K_1 > 0$, the energy density is minimized for $\mathbf{m}$ parallel to $\mathbf{u}_{an}$ and so we have a single easy axis. If, however, $K_1 < 0$ then the energy density is minimized for $\mathbf{m}$ perpendicular to $\mathbf{u}_{an}$, meaning there are now two preferential directions for the magnetization that define the plane perpendicular to $\mathbf{u}_{an}$.

Integrating 1.30 over the whole $\Omega$ region, neglecting the constant term, we arrive at the anisotropy contribution to the free energy functional,

$$F_{an}[\mathbf{M}] = - \int_{\Omega} K_1 \left( \mathbf{u}_{an} \cdot \mathbf{m} \right)^2 \, d^3r. \quad (1.31)$$

1.1.4 The Demagnetizing Interaction

The demagnetizing interaction accounts for the long-range interactions produced by the magnetization distribution within the material. At each position in the magnetic body a magnetostatic or demagnetizing field is felt with a contribution from the whole magnetization distribution. In order to introduce this
effect into the model we have to appropriately describe the demagnetizing field \( H_d \). Let us start by presenting the macroscopic formulation of Maxwell’s equations,

\[
\nabla \cdot \mathbf{E} = \frac{\rho}{\varepsilon_0}, \quad \nabla \times \mathbf{E} = -\frac{\partial \mathbf{B}}{\partial t}, \quad \nabla \cdot \mathbf{B} = 0, \quad \nabla \times \mathbf{B} = \mu_0 \left( \mathbf{j} + \varepsilon_0 \frac{\partial \mathbf{E}}{\partial t} \right). \tag{1.32}
\]

Locally, the basic microscopic quantities inside a material are the microscopic fields \( E_{\text{micro}} \) and \( B_{\text{micro}} \), which fluctuate on the atomic scale. Here we use the macroscopic fields \( E \) and \( B \) which are averages over a macroscopic length scale [19]. Furthermore, in continuous media, we have the constituent equations

\[
\mathbf{D} = \varepsilon_0 \mathbf{E} + \mathbf{P}, \quad \mathbf{H} = \frac{1}{\mu_0} \mathbf{B} + \mathbf{M}, \tag{1.33}
\]

where \( \mathbf{P} \) and \( \mathbf{M} \) are the macroscopic polarization and magnetization, respectively. While the macroscopic polarization \( \mathbf{P} \) and magnetization \( \mathbf{M} \) have units of electric/magnetic dipole per unit volume, their definition in terms of microscopic quantities is not that intuitive, and was unsolved until the 1990s. A modern theory of polarization has been in development since then, and an equivalent theory for magnetization was only addressed in 2005. A review of these theories was published by Resta [26] in 2010.

The modern approach to polarization \( \mathbf{P} \) is in fact to define its variation \( \Delta \mathbf{P} \) in terms of currents, and not charges, avoiding the definition of an “absolute” polarization of a given equilibrium state. This is in agreement with experiments which measure polarization differences. For the case of magnetization, there are two microscopic contributions to the macroscopic \( \mathbf{M} \) as defined in nonrelativistic quantum mechanics: spin magnetization \( \mathbf{M}_{\text{spin}} \) and orbital magnetization \( \mathbf{M}_{\text{orb}} \). While the term \( \mathbf{M}_{\text{spin}} \) is in fact a dipolar density of electron spins, the contribution \( \mathbf{M}_{\text{orb}} \) from the orbital motion of electrons is not and requires a precise definition at the fundamental level. Essentially, the physical meaning of \( \mathbf{P} \) and \( \mathbf{M}_{\text{orb}} \) lies in the basic quantities \( \rho_{\text{micro}}(\mathbf{r}) \) and \( \mathbf{j}_{\text{micro}}(\mathbf{r}) \), the microscopic charge and orbital current densities, which are in no way dipolar densities. More on this topic and its implications can be read in [26].

Going back to the macroscopic formulation, in order to study the magnetostatic case for a magnetized region \( \Omega \), we neglect any currents and electric fields, and we consider a quasi-static field approximation. Thus the third and fourth equation from the set in 1.32 can be written as

\[
\nabla \cdot \mathbf{H}_d = -\nabla \cdot \mathbf{M} \quad \text{and} \quad \nabla \times \mathbf{H}_d = 0, \tag{1.34}
\]

where we have used the relation between \( \mathbf{B} \) and \( \mathbf{H} \) from 1.33, and denominated \( \mathbf{H} \) as the demagnetizing field \( \mathbf{H}_d \). The set of equations 1.34 have to be satisfied for the whole space \( \Omega_{\infty} \). Outside the magnetized region \( \Omega \) the divergence equation becomes simply \( \nabla \cdot \mathbf{H}_d = 0 \). The usual mathematical formalism employed in this problem is to consider the body occupying \( \Omega \) to have a well defined surface \( \partial \Omega \), at which point the discontinuity of \( \mathbf{M}(\mathbf{r}) \) must be appropriately treated. As discussed in [19], physical distributions of magnetization are in fact mathematically well behaved and without discontinuities. However, the idealization that \( \mathbf{M}(\mathbf{r}) \) within \( \Omega \) falls suddenly to zero at the surface \( \partial \Omega \) is sometimes convenient. We will come back to this topic in chapter 3 when we talk about the numerical treatment of the demagnetizing problem, discussing the implications of this consideration, as well as our approach to the problem.
Essentially, for every magnetization distribution $M(r)$, there is a demagnetizing field $H_d(r)$ satisfying Eq. 1.34 that interacts with the system, and so we must account for it as a contribution to the free energy functional. The energy density for a magnetostatic field can be written as

$$f_{\text{demag}} = \frac{1}{2} \mu_0 H_d^2.$$  (1.35)

However, $H_d$ is defined in $\Omega_\infty$ requiring Eq. 1.35 to be integrated over the whole space. Once again recalling the relation between $B$ and $H$ from 1.33 we can rewrite 1.35 as

$$f_{\text{demag}} = \frac{1}{2} \mu_0 H_d \cdot \left( \frac{1}{\mu_0} B - M \right).$$  (1.36)

When we integrate 1.36 over $\Omega_\infty$ the first term vanishes due to the integral orthogonality of $B$ and $H_d$. The contribution to the free energy functional is thus given by the integration of the second term, which is only defined within $\Omega$,

$$F_{\text{demag}}[M] = - \int_{\Omega} \frac{1}{2} \mu_0 (M \cdot H_d) \, d^3r.$$  (1.37)

### 1.1.5 The Zeeman Interaction

An extra term can be added to the free energy functional to account for the introduction of an externally applied field, which has been neglected in the previous interactions. This field, $H_{\text{ext}}$, is independent of the magnetization distribution, and its introduction is now treated with the Gibbs free energy. In fact, equation 1.10 from our thermodynamics introduction, when written for the continuum media, is also a functional where the extra term added to the Helmholtz free energy represents the Zeeman energy contribution,

$$G_{\text{ext}}[M, H_{\text{ext}}] = - \int_{\Omega} \mu_0 (M \cdot H_{\text{ext}}) \, d^3r.$$  (1.38)

### 1.1.6 The Free Energy Functional

The terms we have introduced so far represent the main interactions in the micromagnetic model. Collecting their contributions from the expressions 1.27, 1.31, 1.37 and 1.38 we arrive at

$$G[M, H_{\text{ext}}] = F_{\text{ex}}[M] + F_{\text{an}}[M] + F_{\text{demag}}[M] + G_{\text{ext}}[M, H_{\text{ext}}]$$

$$= \int_{\Omega} \left[ A_{\text{ex}} (\nabla m)^2 - K_1 (u_{\text{an}} \cdot m)^2 - \frac{1}{2} \mu_0 (M \cdot H_d) - \mu_0 (M \cdot H_{\text{ext}}) \right] \, d^3r.$$  (1.40)

where $(\nabla m)^2 = (\nabla m_x)^2 + (\nabla m_y)^2 + (\nabla m_z)^2$. Equation 1.40 represents the micromagnetic free energy functional for a ferromagnetic body.

### 1.2 Micromagnetic Equilibrium

The equilibrium configuration of magnetization arises from minimizing the free energy functional with the restriction that $\|M(r)\| = M_s$. We thus consider that the first-order variation $\delta G$ vanishes for any
variation $\delta m$ of the unit-vector field $m$, where we have the constraint $\|m + \delta m\| = 1$.

**Exchange Variation**

The first-order variation of equation 1.27 can be written as

$$\delta F_{ex} = F_{ex}[m + \delta m] - F_{ex}[m] = \int_{\Omega} 2A_{ex} \nabla m \cdot \nabla \delta m \, d^3r, \quad (1.41)$$

where $\nabla m \cdot \nabla \delta m = (\nabla m_x \cdot \nabla \delta m_x + \nabla m_y \cdot \nabla \delta m_y + \nabla m_z \cdot \nabla \delta m_z)$. Using the vector identity $v \cdot \nabla f = \nabla \cdot (fv) - f \nabla \cdot v$ and using the divergence theorem in the resulting first term of the identity we obtain

$$\delta F_{ex} = -\int_{\Omega} \left[ 2A_{ex} \nabla^2 m \cdot \delta m \right] \, d^3r + \int_{\partial \Omega} \left[ 2A_{ex} \frac{\partial m}{\partial n} \cdot \delta m \right] \, d^2r, \quad (1.42)$$

where $n$ is the outward normal of $\partial \Omega$.

**Anisotropy Variation**

The first-order variation of equation 1.31 is simply

$$\delta F_{an} = -\int_{\Omega} 2K_1(m \cdot u_{an})u_{an} \cdot \delta m \, d^3r. \quad (1.43)$$

**Demagnetizing Variation**

The first-order variation of equation 1.37 is given by

$$\delta F_{demag} = -\int_{\Omega} \frac{1}{2} \mu_0 M_s (\delta m \cdot H_d - m \cdot \delta H_d) \, d^3r, \quad (1.44)$$

which takes into account the fact that the demagnetizing field is dependent on the magnetization. According to the reciprocity theorem [20], however, these two terms are equal, and so we write

$$\delta F_{demag} = -\int_{\Omega} \mu_0 M_s H_d \cdot \delta m \, d^3r. \quad (1.45)$$

**Zeeman Variation**

The external field is independent of the magnetization distribution, and so the first-order variation of equation 1.38 is simply

$$\delta G_{ext} = -\int_{\Omega} \mu_0 M_s H_{ext} \cdot \delta m \, d^3r. \quad (1.46)$$
1.2.1 Brown’s Equations and the Effective Field

Collecting all the terms, the first-order variation of the free energy functional 1.40, which we impose to be zero, can be written as

$$\delta G = - \int_{\Omega} \left[ 2A_{\text{ex}} \nabla^2 m + 2K_1 (m \cdot u_{an}) u_{an} + \mu_0 M_s (H_d + H_{\text{ext}}) \right] \cdot \delta m \, d^3 r \quad + \int_{\partial \Omega} \left[ 2A_{\text{ex}} \frac{\partial m}{\partial n} \cdot \delta m \right] \, d^2 r = 0. \quad (1.47)$$

As we mentioned, we impose the constraint \( \| m + \delta m \| = 1 \), which can be seen as a rotation of the vector field \( m \) by an angle \( \delta \theta \),

$$\delta m = m \times \delta \theta. \quad (1.48)$$

If we consider now the vector identity \( v \cdot (w \times u) = u \cdot (v \times w) = -u \cdot (w \times v) \) and apply this to 1.47,

$$\delta G = - \int_{\Omega} m \times \left[ 2A_{\text{ex}} \nabla^2 m + 2K_1 (m \cdot u_{an}) u_{an} + \mu_0 M_s (H_d + H_{\text{ext}}) \right] \cdot \delta \theta \, d^3 r \quad + \int_{\partial \Omega} \left[ 2A_{\text{ex}} \frac{\partial m}{\partial n} \times m \right] \cdot \delta \theta \, d^2 r = 0. \quad (1.49)$$

We thus arrive at the pair of equations that must be satisfied for the integral 1.49 to vanish,

$$m \times \left[ 2A_{\text{ex}} \nabla^2 m + 2K_1 (m \cdot u_{an}) u_{an} + \mu_0 M_s (H_d + H_{\text{ext}}) \right] = 0, \quad (1.50)$$

$$2A_{\text{ex}} \frac{\partial m}{\partial n} \times m \bigg|_{\partial \Omega} = 0. \quad (1.51)$$

Since \( m \) and \( \frac{\partial m}{\partial n} \) are always orthogonal, equation 1.51 is only satisfied as long as the normal derivative of \( m \) is identically zero. As for the first equation, 1.50, we now introduce the effective field \( H_{\text{eff}} \),

$$H_{\text{eff}}(r) \equiv - \frac{1}{\mu_0 M_s} \frac{\delta G}{\delta m} = \frac{2A_{\text{ex}}}{\mu_0 M_s} \nabla^2 m + \frac{2K_1}{\mu_0 M_s} (m \cdot u_{an}) u_{an} + H_d + H_{\text{ext}}, \quad (1.52)$$

where the first two terms are essentially a field formulation for the exchange and anisotropy interactions that effectively act on the magnetization distributions the same way the demagnetizing field \( H_d \) or the external field \( H_{\text{ext}} \) do. Since we have the relation \( M = M_s m \), it is sometimes common to apply the same normalization to the effective field, \( h_{\text{eff}} = H_{\text{eff}}/M_s \). The focus here is in the normalized exchange field term, where the square root of the prefactor has dimensions of length,

$$l_{\text{ex}} = \sqrt{\frac{2A_{\text{ex}}}{\mu_0 M_s^2}}. \quad (1.53)$$

This constant is referred to as the exchange length \( l_{\text{ex}} \), and it gives an idea of the characteristic length over which the short-range exchange interaction is strongest. Typically \( l_{\text{ex}} \) is of the order of a few nanometers, depending on the material, and so one expects the magnetization to be spatially uniform on a scale of the order of \( l_{\text{ex}} \). In micromangetic simulations it is extremely important to guarantee the spatial discretization is smaller than \( l_{\text{ex}} \) so that the exchange interaction is appropriately incorporated.
With the previous considerations, equations 1.50 and 1.51 are now simplified to

\[ \mathbf{m} \times \mathbf{H}_{\text{eff}} = 0, \quad \left. \frac{\partial \mathbf{m}}{\partial \mathbf{n}} \right|_{\partial \Omega} = 0. \] (1.54)

Equations 1.54 are the so-called Brown’s equations which are used to determine the equilibrium configuration of magnetization within a ferromagnetic body. The first equation essentially states that the sum of the moments of force of the system must vanish in equilibrium. The effective field has a functional dependence on \( \mathbf{m} \), and so these are non-linear equations. However, Brown’s equations only describe the necessary condition for the system to be in equilibrium, but give no information as to how the system arrives at such state. To complete this description of the micromagnetic model, we must now introduce the dynamical equations that describe the time evolution of the magnetization under the influence of an effective field.

### 1.3 Dynamic Equations

The investigation into magnetization dynamics peaked its interest with the rapid growth of the speed and areal density requirements of magnetic storage. In order to satisfy these requirements the knowledge of equilibrium configurations of magnetization, as can be obtained with the free energy functional that we described in the previous sections, was no longer sufficient. The technological implementation of micromagnetics was thus broadened to incorporate the dynamical aspects of the magnetization, and how it evolves with time towards the equilibrium configurations. Research so far has mostly been focused on the dynamical model proposed by Landau and Lifshitz [25] in 1935, which was later modified by Gilbert [27] in 1955. In this section we will overview both approaches.

#### 1.3.1 Landau-Lifshitz Equation

Let us consider the relationship between the magnetic spin moment \( \mu \) and the angular momentum \( \mathbf{L} \) of electrons, as is known from quantum mechanics,

\[ \mu = -\gamma \mathbf{L}, \] (1.55)

where \( \gamma = 2.21 \times 10^5 \text{ m A}^{-1} \text{s}^{-1} \) is the electron gyromagnetic ratio. The classical equation of motion for angular momentum states the change in momentum is equal to the sum of torques in the system. Under a magnetic field \( \mathbf{H} \), the magnetic moment \( \mu \) feels a torque \( \mu \times \mathbf{H} \), allowing us to write

\[ \frac{d\mathbf{L}}{dt} = \mu \times \mathbf{H}. \] (1.56)

Using the relationship 1.55 we obtain an equation that describes the precession of the magnetic spin moment \( \mu \) around the field \( \mathbf{H} \),

\[ \frac{d\mu}{dt} = -\gamma \mu \times \mathbf{H}. \] (1.57)
If we now consider a large group of \( N \) elementary magnetic moments within a small volume \( \Delta V_r \) we can sum 1.57 over all the moments and do a spatial average,

\[
\frac{d}{dt} \left( \frac{1}{\Delta V_r} \sum_{i=1}^{N} \mu_i \right) = -\gamma \left( \frac{1}{\Delta V_r} \sum_{i=1}^{N} \mu_i \right) \times H.
\]

(1.58)

Thus, recalling 1.1 where we defined the continuum magnetization vector field \( \mathbf{M} \),

\[
\frac{\partial \mathbf{M}}{\partial t} = -\gamma \mathbf{M} \times \mathbf{H}.
\]

(1.59)

Equation 1.59, when one considers \( \mathbf{H} = \mathbf{H}_{\text{eff}} \) as seen in 1.52, is essentially the Landau-Lifshitz equation that was proposed in 1935. Note that, imposing a stationary condition \( \partial \mathbf{M} / \partial t = 0 \) gives the same equilibrium condition required by the first Brown equation 1.54.

In the form stated in 1.59, this equation is conservative (hamiltonian). In order to treat energy dissipations Landau and Lifshitz proposed the introduction of a phenomenological torque effectively pushing the magnetization in the direction of the effective field. The full form of the Landau-Lifshitz (LL) equation reads,

\[
\frac{\partial \mathbf{M}}{\partial t} = -\gamma \mathbf{M} \times \mathbf{H}_{\text{eff}} - \frac{\lambda}{M_s} \mathbf{M} \times (\mathbf{M} \times \mathbf{H}_{\text{eff}}),
\]

(1.60)

with the damping term characterized by the phenomenological constant \( \lambda > 0 \), which depends on the material. A scheme of the type of motion described by equation 1.60 is represented in figure 1.1.

1.3.2 The Gilbert Damping

In 1955 Gilbert used a Lagrangian formulation to derive the conservative Landau-Lifshitz equation 1.59 [27] where the generalized coordinates used were the magnetization components \( M_x, M_y \) and \( M_z \). In such a framework it is usual to introduce phenomenological damping through a "viscous" force that is proportional to the time derivative of the generalized coordinates, leading Gilbert to write:

\[
\frac{\partial \mathbf{M}}{\partial t} = -\gamma \mathbf{M} \times \mathbf{H}_{\text{eff}} + \frac{\alpha}{M_s} \mathbf{M} \times \frac{\partial \mathbf{M}}{\partial t},
\]

(1.61)

where \( \alpha \) is a small positive damping constant which also depends on the material. Equation 1.60 is usually referred to as the Landau-Lifshitz-Gilbert (LLG) equation.
Equivalence with the Landau-Lifshitz form

The LL and LLG equations 1.60 and 1.61 are very similar from a mathematical point of view. The LLG equation can, in fact, be written in the same form as the Landau-Lifshitz equation. We start by doing a cross product of $\mathbf{M}$ with 1.61,

$$\mathbf{M} \times \frac{\partial \mathbf{M}}{\partial t} = -\gamma \mathbf{M} \times (\mathbf{M} \times \mathbf{H}_{\text{eff}}) + \frac{\alpha}{M_s} \mathbf{M} \times \left( \mathbf{M} \times \frac{\partial \mathbf{M}}{\partial t} \right).$$  \hspace{1cm} (1.62)

Using the vector identity $\mathbf{u} \times (\mathbf{v} \times \mathbf{w}) = \mathbf{v} (\mathbf{u} \cdot \mathbf{w}) - \mathbf{w} (\mathbf{u} \cdot \mathbf{v})$ together with the fact that $\mathbf{M} \cdot \frac{\partial \mathbf{M}}{\partial t} = 0$ in the second term (we recall $\|\mathbf{M}\| = M_s$ must be conserved), we arrive at

$$\mathbf{M} \times \frac{\partial \mathbf{M}}{\partial t} = -\gamma \mathbf{M} \times (\mathbf{M} \times \mathbf{H}_{\text{eff}}) - \alpha M_s \frac{\partial \mathbf{M}}{\partial t}.$$  \hspace{1cm} (1.63)

This equation can be plugged back into the right hand side of 1.61 and the terms rearranged to produce the following expression:

$$\frac{\partial \mathbf{M}}{\partial t} = -\frac{\gamma}{1 + \alpha^2} \mathbf{M} \times \mathbf{H}_{\text{eff}} - \frac{\gamma \alpha}{(1 + \alpha^2) M_s} \mathbf{M} \times (\mathbf{M} \times \mathbf{H}_{\text{eff}}).$$  \hspace{1cm} (1.64)

Equation 1.64 is usually referred to as the Landau-Lifshitz equation in the Gilbert form, and it is mathematically equivalent to 1.60 as long as it is assumed that

$$\gamma' = \frac{\gamma}{1 + \alpha^2}, \quad \lambda = \frac{\gamma \alpha}{1 + \alpha^2}.$$  \hspace{1cm} (1.65)
Numerical Implementation

In the previous chapter we introduced the theoretical background of the micromagnetic model and its main interactions, with both the variational approach to the determination of equilibrium solutions as well as the dynamical aspects described by the LLG equation. The micromagnetic model, however, is rather complex, and only a handful of problems can be solved analytically. In this chapter we will discuss the general considerations of our numerical implementation of the model.

Software

Before we proceed with our discussion a quick note should be made on the choice of software that was used for this work. While most resource intensive simulations in scientific computing tend to use lower level programming languages such as Fortran or C, we chose to use the computational software Mathematica [28]. It may not be the fastest choice, however Mathematica allows complex algorithms to be developed in relatively small portions of code, thus making it a great benchmarking tool to compare numerical methods. The compact nature of the language makes the code much easier to read and modify, even though it requires some careful implementation to avoid inefficient algorithms. Furthermore, its powerful graphics engine proved very useful when visualizing data regarding magnetization and magnetic field distributions. The discussions in this thesis are still mostly independent of the choice of implementation, and details specific to Mathematica will be outlined.

2.1 Discretization

The first step towards making a continuous model suitable to be numerically treated by a digital computer is to discretize it. The micromagnetic model requires both space and time to be discretized accordingly.

Time discretization is simply the formulation that time evolves in discrete time steps, which may or may not be constant depending on the method used to treat each partial differential equation (PDE). Different types of motion, however, may prove more difficult to represent correctly along this discrete timeline. In our case, the time dependant PDE is the LLG equation, either in the form of 1.61 or 1.64 as we introduced in the previous chapter. Time integration methods for this equation will be discussed in chapter 4.

Spatial discretizations are more complex because we have to approximate a three dimensional geometric domain through a mesh or grid, and different schemes can be considered. The most common schemes
are finite-differences or finite-elements. Finite-difference schemes are centered around the approximation of the derivatives in differential equations through local Taylor Series expansions, transforming them in difference equations. Finite-element schemes divide space into smaller volumes, called elements, inside which the equations become simpler to solve. The contribution of all elements is then used to approximate the solution of the whole system. Finite-element cells can have different shapes, thus providing a more flexible mesh to discretize complex geometries, while finite-difference schemes use a cuboid grid with constant cell dimensions. In turn, finite-elements are fundamentally more complex to implement and usually have the disadvantage in terms of computational speed. Our numerical implementation of the micromagnetic model uses the finite-difference scheme, which we will proceed to introduce in more detail.

2.1.1 Finite-Differences

We now consider a three dimensional cuboid domain, which we define as \( \Omega_T \), to be discretized via a grid of small cuboid cells with dimensions \((\Delta_x, \Delta_y, \Delta_z)\). Each cuboid is thus indexed at its center with three integer values \((i, j, k)\). The first Mathematica consideration arises from the fact that, by default, array elements are enumerated starting from 1 instead of 0, as is usual in C. For a domain discretized in a total of \(N\) cells, with \((N_x, N_y, N_z)\) cells in the \((x, y, z)\) directions, the indices \((i, j, k)\) in Mathematica run from 1 to \(N_x, N_y, N_z\), respectively. Thus, considering the origin to be the center of the first corner cell, indexed as \((1, 1, 1)\), the position vector can be defined as

\[
r_{i,j,k} = (i-1)\Delta_x e_x + (j-1)\Delta_y e_y + (k-1)\Delta_z e_z,
\]

with \(\{e_x, e_y, e_z\}\) the unit vectors in the \((x, y, z)\) directions. The magnetic domain \(\Omega\) is considered to be inside the discretized domain such that \(\Omega \subseteq \Omega_T\), however equality between the two may not always be the case. We may consider, for example, a discretized region around \(\Omega\) that acts as exterior space, or we may include non-magnetic materials in our simulation, such as spacer layers, for other purposes (see for example figure 3.1 in chapter 3).

Differential Operators

As we mentioned, the central part of this scheme is the treatment of derivatives through finite-differences. So, for a certain unidimensional function \(f(x)\) discretized in steps of size \(h\), the derivatives related to positions \(x\) and \(x \pm h\), corresponding to the cells with index \(i\) and \(i \pm 1\), are given by

\[
\frac{\partial f(x + \frac{h}{2})}{\partial x} \approx \frac{f(x + h) - f(x)}{h} + O(h) = \frac{f_{i+1} - f_i}{h} + O(h) \\
\frac{\partial^2 f(x)}{\partial x^2} \approx \frac{f(x + h) - 2f(x) + f(x - h)}{h^2} + O(h^2) = \frac{f_{i+1} - 2f_i + f_{i-1}}{h^2} + O(h^2),
\]

where the local truncation error\(^1\) has a linear and squared variation with the step size, respectively. Note that the first derivative approximation in 2.2 is a forward-shifted center-difference in the sense that, since

\(^1\)The error caused by the truncation of an infinite series or one iteration of a certain iterative numerical approximation.
it requires the values at positions $i$ and $i + 1$, the obtained derivative is considered to be evaluated for the midway position $i + \frac{1}{2}$, corresponding to $f(x + \frac{h}{2})$, which must be stored in a different grid shifted by $h/2$. The second derivative 2.3 uses cells $i + 1$, $i$ and $i - 1$, and so the value is stored in cell $i$ of the same grid.

The finite-differences in 2.2 and 2.3 can be generalized to three dimensions in terms of $(i, j, k)$, thus allowing us to discretize the differential nabla operator which we use to calculate the gradient, divergence, curl, and laplacian,

\[ \nabla = \left( \frac{\partial}{\partial x}, \frac{\partial}{\partial y}, \frac{\partial}{\partial z} \right), \tag{2.4} \]

\[ \nabla f = \left( \frac{\partial f}{\partial x}, \frac{\partial f}{\partial y}, \frac{\partial f}{\partial z} \right), \tag{2.5} \]

\[ \nabla \cdot \mathbf{u} = \frac{\partial u_x}{\partial x} + \frac{\partial u_y}{\partial y} + \frac{\partial u_z}{\partial z}, \tag{2.6} \]

\[ \nabla \times \mathbf{u} = \left( \frac{\partial u_z}{\partial y} - \frac{\partial u_y}{\partial z}, \frac{\partial u_x}{\partial z} - \frac{\partial u_z}{\partial x}, \frac{\partial u_y}{\partial x} - \frac{\partial u_x}{\partial y} \right), \tag{2.7} \]

\[ \nabla^2 f = \frac{\partial^2 f}{\partial x^2} + \frac{\partial^2 f}{\partial y^2} + \frac{\partial^2 f}{\partial z^2}. \tag{2.8} \]

Keeping in mind the grid change imposed by first derivatives, when discretizing physical systems that require the use of these operators it is imperative that the discretization remains consistent so that relations between each operator may still be conserved in their discrete application. This is a common problem when discretizing Maxwell’s equations, for example, although for the case of a cuboid grid it can be solved by organizing the scalar quantities and each component of the vectorial quantities of the system in different grid locations, as we will see in the numerical implementation of the demagnetizing field.

### 2.2 The Discrete Micromagnetic Model

In this section we will go through the important equations of the micromagnetic model that we introduced in the previous chapter and discuss some details of their implementation under the finite-difference scheme. Since all static interactions must come together under the Landau-Lifshitz-Gilbert equation 1.64, we will start our discussion on the numerical implementation of the micromagnetic interactions by the spatial discretization of the LLG equation. For simplicity we consider here our discretized domain is fully magnetic such that $\Omega_T = \Omega$.

#### 2.2.1 Landau-Lifshitz-Gilbert Equation

Both forms of the LLG equation, the time implicit 1.61 and the explicit 1.64 follow the same reasoning for spatial discretization, and so we will proceed our discussion from the explicit form,

\[ \frac{\partial \mathbf{M}}{\partial t} = -\frac{\gamma}{1 + \alpha^2} \mathbf{M} \times \mathbf{H}_{\text{eff}} - \frac{\gamma\alpha}{(1 + \alpha^2)M_s} \mathbf{M} \times (\mathbf{M} \times \mathbf{H}_{\text{eff}}). \tag{2.9} \]
The spatial discretization of equation 2.9 consists in discretizing the continuous fields $M(r)$ and $H_{\text{eff}}(r)$. As we already mentioned, this equation verifies $\|M(r, t)\| = M_s$, and so we must be able to verify this condition after discretization. We must also be able to perform the mathematical cross product between $M$ and $H_{\text{eff}}$ at each location in space. This two considerations require $M$ and $H_{\text{eff}}$ to be discretized on the same grid without separating the three components of each vector field. We thus define our main discretization of $\Omega$ as a grid of cuboid cells where each cell has the vectors $M_{i,j,k}$ and $(H_{\text{eff}})_{i,j,k}$ at its center, which in some way represent the average magnetization and effective field for each cell. This is the usual construction under which the micromagnetic model is discretized, and it allows $(M \times H_{\text{eff}})_{i,j,k}$ to be computed for each cell, as well as the condition $\|M_{i,j,k}\| = M_s$ to be verified at each location in space. We will call this discretization scheme the center-vector grid, as exemplified in figure 2.1.

The energy conservations of the LLG equation will be discussed in chapter 4 as they are related to the time integration aspects.

2.2.2 Exchange Interaction

Regarding the exchange interaction introduced in the previous chapter, we must discretize the free energy expression 1.27 and the exchange contribution to the effective field 1.52,

$$F_{\text{ex}} = \int_{\Omega} A_{\text{ex}} (\nabla m)^2 \, d^3r, \quad H_{\text{ex}} = \frac{2A_{\text{ex}}}{\mu_0 M_s} \nabla^2 m. \quad (2.10)$$

As discussed in literature [29, 30, 31], various formulations can be considered when discretizing the exchange interaction. We start by reformulating the continuous exchange energy term [31],

$$F_{\text{ex}} = -\int_{\Omega} A_{\text{ex}} (m \cdot \nabla^2 m) \, d^3r. \quad (2.11)$$

We thus start by discretizing the exchange field by applying the usual 7 point stencil to the Laplace
operator, which is simply the three dimensional extension of 2.3,

\[
(H_{ex})_{i,j,k} = \frac{2A_{ex}}{\mu_0 M_s} \left[ \frac{m_{i+1,j,k} + m_{i-1,j,k}}{\Delta x^2} + \frac{m_{i,j+1,k} + m_{i,j-1,k}}{\Delta y^2} + \frac{m_{i,j,k+1} + m_{i,j,k-1}}{\Delta z^2} - m_{i,j,k} \left( \frac{2}{\Delta x^2} + \frac{2}{\Delta y^2} + \frac{2}{\Delta z^2} \right) \right].
\] (2.12)

As discussed in [31], the six neighbour approximation already provides good results and is simpler to implement. This leads us to discretize the free energy integral 2.11 as

\[
F_{ex} = -\frac{1}{2} \mu_0 M_s \sum_{i,j,k} (m \cdot H_{ex})_{i,j,k} \Delta x \Delta y \Delta z.
\] (2.13)

From expression 2.12 we note the exchange interaction requires boundary conditions to be defined around the discretized magnetic region. This is done through the introduction of a ghost layer of cells around the boundary cells of \( \Omega \) which we consider to be filled with the same vectors of magnetization as the boundary cells, thus effectively imposing the Neumann boundary condition \( \frac{\partial m}{\partial n} = 0 \) \( \mid \partial \Omega \). Given their short-range dependence on the magnetization, 2.12 and 2.13 have a computational complexity\(^2\) of \( \mathcal{O}(N) \).

2.2.3 Anisotropy Interaction

The anisotropy interaction has a free energy contribution given by equation 1.31 and a field contribution to the effective field 1.52,

\[
F_{an} = -\int_\Omega K_1 (m \cdot u_{an})^2 d^3r, \quad H_{an} = \frac{2K_1}{\mu_0 M_s} (m \cdot u_{an}) u_{an}.
\] (2.14)

These quantities have no dependence on magnetization derivatives, and are thus straightforward to discretize,

\[
F_{an} = -K_1 \sum_{i,j,k} (m_{i,j,k} \cdot u_{an})^2 \Delta x \Delta y \Delta z,
\] (2.15)

\[
(H_{an})_{i,j,k} = \frac{2K_1}{\mu_0 M_s} (m_{i,j,k} \cdot u_{an}) u_{an}.
\] (2.16)

The computation of 2.15 and 2.16 also have a complexity of \( \mathcal{O}(N) \).

2.2.4 Demagnetizing Interaction

The discretization of the exchange and anisotropy terms we presented is consistent with the spatial discretization requirements of the LLG equation. The demagnetizing interaction, on the other hand, is more troublesome. As we saw in the previous chapter, this interaction depends on the demagnetizing field, which must be calculated according to Maxwell’s equations 1.34,

\[
\nabla \cdot H_d = -\nabla \cdot M, \quad \nabla \times H_d = 0.
\] (2.17)

\(^2\)The time required for their calculation increases linearly with the total number of cells \( N \) in the simulation.
Since $\mathbf{H}_d$ has zero curl, we can write it as the gradient of a certain scalar potential, $\mathbf{H}_d = -\nabla V$. This is simply a mathematical scalar potential with no special physical meaning, but it is useful to simplify the problem. We will also introduce a fictitious magnetic charge density, $\rho_M(\mathbf{r}) = -\nabla \cdot \mathbf{M}$. With these two definitions we can rewrite equations 2.17 as a single Poisson equation,

$$\nabla^2 V(\mathbf{r}) = -\rho_M(\mathbf{r}). \quad (2.18)$$

These equations represent a global interaction, and their solution is the most resource intensive static calculation in micromagnetic modelling. We will discuss this problem in detail in the next chapter, where we will start by comparing equation 2.18 with the well-known, and mathematically equivalent, electrostatic potential boundary value problem. For now let us simply analyze how the relations from equations 2.17 and 2.18 behave on a discrete grid.

When we consider $\mathbf{M}$ and $\mathbf{H}_d$ discretized according to the center-vector grid formulation, as shown in figure 2.1, it is easy to see the application of the divergence operator 2.6 in its finite-difference form becomes problematic: the first derivatives of each component must be stored on different surfaces of the grid. For example, $\frac{\partial M_x}{\partial x}$ has to be calculated as a finite-difference between cells $i$ and $i + 1$ for fixed $j$ and $k$ and it is stored in the position $(i + \frac{1}{2}, j, k)$, while $\frac{\partial M_y}{\partial y}$ has to be calculated between cells $j$ and $j + 1$ for fixed $i$ and $k$ and it is stored in the position $(i, j + \frac{1}{2}, k)$. The same reasoning applies for the $z$ component, and thus the three derivatives cannot be directly summed as a localized divergence. A similar problem arises with the cross-derivatives when we apply the curl operator 2.7.

In order to verify Maxwell’s equations on a discrete cuboid grid each component of $\mathbf{M}$ and $\mathbf{H}_d$ should in fact be discretized along the borders of the grid, as illustrated in figure 2.2. This is the usual approach in the Finite-Difference Time-Domain (FDTD) method used in the field of computational electrodynamics, as described initially by Yee [32]. In this construction, which we refer to as the conjugate-dual grid, the three derivatives from the divergence operator are evaluated at the vertices of the cells, which we call the conjugate grid since it has a $(\Delta x, \Delta y, \Delta z)$ shift from the main grid, and the cross-derivatives from the curl operator are evaluated at the surfaces of the main grid. The discretization of the Poisson
equation 2.18 is also consistent. Since the scalar $\rho_M$ is the divergence of $\mathbf{M}$, it is stored in the conjugate grid centered at the vertices of the main grid. Equation 2.18 requires only the application of the Laplace operator 2.8, and since second derivatives require no grid change, the scalar potential $V$ is also discretized on the conjugate grid. Finally, when we apply the gradient operator 2.5 to the scalar potential $V$, the finite-difference reasoning of the first derivatives lead the resulting field $\mathbf{H}_d$ to be again organized in the borders of the main grid, according to figure 2.2.

The conjugate-dual grid construction allows us to study and develop numerical methods to solve the demagnetizing field while focusing on the conservation of Maxwell’s relations. This means that, after applying a numerical method to solve the equations, we can actually verify if the discrete divergence of the field is equal to the discrete divergence of the magnetization and if the discrete curl of the field is null with acceptable numerical precision. However, as we have seen, this discretization is not consistent with the requirements of the LLG equation, and the demagnetizing field contribution to the effective field must still be accounted for in the center-vector grid discretization from figure 2.1. In order to conciliate the two discretizations the approach used was to average the components of the magnetization distribution $\mathbf{M}$ from the center to the borders of the grid before treating the demagnetizing field, and then average the resulting $\mathbf{H}_d$ field components from the borders to the center to account for its contribution to the whole model. On one hand we can guarantee the numerical methods we use conserve Maxwell’s relations 2.17, but on the other we cannot guarantee there is no information loss when we average field components from one discretization scheme to the other. A different approach to this problem was not studied but will be referenced for future work.

The most common way to address the discretization problem of the demagnetizing field in micromagnetic modelling is, in fact, to calculate it considering only the center-vector grid using a tensor convolution method directly between $\mathbf{M}$ and $\mathbf{H}_d$. This method has been validated by comparison with analytical solutions and experimental results [33]. Averaging techniques between grids have also been implemented in numerical methods to calculate the demagnetizing field with good results, through either the inclusion of the magnetic charge distribution, magnetic potential, or both [33, 34, 35]. However, none of these publications give any special attention dedicated to the conservation of the Maxwell’s equations in their discrete form, also validating their results by comparison with analytical solutions and experimental data. The demagnetizing field problem and its numerical solution will be treated in chapter 3.

The free energy contribution of the demagnetizing interaction is given by equation 1.37, which is straightforward to discretize in the center-vector grid, with a complexity of $O(N)$, as

$$
F_{\text{demag}} = -\frac{1}{2} \mu_0 \sum_{i,j,k} (\mathbf{M} \cdot \mathbf{H}_d)_{i,j,k} \Delta_x \Delta_y \Delta_z.
$$

(2.19)

**Final remark on discretization grids**

Considering the grid relations we have just discussed, we recall that our main discretization is considered to be center-vector grid, where we use the notation $N = N_x + N_y + N_z$ as the total number of cells. The conjugate-dual grid discretization also has a total of $N$ cells, but here the vertices are considered to be
the center of the conjugate grid, which has a total of \( N' = N'_x + N'_y + N'_z \), where \( N'_{x,y,z} = N_{x,y,z} + 1 \).

### 2.2.5 Zeeman Interaction

For a given externally applied field \( \mathbf{H}_{\text{ext}} \) discretized in the center-vector grid, the Zeeman energy contribution from equation 1.38 is the same as the demagnetizing contribution, and can be discretized as

\[
G_{\text{ext}} = -\sum_{i,j,k} \mu_0 (\mathbf{M} \cdot \mathbf{H}_{\text{ext}})_{i,j,k} \Delta x \Delta y \Delta z,
\]

(2.20)

### 2.3 Natural Units

So far we have made a few remarks related to the units of physical quantities within the micromagnetic model, most notably how the magnetization field \( \mathbf{m}(\mathbf{r}) \) is the same as \( \mathbf{M}(\mathbf{r}) \) but in units of the saturation magnetization \( M_s \). We also recall from section 1.2.1 the introduction of the exchange length 1.53 which limits the dimensions of the simulation cells. In physical models it is usual to include a set of natural units which are specially adapted to the intrinsic scales of each problem and can be achieved by setting certain physical quantities to unity. Their use in simulations has a couple of advantages:

- First, having everything scaled according to the problem avoids having to deal with very large or very small quantities, which can sometimes lead to numerical errors.
- Second, when looking at numerical results, determining their physical significance becomes much more convenient when they are defined in terms of the physical scale of the problem. One example in micromagnetics is verifying the discretization satisfies the exchange length constraint. The exchange length effectively defines the spatial scale in micromagnetics, but it does not have a fixed value in non-natural units because it depends on the material parameters.

While the discussion of the model so far was done in non-natural units, including its discretization, its implementation into *Mathematica* was done by setting the following quantities to unity:

<table>
<thead>
<tr>
<th>Quantity</th>
<th>Description</th>
<th>SI Units</th>
</tr>
</thead>
<tbody>
<tr>
<td>( M_s )</td>
<td>Saturation magnetization</td>
<td>Material dependent, ((\text{A m}^{-1}))</td>
</tr>
<tr>
<td>( A_{\text{ex}} )</td>
<td>Exchange constant</td>
<td>Material dependent, ((\text{J m}^{-1}))</td>
</tr>
<tr>
<td>( \gamma )</td>
<td>Gyromagnetic ratio</td>
<td>((2.21 \times 10^5 \text{ m A}^{-1}\text{s}^{-1}))</td>
</tr>
<tr>
<td>( \mu_0/2 )</td>
<td>Vacuum permeability</td>
<td>((\frac{4\pi}{2} \times 10^{-7} \text{ N A}^{-2}))</td>
</tr>
</tbody>
</table>

Table 2.1: Physical quantities normalized during implementation. The \( 1/2 \) factor in \( \mu_0 \) is simply to keep the natural unit of length consistent with the conventional exchange length definition 1.53.

These normalizations lead the simulation units to be defined as the following set of natural units:

<table>
<thead>
<tr>
<th>Length</th>
<th>Field</th>
<th>Time</th>
<th>Energy</th>
</tr>
</thead>
<tbody>
<tr>
<td>( \sqrt{2A_{\text{ex}}/\mu_0 M_s^2} )</td>
<td>( M_s )</td>
<td>( 1/\gamma M_s )</td>
<td>( A_{\text{ex}} \sqrt{2A_{\text{ex}}/\mu_0 M_s^2} )</td>
</tr>
</tbody>
</table>

Table 2.2: Natural units used during the simulation.
Demagnetizing Field

In the previous chapters we introduced the micromagnetic model and explained some of the details of its implementation. Now we must tackle more specific and complex problems, starting with the calculation of the magnetostatic self-interaction, or demagnetizing field.

Being a global interaction, the demagnetizing field is by far the most resource intensive static calculation — the value in each cell of discretized space has a contribution of every single cell in the material. If we were to calculate it by brute force for a total of $N$ cells, the time needed would scale with $O(N^2)$. This is not viable, especially considering that when we are dealing with magnetization dynamics the effective field terms must be constantly updated to accommodate the changes in the magnetization.

As such, finding ways to optimize this calculation is of great interest in the field of micromagnetics, and has for a long time been the focus of many studies. Currently, most simulations perform this calculation through a Fast Fourier Transform (FFT) accelerated convolution between a certain demagnetizing kernel\(^1\) and the magnetization distribution. This idea started with the work of Schabes and Aharoni [36], who suggested the kernel could be seen as the interaction field for an array of uniformly magnetized cubes, which could be derived analytically.

In this chapter we will start with a very straightforward linear system approach to the problem based on the direct discretization the Poisson equation with a far away boundary. This is an introductory computational approach to the problem, and it also proved to be a very reliable method, but rather inefficient when compared to FFT methods. We will then focus on the convolution method, discussing the types of kernels that can be considered. We will conclude the chapter by proposing a new numerical kernel that conserves Maxwell’s equations in the conjugate-dual grid discretization scheme. Our method also reduces the number of FFT calculations needed to compute the demagnetizing field when compared to other FFT methods, thus making it more efficient.

3.1 Underlining the Problem

Let us start by rewriting the equations we need to solve, as they will be the focus of this chapter. We recall we are now working with the natural units we introduced in section 2.3, and so we use the notation

\(^1\)A kernel is a quantity that is preprocessed according to the desired output of a problem, storing results that only need to be computed once. The information stored in the kernel describes how the input should be molded into the desired output of the problem, and so further computations use this information to reach the solution more efficiently.
of the normalized fields $\mathbf{m}$ and $\mathbf{h}_d$,

$$
\nabla \cdot \mathbf{h}_d = - \nabla \cdot \mathbf{m} \quad \nabla \times \mathbf{h}_d = 0.
$$

(3.1)

Given a space dependent magnetization distribution $\mathbf{m}(\mathbf{r})$ we must find the demagnetizing field $\mathbf{h}_d(\mathbf{r})$ that satisfies these two Maxwell equations at every point in space. As we mentioned in the previous chapter, we introduce a fictitious magnetic charge density $\rho_m(\mathbf{r}) = - \nabla \cdot \mathbf{m}$ and the mathematical potential $V(\mathbf{r})$, where $\mathbf{h}_d = - \nabla V$, so that we can rewrite equations 3.1 as a single Poisson equation,

$$
\nabla^2 V(\mathbf{r}) = - \rho_m(\mathbf{r}).
$$

(3.2)

Transforming $\mathbf{m}$ into $\rho_m$ and $V$ into $\mathbf{h}_d$ are direct applications of the divergence and gradient operators, which scale with $O(N)$. We are left with transforming $\rho_m$ into $V$, and so our task is set: we must find an efficient and reliable solver for equation 3.2.

This equation is, in fact, mathematically equivalent to the electrostatic potential boundary value problem, which has been extensively studied. The solution to 3.2 can be written as

$$
V(\mathbf{r}) = \frac{1}{4\pi} \int_{\Omega} \frac{1}{||\mathbf{r} - \mathbf{r}'||} \rho_m(\mathbf{r}') \, d^3 r'.
$$

(3.3)

This integral solution has important convolution properties which we will discuss later. The difference from the electrostatic problem arises from the fact that whenever we talk about charges or a charged material, the magnetic charges we are referring to are simply a mathematical construction from the divergence of the magnetization, leading to some discussion on how the surface of the magnetized material should be treated. The usual approach is to assume the distribution $\mathbf{m}(\mathbf{r})$ inside a magnetized region $\Omega$ falls suddenly to zero at the surface $\partial \Omega$. Such a perfectly defined boundary would lead to an infinite divergence at the surface, requiring it to be treated with a different construction. As discussed in Jackson [19], applying the divergence theorem to $\rho_m(\mathbf{r})$ for a cylindrical gaussian surface straddling the surface of the magnetized material gives an effective surface charge density $\sigma_m = \mathbf{n} \cdot \mathbf{m}$, where $\mathbf{n}$ is the outwards normal. An extra term is then included in the scalar potential to represent the surface contribution,

$$
V(\mathbf{r}) = \frac{1}{4\pi} \int_{\Omega} \frac{\rho_m(\mathbf{r}')}{||\mathbf{r} - \mathbf{r}'||} \, d^3 r' + \frac{1}{4\pi} \oint_{\partial \Omega} \frac{\sigma_m(\mathbf{r}')}{||\mathbf{r} - \mathbf{r}'||} \, d^2 r'.
$$

(3.4)

This approach, however, does not have any physical basis since we are not actually dealing with a charged material. Instead of assuming a well defined boundary with an abrupt drop in magnetization, we considered the surface to be an uncertainty region over which the magnetization changes, with its thickness effectively defined by the cell size used in the discretization, disregarding any “special” surface charges. We know there is no actual discontinuity in a physical magnetization distribution, and as long as the field $\mathbf{h}_d$ obtained conserves the discrete Maxwell equations in the system, we know the mathematical solution obtained for $V$ is reliable. Discrete conservation of 3.1 should be the main priority, and the extra surface term in 3.4 should not be needed for this.
3.2 Direct Poisson Discretization

Let us now start our approach to equation 2.18 by discretizing the Laplace operator directly. We recall that we are now working with the conjugate grid, as discussed in the previous chapter. The simplest discrete approximation is to write this operator as a 7 point stencil,

\[
\frac{V_{i+1,j,k} + V_{i-1,j,k}}{\Delta_x^2} + \frac{V_{i,j+1,k} + V_{i,j-1,k}}{\Delta_y^2} + \frac{V_{i,j,k+1} + V_{i,j,k-1}}{\Delta_z^2} - V_{i,j,k} \left( \frac{2}{\Delta_x^2} + \frac{2}{\Delta_y^2} + \frac{2}{\Delta_z^2} \right) = -\rho_{i,j,k}. \tag{3.5}
\]

Note we have dropped the \( m \) subscript in \((\rho_m)_{i,j,k}\) to simplify the notation. Equation 3.5 is the linearized Poisson equation that relates the potential in the cell \((i,j,k)\) with its 6 nearest neighbours. While some physical systems can accommodate higher order approximations of the Laplace operator with more neighbours, these are only viable if there is certainty in how smooth the solution is. In this case, while we can expect \( V \) to be smooth far away from, or even inside \( \Omega \), that will no longer be certain in simulation cells closer to \( \partial \Omega \).

Finding the potential in all the simulation cells is now a question of solving a linear system of equations of the type of 3.5. Before we get to that, however, we must take into account the boundary conditions that must be satisfied. Note that we are referring to the boundary of the problem, and not necessarily the boundary of the charged material.

3.2.1 Boundary Conditions

The most common boundary condition used to solve Poisson’s equation is the open boundary condition that the potential decays to zero at infinity. Working with discretized space, it is quite impossible to create an infinite grid and then set the potential of the “boundary” cells to 0. We can, however, choose finite boundaries and calculate their potential using a multipole expansion. These only need to be far enough from the material for the expansion to be valid. We used the multipole expansion for the electrostatic potential in Cartesian coordinates, as seen in Jackson [19],

\[
V(r) = \frac{1}{4\pi} \left[ \frac{q}{\|r\|} + \frac{p \cdot r}{\|r\|^3} + \frac{1}{2} \sum_{\alpha,\beta=x,y,z} Q_{\alpha\beta} \frac{r_\alpha r_\beta}{\|r\|^5} + \ldots \right], \tag{3.6}
\]

where we only included the first three moments in our approximation. The monopole moment is simply the total charge \( q \), while \( p \) is the dipole moment vector and \( Q_{\alpha\beta} \) is the traceless quadrupole moment tensor:

\[
p = \int r' \rho_m(r') \, d^3r', \tag{3.7}
\]

\[
Q_{\alpha\beta} = \int \left(3r'_\alpha r'_\beta - \|r'\|^2 \delta_{\alpha\beta}\right) \rho_m(r') \, d^3r'. \tag{3.8}
\]

As we’ve mentioned, we have to consider sufficient exterior space outside the charged material for the three moment expansion to be valid. Instead of calculating the three moments for the whole charged material however, we divided the material in chunks, each having its own monopole, dipole and quadrupole
moment. This allowed us to put the boundaries closer to the material, thus minimizing the discretized space $\Omega_T$. According to Rocha [11], using an exterior space in every direction equal to the length of the largest chunk, the three moment expansion gives an error of less than 1% compared to direct integration. Furthermore, since the potential at the far away boundaries is a smooth varying function, we only calculated the contributions from each chunk in certain boundary cells, leaving the rest to be interpolated. All of these considerations are presented in a simplified 2D example in figure 3.1. The more chunks we consider, the closer the boundaries can be to the material. This increases the time needed to calculate the multipole expansions, but decreases the size of the linear system that must be solved afterwards.

An extra parameter was considered for each chunk, which we can call the center of charge,

$$
R_{cm} = \frac{\int r' \rho_m(r') \, d^3r'}{\int \rho_m(r') \, d^3r'} = \frac{P}{q}.
$$

(3.9)

For non neutral chunks this center of charge was used as the reference point for the evaluation of integrals 3.7 and 3.8. For neutral chunks, where $q = 0$, or whenever 3.9 diverged to the exterior of the chunk, the geometrical center was considered instead.

### 3.2.2 Constructing the Linear System

Having calculated the potential for all the boundary cells, we must write the linear system that solves the potential for the interior region. For a total of $N'$ cells with $(N'_x, N'_y, N'_z)$ cells in the $(x, y, z)$ direction, respectively, the linear system will have $(N'_x - 2)(N'_y - 2)(N'_z - 2)$ equations and variables:

$$
Au = b, \quad A \in \mathbb{R}^{n \times n} : n = (N'_x - 2)(N'_y - 2)(N'_z - 2).
$$

(3.10)
According to equation 3.5, $A$ is the block diagonal matrix form of the 7 point discrete Laplace operator, $u$ is the list of the undetermined $V_{i,j,k}$ variables and $b$ includes both the values of $\rho_{i,j,k}$ and the potential values at the boundaries. Creating this linear system is the usual procedure to solve the discrete Poisson equation, but its description in literature is usually limited to a 2D example with constant grid dimensions in both directions [37], for the sake of simplicity. This is useful to understand the problem and can be generalized to three dimensions. Below we present the more general procedure we used to construct matrix $A$ and vector $b$ for three dimensions with cell dimensions $(\Delta_x, \Delta_y, \Delta_z)$.

\[
A = \begin{pmatrix}
B & A_2 \\
A_2 & \ddots \\
& \ddots & A_2 \\
A_2 & B & \end{pmatrix} \Rightarrow \begin{cases}
B \text{ is repeated } N_z' - 2 \text{ times.} \\
(B, A_2) \in \mathbb{R}^{n \times n} : n = (N_y' - 2)(N_z' - 2).
\end{cases}
\]

\[
B = \begin{pmatrix}
C & B_2 \\
B_2 & \ddots \\
& \ddots & B_2 \\
B_2 & C & \end{pmatrix} \Rightarrow \begin{cases}
C \text{ is repeated } N_y' - 2 \text{ times.} \\
(C, B_2) \in \mathbb{R}^{n \times n} : n = (N_z' - 2).
\end{cases}
\]

\[
C = \begin{pmatrix}
- \left( \frac{2}{\Delta_x^2} + \frac{2}{\Delta_y^2} + \frac{2}{\Delta_z^2} \right) & \frac{1}{\Delta_x^2} & \cdots & \frac{1}{\Delta_x^2} \\
\frac{1}{\Delta_x^2} & \ddots & \ddots & \frac{1}{\Delta_x^2} \\
& \ddots & \ddots & \ddots \\
\frac{1}{\Delta_x^2} & \cdots & \frac{1}{\Delta_x^2} & - \left( \frac{2}{\Delta_x^2} + \frac{2}{\Delta_y^2} + \frac{2}{\Delta_z^2} \right)
\end{pmatrix}. 
\]

(3.11)

In our Mathematica implementation, Matrix $A$ was constructed going backwards from $C \rightarrow B \rightarrow A$ with the help of the KroneckerProduct function. In order to construct vector $b$ we first considered a three dimensional grid $B$ with dimensions $(N_x' - 2, N_y' - 2, N_z' - 2)$ such that

\[
b_{i,j,k} = -\rho_{i,j,k},
\]

(3.12)

with $(i, j, k)$ running from 2 to $N_x', N_y', N_z' - 1$. This implies that this method does not consider the charge density in the boundary cells. This is fine, however, since our boundaries must be far away from the charged material. As we’ve mentioned $b$ must also include the boundary conditions. Looking back at equation 3.5 we see that this can be done by subtracting the respective boundary value of $V$ whenever $(i, j, k)$ are side by side with a boundary cell (see gray cells in figure 3.1):

\[
(B_{j,k})_{i=2,N_x'-1} = - \left[ (\rho_{j,k})_{i=2,N_x'-1} + \left( \frac{V_{j,k}}{\Delta_x^2} \right)_{i=1,N_x'} \right],
\]

\[
(B_{i,k})_{j=2,N_y'-1} = - \left[ (\rho_{i,k})_{j=2,N_y'-1} + \left( \frac{V_{i,k}}{\Delta_y^2} \right)_{j=1,N_y'} \right],
\]

\[
(B_{i,j})_{k=2,N_z'-1} = - \left[ (\rho_{i,j})_{k=2,N_z'-1} + \left( \frac{V_{i,j}}{\Delta_z^2} \right)_{k=1,N_z'} \right].
\]

(3.13)
We concluded by using Mathematica’s Flatten to transform three dimensional array $B$ into a unidimensional vector, $b$. We can see $A$ only depends on the dimensions of the cells, and so it only needs to be calculated once for each geometry — this is the first example of a *kernel*. On the other hand, $b$ depends on $\rho_m$ and so the linear system 3.10 must be solved every time the charge distribution changes, along with updating the boundary conditions.

### 3.2.3 Solving the Linear System

Considering this type of problem can easily venture into discretizations of the order of hundreds of cells in each direction, it can lead to rather huge linear systems. The advantage is that matrix $A$ is sparse, symmetric, and diagonally dominant. These are important characteristics that linear solvers can exploit to optimize their performance.

There is a very well known library of low-level routines to apply basic linear operations called BLAS – *Basic Linear Algebra Subprograms* [38]. This library originated in 1979, and since then it has received constant updates, now supporting sparse matrices. It also spawned a lot of solvers that use these routines to perform more advanced operations such as solving linear systems and eigenvalue problems. A well known example is LAPACK [39], however other options like MAGMA [40] are still being developed.

Intel’s *Math Kernel Library* (MKL) [41] is a library that includes low level routines like BLAS, linear solvers like LAPACK, sparse solvers, and also other math routines like FFTs. Furthermore, these routines are specifically optimized to run on Intel processors, which are the most commonly used processors. The fact that the algorithms in these libraries are cache-aware also provides much more efficient memory management, reducing the amount of information transferred between RAM and cache memory — this can sometimes make codes orders of magnitude faster. *Mathematica* supports MKL natively, and this was also an important factor when considering the software choices for the implementation.

There are various iterative methods that can be used to solve large linear systems. We tested the Jacobi and the Successive Over-Relaxation (SOR) methods, which were implemented explicitly according to [37]. *Mathematica*’s function `LinearSolve` also has a couple of methods natively implemented that were tested: Krylov [42] and Pardiso [43], and changing between these is as simple as writing `Method->Pardiso` in the arguments of `LinearSolve`, with the assurance that they will be properly optimized. We will also discuss a spectral method that we included in this section for comparison purposes.

#### Jacobi Method

Matrix $A$ is written as

$$A = D + L + U,$$

(3.14)

where $D$ is the diagonal matrix, $L$ is the lower triangular matrix and $U$ is the upper triangular matrix. The system is solved iteratively via

$$u^{(k+1)} = D^{-1} \left[ b - (L + U)u^{(k)} \right].$$

(3.15)
Matrix $D$ is fully diagonal so it is easily invertible. An initial guess for $u^{(0)}$ must be given and, ideally, the stopping criteria should be
\[
\left\| Au^{(k)} - b \right\| < \epsilon
\]
for some small $\epsilon$, but since these quantities are not part of the Jacobi iterations, calculating $\left\| Au^{(k)} - b \right\|$ for every iteration would slow down the method, and so $\left\| u^{(k)} - u^{(k-1)} \right\| < \epsilon'$ was used instead.

**Successive Over-Relaxation Method**

Using the same considerations for matrix $A$, the system is solved iteratively via
\[
u^{(k+1)} = \nu^{(k)} - \omega(D + \omega L)^{-1} \left[ (L + D + U)\nu^{(k)} - b \right],
\]
where $\omega$ is called the over-relaxation parameter. This method is convergent for $0 < \omega < 2$ and usually converges faster than the Jacobi method for $1 < \omega < 2$, with the downside that it is not always obvious what the best value for $\omega$ is. Every iteration requires the solution of another linear system,
\[
[\omega(D + \omega L)] \nu^* = \left[ (L + D + U)\nu^{(k)} - b \right].
\]
This is a triangular system, and we found that using Mathematica’s `LinearSolve` with the Pardiso method was the fastest way to solve this type of system. The stopping criteria 3.16 was used.

**Spectral Method**

Up to now we have discussed methods to solve linear system 3.5, but our real goal is to solve Poisson’s equation 3.2, and this can also be done through a simple FFT based method. We start by writing $\rho_m$ and $V$ in terms of their reciprocal space equivalents:
\[
\rho_m(r) = \sum_G \tilde{\rho}_m(G) e^{iG \cdot r}, \quad V(r) = \sum_G \tilde{V}(G) e^{iG \cdot r}.
\]

With a finite difference discretization $r$ can be written as a linear contribution with integer coefficients of the primitive lattice vectors, $r = n_1 a_1 + n_2 a_2 + n_3 a_3$, and we obtain a similar construction for the reciprocal lattice $G = m_1 b_1 + m_2 b_2 + m_3 b_3$, where
\[
\begin{align*}
(b_1) &= 2\pi \frac{a_2 \times a_3}{a_1 \cdot [a_2 \times a_3]} \quad \Rightarrow \quad \text{and we define} \quad \begin{cases} a_1 = (\Delta x, 0, 0) \\ a_2 = (0, \Delta y, 0) \\ a_3 = (0, 0, \Delta z) \end{cases} \\
(b_2) &= 2\pi \frac{a_3 \times a_1}{a_2 \cdot [a_3 \times a_1]} \\
(b_3) &= 2\pi \frac{a_1 \times a_2}{a_3 \cdot [a_1 \times a_2]}
\end{align*}
\]
Plugging the definitions 3.19 in equation 3.2 we get

\[ \nabla^2_r \left( \sum_G \tilde{V}(G) e^{iG \cdot r} \right) = -\sum_G \tilde{\rho}_m(G) e^{iG \cdot r} \]

(3.21)

\[ \Leftrightarrow -\sum_G \|G\|^2 \tilde{V}(G) e^{iG \cdot r} = -\sum_G \tilde{\rho}_m(G) e^{iG \cdot r} \]

(3.22)

\[ \Rightarrow \tilde{V}(G) = \frac{\tilde{\rho}_m(G)}{\|G\|^2}. \]

(3.23)

As we can see, in reciprocal space the equation is solved through a simple division operation. The kernel \(\|G\|^2\) is then a very simple squared norm calculation for each cell.

So the procedure is simple: first we calculate \(\tilde{\rho}_m(G)\),

\[ \tilde{\rho}_m(G) = \frac{1}{N} \sum_r \rho_m(r) e^{-iG \cdot r}, \]

(3.24)

then we calculate \(\tilde{V}(G)\) through equation 3.23, transforming it back into \(V(r)\) through 3.19. This method, however, is not equivalent to solving the linear system described earlier because these FFTs impose periodic boundary conditions, and so the potential we obtain includes the contribution from an infinite array of charged periodic images. There is also the issue that equation 3.23 has a singularity for \(G = (0, 0, 0)\), which requires the value of \(\tilde{V}(0, 0, 0)\) to be corrected. This value is completely arbitrary and corresponds to a constant shift in the potential. So in the end this method provides a solution that is only unique up to a constant and corresponds to different boundary conditions. The reasoning for its inclusion in this section was that it is very simple to implement and allowed us to easily benchmark the performance of FFTs in comparison to linear system methods. Knowing that currently most simulations use FFT based methods, this was important to help us decide where to proceed with our work.

**Comparing the Methods**

In order to compare the methods we need to know how their performance scales with the size of the system. We considered a cube with \(N' = n^3\) total cells, for \(n = \{10, 20, \ldots, 100\}\), and measured the time each method took to solve the system. The charge density used is irrelevant. We also did not include the time needed to compute the boundary conditions through the multipole expansion, as this is needed for all the methods and has a different scaling, and so we simply set the values of the potential at the boundary of the cube to be zero. Each calculation was repeated ten times and the average timing was used.

To measure the scaling we used a function of the type

\[ f(N) = p N^s \]

(3.25)

to fit the data obtained from the time measurements, where \(p\) is the prefactor and \(s\) is the scaling parameter. The data and the respective fitted functions are plotted in figure 3.2, with the fitting parameters shown in table 3.1.
Figure 3.2: Performance scaling of various linear system methods and a FFT method for the solution of Poisson’s equation in a charged cube made up of $N$ cells. Calculated on an Intel Core i5-3570k processor.

<table>
<thead>
<tr>
<th>Method</th>
<th>Pardiso</th>
<th>Krylov</th>
<th>Jacobi</th>
<th>SOR</th>
<th>FFT</th>
</tr>
</thead>
<tbody>
<tr>
<td>$p$</td>
<td>$4.00 \times 10^{-12}$</td>
<td>$2.58 \times 10^{-6}$</td>
<td>$4.34 \times 10^{-8}$</td>
<td>$6.87 \times 10^{-9}$</td>
<td>$2.51 \times 10^{-6}$</td>
</tr>
<tr>
<td>$s$</td>
<td>2.35</td>
<td>1.09</td>
<td>1.67</td>
<td>1.66</td>
<td>0.97</td>
</tr>
</tbody>
</table>

Table 3.1: Fitting parameters from function 3.25 to the data in figure 3.2.

The most important parameter for this type of study is the scaling, as can be seen in figure 3.2. The smaller $s$ is, the better the method performs for higher resolutions, and here we can see FFTs have the clear advantage. It is well known FFTs scale with $\mathcal{O}(N \log_2 N)$, however due to the exponential nature of function 3.25 we only see the linear part in the fitting parameters. Looking at the linear system methods, our implementations of the Jacobi and SOR methods seem to float around $\mathcal{O}(N^{3/3})$ scaling, but the SOR method does have a faster convergence translated in the smaller prefactor. The Pardiso method, as we mentioned, proved to be very effective when solving triangular systems, and was also the fastest when solving smaller sized systems due to the very small prefactor, but quickly fell apart with its $\mathcal{O}(N^{2.35})$ scaling. The natively implemented Krylov method provided the best results out of the linear system methods, and so this was the method we proceeded with as the default linear solver.

### 3.2.4 Linear System Accuracy

Up to this point we have discussed a method to solve the demagnetizing scalar potential using a multipole expansion for the boundary values and a linear system solver for the interior cells. Now we must compute

$$h_d = -\nabla V$$  \hspace{1cm} (3.26)
and verify that the resulting field satisfies Maxwell’s equations 3.1. As mentioned in section 2.2.4, a
discrete calculation of 3.26 organizes the three components \((h_x, h_y, h_z)\) along the borders of the main
grid, and these are the values we are interested in to verify the discrete solution of Maxwell’s equations.
So far averaging the field components to the center of the grid is only of interest if we wish to plot the
field to check its topology, as exemplified in figure 3.3 for a uniformly magnetized cube.

In order to verify the accuracy of Maxwell’s equations we started by simulating a unit cube with
\((N_x, N_y, N_z)\) varying between \(\{16, 32, 48, 64\}\), and calculated \(h_d\) considering a random magnetization
in each cell. Maxwell’s equations 3.1 can be represented as the following equalities:

\[
\nabla \cdot h_d = -\nabla \cdot m : \quad \frac{dh_x}{dx} + \frac{dh_y}{dy} + \frac{dh_z}{dz} = -\left( \frac{dm_x}{dx} + \frac{dm_y}{dy} + \frac{dm_z}{dz} \right),
\]

\[
3.27
\]

\[
\nabla \times h_d = 0 : \quad \frac{dh_z}{dy} = \frac{dh_y}{dz}, \quad \frac{dh_x}{dz} = \frac{dh_z}{dx}, \quad \frac{dh_y}{dx} = \frac{dh_x}{dy}.
\]

\[
3.28
\]

In order to verify the accuracy with which these equalities are satisfied we considered the function

\[
f_{\text{prec}}(x_1, x_2) = -\log_{10} \left( \frac{|x_1 - x_2|}{\max\{|x_2|\}} \right),
\]

\[
3.29
\]

This function gives an approximation on the number of digits that are equal between two given values
\(x_1\) and \(x_2\). Note that we may choose to compare the difference between \(x_1\) and \(x_2\) to the local value
of \(x_2\), but when \(x_2\) is very small the ratio may diverge, and so it is arguably better to compare the
difference to a fixed maximum value of \(x_2\) instead. We calculated the twelve derivatives in 3.27 and 3.28
as finite-differences for all the simulation cells. Recalling from section 2.2.4 the divergence derivatives are
stored on the vertices of the grid and the curl derivatives are stored in the surfaces, we use the values at
each position as \(x_1\) and \(x_2\) to calculate 3.29 for the four equalities in 3.27 and 3.28 over all the cells.

Averaging the results over all the simulation cells, the divergence equality 3.27 was satisfied with an
8 digit precision and the three curl equalities in 3.28 were satisfied with a 16 digit precision. While we
obtained a much better precision for the curl equation, this is obviously only dependent on a correct

Figure 3.3: Horizontal cut through the center of a uniformly magnetized cube inserted in a \(16 \times 16 \times 16\)
grid, with the respective demagnetizing field. We use the green color for magnetization grids, or center-
vector grids.
application of the gradient operator to any given scalar potential as long as the fields are properly discretized along the conjugate dual grid. However, having also obtained a good precision for the divergence equation, we can conclude that the linear system method is a reliable procedure to calculate the scalar potential, and thus obtain a physically accurate demagnetizing field.

3.2.5 Shielded Periodic FFTs

As we just discussed, the linear system method is very reliable, but our previous method comparison in 3.2 clearly showed FFTs have the advantage in terms of efficiency, and so they require further study. Before starting to work on the convolution method, we tried to adapt the work of Nogueira et al. [44] in the simulation of charged clusters to our implementation of the periodic spectral method. The idea was to include a fictitious charge density, with a known analytical potential solution, that would cancel out the monopole, dipole and quadrupole of the system. This would effectively nullify most of the interactions between the periodic images during the FFT transformations, shielding them from each other. After obtaining the final potential, we would subtract the analytical potential of the fictitious charges and thus obtain the potential of the non-periodic system. From our tests we had to use point charges as the shielding, and while it was very effective at eliminating the monopole, dipole and quadrupole of the system, it would create its own strong dipoles that would add a lot of noise to the FFT transformations. In the work of Nogueira they were able to use Gaussian charge densities that are continuous and work much better for this type of method, but these do not work well with our geometry and discretization. We decided to abandon this work to focus on the convolution method.

3.3 FFT Convolution Method

In the previous section we discussed the linear system approach to Poisson’s equation, and consequently, the demagnetizing field. As we saw at the end of the section, this method satisfied Maxwell’s relations between $m$ and $h_d$ with high precision in the conjugate-dual grid construction, as discussed in chapter 2. In this section we will study the convolution method which is currently the preferred procedure to calculate the demagnetizing field in most simulations.

As we mentioned in the beginning of this chapter, the analytical solution 3.3 to Poisson’s equation 3.2 is an integral with convolution properties,

$$V(r) = \frac{1}{4\pi} \int_{\Omega} \frac{1}{\|r - r'\|} \rho_m(r') \, d^3r'.$$

$$= (G \ast \rho_m) (r), \quad (3.31)$$

where $G(r - r') = \frac{1}{4\pi\|r - r'\|^2}$ is the Green’s function for the Laplace operator, acting as a kernel that only depends on geometrical properties. The convolution theorem states that, under certain conditions, the Fourier transform $F$ of a convolution is the pointwise product of the Fourier transforms,

$$F\{f \ast g\} = F\{f\} \cdot F\{g\}. \quad (3.32)$$
Considering the $O(N \log_2 N)$ scaling of FFTs, this theorem is very useful to develop efficient Poisson solvers. Solving the equation through this convolution doesn’t require the open boundary conditions to be imposed, as they are already satisfied by integral 3.30. In discretized space this integral is replaced by a sum which represents a discrete convolution. The problem is that using a discrete version of $G(r - r')$ by simply discretizing the position vectors $\|r - r'\|$ directly is basically saying every simulation cell is a monopole with the charge at its center. This is not a good approximation for neighbouring cells, not to mention that it produces a singularity when we try to calculate the self-interaction of each cell.

### 3.3.1 Analytical Kernels

In order to exploit the convolution theorem to efficiently solve the Poisson problem a discrete kernel that correctly represents the continuous convolution must be computed. Schabes and Aharoni [36] derived an analytical expression for the magnetostatic interaction field for a three-dimensional array of ferromagnetic cubes. If we consider the simulation cells as uniformly magnetized cuboids, which makes physical sense under the assumption that the magnetization distribution is uniform on a range of the order of the exchange length $l_{ex}$, this result could be used to approximate a kernel. Since then, many other works surfaced exploring this idea to compute kernels not only for the magnetic scalar potential [34, 45, 35], but also for the demagnetizing field directly [46, 47].

The kernel that describes equation 3.30 is a scalar, but this is usually not the approach taken. Going back to our discussion on surface charges at the beginning of the chapter, we recall they are included in the model through the extra surface term in integral 3.4, and this allows for some convenient mathematical manipulations. Knowing that $\rho_m = -\nabla \cdot m$ and $h_d = -\nabla V$, two equivalent forms of 3.4 can be written:

$$V(r) = \frac{1}{4\pi} \int_{\Omega} m(r') \cdot \nabla r' \frac{1}{\|r - r'\|} d^3 r' = (S \ast m)(r),$$  \hspace{1cm} (3.33)

$$h_d(r) = -\frac{1}{4\pi} \nabla r \int_{\Omega} m(r') \cdot \nabla r' \frac{1}{\|r - r'\|} d^3 r' = (\tilde{N} \ast m)(r).$$  \hspace{1cm} (3.34)

Applying the divergence theorem to the surface contribution in 3.4 joins the two contributions as volume integrals, after which the identity $\nabla \cdot (f v) = f \nabla \cdot v + v \cdot \nabla f$ can be used to obtain 3.33. Integral 3.34 is simply the gradient of 3.33. Naturally, $S$ is a vector describing the transformation between $m$ and $V$, and $\tilde{N}$ is a tensor describing the transformation between $m$ and $h_d$. For finite-difference implementations both were discretized assuming uniformly magnetized cuboids with the respective integrals then solved analytically from that approximation: $\tilde{N}$ in [46] and $S$ more recently in [35]. Both of these kernels employ the surface charge assumption.

The demagnetizing tensor convolution 3.34 is the most commonly used method for the solution of the demagnetizing field, implemented in well known simulations [12, 13, 16]. The vector convolution method 3.33 was later implemented in [16]. Considering our approach to the discretization of Maxwell’s equations, the latter is of more interest to our study because it also solves the scalar potential at the vertices of the simulation cells, allowing us to obtain the demagnetizing field in the conjugate-dual grid formulation. The tensor method, on the other hand, transforms $m$ into $h_d$ directly from the center to the center of the cells, and so we cannot verify if the results satisfy the numerical discretization of Maxwell’s
equations in a consistent manner, as we discussed in section 2.2.4.

**Charge Kernel**

As we just mentioned, most simulations consider kernels that act on the magnetization distribution directly with the surface charge assumption. A kernel acting on the charge density starting from integral 3.30 would require the FFT transformation to be performed on the scalar $\rho_m$ instead of vector $m$. Besides the physical differences from the surface charge assumption, this would also reduce the number of FFTs at the expense of only one extra divergence operation. While Berkov et al. [34] use a kernel that acts on the charge density, the publication is not explicit on how to calculate it. McMichael et al. [48] refer to this kernel as being calculated numerically, and only treat the 2D case. To our knowledge a 3D analytical derivation of a charge kernel has not been used in micromagnetics. Looking at integral 3.30 we thus considered an analogous (but not equivalent) assumption to the other analytical kernel calculations and assume the cells are uniformly charged cuboids, which was already discussed in [34] and [48]. The physical implication is that we are assuming the magnetization distribution to diverge at a constant rate on a range smaller than the cell size. Given the required discretization condition that $\Delta x, y, z < l_{ex}$ this assumption can also be justified by the exchange interaction. Analogous to the analytical derivation in [35], for a scalar potential source at position $r_j$ with volume $V_j = \Delta x \Delta y \Delta z$ we have

$$G_a(r_i - r_j) = \frac{1}{4\pi} \left[ \int_{V_j} \frac{1}{\|r - r'\|} d^3r' \right]_{r = r_i}$$

$$= \frac{1}{4\pi} \int_{-\frac{\Delta x}{2}}^{\frac{\Delta x}{2}} \int_{-\frac{\Delta y}{2}}^{\frac{\Delta y}{2}} \int_{-\frac{\Delta z}{2}}^{\frac{\Delta z}{2}} \frac{dz' dy' dx'}{\sqrt{(x - x')^2 + (y - y')^2 + (z - z')^2}}.$$ (3.36)

In the context of molecular dynamics, however, we found integral 3.36 was solved analytically by Hummer [49] for the electrostatic potential of a unit cube. The result can be generalized for a cuboid with different dimensions, and is thus applicable to our problem. We will not write the full expression here since it is rather large, and it is available in the Wolfram Alpha website by inputting “charged cuboid potential” [50]. We adopted the notation $G_a$ for this analytical approximation since we will introduce another version of $G$ later on.

**Comparing the Kernels**

So far we have introduced two analytical kernels that are of interest to us: $G_a$, which transforms $\rho_m$ into $V$ working only in the vertices of the simulation cells, and $S$, which transforms $m$ into $V$ from the center into the vertices of the simulation cells. The vector kernel $S$ was implemented according to its original publication [35]. The interaction distances between each cell for which $S$ is calculated have to be considered as going from the center of one cell to the various vertices of the grid. The scalar kernel $G_a$ was implemented in a similar way but since it only works with the scalar quantities $\rho_m$ and $V$ the interaction distances were calculated between the vertices of the normal grid, or equivalently, between the centers of the conjugate grid cells. Note that in order to perform a convolution for a conjugate grid with $N_x' \times N_y' \times N_z'$ cells, the scalar kernel must be computed for $(2N_x' - 1)(2N_y' - 1)(2N_z' - 1)$ locations,
and a similar consideration must be made for $S$.

When using the convolution theorem to perform such a calculation adequate zero padding must also be included. Zero padding is simply adding zeros to an array in order to increase its length in a certain way. In this case, having the scalar kernel calculated for $(2N'_x - 1)(2N'_y - 1)(2N'_z - 1)$ locations, the array of $\rho_m$ values had to be padded in order to have the same dimensions. Given the periodic nature of FFTs, this technique avoids issues with circular convolutions.

In order to perform an initial test on the kernels we calculated the magnetic scalar potential of a ferromagnetic slab with the same characteristics as described in the Standard Problem #4 [51]. This is a standard problem used to benchmark dynamical simulations that we will introduce in more detail later and will be often referred during the rest of this thesis. Essentially the problem deals with the magnetization inversion of a thin film similar to what would be found on a bit used in magnetic storage. For now let us simply consider the initial state as shown in figure 3.4. This slab was discretized in a 3D grid with dimensions $N_x \times N_y \times N_z = 128 \times 32 \times 1$, giving us a conjugate grid with dimensions $N'_x \times N'_y \times N'_z = 129 \times 33 \times 2$, centered at the vertices, where the scalar potential was calculated.

While calculating the scalar potential with both kernels resulted in a very similar result, differences were visibly noticeable near the borders of the slab where the divergence of the magnetization was greater. Furthermore, not only were the results different between the two kernels, they were also both different from the solution obtained with the linear system method we have studied previously. To illustrate this, we plotted on figure 3.5 the magnetic scalar potential $V$ along $y$ at the left border of the slab, as marked in red in figure 3.4, for the upper vertices of the simulation cells ($z = 3\,\text{nm}$).

The most important aspect however is to evaluate how both kernels performed in maintaining the discrete relations of Maxwell’s equations. Using the same procedure as we described in section 3.2.4, on average over all the simulation cells, we only obtained a one digit precision for the divergence equation, with the error between the divergences going as high as 10% for cells at the borders of the material. We thus verified the demagnetizing field obtained from both potential solutions convolved with $G_a$ and $S$ could not satisfy Maxwell’s divergence equation with good precision, as we have obtained with the linear system method. While the introduction of the surface charge contribution in the integral 3.4 would be a natural reason to justify this loss in precision, the fact that the potential solution obtained from $G_a$ was also unreliable suggests the constant magnetization and constant charge approximations used in

\begin{align*}
\bullet \quad \text{Thickness: } t &= 3\,\text{nm} \\
\bullet \quad A_{ex} &= 1.3 \times 10^{-11}\,\text{J/m} \\
\bullet \quad M_s &= 8.0 \times 10^5\,\text{A/m} \\
\bullet \quad K &= 0.0
\end{align*}

Figure 3.4: Ferromagnetic slab similar to Permalloy magnetized in an equilibrium $s$-state.
the analytical kernel integrations may also be at fault. Once again the curl equation was satisfied with roughly 16 digit precision.

For the scalar kernel $G_a$ we obtained $\nabla \cdot h_d < -\nabla \cdot m$ and for the vector kernel $S$ we obtained $\nabla \cdot h_d > -\nabla \cdot m$, suggesting there is a certain kernel that gives a solution to $V$ between the solutions so far computed that replicates the linear system solution and does in fact satisfy the divergence equation.

A more detailed comparison between the kernels will be provided, however let us now proceed with the introduction of a third candidate.

### 3.3.2 A Numerical Approach to the Scalar Kernel

So far we have mentioned three analytical kernels integrated from a uniformly magnetized/charged cuboid approximation. We tested two of them: the scalar kernel $G_a$ which we introduced based on [49] and the vector kernel $S$ from [35], but were unable to reach the same reliability of results we obtained with the linear system method from section 3.2. There should be a convolution kernel that gives the same scalar potential solution as the linear system method, however it was not obvious what analytical approach should be employed to compute it. With this in mind we set out to try and extract it numerically from a linear system calculation.

One way to do this would be to, after having the geometry set, solve the scalar potential for any magnetized state through the linear system method and then deconvolve it with the convolution theorem,

$$G_n = F^{-1} \left\{ \frac{F\{V\}}{F\{\rho_m\}} \right\}. \quad (3.37)$$

Independently of the charge distribution, from our tests, this procedure always resulted in the same kernel. This makes sense given that $G_n$ only depends on the geometry of the problem. However, this
does a lot of unnecessary computations because of all the charge related calculations, like the multipole expansion from section 3.2.1, that are not exactly needed to get the information we want.

In order to expedite the computation of $G_n$ we used the following procedure, illustrated in figure 3.6:

- After the geometry is set for the material that is going to be simulated, a unit charge is placed in the first cell of its conjugate grid.

- A discretized box is set around the material. The necessary boundaries are then extended so that they are at least at a distance $\delta$ from the charged cell, where $\delta$ will be referred to in multiples of the largest cell dimension: $\delta = 32 \Leftrightarrow \delta = 32 \times \max\{\Delta_x, \Delta_y, \Delta_z\}$.

- Having only one charged cell, and having this cell at a distance from the boundary that is several times larger than its size (depending on the $\delta$ used), we calculate the boundary conditions through a monopole approximation. The interpolation procedure from the multipole expansion is maintained.

- The potential $V$ is solved for all the interior cells of the box through the linear system method.

- Since we used a single unit charge, the values of $V$ inside the conjugate grid of the material region (painted in yellow) can be used directly for the kernel, without needing to be deconvolved. This corresponds to $N'$ kernel values.

- For a material discretized in a total of $N' = N'_x \times N'_y \times N'_z$ cells (conjugate grid), the full $G_n$ kernel with dimensions $(2N'_x - 1, 2N'_y - 1, 2N'_z - 1)$ is obtained by filling the values of the remaining cells according to the symmetries illustrated on the right of figure 3.6.

As with the other kernels, this is a calculation that only needs to be carried out once for each geometry. The reciprocal kernel $\mathcal{F}\{G_n\}$ can then be stored in memory and every single scalar potential calculated afterwards for that geometry only requires a discrete 3D FFT of the charge density, a point wise product in reciprocal space, and a discrete inverse 3D FFT of the reciprocal space potential, yielding the real space potential:

$$V = \mathcal{F}^{-1}\{\mathcal{F}\{G_n\} \cdot \mathcal{F}\{\rho_m\}\} \quad (3.38)$$
In figure 3.7 we updated the plot of figure 3.5 including the potential obtained from the convolution with the numerical approximation of $G_n$ we just described. The potential solution convolved with $G_n$ matched the direct linear system solution. Varying the value used for the $\delta$ parameter, the Maxwell results after computing the gradient were much better. For $\delta = 8$ we could already satisfy the divergence equation with an average of three digits precision. For higher $\delta$ the precision increased. This makes sense since higher values of $\delta$ make the monopole calculation of the boundary conditions a better approximation. Of course, a higher $\delta$ also means the discretized box around the material is larger and so the linear system takes longer to solve, increasing the time needed to calculate the kernel. However, it is a single calculation. For a dynamical simulation that requires many time steps the difference in time invested on the kernel from using $\delta = 8$ or, for example, $\delta = 64$ is most likely negligible, and so it makes sense to guarantee a more accurate kernel. The accuracy of the kernel is always limited by the accuracy of the method used to solve the linear system. In our tests we found that scalar potentials convolved with a $\delta = 64$ computation of $G_n$ already produced results with comparable precision in the divergence equation to what was obtained with the direct linear system approach.

### 3.3.3 Kernel Accuracy and Efficiency

So far we have talked about various methods to solve the demagnetizing field problem in magnetostatics. In order to compare them with more detail we started by simulating a cubic magnetic particle with side $L = 50\,\text{nm}$ discretized in a cubic grid $n \times n \times n$ for $n = \{16, 32, 64\}$. We considered two cases of magnetization, as shown in figure 3.8:

- **Random Magnetization:** for each grid size we added a random magnetization to each cell, which was maintained constant for all the methods.
Figure 3.8: Vertical cut through the center magnetized cube discretized in a $16 \times 16 \times 16$ grid with both a random magnetization distribution and a flower state distribution.

- **Flower State:** this is a common state of magnetization in a cubic particle and is obtained by relaxing\(^2\) the magnetization state of a uniformly magnetized cubic particle to a minimum of free energy.

First, in table 3.2, we measured the efficiency aspects of the kernel calculations, including the matrix $A$ from the linear system method as presented in section 3.2 and the three kernels $G_a$, $S$ and $G_n$ for the convolution method. We measured both the time it takes to compute each of the kernels at the beginning of the simulation and the time of the post-kernel computations that are required to reach the solution of the demagnetizing field, that is, the computations that must be repeated during the course of a simulation. For the numerical kernel $G_n$ we included two cases where we used $\delta = 32$ and $\delta = 64$. The results presented in table 3.2 are independent of the magnetization distribution. Second, in table 3.3 we present the precision results of each solution calculated as described in section 3.2.4 for both the random and the flower state distribution. All results presented are an average of three repeated calculations.

The speed at which the kernel is computed is not as important as the post-kernel computations or the precision results from table 3.3, however it is still something to consider. The most noticeable difference is between the linear system method and the convolution methods. While the linear system method has a kernel that is very fast to calculate, it pays the price in the timings of the post-kernel calculations with the required multipole expansion of the boundary values and the large linear system that must be solved at each iteration. The convolution methods have an opposite distribution of the time requirements, as the post-kernel operations consist mostly of FFTs, and so these methods are much more efficient at studying time evolutions. For the convolution methods, we notice the computation of the numerical kernel $G_n$ has the advantage when the magnetic domain is larger in every direction than the $\delta$ used. Naturally, when the system is smaller or equal to $\delta$ the computation time of $G_n$ is constant, as should be expected from the scheme we proposed and exemplified in figure 3.6. The difference in time needed to calculate $G_a$ and $S$ was also expected since $G_a$ is a scalar for each cell and $S$ is a three component vector. However, given $G_a$ and $S$ are both an independent calculation for each cell from an analytical expression these timings should have room for improvement with a more careful and parallelized implementation.

\(^2\)A energy minimization algorithm will be discussed in chapter 5.
Regarding the efficiency of the post-kernel calculations in table 3.2, these are the most important timings to compare for the purpose of dynamical simulations. Even though we did not include the demagnetizing tensor $\tilde{N}$ in our study, the most widely used kernel, an efficiency comparison between $S$ and $\tilde{N}$ was done in [35]. The conclusion was that performing a convolution with $S$ was about 1.5 times faster, mostly because $S$ reduced the number of inverse FFTs from three to one, at the cost of only one extra gradient operation. The reciprocal space calculation was also reduced from a tensor-vector product to a vector-vector product. With the scalar kernels $G_a$ or $G_n$, compared to both $\tilde{N}$ and $S$, we also reduced the number of direct FFTs from three to one, maintained only one inverse FFT, and further reduced the reciprocal space calculation to a scalar product. This came at the cost of one extra divergence operation, but like the gradient, it scales with $O(N)$. The results we obtained show that the convolutions with the scalar kernels $G_a$ and $G_n$ run about 1.5 to 2 times faster than those with the vector kernel $S$.

As for reliability aspects, let us now look at table 3.3. We recall these values represent the number of precision digits according to equation 3.29 applied to the divergence equation $\nabla \cdot \mathbf{h}_d = -\nabla \cdot \mathbf{m}$. We included both the average precision value over all the cells as well as the value for the cell with minimum precision. The curl equation was once again satisfied with close to 16 digit precision, as was expected since in all cases the field obtained was a gradient of a scalar potential, and so these results were once again omitted. First of all we notice the linear system method had similar precision results to what we first obtained in section 3.2.4 for both the random and the flower state distributions, once again showing...
the reliability of this method. As for the analytical kernels $S$ and $G_a$, we recall these were calculated from uniform magnetization and charge density approximations, respectively, which are physically based around the existence of the exchange interaction. As expected, these approximation do not perform well for the case of random magnetization since such a distribution does not fit the micromagnetic model. When we increase the grid size for a physical distribution such as the flower state, on the other hand, we see the average precision is much better and increases with the resolution. This is to be expected since we are maintaining the exchange length constant while reducing the size of the simulation cells, and thus making the uniform magnetization and charge density within each cell a better approximation. It should be noted, however, that this increase in average precision does not occur equally in all the cells of the simulation, as can be seen by the low minimum precision values for the analytical kernels. We found cells at the boundary of the material where the divergence of the magnetization is greater to maintain low precision values, once again suggesting either the surface charge effects or the constant charge and magnetization approximations become unreliable in these outer regions.

Finally, with the scheme we developed to approximate the scalar kernel $G_n$ numerically we could recreate the precision results obtained with the linear system method, with high precision values in both the average over all the cells as well as the minimum precision cell, guaranteeing the demagnetizing field solution is reliable over the whole grid of the simulation. Furthermore, the computation of this kernel is not based on any physical approximation, producing similar reliability of results independently of the physical significance of the magnetization distribution. With the simple fine-tuning mechanism we included in the form of the parameter $\delta$ we see a clear precision increase from $\delta = 32$ to $\delta = 64$.

The results obtained so far clearly show the advantages of the numerical kernel $G_n$ in the conjugate-dual grid discretization. Before we present our conclusions, however, let us proceed with another test.

**Standard Problem #4**

So far we have used the precision of Maxwell’s equations as an important factor in comparing the kernels, but when we deal with magnetization dynamics we must average the field components to the center of the cells due to the discretization requirements of the LLG equation, as was discussed in chapter 2. While there is certainly some value in knowing we are averaging from a discretization where our solution to Maxwell’s equations is reliable, we must also compare results after this average has been computed. As such, we solved the standard problem #4 [51] using the numerical kernel $G_n$. As we have mentioned, the initial state is as described in figure 3.4. The dynamical part of the problem consists in studying the magnetization inversion of the thin film. This is done by applying an external field to the slab and recording the time evolution of the magnetization as the system moves towards the new equilibrium. The problem includes two possibilities for the applied field:

\[
\begin{align*}
\text{Field 1:} & \quad \mu_0 H_{ext} = -24.6 e_x + 4.3 e_y + 0.0 e_z \text{ mT} \quad (3.39) \\
\text{Field 2:} & \quad \mu_0 H_{ext} = -35.5 e_x - 6.3 e_y + 0.0 e_z \text{ mT} \quad (3.40)
\end{align*}
\]

The results for the external field 3.39 are presented in figure 3.9 and 3.10. We used a discretization...
Figure 3.9: Time evolution of the average magnetization components according to the Standard Problem #4 [51], for the applied field 3.39. Results obtained with the scalar kernel $G_n$ were compared with the ones computed with the simulation codes $\mu$Max [13, 14] and MicroMagnum [16], as well as the previous data submitted by Rocha [11] to the $\mu$Mag site [52].

Figure 3.10: Magnetization distribution over the slab of Standard Problem #4 [51] when $\langle m_z \rangle$ crosses 0. The $z$ component is color coded. Each vector point is an average of the magnetization of $3 \times 3 \times 2$ cells.

of $256 \times 64 \times 2$ cells. The time evolution was calculated with the Landau-Lifshitz-Gilbert equation 1.64, numerically integrated with a 4th order Runge Kutta method. We will talk about these methods in more detail in the next chapter, dedicated to the time integration of the LLG equation. Here we simply used a very small time step of $\delta t = 0.0001$ in natural units, or roughly $5 \times 10^{-16}$ s given that our intention is to minimize the time evolution errors and validate the static calculations of the problem.

While some small deviations can be observed, these are not uncommon between the various solutions so far submitted to the $\mu$MAG website [52]. The dynamical behaviour of the problem while using the scalar kernel $G_n$ for the computation of the demagnetizing field follows the same oscillation pattern, and the magnetization distribution when $\langle m_z \rangle$ crosses zero is visibly the same as other submissions.

We have tested the numerical kernel $G_n$ on both the conjugate-dual grid discretization and the usual
center-vector grid. We have shown that using this method to calculate the demagnetizing field we can increase the convolution speed by up to two times when compared with the vector kernel $S$. The results obtained also consistently replicate the precision obtained with the linear system method, which we could not accomplish with the analytical kernels tested. Furthermore, the precision results of this kernel are completely independent of both the cell size used and the physical significance of the magnetization distribution, providing a consistent conservation of the discrete Maxwell equations and making the method much more mathematically robust. Finally, by averaging the field components to the usual center-vector grid discretization, we are able to recover the known results of the standard problem #4, thus completely validating both our demagnetizing field calculation and also the other more simple static terms that contribute to the effective field.
As we saw in the previous chapter, the FFT based convolution methods greatly speed up the static calculations related to the demagnetizing field, and various approximated convolution kernels can be calculated for each geometry with good precision. As such, we can efficiently obtain the effective magnetic field correspondent to each magnetization distribution.

For a non-equilibrium magnetization distribution the effective magnetic field created, with or without contributions from exterior sources, will produce a torque upon the distribution at every point in space causing it to shift. At each point in time we thus have a different magnetization distribution producing a different effective field. The distribution reaches equilibrium when it shifts into a configuration where the effective field no longer produces a torque upon the distribution. These dynamical aspects were introduced in chapter 1 when we talked about the Landau-Lifshitz-Gilbert (LLG) equation and its variations.

Solving this equation analytically is only possible for very simple problems, and its numerical treatment is a complicated task. On one hand, it is extremely non-linear due to the dependence of the effective field on the magnetization distribution, and on the other hand it includes two types of motion that act on different time scales: a very fast precession and a very slow damping. General purpose solvers such as Runge-Kutta methods have been widely used in micromagnetic simulations to calculate the time evolution of the magnetization due to their simplicity, but these do not conserve the physical properties of the LLG equation and so a very small time step must be used to correctly mimic the complex LLG motion. Extensive work has been invested into developing integration methods specific to the LLG equation to try and overcome this problem [53, 54, 55, 56, 57].

We will start this chapter with a brief overview of the physical properties of the LLG equation before introducing each of the methods we implemented for its numerical solution. We considered methods from the Runge-Kutta family as well as LLG specific: a norm conservative Gauss-Seidel method [55] and a fully conservative geometrical mid-point method [56].

4.1 Landau-Lifshitz-Gilbert Equation

Let us start by rewriting the LLG equation as was introduced in section 1.3. We present both the original LLG formulation in 4.1 and the Gilbert form of the Landau-Lifshitz equation in 4.2, and we recall the
parameter relations presented in 4.3.

\[
\frac{\partial M}{\partial t} = -\gamma M \times H_{\text{eff}} + \frac{\alpha}{M_s} M \times \frac{\partial M}{\partial t}, \tag{4.1}
\]

\[
\frac{\partial M}{\partial t} = -\gamma_L M \times H_{\text{eff}} - \frac{\lambda}{M_s} M \times (M \times H_{\text{eff}}). \tag{4.2}
\]

\[
\gamma_L = \frac{\gamma}{1 + \alpha^2}, \quad \lambda = \frac{\gamma \alpha}{1 + \alpha^2} \tag{4.3}
\]

While we introduced them as being mathematically equivalent, this is not entirely true. If we consider the physical meaning of \( \gamma \) as being the electron gyromagnetic ratio, then the equations are only physically equivalent in the limit of vanishing damping. Moreover, if we look at the limit of infinite damping, that is, \( \lambda \to \infty \) and \( \alpha \to \infty \), we note 4.1 tends to zero and 4.2 diverges [58]. Assuming that a very large damping should limit the motion of the magnetization distribution, equation 4.1 has better physical agreement. However, most numerical implementations work with equation 4.2 in order to avoid the implicit nature of 4.1, and \( \alpha \) and \( \lambda \) are usually small.

For the purpose of their numerical implementation, let us rewrite 4.1 and 4.2 in our set of natural units as described in section 2.3,

\[
\frac{\partial m}{\partial t} = -m \times h_{\text{eff}} + \alpha m \times \frac{\partial m}{\partial t}, \tag{4.4}
\]

\[
\frac{\partial m}{\partial t} = -\frac{1}{1 + \alpha^2} m \times h_{\text{eff}} - \frac{\alpha}{1 + \alpha^2} m \times (m \times h_{\text{eff}}). \tag{4.5}
\]

Note that, considering a common ferromagnetic material like permalloy (see figure 3.4), we are dealing with a natural unit of time \((\gamma M_s)^{-1}\) that corresponds to roughly five picoseconds. That is the length of time we expect to pass before we start to notice significant changes in the magnetization distribution.

### 4.1.1 Structural Properties

As we have mentioned, the LLG equation has important structural properties that are sometimes overlooked when computing its numerical time integration. A detailed description of these properties can be found in [56], which we will proceed to summarize:

- For each location in time and space, the magnetization vector field is considered to have constant magnitude, \( \|m(r, t)\| = 1 \).

- For an applied field constant in time, the LLG equation imposes a Lyapunov structure on the time evolution of the magnetization [59]. What this means is that, along LLG trajectories, the free energy of the system should decrease. This property guarantees the system tends towards stable equilibrium points.

- As we introduced in section 1.3, neglecting the damping term in both forms of the LLG equation leaves us with a conservative precessional equation, a characteristic usually referred to as an Hamiltonian structure. In the limit of vanishing damping we should then expect the numerical integration of the LLG equation to preserve the free energy of the system.
4.2 Numerical Methods for the LLG Equation

In this section we will overview our numerical implementation of the LLG equation and specify the details related to each of the methods tested. We will start with the simple explicit methods of the Runge-Kutta family and then proceed to more complex semi-implicit methods specific to the LLG equation. We will be working with the magnetization $m$ and the effective field $h_{\text{eff}}$ it produces, and whenever we mention these quantities, we are referring to their spatially discretized counterparts, as discussed in section 2.2.1.

4.2.1 Runge-Kutta Methods (RK)

Let us now consider equation 4.5 and write it in the following manner:

$$\frac{\partial m}{\partial t} = f(t, m). \quad (4.6)$$

For some initial time $t(0)$ we have a magnetization distribution $m(0)$ and from this point forward it evolves in time according to 4.6, which actually represents a set of $N$ partial differential equations, one for each simulation cell, coupled by the effective field $h_{\text{eff}}$.

We evolve the system in discrete time steps of size $\delta t$, and so we call $m(1)$ the distribution after one time step, $m(2)$ after two, and so on. A very simple iteration procedure would be the Euler method:

$$
m^{n+1} = m^n + \delta t f(t^n, m^n) + O(\delta t^2),
\quad t^{n+1} = t^n + \delta t. \quad (4.7)$$

In order to perform the LLG evolution starting from an initial distribution $m(0)$ the first step is thus to calculate the corresponding $h_{\text{eff}}$. Then, $f(t(0), m(0))$ is calculated and all the simulation cells are advanced one step in time according to 4.7. After each time step the effective field must be updated before performing the next iteration. The total number of iterations may be set from the beginning or we may monitor the equilibrium condition 1.54 to decide when to stop the evolution. This is, in fact, the basis of any LLG evolution, however different iteration methods can be considered. While the Euler iteration is a very simple first-order Runge-Kutta method, the fourth-order RK4 is more widely used,

$$
m^{n+1} = m^n + \frac{\delta t}{6} \left( m^{k1} + 2m^{k2} + 2m^{k3} + m^{k4} \right) + O(\delta t^5),
\quad t^{n+1} = t^n + \delta t. \quad (4.8)$$

where there is a contribution from four different increments at different positions of the time interval $\delta t$,

$$
m^{k1} = f(t^n, m^n),
\quad m^{k2} = f \left( t^n + \frac{\delta t}{2}, m^n + \frac{\delta t}{2} m^{k1} \right),
\quad m^{k3} = f \left( t^n + \frac{\delta t}{2}, m^n + \frac{\delta t}{2} m^{k2} \right),
\quad m^{k4} = f \left( t^n + \delta t, m^n + \delta t m^{k3} \right). \quad (4.9)$$
Many explicit Runge-Kutta type methods have been developed by varying the amount of increments and their weights. A generalization of the method for a total of \( s \) increments can be written as

\[
m^{n+1} = m^n + \delta_t \sum_{i=1}^{s} b_i m^{ki}
\]

\[
m^{k1} = f(t^n, m^n)
\]

\[
m^{k2} = f(t^n + c_2 \delta_t, m^n + \delta_t (a_{21} m^{k1}))
\]

\[
m^{k3} = f(t^n + c_3 \delta_t, m^n + \delta_t (a_{31} m^{k1} + a_{32} m^{k2}))
\]

\[
\vdots
\]

\[
m^{ks} = f(t^n + c_s \delta_t, m^n + \delta_t (a_{s1} m^{k1} + \cdots + a_{s,s-1} m^{k_{s-1}}))
\]

(4.10)

Here the increments are weighed by the parameters \( b_i \). Each increment is calculated at a certain position of the interval given by the \( c_i \) parameters, and has a contribution from the previous increments weighed by the \( a_{ij} \) parameters. A more convenient way to arrange this information is in the so called Butcher tableau\(^1\). Below we present the tableau of the general Runge-Kutta method 4.10, and also present an example for the RK4 method 4.8-4.9.

\[
\begin{array}{cccc|cccc}
0 & c_2 & a_{21} & 0 \\
& c_3 & a_{31} & a_{32} & 1/2 & 1/2 \\
& \vdots & \vdots & \vdots & 1/2 & 0 & 1/2 \\
& c_s & a_{s1} & a_{s2} & \cdots & a_{s,s-1} & 1 & 0 & 0 & 1 \\
b_1 & b_2 & \cdots & b_{s-1} & b_s & 1/6 & 1/3 & 1/3 & 1/6
\end{array}
\]

(4.11)

**Dormand-Prince Method**

While most micromagnetic simulations like *MuMax* [13] or *MicroMagnum* [16] use explicit Runge-Kutta methods, these are usually from an embedded type, which employ an adaptive time step that adjusts itself after each iteration based on a certain error estimation. See [37] for a more detailed description on Runge-Kutta methods. Below we present the tableau for the Dormand-Prince method,

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

(4.12)

\(^1\)After John C. Butcher, a New Zealand mathematician.
The first thing to note is that the tableau in 4.12 has two bottom rows. This method includes increments that are tailored in a specific way so that they produce both a fourth and a fifth order solution. The weights in the first row correspond to the fourth order solution and the weights in the second row correspond to the fifth order solution. We thus define the absolute error of the iteration as

\[ \epsilon_{\text{abs}} = \max \left\{ \| \mathbf{m}_{n+1}^{(u)} - \mathbf{m}_{n+1}^{(5u)} \| \right\}. \] (4.13)

The difference between the solutions is a vectorial quantity for each cell, and so we pick the maximum norm between all the cells in order to filter out the "worst offender" and use that as the scalar error parameter. Now, since we are measuring the error between a fourth and a fifth order solution, we expect the error to scale with \( O(\delta t^5) \), and we can use this information to devise a time-step estimation procedure.

For a given time step \( \delta t^{(1)} \) that produces an error \( \epsilon_{\text{abs}}^{(1)} \), the time step \( \delta t^{(0)} \) that would have given a desired error value \( \epsilon_{\text{abs}}^{(0)} \) can be estimated as

\[ \delta t^{(0)} = S \delta t^{(1)} \left( \frac{\epsilon_{\text{abs}}^{(1)}}{\epsilon_{\text{abs}}^{(0)}} \right)^{0.2}, \] (4.14)

where \( S \leq 1 \) is a safety headroom factor since the estimation is not exact. The parameter \( \epsilon_{\text{abs}}^{(0)} \) is essentially the desired accuracy, and should be some small value contextualized to the problem. In our case the error represents a certain difference in rotation of two unit vectors and so we may consider for example \( \epsilon_{\text{abs}}^{(0)} \sim 10^{-5} \). The relation 4.14 is used in two ways. If a time step is taken and we observe that \( \epsilon_{\text{abs}}^{(1)} > \epsilon_{\text{abs}}^{(0)} \) then we use 4.14 to reduce \( \delta t \) and repeat the iteration. If, however, we observe \( \epsilon_{\text{abs}}^{(1)} < \epsilon_{\text{abs}}^{(0)} \) then the iteration was successful and we may use the estimation of 4.14 to increase \( \delta t \) for the next iteration. Note that these error estimations are related to the fourth order solution, and so that is the one that should be accepted to advance the magnetization in time in order to avoid extrapolations.

**Runge-Kutta and LLG Conservations**

The Runge-Kutta type methods we have introduced so far are intrinsically non-conservative. They are fast to implement, and even though each iteration is very fast to calculate it is recursively done so on top of a previous explicit approximation, leading the accumulated error to increase exponentially. A small enough prefactor allows Runge-Kutta methods to mimic a certain motion for a given period of time, and this is usually achieved by reducing the time step (sometimes more than two orders of magnitude smaller than the natural unit of time), but even then structural properties of equations are not conserved.

Regarding the properties described in section 4.1.1, the most obvious problem is that we are using explicit linear increments to approximate the rotation of vectors, and so the condition \( \| \mathbf{m}(r, t) \| = 1 \) is lost after each iteration. The usual approach in micromagnetic simulations that employ these methods is to renormalize the magnetization vectors in every cell after each iteration to impose the norm condition, which is obviously an artificially introduced step in the method that may introduce errors. The Lyapunov structure is, of course, maintained for a small enough \( \delta t \) as is required for any successful LLG evolution, however the total energy of the system in the limit of vanishing damping is not conserved, and so the Hamiltonian structure is also lost.

53
This integration method for the LLG equation was first published by Wang et al. [54], based on his own previous work [53] and later improved by García-Cervera and Ee [55]. The main objective was to find a middle ground between explicit time integrators that require a very small time step and direct implicit integrators that require solving very complicated and coupled systems of equations.

One other reason for the time step constraint in explicit integrators which we have not mentioned yet is the exchange field. It is the strongest short range interaction, and its physics are greatly influenced by the choice of spatial discretization. In fact, decreasing the cell size by a factor of 10 often requires the time step to also be reduced by a factor of 100 in order to maintain the stability of explicit time integration methods [54]. Aiming to overcome this problem this method was designed specifically around the integration of the exchange field term on the LLG equation,

$$\frac{\partial \mathbf{m}}{\partial t} = -\mathbf{m} \times \nabla^2 \mathbf{m} - \alpha \mathbf{m} \times (\mathbf{m} \times \nabla^2 \mathbf{m}) .$$  \hspace{1cm} (4.15)

The development of the method is documented in [54] and will not be discussed here. Its implementation was done according to [55], which we will specify with the required adaptations.

**Specification of the Method**

Let us consider equation 4.5 and write it in the following form:

$$\frac{\partial \mathbf{m}}{\partial t} = -\mathbf{m} \times (\eta \nabla^2 \mathbf{m} + \mathbf{f}) - \alpha \mathbf{m} \times [\mathbf{m} \times (\eta \nabla^2 \mathbf{m} + \mathbf{f})] ,$$  \hspace{1cm} (4.16)

where we have defined the constant \(\eta = \frac{1}{1+\alpha^2}\) and grouped the non-exchange effective field terms into the field \(\mathbf{f} = \frac{1}{1+\alpha^2}(\mathbf{h}_d + \mathbf{h}_{an} + \mathbf{h}_{ext})\). The iteration from \(\mathbf{m}^n\) into \(\mathbf{m}^{n+1}\) with a time step \(\delta_t\) is composed of three steps with two intermediate magnetizations defined as \(\mathbf{m}^*\) and \(\mathbf{m}^{**}\). The first step transforms \(\mathbf{m}^n\) into \(\mathbf{m}^*\), updating each of the components separately as:

$$\begin{pmatrix} m_x^* \\ m_y^* \\ m_z^* \end{pmatrix} = \begin{pmatrix} m_x^n + (g_x^n m_z^n - g_z^n m_y^n) \\ m_y^n + (g_y^n m_x^n - g_x^n m_z^n) \\ m_z^n + (g_z^n m_y^n - g_y^n m_x^n) \end{pmatrix} ,$$  \hspace{1cm} (4.17)

where we have introduced the field \(g^n\) and its intermediate equivalent \(g^*\). Each component \(g_x^n, y, z\) and \(g_x^n, y, z\) of these two fields is obtained by solving the following equations, respectively, which appear from the application of an implicit Gauss-Seidel method to the gyromagnetic term of the LLG equation [54]:

$$\begin{pmatrix} \nabla^2 - \frac{1}{\eta \delta_t} \end{pmatrix} g_{x,y,z}^n = -\frac{1}{\eta \delta_t} (m_{x,y,z}^n + \delta_t f_{x,y,z}^n) ,$$  \hspace{1cm} (4.18)

$$\begin{pmatrix} \nabla^2 - \frac{1}{\eta \delta_t} \end{pmatrix} g_{x,y,z}^* = -\frac{1}{\eta \delta_t} (m_{x,y,z}^* + \delta_t f_{x,y,z}^*) .$$  \hspace{1cm} (4.19)
Equations 4.18 and 4.19 are of the screened Poisson type, which we will discuss in the next section. Note that while the updated values of $m^*_x$ are used to calculate $m^*_y$ and both are used to calculate $m^*_z$, the effective field terms are only updated after obtaining the three components since our demagnetizing field method couples the three components of $\mathbf{m}$ under $\nabla \cdot \mathbf{m}$, and thus we use $f^*_{x,y,z}$ in 4.19. After calculating 4.17 $\mathbf{m}^*$ is obtained and the first step is complete. The second step treats the damping term through an implicit heat type equation for each component of $\mathbf{m}^{**}$,

$$
\begin{pmatrix}
  m_{x}^{**} \\
  m_{y}^{**} \\
  m_{z}^{**}
\end{pmatrix} = 
\begin{pmatrix}
  m_{x} + \alpha \delta t (\eta \nabla^2 m_{x}^{**} + f_x^*) \\
  m_{y} + \alpha \delta t (\eta \nabla^2 m_{y}^{**} + f_y^*) \\
  m_{z} + \alpha \delta t (\eta \nabla^2 m_{z}^{**} + f_z^*)
\end{pmatrix}.
$$

(4.20)

Once again, with the terms rearranged, 4.20 can be written as a set of screened Poisson equations, which must be solved for $m_{x,y,z}^{**}$ to complete the second step of the method,

$$
\left[ \nabla^2 - \frac{1}{\alpha \rho \delta t} \right] m_{x,y,z}^{**} = -\frac{1}{\alpha \rho \delta t} \left( m_{x,y,z}^* + \alpha \delta t f_{x,y,z}^* \right).
$$

(4.21)

The third and final step is simply to project $\mathbf{m}^{**}$ onto the unit sphere,

$$
\mathbf{m}^{n+1} = \frac{1}{\|\mathbf{m}^{**}\|} \mathbf{m}^{**}.
$$

(4.22)

Note that, while this projection is effectively a renormalization like what is done in explicit schemes to impose $\|\mathbf{m}(\mathbf{r}, t)\| = 1$, here it is an actual “valid” part of the method because a Lagrange multiplier for the norm constraint is included from the beginning [53].

This method was shown to be stable independently of the spatial discretization. A stability dependence on the damping constant $\alpha$ was later found, however this did not impede the method from providing good results even for small damping [55]. While this method allowed for a larger time step in each iteration compared to explicit methods, it is only semi-conservative in regards to the norm constraint, with no special attention given to energy conservations.

Screened Poisson Equation

The first two steps of the Gauss-Seidel Projection Method require 4.18-4.19 and 4.21 to be solved numerically. In [55] a undetailed FFT based Poisson solver is used. Here we developed our own solver by adapting the FFT convolution method from chapter 3. A screened Poisson equation can be written as

$$
\left[ \nabla^2 - \lambda^2 \right] v(\mathbf{r}) = -u(\mathbf{r}),
$$

(4.23)

where $v(\mathbf{r})$ acts as the potential and $u(\mathbf{r})$ as the charge density. Much like the Poisson equation 3.2 it also admits an integral solution with convolution properties similar to 3.3,

$$
v(\mathbf{r}) = \frac{1}{4\pi} \int_{\Omega} \frac{e^{-\lambda \|\mathbf{r} - \mathbf{r}'\|}}{\|\mathbf{r} - \mathbf{r}'\|} u(\mathbf{r}') d^3 r',
$$

(4.24)
with the strength of the screening defined by the parameter \( \lambda \). These screened potentials are often referred to as Yukawa potentials. The process to solve this equation is exactly the same as the Poisson equation, but the kernel for the convolution must also include the geometrical term \( e^{-\lambda |r-r'|} \) that represents the screening. This can be easily incorporated into our numerical kernel scheme described in section 3.3.2 with some considerations we obtain

- First, unlike the Poisson equation from chapter 3, equations 4.18, 4.19 and 4.21 work with quantities discretized on the main grid and so the kernel has different dimensions \( N_{x,y,z} \) instead of \( N'_{x,y,z} \).
- Second, following the scheme of figure 3.6, with a unit charge placed on the first cell of the grid, the boundary values must be calculated from a Yukawa monopole, \( v_{\text{mono}}(r) = \frac{e^{-\lambda |r|}}{4\pi |r|} \).
- Third, when the linear system method from section 3.2 is applied to solve the kernel in all the interior cells, the discretization of equation 4.23 implies the diagonal of matrix \( A \) in 3.11 must be replaced with \(- \left( \frac{2}{\Delta^2} + \frac{2}{\Delta^2} + \frac{2}{\Delta^2} + \lambda^2 \right)\).

Following the procedure of section 3.3.2 with these considerations we were able to develop a numerical kernel for equations 4.18 and 4.19 with \( \lambda_1 = \sqrt{1/\eta\delta} \), and for equation 4.21 with \( \lambda_2 = \sqrt{1/\eta\delta} \), thus solving step one and two of the Gauss-Seidel projection method through FFT convolutions.

### 4.2.3 Mid-Point Geometrical Integration (MP)

This method uses an implicit mid-point rule to integrate the LLG equation and is detailed in [56]. It is demonstrated that the mid-point scheme preserves the fundamental properties of the LLG dynamics, which we introduced in section 4.1.1 based around this same publication. It is also claimed that the mid-point method is unconditionally stable. A fully implicit method such as this, however, comes at the cost of computational power, and here the main difficulty is the need to solve a very large system of globally coupled nonlinear equations.

#### Specification of the Method

Here we will adapt some of the notation of [56] to better explain the steps required to perform an iteration:

- The arrays \( \mathbf{m} \) and \( \mathbf{h}_{\text{eff}} = \mathbf{h}_d + \mathbf{h}_{\text{ex}} + \mathbf{h}_{\text{an}} + \mathbf{h}_{\text{ext}} \), for a system of \( N \) cells, are vectors of length \( 3N \).
- We introduce the \( 3N \times 3N \) sparse matrices \( C_{\text{ex}} \) and \( C_{\text{an}} \) which combine the exchange and anisotropy interactions into operators that act on the magnetization such that \( \mathbf{h}_{\text{ex}} = C_{\text{ex}} \cdot \mathbf{m} \) and \( \mathbf{h}_{\text{an}} = C_{\text{an}} \cdot \mathbf{m} \). These are block diagonal matrices constructed from equations 2.12 and 2.16 respectively. The construction of \( C_{\text{ex}} \) was done similarly to matrix \( A \) in 3.11, with the consideration that the 7-point stencil of the Laplace operator from equation 2.12 is being applied to a vectorial quantity instead of a scalar, and that the boundary conditions must also be included in \( C_{\text{ex}} \). A similar \( C_{d} \) matrix could be considered to represent the demagnetizing interaction, however it would be a full matrix instead of sparse since this is a global interaction, and as we will see this matrix will not be included in the execution of the method.

---

2The numerical stability does not depend on the time-step.

---

56
• Lastly, we introduce the operator \( \Lambda \) to represent the cross product over all the cells. It depends on a list of vectors: for example, \( \Lambda(\mathbf{m}) \) arranges the components of \( \mathbf{m} \) in a \( 3N \times 3N \) matrix such that the operation \( \Lambda(\mathbf{m}) \cdot \mathbf{h}_{\text{eff}} \) effectively represents the cross product \((\mathbf{m} \times \mathbf{h}_{\text{eff}})_{i,j,k} \) cell by cell. The transformation of \( \mathbf{m} \) into \( \Lambda(\mathbf{m}) \) is done through a third order block diagonal tensor \( \mathbf{\Gamma} \) such that \( \Lambda(\mathbf{m}) = \mathbf{\Gamma} \cdot \mathbf{m} \). The tensor \( \mathbf{\Gamma} \) has \( N \) equal blocks of size \( 3 \times 3 \times 3 \). Each block permutates the components of each of the \( \mathbf{m}_{i,j,k} \) into the cross product \( 3 \times 3 \) skew-symmetric matrix form.

Since this method is implicit by nature starting from equation 4.4 is actually more advantageous than equation 4.5 since it simplifies the damping term without introducing any further complications. In this method we represent the magnetization, its time derivative and the effective field at the mid-point of a time step \( n \) in terms of \( \mathbf{m}^n \) and \( \mathbf{m}^{n+1} \). Rewriting equation 4.4 with these considerations:

\[
\frac{\mathbf{m}^{n+1} - \mathbf{m}^n}{\delta_t} = -\Lambda \left( \frac{\mathbf{m}^{n+1} + \mathbf{m}^n}{2} \right) \cdot \left[ \mathbf{h}_{\text{eff}} \left( \frac{\mathbf{m}^{n+1} + \mathbf{m}^n}{2}, t_n + \frac{\delta_t}{2} \right) - \alpha \frac{\mathbf{m}^{n+1} - \mathbf{m}^n}{\delta_t} \right].
\] (4.25)

Equation 4.25, when rearranged in terms of \( \mathbf{m}^{n+1} \), represents a map \( \Phi^n \) such that \( \mathbf{m}^{n+1} = \Phi^n(\delta_t, \mathbf{m}^n) \), and d’Aquino et al. [56] use the properties of this map to show the implicit mid-point method conserves the structural properties of the LLG dynamics. Each time step thus requires the system of \( 3N \) nonlinear equations to be solved for the \( 3N \) unknowns in \( \mathbf{m}^{n+1} \). Another way to write the system is

\[
\Phi^n(\mathbf{m}^{n+1}) = 0,
\] (4.26)

which leads to the introduction of the vector function \( \Phi^n(y) \), and consequently, \( \Gamma^n(y') \),

\[
\Phi^n(y) = \left[ I - \alpha \Lambda \left( \frac{y + \mathbf{m}^n}{2} \right) \right] \cdot (y - \mathbf{m}^n) - \delta_t \Gamma^n \left( \frac{y + \mathbf{m}^n}{2} \right), \quad (4.27)
\]

\[
\Gamma^n(y') = -\Lambda(y') \cdot \mathbf{h}_{\text{eff}} \left( y', t^n + \frac{\delta_t}{2} \right). \quad (4.28)
\]

Solving 4.26 can be done through the Newton-Raphson (NR) algorithm. This is a well-known method to approximate the solution for a set of nonlinear equations, and an introduction can be found in [37]. A direct application of the method to 4.26 would require us to treat the Jacobians of 4.27 and 4.28, which we will refer to as \( \mathbf{J}^n_\Phi(y) \) and \( \mathbf{J}^n_\Gamma(y') \) respectively. We recall, however, that the demagnetizing interaction introduces a global coupling in the system, which would lead \( \mathbf{J}^n_\Phi(y) \) to be a full matrix and make the computational cost of the NR algorithm unpractical. The usual approach in such a situation is to instead use a quasi-Newton method which employs a reasonable approximation to the Jacobian. In this case, removing the demagnetizing interaction from the Jacobian leaves us with the sparse matrix approximations \( \tilde{\mathbf{J}}^n_\Phi(y) \) and \( \tilde{\mathbf{J}}^n_\Gamma(y') \),

\[
\tilde{\mathbf{J}}^n_\Phi(y) = I - \alpha \Lambda(\mathbf{m}^n) - \frac{\delta_t}{2} \tilde{\mathbf{J}}^n_\Phi \left( \frac{y + \mathbf{m}^n}{2} \right), \quad (4.29)
\]

\[
\tilde{\mathbf{J}}^n_\Gamma(y') = \Lambda(y') \cdot (\mathbf{C}_{\text{ex}} + \mathbf{C}_{\text{an}}) + \mathbf{\Gamma} \cdot \left[ -(\mathbf{C}_{\text{ex}} + \mathbf{C}_{\text{an}}) \cdot y' + \mathbf{h}_{\text{ext}} \left( t^n + \frac{\delta_t}{2} \right) \right]. \quad (4.30)
\]
We now have all the information required to perform each time step of the mid-point method, which by itself is an iterative procedure:

- We start from the vector list $\mathbf{m}^n$ and define the initial state $\mathbf{y}^{(0)} = \mathbf{m}^n$ which will be iterated upon according to $\mathbf{y}^{k+1} = \mathbf{y}^k + \Delta \mathbf{y}$.

- Each $\Delta \mathbf{y}$ is the solution of a sparse linear system $A\mathbf{x} = \mathbf{b}$ with $A = \tilde{J}_p^n(\mathbf{y}^k)$ and $\mathbf{b} = -F^n(\mathbf{y}^k)$. Following the discussion in 3.2.3 the Krylov method was used\(^3\).

- The two previous steps are repeated until $\|F^n(\mathbf{y}^{(k)})\| < \epsilon$ for some small tolerance $\epsilon$, at which point we know $\mathbf{y}^k$ satisfies equation 4.26 and thus conclude $\mathbf{m}^{n+1} = \mathbf{y}^k$. For the tests presented in our work we used $\epsilon = 10^{-10}$.

As mentioned, d’Aquino et al. [56] show this iteration procedure conserves the fundamental properties of the LLG equation with good precision. This comes at the cost of having to solve very large linear systems at each time step. While the quasi-Newton procedure has the big advantage of making these linear systems sparse, we still expect the computational power required for this method to surpass the previously introduced methods. The Gauss-Seidel Projection Method, while also semi-implicit, exploited the FFT convolution method for each iteration, and we have already shown in section 3.2.3 that FFTs have an overall advantage in resource scaling with grid size. Of course, when dealing with numerical precision, the absolute efficiency of each method will also depend on how well it converges for a large time step. If a certain method has an iteration procedure that takes twice as long to compute but allows for a time step that is ten times larger than another, it is still the more efficient method.

### 4.3 Method Comparison

Now that we have discussed the implementation details of each method, we can focus on their comparison. Our initial objective for this chapter was to overcome the time step constraints of traditional Runge Kutta methods by approaching the time integration with more problem specific methods. In this regard, both the GS and MP methods provided promising results in their publications [55, 56]. Our objective, however, was halted by a problem that surfaced during the time integration tests.

Going back to the solution of the Standard Problem #4, as was discussed at the end of chapter 3, we recall that we used a very small time step $\delta t = 0.0001(\gamma M_s)^{-1}$ with the RK4 method. We found that by increasing the time step a strange numerical noise would propagate itself in the simulation and lead the magnetization to diverge to an anti-parallel state, or anti-ferromagnetic state, over the whole domain. This anti-parallel state would act as a stable attractor\(^4\) even though this is a clear maximum of the exchange energy contribution. At first this seemed like an obvious shortcoming of the explicit Runge-Kutta integrator, however subsequent tests lead us to conclude otherwise. In fact, using the fully implicit Mid-Point method lead to the same behaviour where the magnetization would even follow the exact same

---

\(^3\)An iterative Krylov method to iterate a quasi-Newton method to perform a time iteration.

\(^4\)State towards which a dynamical system tends do evolve.
Further testing of LLG integrations with each static term isolated showed that this exponential increase of the numerical noise during the simulation only manifested itself in the presence of the exchange interaction. Even a simple integration of the LLG equation without the damping term and using only the exchange contribution to the effective field lead to this anti-parallel state. Even though no reports were found on this problem, by testing other open-source codes we found ours is not the only one to display this behaviour [60], although there are other codes that do not have this problem [13].

Nevertheless, we may still compare some aspects of the time integration methods. For a discretization of $128 \times 32 \times 1$ cells we found that using $\delta t = 0.02 (\gamma M_s)^{-1}$ in the solution of the Standard Problem #4 the magnetization would quickly diverge to the aforementioned anti-parallel state, leading the three components of the average magnetization $\langle m_{x,y,z} \rangle$ to drop to zero. Reducing the time step to $\delta t = 0.002 (\gamma_0 M_s)^{-1}$ allowed us to solve a full nanosecond of the problem, as seen in figure 4.1. We can observe that in this regime where we suppressed the noise propagation the solutions are consistent with each other. Even in the zoomed section in figure 4.1 we can observe the RK4 and MP solutions completely overlap, while the GS solution has a small deviation. This can be explained due to the fact that the RK4 and MP methods use exactly the same discretization of the effective field terms, while the GS method uses a different discretization of the exchange term, as detailed in section 4.2.2.

In figure 4.1 we also present the average computational time that was required to perform a full iteration with each of the methods. As expected, the Runge-Kutta method is the fastest where the computational time for each iteration is mostly due to the demagnetizing field calculation. The Gauss-Seidel iteration has an increased computational cost with the extra FFT convolutions required to solve the
screened Poisson equations 4.18 and 4.19. The Implicit Mid-Point method is the most computationally intensive given the quasi-Newton procedure that must be solved with each iteration. This quasi-Newton procedure requires even more resources if we choose to reduce the tolerance $\epsilon$.

Given that we found the noise propagation to be independent of the LLG integration method used and demagnetizing field calculation, to be only related to the exchange field calculation, and since we have obtained the known results of the standard problem #4 with all the methods tested for a small time step, we can conclude both the Gauss-Seidel method and the Implicit Mid-Point method were correctly implemented according to their publications [55, 56]. Furthermore, we can conclude the noise propagation must be related to the numerics of the base equations, which we have narrowed down to

$$\frac{\partial \mathbf{m}}{\partial t} = \mathbf{m} \times \nabla^2 \mathbf{m},$$

(4.31)

presented in natural units. The numerical implementation of this equation must require some subtle treatment, which we did not test, and may prove to be an interesting exercise in numerical analysis for future work. Unfortunately, until the noise propagation problem is solved, we are unable to increase the time step to more acceptable values\(^5\), as we should be able to do with the GS and MP methods, and properly compare these methods in terms of overall efficiency and reliability.

\(^5\)d’Aquino et al. [56] report a time step of $\delta t = 2.5$ ps $\approx 0.4 \left(\frac{\gamma M_s}{\gamma M_s}\right)^{-1}$ with the Implicit Mid-Point method for the solution of the Standard Problem #4, a considerable increase from explicit time integrators.
Spin Valve Simulation

So far we have talked about various aspects of the basic micromagnetic model, from the theoretical background to its numerical implementation. Our work has mostly been aimed at the computational aspect of micromagnetic simulations. The ultimate goal, however, is for these simulations to be used in parallel with experimental tests to optimize the development of micromagnetic devices.

The work at INESC-MN has a very strong experimental component, and one of the main objectives of this thesis was to also improve the simulation possibilities. INESC-MN mentors dozens of students each year who are taking their first steps in scientific research. This research is very commonly focused around the development of sensing devices for biological and biomedical applications, which are reliant on spin valve structures. Spin valves, as we have already introduced in the beginning of this thesis, are a commonly used device not only for sensing applications, but also for the construction of binary structures in MRAMs. In this chapter we will describe our work on the simulation of these structures. Our objective was to build a simple and organized tool that can be used by students at INESC-MN to test different geometries and parameters in the construction of spin valves, also focusing on the simplification of further improvements to the code. The fact that we have been working in Mathematica is advantageous given that this is a commonly used computational software that most students at INESC-MN already have experience with. We based our work on the thesis of Mendes [10], who had previously worked on the simulation of spin valve devices at INESC-MN.

Our work so far has been presented in either SI or natural units, discussed in section 2.3, as these are commonly found in more computational oriented publications. INESC-MN, however, works mostly in the Gaussian-cgs/emu system ($\mu_0 = 4\pi$), and so these are the units we adopt in this chapter.

<table>
<thead>
<tr>
<th>Quantity</th>
<th>SI Units</th>
<th>Conversion Factor</th>
<th>Gaussian-cgs Units</th>
</tr>
</thead>
<tbody>
<tr>
<td>Current</td>
<td>A</td>
<td>$10^{-1}$</td>
<td>emu</td>
</tr>
<tr>
<td>Dimensions</td>
<td>m</td>
<td>$10^2$</td>
<td>cm</td>
</tr>
<tr>
<td>Energy</td>
<td>J</td>
<td>$10^7$</td>
<td>erg</td>
</tr>
<tr>
<td>Magnetization M</td>
<td>A/m</td>
<td>$10^{-3}$</td>
<td>emu/cm</td>
</tr>
<tr>
<td>Magnetic Field H</td>
<td>A/m</td>
<td>$4\pi10^{-3}$</td>
<td>Oe</td>
</tr>
</tbody>
</table>

Table 5.1: Conversion from SI units to Gaussian-cgs units. Note that *emu* is short for "electromagnetic unit" is not an actual unit in the traditional sense as it can be used to represent different physical quantities. The main difference in this system is that $\mathbf{M}$ and $\mathbf{H}$ have different units, with an extra factor of $4\pi$ in the conversion factor, which is equal to a factor of $\mu_0$. Equations relating $\mathbf{M}$ and $\mathbf{H}$ in SI units being transformed into Gaussian-cgs should have $\mathbf{H}$ multiplied by a factor of $1/\mu_0$ to compensate.
5.1 Introduction to Spin Valves

The current flow through a spin valve can be done perpendicular to the plane of the layers (CPP) or through the plane of the layers (CIP). The devices at INESC-MN mostly employ CIP techniques where electrons interact with superficial interfaces between the layers, being scattered to other layers or reflected back to the same layer. More on this subject can be read in the publication by Barnaš et al. [61].

The most basic configuration for a spin valve is a $F_1/NM/F_2/AF$ structure, where $F_1$ and $F_2$ are ferromagnetic layers separated by a non magnetic (NM) spacer, and AF is an antiferromagnetic material. The AF layer, in direct contact with $F_2$, produces a strong exchange interaction which pins the magnetization of $F_2$, thus calling it the pinned layer. The $F_1$ layer on the other hand does not suffer this effect and is allowed to rotate more freely. The spacer, separating the free and the pinned layer, determines how these two layers interact with each other and influences the interaction of the electrical current with the spin valve. With the pinned layer fixed in place and the free layer rotating, the high and low resistance states are obtained for an antiparallel and parallel configurations between the layers. A simple scheme of a spin valve is represented in figure 5.1. From now on we use the superscripts ‘f’ and ‘p’ to refer to the free and pinned layer, respectively.

5.1.1 Antiferromagnetic Pinning

In order to obtain the desired effect from the free layer of the spin valve we must have another ferromagnetic layer that is essentially unaffected by the exterior magnetic fields that we wish to detect, thus remaining in a pinned state. As mentioned, this is done through the introduction of an AF layer. While the magnetic moments in an antiferromagnet tend to align in an antiparallel state with each other, it was discovered that a $F/AF$ coupling produces an unidirectional exchange coupling with high anisotropy [62] which has a strong pinning effect on the adjacent ferromagnetic layer.

In order to model this effect into a simulation, however, we do not explicitly simulate the AF structure. Qualitatively, the AF produces an offset in the hysteresis loop of $\langle M^p \rangle$ vs $H_{ext}$ and increases its coercivity. In the model this is replicated by the introduction of a new energy term of the Zeeman type\(^1\),

$$G_{pin}[M] = - \int_\Omega (M \cdot H_{pin}) \, d^3r,$$

where $H_{pin}$ is called the pinning field that acts on the magnetization of the pinned layer. The discretization of this term is straightforward, according to equation 2.20. The value of $H_{pin}$ is constant depending on

\(^1\)Note that we have multiplied $H$ by $1/\mu_0$ as to be consistent with Gaussian-cgs units.
the AF structure that is being represented and is equal to the offset observed in the hysteresis curve caused by each F/AF coupling, determined experimentally. The anisotropy constant of the pinned layer is usually set to be higher than the free layer to represent the increased coercivity.

5.1.2 Interlayer Coupling

The free layer of the spin valve does not suffer the effect of the pinning field, however it is also not entirely free to rotate. There are two interactions between the free and the pinned layer: an exchange-like coupling and a magnetostatic coupling.

The magnetostatic coupling is simply the effect the demagnetizing field of each layer produces on the other layers, since as we have discussed in this thesis, it is a long range interaction. Essentially, this interaction favours the antiparallel state of the spin valve. Calculating the global demagnetizing field in the whole structure would automatically account for this effect, however this would require the spacer to be discretized along with the ferromagnetic layers, where every cell of the spacer would have \( M_{i,j,k} = 0 \). This would greatly increase the computational cost of the simulation and limit the height of the discretization cell to be equal to the thickness spacer at maximum, which is usually in the range of \( 2.0 \sim 2.5 \text{nm} \) for copper (Cu) spacers. Instead, we introduce an extra field term to each layer negatively proportional to the average magnetization on the other layer,

\[
-H_{d-cp}^{f} = -n_d \mu_0 \langle M^p \rangle, \quad H_{d-cp}^{p} = -n_d \mu_0 \langle M^f \rangle,
\]

where we introduce the demagnetizing coupling factor \( n_d \) which is a small adimensional constant used to tune the magnetostatic coupling between the layers. As intended, this interaction favours the antiparallel state of the spin valve. We treat \( H_{d-cp}^{f} \) and \( H_{d-cp}^{p} \) as magnetostatic fields acting on \( M^f \) and \( M^p \) respectively, each having a corresponding discretized energy contribution to the whole system similar to equation 2.19.

The exchange-like coupling observed in spin valve structures acts through the spacer between the layers and happens due to various physical mechanisms. We are mostly interested in the Néel coupling [63], which originates from the material roughness at the surface, and the RKKY coupling [64], where it was found the RKKY interaction [65] between magnetic impurities serves as a good model to explain the oscillatory behaviour observed in this type of exchange coupling. Depending on the material of the spacer, one of these two exchange couplings will usually dominate over the other. For the case of Cu, for example, the dominating term is the Néel coupling, and its intensity decreases exponentially with the thickness of the spacer [10]. The RKKY coupling on the other hand has an oscillatory behaviour between ferromagnetic and antiferromagnetic coupling that varies with the thickness of the spacer. In the case of Cu this is simply a perturbation to the Néel term, however for other materials the RKKY coupling dominates and the thickness of these spacers can be calibrated to produce antiferromagnetic couplings between the layers (favouring the antiparallel state).

Essentially, when choosing a material to act as a spacer in the spin valve, these effects must be taken into account in order to produce the desired exchange coupling between the layers, whether it is
ferromagnetic or antiferromagnetic. Including these effects into the micromagnetic model can be done through the usual exchange interaction. Each of the top cells of the pinned layer interacts with the cell directly above it, which will be among the bottom cells of the free layer. We model this interlayer exchange based on equation 2.12 through an harmonic mean \[13\] between the material parameters of each layer,

\[
(H^{\text{ex-cp}}_{\text{f}}, i,j) = n_{\text{ex}} \cdot \frac{2}{(A^{\text{ex}}_{\text{f}}/M^{\text{f}}) + (A^{\text{ex}}_{\text{p}}/M^{\text{p}})} \left( \mathbf{m}_{i,j}^{\text{f}} - \mathbf{m}_{i,j}^{\text{p}} \right),
\]

\[
(H^{\text{ex-cp}}_{\text{p}}, i,j) = - (H^{\text{ex-cp}}_{\text{f}}, i,j),
\]

where we have omitted the cell index \(k\) since \(H^{\text{ex-cp}}_{\text{f}}\) only acts in the bottom cells of the free layer and \(H^{\text{ex-cp}}_{\text{p}}\) only acts in the top cells of the pinned layer. They only differ in sign since it corresponds to changing the order of the magnetization vector difference coming from the discretization of the Laplace operator. Here we have introduced another coupling parameter \(n_{\text{ex}}\) which can be used to model different exchange couplings with different intensities. Positive values of \(n_{\text{ex}}\) represent a ferromagnetic coupling between the layers while negative values produce an antiferromagnetic coupling. The energy related to this interaction is calculated similarly to equation 2.13.

### 5.2 Energy Minimization

Given our introduction to spin valves, we have presented three new interactions to be added to our micromagnetic model which are specific to these devices: the pinning effect, the exchange coupling, and the magnetostatic coupling. The interest in spin valves is to study how their magnetization distributions vary with a certain externally applied field, since that is what we wish to detect. In this section we will describe the relaxation procedure we implemented to find equilibrium magnetization distributions that locally\(^2\) minimize the energy of all micromagnetic interactions, according to the already introduced Brown equations 1.54.

For a magnetization distribution \(M^{\text{f}}\) on the free layer and \(M^{\text{p}}\) on the pinned layer, the effective field \(H^{\text{f}}_{\text{eff}}\) and \(H^{\text{p}}_{\text{eff}}\) that acts upon each is a sum of all the field terms so far introduced,

\[
H^{\text{f}}_{\text{eff}} = H^{\text{f}}_{\text{ex}} + H^{\text{f}}_{\text{an}} + H^{\text{f}}_{d} + H^{\text{f}}_{\text{ex-cp}} + H^{\text{f}}_{d-cp} + H_{\text{ext}},
\]

\[
H^{\text{p}}_{\text{eff}} = H^{\text{p}}_{\text{ex}} + H^{\text{p}}_{\text{an}} + H^{\text{p}}_{d} + H^{\text{p}}_{\text{ex-cp}} + H^{\text{p}}_{d-cp} + H_{\text{ext}} + H_{\text{pin}}.
\]

The base terms are calculated for each layer individually, as was done in previous chapters. Each term has its own energy contribution to the total energy that we wish to minimize. The algorithm we use is a conjugate gradient method based around the work of Berkov et al. [34]. For a given magnetization distribution and externally applied field we start by calculating the effective field of each layer and the corresponding total energy. With this magnetization state and its corresponding effective field terms

---

\(^2\)Local energy minima trap the magnetization distribution. Small changes in the external field allow the magnetization to move on to the next minimum. This is why we can obtain hysteresis curves when we perform field sweeps.
saved, we perform a relax iteration on both layers,

\[ m^* = m_{\text{old}} - \alpha_\tau \frac{1}{\mu_0} m_{\text{old}} \times (m_{\text{old}} \times h_{\text{eff}}), \]

\[ m_{\text{new}} = \frac{1}{\|m^*\|} m^*, \]  

(5.5)

where \( \alpha_\tau \) is the relaxation parameter, analogous to the damping parameter in the LLG equation. Note that each layer has natural units for \( m \) and \( h \) specific to each layer’s materials, according to section 2.3. Since these natural units were defined in terms of the SI system, here we must also include a \( \mu_0^{-1} \) factor to accommodate the Gaussian-cgs system.

For the new magnetization distributions, after renormalization, we must recalculate the effective field and corresponding total energy. If the new state is such that \( e_{\text{new}} < e_{\text{old}} \) then we save it and are ready to perform a new iteration. If however the iteration is not successful in reducing the total energy and \( e_{\text{new}} > e_{\text{old}} \), then we restore the previously saved magnetization state and all the field terms, reduce \( \alpha_\tau \rightarrow \alpha_\tau / 2 \) and repeat the iteration. After a certain number of continuously successful iterations we try to increase \( \alpha_\tau \rightarrow 1.1 \alpha_\tau \) to avoid very low relaxation parameters. Finally, when we reach the condition

\[ \max \left\{ \left| \frac{1}{\mu_0} (m \times h_{\text{eff}}) \right| \right\} < \epsilon_r, \]  

(5.6)

for a small value of \( \epsilon_r \), we stop the algorithm and consider the final magnetization state to be in equilibrium according to Brown’s equations 1.54. Note that condition 5.6 is evaluated for the free and pinned layer at the same time, so we require both to be under the torque threshold before we stop the minimization.

5.3 Basic Spin Valve Simulation

With the spin valve specific interactions incorporated into the model and the energy minimization algorithm implemented we are ready to test the operation of a simple spin valve.

The resistance variation related to the GMR effect has been experimentally correlated to the average angle between the magnetic moments of the free and pinned layer [66],

\[ \Delta R \propto - \langle \cos (\theta^p - \theta^f) \rangle. \]  

(5.7)

With this in mind, a well designed sensor will have the various micromagnetic interactions manipulated in order to produce the optimal variation in the quantity \( \langle \cos (\theta^p - \theta^f) \rangle \).

Let us start by presenting an example of a spin valve device. Following the simulation example in the work of Mendes [10], in figure 5.2 we represent a simple permalloy spin valve with a copper spacer, NiFe/Cu/NiFe. The free and pinned layers are equal, however the pinned layer has higher anisotropy to represent the effect of the omitted AF layer. We considered a pinning field along the \( y \) axis of 200 Oe. For the spatial discretization of the spin valve we considered a grid with \( 200 \times 40 \times 2 \) cells, which means we use one vertical cell for each layer.

In order to simulate this spin valve we must introduce the correct coupling to represent the 2.2 nm
We consider a pinning field of $H_{\text{pin}} = 200 \text{ e}_y \text{ Oe}$. The easy axis is also along the $y$ direction for both layers, however the anisotropy constant of the pinned layer is 20 times greater than the free layer.

copper spacer. We recall that in our model the coupling strength is tuned by the adimensional parameters $n_d$ and $n_{\text{ex}}$. For extended use, these parameters should be calibrated through experimental results and simulations for each type of spacer, and we will discuss more on this subject in appendix A. For now, we can estimate their value for this spacer by comparison with the results of Mendes [10], who uses a different coupling equation in the model. In his work Mendes reports an average magnetostatic coupling strength of 50 Oe and exchange coupling strength of 20 Oe. By setting the initial state of our spin valve as both layers magnetized along $y$, and evaluating the average values of $H_{\text{ex-cp}}^f$ and $H_{\text{d-cp}}^f$ we could estimate the same coupling by setting $n_d = 5 \times 10^{-3}$ and $n_{\text{ex}} = 1.6 \times 10^{-3}$.

The transfer curve of this spin valve was obtained with the simulation of a full field sweep between 300 Oe and $-400$ Oe. The external field was applied along the $y$ axis and swept in steps of 1 Oe. In figure 5.3 we present the transfer curve of the average magnetization component $\langle m_y \rangle$, averaged over the whole spin valve. For each step of the external field we used a stopping criteria for the energy minimization algorithm of $\epsilon_r = 5 \times 10^{-4}$. We found that using lower values of both the external field step and the minimization parameter $\epsilon_r$ no longer produced any relevant changes in the transfer curve. In these conditions we were able to complete the field sweep in about four hours with Mathematica 10 running on a Intel 3570k processor. We also included the results obtained from a MuMax [13] simulation. While the simulation in MuMax, which is programmed in CUDA [67], also took about four hours the hardware used was not ideal. On one hand we have Mathematica running on a high-tier desktop processor, on the other we have MuMax running on a slow GeForce 410M laptop graphics card.

The results we obtained for the transfer curve in figure 5.3 are similar to the results of Mendes [10], and we also see a very similar result compared to the MuMax simulation. The transition of $\langle m_y \rangle$ from 1 to 0 corresponds to the free layer rotating. The linear slope of this transition is essential to the operation of the sensor, and is governed by the aspect ratio $L/W$ of the spin valve. Higher aspect ratios increase the average strength of the demagnetizing field along the $x$ axis, which in turn difficult the transition of the magnetic moments between states perpendicular to this direction. As expected, the transition of the free layer has a positive offset of around 30 Oe given that we introduced a magnetostatic coupling with an average strength of 50 Oe and an exchange coupling with an average strength of 20 Oe. Comparing the two simulations in figure 5.3, the difference observed in the transition of this layer can be justified with the differences in the demagnetizing field method used, as we have already discussed in chapter 3.
MuMax also does not include an effective magnetostatic coupling like the one we presented in equation 5.2. While in some cases the spacer might be negligibly thin, it still should have some effect in reducing the magnetostatic coupling. Without an effective coupling we are forced to either neglect the existence of the spacer for the demagnetizing interaction or incorporate the spacer in the discretization, which greatly limits the cell size that can be used. In this regard, we opted to neglect the spacer layer in the MuMax simulation and we observe the respective transition of the free layer has a slightly higher offset than the expected 30 Oe. In general, the offset can be estimated from the average coupling strengths,

\[ H_{\text{off}} = -\langle |H_{\text{d-cpl}}^f| \rangle + \langle |H_{\text{ex-cpl}}^f| \rangle, \]  

(5.8)

since as we mentioned before, the magnetostatic coupling favours the antiparallel state between the layers and the ferromagnetic exchange coupling favours the parallel state.

The hysteresis curve of the pinned layer also matches the expected results. With a pinning strength of 200 Oe and an anisotropy field of around 100 Oe, we obtained a curve with high coercivity centered around \(-250\) Oe. The MuMax simulation on the other hand produced a hysteresis curve with considerably lower coercivity than what was expected from the anisotropy parameters introduced. While we used a conjugate gradient method for the energy minimization, MuMax uses a dynamic integrator of the LLG equation to perform the relaxation procedure\(^3\). An energy minimization algorithm like the conjugate gradient method, while generally faster, is also much more prone to find local minima in the energy landscape than a dynamic integrator, and this results in more points along the transition where the magnetization distribution gets trapped. This is exactly what we see when comparing the two transitions of the pinned layer in figure 5.3. Regarding the coercivity of the free layer, in our simulation it appears higher than it should be, considering this layer has an average anisotropy field of 5 Oe, however this is mainly due to common jumps in magnetization between neighbouring energy minima during the relaxation procedure that lead to artifacts in the curve.

\(^3\)A more recent version of the code already has a conjugate gradient method implemented.
In order to estimate the resistance variation of the sensor we plotted the quantity $-\langle \cos (\theta^p - \theta^f) \rangle$ in figure 5.4. The free layer rotation produces a linear resistance variation centered around 30 Oe, which would be the optimal operating region for this spin valve. We included the minus sign in the plot so that we can clearly see the transition from the low resistance state to the high resistance state when the layers go from parallel to an antiparallel orientation.

The spin valve we have presented serves as a simple example of what this type of structure can achieve. From this point on various cases can be studied, and more than two layers can be added to produce different effects. In the next chapter we will discuss some ideas on where to proceed with the development of this code.

In order to conclude this chapter let us overview the advantages of this simulation tool. First, as we have mentioned, *Mathematica* allows rather complex algorithms to be written in small portions of code, and this makes the code much easier to read and modify. Even if some efficiency is being sacrificed, someone using the tool will have much better control over what is being done in each routine when compared to lower level programming languages. Second, this tool consists of a single *Mathematica* file, or *Notebook*, organized with a modular approach. Both the easier to read code and the modular approach should benefit future implementations of other physical interactions. The versatility of the base software and its graphics engine also allow a wide range of data treatment techniques to be used in the same working environment as the simulation tool.

Our ultimate objective is to encourage students at INESC-MN to share and use this tool to aid their academic work, be it either computational or experimental. With this in mind, we have included in appendix B a brief overview of the *Notebook* from a user’s perspective, as well as some remarks on the organization of its source code.
Concluding Remarks

6.1 Thesis Overview

In this thesis we have overviewed the basic micromagnetic model and its numerical treatment with the focus on the simulation of micromagnetic devices. In the first chapter we introduced the main micromagnetic interactions based around a continuous magnetization framework, and their respective contributions to the free energy of the magnetic body. From the variational approach to the minimization of the free energy we introduced the effective magnetic field which is used to represent these interactions. Finally we introduced the LLG equation to describe the dynamics of the magnetization under the aforementioned effective field.

Regarding the numerical model, we focused our work on the treatment of the long range demagnetizing interaction as well as the time integration of the LLG equation. As we initially discussed in chapter 2 the choice of discretization of the equations behind these two problems is not trivial. For the treatment of the demagnetizing field problem we chose to pursue numerical methods that we can trust to conserve Maxwell’s relations by adopting the Yee cell discretization from the FDTD method. In the third chapter we started by solving the demagnetizing field problem in this discretization scheme with a linear system method, where we did not explicitly introduce a surface charge contribution to the demagnetizing potential. This method proved to be very reliable in conserving Maxwell’s equations, however rather inefficient when compared to FFT convolution methods which are more widely used in other simulations. By adapting the linear system method to calculate a convolution kernel for the demagnetizing potential, we presented a new method to solve the problem also based on FFT convolutions. This method maintained the reliability of the linear system method in the discretization scheme we adopted and surpassed the efficiency of other convolution methods by reducing the number of FFT operations. By averaging field components between the Yee cell discretization and the usual center-vector grid, we showed that a time integration of the LLG equation using our demagnetizing field method produces the same results for the standard problem #4 as other simulation tools.

Advancing to a more detailed study of magnetization dynamics, we started by underlining the important physical properties of the LLG equation regarding both norm and energy conservations. Most open source simulation tools use Runge Kutta methods for the numerical treatment of this equation, however these are not conservative. A review of the literature regarding conservative methods lead us to implement both a semi-implicit Gauss-Seidel method and a fully implicit Mid-Point method. For
the Gauss-Seidel method we adapted our work from chapter 3 to solve the required Screened Poisson equations through FFT convolutions. For the Mid-Point method we were required to solve a large sparse system of linear equations for each iteration of the quasi-Newton algorithm, for which we used the iterative Krylov method also based on our work from chapter 3. A comparison of the methods on the solution of the Standard Problem #4 provided consistent results and allowed us to conclude their implementation was done successfully.

In the final chapter we focused our work on the development of a spin valve simulation tool in Mathematica to be used at INESC-MN. Besides the main micromagnetic interactions we discussed in previous chapters, this tool also includes the exchange and magnetostatic coupling between each layer, as well as the possibility to pin the magnetization of one of the layers. We implemented a conjugate gradient energy minimization algorithm to study the variation of the magnetization distributions between the layers with the variations in a given externally applied field. By correlating the resistance variation of the spin valve to the angle between the average magnetic moment of each layer, this tool allows users to study the transfer curves of resistance for various spin valves with different geometries and materials. Being able to predict this type of behaviour is a valuable resource in the development spin valve devices, which can have both sensory as well as memory related applications. Furthermore, this tool was fully developed in Mathematica with a very compact code structure that is very easy to read and modify for future implementations of other micromagnetic interactions.

6.2 Comments and Future Work

During the development of this thesis various aspects surfaced that could have been further pursued. Regarding the model itself, there are numerous other physical interactions that can be studied and implemented numerically. Recently at INESC-MN there has been a growing interest in the study of the thermal fluctuations of magnetization distributions. The problem is that the introduction of a bias current in a magnetic sensor, used to read the resistance variation, leads to Joule heating on the spin valve. While there were recent plans to study the implications of this effect, the work did not move forward due to the limited access to a simulation tool where the thermal interactions could be implemented. An introduction to thermal interactions has been described in the work of Brown Jr [68]. Spin-transfer torque effects, which describe the interactions between magnetization distributions and spin polarized currents, have also originated interest from researchers. These currents have been found to produce torques upon the magnetization by transferring some of the spin angular momentum stored in conduction electrons [69, 70].

Regarding the numerical treatment of the demagnetizing field problem, as discussed in chapter 2, we used two different discretization schemes in order to conciliate the spatial discretization of Maxwell’s equations with the spatial discretization of the LLG equation. Using averages of the field components to transfer M and H between the two schemes is not ideal as it is a mathematical step with no physical meaning. During our work we considered the possibility of integrating the LLG equation in the conjugate dual grid scheme. Instead of a norm constraint of the type \( \| \mathbf{M}(\mathbf{r}, t) \| = M_s \) an extra term could be included in the free energy to penalize norm deviations from \( M_s \), which should have better physical
agreement with what is observed in reality\textsuperscript{1}. Future work in this regard should focus on a procedure to discretize the LLG equation in this scheme while correctly representing the physical and mathematical constraints of the equation.

As for the numerical integration of the LLG equation in the usual center-vector discretization, the numerical methods we implemented are promising, however further testing is still required to provide better comparisons. The numerical instability that surfaced during these tests, as discussed at the end of chapter 4, was found to be related to equation 4.31. An auxiliary Notebook was prepared to replicate this instability in its simplest form and study this equation. Hopefully further testing will allow the origin of the noise propagation to be determined, and allow the LLG simulations to reach their full potential.

With the spin valve simulation tool operational, the next step should be to include support for a third layer. Spin valves with three ferromagnetic layers allow a synthetic antiferromagnetic (SAF) structure to be included along with the free layer. A SAF structure is a stack of the type F\textsubscript{1}/NM/F\textsubscript{2}/AF, where F\textsubscript{1} and F\textsubscript{2} are separated by a spacer with an antiferromagnetic coupling. Instead of using a F/AF structure as a pinned layer, a SAF can provide a stronger pinning effect, where the pinning field is sometimes one order of magnitude larger than a F/AF structure. Another advantage is that, since F\textsubscript{1} and F\textsubscript{2} remain antiparallel (in the operating region of the spin valve) their net magnetic moment is very small, and so the magnetostatic coupling felt by the free layer is significantly decreased. By decreasing the magnetostatic coupling we can tune the offset field on the transition of the free layer. Since the offset field determines the operating region of the spin valve, this is an important aspect in some applications. One example is the case of biochip sensors that are used at INESC-MN to detect magnetically tagged biomolecules, which produce very small fields in the \{-2, 2\} Oe range. Such an application would ideally require a sensor with zero offset field. Essentially, the code simply needs to be adjusted to include a third layer that will be antiferromagnetically coupled to the middle layer. The logic for this adjustment will be the same as the already implemented double layer interactions, and this will be a simple task with the compact structure of our Mathematica code.

\textsuperscript{1}Considering $\|\mathbf{M}(\mathbf{r}, t)\| = M_\text{s}$ assumes every single electron in a ferromagnet is polarized.
Bibliography


[50] Electric potential of a uniformly charged cuboid.

https://www.wolframalpha.com/input/?i=charged+cuboid+potential.


[52] μMAG – Micromagnetic Modeling Activity Group.


[67] Cuda.


[71] Fenics.
   https://fenicsproject.org/.
Open Source Codes

Before developing our simulation tool we overviewed various open source micromagnetic codes to evaluate their strengths and weaknesses. Various publications mention OOMMF [12] as a very well established micromagnetic simulation tool. Developed around 1999, it still receives support and updates up to this date, and is commonly cited as the code of comparison in the development of other simulation tools. In this respect, however, we looked for more recent codes, where we considered MuMax [13, 14], magnum.fe [15] and MicroMagnum [16].

Out of the three mentioned tools, only magnum.fe uses a finite-element discretization scheme. This code runs as a Python library, however it also requires the external finite-elements library FEniCS [71]. During the installation we could not get the library dependencies to work correctly and we were unable to compile magnum.fe. By contacting the developer it was suggested that the problem could lie in the fact that the latest version of FEniCS was not yet supported by magnum.fe. Given the compilation problems, we did not run any more tests with magnum.fe and excluded it from the rest of our tests.

Both MicroMagnum and MuMax were installed successfully, and in this appendix we will provide some commentary and comparisons given the experience we had with both these codes. Ultimately, MuMax was the simulation tool we most frequently used. All the tests were performed in our working laptop:

- CPU: Intel i5-2430M @2.40GHz, GPU: Nvidia GeForce 410M, RAM: 4 GB.
- Operating System: Linux Mint 17.2 KDE (based on Ubuntu 14.04).

A.1 Tool Comparison

As mentioned, both MuMax and MicroMagnum use a finite-difference discretization. The source code of MuMax is programmed in Golang Go with CUDA routines written in C. MicroMagnum is programmed as a Python module with some routines written in C++. One of the main differences that should be taken into consideration when choosing between the two codes is that MuMax is programmed solely using the CUDA platform [67], which requires a Nvidia GPU. MicroMagnum also has CUDA support, however it still works only with CPU calculations if a supported GPU is not installed. From a user perspective MicroMagnum requires the simulation input to be written as a Python script, while MuMax uses a simplified scripting language specific to the tool.
A.1.1 Installation

At the time of writing, the installation instructions for both tools can be found in their respective online resources [14, 16]. While there are instructions to install both tools in Windows, we only considered the Linux case. In this respect, working with MicroMagnum is more straightforward since the only required packages, Python and FFTW, can be found in the Ubuntu repositories and should pose no problems during installation. As mentioned, the CUDA packages are optional for this tool. Regarding MuMax, the main Golang Go packages are also easy to install from the repositories. The only difficulty can be with the CUDA dependency, since the appropriate Nvidia graphics driver needs to be installed and configured for the CUDA packages to work properly. Various guides on how to install and configure CUDA on Unix systems can be found online depending on the Linux distribution.

With the needed dependencies installed the source code of both tools needs to be initially compiled. Both MicroMagnum and MuMax have an open source code, allowing user made changes to be implemented, after which the code needs to be recompiled. In our tests we were only able to successfully recompile and implement source code changes using MuMax, however this was not possible with all versions of the dependent packages. We found that using version 7.0 of CUDA, version 1.3 of Golang Go and version 3-3.8 of the MuMax source code everything compiled correctly, however we had to test various combinations until we found one that did not produce any errors.

A.1.2 User Interface

When it comes to user interaction with the simulation tool, both MicroMagnum and MuMax use input text scripts where the information related to the simulation is layed out. The simulation tool then interprets the instructions presented and completes the calculations needed, saving the desired output as indicated in the script.

Below we present two examples of the scripts required to solve the Standard Problem #4 [51] in both simulation tools. As we have mentioned before, MicroMagnum uses a script written in Python, while MuMax has its own scripting language. Instructions on how to run the scripts for each simulation tool can be found in their respective online resources [14, 16].

```
#!/usr/bin/python
from magnum import *
from math import pi, cos, sin

# ###########################################################
# Create a world
# ###########################################################
world = World(RectangularMesh(((128, 32, 1), (500e-9, 125e-9, 3.0e-9)), Body("all",
Material.Py(alpha=0.02)))

# ###########################################################
# Relax an s-state as the initial magnetization of the SP4
# ###########################################################
def make_initial_sp4_state():
    # Specify an s-state-like starting state
```
def state0(field, pos):
    u = abs(pi*(pos[0]/field.mesh.size[0]-0.5)) / 2.0
    return 8e5 * cos(u), 8e5 * sin(u), 0

# Relax to get initial state for SP4
solver = create_solver(world, [StrayField, ExchangeField], log=True, do_precess=False,
    evolver="rkf45", eps_abs=1e-4, eps_rel=1e-2)
solver.state.M = state0
solver.state.alpha = 0.5
solver.relax(1.0)
return solver.state.M # return the final magnetization

# Apply an external field H on initial magnetization M0
# 

def apply_field(M0, H, file_prefix):
    solver = create_solver(world, [StrayField, ExchangeField, ExternalField], log=True,
        evolver="rkf45", eps_abs=1e-4, eps_rel=1e-4)
solver.state.M = M0
    solver.state.H_ext_offs = H
    solver.addStepHandler(ZeroCrossChecker(), condition.Always())
    dtsh = DataTableLog(file_prefix + ".odt", title = file_prefix)
    dtsh.addEnergyColumn("E_stray")
    dtsh.addEnergyColumn("E_exch")
    dtsh.addEnergyColumn("E_exch")
    dtsh.addEnergyColumn("E_ext")
    dtsh.addEnergyColumn("E_tot")
solver.addStepHandler(dtsh, condition.EveryNthStep(10))
solver.solve(condition.Time(1.0e-9))

# Main program
#
M0 = make_initial_sp4_state()
writeOMF("sp4_M0.omf", M0)
apply_field(M0, (-24.6e-3/MU0, +4.3e-3/MU0, 0.0), "sp4-1")
apply_field(M0, (-35.5e-3/MU0, -6.3e-3/MU0, 0.0), "sp4-2")

Listing A.1: MicroMagnum script for the standard problem #4.
Immediately we notice MuMax takes a much more simplistic approach to the user interface. Depending on the type of user and their preference, both scripts can have their advantages. On one hand the MicroMagnum script has a lot more information on the main interface, and the fact that it is written in Python might improve customization for someone who already has knowledge of this programming language. On the other hand the MuMax script is much easier to read, and it also has various extra options that can be added to further customize the instructions in the script. The online resources of MuMax [14] provide a complete overview of all the available functions and options that can be included in the input script. In this respect we found the online resources of both MuMax and MicroMagnum to be very complete. Regardless, having no experience with Python and being completely new to micromagnetic simulations, we initially found the scripting language and overall resource organization of MuMax to be more inviting.

A.1.3 Final Remarks

In terms of performance we noticed slightly faster computation timings using MuMax on the solution of the standard problem #4. Both MicroMagnum and MuMax use FFT convolution methods for the solution of the demagnetizing field and explicit Runge-Kutta integrators with adaptative time step for the time integration of the LLG equation, as discussed in chapters 3 and 4 respectively, so the observed speed difference is most likely related to code optimizations.

One aspect that might not be as important to all users but also requires some comments is the organization of the source code. In this respect we found that source code navigation in both tools was quite troublesome. The fact that both MuMax and MicroMagnum use a source code structure with links between different programming languages and platforms leads to a large amount of source files organized in a very obfuscated manner. Regardless, we managed to partially work through it and successfully implement source code changes which were quite useful during the instability tests we mentioned at the end of chapter 4. We suggest users to explore the engine and cuda folders. From our knowledge, the engine files are the Golang Go routines that are called by the main program, which then make calls to the .cu CUDA routines in the cuda folder, where the actual calculations are performed.

Having two different open source simulation tools as developed as MuMax and MicroMagnum is always useful to cross check simulations. However, after our initial installation and tests, we found MuMax to offer a better user experience both in terms of speed and organization, and it was the simulation tool we most often used during our thesis.
In chapter 5 we talked about the simulation of spin valve structures. As mentioned, during our work we developed a simulation tool in Mathematica to be used by students at INESC-MN. In this appendix we will discuss some details about its functionality.

The only requirement to use this tool is to have the main Mathematica software installed. Note however that we used version 10, and so older versions of Mathematica may not have all the required functions incorporated. Students enrolled at IST are included in the Mathematica for Students partnership, and have automatic access to the latest version of the software. With the software installed in any computer, all the information required to run simulations is coded in the same Notebook, so this is the only file users need to interact with.

### B.1 User Interface

The code in a Mathematica notebook is written in cells. Each cell can then be evaluated by the user, producing an output according to the instructions in that given cell. Our tool has two types of cells: initialization cells, where we implemented the source code, and user cells, where instructions for the simulation can be defined. In this section we will only discuss the user cells, leaving the initialization cells for the source code discussion in section B.2.

User cells are essentially the basic type of cell in Mathematica that is created when any code is written. In our tool we include a set of default user cells already prepared start the simulation, which have a light blue background, however more can be created freely in order to do any other simulation tasks a user requires. These are the only visible cells when the notebook is launched, and are the only cells users need to interact with to perform a simulation with the already implemented interactions. The very first user cell at the beginning of the notebook deals with the size of the system, where both the physical dimensions of the system and the dimensions of the discretization grid must be defined:

```mathematica
(* Dimensions must be provided in cm *)

(* Length and Width *)
xx = 5 \times 10^{-4};

(* Spacer Thickness *)
tS = 2.2 \times 10^{-7};

(* Pinned Layer Thickness *)
```
\[ t_P = 5 \times 10^{-7}; \]
\[ t_F = 5 \times 10^{-7}; \]
\[ (* Free Layer Thickness *) \]
\[ n_X = 200; \]
\[ n_Y = 40; \]
\[ n_Z = 2; \]
\[ n_{ZF} = 1; \]
\[ n_{ZP} = 1; \]
\[ initializeGeom[]; \]

Listing B.1: First user cell in our Mathematica tool.

Note that \( n_Z \) refers to the total number of cells in the \( z \) direction, which must be divided appropriately between the vertical cells for the free layer \( n_{ZF} \) and the pinned layer \( n_{ZP} \). While most actions in this cell simply assign numerical values to variables, the last action calls an external function \( initializeGeom[] \). This function is the first example of an initialization cell that contains a set of source code instructions. In this case the function is called right away in this first cell and simply does a couple of operations on the dimensions that were introduced by the user, as we will show in the next section.

The next user cell assigns the values of the material parameters for both the free and pinned layer, as well as the coupling constants between them, and also strength and direction of the pinning field:

\[ (* aEx - (erg/cm) - Exchange Constant *) \]
\[ (* mS - (emu/cm) - Saturation Magnetization *) \]
\[ (* k1 - (erg/cm^3) - First Order Anisotropy Constant *) \]
\[ (* kvec - \{x, y, z\} - Easy Axis Direction *) \]
\[ (* Free Layer *) \]
\[ aExF = 1.3 \times 10^{-6}; \]
\[ mSF = 800; \]
\[ k1F = 2.0 \times 10^3; \]
\[ kvecF = \{0, 1, 0\}; \]
\[ (* Pinned Layer *) \]
\[ aExP = 1.3 \times 10^{-6}; \]
\[ mSP = 800; \]
\[ k1P = 4.0 \times 10^4; \]
\[ kvecP = \{0, 1, 0\}; \]
\[ (* Interlayer Coupling Factor *) \]
\[ sCoup = 1.6 \times 10^{-3}; \]
\[ (* Interlayer Demag Factor *) \]
\[ mDemag = 5 \times 10^{-3}; \]
\[ (* Pinning Field *) \]
\[ pinStr = 200; \]
\[ pinDir = \{0, 1, 0\}; \]
\[ initializeParam[]; \]

Three dimensional vectors introduced as input for both the direction of the easy axis of anisotropy on the layers and the direction of the pinning field will be automatically renormalized as unit vectors.

Finally, the only thing left to do to define the initial state of our system is to introduce some initial magnetization distribution. Currently we can either define an homogeneous magnetization along the
direction of a unit vector for both layers, or a random magnetization can be set:

```plaintext
(* Free Layer Initial Magnetization *)
mIniF = vec[0.1, 1, 0];

(* Pinned Layer Initial Magnetization *)
mIniP = vec[0, 1, 0];

(* Add Perturbation? *)
pertMag = True;

(* Random Initial Magnetization? *)
randomMag = False;
initializeArrays[];
```

The random magnetization option, if set to True, will override the uniform magnetization distribution. There is also the option to add a perturbation to the initial magnetization so that the vectors between each cell are not totally parallel. Once the initializeArrays[] function is called, the notebook creates the three dimensional arrays that will store the magnetization vectors for each cell and the corresponding effective field contributions. The two main arrays are mF and mP which store the values of mf and mp, and will be discussed in the next section along with the general organization of the data arrays in the tool. With the initial magnetization set the system is fully described and we are ready to perform calculations with the implemented physical interactions.

As discussed in chapter 3, the FFT convolution method for the demagnetizing field requires the convolution kernel to be computed at the beginning of the simulation:

```plaintext
outSpace = 64;
Print["Calculating free layer demagnetizing kernel... "]; t1F = (poissonKernel[0, outSpace, nX+1, nY+1, nZF+1];//AbsoluteTiming)[[1]]; recipDemagKernelF = recipKernel;
Print["Calculating pinned layer demagnetizing kernel... "]; t1P = (poissonKernel[0, outSpace, nX+1, nY+1, nZP+1];//AbsoluteTiming)[[1]]; recipDemagKernelP = recipKernel;
Print["Done!"];
Print["Kernels calculated in ", t1F+t1P, " seconds."];```

This cell calculates the demagnetizing kernel for both the free and the pinned layer, and needs to be evaluated by the user only once after setting the values of their dimensions. If these values are changed while working with the notebook, the kernels must be reevaluated. Note that the constant outSpace represents the δ parameter we introduced in section 3.3.2.

The next step to study the behaviour of a spin valve is to set the conditions of the external field sweep that is going to be simulated:

```plaintext
SetDirectory[StringJoin[NotebookDirectory[], " Spin Valve 8"]];

(* Sweep Range *)
w = Join[{-300, 400, 1}, Delete[Range[{-400, 300, 1}, 1]]]/N;

sweepDir = vec[0, 1, 0];

(* Add noise to Zeeman Field? *)
zeemNoise = False;
```
Since files are going to be exported with the data of the field sweep, we recommend setting the working directory of the notebook to the directory of the notebook file, which is not always set by default. The array of external field values should be assigned to the variable `sweep`. Here we use the *Mathematica* functions `Range[min_x, max_x, dx]` and `Join[array_1, array_2]` to produce the desired array of values, which is what we used for our example in figure 5.3. The direction along which the sweep will be performed must also be defined. The final option lets the user choose if the relaxed magnetization distributions, stored in `mF` and `mP`, will be exported. These can be rather large files depending on the number of cells of the discretization, however it might be useful to backup the magnetization files of a given field sweep in case the user wishes to come back to the data later. Here we exported the magnetization distribution every ten steps of the field sweep.

Finally, we include a simple algorithm to perform the actual field sweep:

```mathematica
initializeArrays[];
  sweepPlot = ConstantArray[0., {Length[sweep], 4}];
  n = saveN;
  sweepDir = sweepDir/Norm[sweepDir];
  Do[
    hExt = sweepDir*sweep[[i]];
    relaxMag[];
    sweepPlot[[i, 1]] = sweep[[i]];
    sweepPlot[[i, {2, 3, 4}]] = Mean[{Mean@Flatten[mF, 2], Mean@Flatten[mP, 2]}];
    If[n == saveN, 
      Print["Completed with Zeeman field at", hExt];
      If[saveMag == True, 
        Export["mF" <> ToString[i] <> ".wdx", mF, "WDX"]; 
        Export["mP" <> ToString[i] <> ".wdx", mP, "WDX"];
        n = 0; 
      ]; 
    ]; 
    n++; 
  , {i, 1, Length[sweep]}]
Export["sweepPlot.wdx", sweepPlot, "WDX"];```

The main section is a loop that runs through the sweep array. For each sweep value, the external field vector `hExt` is calculated according to the direction of the sweep, which will then be applied to every cell during the relaxation. The `relaxMag[]` function performs the algorithm described in section 5.2. Once the energy has reached a minimum and the magnetization distribution is in a relaxed state for the corresponding external field value, the information is stored and the field step is complete. Storing the information related to the field sweep can be done in many ways, depending on the interest of the user. For example, the function `Mean@Flatten[mF, 2]` outputs a three component vector which represents the average magnetic moment in units of $M_s$ in the free layer. In our example we defined the array `sweepPlot`
with four entries so that we could save the average magnetization components \( \{m_x, m_y, m_z\} \), averaged between the free and the pinned layer, for every external field value. This was the information we used to plot figure 5.3. Instead, we could also save the cosine of the angle between the vectors \( \text{Mean@Flatten}[mF, 2] \) and \( \text{Mean@Flatten}[mP, 2] \), or calculate this quantity later by importing the saved \( mF \) and \( mP \) arrays, which is what we did to plot figure 5.4.

## B.2 Source Code

In the previous section we overviewed the user interface of our Mathematica notebook, namely the user cells that we defined. In some of these cells we founds calls to source code functions that actually perform the calculations. The source code of the tool is organized in initialization cells, which have a light gray background. When the notebook is opened and the first cell is evaluated, Mathematica should ask the user if they wish to evaluate all the initialization cells. This option should be accepted before doing anything else so that the notebook will function properly. Before we explain what this does, let us look at the first example of an initialization cell:

```mathematica
initializeGeom := Module[{} ,
  z = N@ (tP + tF);
  deltay = N@ (yy/nY);
  deltaz = N@ (zz/nZ);
  If[nZF+nZP != nZ , Print["nZF + nZP != nZ, Please correct the discretization.""];]
```

Listing B.2: First initialization cell in our Mathematica tool.

This cell contains the code that was called by the `initializeGeom[]` function in the first user cell. In our initialization cells we use the Mathematica function `Module[]` to define modules of code that do a certain task. By setting `initializeGeom := Module[...]` we instruct that the code inside this module will not be evaluated immediately, but will be stored in the function `initializeGeom[]` to be evaluated only when this function is called. In this example the module does not take any arguments and only deals with the dimensions of the system, which are global variables in the notebook. The instructions presented are simply to calculate the cell size and verify if the number of vertical cells was correctly distributed between the free and the pinned layer.

Every source code related routine in our notebook is organized in this way, and essentially they are all evaluated at the beginning of the simulation, however none of their tasks are actually performed until the respective function is called. An example of a more complex function in an initialization cell that requires some arguments is the function `poissonKernel[]`, which as we mentioned in the previous section, calculates the demagnetizing kernel for a given geometry:

```mathematica
poissonKernel[lambda_, outSpace_, knX_, knY_, knZ_] := Module[{},
  (* Boundary Calculations *)
  outX = outSpace;
```

87
The code in this cell follows the numerical method we described in section 3.3.2, which should be reviewed to better understand the following discussion. We also recall from our work in section 4.2.2 that we extended our method to provide solutions to screened Poisson equations, and so the first argument this functions requires is the screening parameter \( \lambda \), however this is 0 for normal Poisson equations. The other arguments are simply the \( \delta \) parameter and the size of the grid for which the kernel is being calculated. Following the description of section 3.3.2 the first instructions in this initialization cell defines the number of points on the boundary for which the monopole approximation is calculated. Then, each task required to reach the final demagnetizing kernel by itself is its own function defined in another initialization cell. Starting with the boundary conditions, \( \text{boundPoints[]} \) defines the coordinates of the selected boundary points, \( \text{monopoleCalc[]} \) places a unit charge in the first corner cell of the grid, \( \text{boundInteraction[]} \) calculates the monopole approximation for the potential at the defined boundary points, and finally \( \text{boundInterpolation[]} \) interpolates the values of all the boundary cells between the previously defined boundary points. With the boundary conditions calculated, the linear system is created and solved. The solution of the linear system is then used to construct the kernel. The organization of all the source code functions in initialization cells is done through sections and subsections in the notebook. While all the source code sections are minimized by default, their titles are visible and navigating through the code should be very simple.

Naturally it is not viable to discuss every single source code cell in this appendix, however it is important to discuss some details regarding the organization of the data arrays used in the calculations. When dealing with arrays of data, \textit{Mathematica} has an option to check if an array is \textit{packed}. When an array is not packed \textit{Mathematica} assumes it can contain anything, from reals to integers or even strings, and so operations made with these arrays take a lot longer. If however an array contains only one type
of data it can be packed, instructing *Mathematica* to store it in a more efficient way and thus greatly increasing the speed of calculations made with it. The function `Developer'PackedArrayQ[]` can be used to check if an array is packed, and `Developer'ToPackedArray[]` can be used to pack an array that contains only one type of data\(^1\). In our simulation we work with double precision reals, and each array is defined to be packed from the start. All the operations in the notebook are defined carefully so as to not break the packed state of the data arrays, and this is essential to maintain the efficiency of the tool.

Below we present a list of some useful arrays and functions in the notebook.

<table>
<thead>
<tr>
<th>Arrays</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>(mF), (mP)</td>
<td>Magnetization unit vector field (m^f) and (m^p)</td>
</tr>
<tr>
<td>(mFmS), (mPmS)</td>
<td>Magnetization vector field (M^f) and (M^p)</td>
</tr>
<tr>
<td>(h\text{Exch} F), (h\text{Exch} P)</td>
<td>Exchange field (H^{\text{ex}}_f) and (H^{\text{ex}}_p)</td>
</tr>
<tr>
<td>(h\text{Anis} F), (h\text{Anis} P)</td>
<td>Anisotropy field (H^{\text{an}}_f) and (H^{\text{an}}_p)</td>
</tr>
<tr>
<td>(h\text{Demag} F), (h\text{Demag} P)</td>
<td>Demagnetizing field (H^{\text{d}}_f) and (H^{\text{d}}_p)</td>
</tr>
<tr>
<td>(h\text{Zeem} F), (h\text{Zeem} P)</td>
<td>Zeeman field (H^{\text{ext}}_f) and (H^{\text{ext}}_p)</td>
</tr>
<tr>
<td>(h\text{Coup} F), (h\text{Coup} P)</td>
<td>Exchange coupling field (H^{\text{ex-cp}}_f) and (H^{\text{ex-cp}}_p)</td>
</tr>
<tr>
<td>(h\text{Dem} IF), (h\text{Dem} IP)</td>
<td>Magnetostatic coupling field (H^{\text{d-cp}}_f) and (H^{\text{d-cp}}_p)</td>
</tr>
<tr>
<td>(h\text{Pin})</td>
<td>Pinning field (H^{\text{pin}})</td>
</tr>
<tr>
<td>(h\text{Eff} F), (h\text{Eff} P)</td>
<td>Effective field (H^{\text{eff}}_f) and (H^{\text{eff}}_p)</td>
</tr>
</tbody>
</table>

All free and pinned layer arrays have dimensions \([nX, nY, nZF, 3]\) and \([nX, nY, nZP, 3]\), respectively.

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>reNorm[m]</code></td>
<td>Renormalizes all cells in the unit vector field (m). Example: (mF = \text{reNorm}[mF]);</td>
</tr>
<tr>
<td><code>runHeff[]</code></td>
<td>Calculates the effective field for the current (mF) and (mP) distributions. Requires the demagnetizing kernels to have been evaluated. The values of each field are stored in the respective arrays.</td>
</tr>
<tr>
<td><code>energyCalc[]</code></td>
<td>Calculates the energy given the current magnetization and effective field. Saves the value of each contribution in (e\text{Exch} F), (e\text{Exch} P), (e\text{Demag} F), etc. The total energy of the system is stored in (e\text{Tot}).</td>
</tr>
<tr>
<td><code>relaxMag[]</code></td>
<td>Minimizes the total energy of the system given the current external field. When complete, the distributions (mF) and (mP) will be in a relaxed state.</td>
</tr>
</tbody>
</table>

Table B.1: Main arrays defined in the spin valve simulation tool.

Table B.2: Main functions defined in the spin valve simulation tool.

Many other auxiliary arrays, variables, parameters, and functions, as well as intermediate functions to perform smaller tasks that were discussed over the course of this thesis can be found within the notebook.

Time integration algorithms related to our work in chapter 4, while not included in this specific notebook, were coded using the same logic and can be easily added as additional modules.

---

\(^1\)Note that using only the function `N[]`, which gives the double precision numerical value of an array does not automatically pack it. Another detail is that initializing an array with `ConstantArray[]` produces a packed array, while `Table[]` does not.
During our work we presented a scientific poster *Micromagnetic Device Simulation* at Jornadas da Engenharia Física (III JEF) – Instituto Superior Técnico, Portugal. Our poster, included in the next page, obtained the Honorable Mention prize in the competition.
Micromagnetic Device Simulation

João Pedro Gomes Moutinho
Instituto Superior Técnico – MEFT 2015/2016
INESC - Microsystems and Nanotechnologies

Contact: joao.p.moutinho@gmail.com

Introduction

Being able to predict and control the behaviour of magnetic materials is extremely important, given that these materials are the building blocks of many of today’s technology. From magnetic recording, with ever increasing information density, to the construction of magnetic sensors, the development of these technologies keeps growing. In this poster I will introduce you to my master’s thesis project, Micromagnetic Device Simulation, work that I will be developing at INESC-MN in the next few months under the supervision of Prof. Dr. José Luís Martins and Prof. Dr. Susana Freitas to obtain the Master Degree in Engineering Physics. The micromagnetic device development group at INESC-MN relies on micromagnetic simulations as a supplement to the experimental work to better understand and optimize each device’s inner workings.

Basic Concepts

The most important aspect of ferromagnetic materials is their domain structure. The molecular field in a ferromagnet is strong enough to magnetize the substance to saturation even in the absence of an applied field. After magnetized, the sample would create a magnetic field upon itself, called the demagnetizing field (Fig 1). However, the minimization of the surface energy in the sample causes it to break down into magnetic domains. Each domain is fully magnetized to saturation, but in a different direction, so that there is no net magnetization in the sample as a whole.

With the domains already formed the process of magnetizing a sample is much easier: an external applied field simply rotates the magnetized domains, reversing the process we see in figure 1.

These are the two main aspects that must be considered when modeling this ferromagnetic behaviour: calculating the demagnetizing field and representing the rotation of the magnetizations through the Landau-Lifshitz-Gilbert dynamics.

Demagnetizing Field

Starting from Maxwell's equations we can write for the demagnetizing field:

\[ \nabla \cdot \mathbf{H}_d = \nabla \cdot \mathbf{M} \]  

Equation 2 is called the micromagnetic Poisson problem, and is the main equation used to calculate the demagnetizing field. This is a very resource intensive calculation due to the nature of the Poisson equation.

This computation can be accelerated with a Fast Fourier Transform based method. This method is used often in micromagnetic simulations, and has proven to be reliable in dealing with the space discretization and the boundary conditions [1].

Our idea is to apply a finite difference discretization directly to equation 2.

Writing equation 3 for every cell of space gives us a huge linear system. The calculation for each cell, however, only depends on the information stored in neighboring cells, and so the linear system is sparse!

Advantage: Much easier to parallelize. While FFTs are global in sparse systems you can, for example, have each processing unit taking care of computing one cell of space. With an ensemble of processing units such as the one represented on figure 3, each one only needs to interact with the neighboring units, minimizing delays due to information exchange.

Landau-Lifshitz-Gilbert Dynamics

Using an open source micromagnetic code we simulated the magnetization inversion in a ferromagnetic disk. Initially there is a single domain with a magnetization pointing to the right. We can see different domain forming, locking to a null net magnetization in the middle state. Then they start rearranging into a new single domain, with the magnetization now pointing to the left.

Simulation Example

References

Examples of figures from previous posters:

- Figure 1: Precessional motion

A list of references and links to more information.

Details

Examples of figures from previous posters:

- Figure 2: B. D. Cullity, C. D. Graham, Interactions in magnetic materials, John Wiley & Sons (2009)
- Figure 3: http://www.aacg.bham.ac.uk/magnetic materials/domains.htm
- Figure 4: http://www.iue.tuwien.ac.at/phd/makarov/dissertation59x.png
- Figure 5: http://www.hcs.ufl.edu/
- Figure 6: http://www.aacg.bham.ac.uk/magnetic materials/domains.htm

A list of references and links to more information.

Footnotes: