A general model of MnSi-like helical magnets

by

© **Kyle Hall**

A thesis submitted to the School of Graduate Studies in partial fulfillment of the requirements for the degree of Master of Science.

Department of Physics and Physical Oceanography
Memorial University

September 2020

St. John's, Newfoundland and Labrador, Canada

Abstract

MnSi and other magnets belonging to the B20 designation are known to assume exotic magnetic structures with subtle features. Of these, the most notable is the appearance of long-wavelength helical structures. Previous analyses of these materials considered oft-used and non-specific models to describe these systems. I will present a general, classical model with only nearest neighbour exchange interactions constructed through symmetry considerations. This model is complete up to the determination of the relative strengths of the coupling constants and the inclusion of other interactions. Comparison to other models will reveal a general relationship between this model and those used in previous analyses. Further comparison with experimentally observed features is used to produce magnetic order parameters of the structure and a relationship between their complex values and structure observables.

Also presented are the results of computational simulations using the Effective Field Method. These simulations are conducted with specific anisotropic and Zeeman interactions introduced to the model. Periodic boundary conditions are not used to maintain the incommensurate helical structure. The results of these simulations are analyzed to extract several lattice structure parameters and the action of individual exchange constants is considered. Additionally, the relation between this model and others is discussed and the introduction of isolated skyrmions is observed. Finally, the results of preliminary simulations with applied magnetic fields oriented along the helix wavevector are presented. These final results demonstrate the appearance of a conical phase and illuminate the effect of specific anisotropic terms. The critical field B_{C2}, *i.e.*, the critical field between the conical and field-induced ferromagnetic states, and the relationship between canting and field strength are also reported.

Lay summary

Magnetic materials with a similar structure to manganese silicide (MnSi) are known to exhibit magnetic structures – *i.e.*, the ordered arrangement of individual magnetic moments at distinct points – which are nuanced and complex. These materials are the subject of a large number of experimental and theoretical studies due to the potential uses and understanding that these structures could provide. Previous analyses of these materials considered oft-used and non-specific models, attempting to describe the wide range of behaviours exhibited by the full class of MnSi-like crystals.

I will present a general, classical model considering only a subset of interactions that are invariant within the symmetry of these lattices. This model will be used to better describe the observed phases and make predictions of the behaviours of this material. Further, I will present the results of computational simulations using the Effective Field Method – a method used to determine low-energy states of a system. These simulations will be analyzed to view the reaction of the system to model parameters. From this, I will describe how one may tune model parameters to produce desired physical features.

Acknowledgements

I would first like to thank my supervisor, Dr. Stephanie Curnoe. This thesis is a direct result of her patience, openness, and support. The guidance she conferred to me has been invaluable. I cannot overstate my appreciation of her involvement in my Master's program as a mentor, colleague, and friend.

I would also like to extend my gratitude to Memorial University of Newfoundland, the Department of Physics and Physical Oceanography, the School of Graduate Studies, and all those who are associated with them. The environment provided to me has been understanding, welcoming, and encouraging. Specific thanks are given to Dr. Martin Plumer, Dr. Ivan Saika-Voivod, and Dr. Byron Southern, each of whom I have consulted on multiple topics, and have offered helpful advice.

Further, I must express appreciation towards my friends and family. I have been fortunate to be surrounded by wonderful people throughout my life and work.

Finally, I would like to thank NSERC for their generous funding that supported me throughout this program.

Table of contents

List of tables

List of figures

List of symbols

α	The deviation of the phase shift between two neighbouring layers.
$\mathbf{B}$	The applied magnetic field.
$\mathbf{D}_{ij}$	The Dzyaloshinskii-Moriya vector representing the direction and magnitude of the Dzyaloshinskii-Moriya interaction.
D	The magnitude of the Dzyaloshinskii-Moriya vector components when oriented along a cubic diagonal.
$F_{i,j}$	The jth magnetic order parameter of a given wavevector $\mathbf{k}$ and associated with the irreducible representation i.
γ	The angle out of the plane defined by the [111] vector.
$\mathbf{H}_i$	The effective field, due to all interactions, at site i.
$\mathscr{J}_\alpha^\beta H_\alpha^\beta$	A term in the free energy. $\mathscr{J}_\alpha^\beta$ is the exchange constant and H_α^β is a symmetry-invariant nearest neighbour interaction.
J	The magnitude of the simple Heisenberg interaction.
$S_{i,n}^\alpha$	Magnetic moment component along α associated with sublattice number i in unit cell n.
$S_{i_{ijl}}^\alpha$	Magnetic moment component along α associated with sublattice number i in unit cell $\mathbf{n} + \{i, j, l\}$.
ϵ	A constant complex phase factor of $\exp\left(\frac{-2\pi i}{3}\right)$. The conjugate and square are equivalent $\epsilon^2 = \epsilon^* = \exp\left(\frac{2\pi i}{3}\right)$.

List of abbreviations

DMI	Dzyaloshinskii-Moriya Interaction
EFM	Effective Field Method
NN	Nearest Neighbour
OP	Order Parameter
PEFM/SEFM	Progressive/Stepped Effective Field Method

Chapter 1

Magnetic Crystals & Manganese Silicide

1.1 B20 crystals and space group $P2_13$

Manganese silicide (MnSi), or sometimes manganese monosilicide to differentiate from other $MnSi_x$ materials, is a magnetic diatomic compound with equal quantity of each ion. Compounds of this type (labelled MnSi-like) belong to the B20 Strukturbericht designation, which includes materials isostructural to the prototype compound FeSi. B20 crystals are notable for their exotic magnetic structures. Materials belonging to this group have long been known to exhibit long-wavelength helimagnetic ordering [1, 2]. Other exotic phases, specifically skyrmion phases, have been confirmed to exist in many of the helimagnetic B20 materials including MnSi [3], FeGe [4], FeCoSi [5], MnGe [6], *etc.* and are predicted to exist in all such materials [7, 8]. For reference, a selection of B20 materials displaying helimagnetic ordering – along with properties of this ordering – is presented in Appendix A.

All B20 materials belong to the $P2_13$ space group (No. 198, T^4). This is a primitive, or simple, cubic group with chiral tetrahedral symmetry, T. In this space group, the magnetic Mn ions of MnSi reside in the 4a Wyckoff position. This position can be parametrized by a single variable, x, as

$$\begin{aligned}
\mathbf{r}_1: &\quad \{x, x, x\} \\
\mathbf{r}_2: &\quad \{-x+\frac{1}{2}, -x, x+\frac{1}{2}\} \\
\mathbf{r}_3: &\quad \{-x, x+\frac{1}{2}, -x+\frac{1}{2}\} \\
\mathbf{r}_4: &\quad \{x+\frac{1}{2}, -x+\frac{1}{2}, -x\}
\end{aligned} \tag{1.1}$$

where the numeric labels applied to each site are maintained throughout this thesis. For MnSi, the parameter $x = 0.138$ in units of the cubic lattice parameter $a = 4.560$Å [9]. One can see that ions 2, 3, and, 4 reside outside of the standard cubic unit cell with a side length of 1 in units of lattice parameters (this length scale will be used as the standard throughout this thesis unless otherwise noted). This is normally useful as the 4 positions are nearest neighbours. However, if one desires to make a single, repeatable unit cell bounded by $[0, 1)$ a set of translations can be made to move all ions to equivalent positions within that cell

$$\begin{aligned}
\{x, x, x\} &\xrightarrow{+\{0,0,0\}} \{x, x, x\} \\
\{-x+\frac{1}{2}, -x, x+\frac{1}{2}\} &\xrightarrow{+\{0,1,0\}} \{-x+\frac{1}{2}, -x+1, x+\frac{1}{2}\} \\
\{-x, x+\frac{1}{2}, -x+\frac{1}{2}\} &\xrightarrow{+\{1,0,0\}} \{-x+1, x+\frac{1}{2}, -x+\frac{1}{2}\} \\
\{x+\frac{1}{2}, -x+\frac{1}{2}, -x\} &\xrightarrow{+\{0,0,1\}} \{x+\frac{1}{2}, -x+\frac{1}{2}, -x+1\}
\end{aligned} \tag{1.2}$$

.

This definition of the positions is utilized throughout. The coordination number of each of these sites is 6, with nearest neighbours at a distance of approximately 0.613. The unit cell of MnSi is displayed in Fig 1.1a with some nearest neighbours labelled.

The structure may be described using 4 interlaced cubic sublattices, with corners associated with each ion position. These sublattices will be referenced using the associated numerical label of that position. Alternatively, it may be described as a series of two-dimensional planes along a $\langle 111 \rangle$ axis of the crystal (where $\langle ... \rangle$ and $[...]$ follow the definitions of Ashcroft & Mermin [10]). There are two planes within a unit cell, containing one and three ions. In the specific direction [111] those ions are 1 and

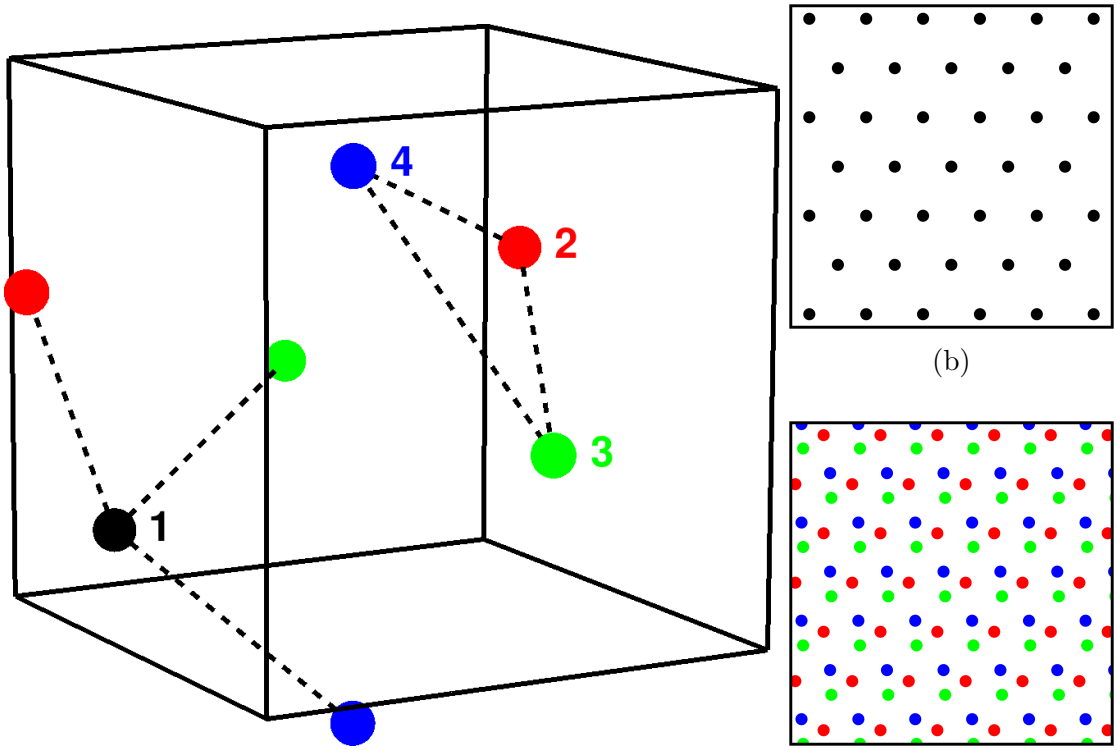


Figure 1.1: (a) The unit cell of MnSi with magnetic Mn ions shown. The four distinct ion positions of the unit cell are represented by different colours: (black, 1), (red, 2), (green, 3), (blue, 4). The 4 numerically labelled ions are within the unit cell, others are not. The dashed lines connecting ions indicate nearest neighbour pairs. (b) Cross-section of the lattice along [111] showing the 2D plane containing only sublattice 1. If only nearest neighbours are considered, there are no interactions in this plane. (c) Cross-section of the lattice along [111] showing the 2D plane containing sublattices 2, 3, and 4. Each small triangle is a set of nearest neighbours.

$\{2, 3, 4\}$ as shown in Fig. 1.1b and Fig. 1.1c. These layers will be denoted 1 and 2, respectively.

The space group $P2_13$ is compelling in part due to its lack of inversion symmetry. Lattices without this symmetry are known to allow interactions which can stabilize exotic magnetic structures.

1.2 Magnetic interactions

1.2.1 Heisenberg model

A frequently used phenomenological model is the Heisenberg exchange interaction. This exchange interaction between two (neighbouring) magnetic moments, S_i and S_j, is represented as a dot product with associated coupling coefficient J_{ij}

$$-J_{\mathrm{ij}}\mathbf{S}_{\mathrm{i}} \cdot \mathbf{S}_{\mathrm{j}}. \tag{1.3}$$

In this simple model it is easy to see that the ground state of any two magnetic moments can only be either ferromagnetic when $J < 0$ or antiferromagnetic when $J > 0$. These simple relations can be made more complicated through frustration, producing atypical structures (see Fig. 1.2). Summing over all pairs we find the spin Hamiltonian

$$\mathscr{H}_{\mathrm{Heis}} = \sum_{\mathrm{i,j}} -J_{\mathrm{ij}}\mathbf{S}_{\mathrm{i}} \cdot \mathbf{S}_{\mathrm{j}} \tag{1.4}$$

which is often simplified such that the coupling coefficient is a constant of all equidistant interactions (*i.e.*, $J_{ij} = J$) or reduced to a small number of constants for specific sets of interactions, *e.g.*, in-layer and between-layer. However, J_{ij} only need be constrained by the symmetry of the lattice. Clearly, these interactions are isotropic as they depend only on the relative orientation of the interacting magnetic moments.

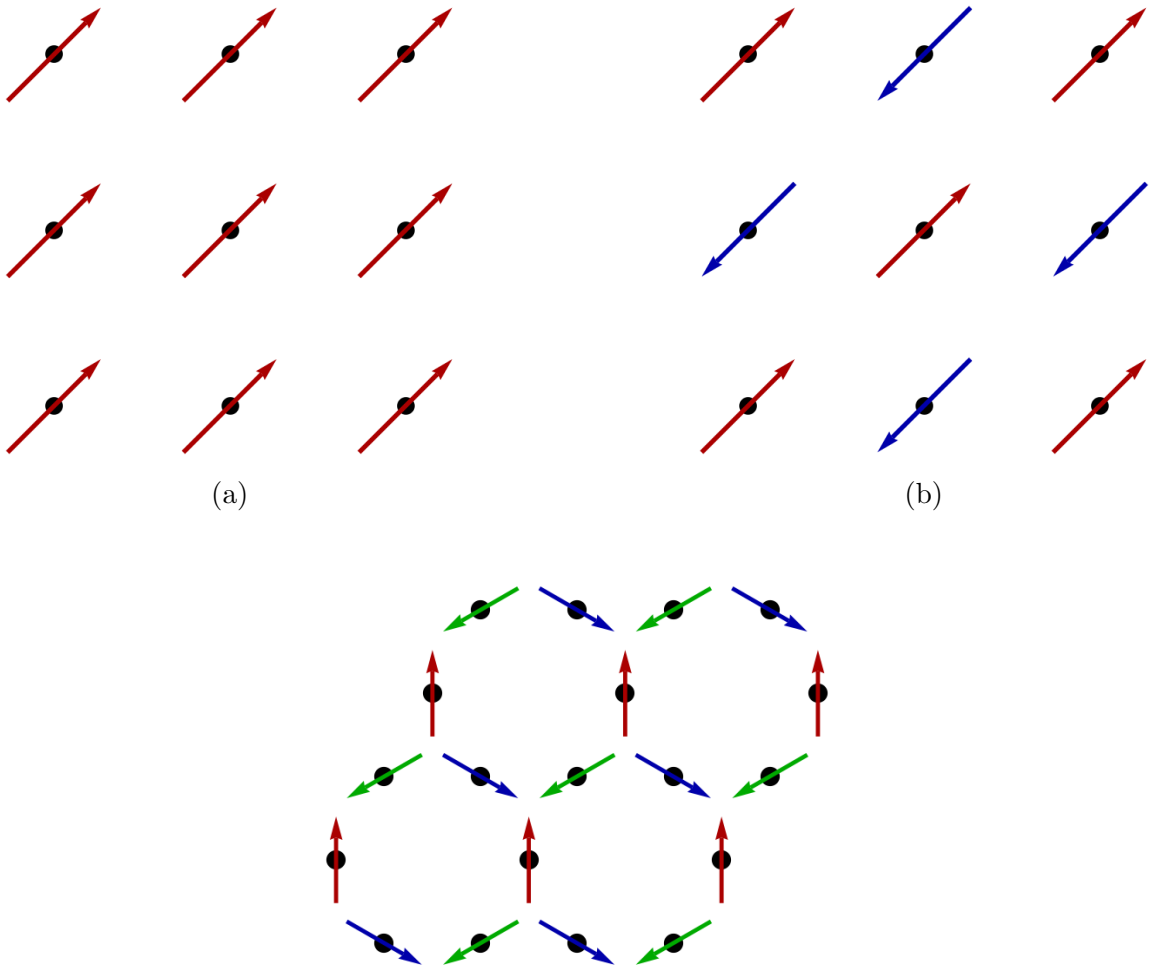


Figure 1.2: Magnetic orderings produced by the Heisenberg interaction. (a) Ferromagnetic ordering on a two-dimensional square lattice. (b) Antiferromagnetic ordering on a two-dimensional square lattice. (c) The $q = 0$ structure of the two-dimensional Kagome lattice, an example of a non-trivial ordering arising through frustration [11].

1.2.2 Dzyaloshinskii-Moriya interactions

The introduction of spin-orbit coupling allows for the appearance of anisotropic terms in the spin hamiltonian. One of these interactions is the antisymmetric exchange interaction, also known as the Dzyaloshinskii-Moriya interaction (DMI). This interaction was first predicted using symmetry as a means for describing the phenomenon

of helical spin structure in nonmetallic magnetic materials [12]. They are described by the cross product of two magnetic moments

$$\mathscr{H}_{\mathrm{DMI}} = \sum_{i,j} \mathbf{D}_{\mathrm{ij}} \cdot (\mathbf{S}_{\mathrm{i}} \times \mathbf{S}_{\mathrm{j}}) \tag{1.5}$$

where the coupling coefficient $\mathbf{D}_{\mathrm{ij}}$ is treated like a vector.

Interactions of this form favour magnetic moments canting out of ferromagnetic or antiferromagnetic alignment. Therefore, through competition with other exchange interactions, they are able to stabilize helical magnetic structures with a fixed chirality and also play an important role in the formation of skyrmion lattices in these materials [13, 14].

The microscopic description was later developed by Moriya, in which the spin-orbit coupling was identified as the mechanism leading to this interaction [15]. In this, a means for determining the orientation of the coupling vector $\mathbf{D}_{\mathrm{ij}}$ was also developed. This term must vanish if the two ions being considered are not identical, including the coordination number of the ion. In cases of identical ions, the symmetry of the lattice is considered. For ions located at $\mathbf{R}_i$ and $\mathbf{R}_j$, making line $\mathbf{R}_{ij}$ with midpoint $\mathbf{O}$

1. $\mathbf{D}_{\mathrm{ij}} = 0$ if a point of inversion is located at $\mathbf{O}$
2. $\mathbf{D}_{\mathrm{ij}} \perp \mathbf{R}_{ij}$ if a mirror plane perpendicular to $\mathbf{R}_{ij}$ passes through $\mathbf{O}$
3. $\mathbf{D}_{\mathrm{ij}} \perp$ mirror plane if a mirror plane includes $\mathbf{R}_{ij}$
4. $\mathbf{D}_{\mathrm{ij}} \perp \mathbf{C}_2$ if there is two-fold rotation axis $\mathbf{C}_2 \perp \mathbf{R}_{ij}$
5. $\mathbf{D}_{\mathrm{ij}} \parallel \mathbf{R}_{ij}$ if there is any rotation axis $\mathbf{C}_n \parallel \mathbf{R}_{ij}$ where $n > 2$

From this, we can see that for lattices lacking a centre of inversion, such as those in the B20 class, these interactions must be present. Further, while in theory, these interactions can differ for different pairs of magnetic sites, these rules enforce that $\mathbf{D}_{ij}$ must belong to the symmetry of the lattice. Though these interactions are often relatively weak relative to the Heisenberg interactions, the appearance of exotic magnetic phases in B20 materials reveal their importance.

1.2.3 Magnetic anisotropy

Other interactions are required for our model to reproduce physical magnetic states. First, the effect of magnetic anisotropies must be considered. A magnetic anisotropy is a term that describes how the magnetic properties of an ion may depend on direction. The causes of magnetic anisotropies and the effects which are modelled by these terms are numerous. Numerical models may also use anisotropy to simulate other physical effects, *e.g.*, applied pressure [16]. Common forms of magnetic anisotropy in a single material include shape anisotropy, magnetoelastic anisotropy, and magnetocrystalline anisotropy. Of these, only magnetocrystalline anisotropy – produced by the structure of the crystal lattice – will be included.

The magnetocrystalline anisotropy is equivalent to crystal electric fields (CEF) expressed using Stevens operator equivalents, given by same-site invariant terms of even order. Although terms of any order and configuration are permissible, very high order terms are generally excluded. I will consider only second and fourth-order terms of the form

$$\begin{aligned} A_2 &= \sum_{\mathbf{n},\mathbf{i},\kappa\mathbf{1}\neq\kappa\mathbf{2}} S^{\kappa 1}_{\mathrm{i,n}} S^{\kappa 2}_{\mathrm{i,n}} \\ A_4 &= \sum_{\mathbf{n},\mathbf{i},\kappa} (S^{\kappa}_{\mathrm{i,n}})^4 \end{aligned} \tag{1.6}$$

where $\kappa 1$ and $\kappa 2$ are non-equivalent components of the magnetic moment. Both anisotropies are invariant under lattice symmetries, as required.

These specific forms are chosen due to their effects on the lattice – the second-order anisotropy can be shown to prefer planar structures – and the inclusion of similar terms in other studies [16]. It is known that helical magnetic structures depend on magnetic anisotropies in multiple ways [17, 18]. Other second and fourth-order anisotropies of this type may exist – in fact, any combination of κ that is not altogether isotropic within the symmetry of the lattice is permissible. However, these prove sufficient to stabilize the structure.

1.2.4 Zeeman term

The final magnetic interaction to consider is the Zeeman term. The Zeeman term arises from the Zeeman effect, describing the interaction of a magnetic moment with an externally applied magnetic field. This interaction takes the form

$$\mathscr{H}_{\text{Zeeman}} = -\sum_i \mathbf{B} \cdot \mathbf{S}_i \tag{1.7}$$

where $\mathbf{B}$ is the applied field vector. In general, the Zeeman term includes a factor for the magnetic moment. Here, all applied fields are scaled such that $\mathbf{B}$ includes this term. The leading negative term favours magnetic moments aligning parallel or to the applied field.

1.3 Magnetic structure

The magnetic structure refers to the ordered arrangement of the magnetic moments of a crystal.

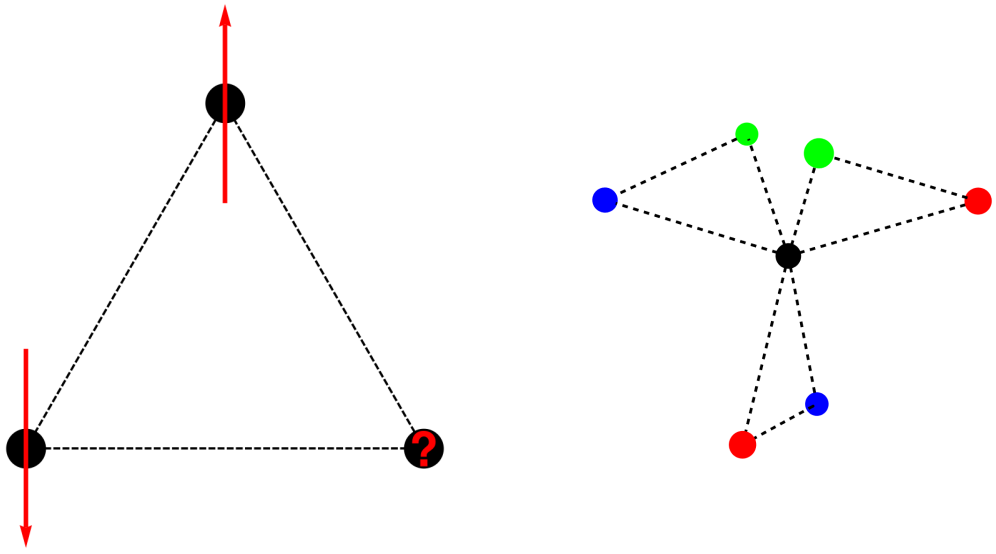
1.3.1 Magnetic frustration

Magnetic frustration occurs within magnetic crystals when competing interactions prevent the system from achieving a state with all bond energies minimized. Frustration often leads to disordered states well within ranges of predicted magnetic order and, in some cases, magnetic disorder persists (*e.g.*, spin ices). If magnetic order is observed, it tends to result in exotic orderings and exotic structures (*e.g.*, helical magnets). The phases of such materials are often hard to predict and must be obtained through experimental or numerical studies. Frustration is brought about through competition engendered by the structure and symmetry of the lattice, referred to as geometric frustration, or as competition between different interactions, *e.g.*, long-range vs short-range.

Geometric frustration occurs when a system cannot assume a state in which all pairwise bonds are minimized due to competition between shared neighbours. A simple-but-illustrative example is antiferromagnetic interactions in a two-dimensional

triangular lattice shown in Fig 1.3a [19]. This phenomenon can be observed in MnSi-like magnets by considering the shared bonds between neighbours of any ion. An example is depicted in Fig 1.3b.

Frustration induced by competing interactions can come in diverse forms. Competition could occur between long and short-range interactions, interactions of different forms, *etc.* Such competition is known to produce helical states in otherwise mundane structures [20, 21]. The general model introduced here will include considerable opportunity for this form of frustration, and both forms of frustration will be included implicitly.



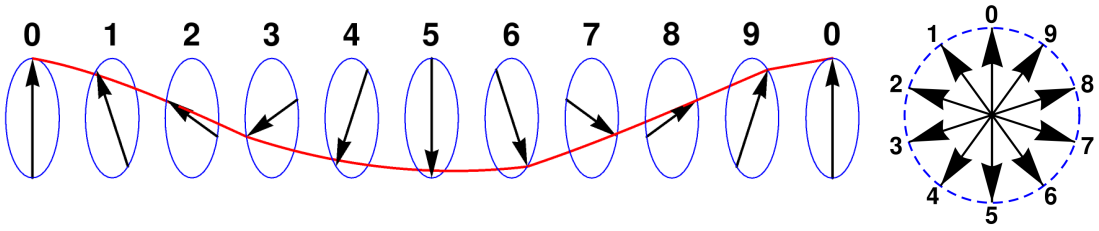
(a) (b)

Figure 1.3: Geometric frustration in (a) the antiferromagnetic Ising model on a two-dimensional triangular lattice and (b) the MnSi crystal structure. (a) The remaining ion cannot assume a unique ground state. Assuming either orientation will result in individual bond energies that are minimized and maximized, respectively. (b) The nearest neighbours of the ion in position 1 in the MnSi crystal lattice. Each neighbour shares a neighbour with ion 1 and is, therefore, part of a triangle of shared interactions. The symmetry of the lattice ensures the same for all ions.

1.3.2 Helical magnets

The terms helical magnet, helical magnetism, and helimagnetism are used to denote the incommensurate, exotic magnetic structure in which magnetic moments rotate

periodically along some axis in the crystal. The structure was first theorized to describe anomalous behaviour in MnO_2 [22] and $Cu_2Cr_2O_5$ [23]. They can exist in both three-dimensional and two-dimensional structures and are stabilized up to room temperatures [24, 25].



(a) (b)

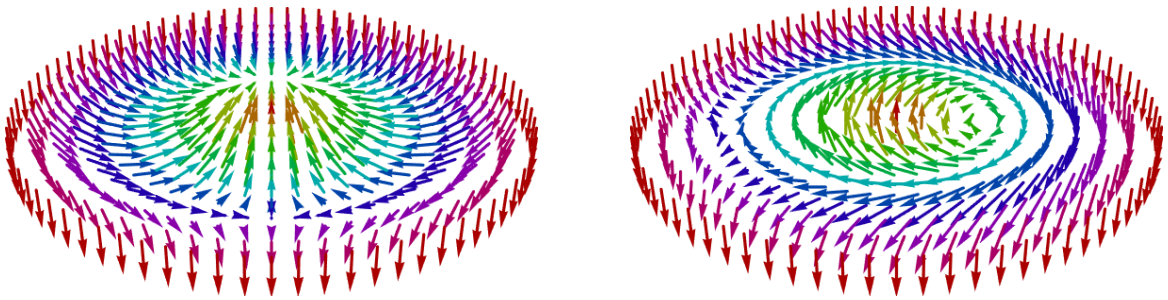
Figure 1.4: Helical magnetism in a three-dimensional crystal as viewed from (a) normal to the axis of rotation and (b) along the axis of rotation. The helix depicted is right-handed with a period of 10 units. The tips of the vector representations trace out a helix in space. In general, the period of the helix will not align with the underlying lattice producing an incommensurate structure that breaks translational symmetry.

The helix has a wavevector $\mathbf{k}$ and the general form of the orientation of a given magnetic moment in the lattice is

$$\mathbf{S}(\mathbf{r}) = \mathbf{x}_1 \cos(\mathbf{k}\cdot\mathbf{r} + \Phi) + \mathbf{x}_2 \sin(\mathbf{k}\cdot\mathbf{r} + \Phi) \tag{1.8}$$

where the vectors $\mathbf{x}_1$ and $\mathbf{x}_2$ form an orthogonal basis with $\mathbf{k}$ and Φ is some additive phase constant. The sign of the wavevector defines the chirality of the helix. The incommensurate nature of the magnetic phase breaks translational symmetries of the lattice.

Helical magnets are a contemporary and active area of magnetic research due to their unique properties and relation to another useful exotic structure, skyrmions [7, 26]. Physical skyrmions were first described in the context of nuclear physics [27] and analogous structures have been described for magnetic systems. A skyrmion is an exotic magnetic phase resembling a topological knot that exists both as an excitation and in stable, lattice-like structures and is shown in Fig. 1.5. These structures have potential uses in several different fields, *e.g.*, spintronics. The relation between helimagnetism and skyrmions is specifically important in MnSi which exhibits a phase of



(a) (b)

Figure 1.5: Models of magnetic skyrmions in the (a) Néel-type and (b) Bloch-type configurations.

stable skyrmion lattices [3].

1.4 MnSi

1.4.1 Experimental observations of MnSi

The material MnSi has been highlighted here due to its general importance to this class of materials. The helical phase was among the first observed in B20 crystals [2]. Likewise, the experimental discovery of a skyrmion phase was among the first [3]. An example phase diagram of MnSi, displaying both of these phases, is shown in Fig. 1.6.

Of particular concern in this thesis is the helical phase of MnSi. This phase was determined to exist at low temperatures and is incommensurate to the lattice with a wavevector $\mathbf{k} = 0.035\text{Å}^{-1}\langle 1,1,1\rangle$, which corresponds to a wavelength of $\lambda \approx 39.38$ in units of lattice vectors. Since this helix is oriented along a $\langle 111\rangle$ direction it is pertinent to measure the wavelength in units of unit cell diagonals. In these units the wavelength $\lambda \approx \frac{39.38}{\sqrt{3}} \approx 22.74$. The orientation of the helix is organized such that the two unique layers along a $\langle 111\rangle$ direction are ferromagnetic within each layer with a phase shift between layers. The angle between a given layer and the similar layer along one full $\langle 111\rangle$ vector is $\theta \approx 0.276$ rad.

Interest in this helical phase has persisted and, in contemporary studies, novel phenomena have been discovered. Specifically, the helical phase was shown to have a measurable shift in the phase between the two unique layers [28]. That is, if one considers the equation for magnetic moments in a helical structure

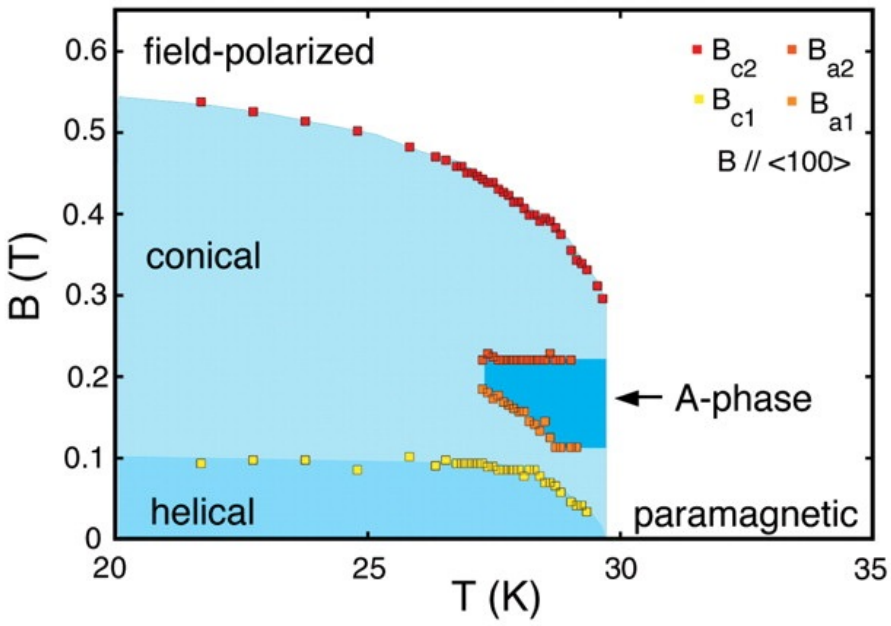


Figure 1.6: The magnetic phase diagram for MnSi displaying helical ordering at low T and applied field B. The skyrmion phase is represented as the A-phase in this diagram. The listed fields are the usual critical fields B_{c1} and B_{c2}, as well as the transition to the A-phase B_{a1} and B_{a2}. From Mühlbauer, S, *et al.* [3]. Reprinted with permission from AAAS.

$$\mathbf{S}(\mathbf{r}) = \mathbf{x}_1 \cos(\mathbf{k}\cdot\mathbf{r} + \Phi) + \mathbf{x}_2 \sin(\mathbf{k}\cdot\mathbf{r} + \Phi) \tag{1.9}$$

then the phase of positions associated with layer 2 will differ by some amount, ϕ, from the value given by this equation. This can also be interpreted as two separate overlapping helices with a phase shift, ϕ. The phase shift can be calculated as the difference between the "expected" angle between the two layers in one unit cell (inferred from the wavevector and the distance between layers), α_{exp}, and that which is observed in experiment, α_{obs}. In MnSi, $\alpha_{exp} \approx 0.483\theta \approx 0.133$ rad and the phase shift was measured to be $\phi = \alpha_{obs} - \alpha_{exp} \approx 2^\circ \approx 0.035$ rad. This feature is particularly interesting as studies of other crystals of this type have not revealed similar behaviour [29].

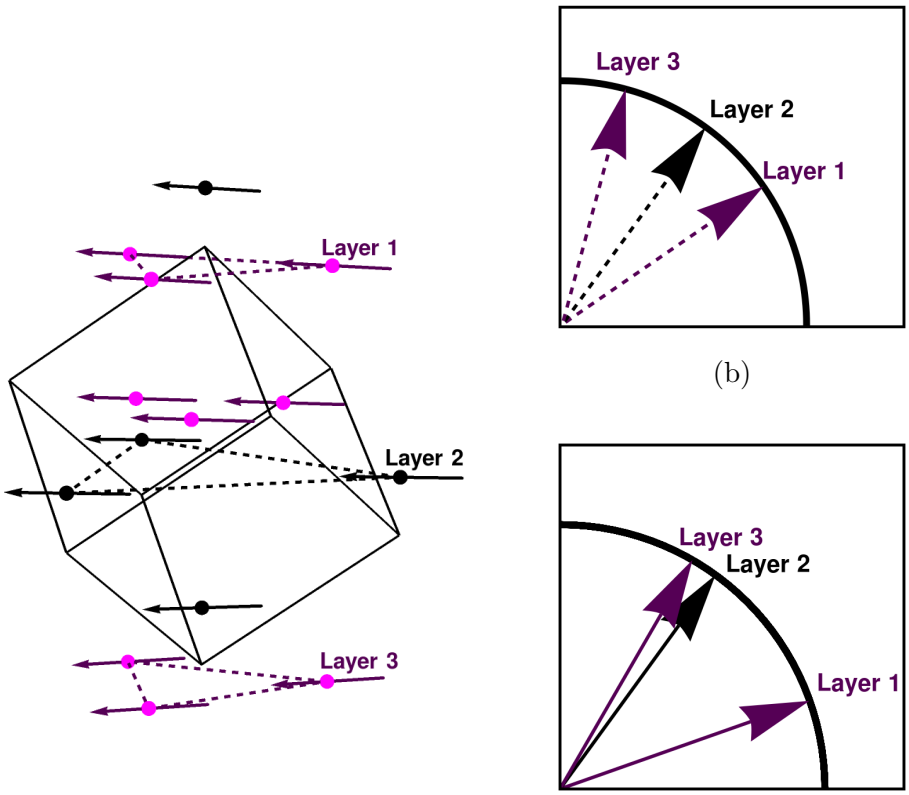
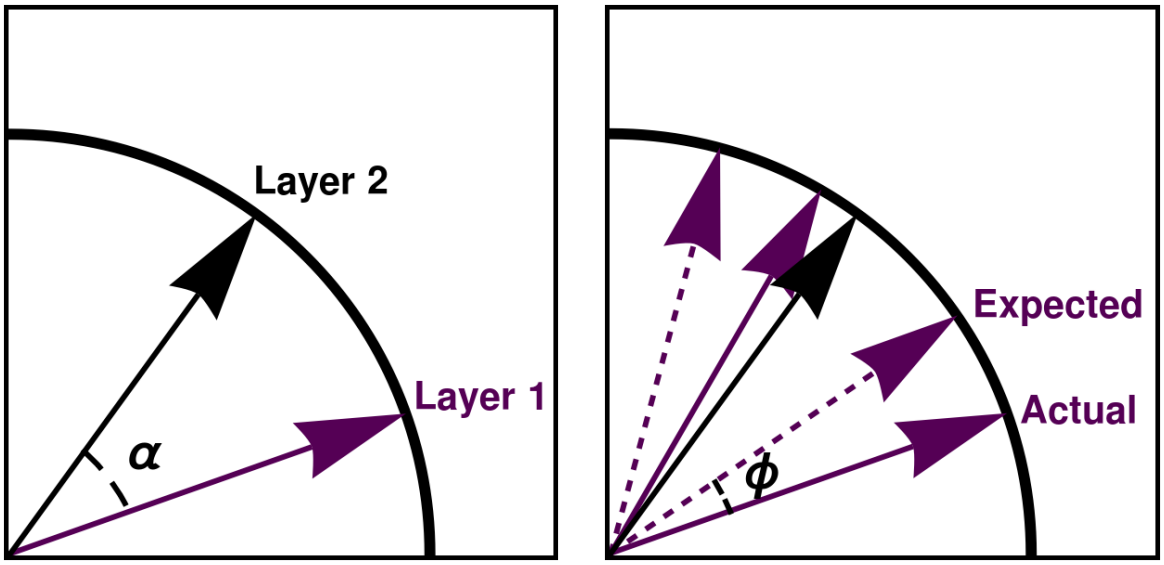


Figure 1.7: A representation of the phase shift in MnSi. (a) A three-dimensional representation of the MnSi crystal lattice with 3 layers highlighted: 2 similar layers at a distance of one cubic diagonal of the unit cell (Layers 1 and 3) and the layer in the same unit cell as the layer above it (Layer 2). (b) The expected relative orientation of the three ferromagnetic layers denoted with dashed arrows. (c) The actual orientation of the three ferromagnetic layers denoted with solid arrows. Modified from [28].



(a) (b)

Figure 1.8: Definitions of the angle parameters (a) α and (b) ϕ using the example from Fig. 1.7. Here, the term *expected* refers to the orientation of the magnetic moments inferred from the measured wavevector and distance between layers.

1.4.2 Other studies of MnSi

The interesting properties and contemporary enthusiasm in MnSi have also attracted a large number of other experimental and analytical studies of the system. I will take advantage of these results as a means to test both the model produced here and the analyses of said model. One such result from Grigoriev *et al.* surmises that the handedness of the helical phase of MnSi and MnSi-like crystals is dependent on the chirality of the lattice itself, allowing for both left and right-handed structures [30]. Further, a known result is that the formation and direction of the helical wavevector are dependent on the sign and magnitude of anisotropic exchange interactions [18]. From these results, one may determine appropriate coupling strengths for the anisotropies A_2 and A_4 such that the desired structure is produced.

Analysis of specific note is the results of Chizhikov and Dmitrienko using a microscopic model including the usual Heisenberg, DMI, and Zeeman interactions, and their associated parameters, as discussed in Section 1.2 [31]. Explicitly, the model used is

$$
\begin{aligned}
E = & \frac{1}{2} \sum_{i} \sum_{j} (-J \mathbf{s}_i \cdot \mathbf{s_j} + \mathbf{D}_{ij} \cdot [\mathbf{s}_i \times \mathbf{s}_j]) \\
& - g\mu_B \sum_{i} \mathbf{H} \cdot \mathbf{s}_i .
\end{aligned}
\tag{1.10}
$$

In this study, they analyzed the lowest energy states of the model and the associated structures. Multiple relationships regarding wavevector direction and magnitude, canting, and the anomalous phase ϕ are reported. These analyses will form a basis with which I will compare the model produced here, with each result presented at the appropriate time.

Chapter 2

The Microscopic Model

To better understand the phenomena, both novel and familiar, present in MnSi and MnSi-type crystals I will construct a microscopic model using as few assumptions as necessary. The intention is to construct a completely general model that will capture all phenomenological features of these crystals. The model will contain only NN exchange interactions as the second and third nearest neighbours are at a distance $\sim$ 50% larger than the first. However, the model could be easily extended to neighbours of any order, or to include weaker, long-range interactions.

2.1 Construction through symmetry

To build the most general model possible only symmetry considerations will be used to generate the terms. Similar work has been done considering spin density models [32], but I will be constructing a microscopic model with these features. Within a unit cell, there are 108 possible two-term nearest neighbour exchange interactions, corresponding to the 4 ions, each with a coordination number of 6, and the 3 spatial components. The total energy of these interactions must be invariant within the symmetry of the lattice.

Number	ITA	Seitz
(1)	1	$\{1 \mid 0\}$
(2)	$2(0,0,1/2)1/4,0,z$	$\{2_{001} \mid 1/2\ 0\ 1/2\}$
(3)	$2(0,1/2,0)0,y,1/4$	$\{2_{010} \mid 0\ 1/2\ 1/2\}$
(4)	$2(1/2,0,0)x,1/4,0$	$\{2_{100} \mid 1/2\ 1/2\ 0\}$
(5)	$3^{-}x,x,x$	$\{3^{-}_{111} \mid 0\}$
(6)	$3^{-}(-1/3,1/3,1/3)x+1/6,-x+1/6,-x$	$\{3^{-}_{\bar{1}\bar{1}1} \mid 0\ 1/2\ 1/2\}$
(7)	$3^{-}(1/3,1/3,-1/3)-x+1/3,-x+1/6,x$	$\{3^{-}_{\bar{1}1\bar{1}} \mid 1/2\ 1/2\ 0\}$
(8)	$3^{-}(1/3,-1/3,1/3)-x-1/6,x+1/3,-x$	$\{3^{-}_{1\bar{1}\bar{1}} \mid 1/2\ 0\ 1/2\}$
(9)	$3^{+}x,x,x$	$\{3^{+}_{111} \mid 0\}$
(10)	$3^{+}-x+1/2,x,-x$	$\{3^{+}_{\bar{1}1\bar{1}} \mid 1/2\ 1/2\ 0\}$
(11)	$3^{+}x+1/2,-x-1/2,-x$	$\{3^{-}_{1\bar{1}\bar{1}} \mid 1/2\ 0\ 1/2\}$
(12)	$3^{+}-x,-x+1/2,x$	$\{3^{-}_{\bar{1}\bar{1}1} \mid 0\ 1/2\ 1/2\}$

Table 2.1: The 12 point group symmetries of space group $P2_13$, including screw axes, along with their notation in both ITA and Seitz. These symmetries include both rotations and screw axes of different order and chirality: 2-fold axes $(2-4)$, right-handed 3-fold axes $(5-8)$, and left-handed 3-fold axes $(9-12)$.

$P2_13$ contains 12 point group symmetries as denoted in Table 2.1. These symmetry operations are applied to bilinear terms in the interaction, for example

$$\{3^{-}_{111} \mid 0\} S_1^x S_2^x = S_1^y S_4^y. \tag{2.1}$$

A guide of the complete list of transformations for each magnetic moment is given in Table 2.2. Therefore, the 108 possible terms will be organized into 9 invariants of 12 symmetrically equivalent terms.

One may note that, since some nearest neighbours inhabit other unit cells, some terms involve sites located in different unit cells. Further, as each ion has two nearest neighbours located in the same sublattice, it is necessary to differentiate which ions are being discussed. For this purpose, the notation used throughout is

$$S^{\kappa}_{a_{ijl}} \tag{2.2}$$

where a refers to the sublattice, ijl represent the lattice translation from the unit

cell being considered, $\mathbf{n}+\{i,j,l\}$, and κ represents the component of the magnetic moment. For example,

$$S^{x}_{1_{0\bar{1}1}} \tag{2.3}$$

would represent the x component of the magnetic moment in sublattice 1 and the unit cell located at $\mathbf{n}+\{0,-1,1\}$.

For simplicity, all 9 terms are generated from a bilinear term of the form $S^{\kappa_1}_{1_{000}}S^{\kappa_2}_{2_{0\bar{1}0}}$ with $\kappa_1,\kappa_2\in\{x,y,z\}$. The first of these combinations, corresponding to $\kappa_1=\kappa_2=x$, is

$$\begin{aligned}H_1=\sum_{\mathbf{n}}\big[&S^{x}_{1_{000}}S^{x}_{2_{0\bar{1}0}}+S^{x}_{1_{000}}S^{x}_{2_{0\bar{1}\bar{1}}}+S^{x}_{3_{000}}S^{x}_{4_{\bar{1}0\bar{1}}}+S^{x}_{3_{000}}S^{x}_{4_{000}}\\&+S^{y}_{1_{000}}S^{y}_{4_{00\bar{1}}}+S^{y}_{1_{000}}S^{y}_{4_{\bar{1}0\bar{1}}}+S^{y}_{2_{0\bar{1}0}}S^{y}_{3_{\bar{1}\bar{1}0}}+S^{y}_{2_{\bar{1}\bar{1}0}}S^{y}_{3_{\bar{1}\bar{1}0}}\\&+S^{z}_{1_{000}}S^{z}_{3_{\bar{1}00}}+S^{z}_{1_{000}}S^{z}_{3_{\bar{1}\bar{1}0}}+S^{z}_{2_{0\bar{1}\bar{1}}}S^{z}_{4_{00\bar{1}}}+S^{z}_{2_{0\bar{1}\bar{1}}}S^{z}_{4_{0\bar{1}\bar{1}}}\big]\end{aligned} \tag{2.4}$$

where the sum is over all unit cells $\mathbf{n}=\{n_x,n_y,n_z\}$. The other 8 combinations are

$$\begin{aligned}H_2=\sum_{\mathbf{n}}\big[&S^{y}_{1_{000}}S^{y}_{2_{0\bar{1}0}}+S^{y}_{1_{000}}S^{y}_{2_{0\bar{1}\bar{1}}}+S^{y}_{3_{\bar{1}00}}S^{y}_{4_{\bar{1}0\bar{1}}}+S^{y}_{3_{\bar{1}0\bar{1}}}S^{y}_{4_{\bar{1}0\bar{1}}}\\&+S^{z}_{1_{000}}S^{z}_{4_{00\bar{1}}}+S^{z}_{1_{000}}S^{z}_{4_{\bar{1}0\bar{1}}}+S^{z}_{2_{0\bar{1}0}}S^{z}_{3_{\bar{1}\bar{1}0}}+S^{z}_{2_{\bar{1}\bar{1}0}}S^{z}_{3_{\bar{1}\bar{1}0}}\\&+S^{x}_{1_{000}}S^{x}_{3_{\bar{1}00}}+S^{x}_{1_{000}}S^{x}_{3_{\bar{1}\bar{1}0}}+S^{x}_{2_{0\bar{1}\bar{1}}}S^{x}_{4_{00\bar{1}}}+S^{x}_{2_{0\bar{1}\bar{1}}}S^{x}_{4_{0\bar{1}\bar{1}}}\big]\end{aligned} \tag{2.5}$$

$$\begin{aligned}H_3=\sum_{\mathbf{n}}\big[&S^{z}_{1_{000}}S^{z}_{2_{0\bar{1}0}}+S^{z}_{1_{000}}S^{z}_{2_{0\bar{1}\bar{1}}}+S^{z}_{3_{\bar{1}00}}S^{z}_{4_{\bar{1}0\bar{1}}}+S^{z}_{3_{\bar{1}0\bar{1}}}S^{z}_{4_{\bar{1}0\bar{1}}}\\&+S^{x}_{1_{000}}S^{x}_{4_{00\bar{1}}}+S^{x}_{1_{000}}S^{x}_{4_{\bar{1}0\bar{1}}}+S^{x}_{2_{0\bar{1}0}}S^{x}_{3_{\bar{1}\bar{1}0}}+S^{x}_{2_{\bar{1}\bar{1}0}}S^{x}_{3_{\bar{1}\bar{1}0}}\\&+S^{y}_{1_{000}}S^{y}_{3_{\bar{1}00}}+S^{y}_{1_{000}}S^{y}_{3_{\bar{1}\bar{1}0}}+S^{y}_{2_{0\bar{1}\bar{1}}}S^{y}_{4_{00\bar{1}}}+S^{y}_{2_{0\bar{1}\bar{1}}}S^{y}_{4_{0\bar{1}\bar{1}}}\big]\end{aligned} \tag{2.6}$$

$$H_4 = \sum_{\mathbf{n}} \big[S^x_{1_{000}} S^y_{2_{0\bar{1}0}} + S^y_{1_{000}} S^x_{2_{0\bar{1}\bar{1}}} - S^x_{3_{\bar{1}00}} S^y_{4_{\bar{1}0\bar{1}}} - S^y_{3_{\bar{1}0\bar{1}}} S^x_{4_{\bar{1}0\bar{1}}} + S^y_{1_{000}} S^z_{4_{00\bar{1}}} + S^z_{1_{000}} S^y_{4_{\bar{1}0\bar{1}}} - S^y_{2_{0\bar{1}0}} S^z_{3_{\bar{1}\bar{1}0}} - S^z_{2_{\bar{1}\bar{1}0}} S^y_{3_{\bar{1}\bar{1}0}} + S^z_{1_{000}} S^x_{3_{\bar{1}00}} + S^x_{1_{000}} S^z_{3_{\bar{1}\bar{1}0}} - S^z_{2_{0\bar{1}\bar{1}}} S^x_{4_{00\bar{1}}} - S^x_{2_{0\bar{1}\bar{1}}} S^z_{4_{0\bar{1}\bar{1}}} \big] \tag{2.7}$$

$$H_5 = \sum_{\mathbf{n}} \big[S^y_{1_{000}} S^x_{2_{0\bar{1}0}} + S^x_{1_{000}} S^y_{2_{0\bar{1}\bar{1}}} - S^y_{3_{\bar{1}00}} S^x_{4_{\bar{1}0\bar{1}}} - S^x_{3_{\bar{1}0\bar{1}}} S^y_{4_{\bar{1}0\bar{1}}} + S^z_{1_{000}} S^y_{4_{00\bar{1}}} + S^y_{1_{000}} S^z_{4_{\bar{1}0\bar{1}}} - S^z_{2_{0\bar{1}0}} S^y_{3_{\bar{1}\bar{1}0}} - S^y_{2_{\bar{1}\bar{1}0}} S^z_{3_{\bar{1}\bar{1}0}} + S^x_{1_{000}} S^z_{3_{\bar{1}00}} + S^z_{1_{000}} S^x_{3_{\bar{1}\bar{1}0}} - S^x_{2_{0\bar{1}\bar{1}}} S^z_{4_{00\bar{1}}} - S^z_{2_{0\bar{1}\bar{1}}} S^x_{4_{0\bar{1}\bar{1}}} \big] \tag{2.8}$$

$$H_6 = \sum_{\mathbf{n}} \big[S^y_{1_{000}} S^z_{2_{0\bar{1}0}} - S^z_{1_{000}} S^y_{2_{0\bar{1}\bar{1}}} - S^y_{3_{\bar{1}00}} S^z_{4_{\bar{1}0\bar{1}}} + S^z_{3_{\bar{1}0\bar{1}}} S^y_{4_{\bar{1}0\bar{1}}} + S^z_{1_{000}} S^x_{4_{00\bar{1}}} - S^x_{1_{000}} S^z_{4_{\bar{1}0\bar{1}}} - S^z_{2_{0\bar{1}0}} S^x_{3_{\bar{1}\bar{1}0}} + S^x_{2_{\bar{1}\bar{1}0}} S^z_{3_{\bar{1}\bar{1}0}} + S^x_{1_{000}} S^y_{3_{\bar{1}00}} - S^y_{1_{000}} S^x_{3_{\bar{1}\bar{1}0}} - S^x_{2_{0\bar{1}\bar{1}}} S^y_{4_{00\bar{1}}} + S^y_{2_{0\bar{1}\bar{1}}} S^x_{4_{0\bar{1}\bar{1}}} \big] \tag{2.9}$$

$$H_7 = \sum_{\mathbf{n}} \big[S^z_{1_{000}} S^y_{2_{0\bar{1}0}} - S^y_{1_{000}} S^z_{2_{0\bar{1}\bar{1}}} - S^z_{3_{\bar{1}00}} S^y_{4_{\bar{1}0\bar{1}}} + S^y_{3_{\bar{1}0\bar{1}}} S^z_{4_{\bar{1}0\bar{1}}} + S^x_{1_{000}} S^z_{4_{00\bar{1}}} - S^z_{1_{000}} S^x_{4_{\bar{1}0\bar{1}}} - S^x_{2_{0\bar{1}0}} S^z_{3_{\bar{1}\bar{1}0}} + S^z_{2_{\bar{1}\bar{1}0}} S^x_{3_{\bar{1}\bar{1}0}} + S^y_{1_{000}} S^x_{3_{\bar{1}00}} - S^x_{1_{000}} S^y_{3_{\bar{1}\bar{1}0}} - S^y_{2_{0\bar{1}\bar{1}}} S^x_{4_{00\bar{1}}} + S^x_{2_{0\bar{1}\bar{1}}} S^y_{4_{0\bar{1}\bar{1}}} \big] \tag{2.10}$$

$$H_8 = \sum_{\mathbf{n}} \big[S^z_{1_{000}} S^x_{2_{0\bar{1}0}} - S^x_{1_{000}} S^z_{2_{0\bar{1}\bar{1}}} + S^z_{3_{\bar{1}00}} S^x_{4_{\bar{1}0\bar{1}}} - S^x_{3_{\bar{1}0\bar{1}}} S^z_{4_{\bar{1}0\bar{1}}} + S^x_{1_{000}} S^y_{4_{00\bar{1}}} - S^y_{1_{000}} S^x_{4_{\bar{1}0\bar{1}}} + S^x_{2_{0\bar{1}0}} S^y_{3_{\bar{1}\bar{1}0}} - S^y_{2_{\bar{1}\bar{1}0}} S^x_{3_{\bar{1}\bar{1}0}} + S^y_{1_{000}} S^z_{3_{\bar{1}00}} - S^z_{1_{000}} S^y_{3_{\bar{1}\bar{1}0}} + S^y_{2_{0\bar{1}\bar{1}}} S^z_{4_{00\bar{1}}} - S^z_{2_{0\bar{1}\bar{1}}} S^y_{4_{0\bar{1}\bar{1}}} \big] \tag{2.11}$$

$$
\begin{aligned}
H_9 = \sum_{\mathbf{n}} \big[& S^x_{1_{000}} S^z_{2_{0\bar{1}0}} - S^z_{1_{000}} S^x_{2_{0\bar{1}\bar{1}}} + S^x_{3_{\bar{1}00}} S^z_{4_{\bar{1}0\bar{1}}} - S^z_{3_{\bar{1}0\bar{1}}} S^x_{4_{\bar{1}0\bar{1}}} \\
& + S^y_{1_{000}} S^x_{4_{00\bar{1}}} - S^x_{1_{000}} S^y_{4_{\bar{1}0\bar{1}}} + S^y_{2_{0\bar{1}0}} S^x_{3_{\bar{1}\bar{1}0}} - S^x_{2_{\bar{1}\bar{1}0}} S^y_{3_{\bar{1}\bar{1}0}} \\
& + S^z_{1_{000}} S^y_{3_{\bar{1}00}} - S^y_{1_{000}} S^z_{3_{\bar{1}\bar{1}0}} + S^z_{2_{0\bar{1}\bar{1}}} S^y_{4_{00\bar{1}}} - S^y_{2_{0\bar{1}\bar{1}}} S^z_{4_{0\bar{1}\bar{1}}} \big]
\end{aligned}
\tag{2.12}
$$

It will prove useful to combine non-colinear terms of the same components into symmetric and antisymmetric terms. These, in combination with the same-site anisotropic terms and the Zeeman term discussed in Section 1.2, form the full, general model in Eq. 2.13.

$$
\begin{aligned}
H^{xx} &= H_1 \\
H^{yy} &= H_2 \\
H^{zz} &= H_3 \\
H^{xy}_s &= \frac{H_4 + H_5}{2} \\
H^{xy}_a &= \frac{H_4 - H_5}{2} \\
H^{yz}_s &= \frac{H_6 + H_7}{2} \\
H^{yz}_a &= \frac{H_6 - H_7}{2} \\
H^{zx}_s &= \frac{H_8 + H_9}{2} \\
H^{zx}_a &= \frac{H_8 - H_9}{2} \\
H_{A2} &= \sum_{\mathbf{n},\mathbf{i},\kappa\mathbf{1}\neq\kappa\mathbf{2}} S^{\kappa 1}_{\mathrm{i,n}} S^{\kappa 2}_{\mathrm{i,n}} \\
H_{A4} &= \sum_{\mathbf{n},\mathbf{i},\kappa} (S^{\kappa}_{\mathrm{i,n}})^4 \\
H_{\mathrm{Zeeman}} &= -\sum_i \mathbf{B} \cdot \mathbf{S}_i
\end{aligned}
\tag{2.13}
$$

In doing so I change the notation such that each term is associated with the components of the $S_{1_{000}}$ and $S_{2_{0\bar{1}0}}$ terms from which they were generated (Eg. $H^{xx} \sim S^x_{1_{000}} S^x_{2_{0\bar{1}0}}$).

The terms of this model can be separated into four groupings that will be referenced throughout: the diagonal terms $\{H^{xx}, H^{yy}, H^{zz}\}$, the symmetric terms $\{H_s^{xy}, H_s^{yz}, H_s^{zx}\}$, the antisymmetric terms $\{H_a^{xy}, H_a^{yz}, H_a^{zx}\}$, and the anisotropic terms $\{H_{A2}, H_{A4}\}$. The full hamiltonian is represented as the sum of these terms with associated coupling constants

$$\begin{aligned}\mathscr{H} = &\mathscr{J}^{xx}H^{xx} + \mathscr{J}^{yy}H^{yy} + \mathscr{J}^{zz}H^{zz} + \mathscr{J}_s^{xy}H_s^{xy} + \mathscr{J}_a^{xy}H_a^{xy} \\ &+ \mathscr{J}_s^{yz}H_s^{yz} + \mathscr{J}_a^{yz}H_a^{yz} + \mathscr{J}_s^{zx}H_s^{zx} + \mathscr{J}_a^{zx}H_a^{zx} \\ &+ \mathscr{J}_{A2}H_{A2} + \mathscr{J}_{A4}H_{A4} + H_{\text{Zeeman}}\end{aligned} \tag{2.14}$$

This general model is complete up to the determination of the relative value of the coupling constants. For the purposes of this thesis $\mathscr{J}^{xx} = 1$ is set. All other terms are defined relative to this term, defining the energy scale used throughout. All related measurements are therefore presented in reduced units with this scale.

(1)	(2)	(3)	(4)	(5)	(6)	(7)	(8)	(9)	(10)	(11)	(12)
$S^{xyz}_{1_{000}}$	$S^{\bar{x}\bar{y}z}_{2_{0\bar{1}0}}$	$S^{\bar{x}y\bar{z}}_{3_{\bar{1}00}}$	$S^{x\bar{y}\bar{z}}_{4_{00\bar{1}}}$	$S^{yzx}_{1_{000}}$	$S^{\bar{y}\bar{z}x}_{4_{00\bar{1}}}$	$S^{\bar{y}z\bar{x}}_{2_{0\bar{1}0}}$	$S^{y\bar{z}\bar{x}}_{3_{\bar{1}00}}$	$S^{zxy}_{1_{000}}$	$S^{\bar{z}\bar{x}y}_{3_{\bar{1}00}}$	$S^{\bar{z}x\bar{y}}_{4_{00\bar{1}}}$	$S^{z\bar{x}\bar{y}}_{2_{0\bar{1}0}}$
$S^{xyz}_{2_{000}}$	$S^{\bar{x}\bar{y}z}_{1_{0\bar{1}1}}$	$S^{\bar{x}y\bar{z}}_{4_{\bar{1}1\bar{1}}}$	$S^{x\bar{y}\bar{z}}_{3_{0\bar{1}\bar{1}}}$	$S^{yzx}_{4_{000}}$	$S^{\bar{y}\bar{z}x}_{1_{0\bar{1}}}$	$S^{\bar{y}z\bar{x}}_{3_{\bar{1}\bar{1}1}}$	$S^{y\bar{z}\bar{x}}_{2_{\bar{1}0\bar{1}}}$	$S^{zxy}_{3_{000}}$	$S^{\bar{z}\bar{x}y}_{1_{\bar{1}10}}$	$S^{\bar{z}x\bar{y}}_{2_{1\bar{1}\bar{1}}}$	$S^{z\bar{x}\bar{y}}_{4_{\bar{1}\bar{1}0}}$
$S^{xyz}_{3_{000}}$	$S^{\bar{x}\bar{y}z}_{4_{\bar{1}\bar{1}0}}$	$S^{\bar{x}y\bar{z}}_{1_{\bar{1}10}}$	$S^{x\bar{y}\bar{z}}_{2_{1\bar{1}\bar{1}}}$	$S^{yzx}_{2_{000}}$	$S^{\bar{y}\bar{z}x}_{3_{0\bar{1}\bar{1}}}$	$S^{\bar{y}z\bar{x}}_{1_{0\bar{1}1}}$	$S^{y\bar{z}\bar{x}}_{4_{\bar{1}1\bar{1}}}$	$S^{zxy}_{4_{000}}$	$S^{\bar{z}\bar{x}y}_{2_{\bar{1}0\bar{1}}}$	$S^{\bar{z}x\bar{y}}_{1_{10\bar{1}}}$	$S^{z\bar{x}\bar{y}}_{3_{\bar{1}\bar{1}1}}$
$S^{xyz}_{4_{000}}$	$S^{\bar{x}\bar{y}z}_{3_{\bar{1}\bar{1}1}}$	$S^{\bar{x}y\bar{z}}_{2_{\bar{1}0\bar{1}}}$	$S^{x\bar{y}\bar{z}}_{1_{10\bar{1}}}$	$S^{yzx}_{3_{000}}$	$S^{\bar{y}\bar{z}x}_{2_{1\bar{1}\bar{1}}}$	$S^{\bar{y}z\bar{x}}_{4_{\bar{1}\bar{1}0}}$	$S^{y\bar{z}\bar{x}}_{1_{\bar{1}\bar{1}0}}$	$S^{zxy}_{2_{000}}$	$S^{\bar{z}\bar{x}y}_{4_{\bar{1}1\bar{1}}}$	$S^{\bar{z}x\bar{y}}_{3_{0\bar{1}\bar{1}}}$	$S^{z\bar{x}\bar{y}}_{1_{0\bar{1}1}}$

Table 2.2: A guide of the transformation of each ion under the symmetries of space group $P2_13$ as described in Table 2.1.

2.2 Relationship to other models

The described model appears distinct from all previous models. However, the details of previous models are embedded within these terms. Detailing these relations will elucidate the function of each term and groupings of terms. It is sufficient to make a comparison to the basic, underlying models described in Section 1.2 since these interactions serve as a throughline between models.

First consider the diagonal terms H^{xx}, H^{yy}, and H^{zz}. These terms are composed of only colinear components of each magnetic moment. If one ignores the coupling constants, the sum of these terms will contain the inner product of every pair of nearest neighbours. However, since each term in this product is associated with a different coupling constant, the sum is of the form

$$\begin{aligned}\mathscr{J}^{xx}H^{xx} + \mathscr{J}^{yy}H^{yy} + \mathscr{J}^{zz}H^{zz} = \\ \sum_{\mathbf{n}} \mathscr{J}^{xx}S^{x}_{1_{000}}S^{x}_{2_{0\bar{1}0}} + \mathscr{J}^{yy}S^{y}_{1_{000}}S^{y}_{2_{0\bar{1}0}} + \mathscr{J}^{zz}S^{z}_{1_{000}}S^{z}_{2_{0\bar{1}0}} + (similar\ terms)\end{aligned} \tag{2.15}$$

where $S_{1_{000}}$ and $S_{2_{0\bar{1}0}}$ are chosen as an illustrative example. It is clear that if one enforces that $\mathscr{J}^{xx} = \mathscr{J}^{yy} = \mathscr{J}^{zz} = J$ that the sum $H^{xx} + H^{yy} + H^{zz}$ reduces to the familiar Heisenberg interaction in Eq. 1.4.

The DMI described in Eq. 1.5 requires an outer-product-like combination of terms. Considering the antisymmetric terms – H_a^{xy}, H_a^{yz}, and H_a^{zx} – one can show that the subtraction of cycled terms provides this structure. Once again summing these terms with the associated coupling constants one arrives at

$$\begin{aligned}\mathscr{J}_a^{xy}H_a^{xy} + \mathscr{J}_a^{yz}H_a^{yz} + \mathscr{J}_a^{zx}H_a^{zx} = \\ \sum_{\mathbf{n}} \mathscr{J}_a^{xy}\left(S^{x}_{1_{000}}S^{y}_{2_{0\bar{1}0}} - S^{y}_{1_{000}}S^{x}_{2_{0\bar{1}0}}\right) + \mathscr{J}_a^{yz}\left(S^{y}_{1_{000}}S^{z}_{2_{0\bar{1}0}} - S^{z}_{1_{000}}S^{y}_{2_{0\bar{1}0}}\right) \\ + \mathscr{J}_a^{zx}\left(S^{z}_{1_{000}}S^{x}_{2_{0\bar{1}0}} - S^{x}_{1_{000}}S^{z}_{2_{0\bar{1}0}}\right) + (similar\ terms) \\ = \sum_{\mathbf{n}} \mathscr{J}_a^{xy}\left(S^{x}_{1_{000}} \times S^{y}_{2_{0\bar{1}0}}\right)_z + \mathscr{J}_a^{yz}\left(S^{x}_{1_{000}} \times S^{y}_{2_{0\bar{1}0}}\right)_x \\ + \mathscr{J}_a^{zx}\left(S^{x}_{1_{000}} \times S^{y}_{2_{0\bar{1}0}}\right)_y + (similar\ terms) \\ = \sum_{\mathbf{n}} \{\mathscr{J}_a^{yz}, \mathscr{J}_a^{zx}, \mathscr{J}_a^{xy}\} \cdot \left(S^{x}_{1_{000}} \times S^{y}_{2_{0\bar{1}0}}\right) + (similar\ terms)\end{aligned} \tag{2.16}$$

Therefore, the vector-like quantity $\{\mathscr{J}_a^{yz}, \mathscr{J}_a^{zx}, \mathscr{J}_a^{xy}\}$ corresponds to the coupling vector $\mathbf{D}_{ij}$. It is known that these vectors will differ for any pair of nearest neighbours

and that they must be related through the symmetry of the lattice. Therefore, the form of this vector for all terms is constant to within cyclic permutations and parity. For example, one can show that for nearest neighbour pair $S_{1_{000}}$ and $S_{3_{\bar{1}\bar{1}0}}$ the vector is $\mathbf{D}_{1_{000}3_{\bar{1}\bar{1}0}} = \{\mathscr{J}_a^{zx}, -\mathscr{J}_a^{xy}, \mathscr{J}_a^{yz}\}$.

If it is assumed that this vector is oriented along one of the $\langle 111 \rangle$ directions, these terms are equal in magnitude $D = |\mathscr{J}_a^{xy}| = |\mathscr{J}_a^{yz}| = |\mathscr{J}_a^{zx}|$. Eq. 2.16 then reduces to a simplified form of DMI sometimes used in the analysis of helical structures

$$\mathscr{H}_{\text{DMI}} = \sum_{ij} \pm D\mathbf{S}_i \times \mathbf{S}_\mathbf{j}. \tag{2.17}$$

One may note that the DMI (like the Heisenberg interaction) may include interactions over greater distances than NN. The interactions presented here are truncated to NN interactions only. Therefore, the antisymmetric terms correspond with DMI only up to NN.

Apropos of the remaining terms: The anisotropy represents physical effects not modelled by exchange interactions. It is otherwise noteworthy that the symmetric terms H_s^{xy}, H_s^{yz}, and H_s^{zx} are not included in any of the above definitions. These interactions, which are allowed by the symmetry of the lattice, have not been considered previously.

Attempting to reduce the symmetric terms as done for the diagonal and antisymmetric terms is non-trivial. This term is not a standard product of vectors but may be represented as a vector product akin to the outer product utilizing addition in place of subtraction. In matrix form this is

$$\mathbf{v} \odot \mathbf{w} = \begin{pmatrix} 0 & v_z & v_y \\ v_z & 0 & v_x \\ v_y & v_x & 0 \end{pmatrix} \begin{pmatrix} w_x \\ w_y \\ w_z \end{pmatrix} = \begin{pmatrix} v_z w_y + v_y w_z \\ v_z w_x + v_x w_z \\ v_y w_x + v_x w_y \end{pmatrix} \tag{2.18}$$

where $\odot$ symbolizes a product that defines the symmetric terms. These terms have no direct analogue but are included here since they conform with the symmetry of the lattice.

This model may be compared to the majority of other models through the relations described here. Numerical and analytic studies of MnSi using microscopic models are generally constructed of these terms, with the exception of differing anisotropic terms [16, 31]. Likewise, continuous, spin density models typically use terms representing the same physical phenomena [13].

This model differs from these in three major facets: the inclusion of the symmetric terms, the allowance for the Heisenberg interactions to be separated among multiple constants, and the explicit restriction of the DM vector into three terms that obey the lattice symmetry.

2.3 Magnetic order parameters

2.3.1 Representations of a space group

In group-theoretical language, a matrix representation of a group is a group of square matrices that is homomorphic to that group. Irreducible representations (IRs) are the set of matrix representations that can not be subdivided into blocks of lower-order matrix representations.

In a space group, the symmetries of the group can be divided into the subgroups of translational symmetries and the point group. In general, the number of translational symmetries of a space group corresponds to the number of lattice sites, or equivalently, the number of points in the Brillouin zone. These translations are commutative which will allow for the same number of one-dimensional IRs of the form $\exp(i\mathbf{k}\cdot\mathbf{R}_n)$ for lattice translations $\mathbf{R}_n$. Therefore, the non-equivalent wavevectors $\mathbf{k} = \frac{2\pi}{N}\{g_x, g_y, g_z\}$ (for $g_i \in [0, N_i]$, where N_i is the number of unit cells in a given direction and $N = N_x N_y N_z$) of the Brillouin zone enumerate the representations.

Each wavevector will define some finite subgroup of the point group containing only transformations that leave $\mathbf{k}$ invariant. This subgroup is commonly called the *Little group of* $\mathbf{k}$. Therefore, the IRs of a space group with a wavevector $\mathbf{k}$ are associated with the IRs of the Little group of $\mathbf{k}$.

2.3.2 Magnetic order parameters

An order parameter (OP) is some quantity that can be measured to be zero in one phase and non-zero in another, such as the appearance of magnetization in the transition from paramagnetism to ferromagnetism. In other words, an OP may be used to determine the phase of a system, as well as any phase transitions. The change in OP can be either discontinuous, as seen in a first-order phase transition, or continuous, as seen in a second-order transition.

It is often the case that a phase transition will be associated with a broken symmetry. In helical magnets, translational symmetries are lost, and possibly inversion symmetry (in systems with inversion symmetries) or rotational symmetries. In a symmetry-breaking phase transition the distinguishing OP must belong to one of the IRs of the underlying space group.

In general, we can construct magnetic OPs associated with a given wavevector through Fourier transforms of the magnetic moments of the system. The order parameters are defined by the complex-valued vectors, $\mathbf{S}_{i_\mathbf{k}}$, produced by these transforms. The Fourier transform is defined as

$$S^{\kappa}_{i_\mathbf{k}} = \frac{1}{N^{\frac{1}{2}}} \sum_{\mathbf{n}} \exp\left(-i\mathbf{k}\cdot\mathbf{r}_\mathbf{n}\right) S^{\kappa}_{i_\mathbf{n}} \tag{2.19}$$

where $\mathbf{r}_n = a\mathbf{n}$ is the real space vector to the corner of the unit cell $\mathbf{n}$, and $\mathbf{k}$ is a wavevector belonging to the Brillouin zone. The inverse Fourier transform is given by

$$S^{\kappa}_{i_\mathbf{n}} = \frac{1}{N^{\frac{1}{2}}} \sum_{\mathbf{n}} \exp\left(i\mathbf{k}\cdot\mathbf{r}_\mathbf{n}\right) S^{\kappa}_{i_\mathbf{k}} \tag{2.20}$$

2.3.3 $k = 0$ example

As an example, consider a magnetic phase with $k = 0$: a structure with infinite wavelength, *i.e.*, invariant under translations. With a k=0 OP, no crystal lattice translation symmetries are broken; the Little group is simply the point group associated with the space group. For $P2_13$ the point group is T. The character table for this point group is shown in Table 2.3.

T					
IR \ Class	E	$4C_3$	$4C_3'$	$3C_2$	
A	1	1	1	1	
E	1 1	ϵ ϵ^2	ϵ^2 ϵ	1 1	
T	3	0	0	-1	
$S_{k=0}$	12	0	0	0	$A \bigoplus E \bigoplus 3T =$ $A \bigoplus E_+ \bigoplus E_- \bigoplus 3T$

Table 2.3: The character table of the crystallographic point group T. E may be reduced into 2 one-dimensional IRs $E = E_+ \bigoplus E_-$ with opposing chirality. The representation of the $4a$ Wyckoff position magnetic moments is given with its reduction to the IRs.

The representation of the magnetic moments of Wyckoff position $4a$ is a twelve-dimensional space – four magnetic moments with three spatial components. One can see from the transformations in Table 2.1 that the components form a basis for a (reducible) representation of the group T, $A \bigoplus E_+ \bigoplus E_- \bigoplus 3T$. This corresponds to 3 one-dimensional (A, E_+, and E_-) and 3 three-dimensional ($3T$) magnetic order parameters. The explicit form of the bases for the representations are

$$
\begin{aligned}
F_A &= \sqrt{3}^{-1} \sum_{\mathbf{n}} \big(S^x_{1_n} + S^y_{1_n} + S^z_{1_n} - S^x_{2_n} - S^y_{2_n} + S^z_{2_n} \\
&\qquad - S^x_{3_n} + S^y_{3_n} - S^z_{3_n} + S^x_{4_n} - S^y_{4_n} - S^z_{4_n} \big) \\
F_{E_+} &= \sqrt{\frac{2}{3}} \sum_{\mathbf{n}} \big(\epsilon^2 \left[-S^x_{1_n} + S^x_{2_n} + S^x_{3_n} - S^x_{4_n} \right] + \epsilon \left[-S^y_{1_n} + S^y_{2_n} \right. \\
&\qquad \left. - S^y_{3_n} + S^y_{4_n} \right] + \left[-S^z_{1_n} - S^z_{2_n} + S^z_{3_n} + S^z_{4_n} \right] \big) \\
F_{E_-} &= \sqrt{\frac{2}{3}} \sum_{\mathbf{n}} \big(\epsilon \left[-S^x_{1_n} + S^x_{2_n} + S^x_{3_n} - S^x_{4_n} \right] + \epsilon^2 \left[-S^y_{1_n} + S^y_{2_n} \right. \\
&\qquad \left. - S^y_{3_n} + S^y_{4_n} \right] + \left[-S^z_{1_n} - S^z_{2_n} + S^z_{3_n} + S^z_{4_n} \right] \big) \\
F_{T^{(1)}_x} &= \sqrt{3}^{-1} \sum_{\mathbf{n}} \big(S^x_{1_n} + S^y_{1_n} + S^z_{1_n} + S^x_{2_n} + S^y_{2_n} - S^z_{2_n} \\
&\qquad + S^x_{3_n} - S^y_{3_n} + S^z_{3_n} + S^x_{4_n} - S^y_{4_n} - S^z_{4_n} \big) \\
F_{T^{(1)}_y} &= \sqrt{3}^{-1} \sum_{\mathbf{n}} \big(S^x_{1_n} + S^y_{1_n} + S^z_{1_n} + S^x_{2_n} + S^y_{2_n} - S^z_{2_n} \\
&\qquad - S^x_{3_n} + S^y_{3_n} - S^z_{3_n} - S^x_{4_n} + S^y_{4_n} + S^z_{4_n} \big) \\
F_{T^{(1)}_z} &= \sqrt{3}^{-1} \sum_{\mathbf{n}} \big(S^x_{1_n} + S^y_{1_n} + S^z_{1_n} - S^x_{2_n} - S^y_{2_n} + S^z_{2_n} \\
&\qquad + S^x_{3_n} - S^y_{3_n} + S^z_{3_n} - S^x_{4_n} + S^y_{4_n} + S^z_{4_n} \big) \\
F_{T^{(2)}_x} &= \sqrt{6}^{-1} \sum_{\mathbf{n}} \big(S^x_{1_n} - 2S^y_{1_n} + S^z_{1_n} + S^x_{2_n} - 2S^y_{2_n} - S^z_{2_n} \\
&\qquad + S^x_{3_n} + 2S^y_{3_n} + S^z_{3_n} + S^x_{4_n} + 2S^y_{4_n} - S^z_{4_n} \big) \\
F_{T^{(2)}_y} &= \sqrt{6}^{-1} \sum_{\mathbf{n}} \big(-2S^x_{1_n} + S^y_{1_n} + S^z_{1_n} - 2S^x_{2_n} + S^y_{2_n} - S^z_{2_n} \\
&\qquad + 2S^x_{3_n} + S^y_{3_n} - S^z_{3_n} + 2S^x_{4_n} + S^y_{4_n} + S^z_{4_n} \big) \\
F_{T^{(2)}_z} &= \sqrt{6}^{-1} \sum_{\mathbf{n}} \big(S^x_{1_n} + S^y_{1_n} - 2S^z_{1_n} - S^x_{2_n} - S^y_{2_n} - 2S^z_{2_n} \\
&\qquad + S^x_{3_n} - S^y_{3_n} - 2S^z_{3_n} - S^x_{4_n} + S^y_{4_n} - 2S^z_{4_n} \big) \\
F_{T^{(3)}_x} &= \sqrt{2}^{-1} \sum_{\mathbf{n}} \big(S^x_{1_n} - S^z_{1_n} + S^x_{2_n} + S^z_{2_n} + S^x_{3_n} - S^z_{3_n} + S^x_{4_n} + S^z_{4_n} \big) \\
F_{T^{(3)}_y} &= \sqrt{2}^{-1} \sum_{\mathbf{n}} \big(-S^y_{1_n} + S^z_{1_n} - S^y_{2_n} - S^z_{2_n} - S^y_{3_n} - S^z_{3_n} - S^y_{4_n} + S^z_{4_n} \big) \\
F_{T^{(3)}_z} &= \sqrt{2}^{-1} \sum_{\mathbf{n}} \big(-S^x_{1_n} + S^y_{1_n} + S^x_{2_n} - S^y_{2_n} - S^x_{3_n} - S^y_{3_n} + S^x_{4_n} + S^y_{4_n} \big)
\end{aligned}
\tag{2.21}
$$

where, in this case, the Fourier transform does not introduce an exponential factor as $\exp(\pm i\mathbf{k}\cdot\mathbf{n}) = 1$ when $k = 0$. In general, these terms would carry this exponential term related to the Fourier transform.

For example, a $k = 0$ OP is found when describing the ferromagnetic state. If, for example, one chooses the ferromagnetic state – in which non-zero magnetization is the usual OP – aligned along the z axis the $F_{T_z^{(1)}}$ and $F_{T_z^{(2)}}$ OPs will necessarily be non-zero. As demonstrated, a given state may not be fully defined by all associated non-zero OPs. *Secondary order parameters* may appear if they result in a symmetry which forms a supergroup of one that is resultant of a *primary order parameter*.

2.3.4 k || ⟨111⟩ helical structure

In the helical structure the translational symmetry of the lattice is broken. The Little group associated with this wavevector is C_3 with character table given in Table 2.4.

C_3 Little Group				
Class / IR	E	C_3	C_3'	
A	1	1	1	
E	1 1	ϵ ϵ^2	ϵ^2 ϵ	
$S_{k=[111]}$	12	0	0	$4A \bigoplus 4E =$ $4A \bigoplus 4E_+ \bigoplus 4E_-$

Table 2.4: The character table of the Little group C_3. The representation of the $4a$ Wyckoff position magnetic moments is given with its reduction to the IRs.

The same process obtains that the magnetic moments belong to the direct sum of IRs $4A \bigoplus 4E_+ \bigoplus 4E_-$. In this case, separating the E term into the constituent parts is appropriate due to the inherent chirality of the helical structure. Therefore, there

are 12 one-dimensional magnetic order parameters belonging to $\mathbf{k} \parallel \langle 111 \rangle$. These are explicitly detailed in Eq. 2.22.

$$
\begin{aligned}
F_{A,1} &= 3^{-1} \sum_{\mathbf{n}} \exp(-i\mathbf{k}\cdot\mathbf{n}) \left(S^x_{1_n} + S^y_{1_n} + S^z_{1_n} \right) \\
F_{A,2} &= 3^{-1} \sum_{\mathbf{n}} \exp(-i\mathbf{k}\cdot\mathbf{n}) \left(S^x_{2_n} + S^z_{3_n} + S^y_{4_n} \right) \\
F_{A,3} &= 3^{-1} \sum_{\mathbf{n}} \exp(-i\mathbf{k}\cdot\mathbf{n}) \left(S^y_{2_n} + S^x_{3_n} + S^z_{4_n} \right) \\
F_{A,4} &= 3^{-1} \sum_{\mathbf{n}} \exp(-i\mathbf{k}\cdot\mathbf{n}) \left(S^z_{2_n} + S^y_{3_n} + S^x_{4_n} \right) \\
F_{E_+,1} &= 3^{-1} \sum_{\mathbf{n}} \exp(-i\mathbf{k}\cdot\mathbf{n}) \left(S^x_{1_n} + \epsilon S^y_{1_n} + \epsilon^2 S^z_{1_n} \right) \\
F_{E_+,2} &= 3^{-1} \sum_{\mathbf{n}} \exp(-i\mathbf{k}\cdot\mathbf{n}) \left(S^x_{2_n} + \epsilon S^y_{4_n} + \epsilon^2 S^z_{3_n} \right) \\
F_{E_+,3} &= 3^{-1} \sum_{\mathbf{n}} \exp(-i\mathbf{k}\cdot\mathbf{n}) \left(S^y_{2_n} + \epsilon S^z_{4_n} + \epsilon^2 S^x_{3_n} \right) \\
F_{E_+,4} &= 3^{-1} \sum_{\mathbf{n}} \exp(-i\mathbf{k}\cdot\mathbf{n}) \left(S^z_{2_n} + \epsilon S^x_{4_n} + \epsilon^2 S^y_{3_n} \right) \\
F_{E_-,1} &= 3^{-1} \sum_{\mathbf{n}} \exp(-i\mathbf{k}\cdot\mathbf{n}) \left(S^x_{1_n} + \epsilon^2 S^y_{1_n} + \epsilon S^z_{1_n} \right) \\
F_{E_-,2} &= 3^{-1} \sum_{\mathbf{n}} \exp(-i\mathbf{k}\cdot\mathbf{n}) \left(S^x_{2_n} + \epsilon^2 S^y_{4_n} + \epsilon S^z_{3_n} \right) \\
F_{E_-,3} &= 3^{-1} \sum_{\mathbf{n}} \exp(-i\mathbf{k}\cdot\mathbf{n}) \left(S^y_{2_n} + \epsilon^2 S^z_{4_n} + \epsilon S^x_{3_n} \right) \\
F_{E_-,4} &= 3^{-1} \sum_{\mathbf{n}} \exp(-i\mathbf{k}\cdot\mathbf{n}) \left(S^z_{2_n} + \epsilon^2 S^x_{4_n} + \epsilon S^y_{3_n} \right)
\end{aligned}
\tag{2.22}
$$

Using phenomenological observations, one can see that the helical state of MnSi can be either left- or right-handed along the positive $\langle 111 \rangle$ axes. Therefore, the helical state will belong to only one set of the F_{E_+} or F_{E_-} OPs, with the F_A terms as potential secondary OPs. Furthermore, these OPs can be used to determine the structure of the state, *e.g.*, the way the magnetic moments are aligned relative to one another in the lattice.

It is useful to invert these terms and represent the magnetic components as a combination of the order parameters. In doing this Eq. 2.22 is rewritten as

$$
\begin{aligned}
S^x_{1_n} &= \sum_k \exp(i\mathbf{k}\cdot\mathbf{n})(F_{A,1} + F_{E_+,1} + F_{E_-,1}) \\
S^y_{1_n} &= \sum_k \exp(i\mathbf{k}\cdot\mathbf{n})(F_{A,1} + \epsilon^2 F_{E_+,1} + \epsilon F_{E_-,1}) \\
S^z_{1_n} &= \sum_k \exp(i\mathbf{k}\cdot\mathbf{n})(F_{A,1} + \epsilon F_{E_+,1} + \epsilon^2 F_{E_-,1}) \\
S^x_{2_n} &= \sum_k \exp(i\mathbf{k}\cdot\mathbf{n})(F_{A,2} + F_{E_+,2} + F_{E_-,2}) \\
S^y_{2_n} &= \sum_k \exp(i\mathbf{k}\cdot\mathbf{n})(F_{A,3} + F_{E_+,3} + F_{E_-,3}) \\
S^z_{2_n} &= \sum_k \exp(i\mathbf{k}\cdot\mathbf{n})(F_{A,4} + F_{E_+,4} + F_{E_-,4}) \\
S^x_{3_n} &= \sum_k \exp(i\mathbf{k}\cdot\mathbf{n})(F_{A,3} + \epsilon F_{E_+,3} + \epsilon^2 F_{E_-,3}) \\
S^y_{3_n} &= \sum_k \exp(i\mathbf{k}\cdot\mathbf{n})(F_{A,4} + \epsilon F_{E_+,4} + \epsilon^2 F_{E_-,4}) \\
S^z_{3_n} &= \sum_k \exp(i\mathbf{k}\cdot\mathbf{n})(F_{A,2} + \epsilon F_{E_+,2} + \epsilon^2 F_{E_-,2}) \\
S^x_{4_n} &= \sum_k \exp(i\mathbf{k}\cdot\mathbf{n})(F_{A,4} + \epsilon^2 F_{E_+,4} + \epsilon F_{E_-,4}) \\
S^y_{4_n} &= \sum_k \exp(i\mathbf{k}\cdot\mathbf{n})(F_{A,2} + \epsilon^2 F_{E_+,2} + \epsilon F_{E_-,2}) \\
S^z_{4_n} &= \sum_k \exp(i\mathbf{k}\cdot\mathbf{n})(F_{A,3} + \epsilon^2 F_{E_+,3} + \epsilon F_{E_-,3})
\end{aligned}
\tag{2.23}
$$

where the inverse Fourier transform is used.

2.4 $\mathbf{k} \parallel \langle 111 \rangle$ order parameter analysis

It is important to determine which structures are permitted under each of the order parameters. However, to make such calculations tractable a number of structural assumptions must be made.

The first determination to be made is that the OPs are consistent with a specific chirality for the helical structure. The assumption is that this is a pure structure, and therefore only one $\mathbf{k}$ is non-zero. For simplicity, I also assume that the specific cubic

diagonal is [111]. Therefore, Eq. 2.23 reduces from a sum over all $\mathbf{k}$ to a single term with $\mathbf{k}$ || [111]. Note that now the exponential becomes

$$\exp(i\mathbf{k}\cdot\mathbf{n}) = \exp(ik(n_x + n_y + n_z)) = \exp(ikl) \tag{2.24}$$

where $k = |\mathbf{k}|$ and $l = n_x + n_y + n_z$ is unique to each two-dimensional layer along [111], although a single l represents both layers sharing a unit cell.

The OPs F_{E_-} and F_{E_+} describe helical symmetries of opposite handedness. To determine which of these OPs must be present one can set other OPs to zero and determine the system structure consistent with each. If $F_{E_+,1}$ is non-zero then the magnetic moment in position 1 will be

$$\mathbf{S}_{1,l} = \exp(ikl)\{F_{E_+,1}, \epsilon^2 F_{E_+,1}, \epsilon F_{E_+,1}\}. \tag{2.25}$$

If one takes the real component of this vector

$$\mathbf{S}_{1,l} = F_{E_+,1}\{\cos(kl), \cos(kl + \frac{2\pi}{3}), \cos(kl - \frac{2\pi}{3})\} \tag{2.26}$$

it can be verified that the real component of this vector will enforce a left-handed structure along $\mathbf{k}$. Therefore, it must be that F_{E_-} coincides with the right-handed structure. From this point on, I will analyze only the right-handed structure and assume all $F_{E_+,j} = 0$.

In the most general form, the magnetic moments of the four sublattices are described by

$$\begin{aligned}
S_{1,l} &= \exp(ikl)\{F_{A,1} + F_{E_-,1}, F_{A,1} + \epsilon^2 F_{E_-,1}, F_{A,1} + \epsilon F_{E_-,1}\} \\
S_{2,l} &= \exp(ikl)\{F_{A,2} + F_{E_-,2}, F_{A,3} + F_{E_-,3}, F_{A,4} + F_{E_-,4}\} \\
S_{3,l} &= \exp(ikl)\{F_{A,3} + \epsilon^2 F_{E_-,3}, F_{A,4} + \epsilon^2 F_{E_-,4}, F_{A,2} + \epsilon^2 F_{E_-,2}\} \\
S_{4,l} &= \exp(ikl)\{F_{A,4} + \epsilon F_{E_-,4}, F_{A,2} + \epsilon F_{E_-,2}, F_{A,3} + \epsilon F_{E_-,3}\}
\end{aligned} \tag{2.27}$$

These magnetic moments are always reported using only the real space component. However, it is important to recognize that these are complex-valued vectors, with

some magnitude aligned with the imaginary axes of the Argand plane.

In this thesis, I will discuss computational simulations using the model developed here in Ch. 4. In these simulations, I will assume that all magnetic moments have fixed, equal magnitude. A set of general criteria coinciding with this assumption for any number of OPs is presented in Appendix B. As an example, it can be shown that this condition requires that $F_{A,1} = 0$ for a pure helical structure. However, the parameter space generated from these criteria is large. In this section, I will present only a subset of these that are sufficient to explain structures observed in the computational studies with the assumptions regarding each.

For a system with no applied field, the physically observed features which should be modelled are

1. All magnetic moments lie in the plane with normal along $\langle 111 \rangle$.
2. Layers 1 and 2 are ferromagnetically aligned, *i.e.*, sublattices 2, 3, and 4 are ferromagnetically aligned in any layer, l.
3. Within a unit cell, magnetic moments in sublattice 1 are a consistent measurable angle, α, from the other three. α should be allowed to differ, by an amount ϕ, from the value obtainable using the wavevector and distance between layers 1 and 2.

Approaching these in order, one may find that if it is assumed that all $F_{A,j} = 0$ then the magnetic moments of sublattice 1 must be perpendicular to the [111] direction. If one also chooses $F_{E_-,2-4} = F_{E_-}$ the magnitude will be constant for eight pairs of relative phases. These are

$$\phi_3 = \phi_2 \pm \frac{n\pi}{3} \qquad\qquad \phi_4 = \phi_2 \mp \frac{n\pi}{3} + m\pi \tag{2.28}$$

for $n = 1, 2$ and $m = 0, 1$. Here, ϕ_j is the phase of OP $F_{E_-,j} = |F_{E_-,j}| \exp(i\phi_j)$.

If the case in which relative phases are equidistant (*i.e.*, all phases differ by $\frac{2\pi}{3}$) is chosen then all magnetic moments will necessarily be perpendicular to the [111] direction and form a right-handed helix. Further, this choice of phases will enforce that the magnetic moments in all layers are ferromagnetically aligned.

If one relaxes the first and second physical properties, then a helical structure with out-of-plane canting may be observed. In this case, all considered OPs may vary. It is useful to choose only the values necessary to produce predictable structures. In the case of canting, a constant $F_{A,j} = F_A$ will allow canting along the [111] axis and coincide with a conical structure. However, with the assumption of equidistant phases associated with the $F_{E_-,j}$ terms, constant non-zero values of $F_{A,j}$ would result in non-constant magnitude of the magnetic moments. These are, therefore, incompatible assumptions.

For more complex features, such as a canting of the entire plane of the helices for a given sublattice, *i.e.*, canting of the normal vector, one can instead choose different relative phases of equal magnitude F_{E_-} terms. Each pair will produce canting towards different directions, defined by a *canted normal vector* for each sublattice $\mathbf{C}_i = [ABC]$, as tabulated in Table 2.5. It is apparent in this that the canting of this kind in each sublattice should vary in direction. A graphical representation of an example canted normal vector can be viewed in Fig. 2.1. If the assumption that there is a constant magnitude F_{E_-} is removed, canting of this form in any direction is possible and is controlled by the ratio of these magnitudes. Further, this form of canting matches with that found in other analytic results referred to in Section 1.4.2.

Note that under the assumptions presented here, a significant canting can not be produced in $\mathbf{S}_1$. Significant canting of this sublattice would only be provoked through non-zero $F_{A,1}$, which is not allowed by the assumption of constant magnitude. Therefore, we should expect no canting in this sublattice.

$\phi_\mathbf{3} - \phi_\mathbf{2}$	$\phi_\mathbf{4} - \phi_\mathbf{2}$	$\mathbf{C}_2 = [ABC]$	$\mathbf{C}_3 = [BCA]$	$\mathbf{C}_4 = [CAB]$
$\pm\frac{2\pi}{3}$	$\mp\frac{2\pi}{3}$	[111]	[111]	[111]
$\pm\frac{\pi}{3}$	$\mp\frac{\pi}{3}$	$[\bar{1}11]$	$[11\bar{1}]$	$[1\bar{1}1]$
$\pm\frac{\pi}{3}$	$\pm\frac{2\pi}{3}$	$[1\bar{1}1]$	$[\bar{1}11]$	$[11\bar{1}]$
$\pm\frac{2\pi}{3}$	$\pm\frac{\pi}{3}$	$[11\bar{1}]$	$[1\bar{1}1]$	$[\bar{1}11]$

Table 2.5: The direction of allowable canting for all phase choices of the $F_{E_-,j}$ OPs with the equal magnitude assumption. Canted normal vectors are equivalent to their vector inverses. A canted normal vector of [111] or $[\bar{1}\bar{1}\bar{1}]$ indicates no canting. All choices cant each ion toward a different cubic diagonal.

Finally, the presence of a non-zero phase shift, ϕ, as defined in Fig. 1.8, is necessarily allowed under any of these structures through an additive phase to the OPs which construct the magnetic moments of layer 2.

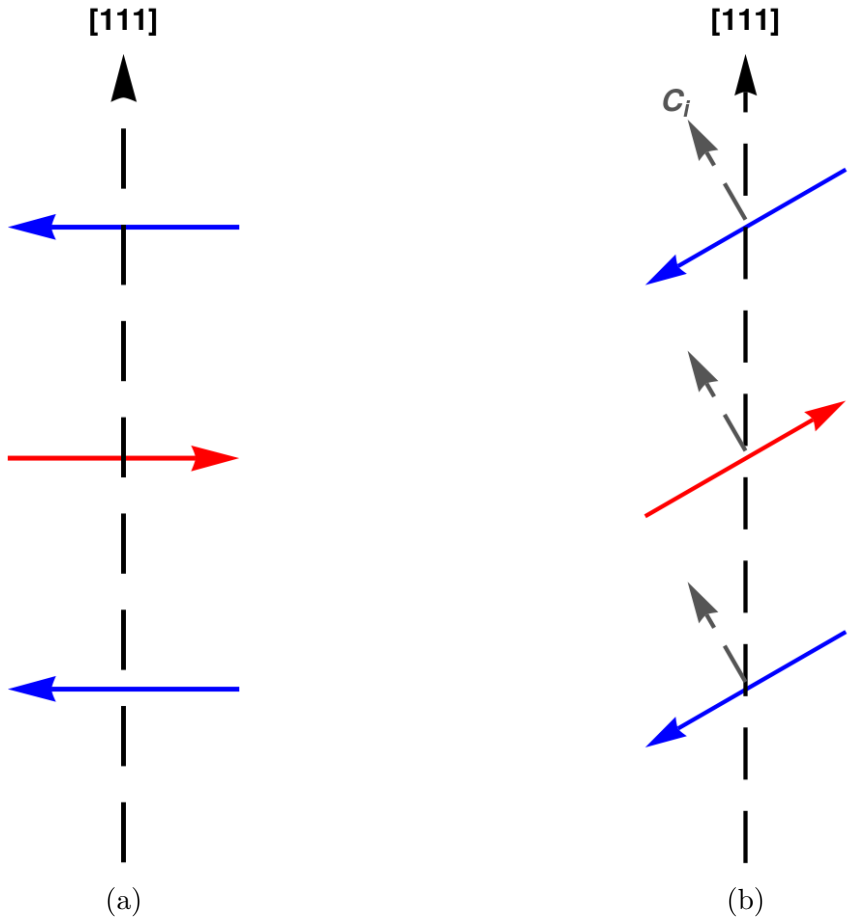


Figure 2.1: The helical plane in relation to [111] with (a) no canting and (b) canted normal vector $\mathbf{C}_i$.

This analysis confirms that all desired features of the final magnetic state are achievable under the assumptions that $\mathbf{k} \parallel [111]$ and constant magnetic moment magnitude. Further, these properties are relatively easy to determine from the measured order parameters – as defined in Eq. 2.23 – alone.

Chapter 3

Computational Methodology

3.1 Effective Field Method

The Effective Field Method (EFM) is a method for determining structures as $T \to 0$. EFM is used to locate local minima in classical systems with pairwise interactions [33]. The interactions described in the model are all bilinear in nature and may be modelled through this simulation. Furthermore, this allows for other interactions such as the Zeeman term, allowing for simulations to be performed with an externally applied magnetic field.

EFM is an iterative algorithm. In every step, each position is chosen in a randomized order – reducing systematic errors arising from the ordering of the moments – and the moment associated with each position is rotated toward their instantaneous, effective field $\mathbf{H}_i$. This field is defined by the sum of all interactions experienced by the selected magnetic moment

$$H_i^\alpha = -\sum_{j,\beta} \mathscr{J}_{i,j}^{\alpha,\beta} \mathscr{S}_j^\beta \tag{3.1}$$

where i, j represent a pair of positions, α, β represent global coordinates, $\mathscr{J}_{i,j}^{\alpha,\beta}$ is the coupling constant of the interaction, and $\mathscr{S}_j^\beta$ is one of the other interactants with which the magnetic moment interacts, *e.g.*, another magnetic moment or a field. This defines the energy of a given magnetic moment as

$$E_i = -\mathbf{S}_i \cdot \mathbf{H}_i. \tag{3.2}$$

It is evident that the outlined process will always reduce energy. The change in energy for each rotation directly aligning a moment with its local effective field is

$$\Delta E_i = -(1 - \cos\theta_i)\,|\mathbf{H_i}| \tag{3.3}$$

where θ_i is the angle between the moment and the field. If this process is allowed to continue until the change in energy is within the desired precision, then a local minimum has been found.

This means that only a local minimum can be guaranteed through a single use of this method. To help ensure that a "ground state" (*i.e.*, global minimum) is found one can run it a large number of times using randomized starting configurations. This sampling of space is then filtered for a subset of configurations associated with the minimum values.

3.1.1 EFM algorithm

The EFM algorithm is:

1. Randomly generate (or import) an initial configuration for the system, $S = \{\mathbf{S}_1, \mathbf{S}_2, ..., \mathbf{S}_n\}$.
2. Randomly order the elements of S
3. Select the next element, $\mathbf{S}_i$, of S and calculate the local effective field $\mathbf{H}_i$ at that position.
4. Align $\mathbf{S}_i$ with the local effective $\mathbf{H}_i$.

Repeating steps $3-4$ over the full configuration S is a single EFM step. The entire process is repeated many times with many random starting configurations to conform to the condition that many minima are found. In this thesis, a typical number of initial configurations used for a single simulation is 2500. The output of an EFM simulation is a set of magnetic moment configurations.

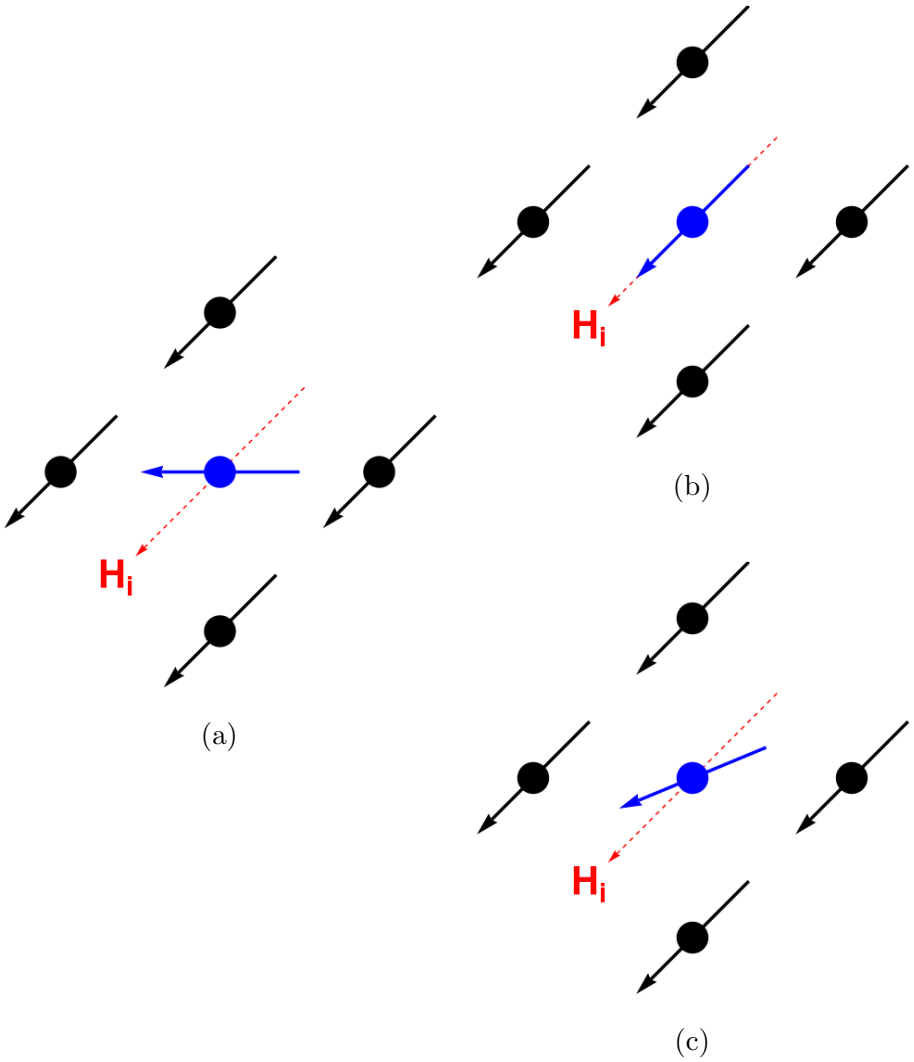


Figure 3.1: An example of the alignment of a magnetic moment (blue) with the local effective field (red, dashed) produced by interactions with its neighbour in the (b) Effective Field Method and (c) Progressive Effective Field Method. In (c) the magnetic moment is rotated through half of the angle between itself and the effective field.

3.1.2 Progressive EFM

The method, as described, may involve large, discontinuous jumps between orientations at any step. That is, the change in orientation is discrete and, in the worst cases, extreme. This effect could increase the probability of a configuration progressing toward a local minimum (that is not the global minimum) when a rotation results in a state space which can not reach a lower energy state with any single rotation, but may do so with multiple rotations. An extreme example of this is presented in Fig. 3.2. Due to this – as well as in the interest of creating a more "physical", continuous process – a moment may be rotated only a portion of the angle towards its local effective field. In the simulations presented here, the altered process will still result in a reduction of energy with each rotation. However, the reduced step size will allow the simulation to avoid local minima more efficiently. In the worst case, the same final states will be achieved with only the addition a proportional number of EFM steps since the standard EFM algorithm is the same as multiple partial rotations in the same direction.

The EFM code that was used in this thesis was prepared using the FORTRAN language and is available in Appendix D.

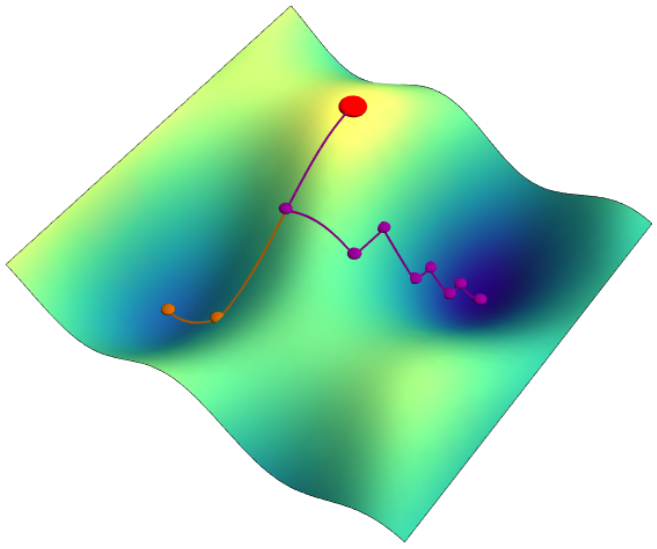


Figure 3.2: An illustrative example of the progression of both the Effective Field Method (orange) and the Progressive Effective Field Method (Purple) in a fictitious energy field. In this example the system starts in the state coloured red and progresses to a local minimum (EFM) and the global minimum (PEFM).

3.2 Data analysis

This computational methodology was chosen with the goal of producing the magnetic structure of the system with varying model parameters. The data collected here will be predominantly qualitative. The major qualities of interest are the relative orientations of the magnetic moments which will be represented as spherical angles relative to $\mathbf{k} \parallel [111]$. These angles are γ, the out-of-plane angle to the plane with normal vector $\mathbf{k}$ (or, equivalently, the angle between $\mathbf{k}$ and the canted normal vector $\mathbf{C}_i$), and θ, the azimuthal angle relative to $[\bar{1}10]$. A graphical representation of these angles is presented in Fig. 3.3. All magnetic moments are normalized to 1 and varying magnitude will not be considered. Therefore, all measurements concerning relative orientations will be derived from these angles or combinations thereof.

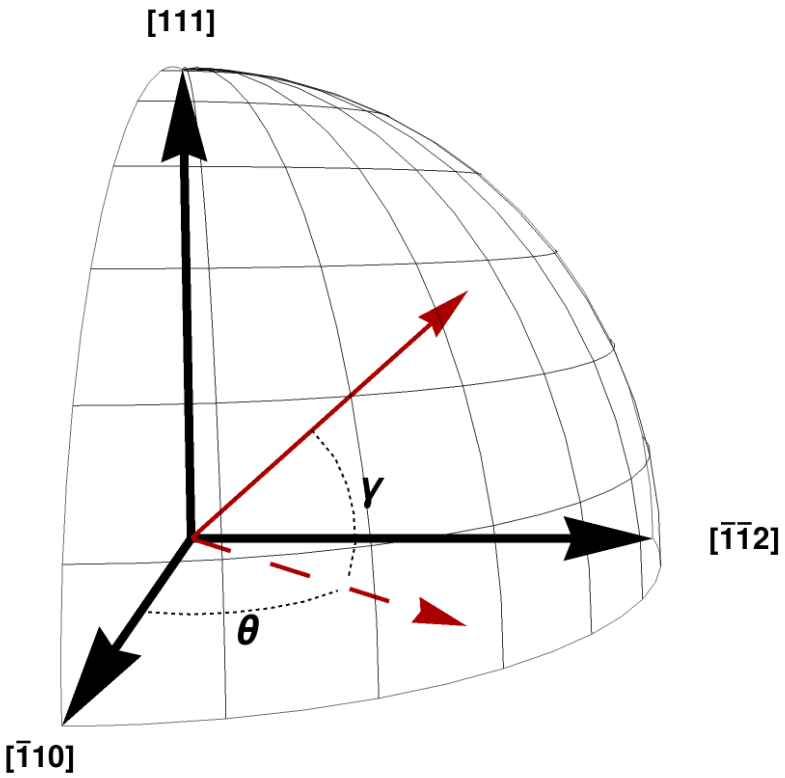


Figure 3.3: The spherical angles, γ and θ, with respect to zenith [111]. γ represents the out-of-plane angle of the magnetic moment and is equivalent to the altitude angle. θ represents the azimuthal angle relative to planar vector $[\bar{1}10]$.

The angle γ itself will be presented using combinations of two forms: combining two-dimensional layers to a single, average magnetic moment and combining sublattices to a single, average magnetic moment. These combinations will be utilized to derive all presented quantities.

Among the values that are derived from these is the wavevector magnitude $|\mathbf{k}|$. The wavevector magnitude is measured in this way due to the finite system size. For all lattice sizes that require a reasonable amount of computing time the Brillouin zone of the lattice is coarse, with $|\mathbf{k}|$ being restricted to large areas of the zone. Long-wavelength features, as helical structures tend to be, have wavevectors that may be indistinguishable from $k = 0$ when using discrete Fourier transforms to determine $|\mathbf{k}|$.

The data collected here will be directional, *i.e.*, made of angles $\theta \in [0, 2\pi)$. One may note that, in the case that the averaged measurements occur over the branch cut, $0 = 2\pi$, the mean and variance measurements will need to account for this discontinuity. For this, I employ *directional statistics*. The processes derived in directional statistics and used for data analysis measurements are fully presented in Appendix C.

3.2.1 Combining minima

The EFM algorithm requires that a large number of samples be generated to ensure a minimum is found. In this process, a large amount of useful data may be discarded or used independently of other samples. To produce more accurate data a sample of $1-2\%$ of the most extreme minima produced from the EFM procedure will be pooled for appropriate measurements.

All data analysis is performed using Python code available in Appendix E.

Chapter 4

Computational Results

The results of the computational studies of this model will be presented in three parts: First, the reduced model with coupling D and J discussed in Section 2.2 will be explored. Following this will be the examination of simulations in which each model coupling constant is varied individually about a selected helical state with DMI strength $D = 0.50$. Finally, a short discussion of the reaction of the system to an applied field will be provided.

All tests will be conducted without periodic boundary conditions (to allow for the incommensurate structure) using a $23 \times 23 \times 23$ lattice. This lattice size is chosen as it is large enough to contain one helical wavelength with the wavevector $\mathbf{k} = 0.035\text{Å}^{-1}$ observed experimentally. This is analogous to a thin film and will allow for measurable edge effects.

4.1 Varying the Dzyaloshinskii-Moriya interaction strength D

As a test of the model, and to select a benchmark state for other tests, the reduction of the model to simple Heisenberg and DM-like terms, as discussed in Section 2.2, is considered. For this, the value of $J = 1$ is chosen for all diagonal, Heisenberg-like terms. The strength of the antisymmetric, DM-like terms will be set to constant D and varied relative to J. All off-diagonal symmetric terms are assumed to be vanishing and the anisotropy coupling constants are set at the relatively large values

of $\mathscr{J}_{A2} = -0.50$ and $\mathscr{J}_{A4} = 0.50$, which will strongly favour the $\mathbf{k} \parallel [111]$ direction.

4.1.1 $\mathbf{D}_{ij}$ direction

First, it is important to report the effect of varying the direction of the $\mathbf{D}_{ij}$ vectors. All instances of this vector reported will be in reference to the $\mathbf{D}_{1_{000}2_{0\bar{1}0}}$ vector unless otherwise noted. As previously mentioned, other choices of i and j must represent different vectors which are symmetric in the lattice.

In studies of two-dimensional microscopic systems with DMI, it was noted that the orientation of the DMI vectors can have large effects on the overall structure of the system [34]. Specifically, they report different triangular lattices that can be compared to the two-dimensional layers here. The resultant structures produced by aligning the $\mathbf{D}_{ij}$ vector along different cubic diagonals in the currently considered, three-dimensional system are tabulated in Table 4.1.

$\mathbf{D}_{ij} \parallel$	**Structure**
$[111]$	Right-handed helix
$[\bar{1}11]$	Left-handed helix
$[1\bar{1}1]$	Ferromagnetic
$[11\bar{1}]$	Right-handed helix

Table 4.1: The magnetic structure of the lattice for differing $\mathbf{D}_{ij}$ directions. Antiparallel $\mathbf{D}_{ij}$ vectors are associated with opposite handedness structures. That is, $\mathbf{D}_{ij}$ is an odd function of $\mathbf{k}$, $\mathbf{D}_{ij}(\mathbf{k}) = -\mathbf{D}_{ij}(-\mathbf{k})$. For example, $\mathbf{D}_{ij} \parallel [111]$ and $\mathbf{D}_{ij} \parallel [\bar{1}\bar{1}\bar{1}]$ represent right and left-handed helical structures, respectively.

The expected result of this test is interpreted from the relation

$$\mathbf{k} = \frac{2(D_z - 2D_x - D_y)}{3J} \tag{4.1}$$

from Chizhikov *et al* [31]. D_x, D_y, and D_z represent the components of $\mathbf{D}_{ij}$. I note that the relationship is written differently in the source. That is due to the choice of $\mathbf{D}_{ij}$ varying from the one chosen here. These choices are symmetrically equivalent

within the lattice, with the C_3^- transformation relating the two. I have applied that transformation here.

In this analysis, I am assuming that all components of the $\mathbf{D}_{ij}$ vector are of the same magnitude, and therefore only differ in sign. Clearly, if $\pm D_x = \mp D_y = \pm D_z$, the relationship predicts a $k = 0$ wavevector. In all other cases, the sign of D_x will match the sign of $\mathbf{k}$. It is also apparent that, if this relationship holds, inverting the vector will simply invert the sign of $\mathbf{k}$. From these results, I conjecture that differing $\mathbf{D}_{ij}$ directions correspond to superpositions of helical structures, as reported in the two-dimensional case. This corresponds to a mixing of F_{E_-} and F_{E_+} order parameters.

It is important to note that the DMI vectors need not be oriented along the cubic diagonals. For example, previous results show that the DMI vectors in MnSi are oriented along the bond directions and not the cubic diagonals [35], *e.g.*,

$$\mathbf{D}_{ij} \;||\; \{\frac{1}{2} - 2x, 1 - 2x, \frac{1}{2}\}. \tag{4.2}$$

This difference would correlate to a variation in the value of the antisymmetric coupling constants, as is explored in Section 4.2. For simplicity, I choose the direction $\mathbf{D}_{ij}$ || [111] (right-handed helix) for all future simulations. With this assumption, the expected result reduces to a linear form

$$|\mathbf{k}| = \frac{4D}{3} \tag{4.3}$$

with $J = 1$ enforced.

4.1.2 Wavevector magnitude |k|

The wavevector is a measure of both the direction and the periodicity of the helical structure. I am assuming that all wavevectors are aligned such that $\mathbf{k}$ || [111] and their magnitude is derived from the periodicity of the lattice as

$$|\mathbf{k}| = \frac{1}{\bar{\lambda}} = \frac{\bar{\theta}}{2\pi} \tag{4.4}$$

where $\bar{\theta}$ is the average angle change between equivalent lattice positions in unit cells

located at $\mathbf{n}$ and $\mathbf{n}+\{1,1,1\}$. Therefore, all values are measured in the units of inverse unit cell diagonals. The wavevector does not vary between layers or sublattices and, therefore, only one is reported.

Measurements of this kind for the three-dimensional system are displayed in Fig. 4.1. In this figure, D and $\mathbf{k}$ are both represented by absolute values as changing the sign of D corresponds with an equivalent sign change of $\mathbf{k}$. In fact, I have observed that a sign change of D corresponds with an equivalent structure of opposite handedness in all measured quantities. Hereafter all values of D will be positive.

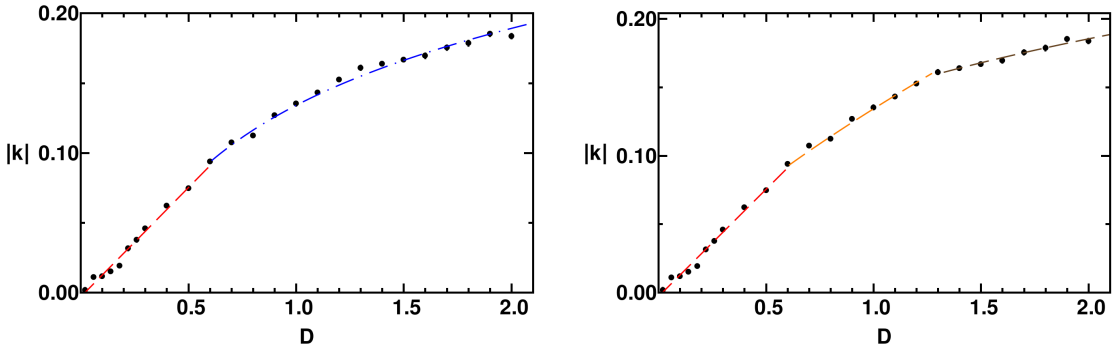


Figure 4.1: Average wavevector magnitude as a function of D. The system behaviour varies in regions. In the region $D < 0.6$ a linear fit is denoted (red; dashed). The region $D > 0.6$ is fit to (a) a logarithmic function (blue; dot-dashed). (b) two power-law fits (orange, brown; dot-dashed). Error bars are within the size of the markers.

It is clear that the linear relationship predicted in Eq. 4.3 does not hold for large values of D. Alternatively, it was reported in the two-dimensional numerical study that $|\mathbf{k}|$ behaved differently in distinct domains of D. This behaviour matches the presented results, with two distinct regions: linear and sub-linear growth. It was assumed in two-dimensions that these regions fit distinct power laws of the form

$$|\mathbf{k}| \propto D^{\beta}. \tag{4.5}$$

with $\beta = 1$ for $D < 0.3$, $\beta \approx 0.8$ for $0.3 < D < 1.0$, and $\beta \to 0$ for $D > 1.5$. It is apparent that a linear relationship remains in the three-dimensional system for the approximate range $D < 0.6$. Further, the results from simulations above this value could correspond to a decaying fitting parameter β: $\beta \approx 0.73$ for $0.6 < D < 1.3$ and $\beta \approx 0.35$ for $D > 1.3$. These boundaries are chosen arbitrarily from observation of the

data. However, a decaying fitting parameter β may also be reasonably approximated by a logarithm. If one assumes that the relationship between wavevector magnitude and DMI strength follows a logarithmic scaling they may use a fit of the form

$$|\mathbf{k}| = A + B\log(D) \tag{4.6}$$

where A and B are the fitting parameters. In this case, for all values $D > 0.6$, a reasonable fit is found to be $A \approx 0.13$, $B \approx 0.08$. All fits discussed here can be viewed in the appropriate log-log or semi-log plot in Fig. 4.2.

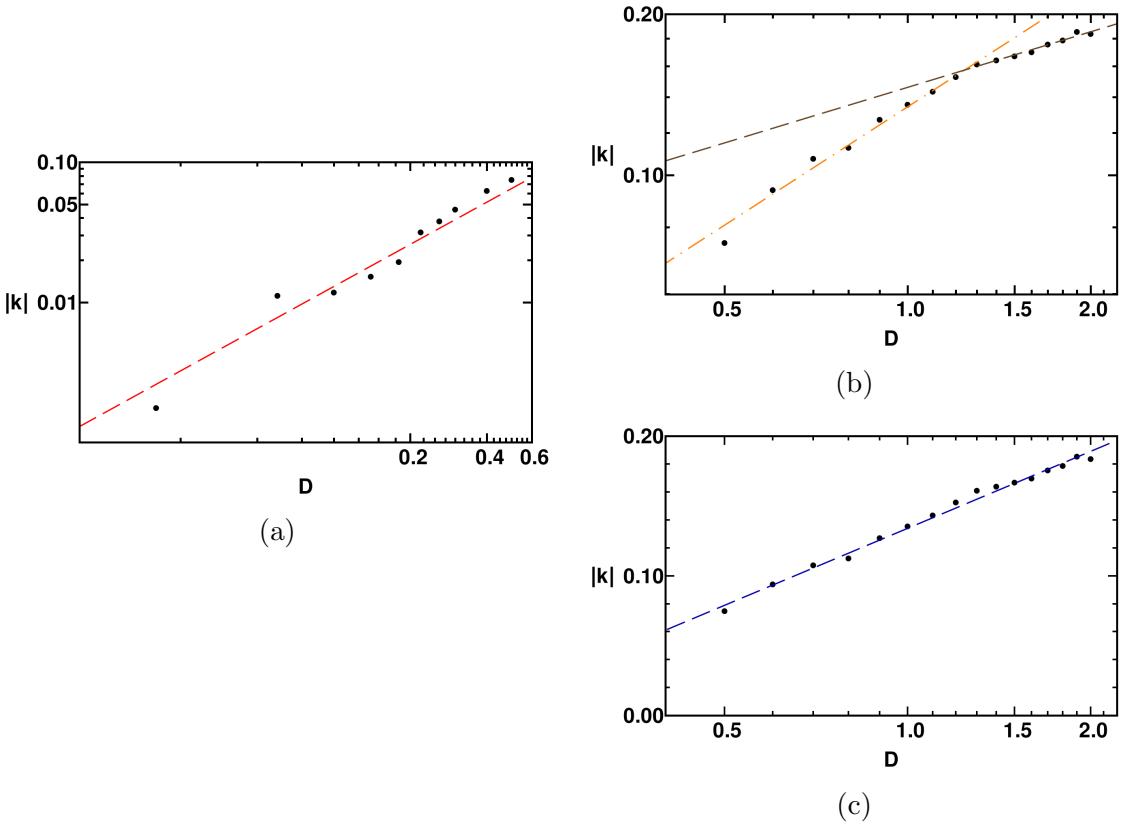


Figure 4.2: Plots of the wavevector magnitude in region (a) $0 < D < 0.6$ on a log-log scale with a linear fit; (b) $0.5 < D < 2.0$ on a log-log scale with two power law fits corresponding with $\beta = 0.73$ (orange; dot-dashed) and $\beta = 0.35$ (brown; dashed); (c) $0.5 < D < 2.0$ on a log-linear scale with logarithmic fit $0.13 + 0.08\log(D)$.

4.1.3 Anomalous phase ϕ

The anomalous phase, as defined in Section 1.4, exhibits similar behaviour as observed with $|\mathbf{k}|$. The angle exhibits distinct domains of variation which goes from approximately linear in $D < 0.6$ to sub-linear for $D > 0.6$. This is displayed in Fig. 4.3.

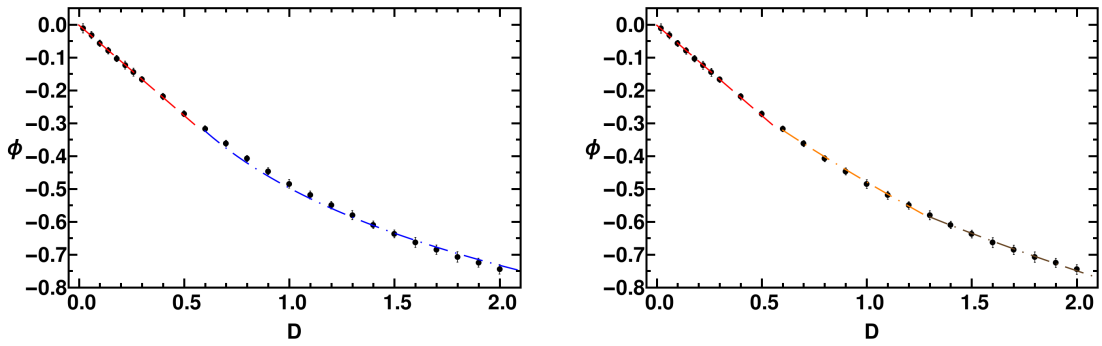


Figure 4.3: Anomalous phase ϕ as a function of D. Similar to Fig. 4.1, the system varies in regions. In the region $D < 0.6$, a linear fit is denoted with the red line. The region $D > 0.6$ is fit to (a) a logarithmic fit (blue; dot-dashed); (b) two power-law fits (orange, brown; dot-dashed). Unseen error bars are within the size of the markers.

As observed in $|\mathbf{k}|$, the dependence of the value on D decays for large values. A linear, logarithmic, and power fits are reported in Fig. 4.4. The reported logarithmic fit, $0.49 + 0.36 \log(D)$, appears to more closely approximate this parameter than the wavevector. Considering the power-law fit, the exponent β varies less between the two regions than in the $|\mathbf{k}|$ case. However, the appearance of a clear variation in this parameter in the same regions for both measurements suggests that the power-law fit variation is appropriate. The variation seen in these parameters – both linear-to-sublinear and within the sublinear region – indicates some change in the underlying structure of the lattice, *i.e.*, a phase transition. In both cases, this should be indicated by the vanishing or appearance of one of the magnetic order parameters. Particularly, one should expect the appearance of a pure helical structure with a specific chirality at some value of D. Section 4.1.5 will discuss this more carefully.

The values of ϕ observed here are very large in comparison to those reported by Dalmas de Réotier *et al* [28]. This discrepancy is confirmed by the linear relationship

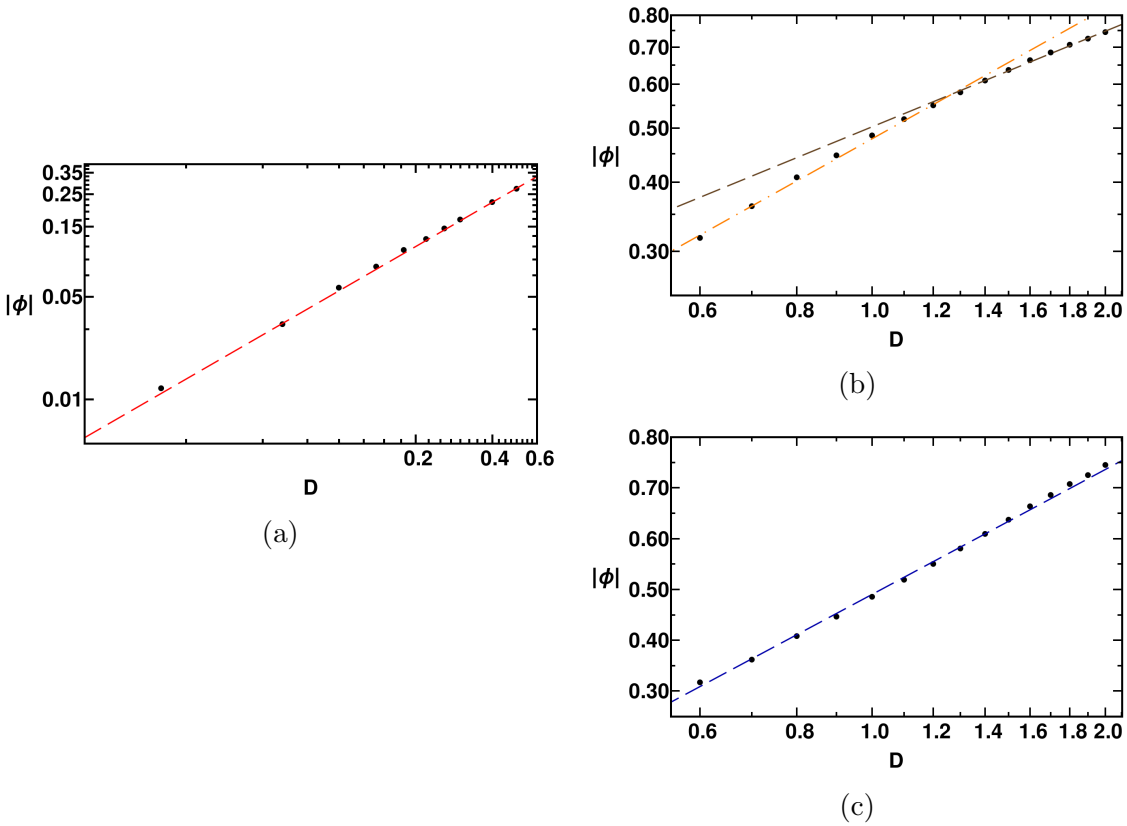


Figure 4.4: Plots of the anomalous phase magnitude in region (a) $0 < D < 0.6$ on a log-log scale with a linear fit and (b) $0.6 < D < 2.0$ on a log-log scale with two power law fits corresponding to $\beta = 0.78$ (orange; dot-dashed) and $\beta = 0.57$ (brown; dashed); (c) $0.6 < D < 2.0$ on a log-linear scale with logarithmic fit $0.49 + 0.36 \log(D)$.

$$\phi \propto -\frac{D_y + D_z}{J} = -2D \tag{4.7}$$

where it is apparent that values of $D \approx J$ will necessarily produce large values of ϕ. If this prediction is consistent, which it appears to be for all $D < 0.6$, then a different choice of $\mathbf{D}_{ij}$ magnitude or direction could produce appropriate ϕ for those observations. Further, it is noted in the experimental source that sublattice 1 nearly aligns with those of the second nearest plane, which is approximately observed here. Therefore, the relatively large wavevector $\mathbf{k}$ describing the helix here will require ϕ to be large, as well. It is also possible that the introduction of other interactions or thermal effects could act to suppress this value. Nonetheless, these results provide an adequate test of the effect of model terms on this feature.

To make it easier to understand the significance of these results – and to compare with those reported from the experimental results – Fig. 4.5 presents a projection of the helices described by **k** and ϕ for a number of simulations.

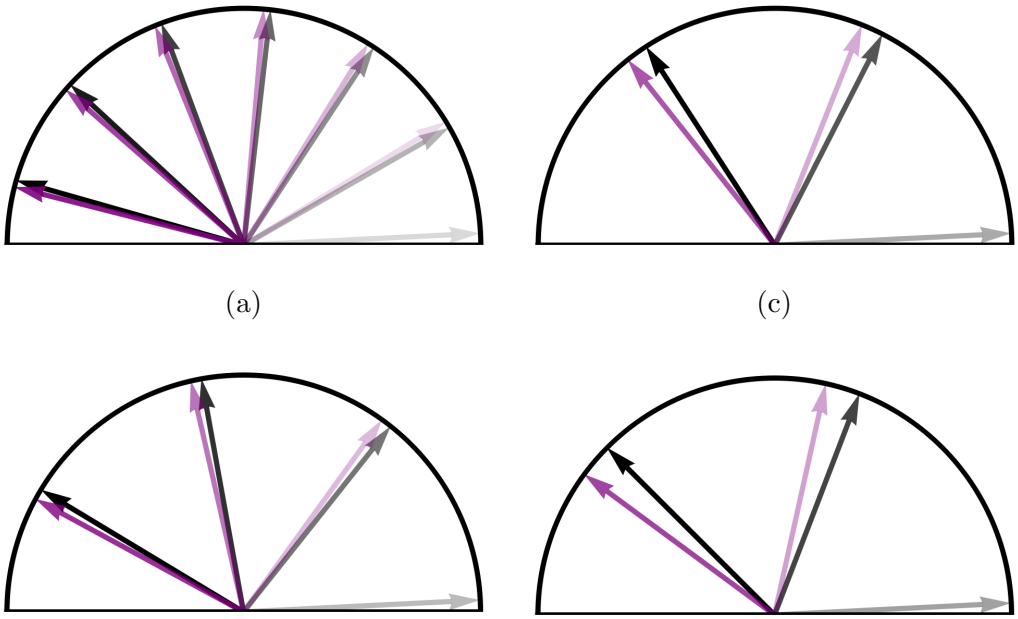


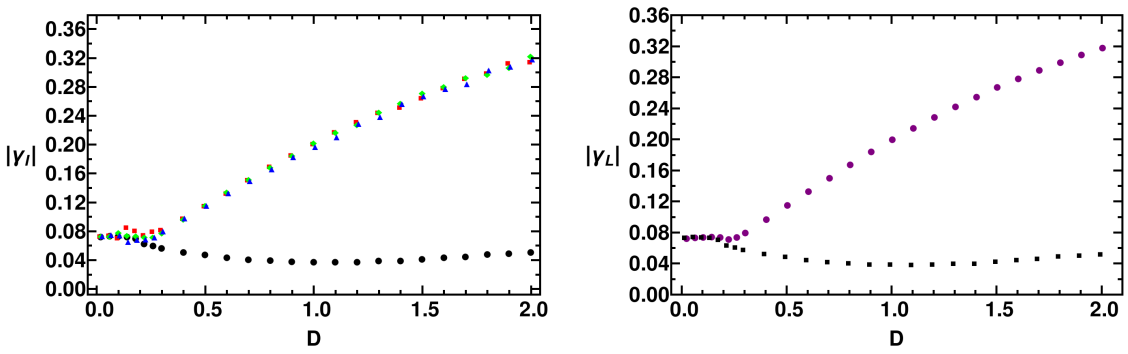
Figure 4.5: Two-dimensional projections of the average in-plane orientation of layers 1 (black) and 2 (purple), produced from the average wavelength and ϕ, as viewed from the [111] direction for (a) $D = 0.50$ (b) $D = 1.00$ (c) $D = 1.50$ (d) $D = 2.00$. Out-of-plane canting is ignored in these depictions. The opacity of a vector represents the layer position along the [111] axis, with darker vectors further along this direction. The position of layer 2 predicted from the distance between layers is roughly centred between the preceding and succeeding layer 1. The value of ϕ is such that it has rotated beyond the position of the succeeding layer.

4.1.4 Out-of-plane angle γ

While considering the OPs of a helical phase it was noted that it was not necessary that layer 2 remain in-plane, *i.e.*, $\gamma \approx 0$. The average measure of these angles over the entire lattice is presented in Fig. 4.6 for both averaged sublattices (γ_{I}) and averaged layers (γ_{L}).

In these measurements, there is a clear transition at $D \approx 0.3$, smaller than changes

in heretofore discussed parameters. Below this value, the absolute out-of-plane angle is consistent between all sublattices and layers. Above this, sublattices $2, 3$ and 4 – *i.e.*, those which comprise layer 2 – cant further out-of-plane as D increases. The canting angle for these three remains roughly consistent. In this same region $\gamma_{\mathrm{I}}^{(1)}$ appears to lessen slightly until a slow growth begins around $D = 1.0$. Consistency between the three magnetic sites contained within layer 2 is seen in all measurements and therefore only the average layer canting will be reported.



(a) (b)

Figure 4.6: The average magnitude of the out-of-plane angle γ for both (a) individual sublattices ($\gamma_I^{(i)}$) and (b) distinct layers ($\gamma_L^{(i)}$). In both cases, all ions in a given layer remain consistent. Therefore, only γ_L will be reported after this.

The canting observed in this phase is not consistent with a conical phase, in which γ would be expected to remain relatively constant throughout the lattice. Instead, γ undergoes sinusoidal variation along the [111] axis and is, therefore, better described by a canted normal vector as discussed in Section 2.4. This can be viewed in Fig. 4.7.

The orientation of this kind of canting is predicted by

$$\mathbf{C}_i \sim || \ (\tau_i). \tag{4.8}$$

where the vectors τ_i associated with each sublattice are

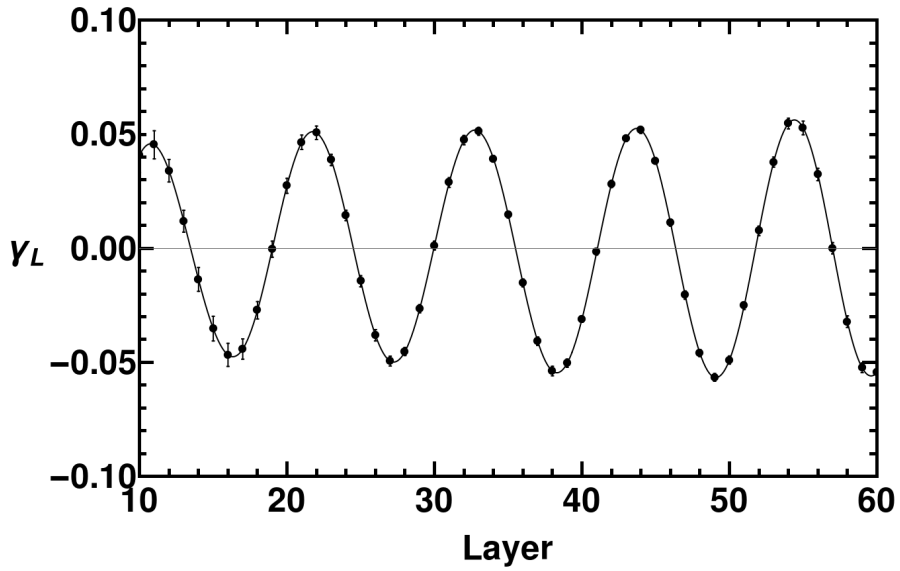


Figure 4.7: An example of the out-of-plane canting of sublattice 1 along the [111] direction for a portion of a selected $D = 0.60$ simulation. Error bars are taken from the error in the absolute value measurements. An interpolated line is included as a guide for the eye.

$$\begin{aligned} \tau_1 &= \{1, 1, 1\} \\ \tau_2 &= \{-1, 1, -1\} \\ \tau_3 &= \{-1, -1, 1\} \\ \tau_4 &= \{1, -1, -1\} \end{aligned} \tag{4.9}$$

and the magnitude of canting again follows the proportionality

$$\gamma \propto \frac{D_y + D_z}{J} = 2D. \tag{4.10}$$

These equations predict no change or a slight decrease in canting for sublattice 1, which is seen in these results. For layer 2, it is again apparent that the linear relationship only holds for a small D, and the relationship decays beyond this.

In such a phase, a large out-of-plane angle will correspond with both a reduction in magnetization and an increase in error for measurements assuming no canting. These concerns are addressed in Fig. 4.8. This figure represents the in-layer magnetization

calculated as

$$M_L = \frac{|\sum_{i \in L} \mathbf{S}_i|}{N} \tag{4.11}$$

for a given layer, L. This value is expected to be near unity for all layers in the helical structure. If the magnetization is near unity it is reasonable to represent a layer by the average magnetic moment of the layer. It is clear that magnetization decreases with D, and therefore $\gamma_L^{(2)}$. This result must be considered for any conclusions made with this data.

The magnetization exhibits different behaviour in defined regions. For $D < 0.3$ magnetization is approximately equivalent between the layers and both layers are nearly ferromagnetic. In the range $0.3 < D < 1.0$ both layers are nearly ferromagnetic, and as D increases the magnetization for each layer separates and decreases slowly. At $D = 1.0$ The magnetization discontinuously drops (which does not coincide with changes in any other features), then increases in $1.0 < D < 1.3$ before decreasing in $1.3 < D < 1.6$. Another discontinuous drop is observed at $D = 1.5$ with similar behaviour as other regions occurring in $1.5 < D < 2.0$. In all regions, the magnetization of layer 2 is lesser than that of layer 1. These values coincide closely with previously defined phases.

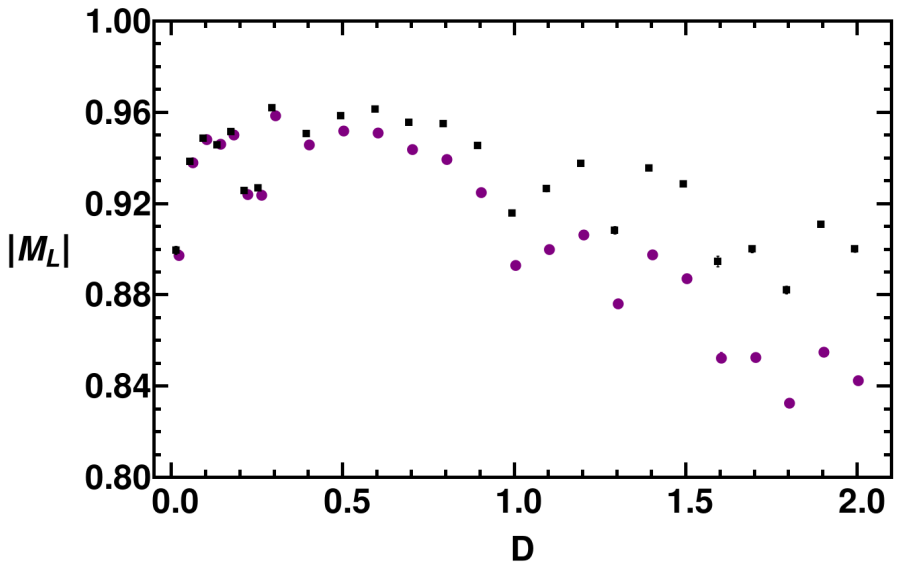


Figure 4.8: The magnetization of the distinct layers as a function of D. The magnetization decreases as D, with $M_L^{(2)}$ expressing a stronger decline.

Finally, it must be noted that the canting of layer 2 magnetic moments need not occur such that they are ferromagnetic, as seen in the magnetization measurements. It is predicted by the results of both Section 2.4 and Section 1.4.2 that the sum of canting angles of each sublattice – when considering the orientation of each canted normal vector – should sum to zero over a layer, *i.e.*, the average canted normal vector should align with [111]:

$$\mathbf{C}_2 + \mathbf{C}_3 + \mathbf{C}_4 \;||\; [111]. \tag{4.12}$$

This is apparent when considering relative canting instead of absolute, which approaches zero. Further – since these 3 vectors are expected to cant in equidistant directions as shown in Eq. 4.9 – this predicts that the angle between the average moment on the three sublattices 2–4, defined here as θ_{ij} (but differing from the θ defined in Section 3.2) and given by the usual expression

$$\theta_{ij} = \frac{1}{L}\sum_{l=1}^{L} \arccos(\bar{\mathbf{s}_i} \cdot \bar{\mathbf{s}_j}) \tag{4.13}$$

where $L = 3 * N - 2$ is the total number of layers, and i and j represent the sublattices being considered, should grow with γ (or, equivalently, D) and should be approximately the same magnitude for each pair of sublattices. This expectation is confirmed with the measurements shown in Fig. 4.9. It is clear that the separation between sublattices increases with D. However, this parameter increases prior to the stratification of γ between layers at $D \approx 0.3$.

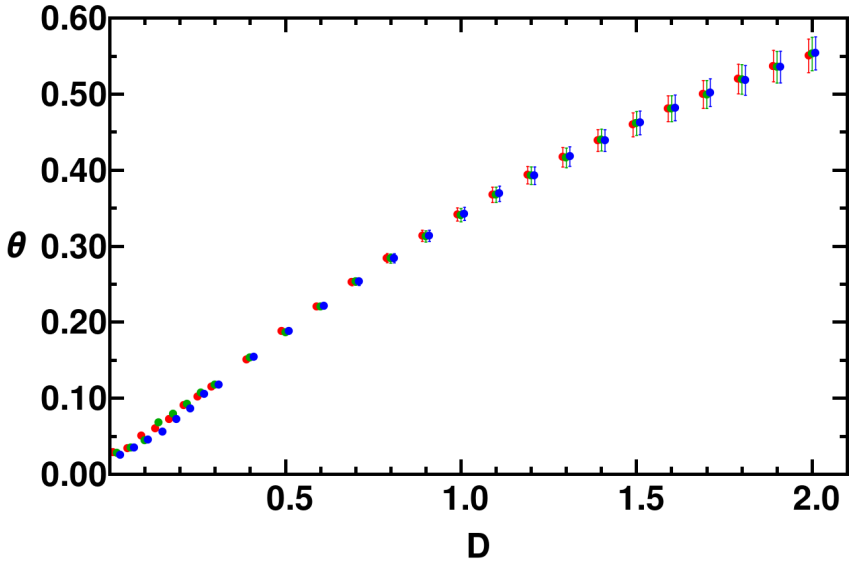


Figure 4.9: The relative angle between sublattices in layer 2. The colour represents the pair of sublattices being compared (red, θ_{23}; green, θ_{24}; blue, θ_{34}).

4.1.5 k || [111] order parameters

Assuming that for all D there exists a pure $\mathbf{k} \parallel [111]$ state one may calculate the magnetic OPs for a configuration using the definition provided in Eq. 2.22. This assumption is valid for the non-extreme cases explored here and the normalized magnitudes are displayed in Fig. 4.10a

As previously determined, a pure helical state with constant magnitude magnetic moments must have all $F_{A,i} = 0$ for all i. The vanishing of these values appears for $D > 0.2$ and coincides with the separation of γ_L for the two layers. Prior to $D = 0.2$, magnetic moments may cant partially towards either [111] direction akin to a conical phase. This value represents the appearance of a planar state.

Likewise, the F_{E_+} parameters decay and vanish at $D \approx 0.50$. This aligns with the separation between linear and sub-linear growth in $\mathbf{k}$ and ϕ. The OP magnitude resembles phase transitions at both of these points, as expected. For $D < 0.5$ the wavelength of the helix is very long, approaching ferromagnetism. The wavelength of the helical structure is larger than the simulation space, which correlates with a superposition of each chirality and therefore both F_{E_-} and F_{E_+}. This oddity disappears when the wavelength of the helix is contained within the simulation space.

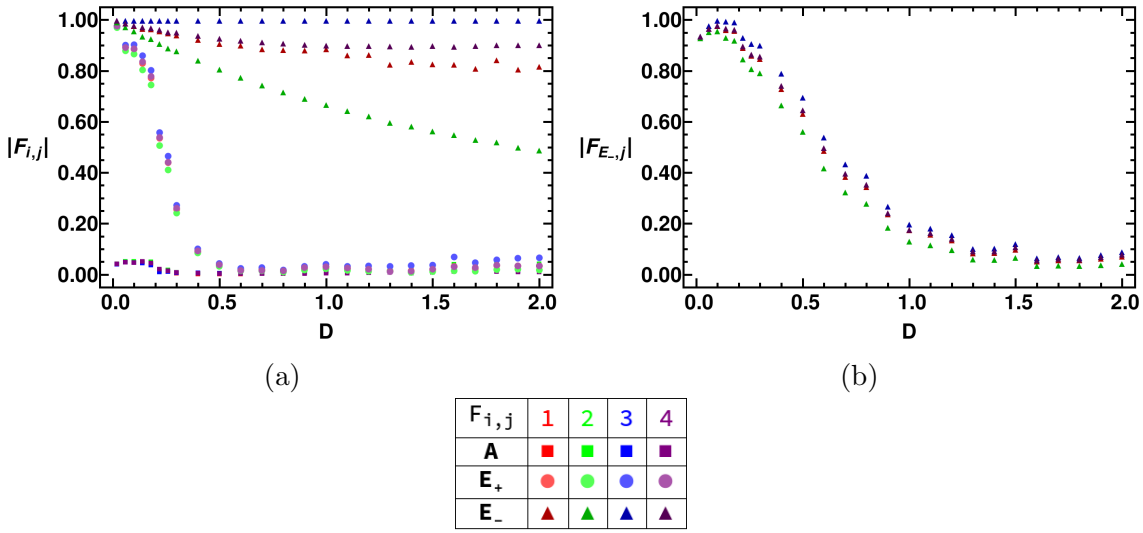


Figure 4.10: (a) Magnetic order parameter magnitude normalized within a single D value such that the maximum is always 1.0. (b) Magnetic order parameters $F_{E_-,j}$ normalized using the maximum value of all simulations. The legend applied to both figures. Marker shape represents IR and colour represents the number.

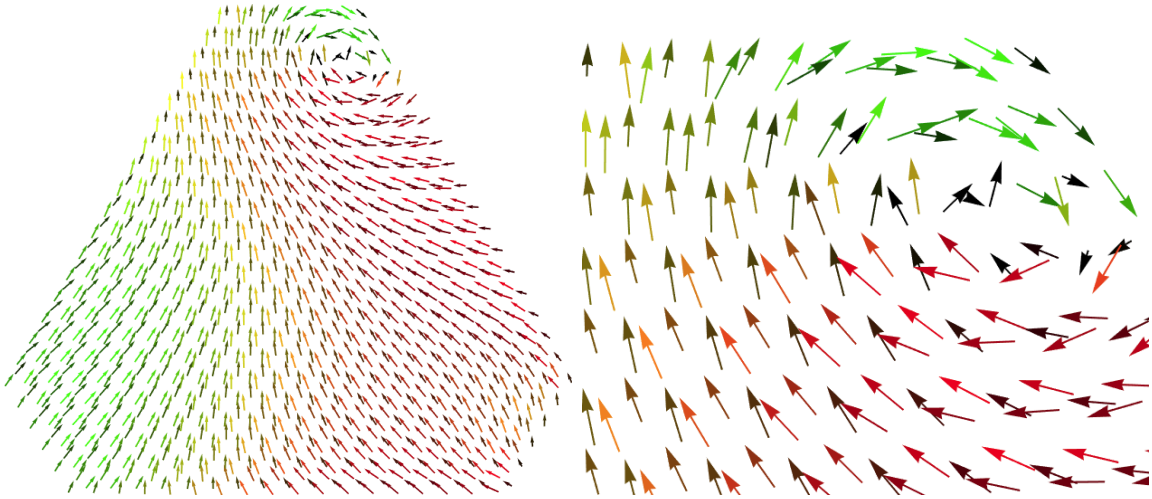
The relative magnitudes of different F_{E_-} terms do not remain constant as D increases. This separation is consistent with the aforementioned appearance of canted normal vectors with the variation displayed here corresponding with partial canting of sublattices 2–4 towards different diagonal cubic axes which increases with D.

It is worth noting that the relative stability of F_{E_-} magnitudes is a consequence of the normalization. If one instead reports only these terms, normalized using the maximum value achieved throughout all simulations, it also decays with D. This decay is much slower than other parameters. This is demonstrated in Fig. 4.10b.

In the range $D > 1.50$ magnetic order parameters F_{E_+} and F_A begin to reemerge. This corresponds with the appearance of other structures in the lattice at these values.

4.1.6 Isolated skyrmions

For values $D > 1.50$ isolated skyrmions begin to appear on the boundaries of the lattice. These are always Bloch-type skyrmions, as was observed in MnSi crystals [3]. These vortex-like structures appear as skyrmion tubes near the edges of the lattice



(a) (b)

Figure 4.11: A single two-dimensional layer of the MnSi lattice displaying the cross-section of an isolated Bloch-type skyrmion. (a) The full layer with skyrmion in upper-right. (b) A zoomed-in image in the region of the skyrmion. The hue of the magnetic moments represents direction while the saturation represents the magnitude of the in-plane component of the moment. Both are viewed from the [111] direction.

and are oriented along directions close to the wavevector. Simulations in which these structures become apparent show a very small energy difference ($\frac{\Delta E}{E} \approx 0.02\%$) within the tolerance of the simulation. An example from a single $D = 2.00$ simulation is presented in Fig. 4.11. This result is unexpected as bulk skyrmions are reported to only appear in small ranges of temperature with non-zero applied fields.

In finite size systems, skyrmions have been shown to be stabilized through finite size effects in three-dimensional thin films [36, 37], as well as chiral bobber phases – a similar vortex-like state in which the skyrmion does not penetrate through the material and terminates in a Bloch-point [38]. The thickness of these thin films is reported using a "confinement ratio," *i.e.*, the ratio between layer thickness and helical wavelength. At the values of D observed here, this ratio is ~ 3, much larger than those reported in these studies. Further, these results suggest that skyrmions should not be stable at zero-field at any thickness. Zero-field skyrmions have been realized in thin film FeGe [39], however, it has been suggested that a skyrmion phase can not be detected through Hall effect measurements alone [40].

Alternatively, strong anisotropy is known to help stabilize skyrmion phases. In two-dimensions, a relationship between skyrmion lattices and anisotropy strength has

been reported [16]. However, there is no mention of isolated skyrmions in these studies. In three-dimensions, it has been shown that isolated skyrmions can be stabilized by specific anisotropy interactions. Specifically, in zero-field isolated skyrmions are stabilized for particularly strong anisotropy [41]. The relatively strong anisotropy present in the simulations reported here, along with the anisotropy which will arise due to edge effects in the simulation space, suggest that the appearance of these vortices could be explained by these interactions.

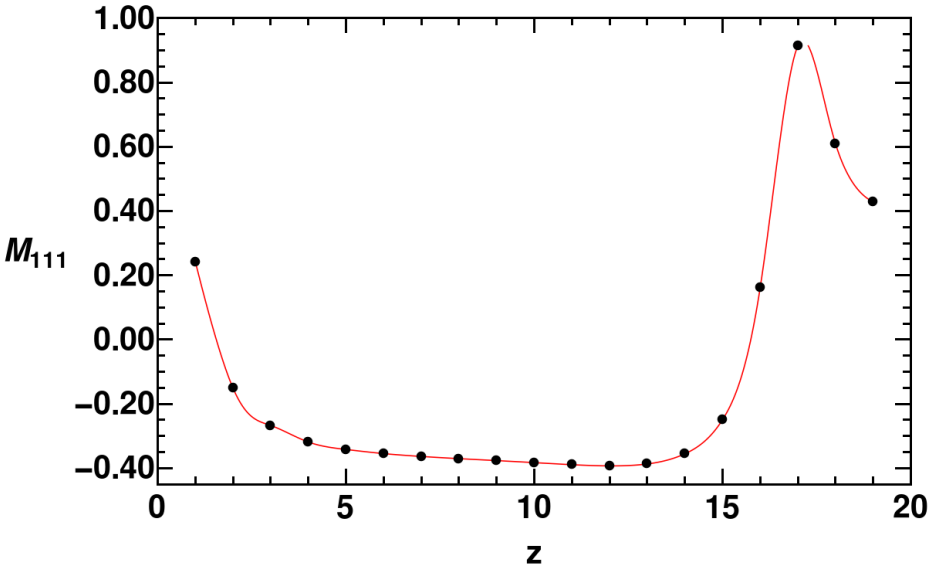


Figure 4.12: Magnetization along the [111] direction for magnetic moments in sublattice 2 along the line $\{1, 22 - z, 4 + z\}$ in Fig. 4.11. The line is produced through cubic interpolation. A sharp peak characteristic of a skyrmion appears. Edge effects are apparent at values $z \approx 1$ and $z \approx 20$.

A skyrmion is always associated with a peak in the magnetization along some direction. The magnetization along one line in the layer is measured in Fig. 4.12. This measure shows a clear peak along the [111] direction, as expected.

The appearance of skyrmions produces large disturbances in the helical structure of the lattice. They also introduce different wavevectors to a lattice and, therefore, the assumption of a pure helical $\mathbf{k} \parallel [111]$ state is inappropriate in regions where they appear.

4.2 Varying all model terms about the $D = 0.50$ helical structure

The ultimate goal of this computational study is to better understand how varying model terms affect the measured parameters. In doing so, one will be allowed greater control over the detail of the magnetic structures produced. Therefore, I will begin by varying the strength of each term around a known $\mathbf{k} \,||\, [111]$ and measure the variation of these features.

The value $D = 0.5$ was selected as the value around which the parameters will be varied. This state is chosen due to its agreeable behaviour: it displays stratification in γ but not a large amount, it sits in the area in between linear and sublinear growth of $|\mathbf{k}|$ and approximates the magnitude observed in experimental studies, displays a large anomalous phase, maintains reasonably high in-layer ferromagnetic magnetization, and occupies a point in which the F_{E_-} order parameters dominate and are of relatively equivalent magnitude.

In this section, variations will extend approximately 10% to either side of any value (or to an absolute value of 0.10 in the case of the hitherto unused symmetric terms).

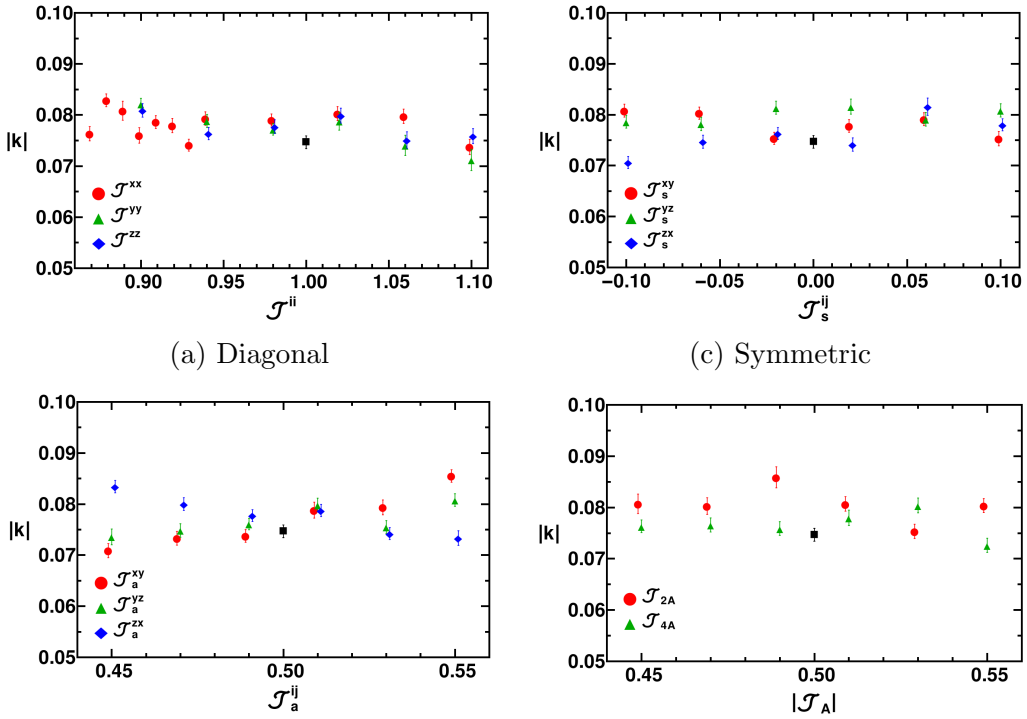
4.2.1 Wavevector magnitude

Considering the relationship reported in Eq. 4.1, one should expect the magnitude of the wavevector to vary with the strength of both the diagonal and antisymmetric terms of the model. However, no relationship to the symmetric and anisotropic terms was described.

The results presented in Fig. 4.13 confirm the expected results, with antisymmetric terms displaying the appropriate linear proportionality in these measurements when the relationship between the $\mathbf{D}_{ij}$ vector and the antisymmetric terms is given by $\{D_x, D_y, D_z\} = \{\mathscr{J}_a^{xy}, \mathscr{J}_a^{yz}, \mathscr{J}_a^{zx}\}$. However, it appears as though the individual diagonal terms do not induce substantial variation in $|\mathbf{k}|$ in this range, along with both the symmetric and anisotropic terms. The lack of variation with diagonal terms is explained by the denominator of the relationship being the reduced constant, J. This value is similar to an average of the three diagonal terms and, therefore, varying

only one of these terms will have only small effects on this value.

The other terms, both symmetric and anisotropic, do not appear to have a significant relationship with this value. In fact, it will become apparent throughout this section that the fourth-order anisotropy will have no significant relationship with any of the measured parameters. Therefore, it is reasonable to suggest that, in the range around $D = 0.50$, this parameter is predominantly dependent on the antisymmetric terms, *i.e.*, $\mathbf{D}_{ij}$.



(a) Diagonal

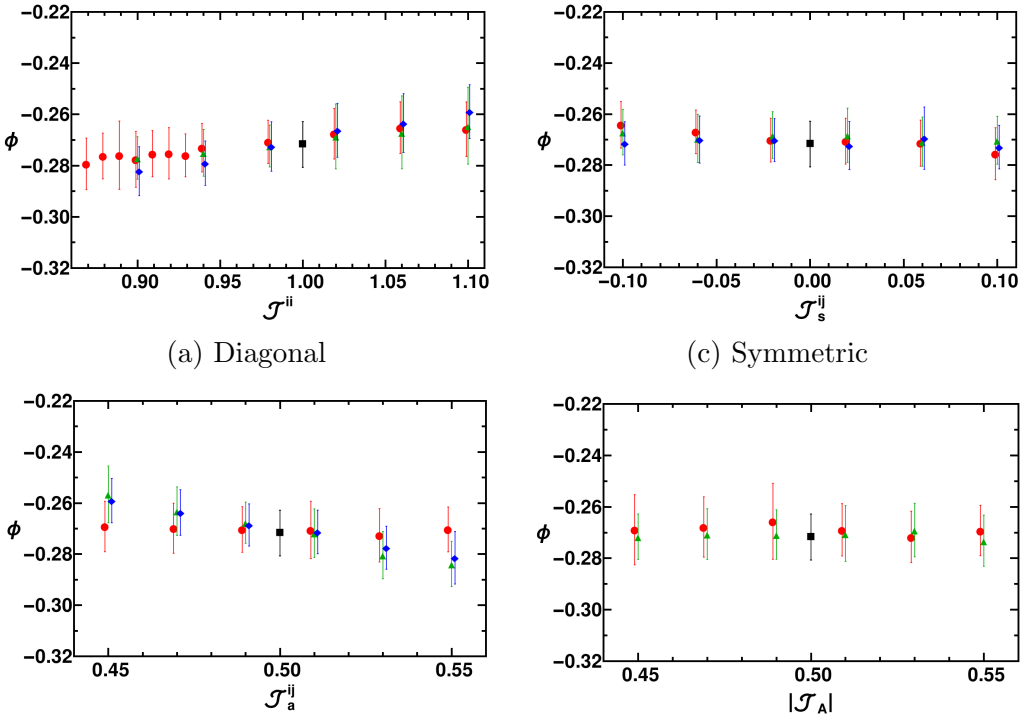
(c) Symmetric

(b) Antisymmetric

(d) Anisotropy

Figure 4.13: Wavevector magnitude as a function of individual coupling constant magnitude for each coupling constant. Coupling constants are grouped in the usual way, with the symbol corresponding to each indicated. The black square represents the $D = 0.50$ result from section 4.1. An absolute value is considered in (d) as the two anisotropic interaction strengths have opposing signs. All other figures in this section match the definitions here.

4.2.2 Anomalous phase



(a) Diagonal

(c) Symmetric

(b) Antisymmetric

(d) Anisotropy

Figure 4.14: Anomalous phase as a function of individual coupling constant magnitude for each coupling constant. Coupling constants are grouped in the usual way, with the symbol corresponding to each indicated. The black square represents the $D = 0.50$ result from section 4.1. Symbols match those in Fig. 4.13.

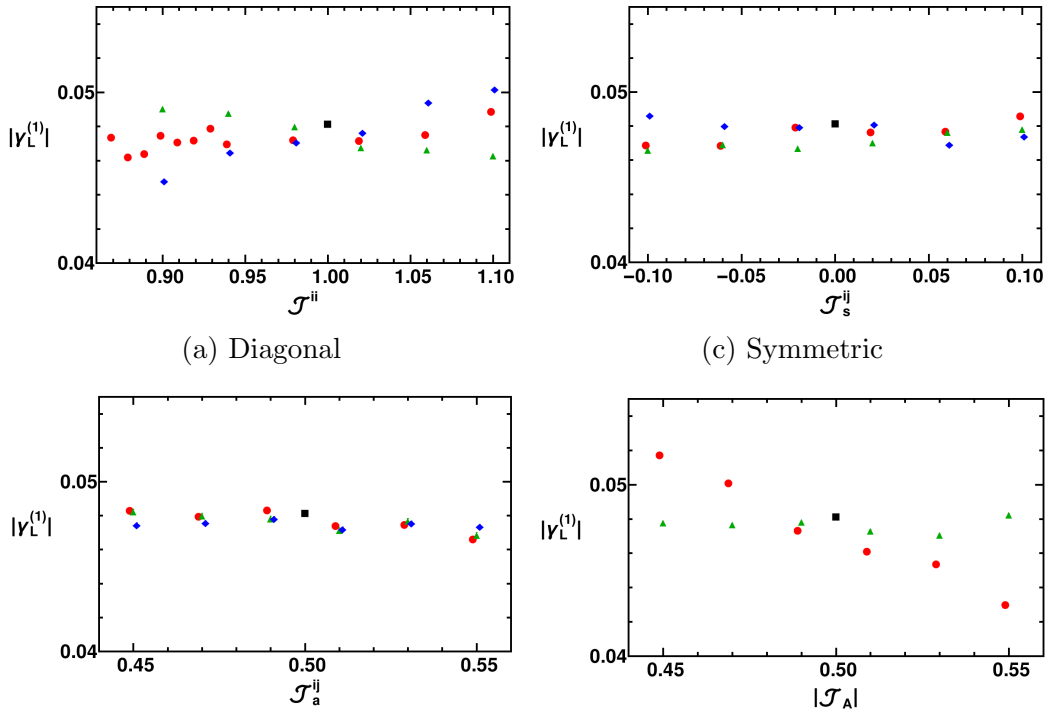
The variation observed in the parameter ϕ is minute over the range of values considered. However, the predicted result (Eq. 4.7) suggests that $\mathcal{J}_a^{yz}$ and $\mathcal{J}_a^{zx}$ should be directly proportional (with a negative constant of proportionality) while $\mathcal{J}_a^{xy}$ should have no effect. This can be seen in Fig. 4.14. The dependence of this measurement on the value of each diagonal term is also observed, with a direct proportionality apparent. This relationship is stronger with the value of $\mathcal{J}^{zz}$, which is not a result of the previous analysis.

The symmetric and anisotropic terms have little effect, with only a potential small dependence on $\mathcal{J}_s^{xy}$ visible. This correlation between the symmetric and antisymmetric values, with $\mathcal{J}_s^{xy}$ producing variation only if $\mathcal{J}_a^{xy}$ does not, is seen throughout

this section. A relationship between similar symmetric and antisymmetric terms occurs naturally when considering their construction, but the full effect is not explored here.

4.2.3 Out-of-plane angle

While the antisymmetric terms produce canting as one would expect from all previous results, the reliance of the system structure on terms that are not directly related to the standard DMI and Heisenberg interactions is most apparent in the canting and relative angle of the sublattices. The canting of layer 1 and layer 2 is measured in Fig. 4.15 and Fig. 4.16, respectively.



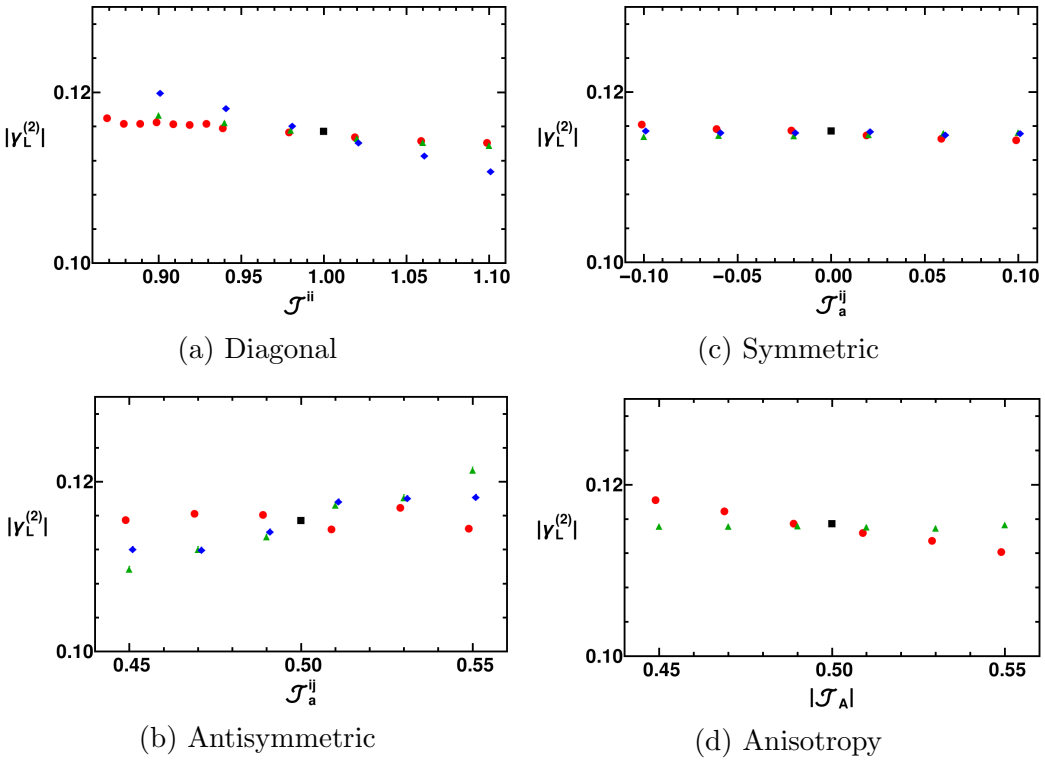
(a) Diagonal

(c) Symmetric

(b) Antisymmetric

(d) Anisotropy

Figure 4.15: Layer 1 out-of-plane angle as a function of individual coupling constant magnitude for each coupling constant. Coupling constants are grouped in the usual way, with the symbol corresponding to each indicated. The black square represents the $D = 0.50$ result from section 4.1. Symbols match those in Fig. 4.13.



(a) Diagonal

(c) Symmetric

(b) Antisymmetric

(d) Anisotropy

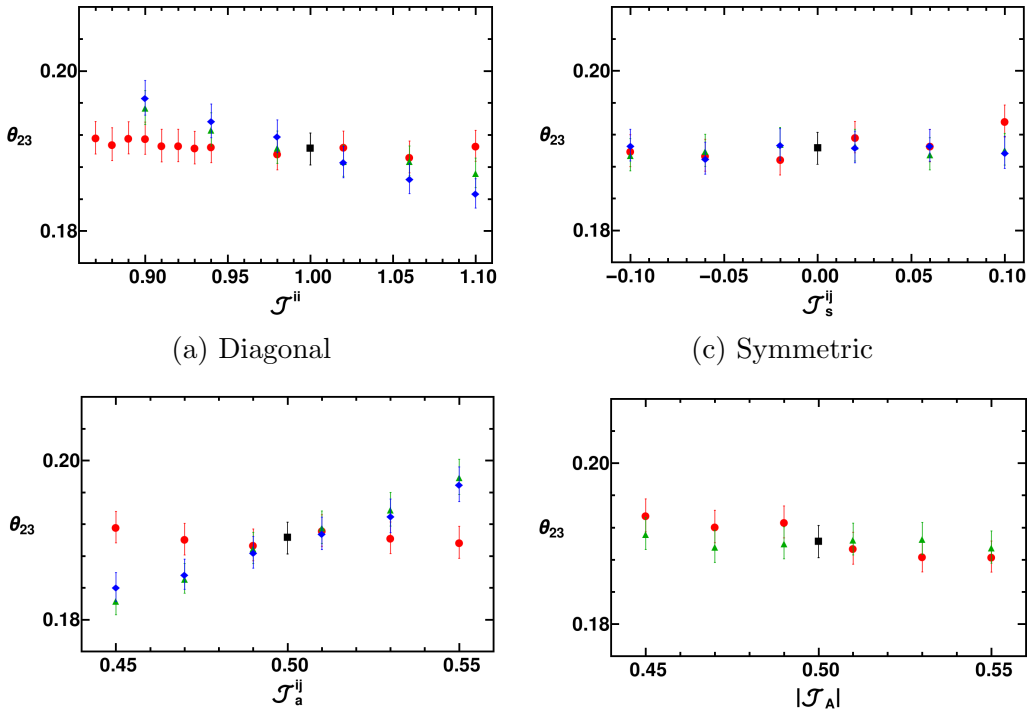
Figure 4.16: Layer 2 out-of-plane angle as a function of individual coupling constant magnitude for each coupling constant. Coupling constants are grouped in the usual way, with the symbol corresponding to each indicated. The black square represents the $D = 0.50$ result from section 4.1. Symbols match those in Fig. 4.13.

In both cases, it is clear that there is a stronger dependence on the diagonal term $\mathcal{J}^{zz}$ in comparison to $\mathcal{J}^{yy}$ and $\mathcal{J}^{xx}$, similar to the anomalous phase. The relationship between canting and the diagonal terms is complex: each layer can have constants of proportionality with differing signs for each of the parameters. Further, the canting of layer 1 exhibits different sign constants of proportionality between the three terms. Therefore, if one desires to minimize canting in both layers, varying these parameters alone may not have the desired effect.

Regarding the antisymmetric terms, the expected relative inertness of $\gamma_L^{(1)}$ is apparent, and the same can be observed in the symmetric terms. The symmetric terms also have little-to-no effect on $\gamma_L^{(2)}$, in contrast to the antisymmetric terms. The one exception being the previously mentioned relation between the terms $\mathcal{J}_s^{xy}$ and $\mathcal{J}_a^{xy}$. These results are more clear in the measurement of inter-sublattice angle for layer 2 in Fig. 4.17, where only θ_{23} is shown due to the equivalence of all terms.

The other parameters appear to have small effects on $\gamma_L^{(1)}$ and $\gamma_L^{(2)}$. The exception is the second-order anisotropy, which is known to prefer planar states in the [111] direction when the value is negative and is strongly correlated with a lower canting angle. However, anisotropies are generally considered to be relatively small.

Finally, the in-layer magnetization is again presented in Fig. 4.18. Here it is apparent that the relative strength of Heisenberg-like diagonal terms and the second-order anisotropy has the strongest effect on the ferromagnetic magnetization of the layers.



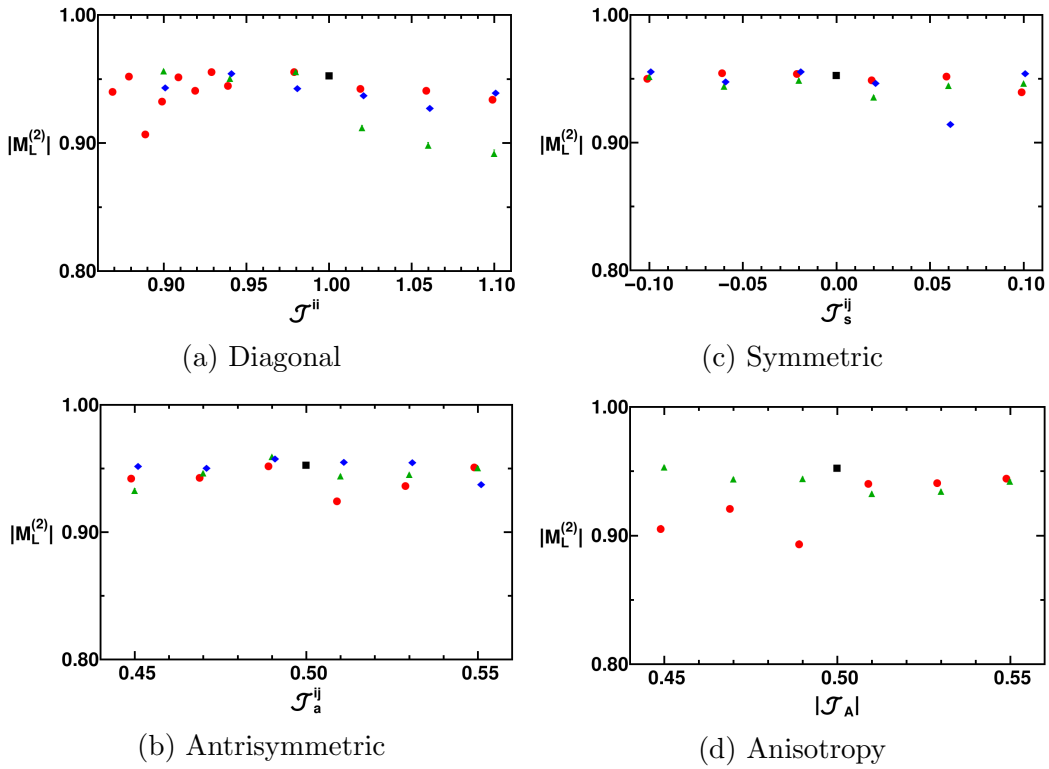
(a) Diagonal

(c) Symmetric

(b) Antrisymmetric

(d) Anisotropy

Figure 4.17: The angle between sublattice 2 and 3 of a layer as a function of individual coupling constant magnitude for each coupling constant. Coupling constants are grouped in the usual way, with the symbol corresponding to each indicated. The black square represents the $D = 0.50$ result from section 4.1. Symbols match those in Fig. 4.13.



(a) Diagonal

(c) Symmetric

(b) Antrisymmetric

(d) Anisotropy

Figure 4.18: The in-layer magnetization of layer 2 as a function of individual coupling constant magnitude for each coupling constant. The magnetization of layer 1 is consistent with this one. Coupling constants are grouped in the usual way, with the symbol corresponding to each indicated. The black square represents the $D = 0.50$ result from section 4.1. Symbols match those in Fig. 4.13.

4.3 Applied field simulations

The final test of the presented model will be with the use of an applied field. Applied fields are predicted to rotate the direction of $\mathbf{k} \parallel \mathbf{B}_{\text{app}}$, as well as canting of the system into a conical structure and, at some critical field, a ferromagnetic state. In the present work, I will consider only the second of these options, presenting results with fields applied along the $\mathbf{k}$.

It must be noted that, due to the finite size of the system, the effect of magnetic fields will be altered from those of an infinite system. Applied fields will have relatively larger effects on magnetic moments with lower coordination numbers, *i.e.*, those near the edges of the lattice. This will affect the structure of the entire lattice. With this

caveat, one may still glean some information from this data.

4.3.1 B || k

With the field applied parallel or anti-parallel to the wavevector of the structure there are two parameters of particular interest. First, one must consider the transition of the pure helical phase into a conical phase with magnetic moments partially oriented along the direction of the applied field. Specifically, the magnitude of the conical canting will be measured and compared to the strength of the applied field. The other parameter is the critical field between the conical and field-induced ferromagnetic states. That is, the applied field (denoted B_{C2} here) at which the helical phase is so strongly aligned with the field that it is no longer distinguishable from the ferromagnetic phase. The second of these parameters can be approximated from the *stiffness*, A, of the helix – a parameter connected to the strength of the DMI. This relationship is given by

$$B_{C2} \approx Ak^2 \tag{4.14}$$

where the usual constants are included within the field and

$$A = \frac{|\mathbf{S}|D}{|\mathbf{k}|} = \frac{D}{|\mathbf{k}|} \tag{4.15}$$

with $|\mathbf{S}| = 1$ enforced [18]. This definition of stiffness is only true for large $\mathbf{k}$ and may not be applicable here. Alternatively, the minimization analysis performed by Chizhikov & Dmitrienko predicts that this field and the conical canting can both be approximated from the equation

$$\sin\gamma = \frac{6B}{(D_z - 2D_x - D_y)^2} \tag{4.16}$$

where, in this case, γ refers to a conical canting and not a canted normal vector. The critical field will occur at $\gamma = \frac{\pi}{2}$. Rearranging Eq. 4.16 that is

$$B_{C2} = \frac{(D_z - 2D_x - D_y)^2}{6} \tag{4.17}$$

for the critical field and

$$\gamma \propto \arcsin(aB) \tag{4.18}$$

or

$$\gamma \propto B \tag{4.19}$$

where the linear approximation applies to $\arcsin(x)$ for small x. One can show that the two predictions for the critical field differ greatly. This is not unexpected due to the small wavevector $\mathbf{k}$. Further, the inclusion of anisotropies will affect this value and the behaviour of these terms considerably. As before, the initial results in this section are produced using simulations with the second and fourth-order anisotropy strength set to $\mathscr{J}_{2A} = -0.50$ and $\mathscr{J}_{4A} = 0.50$, respectively.

However, while significant effects from the fourth-order anisotropy were not observed in the results of Section 4.2, they become apparent here. Analysis of this term reveals that – if the coupling of this term is positive – it prefers magnetic moments to align along a finite set of directions including the cubic lattice vectors. When canting is small, this has a very small effect. However, when canting is large enough and magnetic moments approach a cubic lattice vector, as seen with a sufficiently large applied field, this anisotropy produces "hitching." That is, the helical rotation of the magnetic moments is slowed in this area and the measured wavevector is decreased. The average wavevector magnitude reported in Fig. 4.19 is therefore the average of two wavevectors: One that is approximately equivalent to the $B = 0$ case, and one that is reduced due to this hitching. This is akin to domains of helical and ferromagnetic structures appearing. Therefore, I must suggest that the fourth-order anisotropy should be near vanishing, and is much too large. Fortunately, as the results of Section 4.2 corroborate, the effects of this anisotropy are very small in the pure helical phase, and previous results still provide valuable insight. The effect of varying this parameter for small applied fields is also presented in Fig. 4.19. It can be seen that the suppression of $\mathbf{k}$ is greatly reduced with a smaller anisotropy strength. However, the larger anisotropy is used in other results reported here as it is observed to have little effect on the two physical parameters of interest in this section.

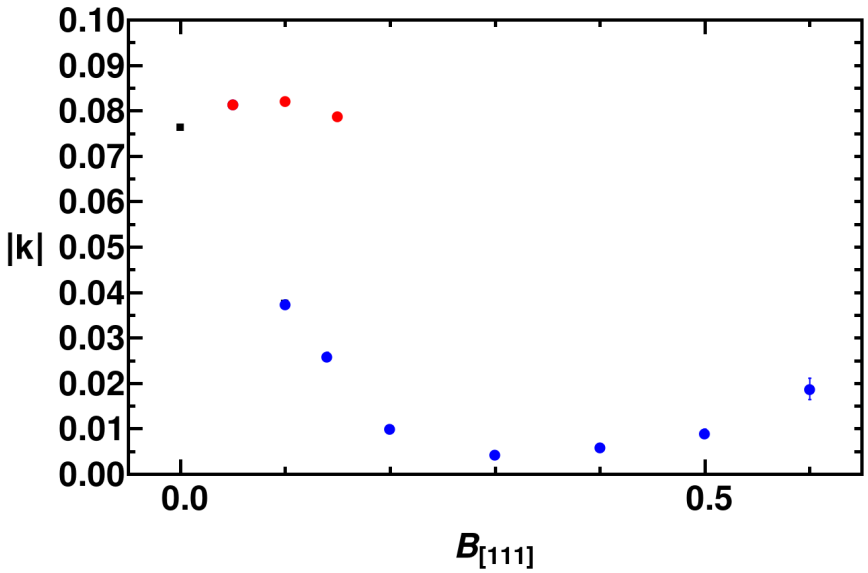


Figure 4.19: The average wavevector magnitude as a function of B. Red markers represent simulations with $\mathscr{J}_{4A} = 0.10$ and blue markers represent simulations with $\mathscr{J}_{4A} = 0.50$. The suppression of the magnitude as B increases is due to the averaging of two separate wavevectors: One that is approximately constant when the magnetic moments are relatively distant from a cubic lattice vector, and a smaller value determined by the proximity of a magnetic moments orientation to a cubic lattice vector.

Despite this concern, the measurement of γ follows the expected behaviour. These measurements are displayed in Fig. 4.19. The two parameters of interest, the canting angle and critical field, may be interpreted directly from these measurements. The relationship between canting and field strength is approximately linear in the range considered and is fit using the function

$$\gamma = c + \arcsin(aB). \tag{4.20}$$

Extrapolating this fit in all sublattices demonstrates an critical field of $B_{C2} = \frac{1}{a} \approx 1.0$. This behaviour is observed with all values of $\mathscr{J}_{A4}$ considered. Due to the small-angle approximation, this closely approximates the relation $\gamma = B$ for these model parameters, as well.

Separately, one can confirm that the minimum value of $|\mathbf{k}|$ above occurs around γ values which are closely aligned with the cubic lattice axes, $\gamma = \frac{\pi}{4}$, as expected.

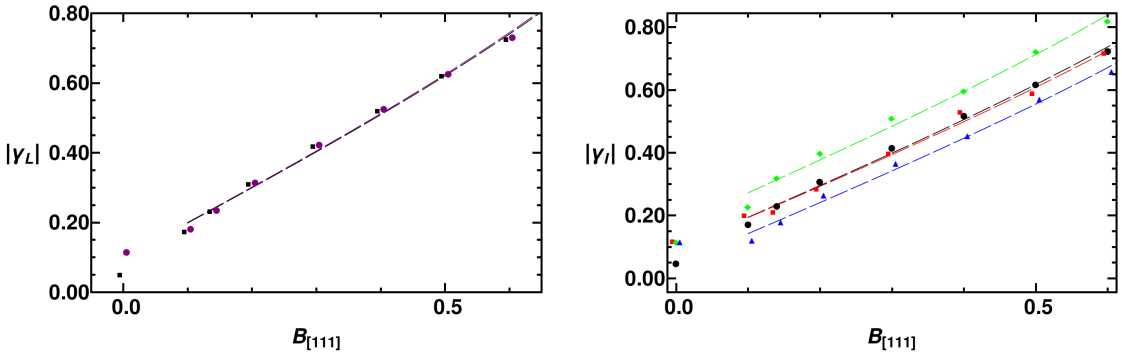


Figure 4.20: The out-of-plane canting of (a) individual sublattices and (b) layers as a function of B. The colouring is the same as that presented in all previous figures.All fits are of the form $c + \arcsin(aB)$ and all predict an critical of $B_{C2} = a \approx 1.0$.

The relation of this critical field and proportionality of γ to other constants is not clear from these tests. The value of both constants, 1.0, is easily derived from all terms considered, and therefore further studies will be required to determine the relation.

Finally, the magnetic order parameters are displayed in Fig. 4.21. In this, the appearance and dominance of $F_{A,j}$ order parameters, as predicted for a conical phase, is visible. Additionally, as the system approaches a ferromagnetic state, the values of the other magnetic OPs $F_{E_+,j}$ and $F_{E_-,j}$ become roughly equivalent. All of these features are expected for an infinite lattice, and their appearance in a finite lattice provides further confidence in the generality of these results of this study.

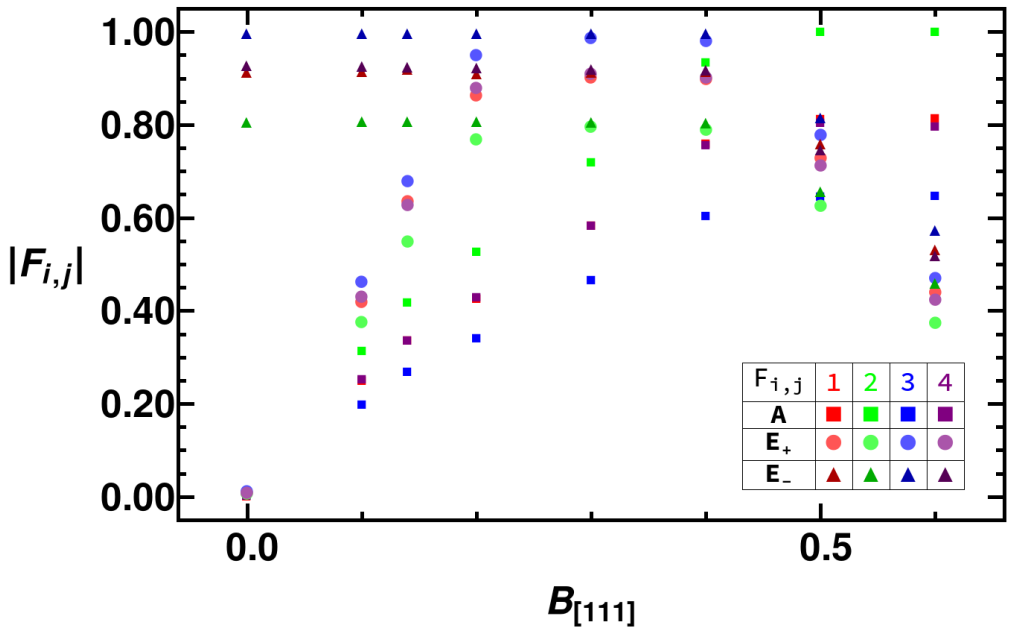


Figure 4.21: Magnetic order parameter magnitude normalized within a single B values such that the maximum is always 1.0. Markers match Fig. 4.10.

Chapter 5

Conclusion

5.1 Discussion

In this thesis, I have provided the details of the construction of a fully general, classical model of MnSi-like crystals up to nearest neighbours. Only the relative strength of 9 bilinear invariants remain variable. The process of construction used can easily be extended to bilinear terms of any order of neighbour, as necessary. This model may be combined with any applicable anisotropy or external interactions, as done here, and used as a predictive model of the magnetic state of these crystals. Analysis of this model provides insight on the relationship between its terms and other common, classical interactions – useful for predictive analysis of the system based on established analytic results. Further, an analysis providing the process of defining the magnetic order parameters of these crystals, based on the symmetry of the lattice and the wavevector $\mathbf{k}$ of a magnetic state was divulged. These OPs may be manipulated with appropriate assumptions to determine the properties applicable to a given state and set of interactions.

The provided model, in conjunction with second and fourth-order anisotropic interactions, as well as the Zeeman interaction, was studied through computational simulations using the Effective Field Method on a finite lattice without periodic boundary conditions. This analysis allowed the model to be examined in comparison to the literature based on these systems and provided some predictive analysis of the action of the coupling associated with each term. This analysis is separated into 3 sections: (1)

A set using the standard classical Heisenberg interactions and DMI in which the DMI strength D is varied relative to the Heisenberg strength J. (2) A set of simulations in which a single, typical helical state is selected and each of the 9 terms is varied about this state. (3) A set of simulations in which a field is applied along $\mathbf{k}$ of the same typical state and the canting and critical field B_{C2} of the helical state are measured.

In the first analysis, the predicted helical state with only a set of same-chirality magnetic order parameters was produced. A linear relationship between the wavevector magnitude, $|\mathbf{k}|$, and the relative strength of the DMI, is apparent for relatively small values of D, as predicted. However, a transition to a sublinear relationship for relatively large values coincides with a large canting of the normal vector of the helix and the wavelength reducing to fit within one simulation cell of $23 \times 23 \times 23$. The canting vector and magnitude of the helices differed between the 4 cubic sublattices of the crystal, as predicted through both the OP analysis and previous minimization analysis using less general models. This splitting of the normal vectors was observed to occur only for $D > 0.3J$ with magnitude increasing with D afterwards. Further, the canting was shown to occur such that the sum of canting in any given two-dimensional layer along the [111] axis vanishes. The anomalous phase, ϕ, was observed and shown to vary with D similarly to $|\mathbf{k}|$. However, the magnitude in these simulations is much larger than those observed in experiment. This suggests that other interactions are important to the magnitude of this parameter. Finally, this set of simulations produced isolated skyrmions near the edge of the simulation cell. These occur only for very large values of $D > 1.50J$ and introduce other magnetic order parameters. Therefore, the reported analysis of simulations in this range is affected by these structures.

Analysis of results produced from varying individual parameters confirmed several properties of the predictive analysis, with the DMI-related antisymmetric terms and Heisenberg-related diagonal terms providing the most prominent effect on the magnetic structure. The other symmetric terms were also shown to have minor effects on the structure, with a relationship between related antisymmetric and symmetric terms qualitatively shown. From this, finer control of the measured lattice parameters is given.

Finally, analysis with a field applied along the helical wavevector revealed the expected conical phase of the lattice. The canting angle is shown to follow a roughly

linear relationship with applied field strength. The helical phase transitions to a conical phase with small applied fields. However, the action of the fourth-order anisotropy prevents this from approaching a pure conical phase. Nonetheless, the critical field is extrapolated from the results as saturation of magnetic moments (aligning along $\mathbf{B}$) and is approximated to be $B = 1.0$. Further analysis will be required to determine the relationship between this value and others of the model.

The above results are shaped by the finite size restriction of the lattice and they may apply more directly to studies of thin films. The chosen lattice size, however, is large enough that these results present a meaningful origin for the study of the general model.

5.2 Future Work

The model provided here may be used in the future analysis of any MnSi-like magnet. A deeper analysis of the included terms, or the introduction of other interactions or thermal effects, would provide for finer control and a better understanding of the magnetic structures. The subtle relationships between model parameters could be further explored through these analyses or further simulations. Additionally, the order parameters derived here could be analyzed or a different choice for their definitions could be made. This change could lead to further predictive power from these values.

Bibliography

[1] L Lundgren, O Beckman, V Attia, S P Bhattacheriee, and M Richardson. Helical spin arrangement in cubic FeGe. *Physica Scripta*, 1(1):69–72, Jan 1970.

[2] Y. Ishikawa, K. Tajima, D. Bloch, and M. Roth. Helical spin structure in manganese silicide MnSi. *Solid State Communications*, 19:525–528, July 1976.

[3] S. Mühlbauer, B. Binz, F. Jonietz, C. Pfleiderer, A. Rosch, A. Neubauer, R. Georgii, and P. Böni. Skyrmion lattice in a chiral magnet. *Science*, 323(5916):915–919, 2009.

[4] X. Z Yu, N Kanazawa, Y Onose, K Kimoto, W. Z Zhang, S Ishiwata, Y Matsui, and Y Tokura. Near room-temperature formation of a skyrmion crystal in thin-films of the helimagnet fege. *Nature materials*, 10(2):106–109, 2010.

[5] W Münzer, A Neubauer, T Adams, S Mühlbauer, C Franz, F Jonietz, R Georgii, P Bni, B Pedersen, M Schmidt, A Rosch, and C Pfleiderer. Skyrmion lattice in the doped semiconductor fe1-xco xsi. *Physical Review B - Condensed Matter and Materials Physics*, 81(4), 2010.

[6] Toshiaki Tanigaki, Kiyou Shibata, Naoya Kanazawa, Xiuzhen Yu, Yoshinori Onose, Hyun Soon Park, Daisuke Shindo, and Yoshinori Tokura. Real-space observation of short-period cubic lattice of skyrmions in mnge. *Nano Letters*, 15(8):5438–5442, 2015.

[7] Alexei N Bogdanov and DA Yablonskii. Thermodynamically stable vortices in magnetically ordered crystals. the mixed state of magnets. *Zh. Eksp. Teor. Fiz*, 95(1):178, 1989.

[8] A. Bauer and C. Pfleiderer. Magnetic phase diagram of mnsi inferred from magnetization and ac susceptibility. *Phys. Rev. B*, 85:214418, Jun 2012.

[9] Jens-Erik Jørgensen and Svend Erik Rasmussen. Refinement of the structure of mnsi by powder diffraction. *Powder Diffraction*, 6(4):194195, 1991.

[10] N. W. Ashcroft and N. D. Mermin. *Solid State Physics*. Saunders College, Philadelphia, 1976.

[11] A. B. Harris, C. Kallin, and A. J. Berlinsky. Possible néel orderings of the kagomé antiferromagnet. *Phys. Rev. B*, 45:2899–2919, Feb 1992.

[12] I. E. Dzyaloshinskiĭ. Theory of helicoidal structures in antiferromagnets, i. nonmetals. *Soviet Physics JETP*, 19(4):963–971, Oct 1964.

[13] P Bak and M H Jensen. Theory of helical magnetic structures and phase transitions in MnSi and FeGe. *Journal of Physics C: Solid State Physics*, 13(31):L881–L885, nov 1980.

[14] S. Seki, X. Z. Yu, S. Ishiwata, and Y. Tokura. Observation of skyrmions in a multiferroic material. *Science*, 336(6078):198–201, 2012.

[15] Tôru Moriya. Anisotropic superexchange interaction and weak ferromagnetism. *Phys. Rev.*, 120:91–98, Oct 1960.

[16] Su Do Yi, Shigeki Onoda, Naoto Nagaosa, and Jung Hoon Han. Skyrmions and anomalous hall effect in a dzyaloshinskii-moriya spiral magnet. *Phys. Rev. B*, 80:054416, Aug 2009.

[17] S. V Maleyev. Cubic magnets with dzyaloshinskii-moriya interaction at low temperature. *Physical Review. B, Condensed Matter and Materials Physics*, 73(17), 2006.

[18] S V Grigoriev, S V Maleyev, A I Okorokov, Yu O Chetverikov, and H Eckerlebe. The magnetic structure of mnsi under an applied field. *Journal of Physics: Condensed Matter*, 19(14):145286, 2007.

[19] Roderich Moessner and Arthur P. Ramirez. Geometrical frustration. *Physics Today*, 59(2):24–29, 2006.

[20] Rick Keesman, Mark Raaijmakers, A. E. Baerends, G. T. Barkema, and R. A. Duine. Skyrmions in square-lattice antiferromagnets. *Phys. Rev. B*, 94:054402, Aug 2016.

[21] H. Y. Yuan, O. Gomonay, and Mathias Kläui. Skyrmions and multisublattice helical states in a frustrated chiral magnet. *Phys. Rev. B*, 96:134415, Oct 2017.

[22] Akio Yoshimori. A new type of antiferromagnetic structure in the rutile type crystal. *Journal of the Physical Society of Japan*, 14(6):807–821, 1959.

[23] T.A. Kaplan. Classical spin-configuration stability in the presence of competing exchange forces. *Physical Review*, 116(4):888–889, 1959.

[24] T Garel. On the critical behaviour of two-dimensional xy helimagnets. *Journal of Physics C: Solid State Physics*, 13(31):L887–L893, 1980.

[25] S. L. Zhang, I. Stasinopoulos, T. Lancaster, F. Xiao, A. Bauer, F. Rucker, A. A. Baker, A. I. Figueroa, Z. Salman, F. L. Pratt, S. J. Blundell, T. Prokscha, A. Suter, J. Waizner, M. Garst, D. Grundler, G. Van Der Laan, C. Pfleiderer, and T. Hesjedal. Room-temperature helimagnetism in fege thin films. *Scientific Reports*, 7(1):1, 2017.

[26] X. Z. Yu, Y. Onose, N. Kanazawa, J. H. Park, J. H. Han, Y. Matsui, N. Nagaosa, and Y. Tokura. Real-space observation of a two-dimensional skyrmion crystal. *Nature*, 465(7300):901, 2010.

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

[28] P. Dalmas de Réotier, A. Maisuradze, A. Yaouanc, B. Roessli, A. Amato, D. Andreica, and G. Lapertot. Determination of the zero-field magnetic structure of the helimagnet mnsi at low temperature. *Phys. Rev. B*, 93:144419, Apr 2016.

[29] A. Yaouanc, P. Dalmas de Réotier, A. Maisuradze, and B. Roessli. Magnetic structure of the mnge helimagnet and representation analysis. *Phys. Rev. B*, 95:174422, May 2017.

[30] S. V. Grigoriev, D. Chernyshov, V. A. Dyadkin, V. Dmitriev, E. V. Moskvin, D. Lamago, Th. Wolf, D. Menzel, J. Schoenes, S. V. Maleyev, and H. Eckerlebe. Interplay between crystalline chirality and magnetic structure in mn 1 x fe x si. *Physical Review B*, 81(1), 2010.

[31] V. A. Chizhikov and V. E. Dmitrienko. Frustrated magnetic helices in MnSi-type crystals. *Physical Review B*, 85(014421):014421, 2012.

[32] M L Plumer and M B Walker. Wavevector and spin reorientation in mnsi. *Journal of Physics C: Solid State Physics*, 14(31):4689–4699, 1981.

[33] L. R. Walker and R. E. Walstedt. Computer model of metallic spin-glasses. *Phys. Rev. B*, 22:3816–3842, Oct 1980.

[34] E. Y. Vedmedenko, L. Udvardi, P. Weinberger, and R. Wiesendanger. Chiral magnetic ordering in two-dimensional ferromagnets with competing dzyaloshinsky-moriya interactions. *Phys. Rev. B*, 75:104431, Mar 2007.

[35] John M Hopkinson and Hae-Young Kee. Origin and consequences of unpinned helical order: Application to mnsi under pressure. *Physical review. B, Condensed matter and materials physics*, 79(1), 2009.

[36] F.N Rybakov, A.B Borisov, and A.N Bogdanov. Three-dimensional skyrmion states in thin films of cubic helimagnets. *Physical Review B - Condensed Matter and Materials Physics*, 87(9), 2013.

[37] A.O Leonov, Y Togawa, T.L Monchesky, A.N Bogdanov, J Kishine, Y Kousaka, M Miyagawa, T Koyama, J Akimitsu, Ts Koyama, K Harada, S Mori, D McGrouther, R Lamb, M Krajnak, S McVitie, R.L Stamps, and K Inoue. Chiral surface twists and skyrmion stability in nanolayers of cubic helimagnets. *Physical review letters*, 117(8):087202, 2016.

[38] Filipp N Rybakov, Aleksandr B Borisov, Stefan Blgel, and Nikolai S Kiselev. New type of stable particlelike states in chiral magnets. *Physical review letters*, 115(11):117201, 2015.

[39] J.C Gallagher, K.Y Meng, J.T Brangham, H.L Wang, B.D Esser, D.W McComb, and F.Y Yang. Robust zero-field skyrmion formation in fege epitaxial thin films. *Physical review letters*, 118(2):027201, 2017.

[40] S. A Meynell, M. N Wilson, J. C Loudon, A Spitzig, F. N Rybakov, M. B Johnson, and T. L Monchesky. Hall effect and transmission electron microscopy of epitaxial mnsi thin films. *Physical review. B, Condensed matter and materials physics*, 90(22), 2014.

[41] M. N Wilson, A. B Butenko, A. N Bogdanov, and T. L Monchesky. Chiral skyrmions in cubic helimagnet films: The role of uniaxial anisotropy. *Physical review. B, Condensed matter and materials physics*, 89(9), 2014.

[42] S. V Grigoriev, V. A Dyadkin, E. V Moskvin, D Lamago, Th Wolf, H Eckerlebe, and S. V Maleyev. Helical spin structure of under a magnetic field: Small angle neutron diffraction study. *Physical review. B, Condensed matter and materials physics*, 79(14), 2009.

[43] B Lebech, J Bernhard, and T Freltoft. Magnetic structures of cubic fege studied by small-angle neutron scattering. *Journal of physics. Condensed matter*, 1(35):6105–6122, 1989.

[44] O. L Makarova, A. V Tsvyashchenko, G Andre, F Porcher, L. N Fomicheva, N Rey, and I Mirebeau. Neutron diffraction study of the chiral magnet mnge. *Physical review. B, Condensed matter and materials physics*, 85(20), 2012.

[45] J Beille, J Voiron, F Towfiq, M Roth, and Z Y Zhang. Helimagnetic structure of the fexco1-xsi alloys. *Journal of Physics F: Metal Physics*, 11(10):2153–2160, 1981.

[46] K. Mardia and P. Jupp. *Directional Statistics*. John Wiley & Sons, Jan. 2000.

[47] Graham J.G. Upton and Bernard Fingleton. *Spatial Data Analysis by Example-Volume 2: Categorical and Directional Data*. John Wiley & Sons, 1989.

Appendix A

Selected Materials Magnetic Properties

Material	x	T_c (K)	λ (nm)	Helix Direction	B_{C2} (mT)	Ref.
$Mn_xFe_{1-x}Si$	0.90	6.8	9.9	$\langle 100 \rangle$ and $\langle 111 \rangle$	~ 650	[42]
	0.92	10.55	10.8		~ 700	
	0.94	16.5	12.8		~ 700	
MnSi		29.5	18.0	$\langle 111 \rangle$	~ 550	[2, 42]
FeGe		278.7 211/245[1]	70.0	$\langle 100 \rangle$ $\langle 111 \rangle$	$200 - 300$	[43]
MnGe		170	$4.0 - 6.0$	$\langle 100 \rangle$		[44]
$Fe_xCo_{1-x}Si$	0.3	8.8	233		~ 6	[43, 45]
	0.5	43.5	81.6		~ 40	
	0.65	58.8	47.2		145	
	0.8	37.4	29.5		180	
	0.9	21.3	43.0		54	

Table A.1: Magnetic properties of selected helical B20 materials. Included are the helical phase transition temperature T_c, helix wavelength λ and direction, and saturation field B_{C2}. Note that these values are dependent on temperature. See references for details.

[1] FeGe transitions from a helix along $\langle 100 \rangle$ to a helix along $\langle 111 \rangle$. This transition experiences hysteresis. $T_{2\downarrow} = 211$ represents a transition to the $\langle 111 \rangle$ direction, $T_{2\uparrow} = 245$ represents a transition to the $\langle 100 \rangle$.

Appendix B

Criteria for constant real magnitude

Any general vector of n complex numbers can be written as the sum of a real vector and complex vector

$$\mathbf{S} = \mathbf{R} + i\mathbf{J} \tag{B.1}$$

and therefore the magnitude of the vector is

$$|\mathbf{S}|^2 = |\mathbf{R}|^2 + |\mathbf{J}|^2. \tag{B.2}$$

This magnitude is invariant under rotations. These vectors are composed of the complex magnetic OPs of the exponential form $F_j \exp(i\phi_j)$ in which each component can be represented as one term that is the sum of all magnetic OPs. Therefore, the above magnitudes can be written

$$\begin{aligned} |\mathbf{R}|^2 &= \sum_{i=1}^{n} (F_i \cos(\phi_i))^2 \\ |\mathbf{J}|^2 &= \sum_{i=1}^{n} (F_i \sin(\phi_i))^2 \end{aligned} \tag{B.3}$$

I am concerned with the real magnitude only. The requirement that the real vector magnitude is constant requires that it does not vary under any additional phase δ. That is

$$|\mathbf{R}|^2 = |\exp(i\delta)\mathbf{R}|^2 = |\mathbf{R}_\delta|^2. \tag{B.4}$$

where the subscript δ represents the rotation. It is easy to see that such a phase will be represented in the above magnitudes as

$$\begin{aligned} |\mathbf{R}_\delta|^2 &= \sum_{i=1}^{n} (F_i \cos(\phi_i + \delta))^2 \\ |\mathbf{J}_\delta|^2 &= \sum_{i=1}^{n} (F_i \sin(\phi_i + \delta))^2 \end{aligned} \tag{B.5}$$

One may also show that, after this rotation, that the new real and imaginary vectors will be made up of portions of both of the original real and imaginary vectors. This is written

$$\begin{aligned} |\mathbf{R}_\delta|^2 &= \cos^2(\delta)|\mathbf{R}|^2 + \sin^2(\delta)|\mathbf{J}|^2 \\ |\mathbf{J}_\delta|^2 &= \sin^2(\delta)|\mathbf{R}|^2 + \cos^2(\delta)|\mathbf{J}|^2 \end{aligned} \tag{B.6}$$

and it is easy to see if we treat the full magnitude of each as real and imaginary numbers and rotate them accordingly. This relationship can be seen graphically in Fig. B.1.

The goal is $\mathbf{R} = \mathbf{R}_\delta$. Replacing each $\mathbf{R}$ with $\mathbf{R}_\delta$ and rearranging gives

$$|\mathbf{R}_\delta|^2 = |\mathbf{J}_\delta|^2 \tag{B.7}$$

for all δ. This condition for any single value δ is necessary, but not sufficient, to produce a constant magnitude. To produce a sufficient criteria one can note that this must be true for any δ. If one chooses two δ such that they do not differ by $\frac{n\pi}{2}$ – in which case both will correspond to the same set of solutions – then the solution to

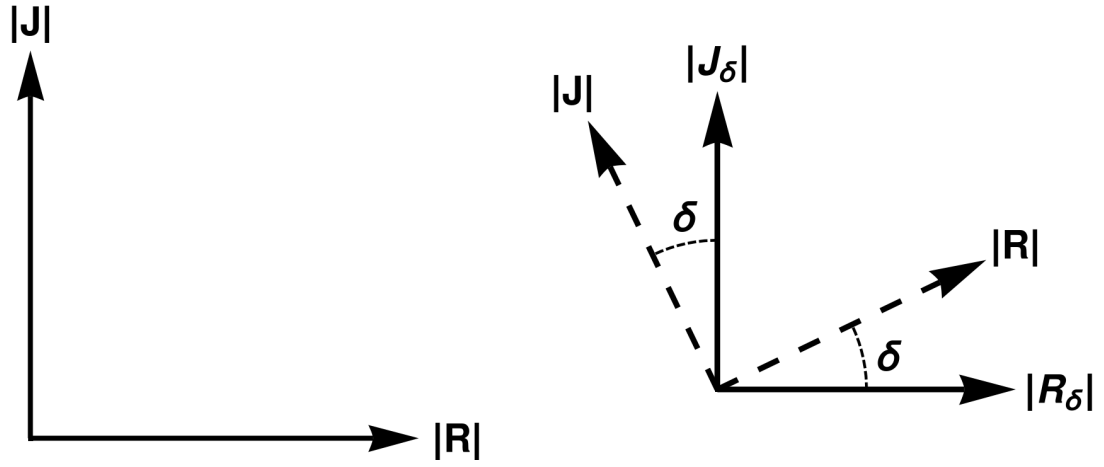
the two equations will be a state of constant real magnitude. These conditions can then be written as

$$\begin{aligned} \sum_{i=1}^{n}(\cos(\phi_i)F_i)^2 &= \sum_{i=1}^{n}(\sin(\phi_i)F_i)^2 \\ \sum_{i=1}^{n}(\cos(\phi_i+\delta)F_i)^2 &= \sum_{i=1}^{n}(\sin(\phi_i+\delta)F_i)^2 \end{aligned} \tag{B.8}$$

where in the first case $\delta = 0$ is chosen without loss of generality. This can be rewritten using the double-angle formula $\cos(2x) = \cos^2(x) - \sin^2(x)$ as

$$\begin{aligned} \sum_{i=1}^{n}\cos(2\phi_i)F_i^2 &= 0 \\ \sum_{i=1}^{n}\cos(2\phi_i+2\delta)F_i^2 &= 0 \end{aligned} \tag{B.9}$$

which are the criteria used in all discussions in this thesis.



(a) (b)

Figure B.1: A graphical representation of the constant magnitude criteria. (a) The real and imaginary magnitudes, $\mathbf{R}$ and $\mathbf{J}$. (b) The real and imaginary magnitudes of the rotated vectors, $\mathbf{R}_\delta$ and $\mathbf{J}_\delta$.

Appendix C

Directional Statistics

In this thesis, most of the measured parameters are – or are derived from – angular (*i.e.*, directional) data. When working with angular data, one must use specialized statistical methods to derive the analogues to the usual statistics, *e.g.*, mean or confidence intervals. As an example of complications that can occur when working with angular data using standard statistics, consider the branch cut of the angle at $\theta = 2\pi = 0$. If a set of data varies over this branch, the standard mean would be approximately πrad different from the true mean (See Fig. C.1). Due to this and many other complications, I use *directional statistics* as described by Mardia & Jupp [46], throughout this thesis.

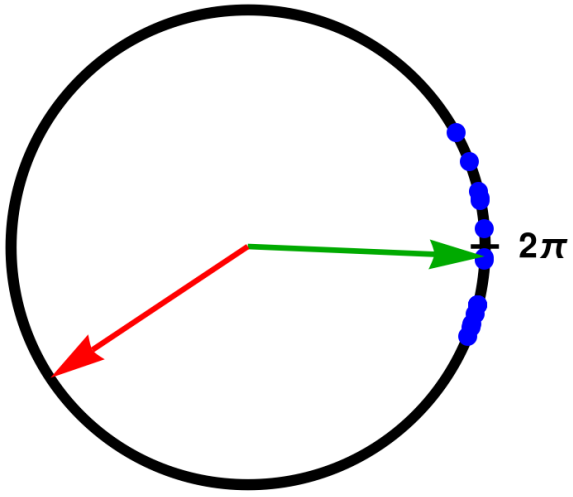


Figure C.1: A graphical representation of the mean of random angular data (blue) as measured by standard statistics (red) and directional statistics (green).

Of particular note here are the calculation of the mean and the confidence intervals used throughout the thesis. These values will both be derived from the average value of the trigonometric functions

$$\bar{C} = \frac{1}{N}\sum_j \cos\theta_j \qquad\qquad \bar{S} = \frac{1}{N}\sum_j \sin\theta_j \tag{C.1}$$

for data θ_j for $1 \leq j \leq N$. These averages are then used to define two values: The mean angular direction

$$\bar{\theta} = \arctan\left(\frac{\bar{S}}{\bar{C}}\right) + n\pi \tag{C.2}$$

where n is a correction to the appropriate quadrant handled by the $atan2$ function in most programming languages.

The second value is the *mean resultant length*

$$\bar{R} = \sqrt{\bar{C} + \bar{S}} \tag{C.3}$$

which can be shown to be the norm of the average vector associated with the data. This value is used in measurements of most statistical values, along with the *resultant length* $R = N\bar{R}$. In this thesis I make no assumptions on the distribution of the data, and therefore use the confidence interval for the mean direction defined by Upton & Fingleton [47]

$$\delta_{T,\alpha} = \arcsin\left(z_{\alpha/2}\sqrt{\frac{N(1-H)}{4R^2}}\right) \tag{C.4}$$

where $z_{\alpha/2}$ is the z-score of the $(1.00-\alpha/2)*100$th percentile of the normal distribution. For example, the 95th percentile, which is used in this thesis, is $z_{0.05} = 1.96$. Therefore, the interval associated with this has a confidence of 90%. The value H is given by

$$H = \frac{1}{N}\left[\cos(2\bar{\theta})\sum_j \cos(2\theta_j) + \sin(2\bar{\theta})\sum_j \sin(2\theta_j)\right]. \tag{C.5}$$

This is the value reported in all pertinent data.

Note that all the details presented here are two-dimensional. The systems considered in this thesis are three-dimensional, and there are different statistics developed for three-dimensional data. However, the angles considered here, θ and γ as shown in Fig. 3.3, are concerned with two-dimensional circles and the above analyses are applicable.

Appendix D

Simulation Code

D.1 Fortran Code

This appendix contains the computer code used in the EFM simulations in three parts: The input file (MnSi_Input.f), the simulation code (SIM_MAIN.f), and a module containing datatype definitions (type.f). This code was prepared for the Fortran 2008 standard.

D.1.1 type

```
MODULE type_module
    IMPLICIT NONE

    INTEGER, PARAMETER, PUBLIC :: sp = KIND(1.0)
    INTEGER, PARAMETER, PUBLIC :: dp = SELECTED_REAL_KIND(2*PRECISION(1.0_sp))

END MODULE type_module
```

D.1.2 MnSi_Input

```
MODULE input_module
    USE type_module, ONLY : dp

!Simulation type declaration!!!!

CHARACTER*10, PARAMETER :: SIM_TYPE="SEFM"    !At the moment this is limited to EFM
    and SEFM
CHARACTER*10, PARAMETER :: MC_TYPE="HEATING"    !Heating/Cooling/Isolated. Note that
    only heating changes simulation, all other are cooling

!Constants
REAL( kind=dp ), PARAMETER :: PI = 4.0_dp*DATAN(1.0_dp) !Pi

!Lattice setup
REAL( kind=dp ), DIMENSION(1:3), PARAMETER :: LAT_VECTOR = (/ 4.558, 4.558, 4.558 /)
     !The length of the lattice vectors in each DIMENSION in Angstroms
REAL( kind=dp ), PARAMETER :: X_VALUE = 0.138    !Distance x used for positioning
    spins inside unit cell

INTEGER, PARAMETER :: SPINS_PER_CELL = 4    !Number of spins per unit cell
INTEGER, PARAMETER :: CELLS = 8    !Total number of unit cells to be used (in each
    DIMENSION: DIMENSION hardcoded to 3)
INTEGER, PARAMETER :: NUM_SPINS = SPINS_PER_CELL*(CELLS**3)    !Number of spins in the
     system

!Simulation setup
REAL( kind=dp ), DIMENSION(1:3), PARAMETER :: B_APP = (/ 0.2d0,0.2d0,0.2d0 /) !
    Applied magnetic field
REAL( kind=dp ), DIMENSION(1:2, 1:13), PARAMETER :: J_EX = Reshape((/ &
1.00d0, 0.00d0, &
0.00d0, 0.00d0, &
0.00d0, 0.00d0, &
0.00d0, 0.00d0, &
0.00d0, 0.00d0, &
0.00d0, 0.00d0, &
0.00d0, 0.00d0, &
0.00d0, 0.00d0, &
0.00d0, 0.00d0, &
0.00d0, 0.00d0, &
0.00d0, 0.00d0, &
0.00d0, 0.00d0, &
0.00d0, 0.00d0 /), (/2,13/)) !The 2*13 exchange coupling constants for NN and NNN

INTEGER, PARAMETER :: STEPS = 5000  !Number of steps to be used for Simulation
LOGICAL, PARAMETER :: PERIODIC_BOUNDS = .FALSE. !If true, periodic boundary
    conditions will be applied
LOGICAL, PARAMETER :: PRESET_INIT = .FALSE. !If true, the code will initialize the
    lattice using the filename below (METROMC ONLY)
LOGICAL, PARAMETER :: CALC_Q = .FALSE.    !If true, the fourier transform of spins
    will be produced
```

```
! TODO: Add ability to pick specific coordinates and spins. Easy, just add an array.

!EFM
INTEGER, PARAMETER :: NUM_CONFIG=1000  ! The number of random configurations to use
    in an EFM simulation.
INTEGER, PARAMETER :: NUM_MIN_OUT=10 ! The number of minima files to be output.
INTEGER, PARAMETER :: EFM_RATIO=10        ! The ratio of the angle between two vectors
     to be rotated
REAL( kind=dp ), PARAMETER :: PERCENT_OUTPUT=0d-3  ! The percentage of random
    configuration results to be output

!METROMC
INTEGER, PARAMETER :: EQ_STEPS = 250000 !Number of equilibriation steps
REAL( kind=dp ), PARAMETER :: MAX_TEMP=0.1, MIN_TEMP=MAX_TEMP    !Maximum and minimum
    temperatures for Monte Carlo simulations
INTEGER, PARAMETER :: STEP_TEMP=1    !The number of steps to take between min and max
    temperature. Set to 1 for isolated temperature
INTEGER, PARAMETER :: STEPS_PER_OUT = 10000    ! Number of spins between a file is
    output
CHARACTER*14, PARAMETER :: PRESET_FILE="spin_input.dat" !The file name for the spin
    input file. Set CHARACTER*# to the appropriate length.
    !   Potentially change to "allocatable" to allow for dynamic string length

!Input variable declaration!!!!

REAL( kind=dp ), DIMENSION(0:NUM_SPINS, 1:3) :: spin    !Array containing all spin
    components (x=1, y=2, z=3)
REAL( kind=dp ), DIMENSION(0:NUM_SPINS, 1:3) :: p_Table  !Tables containing spin
    position information

REAL( kind=dp ) :: energy, mag, chi !Instantaneous observable values
! REAL( kind=dp ) :: eav, mav, cav    !Average observable values

INTEGER, DIMENSION(0:CELLS-1, 0:CELLS-1, 0:CELLS-1, 1:SPINS_PER_CELL) :: u_Table    !
    Table containing unit cell position of each spin. Defined by minimum corner.
INTEGER, DIMENSION(1:NUM_SPINS, 1:4) :: invu_Table  !Inverse lookup table for unit
    cell
INTEGER, DIMENSION(1:NUM_SPINS, 1:2, 1:6) :: n_Neighbour !Table containing the 6
    nearest and next nearest neighbours of each spin

common /tables/ p_Table, u_Table, invu_Table, n_Neighbour, nn_Neigbour !Table used
    for positions
common /spins/ spin !Tables used for spin components

END MODULE input_module
```

D.1.3 Sim_Main

```
! Kyle Hall 2018

PROGRAM SIM_MAIN
    USE input_module, ONLY : SIM_TYPE, CELLS, PERIODIC_BOUNDS, PRESET_INIT, CALC_Q,
        PRESET_FILE, J_EX
    IMPLICIT NONE

    INTEGER :: i, j

write(*,*) "// Run_Sim is set to a ", SIM_TYPE, "simulation with ... "
write(*,*) "    Lattice Size: ", CELLS
write(*,*) "    Periodic Bounds: ", PERIODIC_BOUNDS
write(*,*) "    Preset Initialization: ", PRESET_INIT
if(PRESET_INIT) then
    write(*,*) "        Preset file name: ", PRESET_FILE
endif
write(*,*) "    Calculate FFT:", CALC_Q
write(*,*) "    NN Coupling constants:"
do i=1,3
    write(*,fmt='(A,F7.3,3("  ",X,F7.3))') "        ",J_EX(1,(i-1)*4+1), J_EX(1,(i-1)
        *4+2), J_EX(1,(i-1)*4+3), J_EX(1,(i-1)*4+4)
enddo !i
write(*,fmt='(A,F7.3,2("  ",X,F7.3))') "        ",J_EX(1,13)
write(*,*) "    NNN Coupling constants:"
do i=1,3
    write(*,fmt='(A,F7.3,3("  ",X,F7.3))') "        ",J_EX(2,(i-1)*4+1), J_EX(2,(i-1)
        *4+2), J_EX(2,(i-1)*4+3), J_EX(2,(i-1)*4+4)
enddo !i
write(*,fmt='(A,F7.3,2("  ",X,F7.3))') "        ",J_EX(2,13)
write(*,*)

call init_random_seed()

write(*,*) "// Building lattice ..."
call Lattice_Build

write(*,*) "// Finding nearest neighbours ..."
call Find_NN


write(*,*) "// Starting ", SIM_TYPE, "simulation ..."
if(SIM_TYPE.eq."SEFM") then
    call STEPPED_EFM
else if(SIM_TYPE.eq."EFM") then
    call EFM
else
    write(*,*) "METROMC is not currently available"
    ! call METROMC
endif   !SIM_TYPE
```

```
END PROGRAM SIM_MAIN

!
    !!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!

!SUBROUTINE: Lattice_Build
!   Defines lattice positions and stores them as appropriate in spin, p_table,
    u_table, and the inverse tables.

SUBROUTINE Lattice_Build
    USE input_module, ONLY : p_Table, SPINS_PER_CELL, CELLS, X_VALUE, u_Table,
        invu_Table
    IMPLICIT NONE

    INTEGER :: i,x,y,z,spin_Num

do i=1,3
    p_Table(1,i)=X_VALUE  !First position is (X,X,X)
enddo !i

p_Table(2,1)=0.5d0-X_VALUE    !Second position as defined by S. H. Curnoe shifted
    (0,1,0) units
p_Table(2,2)=1-X_VALUE
p_Table(2,3)=0.5d0+X_VALUE

p_Table(3,1)=1-X_VALUE        !Third position as defined by S. H. Curnoe shifted
    (1,0,0) units
p_Table(3,2)=0.5d0+X_VALUE
p_Table(3,3)=0.5d0-X_VALUE

p_Table(0,1)=0.5d0+X_VALUE    !Fourth position as defined by S. H. Curnoe shifted
    (0,0,1) units (written as 0 for mod 4)
p_Table(0,2)=0.5d0-X_VALUE
p_Table(0,3)=1-X_VALUE

!Expand first unit cell to $CELLS
spin_Num=0   !Initialize the counter for spin to current spin

open(unit=12, file="latticeSites.dat", position="APPEND", action="WRITE", status="
    REPLACE") !Output for positions of the spin

do x=0,CELLS-1
    do y=0,CELLS-1
        do z=0,CELLS-1
            do i=1,SPINS_PER_CELL    !i acts as Atom # as defined by S. H. Curnoe,
                shifted as above
                spin_Num=spin_Num+1

                p_Table(spin_Num,1)=p_Table(Modulo(spin_Num,4),1)+x !Put spin in
                    lattice
                p_Table(spin_Num,2)=p_Table(Modulo(spin_Num,4),2)+y
                p_Table(spin_Num,3)=p_Table(Modulo(spin_Num,4),3)+z
```

```
                u_Table(x,y,z,i)=spin_Num      !Initialize unit cell lookups
                invu_Table(spin_Num,1)=x
                invu_Table(spin_Num,2)=y
                invu_Table(spin_Num,3)=z
                invu_Table(spin_Num,4)=i

                write(unit=12, fmt=*) p_Table(spin_Num, 1), p_Table(spin_Num, 2),
                    p_Table(spin_Num, 3), &
                    invu_Table(spin_Num,1), invu_Table(spin_Num,2), invu_Table(
                        spin_Num,3), invu_Table(spin_Num,4)

            enddo !i
        enddo !x
    enddo !y
enddo !z

flush(unit=12)          !Clear and close output
close(unit=12)

END SUBROUTINE Lattice_Build

!
    !!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!

!SUBROUTINE: Initialize_Spin_Random
!   Initializes all spin on the lattice to a random configuration.
!   $PRESET_INIT is set to FALSE

SUBROUTINE Initialize_Spin_Random
    USE input_module, ONLY : dp, PI, NUM_SPINS, spin
    IMPLICIT NONE

    INTEGER :: i
    REAL( kind=dp ) :: theta, phi

    open(unit=12, file="initialSpinData.dat", position="APPEND", action="WRITE",
        status="REPLACE") !Output for initial spin values

    do i=1,NUM_SPINS

        call random_number(theta)
        theta=PI*theta    !Get random theta and phi angles
        call random_number(phi)
        phi=2*PI*phi

        spin(i,1)=dsin(theta)*dcos(phi)    !Use spherical coordinates to place spin
            on lattice
        spin(i,2)=dsin(theta)*dsin(phi)
        spin(i,3)=dcos(theta)

        write(unit=12, fmt=*) spin(i, 1), spin(i, 2), spin(i, 3)
```

```fortran
    enddo !i

    flush(unit=12)
    close(unit=12)

END SUBROUTINE Initialize_Spin_Random

!
    !!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!

!SUBROUTINE: Initialize_Spin_Preset
!   Initializes all spin on the lattice to a preset configuration.
!   $PRESET_INIT is set to TRUE

SUBROUTINE Initialize_Spin_Preset
    USE input_module, ONLY : NUM_SPINS, spin, PRESET_FILE
    IMPLICIT NONE

    INTEGER :: i !Subroutine iterator

    !!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!

    open(unit=13, file=PRESET_FILE, action="READ") !Input for spin initialization

    do i=1,NUM_SPINS

        read(unit=13,fmt=*) spin(i,1), spin(i,2), spin(i,3)

    enddo ! i

    flush(unit=13)    !Clear and close input/output
    close(unit=13)

    open(unit=12, file="initialSpinData.dat", position="APPEND", action="WRITE",
        status="REPLACE") !Output for initial spin values

    do i=1,NUM_SPINS

        write(unit=12,fmt=*) spin(i,1), spin(i,2), spin(i,3)

    enddo ! i

    flush(unit=12)    !Clear and close input/output
    close(unit=12)

END SUBROUTINE Initialize_Spin_Preset

!
    !!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!

!SUBROUTINE: Find_NN
```

```
!   Finds the nearest neighbour of each spin in the lattice and stores them in table
    n_neighbour
!   $PERIODIC_BOUNDS is FALSE and periodic boundary conditions will not be applied.

SUBROUTINE Find_NN()
    USE input_module, ONLY : PERIODIC_BOUNDS, u_Table, invu_Table, n_Neighbour, CELLS
        , NUM_SPINS
    IMPLICIT NONE

    INTEGER :: x_pl, y_pl, z_pl, x_min, y_min, z_min !Incremented and decremented
        value holders
    INTEGER :: a, x, y, z !Holds atom number and position
    INTEGER :: i !Subroutine iterator

    DO i=1,NUM_SPINS
        x=invu_Table(i,1)    !x,y,z hold the unit cell position
        y=invu_Table(i,2)
        z=invu_Table(i,3)
        a=invu_Table(i,4)    !a holds the atom # of the current atom

        x_min=x-1    !Set the corresponding values
        x_pl=x+1
        y_min=y-1
        y_pl=y+1
        z_min=z-1
        z_pl=z+1

        IF(PERIODIC_BOUNDS) THEN
            if(x_min.lt.0) x_min=CELLS-1     !Check for periodic boundary conditions
            if(x_pl.gt.(CELLS-1)) x_pl=0
            if(y_min.lt.0) y_min=CELLS-1
            if(y_pl.gt.(CELLS-1)) y_pl=0
            if(z_min.lt.0) z_min=CELLS-1
            if(z_pl.gt.(CELLS-1)) z_pl=0
        ENDIF
        if(a.eq.1) then !Definitions of the 6 NN for each spin position in a unit
            cell. Ordered by atom # then number of unit cells away.
            n_Neighbour(i,1,1)=merge(0,u_Table(x,y_min,z,2), y_min.lt.0)      !Check
                for those outside of lattice and set to zero if true
            n_Neighbour(i,1,2)=merge(0,u_Table(x,y_min,z_min,2), z_min.lt.0 .OR.
                y_min.lt.0)
            n_Neighbour(i,1,3)=merge(0,u_Table(x_min,y,z,3), x_min.lt.0)
            n_Neighbour(i,1,4)=merge(0,u_Table(x_min,y_min,z,3), y_min.lt.0 .OR.
                x_min.lt.0)
            n_Neighbour(i,1,5)=merge(0,u_Table(x,y,z_min,4), z_min.lt.0)
            n_Neighbour(i,1,6)=merge(0,u_Table(x_min,y,z_min,4), x_min.lt.0 .OR.
                z_min.lt.0)

            n_Neighbour(i,2,1)=u_Table(x,y,z,2)  !Those in same unit cell can not be
                outside lattice.
            n_Neighbour(i,2,2)=merge(0,u_Table(x,y,z_min,2), z_min.lt.0)
            n_Neighbour(i,2,3)=u_Table(x,y,z,3)
```

```
    n_Neighbour(i,2,4)=merge(0,u_Table(x,y_min,z,3), y_min.lt.0)
    n_Neighbour(i,2,5)=u_Table(x,y,z,4)
    n_Neighbour(i,2,6)=merge(0,u_Table(x_min,y,z,4), x_min.lt.0)
else if(a.eq.2) then
    n_Neighbour(i,1,1)=merge(0,u_Table(x,y_pl,z,1), y_pl.gt.(CELLS-1))
    n_Neighbour(i,1,2)=merge(0,u_Table(x,y_pl,z_pl,1), z_pl.gt.(CELLS-1) .OR.
         y_pl.gt.(CELLS-1))
    n_Neighbour(i,1,3)=u_Table(x,y,z,3)
    n_Neighbour(i,1,4)=merge(0,u_Table(x_min,y,z,3), x_min.lt.0)
    n_Neighbour(i,1,5)=u_Table(x,y,z,4)
    n_Neighbour(i,1,6)=merge(0,u_Table(x,y_pl,z,4), y_pl.gt.(CELLS-1))

    n_Neighbour(i,2,1)=u_Table(x,y,z,1)
    n_Neighbour(i,2,2)=merge(0,u_Table(x,y,z_pl,1), z_pl.gt.(CELLS-1))
    n_Neighbour(i,2,3)=merge(0,u_Table(x,y,z_pl,3), z_pl.gt.(CELLS-1))
    n_Neighbour(i,2,4)=merge(0,u_Table(x_min,y,z_pl,3), x_min.lt.0 .OR. z_pl.
        gt.(CELLS-1))
    n_Neighbour(i,2,5)=merge(0,u_Table(x_min,y,z,4), x_min.lt.0)
    n_Neighbour(i,2,6)=merge(0,u_Table(x_min,y_pl,z,4), x_min.lt.0 .OR. x_pl.
        gt.(CELLS-1))
else if(a.eq.3) then
    n_Neighbour(i,1,1)=merge(0,u_Table(x_pl,y,z,1), x_pl.gt.(CELLS-1))
    n_Neighbour(i,1,2)=merge(0,u_Table(x_pl,y_pl,z,1), x_pl.gt.(CELLS-1) .OR.
         y_pl.gt.(CELLS-1))
    n_Neighbour(i,1,3)=u_Table(x,y,z,2)
    n_Neighbour(i,1,4)=merge(0,u_Table(x_pl,y,z,2), x_pl.gt.(CELLS-1))
    n_Neighbour(i,1,5)=u_Table(x,y,z,4)
    n_Neighbour(i,1,6)=merge(0,u_Table(x,y,z_min,4), z_min.lt.0)

    n_Neighbour(i,2,1)=u_Table(x,y,z,1)
    n_Neighbour(i,2,2)=merge(0,u_Table(x,y_pl,z,1), y_pl.gt.(CELLS-1))
    n_Neighbour(i,2,3)=merge(0,u_Table(x,y,z_min,2), z_min.lt.0)
    n_Neighbour(i,2,4)=merge(0,u_Table(x_pl,y,z_min,2), x_pl.gt.(CELLS-1) .OR
        . z_min.lt.0)
    n_Neighbour(i,2,5)=merge(0,u_Table(x,y_pl,z,4), y_pl.gt.(CELLS-1))
    n_Neighbour(i,2,6)=merge(0,u_Table(x,y_pl,z_min,4), y_pl.gt.(CELLS-1) .OR
        . z_min.lt.0)
else     ! atom # 4 as defined by S. H. Curnoe.
    n_Neighbour(i,1,1)=merge(0,u_Table(x,y,z_pl,1), z_pl.gt.(CELLS-1))
    n_Neighbour(i,1,2)=merge(0,u_Table(x_pl,y,z_pl,1), x_pl.gt.(CELLS-1) .OR.
         z_pl.gt.(CELLS-1))
    n_Neighbour(i,1,3)=u_Table(x,y,z,2)
    n_Neighbour(i,1,4)=merge(0,u_Table(x,y_min,z,2), y_min.lt.0)
    n_Neighbour(i,1,5)=u_Table(x,y,z,3)
    n_Neighbour(i,1,6)=merge(0,u_Table(x,y,z_pl,3), z_pl.gt.(CELLS-1))

    n_Neighbour(i,2,1)=u_Table(x,y,z,1)
    n_Neighbour(i,2,2)=merge(0,u_Table(x_pl,y,z,1), x_pl.gt.(CELLS-1))
    n_Neighbour(i,2,3)=merge(0,u_Table(x_pl,y,z,2), x_pl.gt.(CELLS-1))
    n_Neighbour(i,2,4)=merge(0,u_Table(x_pl,y_min,z,2), x_pl.gt.(CELLS-1) .OR
        . y_min.lt.0)
    n_Neighbour(i,2,5)=merge(0,u_Table(x,y_min,z,3), y_min.lt.0)
```

```
            n_Neighbour(i,2,6)=merge(0,u_Table(x,y_min,z_pl,3), y_min.lt.0 .OR. z_pl.
                gt.(CELLS-1))
        endif
    END DO
END SUBROUTINE Find_NN

!
    !!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!

!SUBROUTINE: EFM
!

SUBROUTINE EFM
    USE input_module, ONLY : dp, spin, STEPS, NUM_SPINS, B_APP, energy, NUM_CONFIG,
        CALC_Q, PERCENT_OUTPUT, NUM_MIN_OUT, PRESET_INIT
    IMPLICIT NONE

    REAL( kind=dp ), DIMENSION(1:3) :: B_Eff, new_Spin    ! The effective field and
        new spin components in each dimension
    REAL( kind=dp ) :: new_Spin_L, e_Start, rand
    REAL( kind=dp ), DIMENSION(1:NUM_MIN_OUT) :: mfile_e
    INTEGER :: i,j,k,m,n ! Subroutine iterator
    INTEGER, DIMENSION(1:NUM_SPINS) :: spin_Order(1:NUM_SPINS) = (/(i, i=1,NUM_SPINS,
        1)/)  !Initialize the spin ordering.
    INTEGER, DIMENSION(1:NUM_MIN_OUT) :: mfile_c    !Store config number for minimums
        to be output at end of simulation
    CHARACTER*15 :: output_file

    INTERFACE
        SUBROUTINE Shuffle(arr)
            INTEGER, INTENT(INOUT) :: arr(:)
        END SUBROUTINE Shuffle
        SUBROUTINE Get_B_Eff(B_Eff, ord, spin_Curr)
            USE input_module, ONLY : dp, n_Neighbour, invu_table, spin, J_EX
            IMPLICIT NONE
            REAL( kind=dp ), INTENT(INOUT) :: B_EFF(:)
            INTEGER, INTENT(IN) :: spin_Curr
            INTEGER, INTENT(IN) :: ord
        END SUBROUTINE Get_B_Eff
    END INTERFACE

    write(*,*) "    Number of configurations: ", NUM_CONFIG
    write(*,*) "    Steps per configuration: ", STEPS
    write(*,*) "    Percentage of runs output: ", PERCENT_OUTPUT*100, "%"
    mfile_e=huge(mfile_e)
    do n=1,NUM_CONFIG                    !Potential parallelization
        IF(PRESET_INIT) then
            CALL Initialize_Spin_Preset
        ELSE
            CALL Initialize_Spin_Random
        ENDIF
        call measure
```

```
e_Start=energy

do i=1,STEPS                        !Can not parallelize
    call Shuffle(spin_Order)

    do j=1, NUM_SPINS               !Can not parallelize
        B_Eff(1:3)=(/ 0.0_dp, 0.0_dp, 0.0_dp /)

        call Get_B_Eff(B_Eff, 1, spin_Order(j))     !Get NN Effective Field
        call Get_B_Eff(B_Eff, 2, spin_Order(j))     !Get NNN Effective Field

        do k=1,3     !Calculate new spin components from B_Eff
            new_Spin(k)=B_APP(k)+B_Eff(k)
        enddo !k

        if(new_Spin(1).eq.0 .AND. new_Spin(2).eq.0 .AND. new_Spin(3).eq.0)
            goto 92    ! For spins in corner with no change

        new_Spin_L=dsqrt(new_Spin(1)**2+new_Spin(2)**2+new_Spin(3)**2)
        new_Spin=new_Spin/new_Spin_L

        spin(spin_Order(j),:)=new_Spin(:)

        92 continue
    enddo !j
enddo !i

call measure
open(unit=15, file="EnergyMeasurement.dat", position="APPEND", action="WRITE"
    , status="UNKNOWN")
write(unit=15,fmt=*) n, e_Start/NUM_SPINS, energy/NUM_SPINS
flush(unit=15)
close(unit=15)
call random_number(rand)
if(energy.lt.MAXVAL(mfile_e)) then
    m=MAXLOC(mfile_e, 1)
    mfile_e(m)=energy
    mfile_c(m)=n
    write(output_file, '(A8,I3.3,A4)') 'MinConf_', m, ".dat"
    write(*,*) "New Minimum"
    write(*,*) "    Config:", n, " Energy: ", energy/NUM_SPINS
    write(*,*) "Overwriting ", output_file
    open(unit=16, file=output_file, action="WRITE", status="REPLACE")
    do i=1,NUM_SPINS
        write(unit=16, fmt=*) spin(i,1),spin(i,2),spin(i,3)
    enddo !j
    flush(unit=16)
    close(unit=16)

    !At this time, all spins and directions will be calculated
    if(CALC_Q) then
        write(*,*) "// Calculating FFT..."
```

```
                do i=1,4
                    write(*,*) "    Spin Number: ", i
                do j=1,3
                    call FFT(i, j)
                enddo !j
                enddo !i
            endif !CALC_Q

        else if(rand.le.PERCENT_OUTPUT) then
            write(*,*) "Random output"
            write(*,*) "    Config:", n, " Energy: ", energy/NUM_SPINS
            write(output_file,'(A5,I6.6,A4)') "Conf_", n, ".dat"
            write(*,*) "Writing to ", output_file
            open(unit=16, file=output_file, position="APPEND", action="WRITE", status
                ="NEW")
            do i=1,NUM_SPINS
                write(unit=16, fmt=*) spin(i,1),spin(i,2),spin(i,3)
            enddo !i
            flush(unit=16)
            close(unit=16)

        endif !output
    enddo !n

    open(unit=16, file="MinEnergy.dat", action="write", status="replace")    !Write
        out MinConfig information
    do i=1,NUM_MIN_OUT
        write(unit=16, fmt='(I3.3,I10,F20.15)') i, mfile_c(i), mfile_e(i)/NUM_SPINS
    enddo !i
    flush(unit=16)
    close(unit=16)

END SUBROUTINE EFM

SUBROUTINE STEPPED_EFM
    USE input_module, ONLY : dp, spin, STEPS, NUM_SPINS, B_APP, energy, NUM_CONFIG,
        CALC_Q,&
     PERCENT_OUTPUT, NUM_MIN_OUT, PRESET_INIT, EFM_RATIO, PI, n_Neighbour
    IMPLICIT NONE

    REAL( kind=dp ), DIMENSION(1:3) :: B_Eff, new_Spin, c    ! The effective field,
        new spin components and cross product in each dimension
    REAL( kind=dp ) :: new_Spin_L, e_Start, rand, theta, dot
    REAL( kind=dp ), DIMENSION(1:NUM_MIN_OUT) :: mfile_e
    REAL( kind=dp ), DIMENSION(1:3,1:3) :: rot,t_rot
    REAL( kind=dp ), DIMENSION(1:3,1:3), PARAMETER :: IDEN = &
    Reshape((/ 1.0_dp, 0.0_dp, 0.0_dp, 0.0_dp, 1.0_dp, 0.0_dp, 0.0_dp, 0.0_dp, 1.0_dp
        /),(/3,3/))
    INTEGER :: i,j,k,m,n ! Subroutine iterator
    INTEGER, DIMENSION(1:NUM_SPINS) :: spin_Order(1:NUM_SPINS) = (/(i, i=1,NUM_SPINS,
        1)/)  !Initialize the spin ordering.
```

```
INTEGER, DIMENSION(1:NUM_MIN_OUT) :: mfile_c     !Store config number for minimums
     to be output at end of simulation
CHARACTER*15 :: output_file

INTERFACE
    SUBROUTINE Shuffle(arr)
        INTEGER, INTENT(INOUT) :: arr(:)
    END SUBROUTINE Shuffle
    SUBROUTINE Get_B_Eff(B_Eff, ord, spin_Curr)
        USE input_module, ONLY : dp, n_Neighbour, invu_table, spin, J_EX
        IMPLICIT NONE
        REAL( kind=dp ), INTENT(INOUT) :: B_EFF(:)
        INTEGER, INTENT(IN) :: spin_Curr
        INTEGER, INTENT(IN) :: ord
    END SUBROUTINE Get_B_Eff
    FUNCTION Cross(a, b) result(c)
        USE input_module, ONLY : dp
        REAL( kind=dp ), DIMENSION(1:3) :: c
        REAL( kind=dp ), DIMENSION(1:3), INTENT(IN) :: a, b
    END FUNCTION
END INTERFACE

write(*,*) "    Number of configurations: ", NUM_CONFIG
write(*,*) "    Steps per configuration: ", STEPS
write(*,*) "    Percentage of runs output: ", PERCENT_OUTPUT*100, "%"
mfile_e=huge(mfile_e)
do n=1,NUM_CONFIG                    !Potential parallelization
    IF(PRESET_INIT) then
        CALL Initialize_Spin_Preset
    ELSE
        CALL Initialize_Spin_Random
    ENDIF
    call measure
    e_Start=energy

    do i=1,STEPS                     !Can not parallelize
        call Shuffle(spin_Order)

        do j=1, NUM_SPINS            !Can not parallelize
            B_Eff(1:3)=(/ 0.0_dp, 0.0_dp, 0.0_dp /)

            call Get_B_Eff(B_Eff, 1, spin_Order(j))     !Get NN Effective Field
            call Get_B_Eff(B_Eff, 2, spin_Order(j))     !Get NNN Effective Field

            new_Spin=B_APP+B_Eff !Calculate new spin components from B_Eff

            if(new_Spin(1).eq.0 .AND. new_Spin(2).eq.0 .AND. new_Spin(3).eq.0)
                goto 92    ! For spins in corner with no change

            new_Spin_L=dsqrt(new_Spin(1)**2+new_Spin(2)**2+new_Spin(3)**2)
            new_Spin=new_Spin/new_Spin_L
```

```
            c=Cross(spin(spin_Order(j),:),new_Spin)
            if(c(1).eq.0 .AND. c(2).eq.0 .AND. c(3).eq.0) goto 92    ! If spins
                are aligned to B_Eff, no change is required

            c=c/dsqrt(c(1)**2+c(2)**2+c(3)**2)
            dot=dot_product(spin(spin_Order(j),:),new_Spin)
            if(dot.gt.1.0_dp) then  !Insure the dot product is within approrpiate
                 parameters [-1,1]
                dot=1.0_dp
            else if(dot.lt.(-1.0_dp)) then
                dot=-1.0_dp
            endif
            theta=DACOS(dot)
            IF (theta.gt.PI) THEN    ! This may never be needed...
                theta=(PI-theta)
            ENDIF
            theta=theta/EFM_RATIO

            rot(1,:)= (/ 0.0_dp, -c(3), c(2) /)
            rot(2,:)= (/ c(3), 0.0_dp, -c(1) /)
            rot(3,:)= (/ -c(2), c(1), 0.0_dp /)
            rot=IDEN+dsin(theta)*rot+(1-dcos(theta))*MATMUL(rot,rot)

            new_Spin=MATMUL(rot,spin(spin_Order(j),:))
            new_Spin_L=dsqrt(new_Spin(1)**2+new_Spin(2)**2+new_Spin(3)**2)
            new_Spin=new_Spin/new_Spin_L

            spin(spin_Order(j),:)=new_Spin(:)
            92 continue
        enddo !j
    enddo !i

    call measure
    open(unit=15, file="EnergyMeasurement.dat", position="APPEND", action="WRITE"
        , status="UNKNOWN")
    write(unit=15,fmt=*) n, e_Start/NUM_SPINS, energy/NUM_SPINS
    flush(unit=15)
    close(unit=15)
    call random_number(rand)
    if(energy.lt.MAXVAL(mfile_e)) then
        m=MAXLOC(mfile_e, 1)
        mfile_e(m)=energy
        mfile_c(m)=n
        write(output_file, '(A8,I3.3,A4)') 'MinConf_', m, ".dat"
        write(*,*) "New Minimum"
        write(*,*) "    Config:", n, " Energy: ", energy/NUM_SPINS
        write(*,*) "Overwriting ", output_file
        open(unit=16, file=output_file, action="WRITE", status="REPLACE")
        do i=1,NUM_SPINS
            write(unit=16, fmt=*) spin(i,1),spin(i,2),spin(i,3)
        enddo !j
        flush(unit=16)
```

```
        close(unit=16)

        !At this time, all spins and directions will be calculated
        if(CALC_Q) then
          write(*,*) "// Calculating FFT..."
          do i=1,4
            write(*,*) "    Spin Number: ", i
          do j=1,3
            call FFT(i, j)
          enddo !j
          enddo !i
        endif !CALC_Q

      else if(rand.le.PERCENT_OUTPUT) then
        write(*,*) "Random output"
        write(*,*) "    Config:", n, " Energy: ", energy/NUM_SPINS
        write(output_file,'(A5,I6.6,A4)') "Conf_", n, ".dat"
        write(*,*) "Writing to ", output_file
        open(unit=16, file=output_file, position="APPEND", action="WRITE", status
          ="NEW")
        do i=1,NUM_SPINS
          write(unit=16, fmt=*) spin(i,1),spin(i,2),spin(i,3)
        enddo !i
        flush(unit=16)
        close(unit=16)

      endif !output
    enddo !n

    open(unit=16, file="MinEnergy.dat", action="write", status="replace")    !Write
      out MinConfig information
    do i=1,NUM_MIN_OUT
      write(unit=16, fmt='(I3.3,I10,F20.15)') i, mfile_c(i), mfile_e(i)/NUM_SPINS
    enddo !i
    flush(unit=16)
    close(unit=16)

END SUBROUTINE STEPPED_EFM

!
    !!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!

!SUBROUTINE: METROMC
!   Runs Metropolis Monte Carlo code from $MIN_TEMP to $MAX_TEMP with step size
    $STEP_TEMP
!   Each temperature is run for $STEPS iterations of the Monte Carlo code.
!   At this time there is no time given for equilibriation

SUBROUTINE METROMC
  USE input_module, ONLY : dp, PI, MC_TYPE, spin, MIN_TEMP, MAX_TEMP, STEP_TEMP,
    STEPS, energy, NUM_SPINS, STEPS_PER_OUT, EQ_STEPS
  IMPLICIT NONE
```

```
REAL( kind=dp ) :: theta, phi, sint, sinp, cost, cosp
REAL( kind=dp ) :: enew, eold, eav, etot, delta, T, new_spin_L, rand
REAL( kind=dp ), DIMENSION(1:3) :: new_Spin, B_Eff
INTEGER :: i, j, k, l, spin_curr

INTERFACE
    SUBROUTINE Get_B_Eff(B_Eff, ord, spin_Curr)
        USE input_module, ONLY : dp, n_Neighbour, invu_table, spin, J_EX
        IMPLICIT NONE
        REAL( kind=dp ), INTENT(INOUT) :: B_EFF(:)
        INTEGER, INTENT(IN) :: spin_Curr
        INTEGER, INTENT(IN) :: ord
    END SUBROUTINE Get_B_Eff
END INTERFACE

write(*,*) "    Min temp: ", MIN_TEMP
write(*,*) "    Max temp: ", MAX_TEMP
write(*,*) "    Temp steps: ", STEP_TEMP
write(*,*) "    Steps per temp: ", STEPS
write(*,*) "    Steps per output: ", STEPS_PER_OUT

do i=1,STEP_TEMP
    if(MC_TYPE.eq."HEATING") then    !Choose heating or cooling.
        T=MIN_TEMP+((MAX_TEMP-MIN_TEMP)/STEP_TEMP)*(i-1)
    else
        T=MAX_TEMP-((MAX_TEMP-MIN_TEMP)/STEP_TEMP)*(i-1)
    endif

    do j=1,EQ_STEPS !Equilibriate
        do k=1,NUM_SPINS

            B_Eff(1:3)=(/ 0.0d0, 0.0d0, 0.0d0 /)

            501 continue
            call random_number(rand)
            spin_curr= int(NUM_SPINS*rand)

            if(spin_curr.gt.NUM_SPINS.or.spin_curr.lt.1) go to 501  !   Ensure
                integer lies within lattice

            call Get_B_Eff(B_Eff, 1, spin_curr)     !Get NN Effective Field
            call Get_B_Eff(B_Eff, 2, spin_curr)     !Get NNN Effective Field

            call random_number(phi)
            call random_number(theta)
            phi=phi*2*PI
            theta=theta*PI

            cosp=cos(phi)
            sinp=sin(phi)
            cost=cos(theta)
```

```
            sint=sin(theta)

            new_spin(1:3)=(/ sint*cosp, sint*sinp, cost /)
            new_spin_L=dsqrt(new_Spin(1)**2+new_Spin(2)**2+new_Spin(3)**2)
            new_spin(1:3)=new_spin(1:3)/new_Spin_L

            enew=0.0d0
            eold=0.0d0
            do l=1,3
                eold=eold-(B_Eff(l)*spin(spin_curr,l))  !Add up the energy terms
                enew=enew-(B_Eff(l)*new_spin(l))
            enddo ! j

            delta=enew-eold

            call random_number(rand)
            if((delta.lt.0.0d0).OR.(rand.le.(dexp(-delta/T)/(1.+dexp(-delta/T))))
                ) then
                spin(spin_curr,1:3)=new_spin(1:3)
            endif
        enddo !k
    enddo !j

    do j=1,STEPS !Run Metropolis MC
        do k=1,NUM_SPINS

            B_Eff(1:3)=(/ 0.0d0, 0.0d0, 0.0d0 /)

            502 continue
            call random_number(rand)
            spin_curr= int(NUM_SPINS*rand)

            if(spin_curr.gt.NUM_SPINS.or.spin_curr.lt.1) go to 502  !   Ensure
                integer lies within lattice

            call Get_B_Eff(B_Eff, 1, spin_curr)     !Get NN Effective Field
            call Get_B_Eff(B_Eff, 2, spin_curr)     !Get NNN Effective Field

            call random_number(phi)
            call random_number(theta)
            phi=phi*2*PI
            theta=theta*PI

            cosp=cos(phi)
            sinp=sin(phi)
            cost=cos(theta)
            sint=sin(theta)

            new_spin(1:3)=(/ sint*cosp, sint*sinp, cost /)
            new_spin_L=dsqrt(new_Spin(1)**2+new_Spin(2)**2+new_Spin(3)**2)
            new_spin(1:3)=new_spin(1:3)/new_Spin_L
```

```
            enew=0.d0
            eold=0.d0
            do l=1,3
                eold=eold-(B_Eff(l)*spin(spin_curr,l))  !Add up the energy terms
                enew=enew-(B_Eff(l)*new_spin(l))
            enddo ! j

            delta=enew-eold

            call random_number(rand)
            if(rand.le.(dexp(-delta/T)/(1.+dexp(-delta/T)))) then
                spin(spin_curr,1:3)=new_spin(1:3)
            endif
        enddo !k

        call measure
        etot=etot+energy

        if(Modulo(j,STEPS_PER_OUT).eq.0) then
            write(*,*) "    TEMP: ", T, j, energy/NUM_SPINS

            call measure     ! measure the energy of the system

            open(unit=12, file="energy_timeseries.dat", position="APPEND", action
                ="WRITE", status="UNKNOWN") !Output for energy

            write(unit=12,fmt=*) T, j, energy/NUM_SPINS

            flush(unit=12)          !Clear and close output
            close(unit=12)

            do k=1,NUM_SPINS
                write(i*1000+(j/STEPS_PER_OUT),*) spin(k,1),spin(k,2),spin(k,3)
            enddo !k

            flush(i*1000+(j/STEPS_PER_OUT))    !Clear the spin orientation output
        endif
    enddo    !j

    eav=(etot/NUM_SPINS)/STEPS
    write(*,*) "    STEP: ",i, eav

    open(unit=12, file="energy_final.dat", position="APPEND", action="WRITE",
        status="UNKNOWN") !Output for energy

    write(unit=12,fmt=*) T, eav, etot

    flush(unit=12)          !Clear and close output
    close(unit=12)

    do k=1,NUM_SPINS
        write(1050,*) spin(k,1),spin(k,2),spin(k,3)
```

```
        enddo !k

        flush(1050)    !Clear the spin orientation output
    enddo !i

END SUBROUTINE METROMC

!
    !!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!

!Function: Cross(a,b)
!   Calculates the cross product of two vectors
FUNCTION Cross(a, b) result(c)
USE input_module, ONLY : dp
  REAL( kind=dp ), DIMENSION(1:3) :: c
  REAL( kind=dp ), DIMENSION(1:3), INTENT(IN) :: a, b

  c(1) = a(2) * b(3) - a(3) * b(2)
  c(2) = a(3) * b(1) - a(1) * b(3)
  c(3) = a(1) * b(2) - a(2) * b(1)
END FUNCTION Cross


!
    !!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!

!SUBROUTINE: Get_B_Eff()
!   Calculates the effective field felt by $spin_Curr as a result of all nearest
    neighbour interactions

SUBROUTINE Get_B_Eff(B_Eff, ord, spin_Curr)
    USE input_module, ONLY : dp, n_Neighbour, invu_table, spin, J_EX
    IMPLICIT NONE

    REAL( kind=dp ), INTENT(INOUT) :: B_Eff(:)
    INTEGER, DIMENSION(1:6) :: neighbour    !Store neighbours in smaller array for
        each spin

    INTEGER, INTENT(IN) :: spin_Curr    ! Current spin being considered
    INTEGER, INTENT(IN) :: ord         ! The order of nearest neighbour being
        considered
    INTEGER :: j
    INTEGER :: a    ! Holds the atom #

    do j=1,6    ! Make smaller array of neighbouring spin (For size efficiency I
        guess.)
        neighbour(j)=n_Neighbour(spin_Curr,ord,j)
    enddo ! j

    a=invu_Table(spin_Curr,4)  !Get the atom# for current spin

    if(a.eq.1) then ! All neighbours defined specific to atom # 1
```

```
!(1,2,3,4,5,6,7,8,9)->(1,2,3,4+5,4-5,6+7,6-7,8+9,8-9)

B_Eff(1)=B_Eff(1)+J_EX(ord,1)*(spin(neighbour(1),1)+spin(neighbour(2),1)) !
    Calculate effective field along x from each energy term
B_Eff(1)=B_Eff(1)+J_EX(ord,2)*(spin(neighbour(3),1)+spin(neighbour(4),1))
B_Eff(1)=B_Eff(1)+J_EX(ord,3)*(spin(neighbour(5),1)+spin(neighbour(6),1))
B_Eff(1)=B_Eff(1)+J_EX(ord,4)*(spin(neighbour(1),2)+spin(neighbour(4),3)+spin
    (neighbour(2),2)+spin(neighbour(3),3))/2.0_dp
B_Eff(1)=B_Eff(1)+J_EX(ord,5)*(spin(neighbour(1),2)+spin(neighbour(4),3)-spin
    (neighbour(2),2)-spin(neighbour(3),3))/2.0_dp
B_Eff(1)=B_Eff(1)+J_EX(ord,6)*(-spin(neighbour(6),3)+spin(neighbour(3),2)+
    spin(neighbour(5),3)-spin(neighbour(4),2))/2.0_dp
B_Eff(1)=B_Eff(1)+J_EX(ord,7)*(-spin(neighbour(6),3)+spin(neighbour(3),2)-
    spin(neighbour(5),3)+spin(neighbour(4),2))/2.0_dp
B_Eff(1)=B_Eff(1)+J_EX(ord,8)*(-spin(neighbour(2),3)+spin(neighbour(5),2)+
    spin(neighbour(1),3)-spin(neighbour(6),2))/2.0_dp
B_Eff(1)=B_Eff(1)+J_EX(ord,9)*(-spin(neighbour(2),3)+spin(neighbour(5),2)-
    spin(neighbour(1),3)+spin(neighbour(6),2))/2.0_dp

B_Eff(2)=B_Eff(2)+J_EX(ord,1)*(spin(neighbour(5),2)+spin(neighbour(6),2)) !
    Calculate effective field along y from each energy term
B_Eff(2)=B_Eff(2)+J_EX(ord,2)*(spin(neighbour(1),2)+spin(neighbour(2),2))
B_Eff(2)=B_Eff(2)+J_EX(ord,3)*(spin(neighbour(3),2)+spin(neighbour(4),2))
B_Eff(2)=B_Eff(2)+J_EX(ord,4)*(spin(neighbour(2),1)+spin(neighbour(5),3)+spin
    (neighbour(1),1)+spin(neighbour(6),3))/2.0_dp
B_Eff(2)=B_Eff(2)+J_EX(ord,5)*(spin(neighbour(2),1)+spin(neighbour(5),3)-spin
    (neighbour(1),1)-spin(neighbour(6),3))/2.0_dp
B_Eff(2)=B_Eff(2)+J_EX(ord,6)*(spin(neighbour(1),3)-spin(neighbour(4),1)-spin
    (neighbour(2),3)+spin(neighbour(3),1))/2.0_dp
B_Eff(2)=B_Eff(2)+J_EX(ord,7)*(spin(neighbour(1),3)-spin(neighbour(4),1)+spin
    (neighbour(2),3)-spin(neighbour(3),1))/2.0_dp
B_Eff(2)=B_Eff(2)+J_EX(ord,8)*(-spin(neighbour(6),1)+spin(neighbour(3),3)+
    spin(neighbour(5),1)-spin(neighbour(4),3))/2.0_dp
B_Eff(2)=B_Eff(2)+J_EX(ord,9)*(-spin(neighbour(6),1)+spin(neighbour(3),3)-
    spin(neighbour(5),1)+spin(neighbour(4),3))/2.0_dp

B_Eff(3)=B_Eff(3)+J_EX(ord,1)*(spin(neighbour(3),3)+spin(neighbour(4),3)) !
    Calculate effective field along z from each energy term
B_Eff(3)=B_Eff(3)+J_EX(ord,2)*(spin(neighbour(5),3)+spin(neighbour(6),3))
B_Eff(3)=B_Eff(3)+J_EX(ord,3)*(spin(neighbour(1),3)+spin(neighbour(2),3))
B_Eff(3)=B_Eff(3)+J_EX(ord,4)*(spin(neighbour(6),2)+spin(neighbour(3),1)+spin
    (neighbour(4),1)+spin(neighbour(5),2))/2.0_dp
B_Eff(3)=B_Eff(3)+J_EX(ord,5)*(spin(neighbour(6),2)+spin(neighbour(3),1)-spin
    (neighbour(4),1)-spin(neighbour(5),2))/2.0_dp
B_Eff(3)=B_Eff(3)+J_EX(ord,6)*(-spin(neighbour(2),2)+spin(neighbour(5),1)+
    spin(neighbour(1),2)-spin(neighbour(6),1))/2.0_dp
B_Eff(3)=B_Eff(3)+J_EX(ord,7)*(-spin(neighbour(2),2)+spin(neighbour(5),1)-
    spin(neighbour(1),2)+spin(neighbour(6),1))/2.0_dp
B_Eff(3)=B_Eff(3)+J_EX(ord,8)*(spin(neighbour(1),1)-spin(neighbour(4),2)-spin
    (neighbour(2),1)+spin(neighbour(3),2))/2.0_dp
```

```
    B_Eff(3)=B_Eff(3)+J_EX(ord,9)*(spin(neighbour(1),1)-spin(neighbour(4),2)+spin
        (neighbour(2),1)-spin(neighbour(3),2))/2.0_dp

else if(a.eq.2) then     ! All neighbours defined specific to atom # 2

    B_Eff(1)=B_Eff(1)+J_EX(ord,1)*(spin(neighbour(1),1)+spin(neighbour(2),1)) !
        Calculate effective field along x from each energy term
    B_Eff(1)=B_Eff(1)+J_EX(ord,2)*(spin(neighbour(5),1)+spin(neighbour(6),1))
    B_Eff(1)=B_Eff(1)+J_EX(ord,3)*(spin(neighbour(3),1)+spin(neighbour(4),1))
    B_Eff(1)=B_Eff(1)+J_EX(ord,4)*(spin(neighbour(2),2)-spin(neighbour(6),3)+spin
        (neighbour(1),2)-spin(neighbour(5),3))/2.0_dp
    B_Eff(1)=B_Eff(1)+J_EX(ord,5)*(spin(neighbour(2),2)-spin(neighbour(6),3)-spin
        (neighbour(1),2)+spin(neighbour(5),3))/2.0_dp
    B_Eff(1)=B_Eff(1)+J_EX(ord,6)*(spin(neighbour(3),3)+spin(neighbour(5),2)-spin
        (neighbour(4),3)-spin(neighbour(6),2))/2.0_dp
    B_Eff(1)=B_Eff(1)+J_EX(ord,7)*(spin(neighbour(3),3)+spin(neighbour(5),2)+spin
        (neighbour(4),3)+spin(neighbour(6),2))/2.0_dp
    B_Eff(1)=B_Eff(1)+J_EX(ord,8)*(spin(neighbour(1),3)+spin(neighbour(4),2)-spin
        (neighbour(2),3)-spin(neighbour(3),2))/2.0_dp
    B_Eff(1)=B_Eff(1)+J_EX(ord,9)*(spin(neighbour(1),3)+spin(neighbour(4),2)+spin
        (neighbour(2),3)+spin(neighbour(3),2))/2.0_dp

    B_Eff(2)=B_Eff(2)+J_EX(ord,1)*(spin(neighbour(3),2)+spin(neighbour(4),2)) !
        Calculate effective field along y from each energy term
    B_Eff(2)=B_Eff(2)+J_EX(ord,2)*(spin(neighbour(1),2)+spin(neighbour(2),2))
    B_Eff(2)=B_Eff(2)+J_EX(ord,3)*(spin(neighbour(5),2)+spin(neighbour(6),2))
    B_Eff(2)=B_Eff(2)+J_EX(ord,4)*(spin(neighbour(1),1)-spin(neighbour(4),3)+spin
        (neighbour(2),1)-spin(neighbour(3),3))/2.0_dp
    B_Eff(2)=B_Eff(2)+J_EX(ord,5)*(spin(neighbour(1),1)-spin(neighbour(4),3)-spin
        (neighbour(2),1)+spin(neighbour(3),3))/2.0_dp
    B_Eff(2)=B_Eff(2)+J_EX(ord,6)*(-spin(neighbour(2),3)-spin(neighbour(6),1)+
        spin(neighbour(1),3)+spin(neighbour(5),1))/2.0_dp
    B_Eff(2)=B_Eff(2)+J_EX(ord,7)*(-spin(neighbour(2),3)-spin(neighbour(6),1)-
        spin(neighbour(1),3)-spin(neighbour(5),1))/2.0_dp
    B_Eff(2)=B_Eff(2)+J_EX(ord,8)*(-spin(neighbour(3),1)-spin(neighbour(5),3)+
        spin(neighbour(4),1)+spin(neighbour(6),3))/2.0_dp
    B_Eff(2)=B_Eff(2)+J_EX(ord,9)*(-spin(neighbour(3),1)-spin(neighbour(5),3)-
        spin(neighbour(4),1)-spin(neighbour(6),3))/2.0_dp

    B_Eff(3)=B_Eff(3)+J_EX(ord,1)*(spin(neighbour(5),3)+spin(neighbour(6),3)) !
        Calculate effective field along z from each energy term
    B_Eff(3)=B_Eff(3)+J_EX(ord,2)*(spin(neighbour(3),3)+spin(neighbour(4),3))
    B_Eff(3)=B_Eff(3)+J_EX(ord,3)*(spin(neighbour(1),3)+spin(neighbour(2),3))
    B_Eff(3)=B_Eff(3)+J_EX(ord,4)*(-spin(neighbour(3),2)-spin(neighbour(5),1)-
        spin(neighbour(4),2)-spin(neighbour(6),1))/2.0_dp
    B_Eff(3)=B_Eff(3)+J_EX(ord,5)*(-spin(neighbour(3),2)-spin(neighbour(5),1)+
        spin(neighbour(4),2)+spin(neighbour(6),1))/2.0_dp
    B_Eff(3)=B_Eff(3)+J_EX(ord,6)*(spin(neighbour(1),2)-spin(neighbour(4),1)-spin
        (neighbour(2),2)+spin(neighbour(3),1))/2.0_dp
    B_Eff(3)=B_Eff(3)+J_EX(ord,7)*(spin(neighbour(1),2)-spin(neighbour(4),1)+spin
        (neighbour(2),2)-spin(neighbour(3),1))/2.0_dp
```

```
        B_Eff(3)=B_Eff(3)+J_EX(ord,8)*(-spin(neighbour(2),1)+spin(neighbour(6),2)+
            spin(neighbour(1),1)-spin(neighbour(5),2))/2.0_dp
        B_Eff(3)=B_Eff(3)+J_EX(ord,9)*(-spin(neighbour(2),1)+spin(neighbour(6),2)-
            spin(neighbour(1),1)+spin(neighbour(5),2))/2.0_dp

    else if(a.eq.3) then

        B_Eff(1)=B_Eff(1)+J_EX(ord,1)*(spin(neighbour(5),1)+spin(neighbour(6),1)) !
            Calculate effective field along x from each energy term
        B_Eff(1)=B_Eff(1)+J_EX(ord,2)*(spin(neighbour(1),1)+spin(neighbour(2),1))
        B_Eff(1)=B_Eff(1)+J_EX(ord,3)*(spin(neighbour(3),1)+spin(neighbour(4),1))
        B_Eff(1)=B_Eff(1)+J_EX(ord,4)*(-spin(neighbour(6),2)+spin(neighbour(1),3)-
            spin(neighbour(5),2)+spin(neighbour(2),3))/2.0_dp
        B_Eff(1)=B_Eff(1)+J_EX(ord,5)*(-spin(neighbour(6),2)+spin(neighbour(1),3)+
            spin(neighbour(5),2)-spin(neighbour(2),3))/2.0_dp
        B_Eff(1)=B_Eff(1)+J_EX(ord,6)*(-spin(neighbour(4),3)-spin(neighbour(2),2)+
            spin(neighbour(3),3)+spin(neighbour(1),2))/2.0_dp
        B_Eff(1)=B_Eff(1)+J_EX(ord,7)*(-spin(neighbour(4),3)-spin(neighbour(2),2)-
            spin(neighbour(3),3)-spin(neighbour(1),2))/2.0_dp
        B_Eff(1)=B_Eff(1)+J_EX(ord,8)*(-spin(neighbour(5),3)-spin(neighbour(3),2)+
            spin(neighbour(6),3)+spin(neighbour(4),2))/2.0_dp
        B_Eff(1)=B_Eff(1)+J_EX(ord,9)*(-spin(neighbour(5),3)-spin(neighbour(3),2)-
            spin(neighbour(6),3)-spin(neighbour(4),2))/2.0_dp

        B_Eff(2)=B_Eff(2)+J_EX(ord,1)*(spin(neighbour(3),2)+spin(neighbour(4),2)) !
            Calculate effective field along y from each energy term
        B_Eff(2)=B_Eff(2)+J_EX(ord,2)*(spin(neighbour(5),2)+spin(neighbour(6),2))
        B_Eff(2)=B_Eff(2)+J_EX(ord,3)*(spin(neighbour(1),2)+spin(neighbour(2),2))
        B_Eff(2)=B_Eff(2)+J_EX(ord,4)*(-spin(neighbour(5),1)-spin(neighbour(3),3)-
            spin(neighbour(6),1)-spin(neighbour(4),3))/2.0_dp
        B_Eff(2)=B_Eff(2)+J_EX(ord,5)*(-spin(neighbour(5),1)-spin(neighbour(3),3)+
            spin(neighbour(6),1)+spin(neighbour(4),3))/2.0_dp
        B_Eff(2)=B_Eff(2)+J_EX(ord,6)*(-spin(neighbour(6),3)+spin(neighbour(1),1)+
            spin(neighbour(5),3)-spin(neighbour(2),1))/2.0_dp
        B_Eff(2)=B_Eff(2)+J_EX(ord,7)*(-spin(neighbour(6),3)+spin(neighbour(1),1)-
            spin(neighbour(5),3)+spin(neighbour(2),1))/2.0_dp
        B_Eff(2)=B_Eff(2)+J_EX(ord,8)*(spin(neighbour(4),1)-spin(neighbour(2),3)-spin
            (neighbour(3),1)+spin(neighbour(1),3))/2.0_dp
        B_Eff(2)=B_Eff(2)+J_EX(ord,9)*(spin(neighbour(4),1)-spin(neighbour(2),3)+spin
            (neighbour(3),1)-spin(neighbour(1),3))/2.0_dp

        B_Eff(3)=B_Eff(3)+J_EX(ord,1)*(spin(neighbour(1),3)+spin(neighbour(2),3)) !
            Calculate effective field along z from each energy term
        B_Eff(3)=B_Eff(3)+J_EX(ord,2)*(spin(neighbour(3),3)+spin(neighbour(4),3))
        B_Eff(3)=B_Eff(3)+J_EX(ord,3)*(spin(neighbour(5),3)+spin(neighbour(6),3))
        B_Eff(3)=B_Eff(3)+J_EX(ord,4)*(-spin(neighbour(4),2)+spin(neighbour(2),1)-
            spin(neighbour(3),2)+spin(neighbour(1),1))/2.0_dp
        B_Eff(3)=B_Eff(3)+J_EX(ord,5)*(-spin(neighbour(4),2)+spin(neighbour(2),1)+
            spin(neighbour(3),2)-spin(neighbour(1),1))/2.0_dp
        B_Eff(3)=B_Eff(3)+J_EX(ord,6)*(spin(neighbour(5),2)+spin(neighbour(3),1)-spin
            (neighbour(6),2)-spin(neighbour(4),1))/2.0_dp
```

```
        B_Eff(3)=B_Eff(3)+J_EX(ord,7)*(spin(neighbour(5),2)+spin(neighbour(3),1)+spin
            (neighbour(6),2)+spin(neighbour(4),1))/2.0_dp
        B_Eff(3)=B_Eff(3)+J_EX(ord,8)*(spin(neighbour(6),1)+spin(neighbour(1),2)-spin
            (neighbour(5),1)-spin(neighbour(2),2))/2.0_dp
        B_Eff(3)=B_Eff(3)+J_EX(ord,9)*(spin(neighbour(6),1)+spin(neighbour(1),2)+spin
            (neighbour(5),1)+spin(neighbour(2),2))/2.0_dp

    else

        B_Eff(1)=B_Eff(1)+J_EX(ord,1)*(spin(neighbour(5),1)+spin(neighbour(6),1)) !
            Calculate effective field along x from each energy term
        B_Eff(1)=B_Eff(1)+J_EX(ord,2)*(spin(neighbour(3),1)+spin(neighbour(4),1))
        B_Eff(1)=B_Eff(1)+J_EX(ord,3)*(spin(neighbour(1),1)+spin(neighbour(2),1))
        B_Eff(1)=B_Eff(1)+J_EX(ord,4)*(-spin(neighbour(5),2)-spin(neighbour(3),3)-
            spin(neighbour(6),2)-spin(neighbour(4),3))/2.0_dp
        B_Eff(1)=B_Eff(1)+J_EX(ord,5)*(-spin(neighbour(5),2)-spin(neighbour(3),3)+
            spin(neighbour(6),2)+spin(neighbour(4),3))/2.0_dp
        B_Eff(1)=B_Eff(1)+J_EX(ord,6)*(spin(neighbour(1),3)-spin(neighbour(4),2)-spin
            (neighbour(2),3)+spin(neighbour(3),2))/2.0_dp
        B_Eff(1)=B_Eff(1)+J_EX(ord,7)*(spin(neighbour(1),3)-spin(neighbour(4),2)+spin
            (neighbour(2),3)-spin(neighbour(3),2))/2.0_dp
        B_Eff(1)=B_Eff(1)+J_EX(ord,8)*(spin(neighbour(6),3)-spin(neighbour(2),2)-spin
            (neighbour(5),3)+spin(neighbour(1),2))/2.0_dp
        B_Eff(1)=B_Eff(1)+J_EX(ord,9)*(spin(neighbour(6),3)-spin(neighbour(2),2)+spin
            (neighbour(5),3)-spin(neighbour(1),2))/2.0_dp

        B_Eff(2)=B_Eff(2)+J_EX(ord,1)*(spin(neighbour(1),2)+spin(neighbour(2),2)) !
            Calculate effective field along y from each energy term
        B_Eff(2)=B_Eff(2)+J_EX(ord,2)*(spin(neighbour(5),2)+spin(neighbour(6),2))
        B_Eff(2)=B_Eff(2)+J_EX(ord,3)*(spin(neighbour(3),2)+spin(neighbour(4),2))
        B_Eff(2)=B_Eff(2)+J_EX(ord,4)*(-spin(neighbour(6),1)+spin(neighbour(2),3)-
            spin(neighbour(5),1)+spin(neighbour(1),3))/2.0_dp
        B_Eff(2)=B_Eff(2)+J_EX(ord,5)*(-spin(neighbour(6),1)+spin(neighbour(2),3)+
            spin(neighbour(5),1)-spin(neighbour(1),3))/2.0_dp
        B_Eff(2)=B_Eff(2)+J_EX(ord,6)*(spin(neighbour(5),3)+spin(neighbour(3),1)-spin
            (neighbour(6),3)-spin(neighbour(4),1))/2.0_dp
        B_Eff(2)=B_Eff(2)+J_EX(ord,7)*(spin(neighbour(5),3)+spin(neighbour(3),1)+spin
            (neighbour(6),3)+spin(neighbour(4),1))/2.0_dp
        B_Eff(2)=B_Eff(2)+J_EX(ord,8)*(spin(neighbour(1),1)+spin(neighbour(4),3)-spin
            (neighbour(2),1)-spin(neighbour(3),3))/2.0_dp
        B_Eff(2)=B_Eff(2)+J_EX(ord,9)*(spin(neighbour(1),1)+spin(neighbour(4),3)+spin
            (neighbour(2),1)+spin(neighbour(3),3))/2.0_dp

        B_Eff(3)=B_Eff(3)+J_EX(ord,1)*(spin(neighbour(3),3)+spin(neighbour(4),3)) !
            Calculate effective field along z from each energy term
        B_Eff(3)=B_Eff(3)+J_EX(ord,2)*(spin(neighbour(1),3)+spin(neighbour(2),3))
        B_Eff(3)=B_Eff(3)+J_EX(ord,3)*(spin(neighbour(5),3)+spin(neighbour(6),3))
        B_Eff(3)=B_Eff(3)+J_EX(ord,4)*(spin(neighbour(1),2)-spin(neighbour(4),1)+spin
            (neighbour(2),2)-spin(neighbour(3),1))/2.0_dp
        B_Eff(3)=B_Eff(3)+J_EX(ord,5)*(spin(neighbour(1),2)-spin(neighbour(4),1)-spin
            (neighbour(2),2)+spin(neighbour(3),1))/2.0_dp
```

```
        B_Eff(3)=B_Eff(3)+J_EX(ord,6)*(-spin(neighbour(6),2)-spin(neighbour(2),1)+
            spin(neighbour(5),2)+spin(neighbour(1),1))/2.0_dp
        B_Eff(3)=B_Eff(3)+J_EX(ord,7)*(-spin(neighbour(6),2)-spin(neighbour(2),1)-
            spin(neighbour(5),2)-spin(neighbour(1),1))/2.0_dp
        B_Eff(3)=B_Eff(3)+J_EX(ord,8)*(-spin(neighbour(5),1)-spin(neighbour(3),2)+
            spin(neighbour(6),1)+spin(neighbour(4),2))/2.0_dp
        B_Eff(3)=B_Eff(3)+J_EX(ord,9)*(-spin(neighbour(5),1)-spin(neighbour(3),2)-
            spin(neighbour(6),1)-spin(neighbour(4),2))/2.0_dp

    endif

    ! Same Site Anisotropy terms
    !!  These terms would be the same for all spins
    B_Eff(1)=B_Eff(1)+J_EX(ord,10)*(spin(spin_Curr,1))                    ! 2nd
         Order CEF terms (Same cite invariants)
    B_Eff(1)=B_Eff(1)+J_EX(ord,11)*(spin(spin_Curr,2)+spin(spin_Curr,3))  !!
    B_Eff(1)=B_Eff(1)+J_EX(ord,12)*(spin(spin_Curr,1)**3)                 ! 4th
         Order CEF terms
    B_Eff(1)=B_Eff(1)+J_EX(ord,13)*(spin(spin_Curr,1)**3 + 2*spin(spin_Curr,1)*spin(
        spin_Curr,2)**2 &    !!
        + 2*spin(spin_Curr,1)*spin(spin_Curr,3)**2)                      !!!

    B_Eff(2)=B_Eff(2)+J_EX(ord,10)*(spin(spin_Curr,2))                   !CEF
        terms are same site invariants
    B_Eff(2)=B_Eff(2)+J_EX(ord,11)*(spin(spin_Curr,1)+spin(spin_Curr,3))  !
    B_Eff(2)=B_Eff(2)+J_EX(ord,12)*(spin(spin_Curr,2)**3)                ! 4th
         Order CEF terms
    B_Eff(2)=B_Eff(2)+J_EX(ord,13)*(spin(spin_Curr,2)**3 + 2*spin(spin_Curr,2)*spin(
        spin_Curr,1)**2 &    !!
        + 2*spin(spin_Curr,2)*spin(spin_Curr,3)**2)

    B_Eff(3)=B_Eff(3)+J_EX(ord,10)*(spin(spin_Curr,3))                   !CEF
        terms are same site invariants
    B_Eff(3)=B_Eff(3)+J_EX(ord,11)*(spin(spin_Curr,1)+spin(spin_Curr,2))  !
    B_Eff(3)=B_Eff(3)+J_EX(ord,12)*(spin(spin_Curr,3)**3)                ! 4th
         Order CEF terms
    B_Eff(3)=B_Eff(3)+J_EX(ord,13)*(spin(spin_Curr,3)**3 + 2*spin(spin_Curr,3)*spin(
        spin_Curr,2)**2 &    !!
        + 2*spin(spin_Curr,3)*spin(spin_Curr,1)**2)

    ! if(ord.eq.1) then
    !     open(unit=17, file="BEFF.dat", position="APPEND", action="WRITE", status="
        UNKNOWN")
    !     write(unit=17,fmt=*) ord, B_Eff(1), B_Eff(2), B_Eff(3)
    !     flush(unit=17)
    !     close(unit=17)
    ! endif

END SUBROUTINE Get_B_Eff
```

```
!
    !!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!

!SUBROUTINE: init_random_seed()
!   Initializes the seed for the built-in RNG

SUBROUTINE init_random_seed()
    IMPLICIT NONE
        Integer :: i, n, clock
        Integer, Dimension(:), Allocatable :: seed

        Call random_seed(size = n)
        Allocate(seed(n))

        Call system_clock(Count=clock)

        seed = clock + 37 * (/ (i - 1, i = 1, n) /)
        Call random_seed(PUT = seed)

        Deallocate(seed)
END SUBROUTINE

!
    !!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!

!SUBROUTINE: Shuffle(arr)
!   Takes an array, $arr, and shuffles it's elements according to the Knuth shuffle

SUBROUTINE Shuffle(arr)

    !Subroutine variables declaration!!!!!!!!!!!!!!

      INTEGER, INTENT(INOUT) :: arr(:)     ! Array to be shuffled, taken in and
          returned

      INTEGER :: i, rand_Pos, temp    ! Utility INTEGERs

      DOUBLE PRECISION :: rand   ! Random number

    !!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!

      do i = size(arr), 2, -1
        call random_number(rand)
        rand_Pos = int(rand * i) + 1
        temp = arr(rand_Pos)
        arr(rand_Pos) = arr(i)
        arr(i) = temp
      end do

END SUBROUTINE Shuffle
```

```
!
   !!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!

!SUBROUTINE: Measure
!   Measure the energy of the current configuration

SUBROUTINE measure
   USE input_module, ONLY : dp, NUM_SPINS, spin, energy!, mag, chi
   IMPLICIT NONE

   REAL( kind=dp ), DIMENSION(1:3) :: B_Eff   ! The effective field in each
      dimension
   INTEGER :: i,j ! Subroutine iterator

   INTERFACE
      SUBROUTINE Get_B_Eff(B_Eff, ord, spin_Curr)
         USE input_module, ONLY : dp, n_Neighbour, invu_table, spin, J_EX
         IMPLICIT NONE
         REAL( kind=dp ), INTENT(INOUT) :: B_EFF(:)
         INTEGER, INTENT(IN) :: spin_Curr
         INTEGER, INTENT(IN) :: ord
      END SUBROUTINE Get_B_Eff
   END INTERFACE

   energy=0.d0

   do i=1,NUM_SPINS     !Iteratively find the effective field of each spin (Should be
       done some other way)
      B_Eff(1:3)=(/ 0.0d0, 0.0d0, 0.0d0 /)

      call Get_B_Eff(B_Eff, 1, i)     !Get NN Effective Field
      call Get_B_Eff(B_Eff, 2, i)     !Get NNN Effective Field

      do j=1,3
         energy=energy-(B_Eff(j)*spin(i,j))  !Add up the energy terms
      enddo ! j
   enddo !i

   energy=energy/2   ! Remove double counting

END SUBROUTINE measure

!
   !!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!

!SUBROUTINE: FFT
!   Perform Fast Fourier Transform on spins belonging to sublattice $spin_num
!   considering the direction $coord_num

SUBROUTINE FFT(spin_num, coord_num)
   USE input_module, ONLY : dp, CELLS, spin, u_Table
   use, intrinsic :: iso_c_binding
```

```
    include 'fftw3.f03'

    INTEGER, INTENT(IN) :: spin_num, coord_num

    TYPE(C_PTR) :: plan
    REAL(C_DOUBLE), DIMENSION(CELLS,CELLS,CELLS) :: in
    COMPLEX(C_DOUBLE_COMPLEX), DIMENSION(CELLS,CELLS,CELLS) :: out
    REAL :: tmp

    INTEGER :: i,j,k
    CHARACTER*10 :: FILE_OUT

    plan = fftw_plan_dft_r2c_3d(CELLS,CELLS,CELLS,in,out,FFTW_ESTIMATE)

    do i=0,CELLS-1
        do j=0,CELLS-1
            do k=0,CELLS-1
                in(i+1,j+1,k+1)=spin(u_Table(i,j,k,spin_num),coord_num) !Mismatch
                    of numbers
            enddo !k
        enddo !j
    enddo !i

    call fftw_execute_dft_r2c(plan, in, out)

    write(FILE_OUT,'(A,I1,A,I1,A)') "Q_S",spin_num,"C",coord_num,".dat"
    open(unit=14, file=FILE_OUT, action="WRITE", status="REPLACE")

    do i=1,CELLS
        do j=1,CELLS
            do k=1,CELLS
                write(unit=14,fmt=*) i-1,j-1,k-1,real(out(i,j,k)),aimag(out(i,j,k)) !
                    Mismatch of numbers
            enddo
        enddo
    enddo

    flush(14)
    close(14)

    call fftw_destroy_plan(plan)

END SUBROUTINE FFT
```

Appendix E

Analysis Code

E.1 Python Code

This appendix contains the computer code used in the analysis of the simulation data simulations in two parts: The analysis of individual data (Analyze.py), and the combination of analyses of different data (Combine.py). This code was prepared using Python 3.

E.1.1 Analyze

```
# Kyle Hall 2020

import numpy as np
from scipy.stats import chi2
from scipy.stats import vonmises
from scipy.stats import norm
from scipy.stats import circmean, circvar
import os
import sys

# Check for command line arguments. If none present, ask for arguments
if not sys.stdin.isatty():
    direc=input("Directory Name: ")
    confRange=int(input("Input number of minimum configurations: "))
else:
    direc=sys.argv[1]
    confRange=int(sys.argv[2])

#DATA FUNCTION DEFINITION###############################################

def ReadFile(direc, file):
    '''

    '''
    input_file = direc+"/"+file
    data = np.genfromtxt(input_file)
    return data

def RawToUC(rawData,direc,fileNum):
    '''

    '''
    ucCount = round(len(rawData)/4)
    side = round(ucCount**(1.0/3.0))
    ucData = np.zeros((side,side,side,4,3))
    count=0
    for i in range(side):
        for j in range(side):
            for k in range(side):
                for a in range(4):
                    ucData[i,j,k,a,:] = rawData[count,:]
                    count+=1
    out_direc = direc+"/MinUC"
    if not os.path.exists(out_direc):
        os.makedirs(out_direc)
    out_file = out_direc + "/MinUC_" + fileNum
    out_data = np.empty([7])
    for i in range(side):
        for j in range(side):
            for k in range(side):
```

```
                for a in range(4):
                    out_data=np.vstack((out_data,[i,j,k,a,ucData[i,j,k,a,0],ucData[i,
                        j,k,a,1],ucData[i,j,k,a,2]]))
    out_data = out_data[1::]
    form = ['%1d','%1d','%1d','%1d','%20.19f','%20.19f','%20.19f']
    np.savetxt(out_file,out_data,fmt=form,delimiter='  ',newline='\n')

    return ucData

# PRIMARY FUNCTION DEFINITION ##############################################

def CalcGamma(ucData,direc,fileNum,
                n=np.array([1,1,1]),
                r=np.array([-1,1,0]),
                y=np.array([-1,-1,2])):
    '''

    '''
    side = len(ucData)
    n = n/np.linalg.norm(n)
    r = r/np.linalg.norm(r)
    y = y/np.linalg.norm(y)

    gData = np.zeros((side,side,side,4))

    for i in range(side):
        for j in range(side):
            for k in range(side):
                for a in range(4):
                    ndot = np.dot(ucData[i,j,k,a], n)
                    rdot = np.dot(ucData[i,j,k,a], r)
                    ydot = np.dot(ucData[i,j,k,a], y)
                    ndot = ndot/np.abs(ndot) if (np.abs(ndot)>1.0) else ndot #Clip to
                        [-1.0,1.0]
                    rdot = rdot/np.abs(rdot) if (np.abs(rdot)>1.0) else rdot
                    ydot = ydot/np.abs(ydot) if (np.abs(ydot)>1.0) else ydot

                    arcgamma = ndot / np.sqrt(ndot**2 + rdot**2 + ydot**2)
                    gamma = np.pi/2 - np.arccos(arcgamma) #Check this
                    gData[i,j,k,a] = gamma

    out_direc = direc+"/Gamma/UC"
    if not os.path.exists(out_direc):
        os.makedirs(out_direc)
    out_file = out_direc + "/GammaUC_" + fileNum
    out_data = np.empty([5])
    for i in range(side):
        for j in range(side):
            for k in range(side):
                for a in range(4):
                    out_data=np.vstack((out_data,[i,j,k,a,gData[i,j,k,a]]))
    out_data = out_data[1::]
```

```
    form = ['%1d','%1d','%1d','%1d','%20.19f']
    np.savetxt(out_file,out_data,fmt=form,delimiter=' ',newline='\n')

    return gData

def CalcTheta(ucData,direc,fileNum,
              r=np.array([-1,1,0]),
              n=np.array([1,1,1]),
              y=np.array([-1,-1,2])):
    '''

    '''
    side = len(ucData)
    r = r/np.linalg.norm(r)
    n = n/np.linalg.norm(n)
    y = y/np.linalg.norm(y)

    tData = np.zeros((side,side,side,4))

    for i in range(side):
        for j in range(side):
            for k in range(side):
                for a in range(4):
                    ndot = np.dot(ucData[i,j,k,a], n)
                    rdot = np.dot(ucData[i,j,k,a], r)
                    ydot = np.dot(ucData[i,j,k,a], y)
                    ndot = ndot/np.abs(ndot) if (np.abs(ndot)>1.0) else ndot #Clip to
                        [-1.0,1.0]
                    rdot = rdot/np.abs(rdot) if (np.abs(rdot)>1.0) else rdot
                    ydot = ydot/np.abs(ydot) if (np.abs(ydot)>1.0) else ydot

                    arctheta = ydot/rdot
                    theta = np.arctan(arctheta)

                    theta = theta + np.pi*theta/abs(theta) if (rdot < 0) else theta

                    tData[i,j,k,a] = theta + 2*np.pi if (theta < 0.0) else theta

    out_direc = direc+"/Theta/UC"
    if not os.path.exists(out_direc):
        os.makedirs(out_direc)
    out_file = out_direc + "/ThetaUC_" + fileNum
    out_data = np.empty([5])
    for i in range(side):
        for j in range(side):
            for k in range(side):
                for a in range(4):
                    out_data=np.vstack((out_data,[i,j,k,a,tData[i,j,k,a]]))
    out_data = out_data[1::]
    form = ['%1d','%1d','%1d','%1d','%20.19f']
    np.savetxt(out_file,out_data,fmt=form,delimiter=' ',newline='\n')
```

```
    return tData

def CalcOP(kMean,ucData, direc, fileNum):
    '''

    '''
    side = len(ucData)

    ftSum = np.zeros((4,3),dtype=np.complex_)

    for i in range(side):
        for j in range(side):
            for k in range(side):
                l = i + j + k
                for a in range(4):
                    for c in range(3):
                        ftSum[a,c] += np.exp(-1j * kMean * l) * ucData[i,j,k,a,c]

    # for a in range(4):
    #     for c in range(3):
    #         print(str(ftSum[a,c]))

    F1r = np.zeros((4),dtype=np.complex_)   #Rectangular form arrays
    F2r = np.zeros((4),dtype=np.complex_)
    F3r = np.zeros((4),dtype=np.complex_)
    F1 = np.zeros((4,2))                    #Exponential form arrays
    F2 = np.zeros((4,2))
    F3 = np.zeros((4,2))
    eps = np.exp( -1j * 2* np.pi/3 )

    F1r[0] = np.sum(ftSum[0])     # S1x + S1y + S1z
    F1r[1] = ftSum[1,0] + ftSum[2,2] + ftSum[3,1]     # S2x + S3z + S4y
    F1r[2] = ftSum[1,1] + ftSum[2,0] + ftSum[3,2]     # S2y + S3x + S4z
    F1r[3] = ftSum[1,2] + ftSum[2,1] + ftSum[3,0]     # S2z + S3y + S4x

    F2r[0] = ftSum[0,0] + eps*ftSum[0,1] + (eps**2)*ftSum[0,2]    # S1x + e S1y + e2
        S1z
    F2r[1] = ftSum[1,0] + eps*ftSum[3,1] + (eps**2)*ftSum[2,2]    # S2x + e2 S3z + e
        S4y
    F2r[2] = ftSum[1,1] + eps*ftSum[3,2] + (eps**2)*ftSum[2,0]    # S2y + e2 S3x + e
        S4z
    F2r[3] = ftSum[1,2] + eps*ftSum[3,0] + (eps**2)*ftSum[2,1]    # S2z + e2 S3y + e
        S4x

    F3r[0] = ftSum[0,0] + (eps**2)*ftSum[0,1] + eps*ftSum[0,2]    # S1x + e2 S1y + e
        S1z
    F3r[1] = ftSum[1,0] + (eps**2)*ftSum[3,1] + eps*ftSum[2,2]    # S2x + e S3z + e2
        S4y
    F3r[2] = ftSum[1,1] + (eps**2)*ftSum[3,2] + eps*ftSum[2,0]    # S2y + e S3x + e2
        S4z
    F3r[3] = ftSum[1,2] + (eps**2)*ftSum[3,0] + eps*ftSum[2,1]    # S2z + e S3y + e2
        S4x
```

```
    F1[:,0] = np.abs(F1r)   #Convert to exponential form. F[x,0] is magnitude, F[x,1]
        is phase.
    F2[:,0] = np.abs(F2r)
    F3[:,0] = np.abs(F3r)
    F1[:,1] = np.angle(F1r)
    F2[:,1] = np.angle(F2r)
    F3[:,1] = np.angle(F3r)

    F1max = np.max(F1[:,0])
    F2max = np.max(F2[:,0])
    F3max = np.max(F3[:,0])
    Fmax = np.max(np.array((F1max, F2max, F3max)))

    # F1[:,0] /= Fmax
    # F2[:,0] /= Fmax
    # F3[:,0] /= Fmax

    out_direc = direc+"/OP"
    if not os.path.exists(out_direc):
        os.makedirs(out_direc)
    VarOP = [F1,F2,F3]
    for i in range(3):
        out_file = out_direc + "/F" + str(i+1) + "_" + fileNum
        out_data = np.empty([3])
        for j in range(4):
            out_data=np.vstack((out_data,[j+1,VarOP[i][j,0],VarOP[i][j,1]]))
        out_data = out_data[1::]
        form = ['%1d','%20.19f','%20.19f']
        np.savetxt(out_file,out_data,fmt=form,delimiter='  ',newline='\n')

    return None

# LAYER AND ION FUNCTION DEFINITION ########################################

def CalcLayerGamma(gData,direc,fileNum):
    '''

    '''


    side=len(gData)

    gLayer = [[[],[]] for i in range(3*side-2)]
    gLMean = np.zeros((3*side-2,2))
    gLAbsMean = np.zeros((3*side-2,2))
    gLVar = np.zeros((3*side-2,2))
    gLCon = np.zeros((3*side-2,2))
    gLAbsVar = np.zeros((3*side-2,2))
    gLAbsCon = np.zeros((3*side-2,2))
    lSize = np.zeros((3*side-2,2))
    tSize = np.zeros((2))
    gMean = np.zeros((2))
```

```
gCon = np.zeros((2))
gAbsMean = np.zeros((2))
gAbsCon = np.zeros((2))
gAbsVar = np.zeros((2))

cSum = np.zeros((3*side-2,2)) # Sum of cosine values
sSum = np.zeros((3*side-2,2)) # Sum of sine values
c2Sum = np.zeros((3*side-2,2)) # Sum of cosine values
s2Sum = np.zeros((3*side-2,2)) # Sum of sine values

cSumAbs = np.zeros((3*side-2,2)) # Sum of absolute cosine values
sSumAbs = np.zeros((3*side-2,2)) # Sum of absolute sine values
c2SumAbs = np.zeros((3*side-2,2)) # Sum of absolute cosine values
s2SumAbs = np.zeros((3*side-2,2)) # Sum of absolute sine values

def ConfInterval(alpha, vm, r, lSize, t, c2Sum, s2Sum):
    if vm==0:
        npercent = norm.ppf(1.-alpha/2.)
        H = (1./lSize)*(np.cos(2*t)*c2Sum + np.sin(2*t)*s2Sum)
        H = 1.0 if (H>1.0) else H # Clip H to range [0,1]
        sigma = np.sqrt((lSize*(1.-H))/(4*r**2))
        conf = np.arcsin(npercent*sigma)
        if np.isnan(conf):
            print(lSize,r,c2Sum,s2Sum,t,sigma,H)
    elif r < (2./3.):
        cpercent = chi2.ppf(1.-alpha)
        temp = np.sqrt((2*lSize*(2*r**2 - lSize*cpercent))/((4*lSize - cpercent)*
            r**2))
        temp /= r
        conf = np.arccos(temp)
    elif r >= (2./3.):
        cpercent = chi2.ppf(1.-alpha)
        temp = np.sqrt(lSize**2 - (lSize**2 - r**2)*np.exp(cpercent/lSize))
        temp /= r
        conf = np.arccos(temp)

    return conf

for i in range(side):
    for j in range(side):
        for k in range(side):
            l = i+j+k
            gLayer[l][0].append(gData[i,j,k,0])
            gLayer[l][1].append(gData[i,j,k,1])
            gLayer[l][1].append(gData[i,j,k,2])
            gLayer[l][1].append(gData[i,j,k,3])
            lSize[l,0] += 1
            lSize[l,1] += 3

            cSum[l,0] += np.cos(gData[i,j,k,0])
            sSum[l,0] += np.sin(gData[i,j,k,0])
            c2Sum[l,0] += np.cos(2*gData[i,j,k,0])
```

```
                s2Sum[l,0] += np.sin(2*gData[i,j,k,0])

                cSumAbs[l,0] += np.cos(abs(gData[i,j,k,0]))
                sSumAbs[l,0] += np.sin(abs(gData[i,j,k,0]))
                c2SumAbs[l,0] += np.cos(2*abs(gData[i,j,k,0]))
                s2SumAbs[l,0] += np.sin(2*abs(gData[i,j,k,0]))

                for a in range(1,4):
                    cSum[l,1] += np.cos(tData[i,j,k,a])
                    sSum[l,1] += np.sin(tData[i,j,k,a])
                    c2Sum[l,1] += np.cos(2*tData[i,j,k,a])
                    s2Sum[l,1] += np.sin(2*tData[i,j,k,a])

                    cSumAbs[l,1] += np.cos(abs(gData[i,j,k,a]))
                    sSumAbs[l,1] += np.sin(abs(gData[i,j,k,a]))
                    c2SumAbs[l,1] += np.cos(2*abs(gData[i,j,k,a]))
                    s2SumAbs[l,1] += np.sin(2*abs(gData[i,j,k,a]))

    for l in range(3*side-2):
        for i in range(2):
            # Relative values
            cSum[l,i] /= lSize[l,i]
            sSum[l,i] /= lSize[l,i]
            rbar = np.sqrt((cSum[l,i])**2 + (sSum[l,i])**2) # Divide by size inside
                or outside?
            r = lSize[l,i]*rbar
            gLMean[l,i] = np.arctan2(sSumAbs[l,i],cSumAbs[l,i])
            gLVar[l,i] = 2*(1 - rbar)
            gLCon[l,i] = ConfInterval(0.05,0,r,lSize[l,i],gLMean[l,i],c2Sum[l,i],
                s2Sum[l,i])

            # Absolute values
            cSumAbs[l,i] /= lSize[l,i]
            sSumAbs[l,i] /= lSize[l,i]
            rbar = np.sqrt((cSumAbs[l,i])**2 + (sSumAbs[l,i])**2) # Divide by size
                inside or outside?
            r = lSize[l,i]*rbar
            gLAbsMean[l,i] = np.arctan2(sSumAbs[l,i],cSumAbs[l,i])
            gLAbsVar[l,i] = 2*(1 - rbar)
            gLAbsCon[l,i] = ConfInterval(0.05,0,r,lSize[l,i],gLAbsMean[l,i],c2SumAbs[
                l,i],s2SumAbs[l,i])

    for j in range(2):      # This calculation does not valvulate confidence
        intervals or use appropriate directional statistics
        for l in range(3*side-2):
            gMean[j] += lSize[l,j]*gLMean[l,j]
            gCon[j] += (lSize[l,j]**2)*gLCon[l,j]
            gAbsMean[j] += lSize[l,j]*gLAbsMean[l,j]
            gAbsCon[j] += (lSize[l,j]**2)*gLAbsCon[l,j]
        tSize[j] = sum(lSize[:,j])
        gMean[j] /= tSize[j]
        gCon[j] /= tSize[j]**2
```

```
            gAbsMean[j] /= tSize[j]
            gAbsCon[j] /= tSize[j]**2

        out_direc = direc+"/Gamma/Layer"
        if not os.path.exists(out_direc):
            os.makedirs(out_direc)
        out_file = out_direc + "/GammaL_" + fileNum
        out_data = np.empty([7])
        for i in range(2):
            out_data=np.vstack((out_data,[-1,i,tSize[i],gMean[i],gCon[i],gAbsMean[i],
                gAbsCon[i]]))
        for l in range(3*side-2):
            for i in range(2):
                out_data=np.vstack((out_data,[l,i,lSize[l,i],gLMean[l,i],gLCon[l,i],
                    gLAbsMean[l,i],gLAbsCon[l,i]]))
        out_data = out_data[1::]
        form = ['%2d','%2d','%5d','%20.19f','%20.19f','%20.19f','%20.19f']
        np.savetxt(out_file,out_data,fmt=form,delimiter='  ',newline='\n')

        return gLMean, gLCon, lSize, gLayer

    def CalcLayerTheta(tData,lSize,direc,fileNum):
        '''

        '''

        side=len(tData)

        tLayer = [[[],[]] for i in range(3*side-2)]
        tLMean = np.zeros((3*side-2,2))
        tLVar = np.zeros((3*side-2,2))
        tLCon = np.zeros((3*side-2,2))

        cSum = np.zeros((3*side-2,2)) # Sum of cosine values
        sSum = np.zeros((3*side-2,2)) # Sum of sine values
        c2Sum = np.zeros((3*side-2,2)) # Sum of cosine values
        s2Sum = np.zeros((3*side-2,2)) # Sum of sine values

        def ConfInterval(alpha, vm, r, lSize, t, c2Sum, s2Sum):
            if vm==0:
                npercent = norm.ppf(1.-alpha/2.)
                H = (1./lSize)*(np.cos(2*t)*c2Sum + np.sin(2*t)*s2Sum)
                H = 1.0 if (H>1.0) else H # Clip H to range [0,1]
                sigma = np.sqrt((lSize*(1.-H))/(4*r**2))
                conf = np.arcsin(npercent*sigma)
                if np.isnan(conf):
                    print(lSize,r,c2Sum,s2Sum,t,sigma,H)
            elif r < (2./3.):
                cpercent = chi2.ppf(1.-alpha)
                temp = np.sqrt((2*lSize*(2*r**2 - lSize*cpercent))/((4*lSize - cpercent)*
                    r**2))
                temp /= r
```

```
        conf = np.arccos(temp)
    elif r >= (2./3.):
        cpercent = chi2.ppf(1.-alpha)
        temp = np.sqrt(lSize**2 - (lSize**2 - r**2)*np.exp(cpercent/lSize))
        temp /= r
        conf = np.arccos(temp)

    return conf

for i in range(side):
    for j in range(side):
        for k in range(side):
            l = i+j+k
            tLayer[l][0].append(tData[i,j,k,0])
            tLayer[l][1].append(tData[i,j,k,1])
            tLayer[l][1].append(tData[i,j,k,2])
            tLayer[l][1].append(tData[i,j,k,3])
            cSum[l,0] += np.cos(tData[i,j,k,0])
            sSum[l,0] += np.sin(tData[i,j,k,0])
            c2Sum[l,0] += np.cos(2*tData[i,j,k,0])
            s2Sum[l,0] += np.sin(2*tData[i,j,k,0])
            for a in range(1,4):
                cSum[l,1] += np.cos(tData[i,j,k,a])
                sSum[l,1] += np.sin(tData[i,j,k,a])
                c2Sum[l,1] += np.cos(2*tData[i,j,k,a])
                s2Sum[l,1] += np.sin(2*tData[i,j,k,a])


for l in range(3*side-2):
    for i in range(2):
        rbar = np.sqrt((cSum[l,i]/lSize[l,i])**2 + (sSum[l,i]/lSize[l,i])**2) #
            Divide by size inside or outside?
        r = lSize[l,i]*rbar
        r2 = np.sqrt((c2Sum[l,i]/lSize[l,i])**2 + (s2Sum[l,i]/lSize[l,i])**2)
        cSum[l,i] /= lSize[l,i]
        sSum[l,i] /= lSize[l,i]
        tLMean[l,i] = np.arctan2(sSum[l,i],cSum[l,i])
        tLMean[l,i] = tLMean[l,i] + 2*np.pi if (tLMean[l,i] < 0.0) else tLMean[l,
            i]
        tLVar[l,i] = 2*(1 - rbar)
        tLCon[l,i] = ConfInterval(0.05,0,r,lSize[l,i],tLMean[l,i],c2Sum[l,i],
            s2Sum[l,i])


out_direc = direc+"/Theta/Layer"
if not os.path.exists(out_direc):
    os.makedirs(out_direc)
out_file = out_direc + "/ThetaL_" + fileNum
out_data = np.empty([6])
for l in range(3*side-2):
    for i in range(2):
```

```
            out_data=np.vstack((out_data,[l,i,lSize[l,i],tLMean[l,i],tLVar[l,i],tLCon
                [l,i]]))
    out_data = out_data[1::]
    form = ['%1d','%1d','%1d','%20.19f','%20.19f','%20.19f']
    np.savetxt(out_file,out_data,fmt=form,delimiter='  ',newline='\n')

    return tLMean, tLVar, tLCon, tLayer

def CalcLayerMag(ucData,lSize,direc,fileNum):
    '''

    '''
    side=len(ucData)
    lMag = np.zeros((3*side-2,2))
    lSum = np.zeros((3*side-2,2,3))
    magMean = np.zeros((2))
    magVar = np.zeros((2))
    tSize = np.zeros((2))

    for i in range(side):
        for j in range(side):
            for k in range(side):
                l=i+j+k
                lSum[l,0] += ucData[i,j,k,0]
                lSum[l,1] += ucData[i,j,k,1] + ucData[i,j,k,2] + ucData[i,j,k,3]

    for l in range(3*side-2):
        lMag[l,0] = np.linalg.norm(lSum[l,0])/lSize[l,0]
        lMag[l,1] = np.linalg.norm(lSum[l,1])/lSize[l,1]

    for j in range(2):
        for l in range(3*side-2):
            magMean[j] += lSize[l,j]*lMag[l,j]
        tSize[j] = sum(lSize[:,j])
        magMean[j] /= tSize[j]
        magVar[j] = np.var(np.vstack(lMag[:,j]))

    out_direc = direc+"/Mag/Layer"
    if not os.path.exists(out_direc):
        os.makedirs(out_direc)
    out_file = out_direc + "/MagL_" + fileNum
    out_data = np.empty([5])
    for i in range(2):
        out_data=np.vstack((out_data,[-1,i,tSize[i],magMean[i],magVar[i]]))
    for l in range(3*side-2):
        for i in range(2):
            out_data=np.vstack((out_data,[l,i,lSize[l,i],lMag[l,i],0]))
    out_data = out_data[1::]
    form = ['%2d','%1d','%5d','%20.19f','%20.19f']
    np.savetxt(out_file,out_data,fmt=form,delimiter='  ',newline='\n')

    return lMag
```

```
def CalcIonGamma(gData,direc,fileNum):
    '''

    '''
    side=len(gData)

    gIon = np.zeros((4,side**3))
    gIMean = np.zeros((4))
    gIAbsMean = np.zeros((4))
    gIVar = np.zeros((4))
    gIAbsVar = np.zeros((4))
    gICon = np.zeros((4))
    gIAbsCon = np.zeros((4))

    cSum = np.zeros((4)) # Sum of cosine values
    sSum = np.zeros((4)) # Sum of sine values
    c2Sum = np.zeros((4)) # Sum of cosine values
    s2Sum = np.zeros((4)) # Sum of sine values

    cSumAbs = np.zeros((4)) # Sum of absolute cosine values
    sSumAbs = np.zeros((4)) # Sum of absolute sine values
    c2SumAbs = np.zeros((4)) # Sum of absolute cosine values
    s2SumAbs = np.zeros((4)) # Sum of absolute sine values

    def ConfInterval(alpha, vm, r, lSize, t, c2Sum, s2Sum):
        if vm==0:
            npercent = norm.ppf(1.-alpha/2.)
            H = (1./lSize)*(np.cos(2*t)*c2Sum + np.sin(2*t)*s2Sum)
            H = 1.0 if (H>1.0) else H # Clip H to range [0,1]
            sigma = np.sqrt((lSize*(1.-H))/(4*r**2))
            conf = np.arcsin(npercent*sigma)
            if np.isnan(conf):
                print(lSize,r,c2Sum,s2Sum,t,sigma,H)
        elif r < (2./3.):
            cpercent = chi2.ppf(1.-alpha)
            temp = np.sqrt((2*lSize*(2*r**2 - lSize*cpercent))/((4*lSize - cpercent)*
                r**2))
            temp /= r
            conf = np.arccos(temp)
        elif r >= (2./3.):
            cpercent = chi2.ppf(1.-alpha)
            temp = np.sqrt(lSize**2 - (lSize**2 - r**2)*np.exp(cpercent/lSize))
            temp /= r
            conf = np.arccos(temp)

        return conf

    count=0
    for i in range(side):
        for j in range(side):
            for k in range(side):
```

```
                for a in range(4):
                    gIon[a,count] = gData[i,j,k,a]
                    cSum[a] += np.cos(gData[i,j,k,a])
                    sSum[a] += np.sin(gData[i,j,k,a])
                    c2Sum[a] += np.cos(2*gData[i,j,k,a])
                    s2Sum[a] += np.sin(2*gData[i,j,k,a])

                    cSumAbs[a] += np.cos(abs(gData[i,j,k,a]))
                    sSumAbs[a] += np.sin(abs(gData[i,j,k,a]))
                    c2SumAbs[a] += np.cos(2*abs(gData[i,j,k,a]))
                    s2SumAbs[a] += np.sin(2*abs(gData[i,j,k,a]))

                count+=1

    for a in range(4):
        # Relative values
        cSum[a] /= count
        sSum[a] /= count
        rbar = np.sqrt((cSum[a])**2 + (sSum[a])**2) # Divide by size inside or
            outside?
        r = count*rbar
        gIMean[a] = np.arctan2(sSumAbs[a],cSumAbs[a])
        gIVar[a] = 2*(1 - rbar)
        gICon[a] = ConfInterval(0.05,0,r,count,gIMean[a],c2Sum[a],s2Sum[a])

        # Absolute values
        cSumAbs[a] /= count
        sSumAbs[a] /= count
        rbar = np.sqrt((cSumAbs[a])**2 + (sSumAbs[a])**2) # Divide by size inside or
            outside?
        r = count*rbar
        gIAbsMean[a] = np.arctan2(sSumAbs[a],cSumAbs[a])
        gIAbsVar[a] = 2*(1 - rbar)
        gIAbsCon[a] = ConfInterval(0.05,0,r,count,gIAbsMean[a],c2SumAbs[a],s2SumAbs[a
            ])

    out_direc = direc+"/Gamma/Ion"
    if not os.path.exists(out_direc):
        os.makedirs(out_direc)
    out_file = out_direc + "/GammaI_" + fileNum
    out_data = np.empty([5])
    for i in range(4):
        out_data=np.vstack((out_data,[i,gIMean[i],gICon[i],gIAbsMean[i],gIAbsCon[i]])
            )
    out_data = out_data[1::]
    form = ['%1d','%20.19f','%20.19f','%20.19f','%20.19f']
    np.savetxt(out_file,out_data,fmt=form,delimiter='  ',newline='\n')

    return gIMean, gICon, gIon

# SECONDARY FUNCTION DEFINITION ##################################
```

```
def CalcAlpha(tLMean, tLCon, lSize, direc, fileNum):
    '''

    '''

    lNum = len(tLMean)

    aLData = np.zeros((lNum))
    aMean = 0.0
    aLCon = np.zeros((lNum))
    aCon = 0.0
    tSize = 0

    for l in range(lNum):
        pSize = (lSize[l,1] + lSize[l,0])
        aLData[l] = tLMean[l,1] - tLMean[l,0] if (abs(tLMean[l,1] - tLMean[l,0]) < np
            .pi) else (tLMean[l,1] - tLMean[l,0]) - (((tLMean[l,1] - tLMean[l,0]))/(
            abs((tLMean[l,1] - tLMean[l,0]))))*2*np.pi
        aLCon[l] = lSize[l,1]*tLCon[l,1] + lSize[l,0]*tLCon[l,0]
        aLCon[l] /= pSize
        if (l != 0):
            aMean += pSize*(aLData[l])
            aCon += pSize*(aLCon[l])
            tSize += pSize

    aMean /= tSize
    aCon /= tSize

    out_direc = direc+"/Alpha"
    if not os.path.exists(out_direc):
        os.makedirs(out_direc)
    out_file = out_direc + "/Alpha_" + fileNum
    out_data = np.empty([3])
    out_data=np.vstack((out_data,[-1,aMean,aCon]))
    for i in range(1,lNum):
        out_data=np.vstack((out_data,[i,aLData[i],aLCon[i]]))
    out_data = out_data[1::]
    form = ['%1d','%20.19f','%20.19f']
    np.savetxt(out_file,out_data,fmt=form,delimiter='  ',newline='\n')

    return aMean, aCon, aLData, aLCon

def CalcBeta(tLMean, tLVar, lSize, direc, fileNum):
    '''

    '''

    lNum = len(tLMean)

    bLData = np.zeros((lNum-3))
    bMean = 0.0
    bLCon = np.zeros((lNum-3))
```

```
    bCon = 0.0
    tSize = 0

    for l in range(lNum-3):
        pSize = (lSize[l,0] + lSize[l+3,0])
        bLData[l] = (tLMean[l+3,0] - tLMean[l,0])
        if abs(bLData[l]) > np.pi:
            bLData[l] +=  (-1)*(abs(bLData[l])/bLData[l])*2*np.pi
        bLCon[l] = lSize[l,0]*tLCon[l,0] + lSize[l+3,0]*tLCon[l+3,0]
        bLCon[l] /= pSize
        if (l != 0):
            bMean += pSize*(bLData[l])
            bCon += pSize*(bLCon[l])
            tSize += pSize

    bMean /= tSize
    bCon /= tSize

    out_direc = direc+"/Beta"
    if not os.path.exists(out_direc):
        os.makedirs(out_direc)
    out_file = out_direc + "/Beta_" + fileNum
    out_data = np.empty([3])
    out_data=np.vstack((out_data,[-1,bMean,bCon]))
    for i in range(1,lNum-3):
        out_data=np.vstack((out_data,[i,bLData[i],bLCon[i]]))
    out_data = out_data[1::]
    form = ['%1d','%20.19f','%20.19f']
    np.savetxt(out_file,out_data,fmt=form,delimiter='  ',newline='\n')

    return bMean, bCon, bLData, bLCon

def CalcPhi(bLData,aLData,bLCon,aLCon,bMean,aMean,bCon,aCon,lSize,direc,fileNum,
            eRatio = 0.4827):
    '''

    '''

    lNum = len(bLData)

    pLData = np.zeros((lNum-3))
    pLMean = 0.0
    pMean = eRatio*bMean - aMean # Is this correct or should I do it the long way?
    pLConData = np.zeros((lNum-3))
    pLCon = 0.0
    pCon = eRatio*bCon + aCon

    for l in range(lNum-3):
        pLData[l] = eRatio*bLData[l] - aLData[l]
        pLConData[l] = eRatio*bLCon[l] + aLCon[l]    # Is this correct or should I
            account for size again?
        if(l != 0):
```

```
            pLMean += pLData[l]
            pLCon += pLConData[l]

    pLMean /= (lNum-4)
    pLCon /= (lNum-4)

    out_direc = direc+"/Phi"
    if not os.path.exists(out_direc):
        os.makedirs(out_direc)
    out_file = out_direc + "/Phi_" + fileNum
    out_data = np.empty([3])
    out_data=np.vstack((out_data,[-1,pMean,pCon]))
    out_data=np.vstack((out_data,[-2,pLMean,pLCon]))
    for i in range(1,lNum-3):
        out_data=np.vstack((out_data,[i,pLData[i],pLConData[i]]))
    out_data = out_data[1::]
    form = ['%1d','%20.19f','%20.19f']
    np.savetxt(out_file,out_data,fmt=form,delimiter='  ',newline='\n')

    return pMean, pCon, pLMean, pLCon

def CalcLambda(bLData,bLCon,bMean,bCon,lSize,direc,fileNum):
    '''

    '''

    lNum = len(bLData)

    lLData = np.zeros((lNum))
    lLConData = np.zeros((lNum))
    lLMean = 0.0
    lLCon = 0.0
    lMean = 2*np.pi / bMean
    lCon = (2*np.pi / bMean**2)*bCon

    for l in range(lNum):
        lLData[l] = 2*np.pi / bLData[l]
        lLConData[l] = (2*np.pi / bLData[l]**2)*bCon   # Is this correct or should I
            account for size again?
        if(l != 0):
            lLMean += lLData[l]
            lLCon += lLConData[l]

    lLMean /= (lNum-1)
    lLCon /= (lNum-1)

    out_direc = direc+"/Lambda"
    if not os.path.exists(out_direc):
        os.makedirs(out_direc)
    out_file = out_direc + "/Lambda_" + fileNum
    out_data = np.empty([3])
    out_data=np.vstack((out_data,[-1,lMean,lCon]))
```

```
    out_data=np.vstack((out_data,[-2,lLMean,lLCon]))
    for i in range(1,lNum-3):
        out_data=np.vstack((out_data,[i,lLData[i],lLConData[i]]))
    out_data = out_data[1::]
    form = ['%1d','%20.19f','%20.19f']
    np.savetxt(out_file,out_data,fmt=form,delimiter='  ',newline='\n')

    return lMean, lCon, lLMean, lLCon, lLData, lLConData

def CalcKVec(lMean, lCon, lLData, lLCon, direc, fileNum):
    '''

    '''

    lNum = len(lLData)

    kLData = np.zeros((lNum))
    kLConData = np.zeros((lNum))
    kLMean = 0.0
    kLCon = 0.0
    kMean = 1 / lMean
    kCon = (1 / lMean**2)*lCon

    for l in range(lNum):
        kLData[l] = 1 / lLData[l]
        kLConData[l] = (1 / lLData[l]**2)*lLCon[l]    # Is this correct or should I
            account for size again?
        if(l != 0):
            kLMean += kLData[l]
            kLCon += kLConData[l]

    kLMean /= (lNum-1)
    kLCon /= (lNum-1)

    out_direc = direc+"/KMag"
    if not os.path.exists(out_direc):
        os.makedirs(out_direc)
    out_file = out_direc + "/KMag_" + fileNum
    out_data = np.empty([3])
    out_data=np.vstack((out_data,[-1,kMean,kCon]))
    out_data=np.vstack((out_data,[-2,kLMean,kLCon]))
    for i in range(1,lNum-3):
        out_data=np.vstack((out_data,[i,kLData[i],kLConData[i]]))
    out_data = out_data[1::]
    form = ['%1d','%20.19f','%20.19f']
    np.savetxt(out_file,out_data,fmt=form,delimiter='  ',newline='\n')

    return kMean, kCon, kLMean, kLCon

# OTHER ##################################################

def FastFourier(ucData):
```

```
    '''
    '''

    return None

def IonAngle(ucData,a,b,fileNum):
    '''
    IonAngle -
        Calculate the average angle between two given ions.

    @params ::  ucData  - The ion data separated into unit cells
                a, b    - The two ion numbers to be measured
                fileNum - The file number being considered

    @return ::  angleMean, angleVar - The average and variance of angles between ions
         a and b in a given unit cell.
                angle   - The data for each unit cell
                sumAngleMean    - The average sum of angles in each unit cell
    '''

    side=len(ucData)
    angle=np.zeros((side,side,side))

    for i in range(side):
        for j in range(side):
            for k in range(side):
                dot = np.dot(ucData[i,j,k,a],ucData[i,j,k,b])
                dot = dot/abs(dot) if (abs(dot) > 1.0) else dot        # Clip to
                    [-1.0,1.0]

                angle[i,j,k] = abs(np.arccos(dot)) # Only care about magnitude

    angleMean = circmean(angle)
    angleVar = circvar(angle)   #This variance is NOT a confidence interval as the
        others are

    out_direc = direc+"/Layer2Angles/Ions" + str(a) + str(b)
    if not os.path.exists(out_direc):
        os.makedirs(out_direc)
    out_file = out_direc + "/Angles_" + fileNum
    out_data = np.empty([5])
    out_data=np.vstack((out_data,[-1,-1,-1,angleMean,angleVar]))
    for i in range(1,side):
        for j in range(1,side):
            for k in range(1,side):
                out_data=np.vstack((out_data,[i,j,k,angle[i,j,k],0.0]))
    out_data = out_data[1::]
    form = ['%1d','%1d','%1d','%20.19f','%20.19f']
    np.savetxt(out_file,out_data,fmt=form,delimiter='  ',newline='\n')

    return angleMean, angleVar, angle
```

```
# main ##########################################################

for n in range(1,confRange+1):
    fNum = str(n).zfill(3)
    file = 'MinConf_'+fNum+".dat"
    print("File: " + direc + "/" + file)
    data = ReadFile(direc, file)
    ucData = RawToUC(data,direc,fNum)
    gData = CalcGamma(ucData,direc,fNum)
    tData = CalcTheta(ucData,direc,fNum)
    gLMean, gLVar, lSize, _ = CalcLayerGamma(gData,direc,fNum)
    lMag = CalcLayerMag(ucData,lSize,direc,fNum)
    tLMean, tLVar, tLCon, _ = CalcLayerTheta(tData,lSize,direc,fNum)
    gIMean, gIVar, _ = CalcIonGamma(gData,direc,fNum)
    aMean, aCon, aLData, aLCon = CalcAlpha(tLMean, tLCon, lSize, direc, fNum)
    bMean, bCon, bLData, bLCon = CalcBeta(tLMean, tLCon, lSize, direc, fNum)
    pMean, pCon, _, _ = CalcPhi(bLData,aLData,bLCon,aLCon,bMean,aMean,bCon,aCon,lSize
        ,direc,fNum)
    lMean, lCon, _,_,lLData, lLCon = CalcLambda(bLData,bLCon,bMean,bCon,lSize,direc,
        fNum)
    kMean, kCon, _, _ = CalcKVec(lMean, lCon, lLData, lLCon, direc, fNum)
    for a in range(1,4):
        for b in range(a+1,4):
            angleMean, angleVar,_, = IonAngle(ucData,a,b,fNum)
    CalcOP(kMean,ucData, direc, fNum)
    # UCToLayer(ucData,direc,str(n).zfill(3))
    # UCToIon(ucData,direc,str(n).zfill(3))
```

E.1.2 Combine

```
# Kyle Hall 2020

import numpy as np
import os
import sys

# Check for command line arguments. If none present, ask for arguments
if not sys.stdin.isatty():
    direc=input("Directory Name: ")
    confRange=int(input("Input number of minimum configurations: "))
    temp=float(input("Input the temperature of the simulation: "))
else:
    direc=sys.argv[1]
    confRange=int(sys.argv[2])
    temp=float(sys.argv[3])

quantities = ('Lambda', 'KMag', 'Phi', 'Gamma', 'Mag', 'Layer2Angles', 'OP')    #
    Choose only the values you want combined.

# DATA IN ##############################################

def ReadFile(direc, fileType, dataType=None):
    '''

    '''
    global confRange

    if(dataType == "Layer"):
        fileName = dataType + "/" + fileType +"L"
    elif(dataType == "UC"):
        fileName = dataType + "/" + fileType + "UC"
    elif(dataType == "Ion"):
        fileName = dataType + "/" + fileType + "I"
    elif(dataType == "Angles"):
        fileName = "Angles"
    elif(fileType == "OP"):
        fileName = "F" + str(dataType)
    else:
        fileName = fileType

    data = []

    form = ['%d', '%20.19f', '%20.19f']
    for i in range(1,confRange+1):
        input_file = direc + "/" + fileName + "_" + str(i).zfill(3)
        data.append(np.genfromtxt(input_file, defaultfmt=form))

    data = np.array(data)

    return data
```

```
# DATA COMBINE ##################################################

def CombineSecondary(direc,quantity,data):
    '''

    '''
    global confRange, temp

    meanData = np.zeros((confRange))
    varData = np.zeros((confRange))
    mean = 0.0
    var = 0.0
    err = 0.0

    for i in range(len(data)):
        meanData[i] = data[i,0,1]
        varData[i] = data[i,0,2]

        mean += meanData[i]
        var += varData[i]**2

    print(np.var(meanData))

    var = var + np.var(meanData)

    mean /= confRange

    var = var/confRange
    err = np.sqrt(var)/(np.sqrt(confRange-1))    #

    out_direc = "./Means"
    if not os.path.exists(out_direc):
        os.makedirs(out_direc)
    out_file = out_direc + "/" + quantity
    if not os.path.exists(out_file):    # Make file if it doesn't exist. Otherwise
        append to it
        aw_switch = 'w'
    else:
        aw_switch = 'a'

    f = open(out_file,aw_switch)
    out_data = np.array([temp,mean,var,err])
    np.savetxt(f,(out_data),fmt='%20.19f',delimiter='  ',newline='  ')
    f.write('\n')
    f.close()

    return mean, var, err, meanData, varData

def CombineGamma(direc, quantity, data,
                     dataType = 'Ion'):
```

```
    '''

    '''
    global temp, confRange

    def GammaIon(data):
        global temp, confRange

        meanData = np.zeros((confRange,4,2))
        varData = np.zeros((confRange,4,2))
        mean =  np.zeros((4,2))
        var =  np.zeros((4,2))
        err =  np.zeros((4,2))
        for i in range(len(data)):
            for j in range(4):
                for k in range(2):
                    meanData[i,j,k] = data[i,j,2*k+1]
                    varData[i,j,k] = data[i,j,2*(k+1)]

                    mean[j,k] += meanData[i,j,k]
                    var[j,k] += varData[i,j,k]**2

        for j in range(4):
            for k in range(2):
                var[j,k] = var[j,k] + np.var(meanData[:,j,k])
                mean[j,k] /= confRange
                var[j,k] /= confRange
                err[j,k] = np.sqrt(var[j,k])/np.sqrt(confRange-1)

        for j in range(4):
            out_direc = "./Means"
            if not os.path.exists(out_direc):
                os.makedirs(out_direc)
            out_file = out_direc + "/" + quantity + "Ion" + str(j+1)
            if not os.path.exists(out_file):    # Make file if it doesn't exist.
                Otherwise append to it
                aw_switch = 'w'
            else:
                aw_switch = 'a'

            f = open(out_file,aw_switch)
            out_data = np.array([temp,mean[j,0],var[j,0],err[j,0],mean[j,1],var[j,1],
                err[j,1]])
            np.savetxt(f,(out_data),fmt='%20.19f',delimiter='  ',newline='  ')
            f.write('\n')
            f.close()

        return mean, var, err, meanData, varData
    def GammaLayer(data):
        '''
        Todo
        '''
```

```
    global temp, confRange

    meanData = np.zeros((confRange,2,2))
    varData = np.zeros((confRange,2,2))
    mean = np.zeros((2,2))
    var = np.zeros((2,2))
    err = np.zeros((2,2))
    for i in range(len(data)):
        for j in range(2):
            for l in range(2):
                meanData[i,l,j] = data[i,l,2*(j+1)+1]
                varData[i,l,j] = data[i,l,2*(j+2)]

                mean[l,j] += meanData[i,l,j]
                var[l,j] += varData[i,l,j]**2

    for l in range(2):
        for k in range(2):
            var[l,k] = var[l,k] + np.var(meanData[:,l,k])
            mean[l,k] /= confRange
            var[l,k] /= confRange
            err[l,k] = np.sqrt(var[l,k])/np.sqrt(confRange-1)

    for j in range(2):
        out_direc = "./Means"
        if not os.path.exists(out_direc):
            os.makedirs(out_direc)
        out_file = out_direc + "/" + quantity + "Layer" + str(j+1)
        if not os.path.exists(out_file):    # Make file if it doesn't exist.
            Otherwise append to it
            aw_switch = 'w'
        else:
            aw_switch = 'a'

        f = open(out_file,aw_switch)
        out_data = np.array([temp,mean[j,0],var[j,0],err[j,0],mean[j,1],var[j,1],
            err[j,1]])
        np.savetxt(f,(out_data),fmt='%20.19f',delimiter='  ',newline='  ')
        f.write('\n')
        f.close()

    return None,None,None,None,None
def GammaUC(data):
    '''
    Todo
    '''
    return None,None,None,None,None

if(dataType == "Layer"):
    mean, var, err, meanData, varData = GammaLayer(data)
elif(dataType == "UC"):
    mean, var, err, meanData, varData = GammaUC(data)
```

```
    elif(dataType == "Ion"):
        mean, var, err, meanData, varData = GammaIon(data)
    else:
        print('Error : Gamma type not valid.')




    return None, None, None, None, None

def CombineMag(direc, quantity, data):
    '''

    '''
    global temp, confRange

    meanData = np.zeros((confRange,2))
    varData = np.zeros((confRange,2))
    mean = np.zeros((2))
    var = np.zeros((2))
    err = np.zeros((2))
    for i in range(len(data)):
        for l in range(2):
            meanData[i,l] = data[i,l,3]
            varData[i,l] = data[i,l,4]

            mean[l] += meanData[i,l]
            var[l] += varData[i,l]**2

    for l in range(2):
        var[l] += np.var(meanData[:,l])
        mean[l] /= confRange
        var[l] /= confRange
        err[l] = np.sqrt(var[l])/np.sqrt(confRange-1)

    for j in range(2):
        out_direc = "./Means"
        if not os.path.exists(out_direc):
            os.makedirs(out_direc)
        out_file = out_direc + "/" + quantity + "Layer" + str(j+1)
        if not os.path.exists(out_file):    # Make file if it doesn't exist.
            Otherwise append to it
            aw_switch = 'w'
        else:
            aw_switch = 'a'

        f = open(out_file,aw_switch)
        out_data = np.array([temp,mean[j],var[j],err[j]])
        np.savetxt(f,(out_data),fmt='%20.19f',delimiter='  ',newline='  ')
```

```
        f.write('\n')
        f.close()

    return None,None,None,None,None

def CombineLayerAngles(direc, quantity, data):
    '''

    '''
    global confRange, temp

    meanData = np.zeros((confRange))
    varData = np.zeros((confRange))
    mean = 0.0
    var = 0.0
    err = 0.0

    for i in range(len(data)):
        meanData[i] = data[i,0,3]
        varData[i] = data[i,0,4]

        mean += meanData[i]
        var += varData[i]**2

    print(np.var(meanData))
    var = var + np.var(meanData)

    mean /= confRange
    var /= confRange
    err = np.sqrt(var)/(np.sqrt(confRange-1))    #

    out_direc = "./Means"
    if not os.path.exists(out_direc):
        os.makedirs(out_direc)
    out_file = out_direc + "/" + quantity
    if not os.path.exists(out_file):    # Make file if it doesn't exist. Otherwise
        append to it
        aw_switch = 'w'
    else:
        aw_switch = 'a'

    f = open(out_file,aw_switch)
    out_data = np.array([temp,mean,var,err])
    np.savetxt(f,(out_data),fmt='%20.19f',delimiter='  ',newline='  ')
    f.write('\n')
    f.close()

    return mean, var, err, meanData, varData

def CombineOP(direc, quantity, op, data):
    '''
    Magnitude only
```

```
    '''

    global confRange, temp

    meanData = np.zeros((confRange,4))
    mean = np.zeros((4))

    for i in range(len(data)):
        for j in range(4):
            meanData[i,j] = data[i,j,1]

            mean[j] += meanData[i,j]

    mean /= confRange

    out_direc = "./Means"
    if not os.path.exists(out_direc):
        os.makedirs(out_direc)
    for j in range(4):
        out_file = out_direc + "/F" + str(op) + "_" + str(j+1)
        if not os.path.exists(out_file):    # Make file if it doesn't exist.
            Otherwise append to it
            aw_switch = 'w'
        else:
            aw_switch = 'a'

        f = open(out_file,aw_switch)
        out_data = np.array([temp,mean[j]])
        np.savetxt(f,(out_data),fmt='%20.19f',delimiter='  ',newline='  ')
        f.write('\n')
        f.close()

    return mean, meanData


    return None
# main ################################################

print(direc)
for q in quantities:
    data_direc = direc + "/" + q
    print('\t'+q)
    if (q == 'Gamma'):
        for dataType in ["Ion", "Layer"]:
            data = ReadFile(data_direc, q, dataType)
            mean, var, err, _, _ = CombineGamma(direc,q,data,dataType)
    elif (q == 'Mag'):
        data = ReadFile(data_direc, q, "Layer")
        mean, var, err, _, _ = CombineMag(direc,q,data)
    elif(q == 'Layer2Angles'):
        for i in range(1,4):
            for j in range(i+1,4):
```

```
            angle_direc = data_direc + "/Ions" + str(i) + str(j)
            data = ReadFile(angle_direc, q, "Angles")
            mean, var, err, _, _ = CombineLayerAngles(direc, q+str(i)+str(j),
                data)
elif(q == 'OP'):
    for k in range(3):
        data = ReadFile(data_direc, q, k+1)
        mean, _ = CombineOP(direc,q,k+1,data)
else:
    data = ReadFile(data_direc, q, None)
    mean, var, err, _, _ = CombineSecondary(direc,q,data)
```