Independent Research Project
Final Report

Eigenmode Analysis Package for the Stable Bloch Point using Computational Micromagnetics

Author:
Arnav Avad

Email: arnav.avad22@imperial.ac.uk

GitHub username: acse-aa3019

Repository: https://github.com/ese-msc-2022/irp-aa3019

Supervisors:
Dr. Marijan Beg
Mr. Martin Lang

MSc in Applied Computational Science and Engineering
Department of Earth Science and Engineering
Imperial College London

September 2023

Abstract

Computational micromagnetics is a well-researched field but lacks open source eigenmode analysis tools, which requires researchers to often step out of the simulation tool environment and develop post-processing scripts themselves. Hence, creating such a tool will allow for more convenient ways of exploring resonant modes. Therefore, in this work, we developed open source eigenmode analysis tools for ringdown and eigenvalue methods and integrated them in UBERMAG, exposing eigenmode simulations to the Python ecosystem. We demonstrate our simulation tools by exploring the eigenmodes of a stable Bloch point, in the system of two stacked discs of different chirality.

Keywords: micromagnetics, eigenmode analysis, ringdown, eigenvalue, Bloch point, **UBERMAG**

Acknowledgement

I would like to take this opportunity to thank my supervisor, Dr Marijan Beg, for his time and effect dedicated to guiding me through this project.

Contents

1 Introduction

Computational simulation software and discretised models are used for the modelling of magnetic phenomena to allow for the study of various materials/structures such as topologically stable quasi-particles called skyrmions [1, 2]. It is widely employed in areas of physics research [3] to materials science [4], spintronics [5] and magnetic device optimisation [6]. Hence, it serves for both academic research and industry application. As technology advances and ever increasingly efficient simulation methods are discovered and software released, the ability to research ever more complex systems will grow. UBERMAG is an example of micromagnetics software used for simulating dynamics and analysis. It allows for a python interface for more convenience in scientific research [7].

1.1 Models

Simulation and analysis of micromagnetic systems requires a defined model. There are two models used; the atomistic and the continuous model. The atomistic model represents individual atoms/ions and their respective magnetic moments within the system. On the contrary, magnetisation density is considered in the continuous model. Therefore, average magnetic moments per unit volume are calculated to represent the system being modelled. Since the continuous model operates with less spatial resolution when compared to the atomistic model, it is more efficient. The reason for using the atomistic model is the fact that it provides an accurate model of atomic level phenomena. Hence, it is useful in modelling small systems such as nanoparticles. On the other hand, the continuous model is computationally efficient and useful for investigating larger systems. Micromagnetic simulation models can be further specified depending on the demands of the researcher and the simulated system.

The energy equations governing the atomistic model are given below as equations 1-4 [8, 9].

$$E_{\text{z}} = -\mu_0 \boldsymbol{\mu} \cdot \mathbf{H} \tag{1}$$

$$E_{\text{an}} = -k(\boldsymbol{\mu} \cdot \mathbf{u})^2 \tag{2}$$

$$E_{\text{ex}} = -J\boldsymbol{\mu}_i \cdot \boldsymbol{\mu}_j \tag{3}$$

$$E_{\text{DMI}} = \pm\mathbf{D} \cdot (\boldsymbol{\mu}_i \times \boldsymbol{\mu}_j) \tag{4}$$

where E, μ, $\mathbf{H}$, $\mathbf{u}$, μ_0, k, J, $\mathbf{D}$ are energy, magnetic moment, external magnetic field, anisotropy axis, magnetic constant, atomistic uniaxial anisotropy energy constant, exchange integral and Dzyaloshinskii vector, respectively.

Note that in equations 1-8 the subscript z, an, ex and DMI refer to the zeeman, anisotropy, exchange and Dzyaloshinskii-Moriya interaction (DMI), respectively.

The energy density equations governing the continuous model are given below as equations 5-8 [8, 9]. The energy can be inferred from the energy density using equation 9 [1].

$$\omega_{\text{z}} = -\mu_0 M_{\text{s}} \mathbf{m} \cdot \mathbf{H} \tag{5}$$

$$\omega_{\text{an}} = -k(\mathbf{m} \cdot \mathbf{u})^2 \tag{6}$$

$$\omega_{\text{ex}} = -A\mathbf{m} \cdot \nabla^2 \mathbf{m} \tag{7}$$

$$\omega_{\text{DMI}} = D\mathbf{m} \cdot (\nabla \times \mathbf{m}) \tag{8}$$

$$E = \int_V \omega \, dV \tag{9}$$

where ω, $\mathbf{m}$, M_{s}, k, A, D are the energy density, magnetisation vector, saturation magnetisation, continuous uniaxial anisotropy energy constant, exchange energy constant and Dzyaloshinskii-Moriya energy constant, respectively.

The continuous model will be the area of focus and, hence, appendix A refers to the derivations to the effective field for each of the energy terms 5-8.

1.2 Analysis Approaches

Depending on the problem being solved, different approaches focus on the various aspects of micromagnetism. Landau-Lifshitz-Gilbert (LLG) equation can be used to model the dynamics of a micromagnetic model shown below as equation 10 and 11 [3, 8–10].

$$\frac{\mathrm{d}\mathbf{m}}{\mathrm{d}t} = -\gamma_0^*(\mathbf{m} \times \mathbf{H}_{\text{eff}}) + \alpha(\mathbf{m} \times \frac{\mathrm{d}\mathbf{m}}{\mathrm{d}t}) \tag{10}$$

$$\frac{\mathrm{d}\mathbf{m}}{\mathrm{d}t} = \frac{-\gamma_0^*}{1+\alpha^2}(\mathbf{m} \times \mathbf{H}_{\text{eff}}) - \frac{\gamma_0^*\alpha}{1+\alpha^2}(\mathbf{m} \times (\mathbf{m} \times \mathbf{H}_{\text{eff}})) \tag{11}$$

where

$$\mathbf{H}_{\text{eff}} = -\frac{1}{\mu_0 M_{\text{s}}} \frac{\delta E}{\delta \mathbf{m}} \tag{12}$$

Note that $\mathbf{H}_{\text{eff}}$ is the effective field, γ_0^* is a modified gyromagnetic ratio and α is Gilbert's damping constant.

In equation 10, the first term is the precession term and the latter is the damping term. Equation 11 can be discretised such that the different magnetisation vectors for each timestep can be recorded throughout the simulation. From these recorded states, the

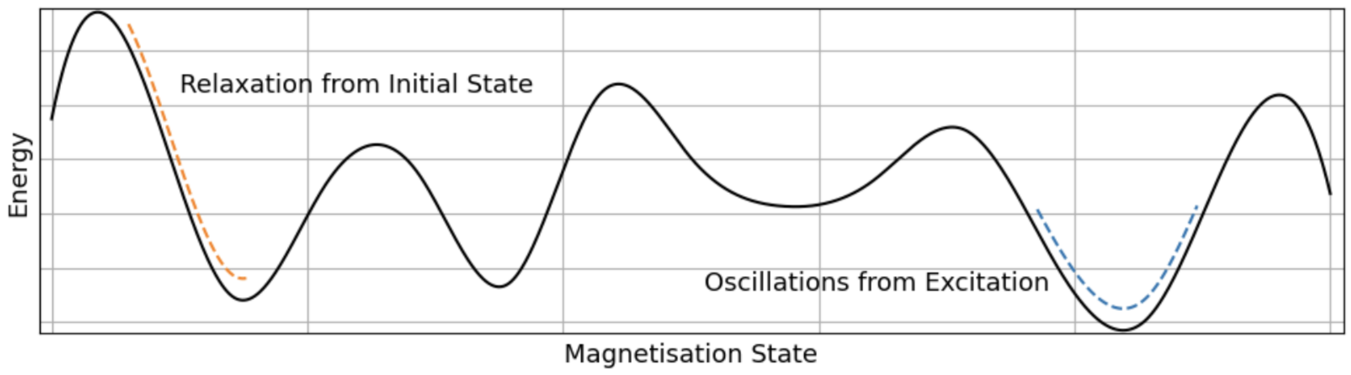


Figure 1: Simplistic diagram showing the dynamics of magnetisations states over the energy landscape of relaxations and oscillations.

data can be analysed for any dynamical phenomena such as the eigenmodes of the system.

Monte Carlo (MC) simulations allows for probabilistic sampling. An example is shown in citation [11] where a system is initialised. The initial state's magnetisation vectors are varied by random amounts with the use of simulated annealing to find the local minimum within the energy landscape of the simulated system. Many minima exist, thus, using MC allows for various stable equilibrated states to be discovered at different modelled temperatures [12, 13]. Figure 1 depicts the idea of relaxation to an equilibrium.

Mean field theory can also be used in micromagnetic simulations [14]. It considers the interaction between the magnetic moments and the effective field from the averaged neighbouring magnetisations. This simplification allows for modelling of larger systems efficiently. An example of mean field theory utilisation is to find local energy minima or stable micromagnetic states within the energy landscape of the system. Using the mean field theory, it is possible to find these equilibrated states at varying temperatures and for different materials similar to MC simulations. But, MC simulations tend to be more computationally expensive [14].

Using the aforementioned methods for simulating micromagnetic systems, numerous phenomena are analysed. The energy landscape, such as the arbitrary example in figure 1, provides a visual representation of the analysis. Relaxation in micromagnetic systems from an initial state to an equilibrium is frequently observed. For example, it is common when searching for stable states throughout the energy landscape [9, 15, 16]. Magnetisation dynamics of the system can also be observed to understand how magnetisation vectors or magnetic vectors vary over time. Resonating states named eigenmodes can be calculated to study the system response to perturbations within the system [1, 9, 17].

Eigenmode analysis is used to find resonant oscillation modes of a micromagnetic system. Currently, two method are used to find this eigenmodes; the ringdown [18] and eigenvalue method [19]. An example of using eigenmode analysis includes the distinction of micromagnetic systems using predetermined eigenmodes from simulations. This has been especially used in the emerging field of skyrmionics [9, 11, 12, 16, 20]. Therefore, one can observe the importance of the field of research.

1.3 Project Objectives

The importance of the project is derived from the need to identify micromagnetic states by using ferromagnetic resonance absorption [21]. Hence when shooting electromagnetic waves of matching frequencies to the eigenmodes, a clear absorption can be observed to identify the state. This experimental method is inexpensive to operate with highly available instruments, creating an computational tool for analysis will shorten the research process. Furthermore, it is useful to know the eigenmodes and their images from a purely physics point of view.

Currently, no open-source code or workflow exists for eigenmode analysis using the ringdown method [18] and the eigenvalue method [19]. However, FinMag is an open source finite element simulation tool with an implemented finite element eigenvalue analysis tool [22]. MAGNUM.NP is also a recently published finite difference simulation framework which uses PYTORCH for the Eigenvalue method [23]. Research groups using computational micromagnetic simulations have different non-standard eigenmode analysis programs. The usual workflow is to write a new script for eigenmode analysis. If the researcher lacks experience in writing well tested and sustainable scientific software with an intuitive interface, a package would facilitate research. Hence, the objectives are to create a python package for eigenmode analysis using the ringdown method and the semi-analytical finite difference eigenvalue method. Further aims are to use this tool for Bloch point eigenmode analysis. Initially demonstrated by Beg et al. [24], a Bloch point is a discontinuity formed by adjacent magnetization vectors. The eigenmodes for this system are outlined within this paper.

This analysis tool provides a smooth workflow for research that requires micromagnetic eigenmode analysis when in combination with UBERMAG [7]. For example, studying topologically stable quasi-particles such as skyrmions would be useful for possible developments as storage devices [2]. Furthermore, this tool should aid in the advancement of spintronics and other related applications. The analysis allows for eigenmode discovery for modelled materials which are able to characterise materials' magnetisation states.

2 Method

The ringdown and eigenvalue methods can be used for eigenmode analysis. It should be noted that both methods vary largely due to the approach taken. The eigenvalue method is semi-analytical, whereas the ringdown method depends on simulation time-series data.

2.1 Ringdown Method

The ringdown method [1, 9, 18, 25] is split into 2 similar methods; spatially averaged and spatially resolved. The difference between the methods are the order of operations which leads to the different properties. However, they are similar in terms of the data required.

The first step for the ringdown method is defining an initial state of a micromagnetic system. The state is allowed to relax using a simulator of choice. For example, UBERMAG [7] is used with the OOMMF backend [26] to reach a local minimum within the energy landscape. Once a relaxed state is reached, the system is excited and simulated with regular discrete timesteps being saved to disc. This timeseries data around this relaxed state is analysed for eigenmodes using Fourier transforms, hence a relaxation and LLG dynamics simulation are required.

The ringdown method requires substantial amount of time due to the fact that two different simulations are required for the analysis. Although it takes a long time, the specific excitation parameters are available for recreation and observation in experiments since the excitation simulated is under the control of the researcher.

2.1.1 Spatially Averaged

The spatially averaged method consists of measuring the change between all magnetisation vectors in the timeseries and the initial magnetisation state. The resulting vectors are averaged for each timestep resulting in one vector for every simulated timestep containing (m_x, m_y, m_z). A Fourier transform is applied to show the time domain data of every component in the frequency domain in which eigenfrequencies are observable as local maxima in the power spectral density graph. The below equation shows how the power spectral density for the spatially averaged method is calculated [1, 18, 25].

$$P_{\mathsf{sa}}(f) = \sum_{k=x,y,z} \left| \sum_{j=1}^{n} \langle \Delta m_k(t_j) \rangle e^{-i2\pi f t_j} \right|^2 \tag{13}$$

2.1.2 Spatially Resolved

On the other hand, the spatially resolved method swaps the order of operations. First the Fourier transform is used to convert from the time domain to the frequency domain for all the nodes of the model. The absolute value of the Fourier coefficients is averaged. Equation 14 shows how the power spectral density is calculated for the spatially resolved method [1, 18, 25]. Since the method uses the Fourier transform for all nodes individually, the method allows for eigenmode imaging.

$$P_{\mathsf{sr}}(f) = \sum_{k=x,y,z} \frac{1}{N} \sum_{i=1}^{N} \left| \sum_{j=1}^{n} m_k(\mathbf{r}_j, t_j) e^{-i2\pi f t_j} \right|^2 \tag{14}$$

2.2 Eigenvalue Method

No excitation is necessary for the semi-analytical Eigenvalue method, proposed by d'Aquino *et al.* [19], since it only uses the initial relaxed equilibrium state magnetisation vectors. Allowing for perpendicular variations, $\mathbf{v}(t)$ is applied to the initial state which results in equation 15 [1, 25]. This initial state with the variations is applied to an undamped LLG equation as shown in equation 16. An eigenvalue problem can then be constructed. Note that a full, explained derivation is written in appendix B.

$$\mathbf{m} = \mathbf{m}_0 + \mathbf{v}(t) \tag{15}$$

$$\frac{\partial \mathbf{m}}{\partial t} = -\gamma_0^*(\mathbf{m} \times \mathbf{H}_{\text{eff}}) \tag{16}$$

$$\frac{\partial}{\partial t}(\mathbf{m}_0 + \mathbf{v}(t)) = -\gamma_0^*(\mathbf{m}_0 + \mathbf{v}(t)) \times \mathbf{H}_{\text{eff}}(\mathbf{m}_0 + \mathbf{v}(t)) \tag{17}$$

Using the fact that $\frac{\partial \mathbf{m}_0}{\partial t} = 0$, $\mathbf{H}_0 \times \mathbf{m}_0 = 0$ and the taylor expansion $\mathbf{H}_{\text{eff}}(\mathbf{m}_0 + \mathbf{v}(t)) = \mathbf{H}_0 + \mathbf{H}'_{\text{eff}}(\mathbf{m}_0) \cdot \mathbf{v}(t) + \mathcal{O}\left(\|\mathbf{v}\|^2\right)$:

$$\frac{\partial}{\partial t}(\mathbf{v}(t)) = -\gamma_0^*(\mathbf{v}(t) \times \mathbf{H}_0 + \mathbf{m}_0 \times (\mathbf{H}'_{\text{eff}}(\mathbf{m}_0)) \cdot \mathbf{v}(t)) \tag{18}$$

Note that $\mathbf{H}_0 = \mathbf{H}_{\text{eff}}(\mathbf{m}_0)$ and all $\mathcal{O}\left(\|\mathbf{v}\|^2\right)$ terms and higher are ignored.

It is known that $\mathbf{H}_0$ can be expressed as a multiple of the magnetisation vectors, $\mathbf{H}_0 = h_0\mathbf{m}_0$,

$$\begin{aligned} \mathbf{v}(t) \times \mathbf{H}_0 &= \mathbf{v}(t) \times h_0\mathbf{m}_0 \\ &= -\mathbf{m}_0 \times h_0\mathbf{v}(t) \end{aligned} \tag{19}$$

$$\begin{aligned} \frac{\partial}{\partial t}(\mathbf{v}(t)) &= \gamma_0^*\mathbf{m}_0 \times \left((h_0 \cdot \mathbb{1} + \mathbf{H}'_{\text{eff}}(\mathbf{m}_0)) \cdot \mathbf{v}(t)\right) \\ &= \gamma_0^*\mathbf{m}_0 \times \mathbf{A}_0 \cdot \mathbf{v}(t) \end{aligned} \tag{20}$$

where $\mathbf{A}_0 = (h_0 \cdot \mathbb{1} + \mathbf{H}'_{\text{eff}}(\mathbf{m}_0)) \cdot \mathbf{v}(t)$ It is known that a cross product can be represented as a matrix transformation of a vector using an skew-symmetric matrix. Equation 20 can be formulated without the explicit cross product as shown below.

$$\frac{\partial}{\partial t}(\mathbf{v}(t)) = \mathbf{A} \cdot \mathbf{v}(t) \tag{21}$$

$$\mathbf{A} = \gamma_0^*\mathbf{\Lambda}(\mathbf{m}_0) \cdot \mathbf{A}_0 \tag{22}$$

$$\tilde{\mathbf{\Lambda}}(\mathbf{m}) = \begin{pmatrix} 0 & -m_3 & m_2 \\ m_3 & 0 & -m_1 \\ -m_2 & m_1 & 0 \end{pmatrix} \tag{23}$$

$$\mathbf{\Lambda}(\mathbf{m}_0) = \begin{pmatrix} \tilde{\mathbf{\Lambda}}(\mathbf{m}_{0,1}) & & \\ & \ddots & \\ & & \tilde{\mathbf{\Lambda}}(\mathbf{m}_{0,N}) \end{pmatrix} \tag{24}$$

From the linear differential equation 21, an exponential ansatz can be used to find a solution, formulating an eigenvalue problem to solve for the angular frequency ω. The ansatz in question is $\mathbf{v}(t) = \text{Re}(\mathbf{v}_0 e^{i\omega t})$. When substituting into the differential equation, it gives $\text{Re}(i\omega\mathbf{v}_0 e^{i\omega t}) = \text{Re}(\mathbf{A} \cdot \mathbf{v}_0 e^{i\omega t})$. Simplifying, gives the final eigenvalue problem to solve:

$$\mathbf{A} \cdot \mathbf{v}_0 = i\omega\mathbf{v}_0 \tag{25}$$

where ω is the angular frequency and $\mathbf{v}_0$ is the eigenmode magnetisation state. The eigenmode frequencies can be calculated by the fact that $f = \frac{\omega}{2\pi}$.

Eigenmode magnetisation dynamics can be observed through the use of equation 26 where $\mathbf{v}_0$ is the eigenvector corresponding to the eigenmode to be observed.

$$\mathbf{m}(t) = \mathbf{m}_0 + \mathbf{v}_0 e^{i\omega t} \tag{26}$$

In comparison to the ringdown method, the eigenvalue method only requires the relaxed state of the system due to its semi-analytical nature. Hence, less time is taken as simulations are time consuming processes. In addition to this, all eigenmodes are found using this method but the excitations required to find the eigenmodes are unavailable. The method does allow for imaging, similar to the spatially resolved ringdown method.

2.2.1 Dimensionality Reduction

The dimensions of the matrices and arrays in the problem can be reduced to 2N rather than 3N because the normal oscillation modes are pointwise orthogonal to the magnetisation state array [19, 25]. Therefore, transforming to a new reference frame using a rotation matrix will allow for this dimensionality reduction, enabling faster computation and more efficient storage utilisation.

The orthogonal vector basis of the reference frame is given as equations 27. Furthermore, the 3D initial reference frame can be transformed from 2D using the matrix shown as equation 28 [19].

$$\begin{aligned} \mathbf{e}_{1,n} &= -\mathbf{m}_{0,n} \times (\mathbf{e}_z \times \mathbf{m}_{0,n}) \\ \mathbf{e}_{2,n} &= \mathbf{e}_z \times \mathbf{m}_{0,n} \\ \mathbf{e}_{3,n} &= \mathbf{m}_{0,n} \end{aligned} \tag{27}$$

$$\mathbf{R}_n = \begin{pmatrix} \mathbf{e}_{1,n}\cdot\mathbf{e}_x & \mathbf{e}_{2,n}\cdot\mathbf{e}_x & \mathbf{e}_{3,n}\cdot\mathbf{e}_x \\ \mathbf{e}_{1,n}\cdot\mathbf{e}_y & \mathbf{e}_{2,n}\cdot\mathbf{e}_y & \mathbf{e}_{3,n}\cdot\mathbf{e}_y \\ \mathbf{e}_{1,n}\cdot\mathbf{e}_z & \mathbf{e}_{2,n}\cdot\mathbf{e}_z & \mathbf{e}_{3,n}\cdot\mathbf{e}_z \end{pmatrix} \quad (28)$$

It is important to note the eigenvectors produced are 2D when solving the eigenvalue problem with the dimensionality reduction. Hence, this array of 2D vectors needs to be transformed by $\mathbf{R}_n^T$ for analysis such as imaging.

3 Python Implementation

The package was developed in the Python programming language along with the NUMPY library [27] for fast array vectorisation and SCIPY library [28] for its functionality homogenous to Numpy arrays such as solving eigenvalue problems. Furthermore, any plotting native to this package was done using MATPLOTLIB [29] while UBERMAG [7] was also used for a variety of reason including producing field objects where necessary. UBERMAG is an integral part of the package to create a smooth workflow between micromagnetic simulation, observation and eigenmode analysis. Python was chosen to support for further development with less experienced programmers in mind. Several sustainability measures were put in place for a smooth development process, like the GitHub workflows to update the package documentation using SPHINX [30] and regular code testing using PYTEST [31].

3.1 Implementation

3.1.1 Package Structure

The UML diagram 2 shows the design of the structure. The entire package has a common abstract class named EIGENMODEMETHODS for template purposes and storing common data required between all the eigenmode analysis methods used. The RINGDOWNMETHOD class and the EIGENVALUE class is derived from this base class. Furthermore, both respective ringdown method classes are derived from the RINGDOWNMETHOD class. The RINGDOWNMETHOD in particular shows how inheritance is leveraged as it contains the bulk of the common helper functions for the ringdown methods such as methods for plotting the time and frequency domain, and finding eigenmodes.

Aside from the main classes, the eigenvalue method required computation of the effective fields of the energy terms being used. Since the OOMMF [26] calculator in UBERMAG [7] offloads the computation to C++ and saves it every time, a set of effective field calculations for various energy terms shown in the UML diagram were implemented in Python by exploiting NUMPY vectorisation. A further class was created to handle the instantiation and calculation of the effective fields with an option to either calculate for the linear terms or for all terms.

3.1.2 Ringdown Method Implementation

The main idea of both ringdown methods was to take a time-series of the magnetisation vectors and Fourier transform this data to analyse it in the frequency domain. Subsequently, each peak would be an eigenmode. Hence, the Fast Fourier transform algorithm provided by SCIPY [28] was used. SCIPY was used instead of NUMPY [27] as it had implemented a version of FFT which is parallelised, increasing the time efficiency. Although the frequent use of NUMPY is observable due to its vectorisation ability. The Python listing below shows an example of how the spatially resolved method was implemented. Note that the EIGENMODE_HELPER is a function to use frequency domain data to find eigenmodes and store them in a PANDAS [32] dataframe. Below is an example of the spatially resolved ringdown method implementation.

```python
def apply(self):
    data = self.driveobj.to_xarray().to_numpy()
    self.ft_sr = scipy.fft.fft(data,axis=0)
    ft_sr = np.copy(self.ft_sr)
    ft_sr = np.log10(np.abs(ft_sr) ** 2)
    np.nan_to_num(ft_sr,copy=False, neginf=0.0,nan=0.0)
    self.psd = np.mean(ft_sr[...,(0, 1, 2)],axis=(1, 2, 3)).T
    self.eigenmode_helper()
```

Listing 1: Excerpt of the spatially resolved method implementation in the eigenmode analysis package.

Specific to the spatially resolved method, the ability to return a single dimensional field of specified components were added. This enabled the magnitude and phase to be plotted.

3.1.3 Eigenvalue Method Implementation

```python
def D(self, v: np.ndarray) -> np.ndarray:
    h = np.empty_like(self.m0, dtype=np.complex128)
    v = v.reshape(self.v_shape).astype(np.complex128)
    Rv = v[:,:,:,(0,)] * self.e1 + v[:,:,:,(1,)] * self.e2
```

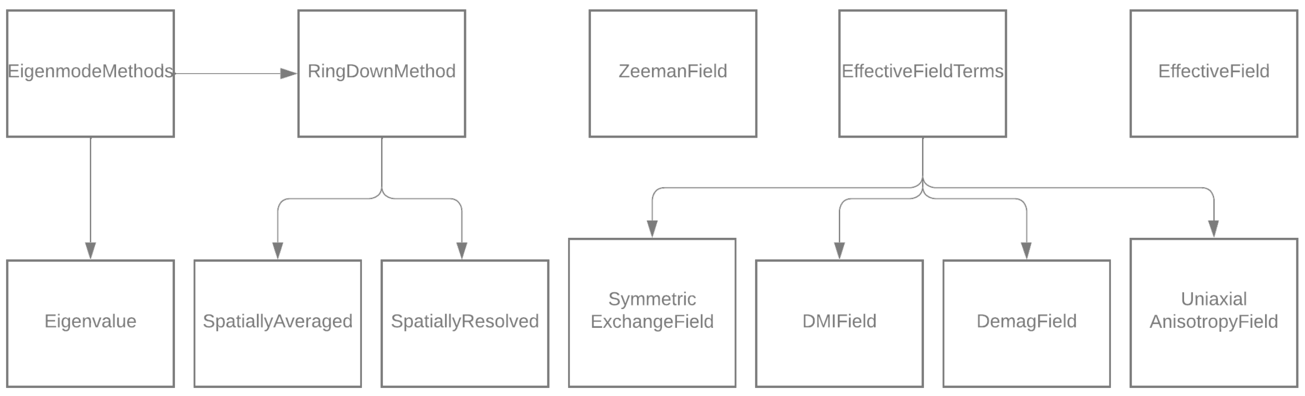


Figure 2: UML Class diagram showing the structure of the python package.

```
    h = self.H_eff.compute_effective_field(Rv.real, linear=True) + (
        1j * self.H_eff.compute_effective_field(Rv.imag, linear=True))
    A = mm.consts.gamma0 * np.cross(self.m0, (self.h0 * Rv) - h)
    return np.stack(
        [(A * self.e1).sum(axis=-1), (A * self.e2).sum(axis=-1)],
        axis=-1,
    ).flatten()
```

Listing 2: Excerpt of the eigenvalue method implementation in the eigenmode analysis package showing the linear operator used in place of an explicit matrix as done in MAGNUM.NP.

D'Aquino's Eigenvalue method [19] is a semi-analytical approach, hence no simulation and time-series is necessary. Instead the effective fields are necessary. Methods for producing the effective fields of each energy terms shown in 5-8 were written using the finite difference method.

In chronological order, the initialisation of the function mainly contained an initialised object for effective field calculations, the drive to extract the relaxed magnetisation state and the set of basis to convert between the 2D and 3D vectors used for memory efficiency.

The 'D' method was used as a linear operator as SCIPY.SPARSE allows for this. In order to solve the eigenvalue problem, the method acted as a linear operator in which it would convert the 2D vector to a 3D vector. It would then calculate the RHS of equation 20. The final step is converting back to a 2D vector for the eigenvalue solver. These calculations are provided in the code listing above.

This linear operator simplifies the code as it removes the need to store large matrices and provides a readable interface. A similar method to MAGNUM.NP [23] was used due to the time-frame of the project for effective field implementations; the demagnetisation tensor was not explicitly implemented so the NUMPY's linear operator [27] was used. All effective field calculations were implemented using NUMPY as UBERMAG [7] (with OOMMFC) saved each field computation increasing the time. However, UBERMAG was used to calculate the demagnetisation tensor and field because the demagnetisation field has a complex implementation.

The apply function simply contained the eigenvalue solver and repackaging of the solution from 2D to 3D vectors and storing the eigenfrequencies in a PANDAS data-frame [32] for analysis. The class in which the method is hosted also provides functionality to visualise magnetisation dynamics.

In addition to the basic implementation of the eigenvalue method, implementations delving into the magnetisation imaging, phase imaging and magnetisation dynamics graphing and animation exist.

3.2 Demo

All the demos[1] shown in appendix C use the FMR (Ferromagnetic Resonance) standard problem [17]. A thin film sample of 120nm $\times$ 120nm $\times$ 10nm is simulated using OOMMF which UBERMAG's backend.

[1]Note that a full demo can be found as a Jupyter notebook on the GitHub repository accompanying the report as shown on the title page.

The energy terms used are the Zeeman energy, demagnetisation energy and the symmetry exchange energy. The necessary constants used for the simulations are shown below:

Magnestisation saturation, $\mathrm{M_s} = 8 \times 10^5 \mathrm{Am^{-1}}$
Gilbert damping $\alpha = 0.008$
Exchange constant, $\mathrm{A} = 1.3 \times 10^{-11} \mathrm{Jm^{-1}}$

For the relaxation or energy minimisation step, the following constants are used for 5ns.

$\mathbf{m}_0 = (0, 0, 1)$
$\mathbf{H}_{\mathrm{zeeman}} = 80 \cdot (1, 0.715, 0) \mathrm{kAm^{-1}}$

For the dynamics stage of the simulation, the LLG equation is used to simulate the oscillations induced from an excitation for 20ns. The sampling is done every 5ps and the initial magnetisation vector array is taken from the previous relaxation.

$\mathbf{H}_{\mathrm{zeeman}} = 80 \cdot (1, 0.7, 0) \mathrm{kAm^{-1}}$

3.3 Validation - FMR (Ferromagnetic Resonance) Standard Problem

For continuation, the same problem as the demo can be used for validation of the developed Python program. The graphs which are shown will only range from 0 to 25GHz as the inaccuracies increase substantially at higher frequencies.

Figure 3 presents the results from the short demo, using both the eigenvalue method and the ringdown methods. The paper where this standard problem was first proposed also used the same methods to create a standard solution with OOMMF [26]. The results outlined [17] suggests that figure 3 are valid due to similarities in both imaging of the eigenmodes and the frequency domain graphs.

The above graphs when compared to the project proposal shows the validity of the python package due to the similarity of the results. In figure 4, all the images are found for all the eigenmodes shown in the proposal. These first 15 eigenmodes match the eigenmodes that were calculated from the FMR proposal paper. Note the regular grid mesh used was $24 \times 24 \times 2$.

4 Bloch Point Eigenmode Analysis

As mentioned previously, the Bloch point [24] is a discontinuity of adjacent magnetisation vectors which was demonstrated using two disc on opposing chirality rather than using single chirality throughout the entire material. It was simulated by using exchange, demagnetisation and Dzyaloshinskii–Moriya interaction (DMI) energy terms. In which, two discs of diameter 150nm and thickness of 30nm (10nm of negative DMI and 20nm of positive coefficients) had opposite DMI chiralities; the constant, D, has opposing signs.

The first step was to simulate a Bloch point, using the paper, 'Stable and manipulable Bloch point' [24]. It consisted of using UBERMAG with the OOMMF calculator. Two sub-regions were defined to account for the different DMI constants. The initial system was given magnetisation vectors, $\hat{e}_{\mathrm{z}}$. A minimisation driver was used to minimise the total energy of the system. The following material parameters were used[2]:

Magnetisation saturation, $\mathrm{M_s} = 3.84 \times 10^5 \mathrm{Am^{-1}}$
Exchange constant, $\mathrm{A} = 8.78 \times 10^{-12} \mathrm{Jm^{-1}}$
DMI constant[3], $\mathrm{D} = \pm 1.58 \times 10^{-3} \mathrm{Jm^{-2}}$

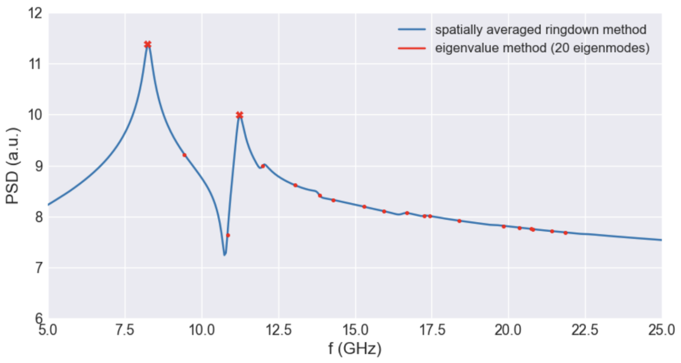
4.1 Method

To analyse the eigenmodes of the simulated Bloch point, the Python tool was used. For the Ringdown method, this required another simulation with defined excitations to take place. A paper published by Beg et al. as 'Dynamics of skyrmionic states in confined helimagnetic nanostructures' [1] was used as inspiration. A cardinal sine otherwise known as a sinc function was used for the excitation for all the eigenmodes below a defined fequency, f_c, equally. The excitations used were $40\mathbf{e}_{\mathrm{plane}}\mathrm{kAm^{-1}}$. After this, the system was allowed to relax for 2ns to reduce any accumulated noise from the excitation. Another simulation was followed for sampling using the Nyquist frequency.

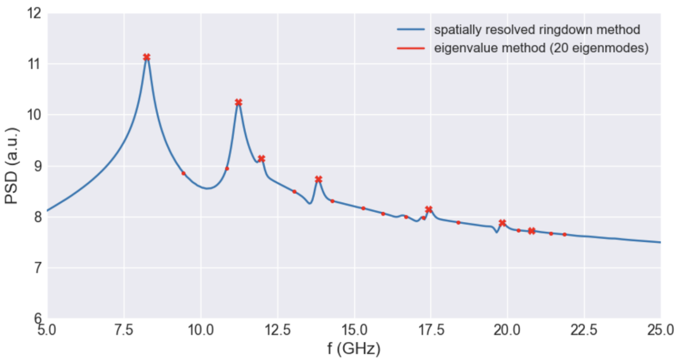
For the simulation time driver, an α of 0.008 was used with a γ_0^* of 2.211×10^5. As for the simulation, the steps are clearly outlined below:

1. Add Sinc Zeeman Field with 0.25ns shift and 100GHz frequency.

[2]The entire Bloch point creation simulation can be found in UBERMAG documentation examples online.
[3]$\pm$ is used to demonstrate the idea each disc will either have a positive or negative DMI constant.



(a) Spatially averaged method power spectral density x component and eigenvalue method eigenmodes.



(b) Spatially resolved method power spectral density x component and eigenvalue method eigenmodes.

Figure 3: Graphs to show the power spectral density vs. the frequency from the 20ns oscillation simulation produced by the python package. The red crosses signify the eigenmodes which match between the eigenvalue method and the specified ringdown method. The red dots signify eigenmodes which do not match.

2. Simulate for 0.5ns to excite all eigenmodes below 100GHz
3. Remove Sinc Zeeman Field
4. Simulate for 2ns to enable the system to relax for noise reduction
5. Simulate for 25ns to sample the oscillations 5000 times

4.2 Results

Using the ringdown method, the power spectral densities (PSDs) of the Bloch point are shown in figure 5 for the in-plane excitation and for the out-of-plane excitation. Both spatially resolved method and spatially averaged methods were used [18]. Furthermore, the spatially resolved method enables imaging of the Bloch point eigenmodes. Hence, in figure 6, some of the eigenmodes have been plotted.

Both the phase and magnitude imaging plots can be used to infer the dynamics of the Bloch point eigenmode. Only the first two eigenmodes for each of the in-plane and out-of-plane excitation are analysed for dynamics. For the in-plane excitation these are $f_0 = 3.68$GHz and $f_1 = 8.60$GHz while for the out-of-plane excitation these are $f_2 = 9.56$GHz and $f_3 = 14.60$GHz. Since the Bloch point is synthesised using two stacked disc of opposing chiralities, it is vital to observe the top and bottom of the plane being observed as well as different planes. As a preliminary observation, all the eigenmodes observed seem to gyrate.

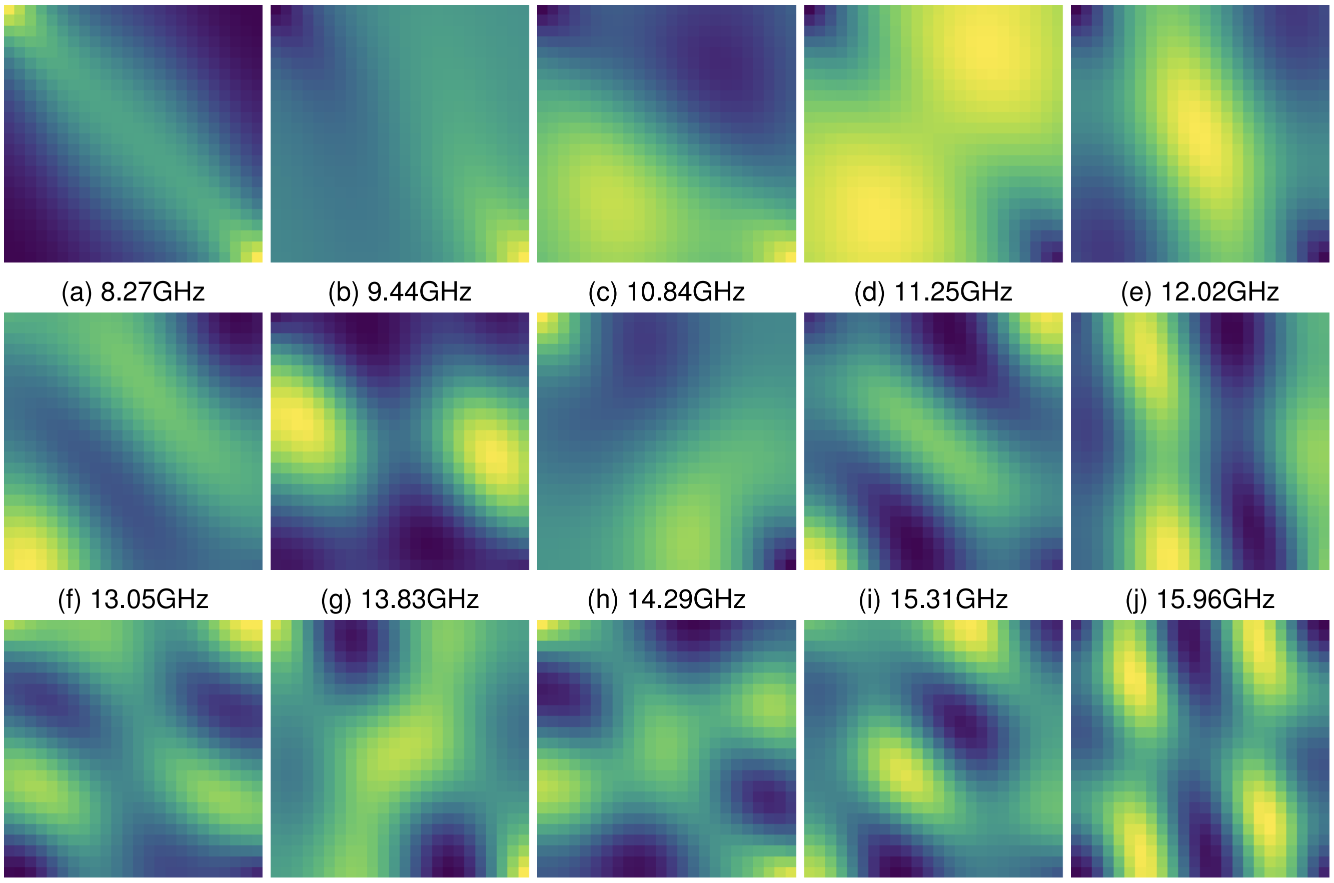
In terms of all the excitations, the phase diagrams shown in figure 6 can be used to infer the dynamics of the eigenmodes. For all the eigenmodes observed, they seems to gyrate around a central point. Depending on the depth of the xy-plane observed, different depths lag behind.

When observing the f_0 x-component, seems to spin with the centre point less defined at the bottom of the disc. The z-component is analogous to a vortex. For f_1, the z-component swirls towards to centre, although the swirl is clockwise at the bottom and anticlockwise at the top. While at the Bloch point depth, an inner circle is separated from the outer circle with the inside left in-phase with the outside right and vice versa. The x-component shows the inside to be out-of-phase with the outside where a swirl forms observing the top surface.

For the out-of-plane excitation, f_2 and f_3 eigenmodes consistently show a phase offset when observing the same point at different depths. For the x-component of f_2 eigenmode, opposite side are out-of-phase. For the f_3 eigenmode, the x-components shows an inner circular domain and an outer domain similar to the f_1 eigenmode. The left inner domain is in phase with the right outer domain and is out-of-phase with the other domains.

5 Discussion

The Python package consists of two tools for eigenmode analysis; one using the ringdown method and the other using the eigenvalue method. Therefore, the two method together provides a wide range of functionality to the researcher.



(a) 8.27GHz (b) 9.44GHz (c) 10.84GHz (d) 11.25GHz (e) 12.02GHz

(f) 13.05GHz (g) 13.83GHz (h) 14.29GHz (i) 15.31GHz (j) 15.96GHz

(k) 16.71GHz (l) 17.25GHz (m) 17.46GHz (n) 18.40GHz (o) 19.84GHz

Figure 4: First 15 eigenmodes detected as shown in the FMR standard problem proposal paper [17] which represents the y components. The finite difference eigenvalue method was used from the package.

5.1 Eigenmode Analysis Package

Numerous benefits and drawbacks underscore the project. One would aim to reduce the number of drawbacks though under the time constraints provided makes it increasingly difficult. In terms of advantages, both methods exist in harmony as outlined above. This allows for a combination of the two analysis techniques for better visualisation and observation using the eigenvalue method while also providing necessary excitations for the recreation of magnetisation states in a lab using the ringdown methods. For example, one can produce simulation data for the ringdown method, observe the specific eigenmode, use the eigenvalue method to find the specific eigenmode and observe the magnetisation dynamics clearly.

On the other hand, due to producing both tools, it reduced the time left for other tasks. This meant a more accurate implementation of the eigenvalue method using finite element methods, as done in Finmag [22], could not be produced although it would have been useful. Python allows for trivial development for scientific purposes. This enables less time to be spent on programming but rather the micromagnetics research at hand. It also avoids limited manipulation of data as the interface is in Python and not a graphical user interface. In addition to this, the python package was implemented with UBERMAG [7] in mind. Due to this integration and the fact that UBERMAG's packages have many features, a more fluid workflow is provided and imaging is made easier. For examples, the subpackage DISCRETISEDFIELD can be used for visulation and MICROMAGNETICDATA can be used to read .omf files.

This eigenmode analysis Python package provides a seamless workflow using various methodologies with UBERMAG [7] integration. This amalgamation of computational efficient practices through the use of standard NUMPY [27] vectorisation and python's readability will facilitate micromagnetics research. The idea is to promote relaxed magnetisation state eigenmode discovery for use in different fields.

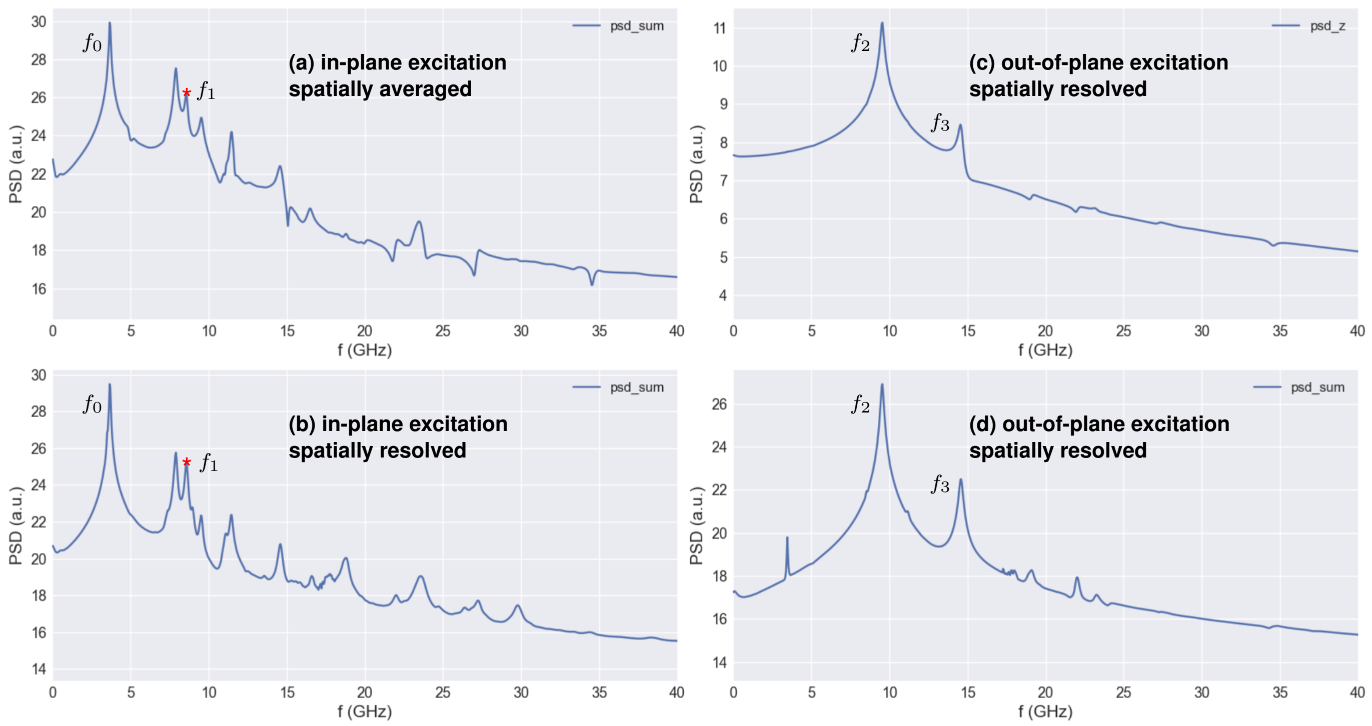


Figure 5: The power spectral densities (PSDs) resulting from the spatially averaged and spatially resolved ringdown methods using an in-plane and out-of-plane excitation in the form of a sinc function. The full method is noted above. The PSD of only the z component is used in figure c due to the negative values of the x and y component PSDs.

There are improvements which can be made with an abundance of time. These vary from ease of use to improvements in efficiency. For example, the PBC demagnetisation field implementation [33] is complex, an initial thought is to allow for more functionality through a periodic boundary conditions (PBC) option.

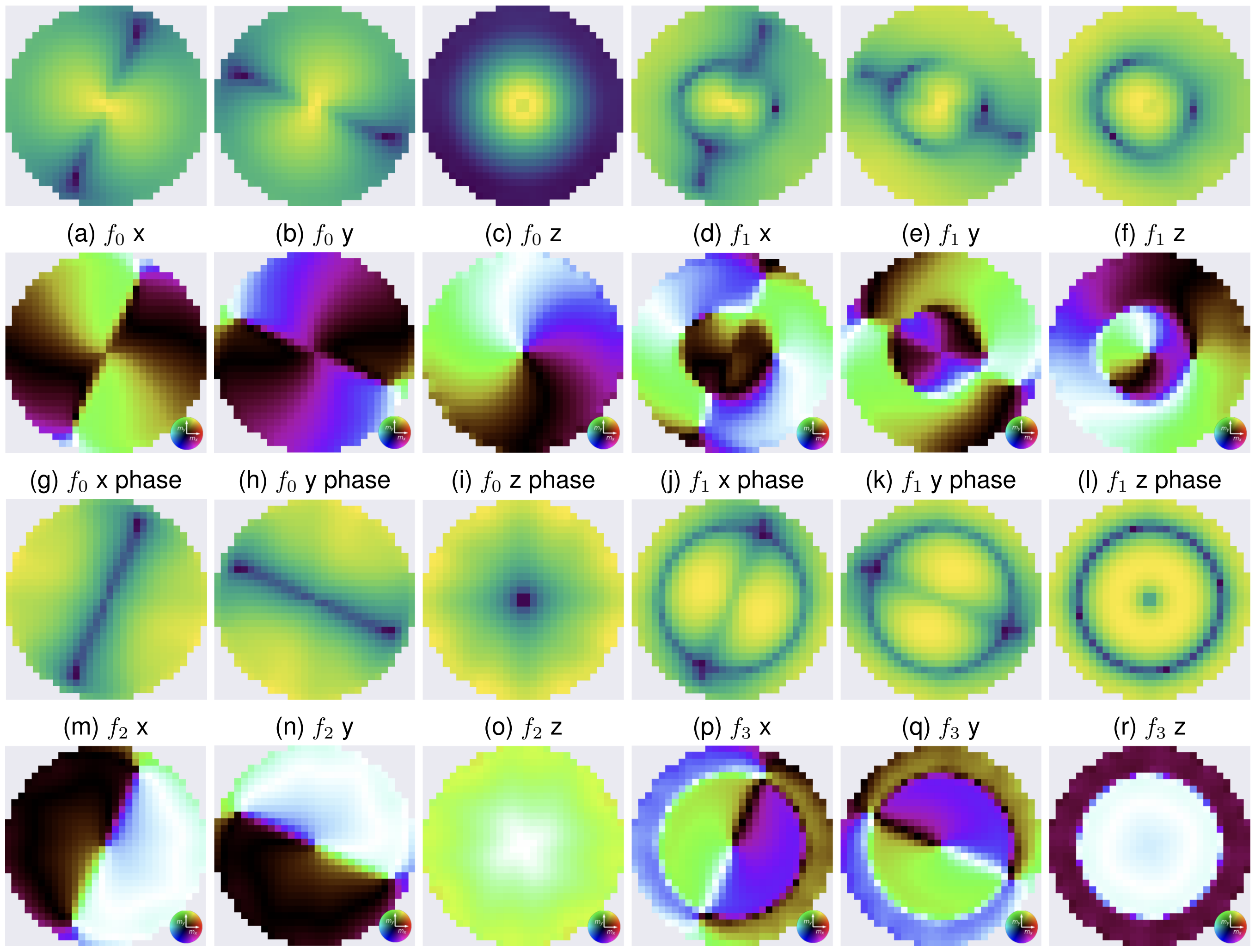
Further to the PBC implementation, varying material symmetries can be modelled. An issue with UBERMAG is saving the computed effective field of each term which slows the program, therefore the computation of the terms had to be coded in Python using NUMPY. Hence, only 5 classes for each term were created including the Zeeman, symmetric exchange, uniaxial anisotropy, demagnetisation and DMI terms. For example, adding DMI terms of varying material symmetries such as D_{2d} and C_{nv} [34] will add more functionality.

In addition to adding more core functionality, using a compiled programming language such as C++ with a python interface, like JOOMMF [35] or UBERMAG, will increase the speed and efficiency of the program. Another method with the same aim of efficiency is to implement parallelism by using libraries such as NUMBA [36]. Although SCIPY [28] and NUMPY [27] have some parallelised methods, explicit parallelism is beneficial. For example, the effective field calculations can be parallelised.

5.2 Bloch Point Eigenmodes

The Bloch point eigenmodes were identified using both the spatially averaged and spatially resolved ringdown methods with the resolved method allowing for imaging of such eigenmodes. Although, the ringdown methods required an excitation to be simulated using UBERMAG's [7] backend which was OOMMF in this case. Therefore, since this excitation definition was necessary, not all eigenmodes are visible. Only those eigenmodes sufficiently excited by the defined external field with the cardinal sine function are visible.

Further to this issue, according to the d'Aquino et al. [19], the operators for the linear fields presented (exchange, demagnetisation and uniaxial anisotropy field) were symmetric. And, the resulting operator, $\mathbf{A}_0$, in equation 20 is positive definite. Although, as the DMI terms is included with opposing chiralities within one system, the positive definiteness vanishes. Hence, the conclusions made in the paper using this fact cannot be assumed to be true in this


(a) f_0 x (b) f_0 y (c) f_0 z (d) f_1 x (e) f_1 y (f) f_1 z

(g) f_0 x phase (h) f_0 y phase (i) f_0 z phase (j) f_1 x phase (k) f_1 y phase (l) f_1 z phase

(m) f_2 x (n) f_2 y (o) f_2 z (p) f_3 x (q) f_3 y (r) f_3 z

(s) f_2 x phase (t) f_2 y phase (u) f_2 z phase (v) f_3 x phase (w) f_3 y phase (x) f_3 z phase

Figure 6: First two bloch point eigenmodes visualised for each seperate x-, y- and z-component using the spatially resolved ringdown method for the in-plane and out-of-plane excitation, respectively. Both the power spectral density and phase is presented. The xy plane is shown. Note that $f_0 = 3.68$GHz and $f_1 = 8.60$GHz is for in-plane excitations. $f_2 = 9.56$GHz and $f_3 = 14.60$GHz are for out-of-plane excitations.

case. For example there may be a case where the determined angular frequencies may have a non-zero imaginary component.

The eigenvalue method was unfortunately not used for eigenmode analysis of the Bloch point as the results seemed to stray with noticeable error. For this reason, the implementation of the method will require more testing for more complex systems outside of the predetermined time constraints of the project.

6 Conclusion

Eigenmode analysis is able to provide valuable insights into micromagnetic relaxed states. It allows for eigenmode discovery which can aid in fields such as spintronics. Hence, this paper outlines a standard open-source eigenmode analysis tool using finite difference discretisation and the validity of the results through the ferromagnetic resonance standard problem.

The initial aims consisting of implementing the ringdown method [18] compatible with UBERMAG [7] and OOMMF [26] for both spatially averaged and spatially resolved methods were satisfied. Furthermore, using finite difference discretisation, the semi-analytical method for the eigenvalue method outlined by d'Aquino et al. [19] was also implemented using NUMPY vectorisation within Python; a researcher friendly language for further development as required. Both these implementation, especially the ringdown methods, will reduce the need for the use of scripting for eigenmode analysis.

UBERMAG was actively used to provide a maintained interface to the OOMMF simulator and extended functionality for system definitions, file reading and plotting and so on. Therefore, creating a package in conjunction to UBERMAG provides a smooth workflow between simulation, observation and analysis.

Furthermore, an investigation into the eigenmodes of the Bloch point was done using the ringdown method. The results were outlined as well as the imaging was completed as shown in figure 6. In terms of the Bloch point, the next step would be to use the eigenvalue method for eigenmode detection and imaging as well as to experimentally test for the eigenmodes and observe the FMR absorption data for possible state identification.

References

[1] Marijan Beg, Maximilian Albert, Marc-Antonio Bisotti, David Cortés-Ortuño, Weiwei Wang, Rebecca Carey, Mark Vousden, Ondrej Hovorka, Chiara Ciccarelli, Charles S. Spencer, Christopher H. Marrows, and Hans Fangohr. Dynamics of skyrmionic states in confined helimagnetic nanostructures. *Phys. Rev. B / arXiv:1604.08347*, 95:014433, Jan 2017. https://link.aps.org/doi/10.1103/PhysRevB.95.014433.

[2] Shijiang Luo and Long You. Skyrmion devices for memory and logic applications. *APL Materials*, 9(5):050901, 05 2021. https://pubs.aip.org/aip/apm/article-pdf/doi/10.1063/5.0042917/13757078/050901_1_online.pdf.

[3] Josef Fidler and Thomas Schrefl. Micromagnetic modelling - the current state of the art. *Journal of Physics D: Applied Physics*, 33(15):R135, aug 2000. https://dx.doi.org/10.1088/0022-3727/33/15/201.

[4] Carl-Martin Pfeiler, Michele Ruggeri, Bernhard Stiftner, Lukas Exl, Matthias Hochsteger, Gino Hrkac, Joachim Schöberl, Norbert J. Mauser, and Dirk Praetorius. Computational micromagnetics with commics. *Computer Physics Communications*, 248:106965, mar 2020. https://doi.org/10.1016%2Fj.cpc.2019.106965.

[5] Joseph Barker and Unai Atxitia. A review of modelling in ferrimagnetic spintronics. *Journal of the Physical Society of Japan / arXiv:2102.11004*, 90(8):081001, aug 2021. https://doi.org/10.7566%2Fjpsj.90.081001.

[6] Jin-Kyu Byun, Iana Volvach, and Vitaliy Lomakin. Fast optimal design of micromagnetic devices using fastmag and distributed evolutionary algorithm. *IEEE Transactions on Magnetics*, 52:1–1, 09 2016. https://www.researchgate.net/publication/301916221_Fast_Optimal_Design_of_Micromagnetic_Devices_Using_FastMag_and_Distributed_Evolutionary_Algorithm.

[7] Marijan Beg, Martin Lang, and Hans Fangohr. Ubermag: Towards more effective micromagnetic workflows. *IEEE Transactions on Magnetics*, 58(2):1–5, 2022. https://ieeexplore.ieee.org/document/9427472.

[8] Claas Abert. Micromagnetics and spintronics: models and numerical methods. *The European Physical Journal B / arXiv:1810.12365*, 92(6), jun 2019. https://doi.org/10.1140%2Fepjb%2Fe2019-90599-6.

[9] Marijan Beg. *Skyrmionic states in confined helimagnetic nanostructures*. PhD thesis, University of Southampton, April 2016. https://eprints.soton.ac.uk/402969/.

[10] T.L. Gilbert. A phenomenological theory of damping in ferromagnetic materials. *IEEE Transactions on Magnetics*, 40(6):3443–3449, 2004. https://ieeexplore.ieee.org/document/1353448.

[11] Juan C. Criado, Peter D. Hatton, Sebastian Schenk, Michael Spannowsky, and Luke A. Turnbull. Simulating magnetic antiskyrmions on the lattice. *arXiv:2109.15020*, 2021. https://arxiv.org/abs/2109.15020.

[12] Stefan Buhrandt and Lars Fritz. Skyrmion lattice phase in three-dimensional chiral magnets from monte carlo simulations. *Phys. Rev. B / arXiv:1304.6580*, 88:195137, Nov 2013. https://link.aps.org/doi/10.1103/PhysRevB.88.195137.

[13] 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 / arXiv:2106.05593*, 130(16), 10 2021. https://doi.org/10.1063/5.0059745.

[14] Ondrej Hovorka and Timothy J. Sluckin. A computational mean-field model of interacting non-collinear classical spins. *arXiv:2007.12777*, 2020. https://arxiv.org/abs/2007.12777.

[15] Rok Dittrich Rasovic, Thomas Schrefl, D. Suess, Werner Scholz, H Forster, and J. Fidler. A path method for finding energy barriers and minimum energy paths in complex micromagnetic systems. *Journal of Magnetism and Magnetic Materials*, 250:12–19, 09 2002. https://www.academia.edu/60388355/A_path_method_for_finding_energy_barriers_and_minimum_energy_paths_in_complex_micromagnetic_systems.

[16] Pavel F. Bessarab, Valery M. Uzdin, and Hannes Jónsson. Method for finding mechanism and activation energy of magnetic transitions, applied to skyrmion and antivortex annihilation. *Computer Physics Communications / arXiv:1502.05065*, 196:335–347, nov 2015. https://doi.org/10.1016%2Fj.cpc.2015.07.001.

[17] Alexander Baker, Marijan Beg, Gregory Ashton, Maximilian Albert, Dmitri Chernyshenko, Weiwei Wang, Shilei Zhang, Marc-Antonio Bisotti, Matteo Franchin, Chun Lian Hu, Robert Stamps, Thorsten Hesjedal, and Hans Fangohr. Proposal of a micromagnetic standard problem for ferromagnetic resonance simulations. *Journal of Magnetism and Magnetic Materials*, 421:428–439, 2017. https://www.sciencedirect.com/science/article/pii/S0304885316307545.

[18] Robert McMichael and Mark Stiles. Magnetic normal modes of nanoelements. *Journal of Applied Physics*, 97, 05 2005. https://www.researchgate.net/publication/228648040_Magnetic_normal_modes_of_nanoelements.

[19] Massimiliano d'Aquino, Claudio Serpico, Giovanni Miano, and Carlo Forestiere. A novel formulation for the numerical computation of magnetization modes in complex micromagnetic systems. *Journal of Computational Physics*, 228(17):6130–6149, 2009. https://www.sciencedirect.com/science/article/pii/S0021999109002642.

[20] Bhartendu Satywali, Volodymyr P. Kravchuk, Liqing Pan, M. Raju, Shikun He, Fusheng Ma, A. P. Petrović, Markus Garst, and Christos Panagopoulos. Microwave resonances of magnetic skyrmions in thin film multilayers. *Nature Communications*, 12(1), mar 2021. https://doi.org/10.1038%2Fs41467-021-22220-1.

[21] Charles Kittel. On the theory of ferromagnetic resonance absorption. *Phys. Rev.*, 73:155–161, Jan 1948. https://link.aps.org/doi/10.1103/PhysRev.73.155.

[22] Marc-Antonio Bisotti, Marijan Beg, Weiwei Wang, Maximilian Albert, Dmitri Chernyshenko, David Cortés-Ortuño, Ryan A. Pepper, Mark Vousden, Rebecca Carey, Hagen Fuchs, Anders Johansen, Gabriel Balaban, Leoni Breth, Thomas Kluyver, and Hans Fangohr. FinMag: finite-element micromagnetic simulation tool, April 2018. https://doi.org/10.5281/zenodo.1216011.

[23] Florian Bruckner, Sabri Koraltan, Claas Abert, and Dieter Suess. magnum.np – a pytorch based gpu enhanced finite difference micromagnetic simulation framework for high level development and inverse design, 2023.

[24] Marijan Beg, Ryan A. Pepper, David Cortés-Ortuño, Bilal Atie, Marc-Antonio Bisotti, Gary Downing, Thomas Kluyver, Ondrej Hovorka, and Hans Fangohr. Stable and manipulable bloch point. *Scientific Reports*, 9(1), may 2019. https://doi.org/10.1038%2Fs41598-019-44462-2.

[25] Maximilian Albert. *Domain wall dynamics and resonant modes of magnetic nanostructures*. PhD thesis, University of Southampton, September 2016. https://eprints.soton.ac.uk/413582/.

[26] D. Porter and M. Donahue. Standard problems in micromagnetics. *World Scientific, Electrostatic and Magnetic Phenomena*, pages 285–324, aug 2020. https://math.nist.gov/~MDonahue/pubs/stdprobs-world_scientific-2020-draft_20180531.pdf.

[27] Charles R. Harris, K. Jarrod Millman, Stéfan J. van der Walt, Ralf Gommers, Pauli Virtanen, David Cournapeau, Eric Wieser, Julian Taylor, Sebastian Berg, Nathaniel J. Smith, Robert Kern, Matti Picus, Stephan Hoyer, Marten H. van Kerkwijk, Matthew Brett, Allan Haldane, Jaime Fernández del Río, Mark Wiebe, Pearu Peterson, Pierre Gérard-Marchant, Kevin Sheppard, Tyler Reddy, Warren Weckesser,

Hameer Abbasi, Christoph Gohlke, and Travis E. Oliphant. Array programming with NumPy. *Nature*, 585(7825):357–362, September 2020. https://doi.org/10.1038/s41586-020-2649-2.

[28] Pauli Virtanen, Ralf Gommers, Travis E. Oliphant, Matt Haberland, Tyler Reddy, David Cournapeau, Evgeni Burovski, Pearu Peterson, Warren Weckesser, Jonathan Bright, Stéfan J. van der Walt, Matthew Brett, Joshua Wilson, K. Jarrod Millman, Nikolay Mayorov, Andrew R. J. Nelson, Eric Jones, Robert Kern, Eric Larson, C J Carey, İlhan Polat, Yu Feng, Eric W. Moore, Jake VanderPlas, Denis Laxalde, Josef Perktold, Robert Cimrman, Ian Henriksen, E. A. Quintero, Charles R. Harris, Anne M. Archibald, Antônio H. Ribeiro, Fabian Pedregosa, Paul van Mulbregt, and SciPy 1.0 Contributors. SciPy 1.0: Fundamental Algorithms for Scientific Computing in Python. *Nature Methods*, 17:261–272, 2020. https://www.nature.com/articles/s41592-019-0686-2.

[29] J. D. Hunter. Matplotlib: A 2d graphics environment. *Computing in Science & Engineering*, 9(3):90–95, 2007.

[30] Georg Brandl. Sphinx documentation. *URL http://sphinx-doc. org/sphinx. pdf*, 2021.

[31] Holger Krekel, Bruno Oliveira, Ronny Pfannschmidt, Floris Bruynooghe, Brianna Laugher, and Florian Bruhin. pytest x.y, 2004. https://github.com/pytest-dev/pytest.

[32] The pandas development team. pandas-dev/pandas: Pandas, February 2020. https://doi.org/10.5281/zenodo.3509134.

[33] K M Lebecki, M J Donahue, and M W Gutowski. Periodic boundary conditions for demagnetization interactions in micromagnetic simulations. *Journal of Physics D: Applied Physics*, 41(17):175005, aug 2008. https://dx.doi.org/10.1088/0022-3727/41/17/175005.

[34] David Cortés-Ortuño, Marijan Beg, Vanessa Nehruji, Leoni Breth, Ryan Pepper, Thomas Kluyver, Gary Downing, Thorsten Hesjedal, Peter Hatton, Tom Lancaster, Riccardo Hertel, Ondrej Hovorka, and Hans Fangohr. Proposal for a micromagnetic standard problem for materials with dzyaloshinskii–moriya interaction. *New Journal of Physics*, 20(11):113015, nov 2018. https://dx.doi.org/10.1088/1367-2630/aaea1c.

[35] Marijan Beg and Hans Fangohr. Jupyter object oriented micromagnetic framework (joommf) example notebooks, Dec 2017. https://nanohub.org/resources/joommf.

[36] Siu Kwan Lam, Antoine Pitrou, and Stanley Seibert. Numba: A llvm-based python jit compiler. In *Proceedings of the Second Workshop on the LLVM Compiler Infrastructure in HPC*, pages 1–6, 2015.

Appendices

A Effective Field Derivations

$$\mathbf{H}_{\text{eff}} = -\frac{1}{\mu_0 M_{\text{s}}} \frac{\delta E}{\delta \mathbf{m}} \tag{A.1}$$

A.1 Zeeman

$$\begin{aligned} \omega_{\text{z}} &= -\mu_0 M_{\text{s}} \mathbf{m} \cdot \mathbf{H} \\ &= -\mu_0 M_{\text{s}} (\mathsf{m}_x H_x + \mathsf{m}_y H_y + \mathsf{m}_z H_z) \end{aligned} \tag{A.2}$$

$$\mathbf{H}_{\text{eff}} = h_x \hat{\mathbf{x}} + h_y \hat{\mathbf{y}} + h_z \hat{\mathbf{z}} \tag{A.3}$$

$$\begin{aligned} h_x &= -\frac{1}{\mu_0 M_{\text{s}}} \left(\frac{\partial \omega_{\text{z}}}{\partial m_x} - \frac{\partial}{\partial x} \frac{\partial \omega_{\text{z}}}{(\partial \frac{\partial m_x}{\partial x})} - \frac{\partial}{\partial y} \frac{\partial \omega_{\text{z}}}{(\partial \frac{\partial m_x}{\partial y})} - \frac{\partial}{\partial z} \frac{\partial \omega_{\text{z}}}{(\partial \frac{\partial m_x}{\partial z})} \right) \\ &= -\frac{1}{\mu_0 M_{\text{s}}} (-\mu_0 M_{\text{s}} H_x) \\ &= H_x \end{aligned} \tag{A.4}$$

$$\begin{aligned} h_y &= -\frac{1}{\mu_0 M_{\text{s}}} \left(\frac{\partial \omega_{\text{z}}}{\partial m_y} - \frac{\partial}{\partial x} \frac{\partial \omega_{\text{z}}}{(\partial \frac{\partial m_y}{\partial x})} - \frac{\partial}{\partial y} \frac{\partial \omega_{\text{z}}}{(\partial \frac{\partial m_y}{\partial y})} - \frac{\partial}{\partial z} \frac{\partial \omega_{\text{z}}}{(\partial \frac{\partial m_y}{\partial z})} \right) \\ &= -\frac{1}{\mu_0 M_{\text{s}}} (-\mu_0 M_{\text{s}} H_y) \\ &= H_y \end{aligned} \tag{A.5}$$

$$\begin{aligned} h_z &= -\frac{1}{\mu_0 M_{\text{s}}} \left(\frac{\partial \omega_{\text{z}}}{\partial m_z} - \frac{\partial}{\partial x} \frac{\partial \omega_{\text{z}}}{(\partial \frac{\partial m_z}{\partial x})} - \frac{\partial}{\partial y} \frac{\partial \omega_{\text{z}}}{(\partial \frac{\partial m_z}{\partial y})} - \frac{\partial}{\partial z} \frac{\partial \omega_{\text{z}}}{(\partial \frac{\partial m_z}{\partial z})} \right) \\ &= -\frac{1}{\mu_0 M_{\text{s}}} (-\mu_0 M_{\text{s}} H_z) \\ &= H_z \end{aligned} \tag{A.6}$$

$$\Rightarrow \mathbf{H}_{\text{eff}} = \mathbf{H} \tag{A.7}$$

A.2 Symmetric Exchange

$$\begin{aligned} \omega_{\text{ex}} &= -A \mathbf{m} \cdot \nabla^2 \mathbf{m} \\ &= A \left((\nabla m_x)^2 + (\nabla m_y)^2 + (\nabla m_z)^2 \right) \\ &= A \left(\left(\frac{\partial m_x}{\partial x} + \frac{\partial m_x}{\partial y} + \frac{\partial m_x}{\partial z} \right)^2 + \left(\frac{\partial m_y}{\partial x} + \frac{\partial m_y}{\partial y} + \frac{\partial m_y}{\partial z} \right)^2 + \left(\frac{\partial m_z}{\partial x} + \frac{\partial m_z}{\partial y} + \frac{\partial m_z}{\partial z} \right)^2 \right) \end{aligned} \tag{A.8}$$

$$\mathbf{H}_{\text{eff}} = h_x \hat{\mathbf{x}} + h_y \hat{\mathbf{y}} + h_z \hat{\mathbf{z}} \tag{A.9}$$

$$\begin{aligned} h_x &= -\frac{1}{\mu_0 M_{\text{s}}} \left(\frac{\partial \omega_{\text{ex}}}{\partial m_x} - \frac{\partial}{\partial x} \frac{\partial \omega_{\text{ex}}}{(\partial \frac{\partial m_x}{\partial x})} - \frac{\partial}{\partial y} \frac{\partial \omega_{\text{ex}}}{(\partial \frac{\partial m_x}{\partial y})} - \frac{\partial}{\partial z} \frac{\partial \omega_{\text{ex}}}{(\partial \frac{\partial m_x}{\partial z})} \right) \\ &= -\frac{1}{\mu_0 M_{\text{s}}} \left(-2A \left(\frac{\partial m_x}{\partial x} + \frac{\partial m_x}{\partial y} + \frac{\partial m_x}{\partial z} \right) \right) \\ &= \frac{2A}{\mu_0 M_{\text{s}}} \nabla^2 m_x \end{aligned} \tag{A.10}$$

$$\begin{aligned} h_y &= -\frac{1}{\mu_0 M_\mathsf{s}}\left(\frac{\partial\omega_\mathsf{ex}}{\partial m_y} - \frac{\partial}{\partial x}\frac{\partial\omega_\mathsf{ex}}{(\partial\frac{\partial m_y}{\partial x})} - \frac{\partial}{\partial y}\frac{\partial\omega_\mathsf{ex}}{(\partial\frac{\partial m_y}{\partial y})} - \frac{\partial}{\partial z}\frac{\partial\omega_\mathsf{ex}}{(\partial\frac{\partial m_y}{\partial z})}\right) \\ &= -\frac{1}{\mu_0 M_\mathsf{s}}\left(-2A\left(\frac{\partial m_y}{\partial x} + \frac{\partial m_y}{\partial y} + \frac{\partial m_y}{\partial z}\right)\right) \\ &= \frac{2A}{\mu_0 M_\mathsf{s}}\nabla^2 m_y \end{aligned} \tag{A.11}$$

$$\begin{aligned} h_z &= -\frac{1}{\mu_0 M_\mathsf{s}}\left(\frac{\partial\omega_\mathsf{ex}}{\partial m_z} - \frac{\partial}{\partial x}\frac{\partial\omega_\mathsf{ex}}{(\partial\frac{\partial m_z}{\partial x})} - \frac{\partial}{\partial y}\frac{\partial\omega_\mathsf{ex}}{(\partial\frac{\partial m_z}{\partial y})} - \frac{\partial}{\partial z}\frac{\partial\omega_\mathsf{ex}}{(\partial\frac{\partial m_z}{\partial z})}\right) \\ &= -\frac{1}{\mu_0 M_\mathsf{s}}\left(-2A\left(\frac{\partial m_z}{\partial x} + \frac{\partial m_z}{\partial y} + \frac{\partial m_z}{\partial z}\right)\right) \\ &= \frac{2A}{\mu_0 M_\mathsf{s}}\nabla^2 m_z \end{aligned} \tag{A.12}$$

$$\Rightarrow \mathbf{H}_\mathsf{eff} = \frac{2A}{\mu_0 M_\mathsf{s}}\nabla^2 \mathbf{m} \tag{A.13}$$

A.3 Uniaxial Anisotropy

$$\begin{aligned} \omega_\mathsf{an} &= -k\left(\mathbf{m}\cdot\mathbf{u}\right)^2 \\ &= -k\left(m_x u_x + m_y u_y + m_z u_z\right)^2 \end{aligned} \tag{A.14}$$

$$\mathbf{H}_\mathsf{eff} = h_x\hat{\mathbf{x}} + h_y\hat{\mathbf{y}} + h_z\hat{\mathbf{z}} \tag{A.15}$$

$$\begin{aligned} h_x &= -\frac{1}{\mu_0 M_\mathsf{s}}\left(\frac{\partial\omega_\mathsf{an}}{\partial m_x} - \frac{\partial}{\partial x}\frac{\partial\omega_\mathsf{an}}{(\partial\frac{\partial m_x}{\partial x})} - \frac{\partial}{\partial y}\frac{\partial\omega_\mathsf{an}}{(\partial\frac{\partial m_x}{\partial y})} - \frac{\partial}{\partial z}\frac{\partial\omega_\mathsf{an}}{(\partial\frac{\partial m_x}{\partial z})}\right) \\ &= -\frac{2k}{\mu_0 M_\mathsf{s}} m_x u_x \end{aligned} \tag{A.16}$$

$$\begin{aligned} h_y &= -\frac{1}{\mu_0 M_\mathsf{s}}\left(\frac{\partial\omega_\mathsf{an}}{\partial m_y} - \frac{\partial}{\partial x}\frac{\partial\omega_\mathsf{an}}{(\partial\frac{\partial m_y}{\partial x})} - \frac{\partial}{\partial y}\frac{\partial\omega_\mathsf{an}}{(\partial\frac{\partial m_y}{\partial y})} - \frac{\partial}{\partial z}\frac{\partial\omega_\mathsf{an}}{(\partial\frac{\partial m_y}{\partial z})}\right) \\ &= -\frac{2k}{\mu_0 M_\mathsf{s}} m_y u_y \end{aligned} \tag{A.17}$$

$$\begin{aligned} h_z &= -\frac{1}{\mu_0 M_\mathsf{s}}\left(\frac{\partial\omega_\mathsf{an}}{\partial m_z} - \frac{\partial}{\partial x}\frac{\partial\omega_\mathsf{an}}{(\partial\frac{\partial m_z}{\partial x})} - \frac{\partial}{\partial y}\frac{\partial\omega_\mathsf{an}}{(\partial\frac{\partial m_z}{\partial y})} - \frac{\partial}{\partial z}\frac{\partial\omega_\mathsf{an}}{(\partial\frac{\partial m_z}{\partial z})}\right) \\ &= -\frac{2k}{\mu_0 M_\mathsf{s}} m_z u_z \end{aligned} \tag{A.18}$$

$$\Rightarrow \mathbf{H}_\mathsf{eff} = -\frac{2k}{\mu_0 M_\mathsf{s}}\begin{pmatrix} m_x u_x \\ m_y u_y \\ m_z u_z \end{pmatrix} \tag{A.19}$$

A.4 Dzyaloshinskii–Moriya interaction (DMI)

$$\begin{aligned} \omega_\mathsf{an} &= D\mathbf{m}\cdot\left(\nabla\times\mathbf{m}\right) \\ &= D\begin{pmatrix} m_x \\ m_y \\ m_z \end{pmatrix}\cdot\begin{pmatrix} \frac{\partial m_z}{\partial y} - \frac{\partial m_y}{\partial z} \\ \frac{\partial m_x}{\partial z} - \frac{\partial m_z}{\partial x} \\ \frac{\partial m_y}{\partial x} - \frac{\partial m_x}{\partial y} \end{pmatrix} \\ &= D\left(m_x\left(\frac{\partial m_z}{\partial y} - \frac{\partial m_y}{\partial z}\right) + m_y\left(\frac{\partial m_x}{\partial z} - \frac{\partial m_z}{\partial x}\right) + m_z\left(\frac{\partial m_y}{\partial x} - \frac{\partial m_x}{\partial y}\right)\right) \end{aligned} \tag{A.20}$$

$$\mathbf{H}_\mathsf{eff} = h_x\hat{\mathbf{x}} + h_y\hat{\mathbf{y}} + h_z\hat{\mathbf{z}} \tag{A.21}$$

$$\begin{aligned}
h_x &= -\frac{1}{\mu_0 M_\mathsf{s}} \left(\frac{\partial \omega_\mathsf{dmi}}{\partial m_x} - \frac{\partial}{\partial x} \frac{\partial \omega_\mathsf{dmi}}{(\partial \frac{\partial m_x}{\partial x})} - \frac{\partial}{\partial y} \frac{\partial \omega_\mathsf{dmi}}{(\partial \frac{\partial m_x}{\partial y})} - \frac{\partial}{\partial z} \frac{\partial \omega_\mathsf{dmi}}{(\partial \frac{\partial m_x}{\partial z})} \right) \\
&= D \left(\frac{\partial m_z}{\partial y} - \frac{\partial m_y}{\partial z} + \frac{\partial m_z}{\partial y} - \frac{\partial m_y}{\partial z} \right) \\
&= 2D \left(\frac{\partial m_z}{\partial y} - \frac{\partial m_y}{\partial z} \right)
\end{aligned} \tag{A.22}$$

$$\begin{aligned}
h_y &= -\frac{1}{\mu_0 M_\mathsf{s}} \left(\frac{\partial \omega_\mathsf{dmi}}{\partial m_y} - \frac{\partial}{\partial x} \frac{\partial \omega_\mathsf{dmi}}{(\partial \frac{\partial m_y}{\partial x})} - \frac{\partial}{\partial y} \frac{\partial \omega_\mathsf{dmi}}{(\partial \frac{\partial m_y}{\partial y})} - \frac{\partial}{\partial z} \frac{\partial \omega_\mathsf{dmi}}{(\partial \frac{\partial m_y}{\partial z})} \right) \\
&= D \left(\frac{\partial m_x}{\partial z} - \frac{\partial m_z}{\partial x} + \frac{\partial m_x}{\partial z} - \frac{\partial m_z}{\partial x} \right) \\
&= 2D \left(\frac{\partial m_x}{\partial z} - \frac{\partial m_z}{\partial x} \right)
\end{aligned} \tag{A.23}$$

$$\begin{aligned}
h_z &= -\frac{1}{\mu_0 M_\mathsf{s}} \left(\frac{\partial \omega_\mathsf{dmi}}{\partial m_z} - \frac{\partial}{\partial x} \frac{\partial \omega_\mathsf{dmi}}{(\partial \frac{\partial m_z}{\partial x})} - \frac{\partial}{\partial y} \frac{\partial \omega_\mathsf{dmi}}{(\partial \frac{\partial m_z}{\partial y})} - \frac{\partial}{\partial z} \frac{\partial \omega_\mathsf{dmi}}{(\partial \frac{\partial m_z}{\partial z})} \right) \\
&= D \left(\frac{\partial m_y}{\partial x} - \frac{\partial m_x}{\partial y} + \frac{\partial m_y}{\partial x} - \frac{\partial m_x}{\partial y} \right) \\
&= 2D \left(\frac{\partial m_y}{\partial x} - \frac{\partial m_x}{\partial y} \right)
\end{aligned} \tag{A.24}$$

$$\Rightarrow \mathbf{H}_\mathsf{eff} = -\frac{2D}{\mu_0 M_\mathsf{s}} \left(\nabla \times \mathbf{m} \right) \tag{A.25}$$

B Eigenvalue Method

Below, a full derivation of the eigenvalue method is shown [19, 25].

$$\frac{\partial \mathbf{m}}{\partial t} = \frac{-\gamma_0^*}{1+\alpha^2}(\mathbf{m} \times \mathbf{H}_{\text{eff}}) - \frac{\gamma_0^* \alpha}{1+\alpha^2}(\mathbf{m} \times (\mathbf{m} \times \mathbf{H}_{\text{eff}})) \tag{B.1}$$

An undamped LLG equation can be produced by setting the Gilbert damping as $\alpha = 0$.

$$\frac{\partial \mathbf{m}}{\partial t} = -\gamma_0^*(\mathbf{m} \times \mathbf{H}_{\text{eff}}) \tag{B.2}$$

We can consider $\mathbf{m}_0$ as the equilibrium magnetisation state and, hence, substitute an infinitesimal perturbation into the conservative LLG equation. This perturbation is $\mathbf{m}_0 + \mathbf{v}(t)$. It is vital that $\mathbf{v}(t) \perp \mathbf{m}_0$ and that $\|\mathbf{m}_0\| = 1$.

$$\frac{\partial}{\partial t}(\mathbf{m}_0 + \mathbf{v}(t)) = -\gamma_0^*(\mathbf{m}_0 + \mathbf{v}(t)) \times \mathbf{H}_{\text{eff}}(\mathbf{m}_0 + \mathbf{v}(t)) \tag{B.3}$$

Using Taylor series to approximate the effective field component we can get the following.

$$\begin{aligned} \mathbf{H}_{\text{eff}}(\mathbf{m}_0 + \mathbf{v}(t)) &= \mathbf{H}_{\text{eff}}(\mathbf{m}_0) + \mathbf{H}'_{\text{eff}}(\mathbf{m}_0) \cdot \mathbf{v}(t) + \mathcal{O}\left(\|\mathbf{v}\|^2\right) \\ &\approx \mathbf{H}_{\text{eff}}(\mathbf{m}_0) + \mathbf{H}'_{\text{eff}}(\mathbf{m}_0) \cdot \mathbf{v}(t) \end{aligned} \tag{B.4}$$

We know that the initial magnetisation state, $\mathbf{m}_0$, is not time dependant, therefore $\frac{\partial \mathbf{m}_0}{\partial t} = 0$ is used to simplify the formulation.

$$\frac{\partial}{\partial t}(\mathbf{m}_0 + \mathbf{v}(t)) = \frac{\partial}{\partial t}\mathbf{v}(t) \tag{B.5}$$

By substituting these realisations, we get the following shown below. It is vital to understand that the last equation does not include as $\mathbf{m}_0 \times \mathbf{H}_{\text{eff}}(\mathbf{m}_0) = \mathbf{0}$ according to Brown's equation. No torque acts on the magnetisation state when in equilibrium.

$$\begin{aligned} \frac{\partial}{\partial t}(\mathbf{m}_0 + \mathbf{v}(t)) &= -\gamma_0^*(\mathbf{m}_0 + \mathbf{v}(t)) \times \mathbf{H}_{\text{eff}}(\mathbf{m}_0 + \mathbf{v}(t)) \\ \frac{\partial}{\partial t}\mathbf{v}(t) &= -\gamma_0^*(\mathbf{m}_0 + \mathbf{v}(t)) \times (\mathbf{H}_{\text{eff}}(\mathbf{m}_0) + \mathbf{H}'_{\text{eff}}(\mathbf{m}_0) \cdot \mathbf{v}(t)) \\ &= -\gamma_0^* \left(\mathbf{m}_0 \times \mathbf{H}_{\text{eff}}(\mathbf{m}_0) + \mathbf{v}(t) \times \mathbf{H}_{\text{eff}}(\mathbf{m}_0) + \mathbf{m}_0 \times \mathbf{H}'_{\text{eff}}(\mathbf{m}_0) \cdot \mathbf{v}(t) + \mathcal{O}\left(\|\mathbf{v}\|^2\right)\right) \\ &= -\gamma_0^* \left(\mathbf{v}(t) \times \mathbf{H}_{\text{eff}}(\mathbf{m}_0) + \mathbf{m}_0 \times \mathbf{H}'_{\text{eff}}(\mathbf{m}_0) \cdot \mathbf{v}(t) + \mathcal{O}\left(\|\mathbf{v}\|^2\right)\right) \\ &\approx -\gamma_0^* \left(\mathbf{v}(t) \times \mathbf{H}_{\text{eff}}(\mathbf{m}_0) + \mathbf{m}_0 \times \mathbf{H}'_{\text{eff}}(\mathbf{m}_0) \cdot \mathbf{v}(t)\right) \end{aligned} \tag{B.6}$$

Furthermore, the effective field at equilibrium can be represented as $\mathbf{H}_{\text{eff}}(\mathbf{m}_0) = h_0\mathbf{m}_0$ where $h_0 = \|\mathbf{H}_{\text{eff}}(\mathbf{m}_0)\|$. The vectors, $\mathbf{m}_0$, have unit length and are parallel to $\mathbf{H}_{\text{eff}}(\mathbf{m}_0)$.

$$\begin{aligned} \mathbf{v}(t) \times \mathbf{H}_0 &= \mathbf{v}(t) \times h_0\mathbf{m}_0 \\ &= -\mathbf{m}_0 \times h_0\mathbf{v}(t) \end{aligned} \tag{B.7}$$

Substitution of equation B.7 allows for further simplification to produce, more or less, the final problem. Although, we can represent the cross product as a dot product as shown further below in equations B.9 and B.10.

$$\begin{aligned} \frac{\partial}{\partial t}\mathbf{v}(t) &= -\gamma_0^* \left(-\mathbf{m}_0 \times h_0\mathbf{v}(t) + \mathbf{m}_0 \times (\mathbf{H}'_{\text{eff}}(\mathbf{m}_0) \cdot \mathbf{v}(t)\right) \\ &= \gamma_0^*\mathbf{m}_0 \times \left((h_0 \cdot \mathbb{1} - \mathbf{H}'_{\text{eff}}(\mathbf{m}_0)) \cdot \mathbf{v}(t)\right) \\ &= \gamma_0^*\mathbf{m}_0 \times \mathbf{A}_0 \cdot \mathbf{v}(t) \\ &= \mathbf{A} \cdot \mathbf{v}(t) \end{aligned} \tag{B.8}$$

where $\mathbf{A}_0 = h_0 \cdot \mathbb{1} + \mathbf{H}'_{\text{eff}}(\mathbf{m}_0)$ and $\mathbf{A} = \gamma_0^* \mathbf{\Lambda}(\mathbf{m}_0) \cdot \mathbf{A}_0$ The cross product can be represented as a matrix transformation using a skew-symmetric matrix.

$$\tilde{\mathbf{\Lambda}}(\mathbf{m}) = \begin{pmatrix} 0 & -m_3 & m_2 \\ m_3 & 0 & -m_1 \\ -m_2 & m_1 & 0 \end{pmatrix} \tag{B.9}$$

$\mathbf{\Lambda}(\mathbf{m}_0)$ represents the larger discretisation.

$$\mathbf{\Lambda}(\mathbf{m}_0) = \begin{pmatrix} \tilde{\mathbf{\Lambda}}(\mathbf{m}_{0,1}) & & \\ & \ddots & \\ & & \tilde{\mathbf{\Lambda}}(\mathbf{m}_{0,N}) \end{pmatrix} \tag{B.10}$$

Since, the eigenvalue problem is disguised in a differential equation, an ansatz is used. $\mathbf{v}(t) = \text{Re}(\mathbf{v}_0 e^{i\omega t})$ is the ansatz in question since $\mathbf{v}_0 \in \mathbb{C}$. Note that ω and f is the angular frequency and frequency, respectively.

$$\begin{aligned} \frac{\partial}{\partial t}(\mathbf{v}(t)) &= \mathbf{A} \cdot \mathbf{v}(t) \\ \frac{\partial}{\partial t}(\text{Re}(\mathbf{v}_0 e^{i\omega t})) &= \mathbf{A} \cdot \text{Re}(\mathbf{v}_0 e^{i\omega t}) \\ \text{Re}(i\omega \mathbf{v}_0 e^{i\omega t}) &= \text{Re}(\mathbf{A} \cdot \mathbf{v}_0 e^{i\omega t}) \\ \mathbf{A} \cdot \mathbf{v}_0 &= i\omega \mathbf{v}_0 \\ &\text{or} \\ \mathbf{A} \cdot \mathbf{v}_0 &= 2i\pi f \mathbf{v}_0 \end{aligned} \tag{B.11}$$

In addition to solving the eigenvalue problem, the magnetisation dynamics can be calculated by the following.

$$\mathbf{m}(t) = \mathbf{m}_0 + \mathbf{v}_0 e^{i\omega t} \tag{B.12}$$

C Demo

The spatially averaged ringdown method is shown below.

```python
import micromagneticdata as md
from eigenanalysis import SpatiallyAveraged

dirname = "path/to/directory"
drive = md.Data(name="nameofsimulation", dirname=dirname)[drivenumber]

SA = SpatiallyAveraged(drive)
SA.apply()

# plot time domain
SA.plot_time_domain(lines=["x", "y"])
# plot frequency domain
SA.plot_freq_domain(lines=["x", "y"])
# show all found eigenmodes as pandas dataframe
SA.eigenmodes_df
```

The spatially resolved ringdown method is very similar to use but also has extra functionality for imaging eigenmodes.

```python
import micromagneticdata as md
import discretisedfield as df
from eigenanalysis import SpatiallyResolved

dirname = "path/to/directory"
drive = md.Data(name="nameofsimulation", dirname=dirname)[drivenumber]

SR = SpatiallyResolved(drive)
SR.apply()

# plot time domain
SR.plot_time_domain(lines=["x", "y"])
# plot frequency domain
SR.plot_freq_domain(lines=["x", "y"])
# show all found eigenmodes as pandas dataframe
SR.eigenmodes_df

# first argument - which PSD to use
# second argument - which eigenmode
# third argument - which component
freq, field = SR.eigenmode_1d_field('y', 3, 'y')
print('Eigenfrequency: ', round(freq, 2), 'GHz')
field.plane('z').mpl.scalar(colorbar=False)

# for plotting the phase of the eigenmode
freq, phase = SR.eigenmode_1d_field_phase('x', 3, 'x')
phase.plane('z').mpl.lightness(
    ax=plt.gca(),
    colorwheel_args=dict(width=0.75, height=0.75),
    colorwheel_xlabel=r"$m_x$",
    colorwheel_ylabel=r"$m_y$",
)
```

A demo of the much faster method to find all eigenmodes, the eigenvalue method, is shown below.

```python
import micromagneticdata as md
import discretisedfield as df
from eigenanalysis import Eigenvalue

# another way of defining the data
drive = md.Drive('nameofsimulation', drivenumber, 'path/to/directory')

```

```python
# define system
lx = ly = 120e-9  # x and y dimensions of the sample(m)
lz = 10e-9  # sample thickness (m)
dx = dy = dz = 5e-9  # discretisation in x, y, and z directions (m)

Ms = 8e5  # saturation magnetisation (A/m)
A = 1.3e-11  # exchange energy constant (J/m)
H = 8e4 * np.array([0.81345856316858023, 0.58162287266553481, 0.0])
alpha = 0.008  # Gilbert damping
gamma0 = 2.211e5

mesh = df.Mesh(p1=(0, 0, 0), p2=(lx, ly, lz), cell=(dx, dy, dz))

system = mm.System(name='stdprobfmr')

system.energy = mm.Exchange(A=A) + mm.Demag() + mm.Zeeman(H=H)

EigenMethod = Eigenvalue(drive, system)
# k is the number of eigenmodes to be found
# tol is the tolerance between eigenvalues for the arnoldi iteration method
EigenMethod.apply(k=20, tol=1e-6)

# shows all found eigenmodes
EigenMethod.eigenmodes_df

# return ubermag field object with its respective eigenmode frequency
freq, field = Eigen.eigenmode_1d_field(5, 'y')
print('Eigenfrequency: ', round(freq, 2), 'GHz')
field.plane('z').mpl.scalar(colorbar=False)

# method to plot the eigenmode phase
freq, phase = Eigen.eigenmode_1d_phase(eigen_index=6, component='z')
phase.plane('z').mpl.lightness(
    ax=plt.gca(),
    colorwheel_args=dict(width=0.75, height=0.75),
    colorwheel_xlabel=r"$m_x$",
    colorwheel_ylabel=r"$m_y$",
)

# method to plot the magnetisation dynamics of a specific eigenmode
EigenMethod.magnetisation_dynamics_1D(1, "x", timesteps=150, imaginary=False)
```