COMPUTATIONAL MICROMAGNETICS WITH COMMICS

CARL-MARTIN PFEILER, MICHELE RUGGERI, BERNHARD STIFTNER, LUKAS EXL, MATTHIAS HOCHSTEGER, GINO HRKAC, JOACHIM SCHÖBERL, NORBERT J. MAUSER, AND DIRK PRAETORIUS

Abstract. We present our open-source Python module *Commics* for the study of the magnetization dynamics in ferromagnetic materials via micromagnetic simulations. It implements state-of-the-art unconditionally convergent finite element methods for the numerical integration of the Landau–Lifshitz–Gilbert equation. The implementation is based on the multiphysics finite element software *Netgen/NGSolve*. The simulation scripts are written in Python, which leads to very readable code and direct access to extensive post-processing. Together with documentation and example scripts, the code is freely available on GitLab.

1. Introduction

Micromagnetism is a continuum theory for ferromagnetic materials located between Maxwell's electromagnetism and quantum theory [32, 11, 56]. The magnetization distribution is modeled as a continuous vector field, where nonlocal magnetostatic interactions and local contributions are taken into account. Typical micromagnetic models exhibit length scales ranging from nanometers to few micrometers, which is often infeasible for atomistic spin dynamics simulations. In recent decades, micromagnetics evolved as a computational field, that nowadays represents a successful tool for numerical studies in materials science with important contemporary applications, e.g., data storage structures like hard disk drives [76, 54], random access memories [61], or nanowires [73, 50], magnetologic devices [24], soft magnetic sensor systems [34], and high performance permanent magnets [72, 23, 43].

1.1. Existing software. In recent years, advances in computer architecture, programming environments, and numerical methods led to the development of several micromagnetic codes, which aim at the numerical integration of the fundamental equation in micromagnetics, the Landau–Lifshitz–Gilbert equation (LLG), a well-accepted model for the magnetization dynamics [58, 48]. Well-known simulation packages based on finite difference discretizations [63] on Cartesian grids are *OOMMF* [39], recently extended for GPU

Date: December 17, 2018.

Acknowledgements. This research has been supported by the Vienna Science and Technology Fund (WWTF) through the project *Thermally controlled magnetization dynamics* (grant MA14-44) and by the Austrian Science Fund (FWF) through the doctoral school *Dissipation and dispersion in nonlinear PDEs* (grant W1245), the special research program *Taming complexity in partial differential systems* (grant F65), and the project *Reduced order approaches for micromagnetics* (grant P31140). Partial support from the Engineering and Physical Sciences Research Council (EPSRC) through the projects *Picosecond dynamics of magnetic exchange springs* (grant EP/P02047X/1) and *Coherent spin waves for emerging nanoscale magnonic logic architectures* (grant EP/L019876/1) is thankfully acknowledged.

usage [46] and endowed with a user-friendly Python interface [27], *MuMax*^{3} (GPU) [78], *MicroMagnum* [5] (CPU and GPU), *magnum.fd* [4], and *Fidimag* [31]. These implementations are both memory and computationally efficient owing to the uniform mesh and the particularly advantageous utilization of the fast Fourier transform for the long range field part [29, 9]. However, approaches based on the Finite Element Method (FEM) are geometrically more flexible [71] and provided in the scientific codes *MagPar* [70], *TetraMag* [53] and its successor *tetmag2*, *Nmag* [44] and its successor *Finmag* [30], as well as in commercial software like *FastMag* [35], *FEMME* [3], and *magnum.fe* [8].

1.2. Numerical analysis. For an introduction to the mathematical analysis of numerical integrators for dynamic micromagnetic simulations, we refer to the monographs [66, 18], the review articles [57, 47, 36], and the references therein. The ultimate goal is the development of unconditionally convergent integrators, i.e., numerical schemes for which (a subsequence of) the output converges towards a weak solution of LLG in the sense of [15] without requiring any restrictive CFL-type coupling condition on the temporal and spatial discretization parameters. In our work, we consider two types of methods characterized by such good theoretical properties: the *tangent plane scheme* [12, 13] and the *midpoint scheme* [25]; see Section 3 for more details.

1.3. Contributions. This work presents our novel open-source Python module *Commics* (COmputational MicroMagnetICS) to perform computational studies of the magnetization dynamics in ferromagnetic materials via micromagnetic simulations. The software is based on the multiphysics finite element software *Netgen/NGSolve* [6] and is made user-friendly by a high-level Python interface. While many existing codes popular in the physics community are fairly performance optimized, they often lack a thorough mathematical convergence analysis. In contrast to that, our implementation arises from recent results in the numerical analysis of unconditionally convergent LLG integrators. The code is freely available on GitLab [1] together with documentation and example scripts.

1.4. Outline. This work is organized as follows: We fix the notation and the precise micromagnetic setting in Section 2. In Section 3, the implemented algorithms are briefly presented. Section 4 demonstrates the exemplary use of *Netgen/NGSolve* (*NGS*) and discusses the integration of the boundary element library *BEM++* [75]. Finally, Section 5 provides Python scripts for several benchmark problems, in order to verify our module and to demonstrate its usage.

2. Micromagnetic setting

Let $\Omega \subset \mathbb{R}^3$ denote the volume occupied by a ferromagnet. In micromagnetics, the quantity of interest is the magnetization $\boldsymbol{M} : \Omega \to \mathbb{R}^3$ (in A/m). If the temperature is constant and far below the so-called Curie temperature of the material, the modulus of the magnetization is constant, i.e., it holds that $|\boldsymbol{M}| = M_\mathrm{s}$ with $M_\mathrm{s} > 0$ being the saturation magnetization (in A/m). Let $\boldsymbol{m} := \boldsymbol{M}/M_\mathrm{s}$ denote the normalized magnetization. The

magnetic state of Ω is described in terms of the magnetic Gibbs free energy (in J)

$$\begin{aligned}\mathcal{E}(\boldsymbol{m}) = A\int_\Omega |\nabla \boldsymbol{m}|^2\,\mathrm{d}\boldsymbol{x} \;-\; K\int_\Omega (\boldsymbol{a}\cdot\boldsymbol{m})^2\,\mathrm{d}\boldsymbol{x} \;+\; D\int_\Omega (\nabla\times\boldsymbol{m})\cdot\boldsymbol{m}\,\mathrm{d}\boldsymbol{x}\\ -\;\frac{\mu_0 M_\mathrm{s}}{2}\int_\Omega \boldsymbol{H}_\mathrm{s}\cdot\boldsymbol{m}\,\mathrm{d}\boldsymbol{x} \;-\; \mu_0 M_\mathrm{s}\int_\Omega \boldsymbol{H}_\mathrm{ext}\cdot\boldsymbol{m}\,\mathrm{d}\boldsymbol{x}.\end{aligned} \tag{1}$$

The energy in (1) is the sum of exchange energy, uniaxial anisotropy, bulk Dzyaloshinskii–Moriya interaction (DMI), magnetostatic energy, and Zeeman contribution, respectively. The involved material parameters and physical constants are the exchange stiffness constant $A > 0$ (in J/m), the anisotropy constant $K \geq 0$ (in J/m^3), the easy axis $\boldsymbol{a} \in \mathbb{R}^3$ with $|\boldsymbol{a}| = 1$ (dimensionless), the DMI constant $D \in \mathbb{R}$ (in J/m^2), and the vacuum permeability $\mu_0 = 4\pi \cdot 10^{-7}\,\mathrm{N/A^2}$. Moreover, $\boldsymbol{H}_\mathrm{ext}$ and $\boldsymbol{H}_\mathrm{s}$ denote the applied external field (assumed to be unaffected by variations of $\boldsymbol{m}$) and the stray field, respectively (both in A/m). The stray field (sometimes also referred to as demagnetizing or dipolar field) solves the magnetostatic Maxwell equations

$$\nabla\cdot(\boldsymbol{H}_\mathrm{s} + M_\mathrm{s}\boldsymbol{m}\chi_\Omega) = 0 \quad \text{in } \mathbb{R}^3, \tag{2a}$$

$$\nabla\times\boldsymbol{H}_\mathrm{s} = \boldsymbol{0} \quad \text{in } \mathbb{R}^3, \tag{2b}$$

where $(\boldsymbol{m}\chi_\Omega)(\boldsymbol{x}) = \boldsymbol{m}(\boldsymbol{x})$ in Ω and $(\boldsymbol{m}\chi_\Omega)(\boldsymbol{x}) = 0$ elsewhere. Stable magnetization configurations are those which minimize the magnetic Gibbs free energy (1). The dynamics towards equilibrium of the magnetization is governed by LLG

$$\partial_t\boldsymbol{m} = -\gamma_0\,\boldsymbol{m}\times\big[\boldsymbol{H}_\mathrm{eff}(\boldsymbol{m}) + \boldsymbol{T}(\boldsymbol{m})\big] + \alpha\,\boldsymbol{m}\times\partial_t\boldsymbol{m} \qquad \text{in } (0,\infty)\times\Omega, \tag{3a}$$

$$\partial_{\boldsymbol{n}}\boldsymbol{m} = -\frac{D}{2A}\,\boldsymbol{m}\times\boldsymbol{n} \qquad \text{on } (0,\infty)\times\partial\Omega, \tag{3b}$$

$$\boldsymbol{m}(0) = \boldsymbol{m}^0 \quad \text{with } |\boldsymbol{m}^0| = 1 \qquad \text{in } \Omega. \tag{3c}$$

In (3), $\gamma_0 = 2.212\cdot 10^5\,\mathrm{m/(A\,s)}$ is the gyromagnetic ratio of the electron, $\alpha \in (0,1]$ is the dimensionless Gilbert damping parameter, and $\boldsymbol{n} : \partial\Omega \to \mathbb{R}^3$ with $|\boldsymbol{n}| = 1$ denotes the outward-pointing unit normal vector to $\partial\Omega$. The effective field $\boldsymbol{H}_\mathrm{eff}(\boldsymbol{m})$ is related to the functional derivative of the energy with respect to the magnetization and takes the form

$$\begin{aligned}\boldsymbol{H}_\mathrm{eff}(\boldsymbol{m}) :&= -\frac{1}{\mu_0 M_\mathrm{s}}\frac{\delta\mathcal{E}(\boldsymbol{m})}{\delta\boldsymbol{m}}\\ &= \frac{2A}{\mu_0 M_\mathrm{s}}\,\Delta\boldsymbol{m} + \frac{2K}{\mu_0 M_\mathrm{s}}\,(\boldsymbol{a}\cdot\boldsymbol{m})\boldsymbol{a} - \frac{2D}{\mu_0 M_\mathrm{s}}\,\nabla\times\boldsymbol{m} + \boldsymbol{H}_\mathrm{s} + \boldsymbol{H}_\mathrm{ext}.\end{aligned}$$

Finally, the term $\boldsymbol{T}(\boldsymbol{m})$ collects all nonenergetic torque terms, which arise, e.g., when an electric current flows in a conducting ferromagnet. For instance, the Oersted field $\boldsymbol{T}(\boldsymbol{m}) = \boldsymbol{H}_\mathrm{c}$ (in A/m) is described by the magnetostatic Maxwell equations

$$\nabla\cdot\boldsymbol{H}_\mathrm{c} = 0 \qquad \text{in } \mathbb{R}^3, \tag{4a}$$

$$\nabla\times\boldsymbol{H}_\mathrm{c} = \boldsymbol{J}_\mathrm{e}\chi_\Omega \quad \text{in } \mathbb{R}^3, \tag{4b}$$

where $\boldsymbol{J}_\mathrm{e}$ denotes the electric current density (in A/m^2). Two other prominent examples are related to the so-called spin transfer torque [74, 28], which arises in the presence of

spin-polarized currents. The Slonczewski contribution [74], which takes the form

$$\begin{aligned} \boldsymbol{T}(\boldsymbol{m}) &= \frac{\hbar|\boldsymbol{J}_\mathrm{e}|G(\boldsymbol{m}\cdot\boldsymbol{p},P)}{e\mu_0 M_\mathrm{s} d}\boldsymbol{m}\times\boldsymbol{p}, \\ \text{with}\quad G(\boldsymbol{m}\cdot\boldsymbol{p},P) &= \left[\frac{(1+P)^3(3+\boldsymbol{m}\cdot\boldsymbol{p})}{4P^{3/2}} - 4\right]^{-1}, \end{aligned} \tag{5}$$

is used for the simulation of switching processes in structures with current-perpendicular-to-plane injection geometries, e.g., magnetic multilayers. The involved physical quantities are the reduced Planck constant $\hbar > 0$ (in J s), the elementary charge $e > 0$ (in A s), a dimensionless polarization parameter $P \in (0,1)$, a dimensionless unit vector $\boldsymbol{p} \in \mathbb{R}^3$ representing the magnetization of a uniformly-magnetized polarizing layer (the so-called fixed layer), and the thickness $d > 0$ of the so-called free layer (in m). The Zhang–Li contribution [79, 77] is used for the simulation of the current-driven motion of domain walls in single-phase samples characterized by current-in-plane injection geometries and, according to [79] (resp., [77]), takes the form

$$\boldsymbol{T}(\boldsymbol{m}) = -\frac{1}{\gamma_0}[\boldsymbol{m}\times(\boldsymbol{u}\cdot\nabla)\boldsymbol{m} + \xi(\boldsymbol{u}\cdot\nabla)\boldsymbol{m}], \tag{6a}$$

$$\text{with}\quad \boldsymbol{u} = -\frac{P\mu_B}{eM_\mathrm{s}(1+\xi^2)}\boldsymbol{J}_\mathrm{e} \quad \left(\text{resp., } \boldsymbol{u} = -\frac{Pg_\mathrm{e}\mu_B}{2eM_\mathrm{s}}\boldsymbol{J}_\mathrm{e}\right). \tag{6b}$$

The involved physical quantities are the spin velocity vector $\boldsymbol{u} \in \mathbb{R}^3$ (in m/s), the dimensionless ratio of nonadiabaticity $\xi > 0$, the Bohr magneton $\mu_B > 0$ (in A m^2), and the dimensionless g-factor of the electron $g_\mathrm{e} \approx 2$.

In (1) and (3), to fix the ideas, we considered the case of the bulk DMI as a prototype for chiral interactions [40, 64]. However, our implementation also covers the case of the interfacial DMI; see, e.g., [37, 68]. If $D = 0$ (no chiral interaction), then the boundary conditions (3b) become homogeneous Neumann boundary conditions.

3. Algorithms

The algorithms implemented in our Python module *Commics* employ a uniform partition of the time interval with constant time-step size $\Delta t > 0$. For the spatial discretization, we consider a tetrahedral mesh $\mathcal{T}_h$ of the ferromagnet Ω with mesh size $h > 0$. The associated FEM space of piecewise affine and globally continuous functions reads

$$V_h := \{\varphi_h \colon \Omega \to \mathbb{R} \text{ continuous} : \varphi_h|_K \text{ is affine for all elements } K \in \mathcal{T}_h\}. \tag{7}$$

For each time-step $n = 0, 1, 2, \ldots$, we seek for approximations

$$(V_h)^3 \ni \boldsymbol{m}_h^n \approx \boldsymbol{m}(n\Delta t) \quad \text{such that} \quad |\boldsymbol{m}_h^n(\boldsymbol{z})| = 1 \text{ for all nodes } \boldsymbol{z} \text{ of } \mathcal{T}_h,$$

i.e., for any time-step, the approximate magnetization satisfies the unit-length constraint $|\boldsymbol{m}| = 1$ at the nodes of the mesh.

3.1. Tangent plane scheme. Tangent plane schemes (sometimes also referred to as *projection methods*) are based on variational formulations of the equivalent form of LLG

$$\alpha\,\partial_t\boldsymbol{m} + \boldsymbol{m}\times\partial_t\boldsymbol{m} = \gamma_0(\boldsymbol{H}_\mathrm{eff}(\boldsymbol{m}) + \boldsymbol{T}(\boldsymbol{m})) - \gamma_0[(\boldsymbol{H}_\mathrm{eff}(\boldsymbol{m}) + \boldsymbol{T}(\boldsymbol{m}))\cdot\boldsymbol{m}]\boldsymbol{m}.$$

The orthogonality $\boldsymbol{m} \cdot \partial_t \boldsymbol{m} = 0$, which characterizes any solution of (3a), is enforced at the discrete level by considering the discrete tangent space

$$\boldsymbol{\mathcal{K}}_h(\boldsymbol{m}_h^n) := \{\boldsymbol{\varphi}_h \in (V_h)^3 : \boldsymbol{\varphi}_h(\boldsymbol{z}) \cdot \boldsymbol{m}_h^n(\boldsymbol{z}) = 0 \text{ for all nodes } \boldsymbol{z} \text{ of } \mathcal{T}_h\} \subset (V_h)^3.$$

For each time-step, one has to solve a constrained linear system to compute $\boldsymbol{v}_h^n \approx \partial_t \boldsymbol{m}(n\Delta t)$ in $\boldsymbol{\mathcal{K}}_h(\boldsymbol{m}_h^n)$. With $\boldsymbol{m}_h^n$ and $\boldsymbol{v}_h^n$ at hand, one then computes

$$\boldsymbol{m}_h^{n+1} \in (V_h)^3 \quad \text{by} \quad \boldsymbol{m}_h^{n+1}(\boldsymbol{z}) := \frac{\boldsymbol{m}_h^n(\boldsymbol{z}) + \Delta t\, \boldsymbol{v}_h^n(\boldsymbol{z})}{|\boldsymbol{m}_h^n(\boldsymbol{z}) + \Delta t\, \boldsymbol{v}_h^n(\boldsymbol{z})|} \quad \text{for all nodes } \boldsymbol{z} \text{ of } \mathcal{T}_h.$$

The original tangent plane scheme from [12] is formally first-order in time and was analyzed for the energy being only the exchange contribution. The scheme was extended to general lower-order effective field contributions [14, 33], DMI [52], and the coupling with other partial differential equations, e.g., various forms of Maxwell's equations [60, 19, 59, 42], spin diffusion [10], and magnetostriction [20]. A projection-free version of the method was analyzed in [10, 67]. In a variant from [13, 38], the formal convergence order in time has been increased from one to two. Effective solution strategies and preconditioning for the resulting constrained linear system have recently been proposed in [55].

3.2. Midpoint scheme. The midpoint scheme is based on a variational formulation of the Gilbert form (3a) of LLG. It consists of two fundamental ingredients: the implicit midpoint rule in time and the mass-lumped L^2-product in space, defined as

$$\langle \boldsymbol{\varphi}, \boldsymbol{\psi} \rangle_h := \int_\Omega \mathcal{I}_h(\boldsymbol{\varphi} \cdot \boldsymbol{\psi}) \mathrm{d}\boldsymbol{x} \approx \langle \boldsymbol{\varphi}, \boldsymbol{\psi} \rangle_{\boldsymbol{L}^2(\Omega)} \quad \text{for all } \boldsymbol{\varphi}, \boldsymbol{\psi} \colon \Omega \to \mathbb{R}^3 \text{ continuous.} \tag{8}$$

Here, $\mathcal{I}_h$ is the standard nodal interpolant associated with $\mathcal{T}_h$. The resulting scheme is second-order in time, inherently preserves the unit-length constraint and the energy of the solutions, but requires the solution of one *nonlinear* system for $\boldsymbol{m}_h^{n+1/2} := (\boldsymbol{m}_h^{n+1} + \boldsymbol{m}_h^n)/2 \in (V_h)^3$ per time-step. The midpoint scheme was proposed and analyzed in [25]. The scheme was extended to lower-order terms [65], the coupling with the Maxwell equation [17], and a variant of LLG in heat-assisted magnetic recording [21, 22]. The resulting nonlinear system is usually solved with constraint-preserving fixed-point iterations, which, however, spoil the unconditional convergence.

3.3. Magnetostatic Maxwell equations. For the computation of stray field and Oersted field, i.e., for the numerical solution of the magnetostatic Maxwell equations (2) and (4), we follow a hybrid FEM-BEM approach, which combines FEM with the boundary element method (BEM); see [45, 51]. The method uses the superposition principle and computes the magnetic scalar potential u such that $\boldsymbol{H}_\mathrm{s} = -\nabla u$ and the Oersted field $\boldsymbol{H}_\mathrm{c}$ by splitting the problem into two parts, where BEM techniques for the evaluation of the double-layer potential are employed; see, e.g., [65, Chapter 4.1] for details.

4. Implementation

Our Python module *Commics* is based on the *Netgen/NGSolve* [6] (*NGS*) FEM software and provides a tool to perform micromagnetic simulations with the algorithms described in Section 3 for a variety of energy contributions and dissipative effects. It is

purely Python-based, provides extensive simulation data for reproducibility and post-processing, and automatically takes care of, e.g., the definition and the assembly of underlying (bi)linear forms as well as the numerical solution of the arising linear and nonlinear systems. Although not needed for the use of *Commics*, this section advertises some of the core features of *NGS*, as well as the coupling of *NGS* with *BEM++*.

4.1. Basic features. Geometry handling, mesh-generation, FEM spaces, and assembly routines are intentionally hidden from the user of *Commics* and are internally covered in the *NGS* framework. For instance, given an *NGS* object `mesh`, representing a tetrahedral mesh of the domain Ω, the discrete vector-valued product space $(V_h)^3$ is simply generated by

```
Vh3 = VectorH1(mesh, order=1)
```

where the syntax `VectorH1` indicates that `Vh3` is a proper subspace of the vector-valued Sobolev space $(H^1(\Omega))^3$.

4.2. Symbolic (bi)linear forms. Although the use of *Commics* requires minimal knowledge of *NGS*, we shortly describe one feature of the library, namely the definition of (bi)linear forms: *NGS* allows the symbolic definition of (time-dependent) (bi)linear forms. For instance, given the space `Vh3` defined in Section 4.1, the LLG-specific cross-product bilinear form

$$\langle \boldsymbol{m}_h^n \times \boldsymbol{\psi}_h, \boldsymbol{\varphi}_h \rangle_{\boldsymbol{L}^2(\Omega)} \quad \text{for all } \boldsymbol{\varphi}_h, \boldsymbol{\psi}_h \in (V_h)^3 \tag{9}$$

is symbolically defined on the Python level by the following code snippet:

```
# grid-, trial- and testfunction
psi = Vh3.TrialFunction()
phi = Vh3.TestFunction()
m = GridFunction(Vh3)
# bilinear form
LHS = BilinearForm(Vh3)
ir = IntegrationRule(TET, order=3)
LHS += SymbolicBFI(Cross(m, psi) * phi, intrule=ir)
```

In the last line, `SymbolicBFI` with the test function `phi` and the trial function `psi` realizes the bilinear form (9) and adds it to the left-hand side `LHS` for further handling. Note that via the `intrule` option, the order of the quadrature is explicitly set to 3, since the realization of the bilinear form corresponds to the exact integration of piecewise cubic polynomials on tetrahedra, hence also the parameter `TET`. For the midpoint scheme, the corresponding bilinear form employs the mass-lumped L^2-product from (8), i.e.,

$$\langle \boldsymbol{m}_h^n \times \boldsymbol{\psi}_h, \boldsymbol{\varphi}_h \rangle_h \quad \text{for all } \boldsymbol{\varphi}_h, \boldsymbol{\psi}_h \in (V_h)^3. \tag{10}$$

This bilinear form can be defined in the same way as above specifying the corresponding quadrature rule on the reference tetrahedron $\mathrm{conv}\{\boldsymbol{0}, \boldsymbol{e}_1, \boldsymbol{e}_2, \boldsymbol{e}_3\}$:

```
ir = IntegrationRule(points=[(0,0,0), (1,0,0), (0,1,0), (0,0,1)], \
                     weights=[1/24, 1/24, 1/24, 1/24])
```

4.3. Coupling with *BEM++*. For the approximate computation of the magnetostatic fields with the hybrid FEM-BEM approach from Section 3.3, the evaluation of the double-layer potential is required. To that end, we incorporate into *Commics* the corresponding functionality of the BEM software *BEM++* [75], including matrix compression techniques [49]. The coupling of *NGS* and *BEM++* is done on the Python level with the *ngbem* module [2]; see Appendix A.

5. Using Commics: Standard problems and numerical experiments

In this section, we present some numerical experiments performed with *Commics*. For each proposed example, we also include the executable Python script to run the simulation.

5.1. Using *Commics*. To run a micromagnetic simulation with *Commics*, the user has to define an object of class `commics.Integrator` and call its `Integrate` method.

The essential inputs to initialize such an object are the geometry as a `commics.Geometry` object defining the ferromagnetic domain and the meshing strategy (see Section 5.2), as well as an object of type `commics.Parameters` specifying, among other things, material parameters, an applied field or current, the initial magnetization state, and the time discretization strategy. Moreover, the desired time integration scheme has to be chosen. The algorithms for the numerical integration of LLG implemented in *Commics* described in Section 3 can be selected in the following way:

- The first-order tangent plane scheme from [12] with explicit integration of the lower-order terms in time [14, 33] is provided as `TPS1`.
- The projection-free tangent plane scheme (with explicit integration of the lower-order terms in time) from [10, 67] is provided as `TPS1PF`.
- The second-order tangent plane scheme from [13] and its improved version from [38] are available as `TPS2` and `TPS2AB`, respectively.
- The midpoint scheme from [25] and its improved version from [65] are provided as `MPS` and `MPSAB`, respectively.

Various example scripts are provided subsequently in this section. Further details can be accessed with the Python built-in `help()` function available for *Commics* classes.

5.2. Geometry specification and meshing. *Commics* provides several ways to specify a geometry and to generate a corresponding mesh:

- For complex geometries, the submodule `netgen.csg` of *NGS* provides a rich number of possibilities to define geometries; see [6].
- For geometries often encountered in micromagnetics, e.g., cuboids and disks, one can simply provide the dimensions of the sample, a scale factor, and the desired maximum mesh size; see, e.g., the code snippets in Section 5.3.1 and Section 5.6. Then, `netgen.csg` will automatically be used with appropriate settings.
- Existing *NGS* meshes/geometries (stored as `.vol`-files) can simply be loaded; see, e.g., the code snippet in Section 5.3.2.

Meshes automatically generated by *NGS* are unstructured and obtained by the advancing front method; see [69] for details. However, due to mesh quality and shape optimizations, these meshes do not necessarily satisfy the prescribed maximum mesh size. *Commics*

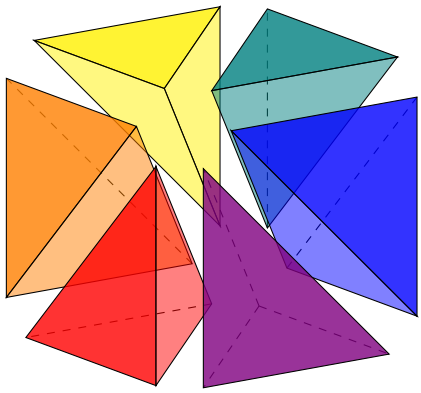

FIGURE 1. Uniform decomposition of a cube into six tetrahedra.

provides two possibilities to bypass this drawback: For cuboidal geometries, setting the *Commics* option `structuredMesh=True` allows for structured meshes. First, the sample is uniformly split into cuboidal cells of prescribed size. Then, each cell is split into six tetrahedra in such a way that any tetrahedron has three mutually perpendicular edges; see Figure 1. This strategy is for example used in Section 5.4. For general geometries, *Commics* provides the means to repeatedly generate a new mesh (by prescribing a smaller and smaller mesh size each time) in *NGS*, until the initial mesh size specification is satisfied. This strategy can be enabled by setting `forceNetgenMeshSize=True` as done in Section 5.6.

5.3. μMAG standard problem #4. To describe the key aspects of a *Commics* script, we consider the μMAG standard problem #4 [7].

The objective is the simulation of the magnetization dynamics in a thin permalloy film of dimensions $500\,\mathrm{nm} \times 125\,\mathrm{nm} \times 3\,\mathrm{nm}$ under the influence of a constant applied external field. We split the experiment into two parts: In the first stage, we obtain the so-called equilibrium S-state, which is saved to serve as the initial configuration for the second stage, where the switching dynamics is simulated.

5.3.1. *Obtaining the S-state.* We consider a structured tetrahedral mesh of the given cuboid into cells of size $h_x \times h_y \times h_z$, which are decomposed into tetrahedra as depicted in Figure 1. The dimension of the cells is chosen as $h_x = h_y = 125/69\,\mathrm{nm}$ and $h_z = 1.5\,\mathrm{nm}$. This corresponds to 228 528 elements with diameter $h = 2.97\,\mathrm{nm}$, 58 170 vertices, as well as 78 936 surface elements. The material parameters of permalloy read $M_s = 8 \cdot 10^5\,\mathrm{A/m}$, $A = 1.3 \cdot 10^{-11}\,\mathrm{J/m}$, $D = 0\,\mathrm{J/m^2}$, and $K = 0\,\mathrm{J/m^3}$. To speed up the process, we deliberately choose the large value $\alpha = 1$ for the Gilbert damping parameter. For the simulation, we use a constant time-step size $\Delta t = 0.1\,\mathrm{ps}$.

The problem description suggests to obtain the S-state by applying a slowly reducing external field pointing in the $(1,1,1)$-direction. We start with a uniform initial state $\boldsymbol{m}_h^0 \equiv (1,0,0)$ and let the magnitude $|\boldsymbol{H}_{\mathrm{ext}}|$ of the external field decrease linearly from $30/\mu_0$ to $0\,\mathrm{mT}$ over a period of $1\,\mathrm{ns}$. In *Commics* scripts, non-constant fields can be described using time- and space-dependent Python `lambda`-functions. Further, we relax the system for 1 ns without applying any external field and store the obtained S-state as `sp4sState.vtk` for later use.

```
from commics import *
# specify geometry and parameters
h_xy, h_z = 125/69, 1.5
geo = Geometry(geometry=Cuboid(500, 125, 3), meshSize=(h_xy, h_xy, h_z), \
               structuredMesh=True, scaling=1e-9)
par = Parameters(A=1.3e-11, Ms=8e+5, K=0, D=0, gamma0=2.211e+5, \
                 alpha=1.0, m0=(1.0, 0.0, 0.0), \
                 T_start=0, T_end=1e-9, timeStepSize=0.1e-12)
# define time dependent applied field via lambda function
from math import sqrt
field = lambda t,x,y,z : (par.T_end - t) / (par.T_end - par.T_start) \
        * 30.0e-3 / par.mu0 / sqrt(3.0)
par.H_ext = (field, field, field)
# define integrator and run simulation from T_start to T_end
sp4 = Integrator(scheme=TPS2AB, geometry=geo, parameters=par)
sp4.Integrate()
# Relax for another nanosecond and save the mesh and the result
sp4.Integrate(duration=1e-9, relax=True)
sp4.SaveMesh("sp4mesh")
sp4.SaveMagnetization("sp4sState")
```

5.3.2. *Switching.* We assume that the folder `data` contains the two files `sp4sState.vtk` and `sp4mesh.vol` saved from the simulation described in Section 5.3.1. As stated in the problem description, we choose $\alpha = 0.02$ and set the external field to $\boldsymbol{H}_{\text{ext}} = (-24.6, 4.3, 0)/\mu_0$ mT. Then, we run the simulation for 3 ns.

```
from commics import *
# specify geometry and parameters
geo = Geometry(geometry="data/sp4mesh.vol", scaling=1e-9)
par = Parameters(A=1.3e-11, Ms=8e+5, K=0, D=0, gamma0=2.211e+05, \
                 alpha=0.02, timeStepSize=0.1e-12, m0="data/sp4sState.vtk")
par.H_ext = (-24.6e-3/par.mu0, 4.3e-3/par.mu0, 0.0)
# define integrator and run simulation
sp4 = Integrator(scheme=TPS2AB, geometry=geo, parameters=par)
sp4.Integrate(duration=3e-9)
```

For comparison, the desired output of this benchmark problem is the evolution of the x-, y- and z-component of the spatially averaged magnetization. Figure 2 shows, that our results match those computed by the finite difference code *OOMMF* [39] available on the μMAG homepage [7]. Further, Figure 3 visualizes the magnetization at the time when the x-component of the spatially averaged magnetization first crosses zero.

5.3.3. *Meshing strategy and integrator.* In Section 5.3.1, we considered a structured mesh and the `TPS2AB` integrator from [38]. To compare the results, we additionally repeat the simulation on an unstructured mesh using the midpoint scheme: To simulate the dynamics on an unstructured mesh generated by *NGS*, we replace the definition of the geometry in Section 5.3.1 by

```
geo = Geometry(geometry=Cuboid(500, 125, 3), meshSize=3, scaling=1e-9)
```

This results in an unstructured mesh containing 48 792 elements, 16 682 vertices and 33 360 surface elements, which corresponds to an actual mesh size of 5.19 nm. To repeat

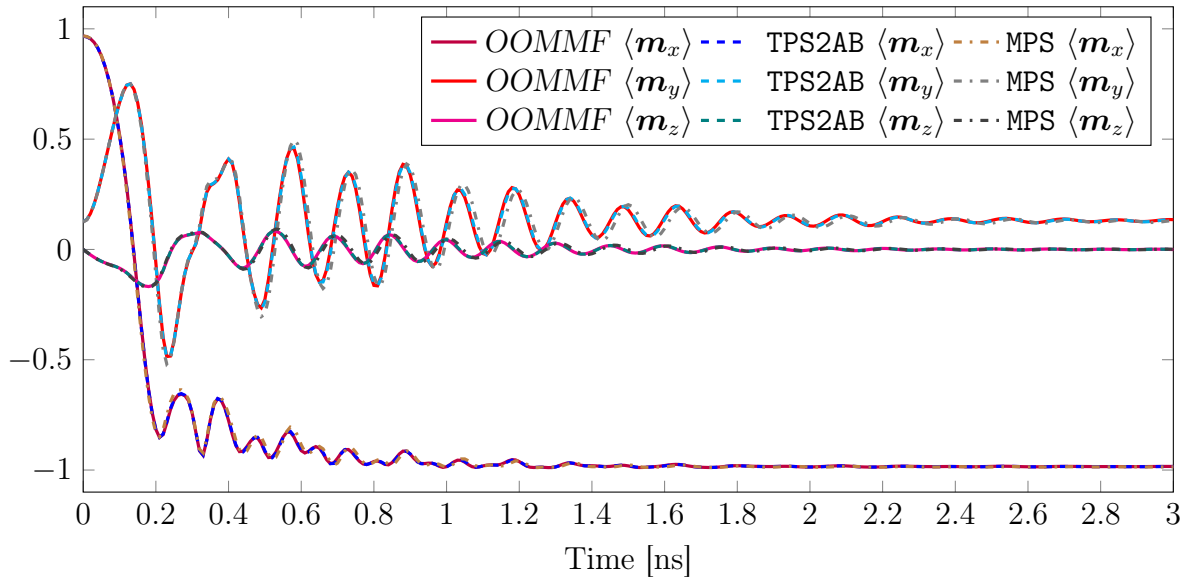

FIGURE 2. μMAG standard problem #4 from Section 5.3: Time evolution of the spatially averaged magnetization components computed with *Commics* (`TPS2AB` and `MPS`) compared to the results of *OOMMF*.

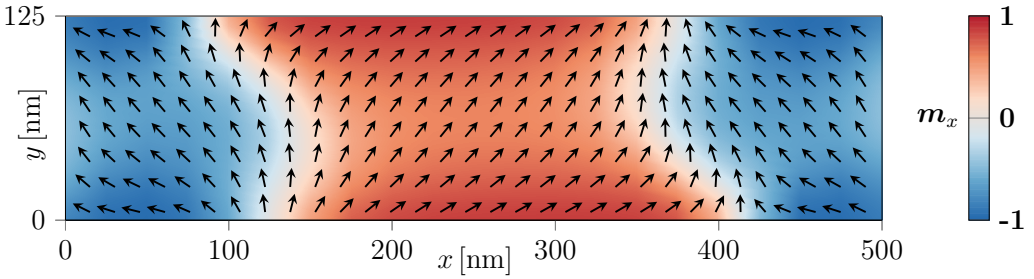

FIGURE 3. μMAG standard problem #4 from Section 5.3: Snapshot of the magnetization when the x-component of the spatially averaged magnetization first crosses zero ($t = 138.2\,\mathrm{ps}$).

the simulation using the midpoint scheme from [25], we replace the definition of the integrator in Section 5.3.1 and Section 5.3.2 by

```
sp4 = Integrator(scheme=MPS, geometry=geo, parameters=par)
```

Although the mesh size is close to the exchange length of the material (5.69 nm), qualitatively the results match those computed by *OOMMF* well; see Figure 2.

5.4. μMAG standard problem #5. The spintronic extensions of LLG from [79, 77] are the subject of the μMAG standard problem #5 [7]. The sample under consideration is a permalloy film with dimensions $100\,\mathrm{nm} \times 100\,\mathrm{nm} \times 10\,\mathrm{nm}$ aligned with the x, y, and z axes of a Cartesian coordinate system, with origin at the center of the film. We consider the same material parameters as in Section 5.3 and $\alpha = 0.1$. The initial state is obtained by solving (3) with $\boldsymbol{T} \equiv \boldsymbol{0}$ and $\boldsymbol{m}^0(x, y, z) = (-y, x, R)/\sqrt{x^2 + y^2 + R^2}$ with $R = 10\,\mathrm{nm}$

and maximal damping $\alpha = 1$ for 1 ns, which is a sufficiently long time for the system to reach equilibrium. Given $P\boldsymbol{J}_e = (1 \cdot 10^{12}, 0, 0)$ A/m^2 and $\xi = 0.05$, we set $\boldsymbol{T}$ according to the expression in (6a). Then, we solve (3) with the relaxed magnetization configuration as initial condition for 8 ns, which turns out to be a sufficiently long time to reach the new equilibrium; see Figure 4 and Figure 5. We consider a structured tetrahedral mesh of the given cuboid into cells of size $h_x \times h_y \times h_z$, which are decomposed into tetrahedra as depicted in Figure 1. The dimension of the cells is chosen as $h_x = h_y = h_z = 5/3$ nm. This corresponds to 129 600 elements with diameter $h = 2.89$ nm, 26 047 vertices, as well as 17 280 surface elements.

```
from commics import *
# specify geometry and parameters
h = 5 / 3
geo = Geometry(geometry=Cuboid((-50,-50,-5), (50,50,5)), meshSize=(h,h,h), \
                structuredMesh=True, scaling=1.0e-09)
par = Parameters(A=1.3e-11, Ms=8.0e+05, K=0.0, gamma0=2.211e+05, alpha=1.0, \
                 spintronicsCoupling="zhangLi", g=2.0, P=1.0, \
                 Je=(1.0e+12, 0.0, 0.0), xi=0.05, \
                 timeStepSize=0.1e-12, T_start=-1.0e-09)
# space dependent initial condition (scaled domain); normalized automatically
from ngsolve import x, y
par.m0 = (-y, x, 10.0)
# define integrator and relax configuration
sp5 = Integrator(scheme=TPS2AB, geometry=geo, parameters=par)
sp5.Integrate(duration=1.0e-09, relax=True)
# set damping alpha and run simulation with specified current
sp5.SetParameter_alpha(0.1)
sp5.Integrate(duration=8.0e-09)
```

5.5. Standard problem for ferromagnetic resonance simulations. Ferromagnetic resonance (FMR) is a well-established experimental technique for the study of ferromagnetic materials. A typical application of FMR consists in perturbing the magnetization of a system from its equilibrium by a sufficiently weak excitation and studying the induced magnetization dynamics, which is basically made of damped oscillations around the initial equilibrium. The resulting resonance frequencies and the eigenmodes of the system give some insights on the magnetic properties of the material and are used, e.g., for the experimental measurement of model parameters like the Gilbert damping constant or the saturation magnetization [41, 62].

In this section, we compute with *Commics* a problem for FMR simulations recently proposed in [16]. The computational domain is a cuboid of permalloy with dimensions 120 nm × 120 nm × 10 nm. The material parameters are the same as in Section 5.3. During the first stage, we set $\alpha = 1$ and consider a constant applied external field of magnitude $|\boldsymbol{H}_{\text{ext}}| = 8 \cdot 10^4$ A/m pointing in the direction $(1, 0.715, 0)$. We initialize the system with a uniform ferromagnetic state $\boldsymbol{m}_h^0 \equiv (1, 0, 0)$ and let the system evolve for 5 ns. The resulting state is then used as initial condition for the second stage, in which we set $\alpha = 0.008$, change the direction of the applied external field to $(1, 0.7, 0)$ but keep $|\boldsymbol{H}_{\text{ext}}| = 8 \cdot 10^4$ A/m , and let the system evolve to the new equilibrium for 20 ns. We consider a structured tetrahedral mesh of the given cuboid into cells of size $h_x \times h_y \times h_z$, which are decomposed into tetrahedra as depicted in Figure 1. The dimension of the cells is chosen as $h_x = h_y = h_z = 2$ nm. This corresponds to 108 000 elements, 22 326 vertices,

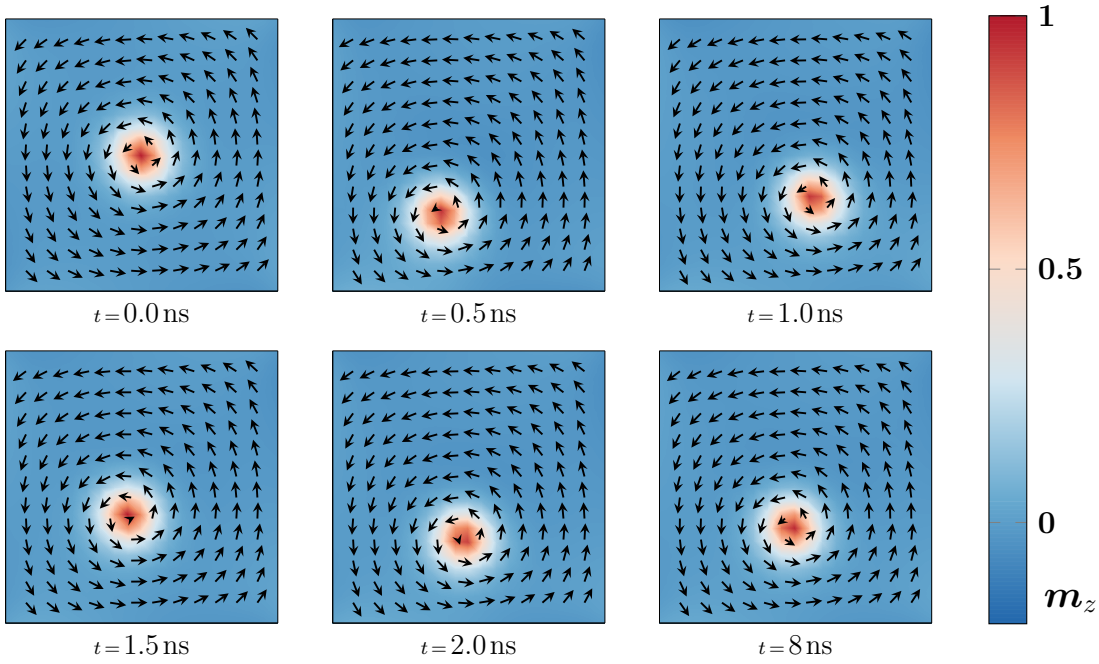

FIGURE 4. μMAG standard problem #5: Magnetization in the xy-plane viewed from the top at different times. Starting from the relaxed configuration at $t = 0.0\,\mathrm{ns}$, the vortex (red) follows a spiral-like motion. After $t = 8.0\,\mathrm{ns}$ no further movements are observed.

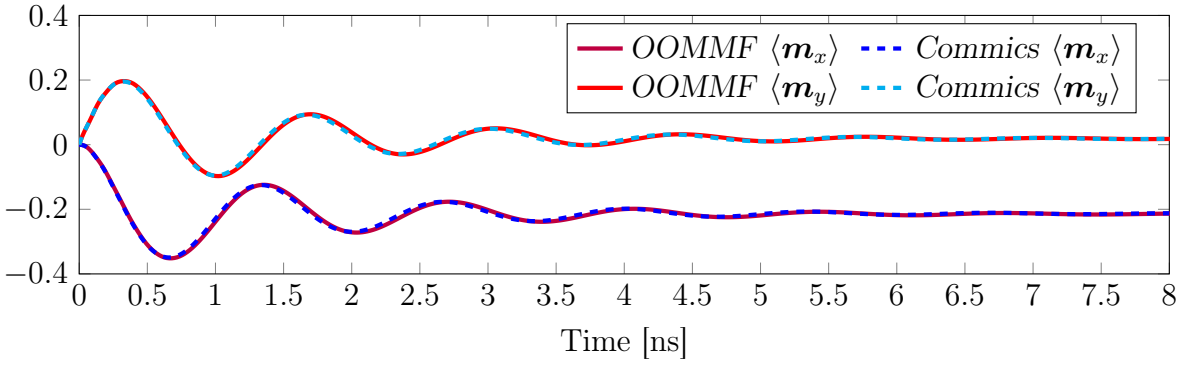

FIGURE 5. μMAG standard problem #5: Evolution of the spatially averaged x- and y-component of $\boldsymbol{m}$.

as well as 16 800 surface elements, and yields a mesh size of $h = 3.46\,\mathrm{nm}$. We compare our results obtained with *Commics* to those presented in [16]. There, the authors use the finite difference code *OOMMF* [39] and investigate the evolution of the y-component of the spatially averaged magnetization, as well as its power spectrum S_y. The power spectrum is obtained by a discrete Fourier transform as described in [16, Section 2.3.1]. Our results match well with those of [16]; see Figure 6 and Figure 7.

```
from commics import *
# specify geometry and parameters
geo = Geometry(geometry=Cuboid(120, 120, 10), meshSize=(2,2,2), \
               structuredMesh=True, scaling=1.0e-09)
par = Parameters(A=1.3e-11, Ms=8.0e+05, K=0.0, gamma0=2.210173e+05, \
                 alpha=1.0, timeStepSize=0.1e-12, T_start=-5.0e-9, \
                 m0=(0, 0, 1))
# define integrator, choose number of threads, specify folder for results
fmr = Integrator(scheme=TPS2AB, geometry=geo, parameters=par, numthreads=8)
fmr.SetResultsFolder("FMR_Result")
# obtain initial condition
amplitude = 80.0e+03
e = (1.0, 0.715, 0.0)
e_length = (sum(e[j]**2 for j in range(3)))**0.5
H_ext = (amplitude*e[0]/e_length, amplitude*e[1]/e_length, 0.0)
fmr.SetParameter_H_ext(H_ext)
fmr.Integrate(duration=5.0e-09)
# change direction of applied field and run simulation
fmr.SetParameter_alpha(0.008)
e = (1.0, 0.7, 0.0)
e_length = (sum(e[j]**2 for j in range(3)))**0.5
H_ext = (amplitude*e[0]/e_length, amplitude*e[1]/e_length, 0.0)
fmr.SetParameter_H_ext(H_ext)
fmr.Integrate(duration=20.0e-09)
```

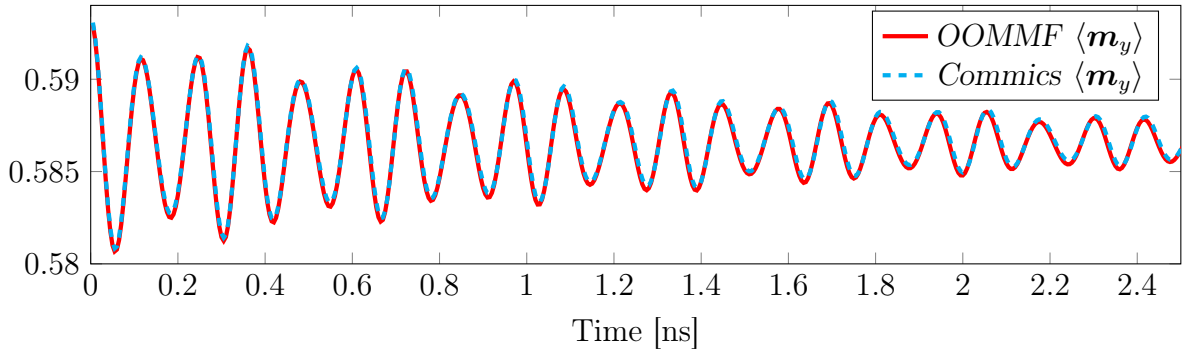

FIGURE 6. Ferromagnetic resonance simulation from Section 5.5: Time evolution of $\langle \boldsymbol{m}_y \rangle$ obtained with *Commics* compared to the results computed with *OOMMF*.

5.6. Current-induced dynamics of skyrmions in nanodisks. With this experiment, we aim to show how *Commics* can be used to numerically investigate the stability and the induced dynamics of magnetic skyrmions in helimagnetic materials in response to spin-polarized currents.

We consider a magnetic nanodisk of diameter 120 nm (x_1x_2-plane) and thickness $d = 10\,\text{nm}$ (x_3-direction). We use the material parameters of iron-germanium (FeGe), i.e., $M_\text{s} = 3.84 \cdot 10^5\,\text{A/m}$, $A = 8.78 \cdot 10^{-12}\,\text{J/m}$, $D = 1.58 \cdot 10^{-3}\,\text{J/m}^2$, and $K = 0\,\text{J/m}^3$; see, e.g., [26]. The initial condition for our experiment is obtained by relaxing a uniform out-of-plane ferromagnetic state $\boldsymbol{m}^0 \equiv (0, 0, 1)$ for 2 ns. For the relaxation process, we

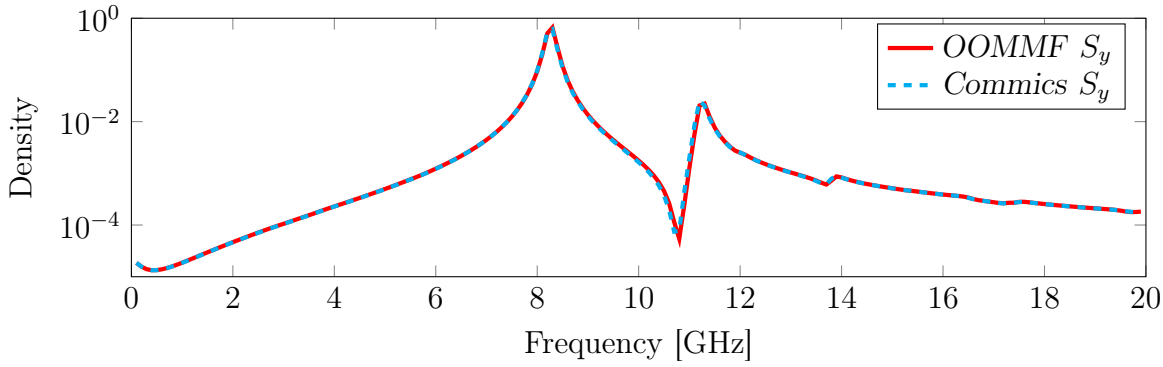

FIGURE 7. Ferromagnetic resonance simulation from Section 5.5: Power spectrum S_y obtained by discrete Fourier transform of the spatially averaged y-component of the magnetization $\langle \boldsymbol{m}_y \rangle$.

choose the large value $\alpha = 1$ for the Gilbert damping constant. The resulting relaxed state is the skyrmion depicted in Figure 8.

Starting from this configuration, we apply a perpendicular spin-polarized current pulse $\boldsymbol{J}_e(t) = (0, 0, J(t))$ of maximum intensity $J_{\max} > 0$ for 150 ps; see Figure 9. To model the resulting spin transfer torque, we include $\boldsymbol{T}$ from (5). Then, we turn off the current density and let the system evolve for 20 ns. In order to capture all possible excitation modes during the application of the pulse and the subsequent relaxation process, we set the value of the Gilbert damping constant to $\alpha = 0.002$, which is considerably smaller than the experimental value of $\alpha = 0.28$ measured for FeGe; see [26].

In Figure 9, we plot the time evolution of the first component of the spatially averaged magnetization of the sample after a current pulse with $J_{\max} = 1 \cdot 10^{12}\,\mathrm{A/m^2}$, $\boldsymbol{p} = (0, 1, 0)$, and $P = 0.4$. The induced dynamics is a damped precession of the skyrmion around the center of the sample; see Figure 8. We consider an unstructured tetrahedral mesh of the nanodisk generated by *NGS*. For a desired mesh size of 7.5 nm, the automatically generated mesh consists of 35 390 elements with maximum diameter $h = 6.1$ nm, 8436 vertices, as well as 9586 surface elements.

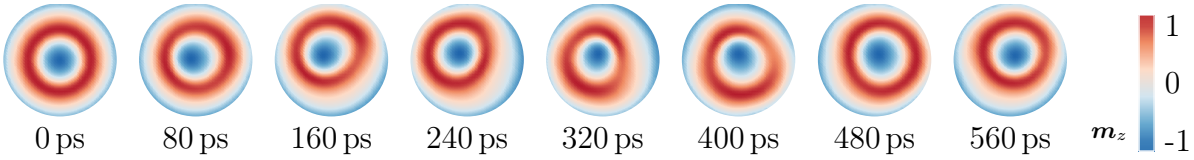

FIGURE 8. Snapshots of the skyrmion dynamics from Section 5.6: Magnetization in the xy-plane viewed from the top at different times. Starting from the relaxed configuration at $t = 0$ ps, the skyrmion is deflected from the center of the disk by a current pulse. Then, the skyrmion oscillates around the center of the disk with an observed period of approximately 400 ps. Due to damping, over the relaxation period of 20 ns the amplitude of the oscillations decreases to almost zero, and the initial equilibrium configuration is restored.

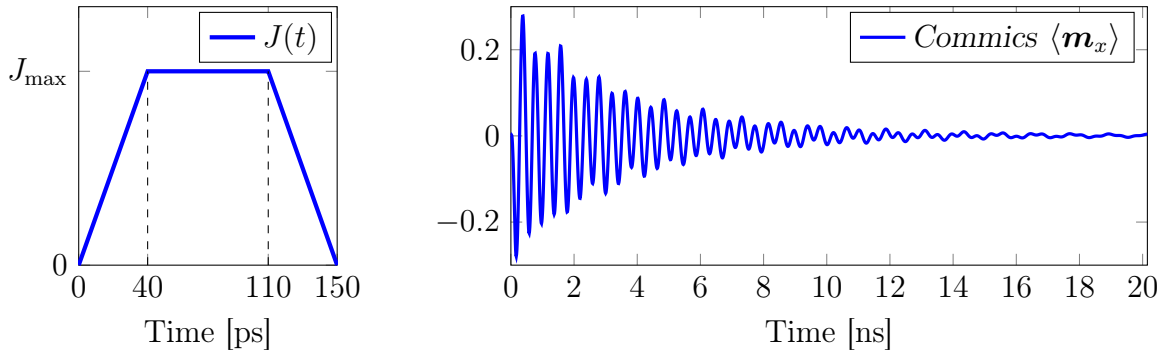

FIGURE 9. Simulation of the skyrmion dynamics from Section 5.6. Structure of the applied current pulse (left). Time evolution of the x-component of the spatially averaged magnetization (right).

```
from commics import *
# specify geometry and parameters
geo = Geometry(geometry=Disk(120.0, 10.0), meshSize=7.5, \
               forceNetgenMeshSize=True, scaling=1.0e-09)
par = Parameters(A=8.78e-12, Ms=3.84e+05, D=1.58e-03, dmCoupling="bulk", \
                 spintronicsCoupling="slonczewski", d=10.0e-09, P=0.4, \
                 p=(0.0, 1.0, 0.0), \
                 alpha=1.0, timeStepSize=1e-13, theta=1.0, m0=(0, 0, 1))
# define integrator, choose number of threads
helimag = Integrator(scheme=TPS1, geometry=geo, parameters=par, numthreads=8)
# choose to save magnetization as .vtk-file every X seconds, run simulation
helimag.RecordMagnetization(every=10.0e-12)
helimag.Integrate(duration=2.0e-9, relax=True)
# prepare JeMax, alpha, and points in time
helimag.SetParameter_alpha(0.002)
JeMax = 1.0e+12
T0, T1, T2, T3 = 0, 40.0e-12, 110.0e-12, 150.0e-12
# set increasing current, run simulation
helimag.SetParameter_Je(lambda t,x,y,z : (t-T0)/(T1-T0)*JeMax)
helimag.Integrate(duration=T1-T0)
# set constant current, run simulation
helimag.SetParameter_Je(JeMax)
helimag.Integrate(duration=T2-T1)
# set decreasing current, run simulation
helimag.SetParameter_Je(lambda t,x,y,z : (T3-t)/(T3-T2)*JeMax)
helimag.Integrate(duration=T3-T2)
# final relaxation
helimag.Integrate(duration=20.0e-9, relax=True)
```

Appendix A. Coupling of *NGS* with *BEM++*

In this section we show how *NGS*, *BEM++*, and the *ngbem* module can be used to perform stray field computations following the approach proposed in [45]:

- *NGS* is used for mesh generation and the FEM problems.
- *BEM++* provides boundary integral operators and an interface for solving the BEM problem.
- *ngbem* provides the means to extract a boundary element mesh from the volume mesh together with a mapping between the corresponding degrees of freedom.

To test the procedure we consider the uniformly magnetized unit ball. Then, the stray field is given by $\boldsymbol{H}_\mathrm{s}(\boldsymbol{M}) \equiv -\boldsymbol{M}/3$ in Ω. The complete code is provided in the following Python script.

```
# Demonstration of the Fredkin-Koehler approach for stray field computations
# using ngsolve, ngbem, bempp
import ngsolve, ngbem, bempp.api, commics, numpy
# mesh, femSpaces, GridFunctions
geo = commics.Geometry(commics.UnitBall(), meshSize=0.3)
geo.GetReady()
mesh = geo.GetMesh()
Vh3 = ngsolve.VectorH1(mesh, order=1)
Vh = ngsolve.H1(mesh, order=1)
VhD = ngsolve.H1(mesh, order=1, dirichlet="bc_dirichlet")
m = ngsolve.GridFunction(Vh3)
m.components[0].vec.FV().NumPy()[:] = 1.0
u1 = ngsolve.GridFunction(Vh)
u2 = ngsolve.GridFunction(VhD)
hs = -u1.Deriv() - u2.Deriv()
# Neumann FEM problem
a1 = ngsolve.BilinearForm(Vh)
a1 += ngsolve.Laplace(1.0)
c1 = ngsolve.Preconditioner(a1, "local")
a1.Assemble()
solverU1 = ngsolve.CGSolver(mat=a1.mat, pre=c1.mat)
phi1 = Vh.TestFunction()
f1 = ngsolve.LinearForm(Vh)
f1 += ngsolve.SymbolicLFI(phi1.Deriv() * m)
measOmega = ngsolve.Integrate(ngsolve.CoefficientFunction(1.0), mesh)
# BEM problem
bemSpace, op_NgToBem = ngbem.H1_trace(Vh)
op_NgToBem.eliminate_zeros()
bemToNgIdx = numpy.full(op_NgToBem.shape[0], fill_value = Vh.ndof, dtype=int)
bemToNgIdx[op_NgToBem.row] = op_NgToBem.col
bem_K = bempp.api.operators.boundary.laplace.double_layer(bemSpace, bemSpace, \
                                                          bemSpace)
bem_I = bempp.api.operators.boundary.sparse.identity(bemSpace, bemSpace, \
                                                     bemSpace)
bemRhsOp = bem_K - 0.5*bem_I
bemLhs = bem_I
# Dirichlet FEM problem
a2 = ngsolve.BilinearForm(VhD)
a2 += ngsolve.Laplace(1.0)
c2 = ngsolve.Preconditioner(a2, "bddc")
a2.Assemble()
f2 = ngsolve.LinearForm(VhD)
```

```
f2.Assemble()
bvp2 = ngsolve.BVP(bf=a2, lf=f2, gf=u2, pre=c2)
# solve Neumann FEM problem
f1.Assemble()
u1.vec.data = solverU1 * f1.vec
u1.vec.FV().NumPy()[:] -= ngsolve.Integrate(u1, mesh) / measOmega
# solve BEM problem
bem_u1 = bempp.api.GridFunction(bemSpace, \
                                coefficients=u1.vec.FV().NumPy()[bemToNgIdx])
bem_rhs = bemRhsOp * bem_u1
bem_g = bempp.api.linalg.iterative_solvers.gmres(bemLhs, bem_rhs)[0]
# solve Dirichlet FEM problem
u2.vec.FV().NumPy()[bemToNgIdx] = bem_g.coefficients
bvp2.Do()
# check results
TOL = 0.01
HS = ngsolve.GridFunction(Vh3)
for d in range(3): HS.components[d].Set(hs[d])
passed = all(abs(-1/3 - HS.components[0].vec.FV().NumPy()) < TOL) \
         and all(abs(HS.components[1].vec.FV().NumPy()) < TOL) \
         and all(abs(HS.components[2].vec.FV().NumPy()) < TOL)
print("PASSED STRAYFIELD TEST:", passed)
```

References

[1] Commics – A Python module for Computational Micromagnetics. `https://gitlab.asc.tuwien.ac.at/cpfeiler/commics`. Accessed on December 6, 2018.

[2] Coupling NGsolve to BEM++. `https://github.com/arieder/ngbem`. Accessed on December 6, 2018.

[3] FEMME. `http://suessco.com/en/simulations/solutions/femme-software/`. Accessed on December 6, 2018.

[4] magnum.fd. `http://micromagnetics.org/magnum.fd`. Accessed on December 6, 2018.

[5] MicroMagnum. `http://micromagnum.informatik.uni-hamburg.de`. Accessed on December 6, 2018.

[6] Netgen/NGSolve finite element library. `https://ngsolve.org`. Accessed on December 6, 2018.

[7] NIST micromagnetic modeling activity group (μMAG) website. `https://www.ctcms.nist.gov/~rdm/mumag.html`. Accessed on December 6, 2018.

[8] C. Abert, L. Exl, F. Bruckner, A. Drews, and D. Suess. magnum.fe: A micromagnetic finite-element simulation code based on FEniCS. *J. Magn. Magn. Mater.*, 345:29–35, 2013.

[9] C. Abert, L. Exl, G. Selke, A. Drews, and T. Schrefl. Numerical methods for the stray-field calculation: A comparison of recently developed algorithms. *J. Magn. Magn. Mater.*, 326:176–185, 2013.

[10] C. Abert, G. Hrkac, M. Page, D. Praetorius, M. Ruggeri, and D. Suess. Spin-polarized transport in ferromagnetic multilayers: An unconditionally convergent FEM integrator. *Comput. Math. Appl.*, 68(6):639–654, 2014.

[11] A. Aharoni. *Introduction to the theory of ferromagnetism*. Oxford University Press, Oxford, second edition, 2001.

[12] F. Alouges. A new finite element scheme for Landau–Lifchitz equations. *Discrete Contin. Dyn. Syst. Ser. S*, 1(2):187–196, 2008.

[13] F. Alouges, E. Kritsikis, J. Steiner, and J.-C. Toussaint. A convergent and precise finite element scheme for Landau–Lifschitz–Gilbert equation. *Numer. Math.*, 128(3):407–430, 2014.

[14] F. Alouges, E. Kritsikis, and J.-C. Toussaint. A convergent finite element approximation for Landau–Lifschitz–Gilbert equation. *Physica B*, 407(9):1345–1349, 2012.

[15] F. Alouges and A. Soyeur. On global weak solutions for Landau–Lifshitz equations: Existence and nonuniqueness. *Nonlinear Anal.*, 18(11):1071–1084, 1992.

[16] A. Baker, M. Beg, G. Ashton, M. Albert, D. Chernyshenko, W. Wang, S. Zhang, M.-A. Bisotti, M. Franchin, C. L. Hu, R. Stamps, T. Hesjedal, and H. Fangohr. Proposal of a micromagnetic standard problem for ferromagnetic resonance simulations. *J. Magn. Magn. Mater.*, 421:428–439, 2017.

[17] L'. Baňas, S. Bartels, and A. Prohl. A convergent implicit finite element discretization of the Maxwell-Landau–Lifshitz–Gilbert equation. *SIAM J. Numer. Anal.*, 46(3):1399–1422, 2008.

[18] L'. Baňas, Z. Brzeźniak, M. Neklyudov, and A. Prohl. *Stochastic ferromagnetism: Analysis and numerics*. De Gruyter, Berlin, 2014.

[19] L'. Baňas, M. Page, and D. Praetorius. A convergent linear finite element scheme for the Maxwell–Landau–Lifshitz–Gilbert equations. *Electron. Trans. Numer. Anal.*, 44:250–270, 2015.

[20] L'. Baňas, M. Page, D. Praetorius, and J. Rochat. A decoupled and unconditionally convergent linear FEM integrator for the Landau–Lifshitz–Gilbert equation with magnetostriction. *IMA J. Numer. Anal.*, 34(4):1361–1385, 2014.

[21] L'. Baňas, A. Prohl, and M. Slodička. Modeling of thermally assisted magnetodynamics. *SIAM J. Numer. Anal.*, 47(1):551–574, 2009.

[22] L'. Baňas, A. Prohl, and M. Slodička. Numerical scheme for augmented Landau–Lifshitz equation in heat assisted recording. *J. Comput. Appl. Math.*, 236(18):4775–4787, 2012.

[23] S. Bance, H. Özelt, T. Schrefl, M. Winklhofer, G. Hrkac, G. Zimanyi, O. Gutfleisch, R.F.L. Evans, R.W. Chantrell, T. Shoji, M. Yano, N. Sakuma, A. Kato, and A. Manabe. High energy product in battenberg structured magnets. *Appl. Phys. Lett.*, 105(19):192401, 2014.

[24] S. Bance, T. Schrefl, G. Hrkac, A. Goncharov, D. A. Allwood, and J. Dean. Micromagnetic calculation of spin wave propagation for magnetologic devices. *J. Appl. Phys.*, 103(7):07E735, 2008.

[25] S. Bartels and A. Prohl. Convergence of an implicit finite element method for the Landau–Lifshitz–Gilbert equation. *SIAM J. Numer. Anal.*, 44(4):1405–1419, 2006.

[26] M. Beg, M. Albert, M.-A. Bisotti, D. Cortés-Ortuño, W. Wang, R. Carey, M. Vousden, O. Hovorka, C. Ciccarelli, C. S. Spencer, C. H. Marrows, and H. Fangohr. Dynamics of skyrmionic states in confined helimagnetic nanostructures. *Phys. Rev. B*, 95(1):014433, 2017.

[27] M. Beg, R. A. Pepper, and H. Fangohr. User interfaces for computational science: A domain specific language for OOMMF embedded in Python. *AIP Adv.*, 7(5):056025, 2017.

[28] L. Berger. Emission of spin waves by a magnetic multilayer traversed by a current. *Phys. Rev. B*, 54(13):9353–9358, 1996.

[29] D. V. Berkov, K. Ramstöcck, and A. Hubert. Solving micromagnetic problems. Towards an optimal numerical method. *phys. stat. sol. (a)*, 137(1):207–225, 1993.

[30] M.-A. Bisotti, M. Beg, W. Wang, M. Albert, D. Chernyshenko, D. Cortés-Ortuño, R. A. Pepper, M. Vousden, R. Carey, H. Fuchs, A. Johansen, G. Balaban, L. Leoni Breth, T. Kluyver, and H. Fangohr. FinMag: finite-element micromagnetic simulation tool (Version 0.1). Zenodo. `http://doi.org/10.5281/zenodo.1216011`. Accessed on December 6, 2018.

[31] M.-A. Bisotti, D. Cortés-Ortuño, R. Pepper, W. Wang, M. Beg, T. Kluyver, and H. Fangohr. Fidimag – A finite difference atomistic and micromagnetic simulation package. *J. Open Res. Software*, 6(1), 2018.

[32] W. F. Brown. *Micromagnetics*. Interscience tracts on physics and astronomy. Interscience Publishers, New York, 1963.

[33] F. Bruckner, M. Feischl, T. Führer, P. Goldenits, M. Page, D. Praetorius, M. Ruggeri, and D. Suess. Multiscale modeling in micromagnetics: Existence of solutions and numerical integration. *Math. Models Methods Appl. Sci.*, 24(13):2627–2662, 2014.

[34] H. Brueckl, A. Satz, K. Pruegl, T. Wurft, S. Luber, W. Raberg, J. Zimmer, and D. Suess. Vortex magnetization state in a GMR spin-valve type field sensor. In *2017 IEEE International Magnetics Conference (INTERMAG)*, page 1, 2017.

[35] R. Chang, S. Li, M. V. Lubarda, B. Livshitz, and V. Lomakin. FastMag: Fast micromagnetic simulator for complex magnetic structures. *J. Appl. Phys.*, 109(7):07D358, 2011.

[36] I. Cimrák. A survey on the numerics and computations for the Landau–Lifshitz equation of micromagnetism. *Arch. Comput. Methods Eng.*, 15(3):277–309, 2008.
[37] A. Crépieux and C. Lacroix. Dzyaloshinsky–Moriya interactions induced by symmetry breaking at a surface. *J. Magn. Magn. Mater.*, 182(3):341–349, 1998.
[38] G. Di Fratta, C.-M. Pfeiler, D. Praetorius, M. Ruggeri, and B. Stiftner. Linear second order IMEX-type integrator for the (eddy current) Landau–Lifshitz–Gilbert equation. preprint, arXiv:1711.10715, 2017.
[39] M. J. Donahue and D. G. Porter. OOMMF user's guide, Version 1.0. Interagency Report NISTIR 6376, National Institute of Standards and Technology, Gaithersburg, MD, 1999.
[40] I. Dzyaloshinskii. A thermodynamic theory of 'weak' ferromagnetism of antiferromagnetics. *J. Phys. Chem. Solids*, 4(4):241–255, 1958.
[41] M. Farle. Ferromagnetic resonance of ultrathin metallic layers. *Rep. Prog. Phys.*, 61(7):755, 1998.
[42] M. Feischl and T. Tran. The Eddy Current–LLG equations: FEM-BEM coupling and a priori error estimates. *SIAM J. Numer. Anal.*, 55(4):1786–1819, 2017.
[43] J. Fischbacher, A. Kovacs, H. Özelt, M. Gusenbauer, T. Schrefl, L. Exl, D. Givord, N. M. Dempsey, G. Zimanyi, M. Winklhofer, et al. On the limits of coercivity in permanent magnets. *Appl. Phys. Lett.*, 111(7):072404, 2017.
[44] T. Fischbacher, M. Franchin, G. Bordignon, and H. Fangohr. A systematic approach to multiphysics extensions of finite-element-based micromagnetic simulations: Nmag. *IEEE Trans. Magn.*, 43(6):2896–2898, 2007.
[45] D. R. Fredkin and T. R. Koehler. Hybrid method for computing demagnetization fields. *IEEE Trans. Magn.*, 26(2):415–417, 1990.
[46] S. Fu, W. Cui, M. Hu, R. Chang, M. J. Donahue, and V. Lomakin. Finite-difference micromagnetic solvers with the object-oriented micromagnetic framework on graphics processing units. *IEEE Trans. Magn.*, 52(4):1–9, 2016.
[47] C. J. García-Cervera. Numerical micromagnetics: A review. *Bol. Soc. Esp. Mat. Apl. SeMA*, 39:103–135, 2007.
[48] T. L. Gilbert. A Lagrangian formulation of the gyromagnetic equation of the magnetization fields. *Phys. Rev.*, 100:1243, 1955. Abstract only.
[49] W. Hackbusch. A sparse matrix arithmetic based on $\mathcal{H}$-matrices. Part I: Introduction to $\mathcal{H}$-matrices. *Computing*, 62(2):89–108, 1999.
[50] R. Hertel. Micromagnetic simulations of magnetostatically coupled nickel nanowires. *J. Appl. Phys.*, 90(11):5752–5758, 2001.
[51] R. Hertel and A. Kákay. Hybrid finite-element/boundary-element method to calculate oersted fields. *J. Magn. Magn. Mater.*, 369:189–196, 2014.
[52] G. Hrkac, C.-M. Pfeiler, D. Praetorius, M. Ruggeri, A. Segatti, and B. Stiftner. Convergent tangent plane integrators for the simulation of chiral magnetic skyrmion dynamics. preprint, arXiv:1712.03795, 2017.
[53] A. Kakay, E. Westphal, and R. Hertel. Speedup of FEM micromagnetic simulations with graphical processing units. *IEEE Trans. Magn.*, 46(6):2303–2306, 2010.
[54] A. Kovacs, H. Özelt, M.E. Schabes, and T. Schrefl. Numerical optimization of writer and media for bit patterned magnetic recording. *J. Appl. Phys.*, 120(1):013902, 2016.
[55] J. Kraus, C.-M. Pfeiler, D. Praetorius, M. Ruggeri, and B. Stiftner. Iterative solution and preconditioning for the tangent plane scheme in computational micromagnetics. preprint, arXiv:1808.10281, 2018.
[56] H. Kronmüller. General micromagnetic theory. In H. Kronmüller, P. Parkin, J. E. Miltat, and M. R. Scheinfein, editors, *Handbook of Magnetism and Advanced Magnetic Materials*. John Wiley & Sons, Ltd, 2007.
[57] M. Kružík and A. Prohl. Recent developments in the modeling, analysis, and numerics of ferromagnetism. *SIAM Rev.*, 48(3):439–483, 2006.
[58] L. Landau and E. Lifshitz. On the theory of the dispersion of magnetic permeability in ferromagnetic bodies. *Phys. Zeitsch. der Sow.*, 8:153–168, 1935.

[59] K.-N. Le, M. Page, D. Praetorius, and T. Tran. On a decoupled linear FEM integrator for eddy-current-LLG. *Appl. Anal.*, 94(5):1051–1067, 2015.
[60] K.-N. Le and T. Tran. A convergent finite element approximation for the quasi-static Maxwell–Landau–Lifshitz–Gilbert equations. *Comput. Math. Appl.*, 66(8):1389–1402, 2013.
[61] A. Makarov, V. Sverdlov, D. Osintsev, and S. Selberherr. Fast switching in magnetic tunnel junctions with two pinned layers: Micromagnetic modeling. *IEEE Trans. Magn.*, 48(4):1289–1292, 2012.
[62] R. D. McMichael and M. S. Stiles. Magnetic normal modes of nanoelements. *J. Appl. Phys.*, 97(10):10J901, 2005.
[63] J. E. Miltat and M. J. Donahue. Numerical micromagnetics: Finite difference methods. In H. Kronmüller, P. Parkin, J. E. Miltat, and M. R. Scheinfein, editors, *Handbook of Magnetism and Advanced Magnetic Materials*. John Wiley & Sons, Ltd, 2007.
[64] T. Moriya. Anisotropic superexchange interaction and weak ferromagnetism. *Phys. Rev.*, 120(91):91, 1960.
[65] D. Praetorius, M. Ruggeri, and B. Stiftner. Convergence of an implicit-explicit midpoint scheme for computational micromagnetics. *Comput. Math. Appl.*, 75(5), 2018.
[66] A. Prohl. *Computational micromagnetism*. B. G. Teubner, Stuttgart, 2001.
[67] M. Ruggeri. *Coupling and numerical integration of the Landau–Lifshitz–Gilbert equation*. PhD thesis, TU Wien, Institute for Analysis and Scientific Computing, 2016.
[68] J. Sampaio, V. Cros, S. Rohart, A. Thiaville, and A. Fert. Nucleation, stability and current-induced motion of isolated magnetic skyrmions in nanostructures. *Nat. Nanotechnol.*, 8(11):839–844, 2013.
[69] J. Schöberl. NETGEN An advancing front 2D/3D-mesh generator based on abstract rules. *Comput. Vis. Sci.*, 1(1):41–52, 1997.
[70] W. Scholz, J. Fidler, T. Schrefl, D. Suess, R. Dittrich, H. Forster, and V. Tsiantos. Scalable parallel micromagnetic solvers for magnetic nanostructures. *Comput. Mater. Sci.*, 28(2):366–383, 2003.
[71] T. Schrefl, G. Hrkac, S. Bance, D. Suess, O. Ertl, and J. Fidler. Numerical methods in micromagnetics (finite element method). In H. Kronmüller, P. Parkin, J. E. Miltat, and M. R. Scheinfein, editors, *Handbook of Magnetism and Advanced Magnetic Materials*. John Wiley & Sons, Ltd, 2007.
[72] H. Sepehri-Amin, T. Ohkubo, S. Nagashima, M. Yano, T. Shoji, A. Kato, T. Schrefl, and K. Hono. High-coercivity ultrafine-grained anisotropic Nd–Fe–B magnets processed by hot deformation and the Nd–Cu grain boundary diffusion process. *Acta Mater.*, 61(17):6622–6634, 2013.
[73] R. Skomski, H. Zeng, M. Zheng, and D. J. Sellmyer. Magnetic localization in transition-metal nanowires. *Phys. Rev. B*, 62(6):3900, 2000.
[74] J. C. Slonczewski. Current-driven excitation of magnetic multilayers. *J. Magn. Magn. Mat.*, 159(1–2):L1–L7, 1996.
[75] W. Śmigaj, T. Betcke, S. Arridge, J. Phillips, and M. Schweiger. Solving boundary integral problems with BEM++. *ACM Trans. Math. Softw.*, 41(2):6:1–6:40, 2015.
[76] D. Suess, C. Vogler, C. Abert, F. Bruckner, R. Windl, L. Breth, and J. Fidler. Fundamental limits in heat-assisted magnetic recording and methods to overcome it with exchange spring structures. *J. Appl. Phys.*, 117(16):163913, 2015.
[77] A. Thiaville, Y. Nakatani, J. Miltat, and Y. Suzuki. Micromagnetic understanding of current-driven domain wall motion in patterned nanowires. *Europhys. Lett.*, 69(6):990–996, 2005.
[78] A. Vansteenkiste, J. Leliaert, M. Dvornik, M. Helsen, F. Garcia-Sanchez, and B. Van Waeyenberge. The design and verification of MuMax3. *AIP Adv.*, 4(10):107133, 2014.
[79] S. Zhang and Z. Li. Roles of nonequilibrium conduction electrons on the magnetization dynamics of ferromagnets. *Phys. Rev. Lett.*, 93(12):127204, 2004.

College of Engineering, Mathematics and Physical Sciences, University of Exeter, North Park Road, Exeter, UK
E-mail address: g.hrkac@exeter.ac.uk

Faculty of Mathematics, University of Vienna, Oskar-Morgenstern-Platz 1, 1090 Vienna, Austria
E-mail address: michele.ruggeri@univie.ac.at

Institute for Analysis and Scientific Computing, TU Wien, Wiedner Hauptstrasse 8–10, 1040 Vienna, Austria
E-mail address: carl-martin.pfeiler@asc.tuwien.ac.at
E-mail address: bernhard.stiftner@asc.tuwien.ac.at
E-mail address: matthias.hochsteger@tuwien.ac.at
E-mail address: joachim.schoeberl@tuwien.ac.at
E-mail address: dirk.praetorius@asc.tuwien.ac.at

Wolfgang Pauli Institute c/o Faculty of Mathematics, University of Vienna, Oskar-Morgenstern-Platz 1, 1090 Vienna, Austria
E-mail address: lukas.exl@univie.ac.at
E-mail address: norbert.mauser@univie.ac.at