Metropolis Monte Carlo Micromagnetics Simulator

Thomas William Unitt-Jones

September 2023

Abstract

Monte Carlo simulations for magnetics in both the atomistic and continuous paradigm have an advantage over their non-stochastic counterparts in that they can model at elevated (non-zero) temperatures and are naturally parallelisable. The preferred Monte Carlo algorithm is the Metropolis algorithm using the Boltzmann factor of the current state's energy and a proposed state's energy as the probability of transition between them. This method has been used to successfully simulate systems at elevated temperature but these simulations prove to be slow if not parallelised since they require many iterations until they stop evolving. We implement a simulation framework for Zeeman, anisotropy, exchange, and DMI interactions in both atomistic and continuous set-ups. It will be integrated into the open-source Ubermag API for Python-based micromagnetic simulations, making it available to the community. This will enable running Monte Carlo simulations within the Python ecosystem, in particular exposing the driver to Jupyter Notebook where transparent, reusable workflows can be documented.

Contents

1 Introduction

In section 1, we will discuss the emergence of computer science as a key constituent of research. We will motivate the study of micromagentics and explain how computational models can be used to carry out research in this field, discussing the Metropolis Monte Carlo algorithm in the context of micromagnetics. In section 2, we will summarise the four magnetic interactions treated by this simulator.

With remarkable advances in computer technology, computational science has become the third pillar of research alongside theory and experiment in both industry and academia. In the computer age, computational science now bridges the gap between theory and experiment by providing a flexible framework in which to experiment with the whole scope of possible and impossible configurations. It also allows for previously intractable equations to be approximated by numerical methods. In science and engineering, computational models have become the laboratories in which researchers now pioneer new ideas before committing them to real world experiments.

Micromagnetics, sometimes called nanomagnetics, is the study of magnetic interactions on the nanoscale. Since the atomic spacing of many common magnetic materials is on the order of magnitude of the ångström (1Å=0.1nm), micromagnetics is concerned with the dynamics of the magnetism of regions containing a small number of atoms or even single atoms.

Currently, the main interest in micromagnetics is in understanding the properties of nanoscale quasiparticles, namely the two-dimensional skyrmion [1, 2] and the three - dimensional Bloch point [3], because these non-trivial topological arrangements show potential to revolutionise the way we store and communicate information [4]. Both arise due to the Dzyaloshinskii-Moriya interaction (DMI) [5, 6, 7, 8] found in magnetic materials with broken inversion symmetry. Skyrmions were hypothesised in the 1960s and observed experimentally for the first time in 2009 [9]. Researchers have proposed for them to be used in information storage [4] and logic computing gates in neuromorphic devices (biologically inspired computers) [10]. Other research applies them to racetrack memory-like set-ups [11]

as quick and robust memory read-write alternatives to mechanical hard disks where the presence of a skyrmion represents 'one' and the lack thereof represents 'zero' bits.[1] Whereas skyrmions emerge in regions of constant DMI, Bloch points have been shown to occur on the boundaries between two nanodiscs of different chiralities and can exist in two types (HH and TT) [13]. It is hypothesised [13, 14] that if used in a racetrack-memory type scenario, the two types of Bloch points could represent the 'one' and 'zero' bits, giving Bloch point racetrack memory an advantage over the skyrmionic one in which the nano-particles must maintain their spacing.

Beyond memory-based applications, further research looks into utilising nanoscale magnetic particles in biomedicine where synthetic magnetic nanoparticles (SMN) can be used as probes for molecular imaging for improved magnetic resonance imagery (MRI) [15].

Experiments in nanoscale magnetics are difficult to engineer with current methods making use of neuron scattering as a method of observation [9]. This is where computational science comes in.

1.1 Micromagnetic simulations

Two paradigms exist to describe the magnetic interactions on nanoscale; atomistic and continuous (section 2). The atomistic equations apply to individual atoms and can be directly implemented in numerical models, but not solved analytically. For this, the continuous equations, limiting approximations of the atomistic equations over small volumes, are more apt. Regardless, the continuous equations must be discretised to be implemented numerically. If the correct discretisation stencils are used, the equations for each interaction expressed in both paradigms are equivalent up to rescaling (appendix A) and differ in units by division by unit volume. Thus the disparate equations can be reconciled in a numerical setting.

Now we pose our model. Let our system be a centrosymmetic lattice of dimensions N_x by N_y by N_z. The lattices nodes are either individual atoms (atomistic) or small magnetised volumes (continuous). We assume each node possesses a three-dimensional vector called the spin (atomistic) or magnetisation (continuous) whose orientation and magnitude represent

[1] Racetrack memory was originally intended to be made of nanoscale domain walls but less current would be required to move skyrmions than domain walls [12].

the direction and strength of that atom's (atomistic) or volume's (continuous) magnetism. Each node has an associated energy (atomistic) or energy density (continuous) consisting of intrinsic and local contributions that depend on the properties of the magnetic material and external magnetic fields it may be subject to.

We consider four magnetic interactions; Zeeman, uniaxial anisotropy, exchange, and DMI (section 2). Zeeman and anisotropy are intrinsic interactions whereas exchange and DMI are local. Each node in a centrosymmetric lattice has up to six nearest neighbours (figure 1; left) and local interactions concern themselves with these neighbour nodes while intrinsic interactions limit themselves to only their own node.

In general, such models are referred to as spin models.

1.2 Finding energy minima

In accordance with the laws of thermodynamics, a given system will relax to a lower state of energy which corresponds to finding a local or global minimum of the 'energy landscape'. This is the state in which a physical system would find itself naturally. We achieve this computationally by updating the spins or magnetisations in our model via some deterministic or stochastic mechanism that incorporates the interaction equations, in other words, by running a simulation.

The time evolution of ferromagnetic materials can be described through the Landau-Lifshitz-Gilbert (LLG) [16, 17] differential equation (and its variants). Analytical solutions to this differential equation are only tractable for simple problems hence the need for numerical methods like gradient descent to find solutions. Other methods do not describe the dynamics of the system over time steps (so there is no notion of how long a system has been evolving) but over iteration steps which have a purely computational significance. Mean-field algorithms [18] make iteration-based approximations by having spins (or magnetisations) interact with averages of the ensemble of spins (or magnetisations) rather than individual ones. Stochastic approaches that use a Monte Carlo algorithm such as the Metropolis Monte Carlo (MMC) [19] algorithm rely on Markov chain theory to reach a minimum. This is the method that we take forward.

1.3 Metropolis Monte Carlo algorithm

The MMC uses a random mechanism to select a single or multiple nodes in the lattice of spins[2] and to perturb them. The perturbations, called proposals, are then accepted based on the change in energy that would be incurred due to that perturbation. It is common to use the Boltzmann factor as the probability of transition which is the quotient of the probabilities of the system being in the current and proposed states. Concretly, we have

$$p_{A\to B} = e^{-\frac{E_B - E_A}{k_B T}} = e^{-\frac{\Delta E}{k_B T}}$$

where $p_{A\to B}$ is the probability of transitioning from state A to state B, E_i is the energy of state i, k_B is the Boltzmann constant, and T is the temperature in Kelvin.

Naturally, MMC methods may experience more iterations than their PDE-solving counterparts but each iteration can be considerably faster. These updates occur until the system stops evolving or the iterations time out.

1.4 Advantages of MMC algorithm

The advantages of MMC algorithms are threefold; they are naturally parallelisable, they can simulate at elevated (non-zero) temperature, and they follow a simple procedure.

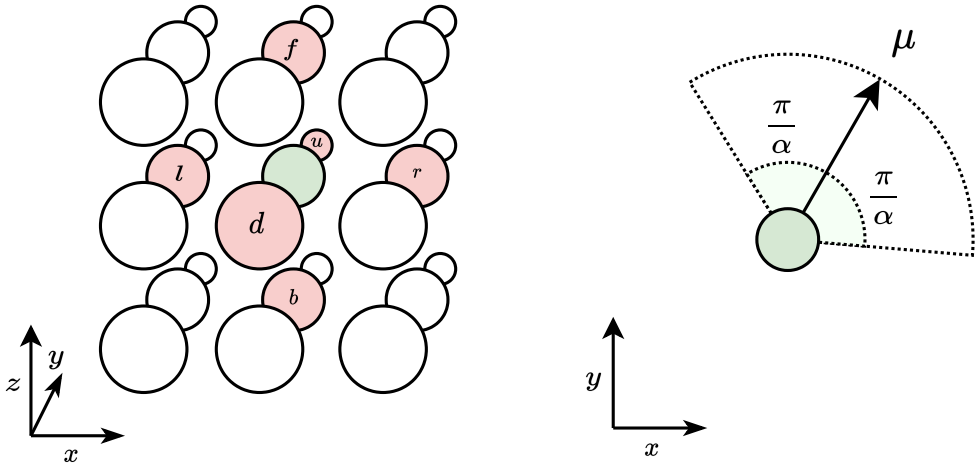
Elevated temperature corresponds to vibration in the system, meaning that with some non-zero probability, less favourable, more energetic system states can be accepted. Being able to simulate at elevated temperature makes the algorithm more applicable since it is an unrealistic assumption that a magnetic material is maintained at absolute zero.

Monte Carlo methods are naturally parallelisable since they involve independent random proposals. Many have designed and implemented micromagnetic drivers that benefit from the speed-up of parallelisation on CPU [20] or GPU [21, 22, 23, 24] but none are currently in Ubermag (section 3).

The algorithm is conceptually simple so can be summarised in a few lines (section 4) which makes easy to implement, maintain, and modify.

[2] 'Spin' and 'atom' can be understood as 'magnetisation' and 'cell' in the continuous context.

Figure 1: Left: In a centrosymmetric lattice, the green (central) atom has six nearest neighbours shown in red (with letter labels). In the code, these are called *right* (positive x), *left* (negative x), *up* (positive y), *down* (negative y), *front* (positive z), and *back* (negative z). For faces, edges, or corners of the lattice domain, not all of these neighbours exist. **Right:** Diagram showing the search cone in two-dimensions, where $0 \leq \alpha \leq 1$. In three dimensions, proposals are generated uniformly from the spherical sector centred at spin $\boldsymbol{\mu}$ (or in continuous lexicon, magnetisation $\boldsymbol{m}$) with angle of revolution $\frac{\pi}{\alpha}$ such that $\alpha = 1$ gives the search cone as the whole unit sphere and $\alpha = 0$ gives no search.



2 Magnetic energies

In our simulator, we consider four energies corresponding to four interactions: Zeeman, uniaxial anisotropy, exchange, and DMI (with T crystalographic class). Each can be considered in the atomistic or continuous formulation of equations which are equivalent up to rescaling of constants under certain discretisations (appendix A).

This appendix explains the tendency of each interaction and gives the atomistic and continuous versions of the equations. Atomistic equations detail what happens on an atomic level and give the values of energies due to the interaction of individual dipoles; continuous equations are derived by taking the limits of these equations over small volumes. While the atomistic equations feature spins $\boldsymbol{\mu}$, continuous equations feature the magnetisation $\boldsymbol{m} = \frac{\sum_{i \in V} \boldsymbol{\mu}_i}{V}$ for some discretisation volume V. In both cases, $\boldsymbol{\mu}$ or $\boldsymbol{m}$ are unit vector fields whose units differ by division of unit volume and their magnitude is given by the magnetic saturation M_s.

If N_x, N_y and N_z are the nodes in each direction of our magnetic lattice, then in the atomistic sense the nodes correspond to atoms and in the continuous sense they correspond

to discretisation volumes. We denote a generic node's moment or magnetisation by $\boldsymbol{\mu}_{i,j,k}$ or $\boldsymbol{m}_{i,j,k}$ where $1 \leq i \leq N_x$, $1 \leq j \leq N_y$ and $1 \leq k \leq N_z$. We denote the energy of an atom by $E_{i,j,k}$ and the energy densities by w. To find the total energy of a system due to an interaction, we sum over all nodes in the atomistic paradigm and integrate with respect to volume in continuous. We denote the six nearest neighbours to (i, j, k) in the centrosymmetric lattice structure (figure 1; left) by the set $N_{i,j,k}$. We denote the unit vector between atom or cell (i, j, k) and a nearest neigbour $n \in N_{i,j,k}$ by $\hat{\boldsymbol{r}}_{(i,j,k),n}$.

2.1 Exchange

The exchange energy is minimised when all moments or magnetisations are aligned in parallel to each other if $J > 0$, known as a ferromagnetic material, or antiparallel with respect to each other when $J < 0$, known as an antiferromagnetic material. The atomistic equation for the energy of one atom is

$$E_{ex_{i,j,k}} = -J \sum_{n \in N_{i,j,k}} \boldsymbol{\mu}_{i,j,k} \cdot \boldsymbol{\mu}_n$$

which in the limit $V \to 0$ becomes

$$w_{ex} = A\boldsymbol{m} \cdot \nabla^2 \boldsymbol{m}$$

There tends to be a zero-gradient Neumann boundary condition on $\boldsymbol{m}$. If the continuous equation is discretised, this is equivalent to copying the edge values of $\boldsymbol{m}$ into ghost nodes.

2.2 Dzyaloshinskii-Moriya Interaction (DMI)

DMI is found in materials with broken inversion symmetry. It is a local interaction and the energy is minimised when neighbours are mutually at right angles with respect to each other and to the vector between them. For a row of particles (nano-wire), this means having their moments twisted around their axis with the direction of the twist dependant on whether the constant $D > 0$ or $D < 0$. On the edge of magnetic domains, combined with exchange, this causes so-called edge-tilting. The atomistic equation for energy of one

atom is

$$E_{DMI_{i,j,k}} = -D \sum_{n \in N_{i,j,k}} \hat{\boldsymbol{r}}_{(i,j,k),n} \cdot (\boldsymbol{\mu}_{i,j,k} \times \boldsymbol{\mu}_n)$$

which in limit $V \to 0$ becomes

$$w_{DMI} = -D\boldsymbol{m} \cdot \nabla \times \boldsymbol{m}$$

Usually we impose a zero Dirichlet boundary condition which corresponds to padding $\boldsymbol{m}$ with ghost cells of value zero under discretisations.

2.3 Zeemam

The Zeeman interaction is intrinsic and involves the interplay of a magnetic material with an external magnetic field $\boldsymbol{H}$. The energy is minimised if $\boldsymbol{\mu}$ or $\boldsymbol{m}$ are aligned with $\boldsymbol{H}$. In the atomistic formulation, the energy of one atom is given by

$$E_{Z_{i,j,k}} = -\mu_0 \boldsymbol{H} \cdot \boldsymbol{\mu}_{i,j,k}$$

for $\mu_0 > 0$ the magnetic permeability constant. In the limit $V \to 0$, this becomes

$$w_Z = -\mu_0 M_s \boldsymbol{m} \cdot \boldsymbol{H}.$$

2.4 Uniaxial anisotropy

The anisotropy interaction is also intrinsic. Its energy is minimised when $\boldsymbol{\mu}$ or $\boldsymbol{m}$ are parallel or anti-parallel with $\hat{\boldsymbol{u}}$ when $K > 0$ known as easy-axis, or perpendicular to it when $K < 0$ known as easy-plane. The energy of one atom is given by

$$E_{a_{i,j,k}} = -K(\boldsymbol{\mu}_{i,j,k} \cdot \hat{\boldsymbol{u}})^2$$

which in the limit $V \to 0$ becomes

$$w_a = -K(\boldsymbol{m} \cdot \hat{\boldsymbol{u}})^2$$

with the constant K unchanged. Since $\boldsymbol{m}$ and $\hat{\boldsymbol{u}}$ are both unit vectors, this can also be rewritten as

$$-K(\boldsymbol{m}\cdot\hat{\boldsymbol{u}})^2 = -K\left(1-(\boldsymbol{m}\times\hat{\boldsymbol{u}})^2\right) = K(\boldsymbol{m}\times\hat{\boldsymbol{u}})^2 - K.$$

Since we can adjust the reference level, this can be expressed as $K(\boldsymbol{m}\times\hat{\boldsymbol{u}})^2$.

3 Context and aim of project

In this section, we will discuss the importance of exposing computational scientific workflows to general purpose programming tools. Then we will explain the motivation for our project, in that the current micromagnetics umbrella API, Ubermag, does not encompass an MMC driver.

3.1 Scientific workflows in Jupyter Notebooks

Former and contemporary published computational research makes use of a wide range of codebases and domain-specific languages. Sometimes these contain unwieldy legacy code that is inaccessible to scientists that have not developed it. Someone wishing to enhance their own research with an other's code may be required to spend hours learning how to use it before benefiting from it, let alone being able to simulate the same results for lack of knowing the exact procedure.

The importance of being able to reproduce one's own scientific workflows via general purpose programming tools has therefore become a focus in the community [25]. In particular, exposing workflows to Jupyter Notebooks makes research transparent, reproducible, and reusable. Jupyter has become the preferred choice of common programming tool since Python is extremely versatile, well supported, and a go-to language used by all scientists no matter their computational background. The possibility to add GUI with IPyWidgets contributes further to its suitability because the workflows presented through it can be streamlined or embellished as required. If researchers packaged their workflows, files, and configurations of their experiments in a way that could be exposed to Jupyter Notebooks, we would bypass the common challenges of not having access to the data, not knowing about the hardware and software configurations, and not knowing the exact experimental procedure.

3.2 Integration of MMC driver into Ubermag

There exist several open source micromagnetic simulation packages. OOMMF [26] and mumax^3 [27] are popular finite-difference libraries while NMAG [28] uses finite element algorithms. Fidimag [29] has options for both.

All of the aforementioned codebases have been combined into a Jupyter-friendly, Python-based API called Ubermag [30, 31]. Ubermag's goal is to put the ideals expressed in subsection 3.1 into practice by providing a consistent, logical Python-based API allowing for computational micromagnetic experiments to be readily shared among physicists. As an illustration, OOMMF has a C++ and Tcl core making it a challenge to navigate and modify for the average physicist but using Ubermag bypasses direct interaction with its code.

Despite the advantages that MMC drivers can offer (subsection 1.4), there is currently no MMC driver in the any of the libraries that Ubermag encompasses. The aim of this project is to write and test code for a CPU-parallelised MMC driver that can be accessed through the Ubermag API. Within the ethos of the discussion in subsection 3.1, any software tool that we develop for the purpose of research should be interoperable with the Jupyter ecosystem, and if it is a micromagnetics driver then it should be compatible with Ubermag.

4 Methodology

In this section, we will explain the implementation of our CPU-parallelised MMC driver using MPI. We will firstly translate the MMC driver into pseudocode, secondly introduce the checkerboard domain decomposition scheme, and finally explain how these are combined in our MPI implementation.

4.1 Coding the MMC algorithm

Let our spin model system be as described in subsection 1.1 with lattice of size $s = N_x \times N_y \times N_z$. Let the current iteration be i, the maximum number of iterations i_{max}, and the temperature T.

Before detailing the algorithm, let us discuss how the magnetisations[3] and energy densities of the system are stored in order to avoid unnecessary recalculations and ensure correct value access.

Magnetisations are stored in a lattice of dimensions $N_x \times N_y \times N_z \times 3$, and Zeeman and anisotropy energy densities in lattices of size $N_x \times N_y \times N_z$ where each element of the lattice corresponds to a magnetisation in the magnetisation lattice. These regular, 'non-offset' lattice are the diamond-type lattices in figure 2.

Since exchange and DMI interactions occur between magnetisations and are symmetric, we store these energy densities 'between' nodes in an 'offset lattice' (and they will later be double counted). There are three lattices for each of these interactions:

1. An $(N_x + 1) \times N_y \times N_z$ to represent x-direction nearest neighbour energy density terms with the first and last x-direction $1 \times N_y \times N_z$ slices being zero;

2. An $N_x \times (N_y + 1) \times N_z$ to represent y-direction nearest neighbour energy density terms with the first and last y-direction $N_x \times 1 \times N_z$ slices being zero;

3. An $N_x \times N_y \times (N_z + 1)$ to represent z-direction nearest neighbour energy density terms with the first and last z-direction $N_x \times N_y \times 1$ slices being zero.

[3] In this section we use continuous nomenclature.

These lattices are the circle-type lattices in figure 2. A magnetisation's energy due to exchange or DMI is split into six parts, one part due to each nearest neighbour. These must be summed to find the total energy due to the interactions for that cell, with two terms coming from each of the energy density lattices. Likewise, to calculate a perturbation magnetisation's energy, each of these six terms must be recalculated. In practice it may help to think of these fourteen summands (one Zeeman, one anisotropy, six exchange, six DMI) as different energy terms, some of which are stored adjacently in the energy density lattices, as this is how they are treated in the code.

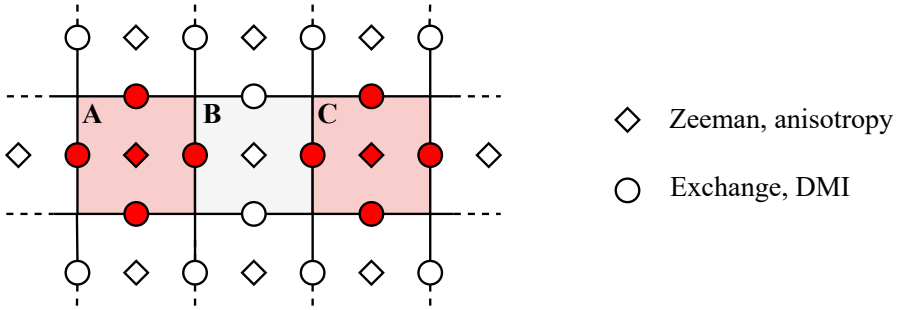
With this in mind, the MMC algorithm should be implemented as follows:

Algorithm A: MMC

1. Initialise the lattice of magnetisations, calculate the corresponding energy density lattices for each interaction, and calculate the total energy E by summing over the energy density lattices (doubling counting the exchange and DMI lattices).
2. Set $i = 0$. While $i < i_{max}$:
   (a) Select a cell randomly uniformly across the lattice.
   (b) Generate a random proposal for the cell's magnetisation based on its current magnetisation and a search-cone parameter α (see subsection 4.1.1).
   (c) Calculate the energy change ΔE due to this perturbation by summing the current magnetisation's energy and calculating the proposed magnetisation's energy (which should then be stored to prevent recalculation in the case that the perturbation is accepted) and finding the difference.
   (d) Accept or reject the perturbation with probability $e^{-\frac{\Delta E}{k_b T}}$; if accepted redistribute the proposed magnetisation and the fourteen energy density terms into the corresponding arrays and set $E \to E + \Delta E$; else nothing changes.

As for initial magnetisations, it is common to either randomise them (corresponding to a demagnetised magnetic material obtained through heating past the Curie temperature) or to have uniform magnetisations (corresponding to a material that has been exposed to a strong external field). Our code has options to initialise the magnetisations randomly, or uniformly in the x-, y-, or z-directions.

Figure 2: Zeeman and anisotropy energy densities are proper to each cell so are stored in a lattice of the same first three dimensions as the magnetisation lattice, marked as diamonds. The values of exchange and DMI energy densities between neighbouring nodes are stored in offset lattices, marked as circles (in two dimensions there are four summands to each interaction).



4.1.1 Varying search parameter α

In some MMC drivers [21] the size of the possible 'search cone' (figure 1; right) is based on the local acceptance rate and is essential for good convergence of the algorithm [21]. Preliminary testing of our algorithm proves that for larger systems involving more than two interactions, for example in the case of reproducing a skyrmion, it may help to decay the parameter α exponentially.

A Equivalence between continuous and atomistic equations

For specific discretisations of the continuous equations in appendix 2, there is an equivalence between atomistic and continuous equations. If this equivalence is derived and employed in the simulator, this can be leveraged to give the user the choice of whether they want their parameters to be interpreted in the continuous or atomistic paradigm.

For the Zeeman interaction, we must simply multiply by M_s (appendix 2.3). For uniaxial anisotropy, there is no change in constants so the same equations (appendix 2.4) can be applied directly up to change in reference level[4]. Some thought need to be be put in to how to rescale the continuous constants A and D to obtain atomistic constants (or rather, vectors) $\boldsymbol{J}$ and $\boldsymbol{d}$ for the exchange interaction and DMI under discretisation.

The discretisation comes down to which stencil is used. We use central difference with stencils that do not extend beyond the nearest neighbours (otherwise they will not be compatible with the checkerboard algorithm; OOMMF also uses the same stencils). We will derive the conversions for the three-dimensional case below.

[4]For OOMMF's calculator to which we compare our values, there is a reference level K added to the expression if $K > 0$

A.1 Exchange

Using a $\frac{1}{\Delta^2}(1,-2,1)$ central difference stencil for the Laplace operator, we can derive $w_{E_{i,j,k}} = [-A\boldsymbol{m}\cdot\nabla^2\boldsymbol{m}]_{i,j,k}$ by first breaking up the components of the Laplacian

$$
\begin{aligned}
\left[\nabla^2\boldsymbol{m}\right]^x_{i,j,k} &\approx \frac{\boldsymbol{m}^x_{i+1,j,k} - 2\boldsymbol{m}^x_{i,j,k} + \boldsymbol{m}^x_{i-1,j,k}}{\Delta x^2} + \frac{\boldsymbol{m}^x_{i,j+1,k} - 2\boldsymbol{m}^x_{i,j,k} + \boldsymbol{m}^x_{i,j-1,k}}{\Delta y^2} + \frac{\boldsymbol{m}^x_{i,j,k+1} - 2\boldsymbol{m}^x_{i,j,k} + \boldsymbol{m}^x_{i,j,k-1}}{\Delta z^2} \\
\left[\nabla^2\boldsymbol{m}\right]^y_{i,j,k} &\approx \frac{\boldsymbol{m}^y_{i+1,j,k} - 2\boldsymbol{m}^y_{i,j,k} + \boldsymbol{m}^y_{i-1,j,k}}{\Delta x^2} + \frac{\boldsymbol{m}^y_{i,j+1,k} - 2\boldsymbol{m}^y_{i,j,k} + \boldsymbol{m}^y_{i,j-1,k}}{\Delta y^2} + \frac{\boldsymbol{m}^y_{i,j,k+1} - 2\boldsymbol{m}^y_{i,j,k} + \boldsymbol{m}^y_{i,j,k-1}}{\Delta z^2} \\
\left[\nabla^2\boldsymbol{m}\right]^z_{i,j,k} &\approx \frac{\boldsymbol{m}^z_{i+1,j,k} - 2\boldsymbol{m}^z_{i,j,k} + \boldsymbol{m}^z_{i-1,j,k}}{\Delta x^2} + \frac{\boldsymbol{m}^z_{i,j+1,k} - 2\boldsymbol{m}^z_{i,j,k} + \boldsymbol{m}^z_{i,j-1,k}}{\Delta y^2} + \frac{\boldsymbol{m}^z_{i,j,k+1} - 2\boldsymbol{m}^z_{i,j,k} + \boldsymbol{m}^z_{i,j,k-1}}{\Delta z^2}.
\end{aligned}
$$

After taking the dot product with $\boldsymbol{m}_{i,j,k}$, we have

$$
\begin{aligned}
-\frac{1}{A}w_{E_{i,j,k}} \approx & \frac{\boldsymbol{m}^x_{i,j,k}\boldsymbol{m}^x_{i+1,j,k}}{\Delta x^2} + \frac{\boldsymbol{m}^x_{i,j,k}\boldsymbol{m}^z_{i-1,j,k}}{\Delta x^2} + \frac{\boldsymbol{m}^x_{i,j,k}\boldsymbol{m}^x_{i,j+1,k}}{\Delta y^2} + \frac{\boldsymbol{m}^x_{i,j,k}\boldsymbol{m}^x_{i,j-1,k}}{\Delta y^2} + \frac{\boldsymbol{m}^x_{i,j,k}\boldsymbol{m}^x_{i,j,k+1}}{\Delta z^2} + \frac{\boldsymbol{m}^x_{i,j,k}\boldsymbol{m}^x_{i,j,k-1}}{\Delta z^2} \\
& + \frac{\boldsymbol{m}^y_{i,j,k}\boldsymbol{m}^y_{i+1,j,k}}{\Delta x^2} + \frac{\boldsymbol{m}^y_{i,j,k}\boldsymbol{m}^y_{i-1,j,k}}{\Delta x^2} + \frac{\boldsymbol{m}^y_{i,j,k}\boldsymbol{m}^y_{i,j+1,k}}{\Delta y^2} + \frac{\boldsymbol{m}^y_{i,j,k}\boldsymbol{m}^y_{i,j-1,k}}{\Delta y^2} + \frac{\boldsymbol{m}^y_{i,j,k}\boldsymbol{m}^y_{i,j,k+1}}{\Delta z^2} + \frac{\boldsymbol{m}^y_{i,j,k}\boldsymbol{m}^y_{i,j,k-1}}{\Delta z^2} \\
& + \frac{\boldsymbol{m}^z_{i,j,k}\boldsymbol{m}^z_{i+1,j,k}}{\Delta x^2} + \frac{\boldsymbol{m}^z_{i,j,k}\boldsymbol{m}^z_{i-1,j,k}}{\Delta x^2} + \frac{\boldsymbol{m}^z_{i,j,k}\boldsymbol{m}^z_{i,j+1,k}}{\Delta y^2} + \frac{\boldsymbol{m}^z_{i,j,k}\boldsymbol{m}^z_{i,j-1,k}}{\Delta y^2} + \frac{\boldsymbol{m}^z_{i,j,k}\boldsymbol{m}^z_{i,j,k+1}}{\Delta z^2} + \frac{\boldsymbol{m}^z_{i,j,k}\boldsymbol{m}^z_{i,j,k-1}}{\Delta z^2} \\
& - \frac{2(\boldsymbol{m}^x_{i,j,k})^2}{\Delta x^2} - \frac{2(\boldsymbol{m}^x_{i,j,k})^2}{\Delta y^2} - \frac{2(\boldsymbol{m}^x_{i,j,k})^2}{\Delta z^2} \\
& - \frac{2(\boldsymbol{m}^y_{i,j,k})^2}{\Delta x^2} - \frac{2(\boldsymbol{m}^y_{i,j,k})^2}{\Delta y^2} - \frac{2(\boldsymbol{m}^y_{i,j,k})^2}{\Delta z^2} \\
& - \frac{2(\boldsymbol{m}^z_{i,j,k})^2}{\Delta x^2} - \frac{2(\boldsymbol{m}^z_{i,j,k})^2}{\Delta y^2} - \frac{2(\boldsymbol{m}^z_{i,j,k})^2}{\Delta z^2} \\
= & \left(\frac{\boldsymbol{m}^x_{i,j,k}\boldsymbol{m}^x_{i+1,j,k}}{\Delta x^2} + \frac{\boldsymbol{m}^y_{i,j,k}\boldsymbol{m}^y_{i+1,j,k}}{\Delta x^2} + \frac{\boldsymbol{m}^z_{i,j,k}\boldsymbol{m}^z_{i+1,j,k}}{\Delta x^2}\right) + \left(\frac{\boldsymbol{m}^x_{i,j,k}\boldsymbol{m}^x_{i-1,j,k}}{\Delta x^2} + \frac{\boldsymbol{m}^y_{i,j,k}\boldsymbol{m}^y_{i-1,j,k}}{\Delta x^2} + \frac{\boldsymbol{m}^z_{i,j,k}\boldsymbol{m}^z_{i-1,j,k}}{\Delta x^2}\right) \\
& + \left(\frac{\boldsymbol{m}^x_{i,j,k}\boldsymbol{m}^x_{i,j+1,k}}{\Delta y^2} + \frac{\boldsymbol{m}^y_{i,j,k}\boldsymbol{m}^y_{i,j+1,k}}{\Delta y^2} + \frac{\boldsymbol{m}^z_{i,j,k}\boldsymbol{m}^z_{i,j+1,k}}{\Delta y^2}\right) + \left(\frac{\boldsymbol{m}^x_{i,j,k}\boldsymbol{m}^x_{i,j-1,k}}{\Delta y^2} + \frac{\boldsymbol{m}^y_{i,j,k}\boldsymbol{m}^y_{i,j-1,k}}{\Delta y^2} + \frac{\boldsymbol{m}^z_{i,j,k}\boldsymbol{m}^z_{i,j-1,k}}{\Delta y^2}\right) \\
& + \left(\frac{\boldsymbol{m}^x_{i,j,k}\boldsymbol{m}^x_{i,j,k+1}}{\Delta z^2} + \frac{\boldsymbol{m}^y_{i,j,k}\boldsymbol{m}^y_{i,j,k+1}}{\Delta z^2} + \frac{\boldsymbol{m}^z_{i,j,k}\boldsymbol{m}^z_{i+1,j,k}}{\Delta z^2}\right) + \left(\frac{\boldsymbol{m}^x_{i,j,k}\boldsymbol{m}^x_{i,j,k-1}}{\Delta z^2} + \frac{\boldsymbol{m}^y_{i,j,k}\boldsymbol{m}^y_{i,j,k-1}}{\Delta z^2} + \frac{\boldsymbol{m}^z_{i,j,k}\boldsymbol{m}^z_{i,j,k-1}}{\Delta z^2}\right) \\
& - 2\left(\frac{1}{\Delta x^2} + \frac{1}{\Delta y^2} + \frac{1}{\Delta z^2}\right)
\end{aligned}
$$

where the final line comes from the fact that $\boldsymbol{m}_{i,j,k}$ are unit vectors. After collecting terms, this simplifies to

$$\begin{aligned}-\frac{1}{A}w_{E_{i,j,k}} \approx \boldsymbol{m}_{i,j,k}\cdot\boldsymbol{m}_{i+1,j,k} - \frac{1}{\Delta x^2} + \boldsymbol{m}_{i,j,k}\cdot\boldsymbol{m}_{i-1,j,k} - \frac{1}{\Delta x^2} \\ + \boldsymbol{m}_{i,j,k}\cdot\boldsymbol{m}_{i,j+1,k} - \frac{1}{\Delta y^2} + \boldsymbol{m}_{i,j,k}\cdot\boldsymbol{m}_{i,j-1,k} - \frac{1}{\Delta y^2} \\ + \boldsymbol{m}_{i,j,k}\cdot\boldsymbol{m}_{i,j,k+1} - \frac{1}{\Delta z^2} + \boldsymbol{m}_{i,j,k}\cdot\boldsymbol{m}_{i,j,k-1} - \frac{1}{\Delta z^2}.\end{aligned}$$

So we have

$$w_{E_{i,j,k}} \approx |\boldsymbol{J}\cdot\hat{\boldsymbol{r}}_{(i,j,k),n}| - |\boldsymbol{J}\cdot\hat{\boldsymbol{r}}_{(i,j,k),n}|\sum_{n\in N}(\boldsymbol{m}_{i,j,k}\cdot\boldsymbol{m}_n)$$

where $\boldsymbol{J} = \frac{A}{\Delta x^2}\hat{\boldsymbol{x}} + \frac{A}{\Delta y^2}\hat{\boldsymbol{y}} + \frac{A}{\Delta z^2}\hat{\boldsymbol{z}}$, that is, the value of J depends on the direction of the nearest neighbour (but not the sign of $\hat{\boldsymbol{r}}$). At this (zero) reference level, the zero-gradient boundary condition on the Laplacian is equivalent to

$$\boldsymbol{J} - \boldsymbol{J}\cdot(\boldsymbol{m}_{i,j,k}\cdot\boldsymbol{m}_{i,j,k}) = \boldsymbol{J} - \boldsymbol{J} = 0$$

since $\boldsymbol{m}_{i,j,k}$ are unit vectors. In other words, we consider there to be a padding of zero-vectors beyond our lattice.

A.2 Dzyaloshinskii-Moriya Interaction (DMI)

Using a $\frac{1}{2\Delta}(1,0,-1)$ central difference stencil for the curl operator, we break $w_{DMI_{i,j,k}} = [-D\boldsymbol{m}\cdot(\nabla\times\boldsymbol{m})]_{i,j,k}$ into parts

$$\begin{aligned}[\nabla\times\boldsymbol{m}]^x_{i,j,k} &\approx \frac{\boldsymbol{m}^z_{i,j+1,k} - \boldsymbol{m}^z_{i,j-1,k}}{2\Delta y} - \frac{\boldsymbol{m}^y_{i,j,k+1} - \boldsymbol{m}^y_{i,j,k-1}}{2\Delta z} \\ [\nabla\times\boldsymbol{m}]^y_{i,j,k} &\approx \frac{\boldsymbol{m}^x_{i,j,k+1} - \boldsymbol{m}^x_{i,j,k-1}}{2\Delta z} - \frac{\boldsymbol{m}^z_{i+1,j,k} - \boldsymbol{m}^z_{i-1,j,k}}{2\Delta x} \\ [\nabla\times\boldsymbol{m}]^z_{i,j,k} &\approx \frac{\boldsymbol{m}^y_{i+1,j,k} - \boldsymbol{m}^y_{i-1,j,k}}{2\Delta x} - \frac{\boldsymbol{m}^x_{i,j+1,k} - \boldsymbol{m}^x_{i,j-1,k}}{2\Delta y}.\end{aligned}$$

Combining these terms after dotting, we have

$$
\begin{aligned}
-\frac{1}{D} w_{DMI_{i,j,k}} \approx & \frac{\boldsymbol{m}^x_{i,j,k}\boldsymbol{m}^z_{i,j+1,k}}{2\Delta y} - \frac{\boldsymbol{m}^x_{i,j,k}\boldsymbol{m}^z_{i,j-1,k}}{2\Delta y} - \frac{\boldsymbol{m}^x_{i,j,k}\boldsymbol{m}^y_{i,j,k+1}}{2\Delta z} + \frac{\boldsymbol{m}^x_{i,j,k}\boldsymbol{m}^y_{i,j,k-1}}{2\Delta z} \\
& + \frac{\boldsymbol{m}^y_{i,j,k}\boldsymbol{m}^x_{i,j,k+1}}{2\Delta z} - \frac{\boldsymbol{m}^y_{i,j,k}\boldsymbol{m}^x_{i,j,k-1}}{2\Delta z} - \frac{\boldsymbol{m}^y_{i,j,k}\boldsymbol{m}^z_{i+1,j,k}}{2\Delta x} + \frac{\boldsymbol{m}^y_{i,j,k}\boldsymbol{m}^z_{i-1,j,k-1}}{2\Delta x} \\
& + \frac{\boldsymbol{m}^z_{i,j,k}\boldsymbol{m}^y_{i+1,j,k}}{2\Delta x} - \frac{\boldsymbol{m}^z_{i,j,k}\boldsymbol{m}^y_{i-1,j,k}}{2\Delta x} - \frac{\boldsymbol{m}^z_{i,j,k}\boldsymbol{m}^x_{i,j+1,k}}{2\Delta y} + \frac{\boldsymbol{m}^z_{i,j,k}\boldsymbol{m}^x_{i,j-1,k}}{2\Delta y} \\
= & -\left(\frac{\boldsymbol{m}^y_{i,j,k}\boldsymbol{m}^z_{i+1,j,k}}{2\Delta x} - \frac{\boldsymbol{m}^z_{i,j,k}\boldsymbol{m}^y_{i+1,j,k}}{2\Delta x}\right) + \left(\frac{\boldsymbol{m}^z_{i,j,k}\boldsymbol{m}^y_{i-1,j,k}}{2\Delta x} - \frac{\boldsymbol{m}^z_{i,j,k}\boldsymbol{m}^y_{i-1,j,k}}{2\Delta x}\right) \\
& - \left(\frac{\boldsymbol{m}^x_{i,j,k}\boldsymbol{m}^z_{i,j+1,k}}{2\Delta y} - \frac{\boldsymbol{m}^z_{i,j,k}\boldsymbol{m}^x_{i,j+1,k}}{2\Delta y}\right) + \left(\frac{\boldsymbol{m}^x_{i,j,k}\boldsymbol{m}^z_{i,j-1,k}}{2\Delta y} - \frac{\boldsymbol{m}^z_{i,j,k}\boldsymbol{m}^x_{i,j-1,k}}{2\Delta y}\right) \\
& - \left(\frac{\boldsymbol{m}^x_{i,j,k}\boldsymbol{m}^y_{i,j,k+1}}{2\Delta z} - \frac{\boldsymbol{m}^y_{i,j,k}\boldsymbol{m}^x_{i,j,k+1}}{2\Delta z}\right) + \left(\frac{\boldsymbol{m}^x_{i,j,k}\boldsymbol{m}^y_{i,j,k-1}}{2\Delta z} - \frac{\boldsymbol{m}^y_{i,j,k}\boldsymbol{m}^x_{i,j,k-1}}{2\Delta z}\right) \\
= & \left(\frac{\boldsymbol{m}_{i,j,k} \times \boldsymbol{m}_{i+1,j,k}}{2\Delta x}\right) \cdot -\hat{\boldsymbol{x}} + \left(\frac{\boldsymbol{m}_{i,j,k} \times \boldsymbol{m}_{i-1,j,k}}{2\Delta x}\right) \cdot \hat{\boldsymbol{x}} \\
& + \left(\frac{\boldsymbol{m}_{i,j,k} \times \boldsymbol{m}_{i,j+1,k}}{2\Delta y}\right) \cdot -\hat{\boldsymbol{y}} + \left(\frac{\boldsymbol{m}_{i,j,k} \times \boldsymbol{m}_{i,j-1,k}}{2\Delta y}\right) \cdot \hat{\boldsymbol{y}} \\
& + \left(\frac{\boldsymbol{m}_{i,j,k} \times \boldsymbol{m}_{i,j,k+1}}{2\Delta z}\right) \cdot -\hat{\boldsymbol{z}} + \left(\frac{\boldsymbol{m}_{i,j,k} \times \boldsymbol{m}_{i,j,k-1}}{2\Delta z}\right) \cdot \hat{\boldsymbol{z}}.
\end{aligned}
$$

So we have

$$
w_{DMI_{i,j,k}} \approx \left(\boldsymbol{d} \cdot \hat{\boldsymbol{r}}_{(i,j,k),n}\right) \cdot \sum_{n \in N} \boldsymbol{m}_{i,j,k} \times \boldsymbol{m}_n
$$

where $\boldsymbol{d} = \frac{D}{2\Delta x}\hat{\boldsymbol{x}} + \frac{D}{2\Delta y}\hat{\boldsymbol{y}} + \frac{D}{2\Delta z}\hat{\boldsymbol{z}}$, that is, the value d depends on the direction of the nearest neighbour.

References

[1] T.H.R. Skyrme. A unified field theory of mesons and baryons. *Nuclear Physics*, 31:556–569, March 1962.

[2] A N Bogdanov and D A Yablonskii. Thermodynamicallystable "vortices" in magneticallyordered crystals.The mixed state of magnets. *Zh. Eksp. Teor. Fiz.*, (95):178–182, 1989.

[3] Juan C. Criado, Sebastian Schenk, Michael Spannowsky, Peter D. Hatton, and L. A. Turnbull. Simulating anti-skyrmions on a lattice. *Scientific Reports*, 12(1):19179, November 2022.

[4] B D Terris and T Thomson. Nanofabricated and self-assembled magnetic structures as data storage media. *Journal of Physics D: Applied Physics*, 38(12):R199–R222, June 2005.

[5] Hongxin Yang, André Thiaville, Stanislas Rohart, Albert Fert, and Mairbek Chshiev. Anatomy of Dzyaloshinskii-Moriya Interaction at Co / Pt Interfaces. *Physical Review Letters*, 115(26):267210, December 2015.

[6] Tôru Moriya. Anisotropic Superexchange Interaction and Weak Ferromagnetism. *Physical Review*, 120(1):91–98, October 1960.

[7] Tôru Moriya. New Mechanism of Anisotropic Superexchange Interaction. *Physical Review Letters*, 4(5):228–230, March 1960.

[8] I. Dzyaloshinsky. A thermodynamic theory of "weak" ferromagnetism of antiferromagnetics. *Journal of Physics and Chemistry of Solids*, 4(4):241–255, January 1958.

[9] S. Mühlbauer, B. Binz, F. Jonietz, C. Pfleiderer, A. Rosch, A. Neubauer, R. Georgii, and P. Böni. Skyrmion Lattice in a Chiral Magnet. *Science*, 323(5916):915–919, February 2009.

[10] Xichao Zhang, Yan Zhou, Kyung Mee Song, Tae-Eon Park, Jing Xia, Motohiko Ezawa, Xiaoxi Liu, Weisheng Zhao, Guoping Zhao, and Seonghoon Woo. Skyrmion-electronics: writing, deleting, reading and processing magnetic skyrmions toward spintronic applications. *Journal of Physics: Condensed Matter*, 32(14):143001, April 2020.

[11] Luc Thomas Stuart S. P. Parkin, Masamitsu Hayashi. Magnetic Domain-Wall Racetrack Memory. *Science*, 320(5873):5, November 2008.

[12] Xichao Zhang, G. P. Zhao, Hans Fangohr, J. Ping Liu, W. X. Xia, J. Xia, and F. J. Morvan. Skyrmion-skyrmion and skyrmion-edge repulsions in skyrmion-based racetrack memory. *Scientific Reports*, 5(1):7643, January 2015.

[13] Martin Lang, Marijan Beg, Ondrej Hovorka, and Hans Fangohr. Bloch points in nanostrips. *Scientific Reports*, 13(1):6910, April 2023.

[14] Martin Lang, Swapneel Amit Pathak, Samuel J. R. Holt, Marijan Beg, and Hans Fangohr. Controlling stable Bloch points with electric currents, July 2023. arXiv:2307.10170 [cond-mat, physics:physics].

[15] Young-wook Jun, Jae-Hyun Lee, and Jinwoo Cheon. Chemical Design of Nanoparticle Probes for High-Performance Magnetic Resonance Imaging. *Angewandte Chemie International Edition*, 47(28):5122–5135, June 2008.

[16] T.L. Gilbert. Classics in Magnetics A Phenomenological Theory of Damping in Ferromagnetic Materials. *IEEE Transactions on Magnetics*, 40(6):3443–3449, November 2004.

[17] L Landau and E Lifshits. On the theory of dispersion of magnetic permeability in ferromagnetic bodies. *Phys. Zeitsch. der Sow.*, 53, 1935.

[18] Ondrej Hovorka and Timothy J. Sluckin. A computational mean-field model of interacting non-collinear classical spins, July 2020. arXiv:2007.12777 [cond-mat].

[19] Nicholas Metropolis, Arianna W. Rosenbluth, Marshall N. Rosenbluth, Augusta H. Teller, and Edward Teller. Equation of State Calculations by Fast Computing Machines. *The Journal of Chemical Physics*, 21(6):1087–1092, June 1953.

[20] Thomas Fischbacher, Matteo Franchin, Giuliano Bordignon, Andreas Knittel, and Hans Fangohr. Parallel execution and scriptability in micromagnetic simulations. *Journal of Applied Physics*, 105(7):07D527, April 2009.

[21] Serban Lepadatu. Micromagnetic Monte Carlo method with variable magnetization length based on the Landau–Lifshitz–Bloch equation for computation of large-scale thermodynamic equilibrium states. *Journal of Applied Physics*, 130(16):163902, October 2021.

[22] Tobias Preis, Peter Virnau, Wolfgang Paul, and Johannes J. Schneider. GPU accelerated Monte Carlo simulation of the 2D and 3D Ising model. *Journal of Computational Physics*, 228(12):4468–4477, July 2009.

[23] M. Weigel and T. Yavorskii. GPU accelerated Monte Carlo simulations of lattice spin models. *Physics Procedia*, 15:92–96, 2011.

[24] Benjamin Block, Peter Virnau, and Tobias Preis. Multi-GPU Accelerated Multi-Spin Monte Carlo Simulations of the 2D Ising Model. *Computer Physics Communications*, 181(9):1549–1556, September 2010. arXiv:1007.3726 [math-ph, physics:physics].

[25] Marijan Beg, Juliette Taka, Thomas Kluyver, Alexander Konovalov, Min Ragan-Kelley, Nicolas M. Thiery, and Hans Fangohr. Using Jupyter for Reproducible Scientific Workflows. *Computing in Science & Engineering*, 23(2):36–46, March 2021.

[26] M. J. Donahue, D. G. Porter. OOMMF User's Guide, Version 1.0. Technical Report NISTIR 6376, National Institute of Standards and Technology, 1999.

[27] A. Vansteenkiste and B. Van De Wiele. MuMax: A new high-performance micromagnetic simulation tool. *Journal of Magnetism and Magnetic Materials*, 323(21):2585–2591, November 2011.

[28] Fischbacher, Thomas and Franchin, Matteo and Bordignon, Giuliano and Knittel, Andreas and Fangohr, Hans. A systematic approach to multiphysics extensions of finite-element-based micromagnetic simulations: Nmag. *IEEE Transactions on Magnetics*, 43(6), 2007.

[29] Marc-Antonio Bisotti, David Cortés-Ortuño, Ryan Pepper, Weiwei Wang, Marijan Beg, Thomas Kluyver, and Hans Fangohr. Fidimag – A Finite Difference Atomistic and Micromagnetic Simulation Package. *Journal of Open Research Software*, 6(1):22, September 2018.

[30] Marijan Beg, Ryan A. Pepper, and Hans Fangohr. User interfaces for computational science: A domain specific language for OOMMF embedded in Python. *AIP Advances*, 7(5):056025, May 2017.

[31] H. Fangohr M. Beg, M. Lang. Ubermag: Towards more effective micromagnetic workflows. *IEEE Transactions on Magnetics 58, 7300205*, 2022.