UNIVERSITY OF EDINBURGH

DOCTORAL THESIS

Studies of Condensed Matter Excitations in Bicollinear Magnetic Materials and Disordered Perovskites using Inelastic Neutron Scattering Techniques

---

Author:	*Supervisor:*
Ryan Drew TRAVERS	Dr. Chris STOCK

A thesis submitted in fulfillment of the requirements
for the degree of Doctor of Philosophy

in the

School of Physics and Astronomy

September 7, 2022

Declaration of Authorship

I, Ryan Drew Travers, declare that this thesis titled, "Studies of Condensed Matter Excitations in Bicollinear Magnetic Materials and Disordered Perovskites using Inelastic Neutron Scattering Techniques" and the work presented in it are my own. I confirm that:

- The data presented in this thesis was obtained in an experiment carried out by Dr Chris Stock. I played a substantial role in the preparation of the single crystal samples, and the data analysis and interpretation are my own work.
- I certify that this thesis is my own work, except where indicated by referencing, and the work presented in it has not been submitted in support of another degree or qualification from this or any other university or institute of learning.
- I declare that this thesis is an original report of my research and that any collaborative contributions, such as with Dr Chris Hooley and Janika Reichstetter, have been indicated clearly and acknowledged. Due references have been provided on all supporting literatures and resources.

Signed:

Date:

Abstract

Neutron scattering is an important research technique that has furnished a wealth of information about condensed matter systems ever since Ernest O. Wollan and Clifford G. Shull first demonstrated its ability to probe the structure of polycrystalline samples, such as NaCl, on the Clinton Pile—a graphite moderated nuclear reactor built for research purposes at Oak Ridge, Tennessee. Since those early successful demonstrations, the experimental technique has been continuously improved such that it is now regularly used as a research tool by scientists in a plethora of different fields to glean information about the structure or dynamics of their samples. An example of these advances in the use of neutron scattering came when Bertram N. Brockhouse invented the triple-axis spectrometer, which was an instrument that facilitated the study of lattice dynamical behaviour within condensed matter systems. Without this innovation it would be much more challenging to study phonons—the quasi-particles representing the collective vibrations of atoms—within the perovskites and iron chalcogenides that were researched for this thesis.

This thesis details how neutron scattering has been used as a tool to study the structure and lattice dynamical behaviour of the relaxor ferroelectric, $PbMg_{1/3}Nb_{2/3}O_3$, and the iron-based chalcogenide, $Fe_{1+y}Te$. The study of $PbMg_{1/3}Nb_{2/3}O_3$ was to investigate the structure of the higher energy phonon modes, complementing previous research that successfully characterised the lower energy acoustic and optical phonons of this crystal, and to use information obtained from neutron spectroscopy data to compare the lattice dynamical behaviour of this crystal with ordered perovskites, such as $SrTiO_3$. An interest in studying the higher energy phonons has been aroused because the disorder within $PbMg_{1/3}Nb_{2/3}O_3$, such as the presence of short-range polar nano-regions between the ferroelectric Curie temperature and the so-called Burns temperature (the temperature at which the short-range polar order emerges), leads to the energy broadening and dampening of the transverse optical phonons and triggers their precipitous collapse into the acoustic phonon branch. Previous neutron scattering studies suggested that the band of phonon scattering located at energy scales where the higher energy longitudinal and transverse optical phonons were predicted to be present were due to the occupancy disorder on the B-site of the perovskite between the Mg and Nb atoms. However, in this thesis investigations of the longitudinal and transverse scattering channels revealed lattice dynamical behaviour similar to those expected in $SrTiO_3$, which has no occupancy disorder on the B-site. The hypothesis of disorder within the higher energy modes is further negated by studying the structure factors at various reciprocal lattice vectors that reveals a close agreement with a model for the lattice dynamics that involves the motion of the oxygen and B-site ions.

The second study within this thesis investigates the softening of the transverse acoustic phonons within the $Fe_{1+y}Te$ system using time-of-flight neutron spectroscopy. This iron-based system has a complex structural, magnetic and electronic order that depends on the concentration of interstitial Fe atoms within the crystal structure and is often compared with other iron-based systems that share similar physical properties, such as FeSe, underdoped $Ba(Fe_{0.94}Co_{0.03})_2As_2$ and optimally doped $Ba(Fe_{0.94}Co_{0.06})_2As_2$. Studies of these systems revealed softening of the acoustic phonons close to the Brillouin zone centre ($q\sim 0$) that is indicative of the role that nematic order of the electronic charge has on the lattice dynamics within their crystal structures. However, investigations into the softening of the transverse acoustic phonon within low-y $Fe_{1+y}Te$ reveal that the softening within this system is mostly contained to the Brillouin zone edge and that the entire phonon branch is being driven to lower energies by scattering channels that open up as the tetragonal–monoclinic phase transition temperature is approached. The origin of the scattering channels is unknown, but the electronic nematic order could still be playing a role, albeit one that differs in effect to that observed in the other iron-based systems.

The final study is a theoretical investigation into the origin of the bicollinear double stripe antiferromagnetic order that is observed in $Fe_{1+y}Te$ with a low interstitial iron concentration. The hypothesis is that this complicated magnetic ground state comes about due to the interplay between the spin and orbital degrees of freedom within the crystal due to the Jahn-Teller effect and the small energy scale associated with the splitting of the d_{Xz} and d_{Yz} orbitals in the magnetic (S=1) Fe atoms. This builds on previous theoretical studies which proposed models of the magnetic interactions within the FeTe single crystals that negated the possibility of an interplay between the spin and orbital degrees of freedom and hoped to explain the double stripe magnetic order by a model that included either the spin or orbital physics but not both. A Rayleigh-Schrödinger perturbative model has been developed that includes the second and fourth order virtual hopping of the electrons on neighbouring magnetic Fe atoms surrounded by tetrahedra of non-magnetic Te atoms. A domain space for each of the Fe atoms in the S=1 spin state has 3 degrees of freedom for the spin projection and 2 degrees of freedom for the orbital projection meaning that the resultant Hamiltonian that couples two neighbouring magnetic Fe atoms has a 36×36 matrix structure. This effective Hamiltonian leads to interactions that include spin-only terms, orbital pseudo-spin only terms and terms that describe the interaction between both the spin and orbital physics. The spin operators include single-ion anisotropy, an Ising term that correlates spin projections, an Ising-like term that correlates "mid-planeness" and a spin exchange term. The orbital pseudo-spin operators include a single-site "orbital transverse field" term and a two-site "orbital transverse field" term. The interplay between the spin and orbital degrees of freedom is produced by the multiplication of the spin-only and orbital pseudo-spin only terms in the effective Hamiltonian.

Lay Summary

This thesis aims to study the properties of materials using neutron scattering. In this technique, a beam of neutrons is directed towards and scatters from a sample made from the material to be studied and the scattered neutrons are detected. Information about the scattered neutrons, viz., the position of detection with respect to the sample, their energy and momentum, can be used to study the material and reveal information about its structure and the way in which atoms move within it. For this thesis, neutron scattering data was analysed in order to understand the collective motion of atoms within lead magnesium niobate crystals (PMN) and whether their motion is dominated by the irregular structure inherent in the crystal. This irregular structure comes from the fact that one of the sites in the lattice structure is irregularly occupied by different atoms—it is occupied by Mg and Nb atoms with probabilities of one-third and two-thirds respectively. This is an important research question since the collective vibrations of atoms over a long length scale would indicate that certain properties of these crystals can transcend the short length scales more commonly associated with disordered structures.

Neutron scattering data has also been analysed to investigate the properties of iron telluride, $Fe_{1+y}Te$. $1 + y$ in the chemical formula of $Fe_{1+y}Te$ indicates that the crystal has a ratio of Fe to Te atoms of 1+y:1, and therefore increasing y increases the number of Te atoms relative to the number of Fe atoms in the crystal. This iron-based system has been studied close to a critical temperature at which it transitions from a higher temperature structure to a lower temperature structure. This structural change can be summarised as the atoms within the crystal changing positions to lower the overall energy of the crystal. The neutron scattering data revealed that the collective vibration of the constituent atoms within the crystal responded to the instabilities triggered by the approach of this critical transition temperature. With reference to previous studies on similar iron-based systems an attempt was made to see if the neutron scattering data, and specifically the observed energies associated with the vibrations of the crystal, could be explained by the emergence of a difference in the physical properties of the system along two perpendicular directions in the plane of iron atoms in the crystal. This hypothesis, at least to the extent described by previous studies, could not be supported by the neutron scattering data meaning that the exact behaviour of the crystal continues to allude us. Nonetheless, there were enough qualitative similarities in the neutron scattering data with the previous studies to suggest that the basic idea behind the hypothesis was approaching the true cause of the observed behaviour of the crystal.

The final substantive chapter of the thesis contains the work carried out for a collaborative theory project. This had the objective of investigating the magnetism and electronic structure of $Fe_{1+y}Te$ by developing a model derived from first principles classical and quantum mechanical forces and seeing whether the results of this model produces the experimentally observed properties of the crystal. An implicit assumption within the model is that the magnetism and electronic structure of the crystal is believed to derive from the geometrical distortion of the crystal structure and the relative displacement of the Fe-Te bonds, including both their re-orientation and extension or contraction.

Acknowledgements

I would like to acknowledge a few individuals without whose support it would have been impossible to carry out the research projects presented for this thesis. For the experimental projects involving $\mathrm{PbMg_{1/3}Nb_{2/3}O_3}$ and $\mathrm{Fe}_{1+y}\mathrm{Te}$, I would like to thank Dr Chris Stock who taught me both how to synthesis polycrystalline and single crystal samples and how to analyse X-ray and neutron scattering data to understand both the static and dynamical properties of condensed matter materials. Notwithstanding the disruption to the Coronavirus pandemic 2019 and the concomitant impact on access to neutron scattering facilities, he has done his upmost to help me to understand how the neutron scattering instruments function and how they can be best utilised to study the behaviour of single crystals. One benefit of the Coronavirus pandemic is that I was able to start collaborating with Dr Chris Hooley, and later with Janika Reichstetter, on a project to understand the origin of the magnetic interactions in $\mathrm{Fe}_{1+y}\mathrm{Te}$. The supervision of Dr Chris Hooley helped me to develop a deeper understanding of the link between the crystal and electronic structure of condensed matter and their magnetism and how the latter can be derived from a first principles study of the former.

Contents

List of Figures

List of Tables

List of Abbreviations

DFT	**D**ensity **F**unctional **T**heory
DOS	**D**ensity **O**f **S**tates
FWHM	**F**ull **W**idth **H**alf **M**aximum
LO	**L**ongitudinal Optic
PSI	**P**aul **S**cherrer **I**nstitut
PMN	**P**b**M**g$_{1/3}$**N**b$_{2/3}$O$_3$
PSD	**P**osition **S**ensitive **D**etector
PZN	**P**b**Z**n$_{1/3}$**N**b$_{2/3}$O$_3$
SINQ	Swiss Spallation Neutron Source
ToF	**T**ime-**o**f-**F**light
TAS	**T**riple-**A**xis-**S**pectrometer
TO	**T**ransverse Optic

Physical Constants

Ångstrom	$\text{Å} = 1 \times 10^{-10}\text{m}$
Electron Charge	$e = 1.602176634 \times 10^{-19}\text{C}$
Nuclear magneton	$\mu_N = 5.050783699 \times 10^{-27}\ \text{J/T}$
Permeability of free space	$\mu_0 = 1.25663706212 \times 10^{-6}\text{H/m}$
Plank's Constant	$h = 4.135667696 \times 10^{12}\text{meV}\cdot\text{s}$
Reduced Plank's Constant	$\hbar = 6.582119569 \times 10^{-13}\text{meV}\cdot\text{s}$

List of Symbols

$\mathbf{e}$	atomic displacement vector
$\mathbf{M}(\mathbf{Q})$	Fourier transform of the magnetisation
$\mathbf{k_i}$	incident wavevector
$\mathbf{B}(\mathbf{r})$	localised magnetic field
$\mathbf{M}(\mathbf{r})$	localised magnetisation
$\mathbf{s}$	normal mode
$\mathbf{r}$	position vector
$\mathbf{d}$	position vector of atom within the unit cell
$\mathbf{G}$	reciprocal lattice vectors
$\mathbf{Q}$	scattering vector
$\mathbf{k_f}$	scattered wavevector
$n(\omega)$	Bose-Einstein distribution
W_d	Debye-Waller factor exponent
$V(\mathbf{Q})$	Fourier transform of the interaction potential
E_i	incident neutron energy
k_i	incident wavevector
$V(\mathbf{r})$	interaction potential
y	interstitial iron concentration
s_I	intrinsic spin of the interacting nucleus
s_n	intrinsic spin of the neutron
m_n	mass of the neutron
q	reduced wavevector
$S(\mathbf{Q},\omega)$	scattering function
b	scattering length
E_f	scattered neutron energy
k_f	scattered wavevector
T	temperature
v_0	volume of the unit cell
ν	frequency
σ	interaction cross section
μ_n	magnetic dipole moment
τ	particle lifetime
Γ	particle linewidth
Ω	solid angle
β	thermodynamic beta
λ	wavelength

Chapter 1

General Introduction

1.1 What is the neutron?

The objective of this thesis is to study the structural and dynamical properties of condensed matter using neutrons as the probe. However, before delving deeper into the theoretical and experimental studies undertaken for completion of this thesis, it would be conducive for the reader to have a basic understanding of the neutron. It should be appreciated that without understanding the particle used as a probe it is impossible to understand how it interacts with other materials. It will also help the reader to understand the differences between the neutron and other particles used for probing materials, e.g., X-rays, and why an experimentalist might want to choose one over the other and vice versa. As with all fields of research it also helps to understand the historical developments and achievements because without their effort there would not be the means for carrying out the work that we do today.

1.1.1 A composite particle with no net charge?

Before the discovery of the neutron by James Chadwick in 1932, it had long been suggested that there existed a small particle with no net electrical charge, but the question as to its composition perplexed scientists for a long time [1]. For example, Walther Nernst suggested that a proton and electron could combine to form a particle with no net charge and that it would have the properties of the so-called æther- the fifth element that was believed to be the medium through which light travelled. It was also erroneously suggested by Louis de Broglie that the γ-radiation that emanated from radioactive substances was composed of small electrically neutral particles and that these particles would break up and give off electrons. Our knowledge about the existence of a neutral subatomic particle, which was later to be called the "neutron", was brought much closer when Ernest Rutherford, a New Zealander whose work in Montreal, Manchester and Cambridge, culminated in his theorising that under certain conditions the electron and proton in an hydrogen atom might combine to form a particle which possesses an external field that is practically zero, except for very close to the particle, and that this particle would be difficult to contain in a sealed vessel. His theorising was especially prescient in his remarks about the difficulty with which these particles with no net charge would be able to enter deep into the bulk of materials:

> *"Its presence would probably be difficult to detect by the spectroscope, and it may be impossible to contain it in a sealed vessel. On the other hand, it should enter readily the structure of atoms, and may either unite with the nucleus or be disintegrated by its intense field"*

In this quotation it is already possible to appreciate the advantages and disadvantages of using these electrically neutral subatomic particles for probing materials: owing to the low cross section of neutrons for interactions via the Coulombic force they are able to penetrate deep into the bulk of materials, and therefore permit the study of bulk properties of materials, but, on the other hand, it makes detecting them, after they have interacted with materials, much more challenging. It was further prescient of Ernest Rutherford to surmise that the particle's external field would not be exactly zero, and non-negligible at very small distances from the centre, as this preceded the discovery of quarks by many decades. He clearly understood that this particle with net neutral charge must be composed of other particles whose electrical charges cancel each other out. In fact, this is exactly the case with the neutron: it is composed of one up quark and two down quarks with their charges being $\frac{2}{3}e$ and $-\frac{1}{3}e$ respectively, thus combining to produce no net charge.

As mentioned in James Chadwick's Nobel Prize lecture [1], he began to think about the experiments carried out, amongst many others, by Frédéric and Irène Joliot-Curie. They experimented with radiation from a beryllium sample that passed through a thin window into an ionisation chamber composed of air. When paraffin wax, a material containing a lot of hydrogen, was placed in front of the window the ionisation within the chamber increased, which the Joliot-Curies proved to be protons being ejected from the wax. James Chadwick experimented with other materials placed in front of the thin window to the ionisation chamber (e.g., lithium, beryllium and boron etc.) and observed that the ejected particles had varying ranges. The differences in the ranges of the ejected particles indicated to him that these were recoiling from a collision with a massive particle emanating from the beryllium source. By comparing the ranges of recoiling particles from the paraffin wax and when the ionisation chamber was filled with nitrogen, he found that the radiation coming from the beryllium source must have a mass that is approximately the same as the proton's mass. Furthermore, since this radiation penetrated deeply into materials—as opposed to protons which only penetrated into the surface of materials owing to its electric charge—this particle must be electrically neutral. This discovery culminated in his being awarded the Nobel Prize in Physics in 1935 and started a chain of research into how this particle might be used for other research purposes.

1.1.2 Properties of the neutron

In the previous section the reader was introduced to the history of the search for the neutron by scientists such as Ernest Rutherford and James Chadwick. Some of the properties of the neutron, such as it being electrically neutral but being composed of charged quarks, have already been broached. In this section the reader will be introduced to some other fundamental properties of the neutron and it will be explained why these properties make the neutron appropriate as a probe for studying the properties of materials and, in a lot of cases, better for studying the structure and dynamics of materials than when using X-rays (i.e., photons).

Up till now we have been implicitly discussing the properties of the neutron in terms of the classical physical picture, but as quantum theory teaches us particles can also be represented as spatially localised waves travelling through space, i.e., the wave-particle duality in physics. As a wave the neutron's wave-function is represented by a wave packet, as illustrated in figure 1.1, with the wavelength being approximately equal to the distance between two wave crests or troughs. As described by Louis de Broglie in 1924 [2], the energy, E, and momentum, p, of particles, in the wave description of quantum mechanics is related to the wavelength, λ, and frequency, ν, through the following equations:

$$E = \hbar\nu = \frac{h^2}{2m}\lambda^2; \tag{1.1}$$

$$p = \hbar k = \frac{h}{\lambda}; \tag{1.2}$$

where h and $\hbar$ are Planck's constant and the reduced Planck's constant respectively, k is the wavevector given by $k = \frac{2\pi}{\lambda}$, and m is the particle's mass—which, in the case of the neutron, will be represented as m_n forthwith. Supposing that we have neutrons that have reached thermal equilibrium at room temperature, i.e., T=300K, then the average energy of these neutrons will be E=$300K_B$, and, using the relation above in eq. 1.1, their average de Broglie wavelength will be approximately 1.8×10^{-10}m. The de Broglie wavelength of neutrons at thermal temperatures is therefore approximately the same as the lattice spacing in condensed matter, i.e., the spacing between atoms on adjacent sites of crystalline materials. The consequence of this is that neutrons with thermal energies can be used to probe the structure of materials in condensed mater physics—this is highly convenient for experimentalists because it means that facilities can work at normal temperature ranges and do not need to expend a lot of energy in changing the temperature of a moderator substance.

What about some other properties of the neutron? One property of the neutron, which prima facie could be slightly perplexing, is that the particle has a magnetic moment and an intrinsic spin. This would appear to contradict the electrical neutrality of the particle since both a magnetic moment and spin come from a particle with an electrical charge. However, this is premised on the neutron having no internal structure but, as has been aforementioned, the neutron is composed of three quarks which have either a charge of $\frac{2}{3}e$ or $-\frac{1}{3}e$. The non-homogeneous electrical structure of the neutron means that on spinning these quarks can produce both magnetic moments and intrinsic spins that combine to produce a non-zero magnetic moment and spin of the neutron. These have been calculated as μ_n=-1.913μ_N and $s_n = \frac{1}{2}$ respectively [3] where μ_N is the nuclear magneton.

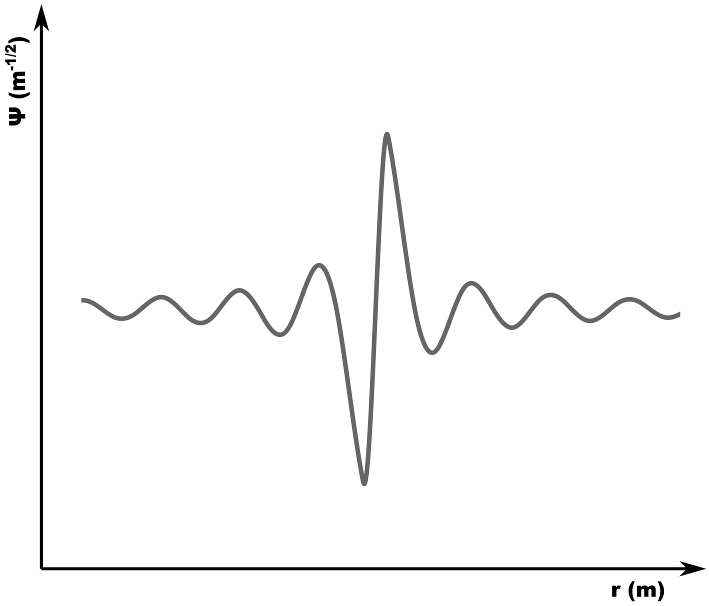


Figure 1.1: The quantum description of the neutron is an oscillatory wavefunction that is localised about a point in space where it is most probable to be located after measuring its position.

One might ask the question why does the neutron have a spin of $\frac{1}{2}$ when the three quarks, which are fermions, could combine to form a spin of $\frac{1}{2}$ or $\frac{3}{2}$. The quarks, like electrons in an atom, have bound states in the nucleus and have to obey Pauli's exclusion principle, which states that no two particles can have the same quantum state. As each state of the quark has a degeneracy of two—the quark can have up or down spin—the ground state is formed by two down quarks being in the same quantum state with opposite spin, which leads to the total spin of the particle being $\frac{1}{2}$. It is possible that a particle is formed in a higher energy state in which case the spin of this particle could be $\frac{3}{2}$ (a so-called delta particle) but the energy scales between the ground and excited state of nucleons is much greater than the energy scales within an atomic nucleus meaning that this is only possible on earth within large particle accelerators.

Since the neutron has a magnetic moment and intrinsic spin, it is able to interact with other particles with moments and spins meaning that it can probe both the structure and dynamics of magnetic materials in a similar way as its nuclear interaction with nuclei enables the determination of structure and dynamics of lattices in condensed matter. The interaction mechanism is rather complicated because there are many contributions to the magnetisation of materials, i.e., the intrinsic spin of electrons and electron currents, but it should at least be appreciated by the reader that having a moment and spin leads to the neutron being an excellent experimental probe for condensed matter physics and, in fact, it was only because of magnetic neutron scattering that the antiferromagnetic state was proven to exist in 1951 by Clifford Shull [4]—experiments for which he was awarded the Nobel Prize in Physics in 1994 [5].

Some of the properties of the neutron discussed here give an explanation as to why it might be preferable to use neutrons as a probe for studying the structure and dynamics of condensed matter. For instance, having no net nuclear charge means that neutrons can penetrate much more deeply into the bulk of materials as opposed to X-rays, which probe the surface layer of crystals. Furthermore, the magnetic moment and intrinsic spin of the neutron means that it has a large magnetic scattering cross section and is able to be used for studying physical phenomena such as antiferromagnetism, chiral magnetic structure and spin waves. As a technique, the use of neutrons in inelastic neutron scattering also provides access to a larger range of energy and momentum compared with techniques that rely on the use of photons to probe condensed matter systems. This is clearly demonstrated in Fig. 1.2 that shows the energies and momenta that are able to be accessed in inelastic neutron scattering and other popular experimental techniques, including Raman scattering and inelastic X-ray scattering. There are some overlapping energy and momentum ranges for different techniques (e.g., there is the overlap between inelastic neutron and X-ray scattering), which means that for those wanting to probe within this range it is possible to choose between the techniques. In this case, the choice of which technique to use might be decided by comparing the cross sections of neutrons and X-rays with the sample. Note that in Fig. 1.2 the energy scale is in units

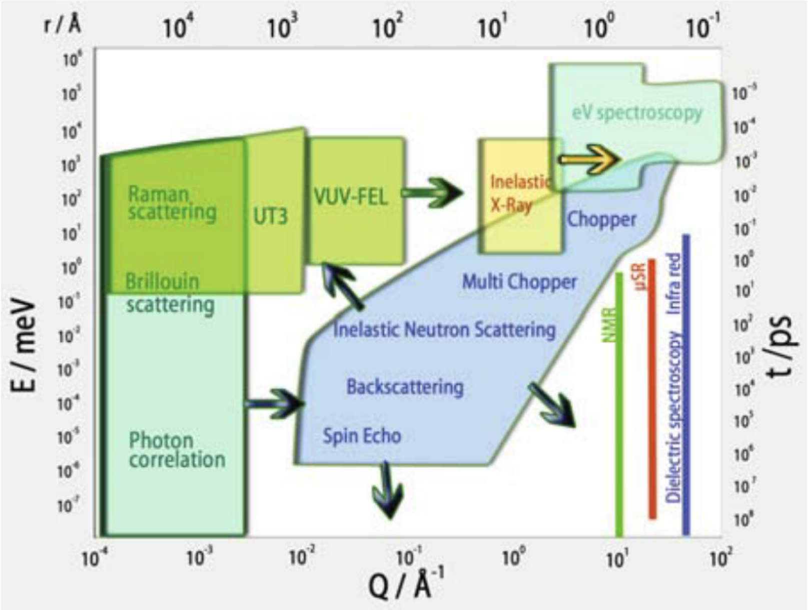


FIGURE 1.2: One of the several advantages of inelastic neutron scattering experiments is that it is possible to access a large range of both energy and momentum compared with some of the other techniques that are regularly used for probing condensed matter systems, such as Raman scattering and inelastic X-ray scattering [6].

of meV, which is a natural energy scale to use for studying phonons and spin waves in condensed matter systems. However, it is possible to also see the unit THz being used in the scientific literature, which has the conversion of $1\text{THz} = 4.1\text{meV}$.

1.2 Neutron scattering

In this section, the reader will be introduced to some of the basics of neutron scattering. It will cover enough material for the general physicist to be able to understand the experimental techniques practised in this thesis for studying the structure and dynamics of condensed matter. If the reader would like to understand a certain topic herein discussed, it would be recommended that they read some of the seminal general purpose and applied textbooks in the field of neutron scattering- these have been heavily relied upon to produce this introductory summary [3, 7, 8].

The best approach for understanding neutron scattering is to first comprehend how a neutron might interact with an isolated bound nucleus that is fixed to a point in empty space (meaning that no energy transfer can take place during the interaction of the neutron with the isolated nucleus). It was shown in the preceding section that a neutron's wavelength at thermal energies is of the order of 10^{-10}m, but the strong nuclear force, one of the fundamental forces that a neutron experiences, with the other being electromagnetic forces, only acts on length scales up to roughly 10^{-15}m. Hence, when a neutron impinges on an isolated bound nucleus, in a quantum mechanical picture of particle interactions, it would appear that the incoming neutron is a plane wave interacting with a point particle in space. Since the probability of interaction between the neutron and the nucleus is so small (a corollary of the penetration depth of the neutron being so large) the outgoing neutron can be represented by the unaffected plane wave and a spherical wave centred at the nucleus resulting from the interaction between the two particles. The wavefunctions of the plane wave that impinges on the isolated bound nucleus, ψ_i, and the spherical wave created by the interaction between the incoming neutron and the nucleus, ψ_{sph}, are given by the following:

$$\psi_i = \exp{(i\mathbf{k} \cdot \mathbf{r})}, \tag{1.3}$$

$$\psi_{sph} = -\frac{b}{r}\exp{(i\mathbf{k} \cdot \mathbf{r})}, \tag{1.4}$$

where $\mathbf{k}$ is the wavevector of the plane wave and b is the nuclear scattering length parameter representing the strength of interaction between the incoming neutron and the isolated nucleus. The minus sign is arbitrary and simply means that the scattering potential is repulsive rather than attractive. The outgoing wave is a sum of the initial plane wave and the spherical wave in the asymptotic limit, i.e., $\vec{r} \to \infty$. This explains the breakdown of the conservation of probability close to the centre of the scattering potential.

To physically interpret a real nuclear scattering length it is conducive for the reader's understanding to first determine the cross section for scattering using the ratio of scattered neutron flux to the ratio of incoming flux at a constant distance from the isolated bound nucleus, i.e., the neutron scattering cross section, σ_{scat}. If the speed of the neutrons is v then the incoming flux is v and the scattered flux through the same sphere of radius r is $4\pi r^2 \times b^2\frac{v}{r^2}$, giving a scattering cross section of the following:

$$\sigma_{scat} = 4\pi b^2. \tag{1.5}$$

The real part of the nuclear scattering length thus contributes to the nuclear scattering cross section. Without proving it in this thesis (the reader may wish to consult one of the aforementioned texts on neutron scattering) the nuclear absorption cross section is proportional to the imaginary part of the nuclear scattering length, i.e., $\sigma_{abs} = \frac{4\pi b_{imag}^2}{k}$, and is greater when the neutron is travelling at a slower speed (when ignoring the absorption resonance peaks). Hence, by looking up nuclear scattering lengths of isotopes it is possible to know which ones strongly scatter or absorb neutrons. It should be noted that there are often large differences between isotopes of the same element- an example being between the most abundant isotope of He with an absorption cross section of 0.00747(1) barn (for neutrons travelling at 2,200m/s) and ^{3}He with an absorption cross section of 5333.(7.) barn for the same speed of neutrons [9]. The difference in absorption cross sections can be explained by the stability of the nucleus and whether absorbing the neutron will increase its stability.

1.2.1 Neutron scattering for condensed matter

The field of condensed matter is much more interested in how neutrons scatter from structured materials rather than from isolated bound nuclei. It will therefore be much more useful to give a brief description of some of the types of interactions that can occur when neutrons interact with structured materials, which can be categorised into elastic and inelastic scattering and coherent and incoherent scattering and by the force that mediates the interaction (e.g., strong nuclear force or magnetic interactions).

For the benefit of the reader the concepts of coherent and incoherent scattering will be explained in the case of elastic scattering. Elastic scattering, as the name suggests, refers to an interaction process in which there is no net gain or loss of energy of the neutron. In inelastic scattering processes, on the other hand, energy is transferred between the incoming neutron and the particle with which the neutron is interacting with the force being mediated by a strong nuclear force or electromagnetic forces due to the magnetic moment and intrinsic spin of the neutron. For an elastic scattering process in which neutrons scatter from an ordered condensed matter sample with atoms at positions $\mathbf{r}_j$, the differential cross section (i.e., the ratio of the scattered flux of neutrons into a solid angle to the incident flux per unit solid angle) can be written in the following form:

$$\frac{d\sigma}{d\Omega} = |\sum_j b_j \exp{(i\mathbf{Q} \cdot \mathbf{r}_j)}|^2, \tag{1.6}$$

where $\mathbf{Q}$ is the scattering vector given by $\mathbf{Q} = \mathbf{k}_i - \mathbf{k}_f$, the incoming and scattered neutron wavevectors respectively, and b_j is the scattering length of the atom positioned at site $\mathbf{r}_j$. This expression comes from the constructive and destructive interference between scattered waveforms from neutrons interacting with the plurality of atomic sites—the same as for Bragg X-ray diffraction within a crystal. The plane formed by the scattering triangle, which is formed by this relationship and shown in Fig. 1.3, is called the scattering plane. In general, $\mathbf{k}_i$ and $\mathbf{k}_f$ do not have to be the same magnitude (i.e., if it is not restricted to the case of elastic scattering) and the plane formed by the scattering triangle, which is formed by the aforementioned relationship and shown in Fig. 1.3, is called the scattering plane.

To explain the concept of coherent and incoherent scattering it is necessary to remind the reader of the isotopic dependence of the scattering lengths, b. In condensed matter physics there is often a high degree of order within crystalline compounds and at any atomic site the element situated at that site could be in its isotopic form i with a probability of p_i. Assuming that there is no correlation between there being one isotope on one site and another isotope on another site, the entire crystal can be assumed to have a random distribution of isotopes at each atomic site. For a large flux of neutrons targeting a large sample, the scattering described in eq. 1.6 can be simplified as the statistical average of the scattering lengths at sites j:

$$\frac{d\sigma}{d\Omega} \sim \sum_{j,j'} \overline{b_j b_{j'}} \exp\left(i\mathbf{Q} \cdot (\mathbf{r}_j - \mathbf{r}_{j'})\right), \tag{1.7}$$

where $\overline{b_j b_{j'}}$ is the statistical average of the scattering lengths of isotopes at the two indexed sites. Splitting this expression into one part containing the same site and another part containing different sites:

$$\frac{d\sigma}{d\Omega} \sim \sum_{j \neq j'} \overline{b_j b_{j'}} \exp\left(i\mathbf{Q} \cdot (\mathbf{r}_j - \mathbf{r}_{j'})\right) + \sum_{j} \overline{b_j^2}, \tag{1.8}$$

and since there is no correlation between the isotopes on different sites this can be reformulated as:

$$\begin{aligned} \frac{d\sigma}{d\Omega} &\sim \sum_{j \neq j'} \bar{b}_j \bar{b}_{j'} \exp\left(i\mathbf{Q} \cdot (\mathbf{r}_j - \mathbf{r}_{j'})\right) + \sum_{j} \overline{b_j^2} \\ &= \sum_{j,j'} \bar{b}_j \bar{b}_{j'} \exp\left(i\mathbf{Q} \cdot (\mathbf{r}_j - \mathbf{r}_{j'})\right) + \sum_{j} (\overline{b_j^2} - \bar{b}_j^{\,2}), \end{aligned} \tag{1.9}$$

where the coherent and incoherent scattering are the first and second part of the second line of eq. 1.9 respectively. As can be seen the coherent scattering relates to the relative position of the atoms within the structured material and can therefore give the experimentalist information about the structural composition of the sample being investigated. On the other hand, the incoherent scattering does not contain information about the structure of the sample but does give some information relating to its isotopic composition.

From the coherent scattering formula it is possible to derive Bragg's law for the constructive interference of scattered waves. This constructive interference occurs in periodic crystals (i.e., crystals that possess translational symmetry) when the scattering vector, $\mathbf{Q}$, is equal to one of the reciprocal lattice vectors, τ, as is the case in Fig. 1.3. Since τ must be equal to an integer number of 2π multiplied by an inter-planar distance, d, the Bragg law equation, $n\lambda = 2d \sin\theta$, can be recovered.

This can also be applied to inelastic neutron scattering with the coherent scattering depending on the product of the statistical averages of the scattering lengths on two sites and the incoherent scattering depending on the variance of the scattering lengths on one site. The incoherent inelastic scattering is often seen as the background to the coherent inelastic scattering events but there is still some information that is able to be extracted about the dynamics of the system: in phonon scattering, for instance, it is possible to extract information about the density of phonon states from this incoherent inelastic background scattering.

Before proceeding to inform the reader about phonon and magnetic scattering by neutrons, it would be remiss to finally add that there is another dependence of the scattering length on the total spin of the interacting neutron and nucleus. These have spins of $s_n = \frac{1}{2}$ and $s_I = I$ respectively meaning that the total spin of the combined system can be either $I - \frac{1}{2}$ or $I + \frac{1}{2}$. The probability of the system being in each of these spin configurations is proportional

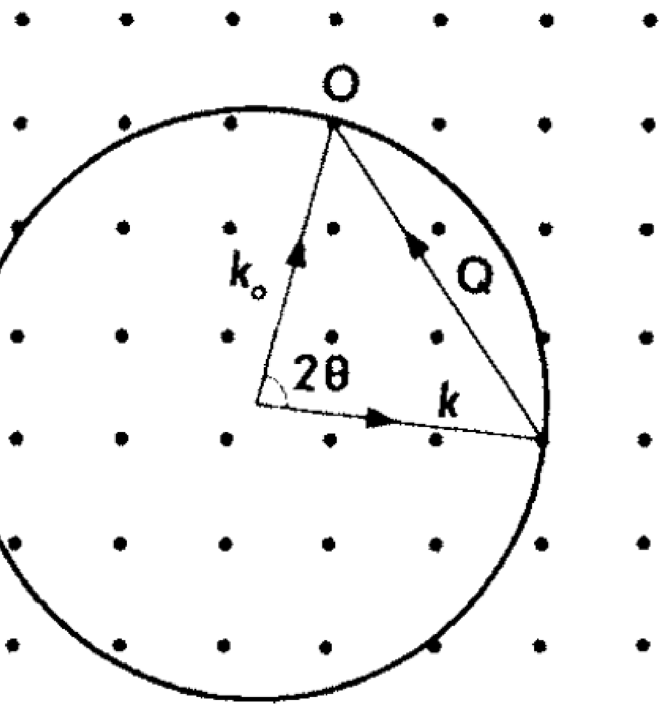


FIGURE 1.3: The scattering triangle formed by the scattering vector, **Q**, and its relationship with the momentum vectors of the incoming and scattered neutron. The scattering plane is constructed such that its normal vector is perpendicular to both the incoming and scattered momentum vectors (denoted as $\mathbf{k}_0$ and **k** in this schematic diagram [10]). Bragg's law occurs when the scattering vector is equal to one of the reciprocal lattice vectors.

to the number of spin projections meaning that the system has a probability of $\frac{I}{2I+1}$ of being in the lower spin state and $\frac{I+1}{2I+1}$ of being in the higher spin state. These probabilities can be used to calculate the cross sections for coherent and incoherent neutron scattering using the formulae disclosed in eq. 1.9.

How are inelastic scattering processes dealt with in the quantum mechanical picture of particle interactions? The assumption in the previous section that the scattered wave is small compared to the incident plane wave can be viewed as that the interaction potential, V, is a small perturbation to the physical system composed of the travelling neutron and the particle. Being a small perturbation this interaction can be treated using standard quantum perturbation theory with the rate of transition, $T_{i\to f}$, of the neutron from an initial state, i, to a final state, f, given by Fermi's golden rule:

$$T_{i\to f} = \frac{2\pi}{\hbar}|\langle f|\, V\,|i\rangle|^2\frac{d\Omega}{4\pi} \tag{1.10}$$

if the scattering process obeys energy and momentum conservation and zero otherwise. It is superfluous for the reader's understanding of this thesis to know the derivation of the complete double differential cross section for inelastic neutron scattering from this transition rate, instead it would suffice to simply know that this can be written as:

$$\frac{d^2\sigma}{d\Omega dE_f} = \frac{k_f}{k_i}\Big(\frac{m_n}{2\pi\hbar^2}\Big)^2\sum_{\lambda_i} p_{\lambda_i}\sum_{\lambda_f}|\,\langle\sigma_f\lambda_f|\,V(\mathbf{Q})\,|\sigma_i\lambda_i\rangle\,|^2\delta(E_{\lambda_f}-E_{\lambda_i}-\hbar\omega), \tag{1.11}$$

where the $|\sigma\lambda\rangle$ are the quantum states of the neutron including its spin, momentum and wavelength; p_{λ_i} is the probability of being in the initial neutron state and $V(\mathbf{Q})$ is the Fourier transform of the interaction potential:

$$V(\mathbf{Q}) = \langle\mathbf{k}_f|\,V(\mathbf{r})\,|\mathbf{k}_i\rangle = \int V(\mathbf{r})\exp{(i\mathbf{Q}\cdot\mathbf{r})}. \tag{1.12}$$

The interaction potential depends on the force mediating the interaction between the incoming neutron and the particle participating in the scattering event. In the case of nuclear scattering and magnetic scattering these will be described to the reader in the following two sections with enough detail being disclosed so that the reader can fully comprehend the experiments that have been done for completion of this thesis.

1.2.2 Principle of detailed balance

One of the important results in scattering theory, which is often used in the analysis of experimental neutron scattering data, is the principle of detailed balance. This relates the scattering of the neutron in energy gain mode to the scattering in energy loss mode and is due to the different occupancies of the energy levels of a system when the system is in thermal equilibrium. The relationship between the two types of scattering can be derived starting from the expression for the double differential scattering in eq. 1.11, which can be reformulated as the following:

$$\frac{d^2\sigma}{d\Omega dE_f} = \frac{k_f}{k_i} S(\mathbf{Q}, \omega), \tag{1.13}$$

where $S(\mathbf{Q}, \omega)$ is called the scattering function and measures the probability that a neutron imparts an energy of $\hbar\omega$ and a momentum change of $\hbar\mathbf{Q}$ to the system whilst maintaining the conservation of energy and momentum. This is a conditional probability equal to the probability that the neutron gains or loses both energy and momentum by making a transition from an initial state to a final state whilst the system does the reverse, i.e., whatever the neutron gains or loses in energy the system does the reverse change. Since the transition probability of the neutron going into a lower energy state or a higher energy state (where the difference in energy is equal to $\hbar\omega$) is assumed to be the same, the only differences in the scattering function must come from the difference in the probability of the system starting from a higher or lower energy state. Assuming that the crystal's symmetry means that the probability the creation or annihilation of a phonon mode with momentum of either $\pm\hbar\mathbf{Q}$ is the same, then the only difference in the scattering function must come from the different occupancies of the energy levels of the crystal when it is at thermal equilibrium. These occupancy probabilities are governed by the Boltzmann distribution (phonons have no spin and are therefore Bosonic particles) meaning that the ratio of the probabilities of the crystal being in one state or a higher state where the difference in energy between the two states is $\hbar\omega$ is $\exp(-\beta\hbar\omega)$, i.e., the probability that the excited state is occupied is this factor smaller than the probability that the lower energy state within the system is occupied. This leads to the following relation between the scattering functions with the system losing or gain energy:

$$S(\mathbf{Q}, -\omega) = \exp(-\beta\hbar\omega) S(\mathbf{Q}, \omega), \tag{1.14}$$

where the left and right hand side of the equation represent the system losing energy and gaining energy respectively. The result of this is that in inelastic neutron scattering experiments the broadened peaks appearing in a constant-$\mathbf{Q}$ scan at $\pm\hbar\omega$ will differ in their peak height by the factor in eq 1.14. This relation will be used in this thesis when analysing the inelastic neutron scattering data collected from the constant-$\mathbf{Q}$ scans when investigating $\mathrm{Fe}_{1+y}\mathrm{Te}$ single crystals.

1.2.3 Phonon scattering by neutrons

This thesis focuses a lot on the lattice dynamical behaviour of condensed matter and therefore it would be remiss to not explain how neutrons interact with phonons within these systems and what information about their dynamics can be extracted from the different types of neutron scattering that have already been discussed.

What are phonons? They are quantised particles representing the collective excitation of atoms which are arranged in a periodic structure. The name *phonon* comes from the Greek word *phone* for sound or voice, which is apt because long-wavelength longitudinal acoustic phonons are responsible for the mediation of sound. Depending on the collective motion of the atoms within a lattice the phonons can be either longitudinal or transverse (or a combination of the two as will be discussed in due course) and acoustic or optical. The difference between longitudinal and transverse phonons is that the displacement of atoms from their equilibrium positions within the lattice is along the direction of propagation of the wave or perpendicular to the direction of propagation. The direction of propagation is given by the direction of the wavevector, $\mathbf{k}$, of the collective motion with its magnitude being equal to $\frac{2\pi}{\lambda}$. Acoustic phonons are when all the atoms are moving in phase with one another whilst optical phonons label the collective motion described by atoms moving out of phase with each other.

Neutrons interact with phonons principally through the strong nuclear force with the moving nuclei within the excited periodic structure. It is possible for neutrons to extract information about the lattice dynamical behaviour by measuring the magnetic inelastic scattering due to the coupling between the phonons and local magnetic moments or electron spin densities; however, it is often much more simple and effective to directly measure the inelastic neutron scattering due to the strong nuclear interaction. This section will therefore give a brief outline of the strong nuclear force interaction potential and any formula that will be useful for understanding the inelastic neutron scattering experiments looking at phonons.

As was mentioned in passing in Sec. 1.2, due to the short range of the strong nuclear force (on the order of 1fm) compared to the typical wavelengths of neutrons in thermal neutron scattering experiments, which are on the order of 1Å, the nucleus appears to the incident neutron as a point particle. Without loss of understanding, it is therefore possible to ignore the fact that the exact behaviour of the strong nuclear force is currently unknown and make the assumption that the interaction potential for these interactions is proportional to a Dirac delta function centred at the nucleus:

$$V(\mathbf{r}) = \frac{2\pi\hbar^2}{m_n} b\delta(\mathbf{r}), \tag{1.15}$$

where the pre-factor ensures agreement with the known elastic scattering cross section for an isolated bound nucleus (i.e., $\frac{d\sigma}{d\Omega} = b^2$).

For a periodic arrangement of atoms within condensed matter, the interaction potential will be a sum over all the atoms within the sample—this would be a sum over all the atoms in the basis and all directions within the lattice. When dealing with phonons the atoms are being displaced meaning that the position vectors within eq. 1.15 are time-dependent. When a neutron interacts with the lattice that is undergoing collective motion it can interact inelastically and either create or annihilate one or more phonon modes. Due to the lowering of the degrees of freedom when more phonons are created or destroyed in the scattering event (see the diagram in Fig.1.4 that shows the creation of two

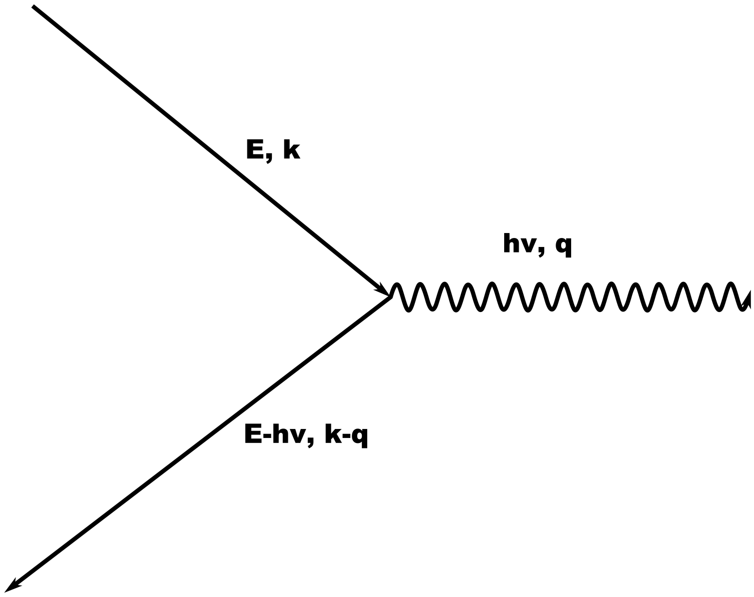


FIGURE 1.4: Diagram of neutrons scattering and producing a phonon with the constraint that the energy and momentum imparted to the phonon balances the loss of both the energy and momentum of the incident neutron.

phonons) one-phonon events are much more likely to happen. Without proving the result, since it is not a prerequisite for understanding the experimental neutron scattering data within this thesis, suffice it to know that the one-phonon coherent and incoherent scattering have the following double differential cross sections:

$$\begin{aligned}(\frac{d^2\sigma}{d\Omega dE_f})_{1ph,coh} =& \frac{k_f}{k_i}\frac{(2\pi)^3}{v_0}\sum_{\mathbf{G}}\sum_{\mathbf{s}}|\sum_{d}\frac{\bar{b}_d}{\sqrt{m_d}}\mathbf{Q}\cdot\mathbf{e}_{d\mathbf{s}}\exp(-W_d)\exp(i\mathbf{Q}\cdot\mathbf{d})|^2\\ &\frac{1}{2\omega_s}\big(n(\omega_{\mathbf{s}})\delta(\mathbf{Q}+\mathbf{Q}-\mathbf{G})\delta(\omega+\omega_s)\\ &+\{n(\omega_{\mathbf{s}})+1\}\delta(\mathbf{Q}-\mathbf{Q}-\mathbf{G})\delta(\omega-\omega_s)\big);\end{aligned} \tag{1.16}$$

$$\begin{aligned}(\frac{d^2\sigma}{d\Omega dE_f})_{1ph,inc} =& \frac{k_f}{k_i}\sum_{\mathbf{s}}\sum_{d}\frac{(\sigma_{inc})_d}{4\pi}\frac{1}{m_d}|\mathbf{Q}\cdot\mathbf{e}_{d\mathbf{s}}|^2\exp(-2W_d)\\ &\frac{1}{2\omega_s}\big(n(\omega_{\mathbf{s}})\delta(\omega+\omega_s)\\ &+\{n(\omega_{\mathbf{s}})+1\}\delta(\omega-\omega_s)\big);\end{aligned} \tag{1.17}$$

where v_0 is the volume of the unit cell of the lattice, $\mathbf{G}$ are the reciprocal lattice vectors, $\mathbf{s}$ are the phonon modes, d are the equilibrium positions of the atoms within each unit cell, $\mathbf{e}_{\mathbf{s}d}$ are the atomic displacement vectors for atom d of mode $\mathbf{s}$ (the atomic displacements will differ for longitudinal and transverse phonon modes and whether they are acoustic or optical phonon modes), $n(\omega_s)$ is the Bose-Einstein distribution for bosonic particles with an energy of $\omega_\mathbf{s}$ (i.e., particles not obeying the Pauli exclusion principle), and W_d is the Debye-Waller exponent that accounts for thermal motion of the particles. More information about the Debye-Waller factor and how it might affect the magnitude and isotropy of the neutron scattering can be realised by writing its exponent as an expansion of creation and annihilation operators and independently averaging over vibration modes, as is done by [8]:

$$2W = \frac{\hbar}{2MN} \sum_{\mathbf{s}} \frac{(\mathbf{Q} \cdot \mathbf{e}_{d\mathbf{s}})^2}{\omega_\mathbf{s}} \coth{(\frac{1}{2}\hbar\omega_\mathbf{s}\beta)}. \tag{1.18}$$

It ought to be noted that even at absolute zero temperature the Debye-Waller factor is finite due to the zero point energy fluctuations owing to the Heisenberg uncertainty principle. This states that the position and velocity of the particle can never be simultaneously known to absolute precision and therefore it is possible to have random fluctuations of the kinetic and potential energy of the crystal even at absolute zero. At very high temperatures, the average magnitude of the vibrational displacements of the atoms increases. This results in a much larger exponent that makes the Debye-Waller factor, $\exp{(-2W)}$, tend towards one. This means that even if the crystal creates an anisotropy in the scattering due to the Debye-Waller factor its effect reduces at greater temperatures. This anisotropy comes from the fact that modes along different directions in the crystal may have different atomic displacements owing to the symmetries of crystal structure. This would not be the case, for instance, in a cubic crystal structure because distortions of the crystal in all three directions of the lattice are symmetric. This is the case for ordered perovskites such as $SrTiO_3$.

The Dirac delta functions relate to the conservation requirements in the one-phonon scattering including both physical constraints that the energy and momentum of the system are conserved during the interaction. Both the coherent and incoherent one-phonon scattering include the possibility for the creation of destruction of a phonon as indicated by the change of signs in the two formulae. In a real inelastic neutron scattering experiment the intensity peaks are broadened around a central peak value. Why might this be happening? There are a several reasons for this broadening with an in-exhaustive list of possible causes being the resolution function of the neutron scattering instrument (which is a function of both E and $\mathbf{Q}$), the mosaicity of the crystal, which indicates that there is a spread of the axes around a mean orientation, and most interestingly that the interatomic forces between the atoms are anharmonic. The anharmonic forces might be due to the scattering of the phonons by localised or conduction electrons or phonon-phonon scattering to name but a few.

The reader should be aware that inelastic neutron scattering is only sensitive to phonons when there is a component of the scattering vector, $\mathbf{Q}$, along the displacement vectors of the atoms within the lattice, $\mathbf{e}_{ds}$, which is due to the dot product in the two equations. Hence, neutron scattering cannot probe phonon modes with displacements that are only perpendicular to the scattering vector. It should also be recognised that more information about the phonons can be obtained from the coherent inelastic scattering due to the presence of the phonon scattering structure factor in eq. 1.16 (this is the summation over $\mathbf{d}$), which is why the incoherent scattering is often seen as the background to the coherent scattering intensities in neutron scattering experiments. However, there is some information about the phonons that can still be extracted from the incoherent inelastic scattering: it can be shown that eq. 1.17 is proportional to the density of state of phonon modes, $g(\omega)$, which is helpful for determining the phonon energies and degree of phonon damping, which can be due to a multitude of reasons such as phonon-phonon interactions due to the anharmonicity of the lattice potential energies, electron-phonon coupling in metallic compounds, and scattering from magnetic excitations. Phonon dampening is a critical phenomenon that can tell you lots of information about condensed matter and the reader should be aware of this when reading, later in this thesis, about the softening of phonon modes in one of the studied compounds, $\mathrm{Fe}_{1+y}\mathrm{Te}$.

1.2.4 Transverse and longitudinal phonon modes

It has already been stated that phonon modes can be categorised depending on how those atoms within the crystal that are involved in the collective excitation are displaced, and specifically the direction that they are displaced in compared with the direction of the wave propagation. In summary, in the case of longitudinal modes the atoms are displaced perpendicular to the plane of propagation whereas in transverse modes they are displaced parallel to the plane of propagation (parallel and perpendicular to the wavevector respectively). In reality, the propagation of these collective waves is not so simple and waves cannot often be entirely classified as one or the other type meaning that a phonon mode has atomic displacements with components parallel and perpendicular to the plane of propagation of the wave. The classification of phonon modes, like linear elastic waves in continuous media, needs to be split into further sub-categories such that waves are either pure or quasi-longitudinal or transverse. In the case of a pure longitudinal or transverse mode, as the name suggests, the atomic displacements have components solely perpendicular or parallel to the plane of wave propagation. Quasi-longitudinal or quasi-transverse modes mean that the components of the atomic displacements are to a large extent either perpendicular or parallel to the plane of wave propagation. In most materials, phonon modes will not be pure modes due to the anisotropy of the crystal and in particular along the direction of the propagation of the measured wave. However, there are a few exceptions to this when one pure longitudinal and two pure (and degenerate) transverse modes are observed: this is when the crystal is isotropic in general or the phonon modes are being measured along one of the so-called high symmetry directions of the crystal, which are directions along which the propagation of the wave is isotropic. Certain labels are often used in the field to denote the pure and quasi-transverse modes: T1 is used to denote a pure transverse mode and T2 is used to denote a quasi-transverse mode. In general, there are always three normal modes in any direction but observing a splitting of the degenerate transverse modes gives information about the symmetry of the crystal.

If there are N atoms in the unit cell of a crystal, then there will be a total of 3N phonon modes. Hence, along one of the high symmetry directions of the crystal there will be a total of 2N degenerate phonon modes because N of these will be pure longitudinal modes and 2N will be doubly degenerate pure transverse modes. In this thesis, the $\mathrm{PbMg_{1/3}Nb_{2/3}O_3}$ and $\mathrm{Fe_{1+y}Te}$ crystals are studied. $\mathrm{PbMg_{1/3}Nb_{2/3}O_3}$ has a cubic lattice and therefore has three equivalent [100] symmetry directions, four equivalent [111] symmetry directions and six equivalent $[1\bar{1}0]$ symmetry directions. On the other hand, since $\mathrm{Fe_{1+y}Te}$ has a tetragonal lattice, it has one [100] symmetry direction, two equivalent [100] symmetry directions and two equivalent $[1\bar{1}0]$ symmetry directions [11]. Hence, in order to measure the pure longitudinal and pure transverse phonon modes within these two systems it is necessary to choose the appropriate axis along which to measure the dispersion curves.

1.3 Inelastic neutron scattering instruments

In the previous section, the theory of neutron scattering was considered with an eye to the application of neutrons for studying lattice dynamics and magnetism within condensed matter. In particular, the reader should now have a basic understanding of how neutrons can be used to investigate phonons in condensed matter and how the coherent inelastic scattering can be used to characterise the phonon dispersion curves for either longitudinal or transverse and acoustic or optical modes. In this section, the reader will be introduced to the practical implementation of neutron scattering experiments such as the production of neutrons, the experimental apparatus and the detection of scattered neutrons in a phase space volume element defined by the points (E, $\mathbf{Q}$) and ($E + \delta E$, $\mathbf{Q} + \mathbf{Q} + \delta\vec{Q}$). A short, but more detailed, description of the MERLIN and EIGER instruments at ISIS and the Paul Scherrer Institut (PSI) will then be given since the experimental neutron scattering data contained within this thesis was obtained here.

1.3.1 Production of neutrons

The production of a sufficient flux of neutrons is critical for carrying out neutron scattering experiments. Historically, the method for generating neutrons for experimental purposes was by inducing nuclear fission in a nuclear reactor. In this process heavy nuclei are bombarded with neutrons and split into lighter nuclei with more neutrons given off as a product to the reaction. The reason why these neutrons are released is that lighter nuclei require a smaller fraction of their nucleons to be neutrons for stability and so in producing lighter nuclei more neutrons must be emitted. To be a self-sustaining chain reaction the facilities are designed such that on average one of these neutrons causes the fission of another nuclei. The neutrons that do not contribute to further fission reactions are allowed to escape from the nuclear reactor for use in neutron scattering experiments. An example of a nuclear fission reaction with $^{235}_{92}\mathrm{U}$ as the fissile material is shown below:

$$^{1}_{0}\mathrm{n} + {}^{235}_{92}\mathrm{U} \rightarrow {}^{141}_{56}\mathrm{Ba} + {}^{92}_{36}\mathrm{Kr} + 3{}^{1}_{0}\mathrm{n}, \tag{1.19}$$

which would mean that on average (assuming that this is the only allowed nuclear decay path for $^{235}_{92}\mathrm{U}$) two of these neutrons would be allowed to escape from the nuclear reactor and be used for neutron scattering experiments.

Unfortunately, the neutrons produced as products to the nuclear fission reaction have energies that are far in excess of those required for studying condensed matter. They have energies of approximately 1MeV rather than the energy scales seen in condensed matter, i.e., 1meV to 1eV. To obtain the latter energies these neutrons need to be cooled down, or moderated, by repeated collisions with particles within another medium. After several collisions within the so-called moderator the neutrons will have a Maxwellian distribution of energies equal to that of the moderator. The principle choices for nuclear moderators are light and heavy water (H_20 and D_20 respectively) and solid graphite rods. For this thesis, it is unnecessary to understand the reasons for choosing one over the other, but it is salient to understand that the temperature of the moderator is critical for selecting neutrons with energies on the scale of phenomena within a material that an experimentalist wishes to study. Neutrons are often categorised into three broad temperature ranges: thermal, epithermal and fast neutrons (see Fig. 1.5). Since phonon modes in condensed matter have energies less than 100meV, moderators that produce thermal neutrons are chosen for neutron scattering experiments.

An alternative source of neutrons that is increasingly replacing the nuclear reactor sources is a spallation source. The spallation of neutrons works by first accelerating protons to energies in the range of a few GeVs in synchrotrons, cyclotrons or linear accelerators and using them to bombard a heavy element target to cause the ejection of several neutrons. The noun *spall* comes from the mining industry and is used to describe the chips that flake off minerals when they are put under pressure or crushed. The similarity between this process in the mining sector and the use of protons to cause the ejection of neutrons is why the reaction is called neutron spallation. The accelerated protons cause the heavy nuclei to go into an energetically unstable excited state which quickly decays into lighter nuclei along with the ejection of several neutrons. These neutrons can be transported to the neutron scattering experimental instrumentation using beam tubes and guides.

What are the benefits of using nuclear fission or neutron spallation as a source? One of the reasons to choose nuclear fission as a source for neutrons is that the flux achieved by nuclear reactors is often higher than the flux that can be obtained from spallation sources. Higher neutron fluxes have the advantage that the neutron scattering scans of a sample can be achieved in less time and the statistics are much improved with coherent scattering peaks more clearly defined and statistically significant. The neutron flux from spallation sources

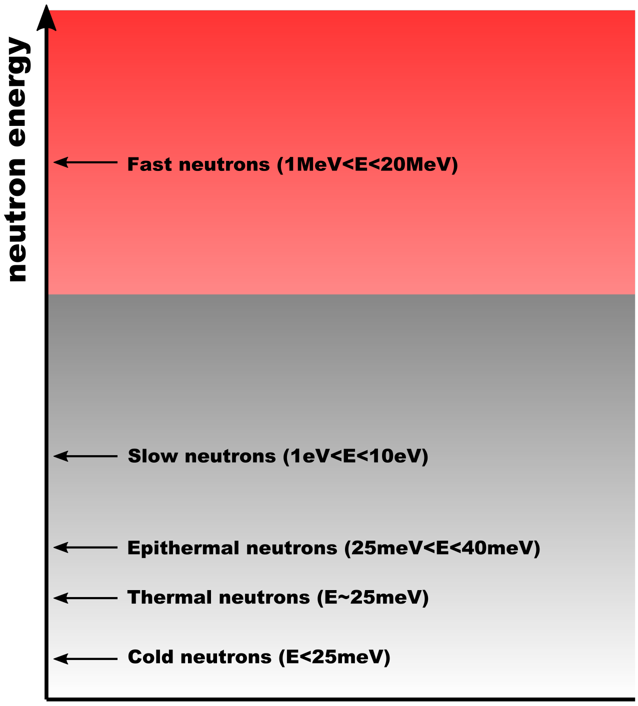


FIGURE 1.5: An illustrative guide to the terminology used for particular incident neutron energies within the field of neutron scattering. The neutron moderators allow the instrument to have an incident flux of neutrons within a particular energy range because of the thermal equilibrium reached within the moderator medium.

is commonly pulsed (e.g., the ISIS Neutron and Muon Spallation Source has a pulse frequency of 50Hz [12]) which helps in determining the incident neutrons' energies and wavelengths and therefore in carrying out time-of-flight measurements. The benefit of being able to do time-of-flight measurements can reduce the time needed to carry out experiments and mitigate the difference in neutron flux with nuclear reactor sources.

1.3.2 General features of neutron scattering instruments

The reader has now been introduced to the common types of neutron sources, i.e., the nuclear reactor and spallation sources. In this section there will follow a discussion of how the neutrons are used in practice for carrying out neutron scattering experiments and how instruments are set up to measure the intensities of scattered neutrons at specific energies and scattering vectors. There will be a review of the general features of both time-of-flight (ToF) and triple-axis spectrometers (TAS) because not only are these techniques used in this thesis but they are also the most common techniques used in all neutron scattering experiments. It is therefore critical for understanding the studies within this thesis that the reader has a sufficient degree of knowledge on these two innovative techniques.

In time-of-flight spectroscopy, the source of neutrons is trains of pulses and the neutron detectors form pixellated arrays in order to detect a large range of energies, $\hbar\omega$, and momentum transfers, $\mathbf{Q}$, of incoming neutrons. As such ToF spectrometers can survey large regions of ($\hbar\omega$, $\mathbf{Q}$) space and do so quickly. The time from when the neutrons are ejected from the heavy nuclei target to being detected by the pixellated array is used to determine the energy and momentum of either the incident (direct geometry ToF) or scattered (indirect geometry ToF) neutrons depending on the set up of the ToF apparatus. In either the direct or indirect geometry ToF there are two length scales measuring the distances between the neutron source and sample and the sample to the detector respectively. The time to traverse one of these distances is fixed by setting either the incident or scattered neutron energy using choppers or monochromator crystals. In common to both geometries a background chopper (often called a t_0 chopper) that is made from materials with high absorption properties is placed in front of a beam aperture at the moment of proton impact (this is the reason why it is named a t_0 chopper) to block out fast neutrons and gamma rays that would otherwise spoil the experiment by contributing to background radiation [13]. This is necessary because fast neutrons and gamma rays do not obey Bragg's law and will interact with the nuclei in the sample forming resonance peaks in the neutron cross section. The resonances occur because fast neutrons have enough energy for them to combine with target nuclei and form composite nuclei in an excited state. The resonance peaks in the neutron cross section occur above $\sim$eV meaning that it is essential for the t_0 choppers to be designed such that they block out neutrons with energies greater than this lower threshold [14]. In some designs of time-of-flight instruments a disc chopper is also used after the t_0 choppers to block out neutrons with energies higher than $\sim$100meV.

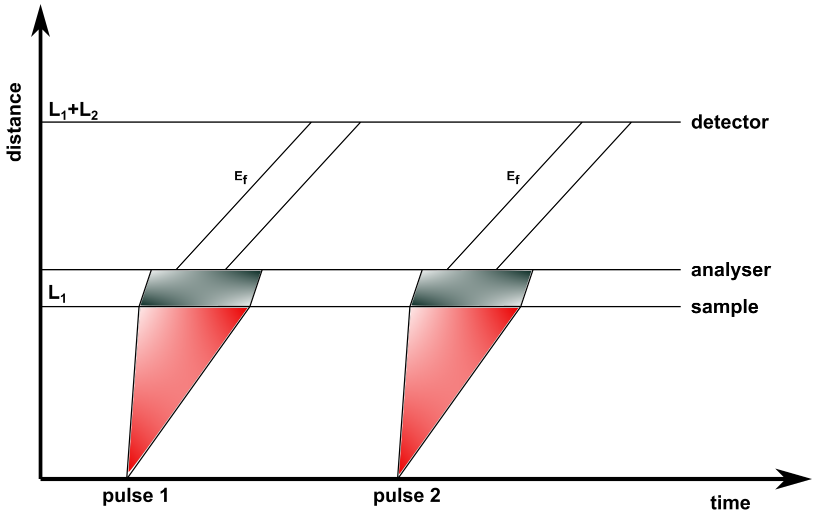


FIGURE 1.6: A schematic diagram of the indirect geometry inelastic neutron scattering instruments. Each pulse contains neutrons of varying wavelengths that scatter from the sample towards the analyser crystal, which selects neutrons of a particular wavelength from the Bragg reflections. Higher order Bragg reflections can be filtered out of the scattered beam using neutron filters.

As has already been mentioned, in direct geometry ToF the incident neutron momentum, k_i, and thus energy, is fixed. This is done using a Fermi chopper that rotates at integer multiples (the integer is called the repetition rate) of the source frequency so that only neutrons of particular energies can pass through to the sample and not be blocked from the neutron beam (see Fig. 1.6) [3]. The multi-repetition mode can also be achieved with disc choppers rather than Fermi choppers. The difference between these two types of choppers being that the former is a large, thick cylinder made from a material with a high neutron absorption cross section and has slits along the outer edges in order to allow neutrons to pass through it. This is in contrast to a Fermi chopper that has curved slits that run through the chopper from one side to the other. The repetition rate gives the number of different incident neutron energies that are permitted to pass through to the sample and scatter into the pixellated array of detectors. These incident energies need to be chosen in order to avoid neutrons from one pulse overlapping with neutrons in another pulse within the instrument, which would make time-of-flight measurements impossible. An alternative method for reducing the constraints on the energy choices for avoiding any overlapping pulses of neutrons is to further the distance between the source and sample with respect to the other distance between the sample and the detector array, which is thought about in the designing of ToF instruments before construction. For the experimentalist who is constrained by the instrument they are using, they must consider what combination of neutron energies will be allowed. A considerable advantage in using a higher repetition rate is that more data can be collected in less time as multiple experiments are in essence being carried out at the same time.

For an indirect geometry ToF (i.e., the configuration of a ToF neutron scattering instrument that fixes the final energy of the scattered neutrons using an analyser crystal) a neutron filter is placed colinear with the axis pointing from the analyser crystal, which Bragg diffracts the scattered neutrons from the sample, and the detectors. When the scattered neutrons Bragg diffract in the analyser crystal this selects neutrons with a wavelength of λ, but as has been stated previously, higher order Bragg scattering happens at $\lambda/2$, $\lambda/3$, $\lambda/4$, etc. An ideal neutron filter would therefore be constructed from materials with a high transmissibility of neutrons at a wavelength of λ but a low transmissibility, and therefore a high absorption and scattering cross section, at the higher order Bragg scattering. Depending on the cut-off requirements to suppress these higher energy neutrons various materials can be used as filters. For instance, on EIGER at the Paul Scherrer Institut (PSI), which is discussed in further detail in Sec. 1.3.4, the (002) Bragg reflection of pyrolytic graphite is used to filter higher energy neutrons out of the beam (see Fig. 1.9) [15]. Pyrolytic graphite is a material in which all the crystallites (also called the grains of a quasi-ordered crystal that meet at grain boundaries) have a high degree of orientation preference. Since it is a hexagonal crystal system it can be described in that the basal axes ($\mathbf{a}_1$, $\mathbf{a}_2$ and $\mathbf{a}_3$) are all aligned but the hexagonal c-axis have a random orientation but preferentially along one direction. This means that there is a spread about a preferred axis with a mosaicity that is defined by the distribution of the angular spread of the c-axes. Pyrolytic graphite is chosen as a neutron filter because a single crystal attenuates both the lower order and higher order Bragg scattered neutrons whereas pyrolytic graphite provides a sufficient degree of higher order filtration without a large reduction in the lower order neutrons.

As described in Sec. 1.2, in order to calculate the scattering probabilities it is necessary to normalise the scattered intensities by the incident neutron flux. To do this in neutron scattering experiments a monitor is placed before the neutrons scatter from the sample. This monitor is a detector that can measure the time-integrated neutron intensities because the signal that it gives is linearly proportional to the incident flux. The monitor will have its own resolution function because the probability that the neutrons will interact with the atoms within the monitor will be dependent on the incident energy of the neutrons. Correcting for the resolution of the monitor and the detectors placed after the sample (along with the other elements of the time-of-flight instrument) allows the user to determine the scattering function as a function of both the energy and momentum transfer and use this information to help understand the condensed matter system being studied.

In neutron scattering experiments the important information is contained within the scattering function, $S(\hbar\omega, \mathbf{Q})$, which is related to the double differential scattering cross section presented in Sec. 1.2.3 using the following relation:

$$\frac{d^2\sigma}{d\Omega d\hbar\omega} = \frac{k_f}{k_i} S(\hbar\omega, \mathbf{Q}). \tag{1.20}$$

However, in ToF spectroscopy the intensity is measured in the time domain (i.e., it is measuring the differential cross section per unit time) rather than directly being able to access the differential cross section per unit energy as shown in eq. 1.20. It can be demonstrated that for direct geometry ToF (a similar approach can be applied to indirect geometry ToF) the two differential cross sections can be related by the total time it takes for the neutrons to traverse from the target to the detector array, t_T, the time for the neutron to reach the sample from the source, t_s, the distance from the source to the sample, L_s, and the distance from the sample to the detectors, L_d:

$$\frac{d^2\sigma}{d\Omega d\hbar\omega} = \frac{(t_T - t_s)^3}{m_n L_d^2} \frac{d^2\sigma}{d\Omega dt_T}. \tag{1.21}$$

Removing the $\frac{k_f}{k_i}$ factor from the above expression, it can be demonstrated that the scattering function, $S(\mathbf{Q}, \omega)$, is proportional to the fourth power of the difference in the two times, t_T and t_s:

$$S(\hbar\omega, \mathbf{Q}) = \frac{L_s(t_T - t_s)^4}{t_s m_n L_d^3} \frac{d^2\sigma}{d\Omega dt_T}. \tag{1.22}$$

The same analysis can be done for the indirect ToF geometry, which gives a similar expression for converting the double differential cross section in the time domain to the scattering function:

$$S(\hbar\omega, \mathbf{Q}) = \frac{t_d(t_T - t_d)^2}{m_n L_s L_d} \frac{d^2\sigma}{d\Omega dt_T}. \tag{1.23}$$

ToF instruments at neutron scattering facilities, such as on EIGER at the PSI, often have software for the conversion of the double differential cross section in the time domain to the energy transfer domain or the scattering function. This will have to be done for each pixel of the neutron detectors because the length from the sample to the detector will be different for each scattering vector, $\mathbf{Q}$.

In a direct geometry ToF instrument the maximum neutron energy loss that can be observed (due to the energy-momentum conservation laws) is equal to the incident energy, whereas the maximum energy gain in an indirect geometry ToF instrument is equal to the final selected energy by an analyser crystal.

What are the advantages and disadvantages of having a direct or indirect geometry ToF instrumental set-up? One aspect to consider for the direct geometry ToF instruments is that the energy resolution of the instrument increases with increasing incident neutron energy. This is, however, not necessarily a problem because the incident energy can be chosen so that the resolution suits the excitations that one is observing. For instance, if there is an experiment measuring both low and high energy phonons, it is possible to have a repetition rate of $p = 2$ selecting one incident neutron energy for measuring the lower energy modes and another energy for measuring the higher energy modes. In this way the higher resolution when using the lower incident neutron energy allows for better characterisation of the lower energy phonon modes whilst still enabling the determination of the higher phonon modes with the higher incident neutron energy. In this thesis, this technique (using a repetition rate of

$$\begin{aligned} \mathbf{p} = \hbar\mathbf{Q} &= \hbar(\mathbf{k}_f - \mathbf{k}_f) \\ E &= \frac{\hbar^2}{2m_n}(\mathbf{k}_f^2 - \mathbf{k}_f^2) \end{aligned}$$

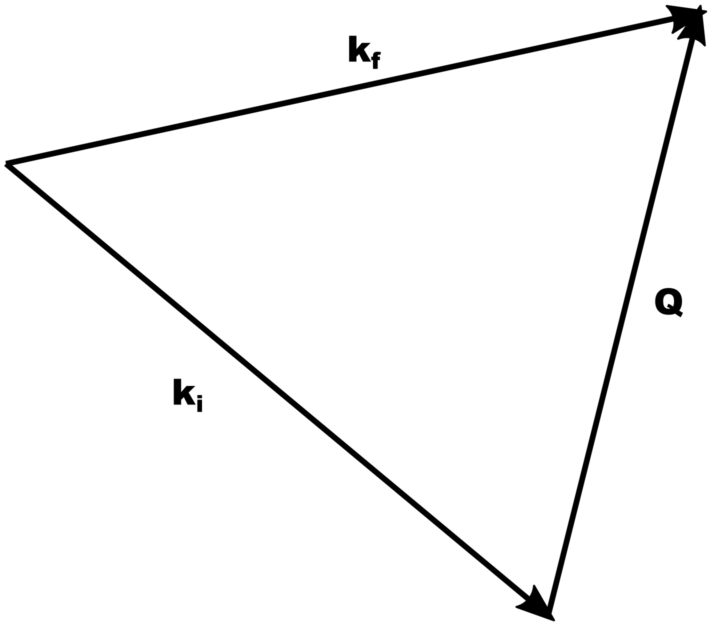


Figure 1.7: The energies and momenta of the incident and scattered neutrons are constrained by the conservation of energy and momentum for the scattering process.

$p = 2$ with incident neutron energies of E_i=75meV and E_i=150meV) could have been utilised in order to characterise the lower and higher energy phonon modes in $PbMg_{1/3}Nb_{2/3}O_3$, but, in the end, these experiments were undertaken separately. An issue with using the indirect geometry ToF instrument is that in selecting the final neutron energy after being scattered by a sample with an analyser crystal there will be spurious peaks associated with higher order Bragg scattering, i.e., $\frac{\lambda}{2}$, $\frac{\lambda}{3}$ etc. For this reason, a direct geometry ToF instrumental set-up is often preferred when weak scattering signals are present owing to the improved background scattering. The weak signal observed for the highest energy phonon modes, which is due to the inverse proportionality of the differential nuclear scattering cross section with phonon frequency, is one example of a phenomenon that would benefit from the use of a direct geometry ToF instrument.

The other fundamentally useful neutron scattering instrument is the triple-axis spectrometer (TAS). This is different from the ToF instrument because both the incident and final neutron energies are selected using a monochromator and analyser crystal, which is often made from a pyrolytic graphite crystal using the (002) Bragg reflection. The sample is placed on a goniometer which can rotate about an horizontal and vertical axis meaning that, along with the ability to independently set the two angles that select the incident and final neutron energies, the TAS can access any trajectory in the Brillouin zone and energy gain or loss by carefully setting these angles (see Fig. 1.7). The ability of the TAS to perform constant-energy or constant-$\mathbf{Q}$ scans (whereby the energy transfer or $\mathbf{Q}$-vector is fixed) is also very useful because scans can, for instance, be performed along high-symmetry directions to compare with theoretical calculations, e.g., density functional perturbation theory for phonon modes, that are often carried out along these directions.

Having introduced the reader to the practical implementation of neutron scattering for studying condensed matter, in the following sections there will be an explanation of the instruments used to carry out the studies within this thesis, namely the MERLIN and EIGER instruments at ISIS in Oxfordshire and the Paul Scherrer Institut (PSI) in Switzerland respectively.

1.3.3 MERLIN

The MERLIN instrument at ISIS is based on the time-of-flight direct geometry spectroscopy technique that was addressed in the previous section. It was utilised in this thesis for characterising the longitudinal and transverse higher energy phonon modes in $PbMg_{1/3}Nb_{2/3}O_3$ and for studying the displacement disorder of atoms within the perovskite structure; hence, it will be conducive for the reader to have some basic understanding about the MERLIN instrument before proceeding to the sections pertaining to the experiment and the inelastic neutron scattering data. The information that will be described here can be complemented by externally cited material [16].

The ISIS Neutron and Muon Source currently has two target stations (target station 1 and 2) that are separately providing neutrons and muons to several neutron scattering and muon spectroscopy instruments. The neutrons are produced when bunches of protons, which have been accelerated around a fixed-loop path within a magnetic field of a synchrotron, collide with a heavy tungsten target. The highly energetic collisions induce neutron spallation reactions providing a pulsed source of neutrons to the neutron scattering instruments within a particular target station. Before being channelled into the neutron scattering instruments these fast neutrons are moderated by one of four moderators depending on the neutron energies required by an experiment: one uses liquid hydrogen at 20K, two comprise water at room temperature and the other utilises methane at 100K [17]. To reduce the neutron flux loss through the walls of the moderators, a beryllium reflector is placed around the moderators to reflect neutrons back into the moderator so that they can continue scattering with the molecules within the moderator medium in order to achieve thermal

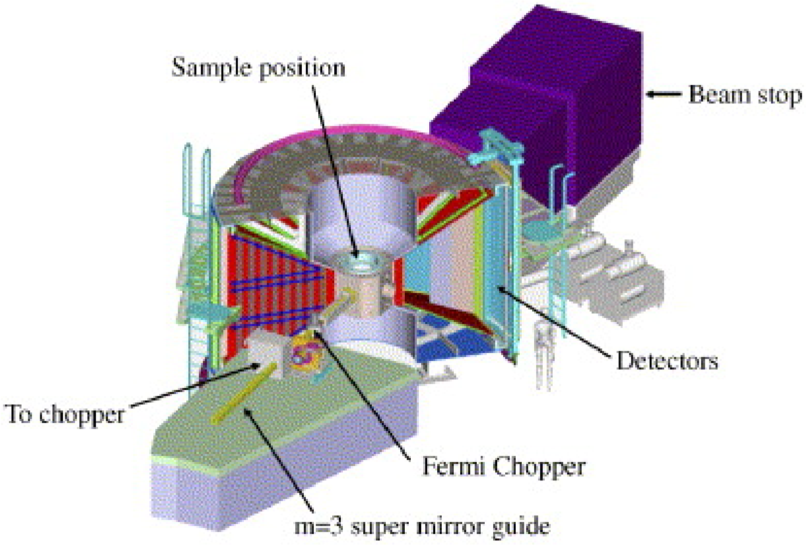


FIGURE 1.8: The MERLIN time-of-flight instrument on the ISIS Neutron and Muon Source with the sample environment seen with respect to other principle component parts of the direct geometry neutron scattering instrumental set-up [16].

equilibrium while maintaining sufficient neutron flux for neutron scattering experiments. The reflection of neutrons back into the moderator has the same theoretical principles as optical reflectivity but with different refractive indices for the incident neutrons that depend on their de Broglie wavelength and the real part of the neutron scattering length of the material used as the reflector. Beryllium is thus chosen as a reflector for the moderators due to its favourable neutron refractive index.

Using a combination of beam guides and neutron super-mirrors, which are multi-layered neutron mirrors with varying layer thicknesses to produce maximal neutron reflectance, the moderated neutrons are guided to one of the instruments on the target stations. MERLIN is located on target station 1 and was constructed to complement another spectrometer called MAPS—which is also located on the same target station in the ISIS Neutron and Muon Source. Being a direct geometry time-of-flight neutron scattering instrument, MERLIN fixes the incident neutron energies using a Fermi chopper that can be set to multi-repetition mode as was broached in the previous section (see Fig. 1.8). It has a detector array beyond the sample mount that spaces a scattering angle range of between -35° and 145° in the horizontal plane and ±30° in the vertical plane. The detector array comprises 3m long position sensitive detectors (PSDs) that are able to tag scattered neutrons by their position and time of detection (this enables the time-of-flight instrument to be in event mode whereby the experimentalist can choose the bin sizes rather than being pre-set for the instrument). The ability to access a large range of $(\omega, \mathbf{Q})$ space is one of the reasons for why the MERLIN instrument is chosen for studying complex condensed matter phenomena.

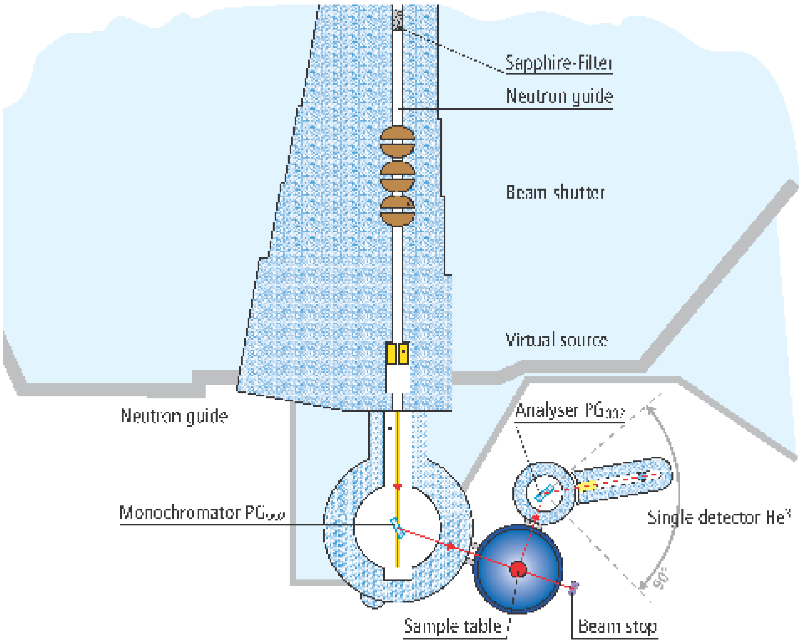


FIGURE 1.9: The EIGER triple-axis spectrometer on SINQ at the PSI has a sapphire filter, pyrolytic graphite monochromator and analyser crystals and ^{3}He-tube detectors [18].

1.3.4 EIGER

In contrast to the MERLIN instrument, the EIGER instrument at the PSI is a thermal neutron triple-axis spectrometer. It is located on the Swiss Spallation Neutron Source (SINQ) at the PSI that provides a steady-state source of neutrons from a neutron spallation reaction- the only one of its kind in the world. It has been used for this thesis for studying soft phonons in $Fe_{1+y}Te$ and for understanding the lattice dynamical behaviour close to the structural and magnetic phase transitions of the single crystal. In SINQ, a 1.5mA current of fast protons in a 600MeV beam is used to spall neutrons from a heavy lead target releasing on average about 10 neutrons from each collision event. In this way a total flux of 10^{17}/s neutrons is produced by the cyclotron accelerator putting the SINQ in the category of a medium neutron flux source. If the desire is to have thermal neutrons or liquid deuterium for cold neutron scattering experiments, the neutrons produced by the spallation reactions are moderated by heavy water (D_20). In a similar manner to the MERLIN instrument these moderated neutrons are channelled to the neutron scattering instruments using a system of neutron guides, which are glass conduits coated in a material with a high neutron reflectance (these neutron guides can be observed in Fig. 1.10 which shows the neutron guide to the EIGER instrument).

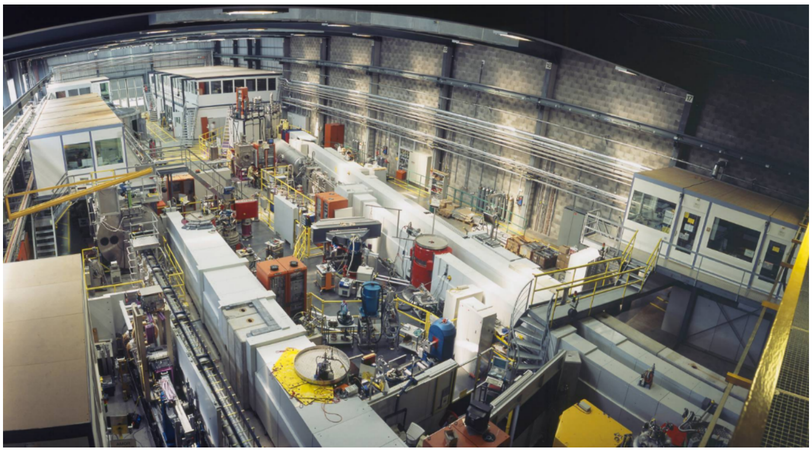


FIGURE 1.10: A panoramic view of the neutron guide hall in the Spallation Neutron Source, SINQ, from the viewing platform [19].

Once the thermal or cold neutrons have been transported along the neutron guides to the triple-axis spectrometer instrument, these pass through a fast neutron filter, which is made from sapphire crystals, i.e., Al_2O_3, as shown in Fig. 1.9. These fast neutron filters work because the crystals have a wavelength-dependent transparency to neutrons such that fast neutrons have a much greater probability of being absorbed or scattered out of the beam than neutrons with slower neutrons. Since there is a conflict between the desire to reduce the background to the scattering intensities and to maintain a high incident neutron flux, a balance is needed when introducing fast neutron filters to a neutron scattering instrument. After passing through the fast neutron filter, the neutron beam Bragg reflects from the monochromator crystal that selects the incident neutron momentum, $\mathbf{k}_i$. Higher-order Bragg reflections also mean that faster neutrons with wavelengths of $\frac{\lambda}{n}\ \forall\, n \geq 2$ scatter in the same direction towards the sample, which is positioned in the goniometer, thus contributing to the background. The neutrons scatter from the sample and those with a final momentum of $\mathbf{k}_f$ are selected by an analyser crystal (both monochromator crystals in the EIGER triple-axis spectrometer are made from pyrolytic graphite and use the (002) Bragg reflection) and diffract towards neutron detectors made from tubes filled with ^{3}He.

How do these ^{3}He-tube detectors function? ^{3}He has a high neutron absorption cross section with this nuclear reaction producing the charged by-products of a proton, ^{1}H, and a tritium ion, ^{3}H:

$$ {}^1_0\mathrm{n} + {}^3\mathrm{He} \rightarrow {}^3\mathrm{H} + {}^1\mathrm{H} + 0.764 MeV, \tag{1.24} $$

with the nuclear energy released in this reaction being shared between the reaction products. A voltage is applied across the ^{3}He-tube detector and the charged particles move towards a stopping gas in proportional counters with a charged cloud inducing a current that is proportional to the number of detected neutrons (assuming that the counter is in the linear proportionality regime of the ^{3}He-tube detector).

1.4 What has been studied?

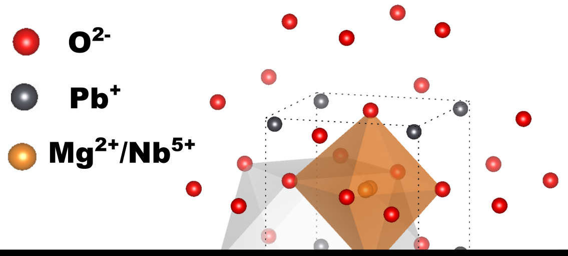
In this section, the reader will be introduced to the solid-state crystals studied for this thesis, namely $PbMg_{1/3}Nb_{2/3}O_3$ and $Fe_{1+x}Te$. Their structure, and in the case of $Fe_{1+x}Te$, magnetic phases will be presented from previous scientific studies and review articles in order for the reader to have an appreciation for the context with which these crystals are to be studied for this thesis. Although this is not a thesis studying their industrial applicability (for instance in the solid-state electronic sector) it will nonetheless be mentioned for the general interest. This is because the objective of condensed matter research is ultimately to understand how solid-state compounds can be used to produce better and more efficient devices and improve the total factor productivity of society.

1.4.1 Brief introduction to $PbMg_{1/3}Nb_{2/3}O_3$

$PbMg_{1/3}Nb_{2/3}O_3$ (PMN)—and the related compound $PbZn_{1/3}Nb_{2/3}O_3$ (PZN)—is a prototypical relaxor ferroelectric (the unit cell in the cubic phase is shown in Fig. 1.11) that displays temperature and frequency broadened peaks in the dielectric response [20, 21]. Relaxor ferroelectrics are a type of ferroelectric (materials that display a spontaneous polarisation that can be reversed in direction by the application of an external electric field) that possesses the property of high dielectric constants and large electrostriction. Having large electrostriction means that these materials can change shape on the application of an external field, which, since they are ferroelectrics, can also simultaneously act to change their polarisation. Electrostriction is different to piezoelectricity in that the strains of the crystal lattice in the former phenomenon change with the square of the applied electric field as opposed to linear changes with the latter. PMN is specifically a lead-based relaxor ferroelectric with a simple perovskite structure—it has a general chemical formula, $PbBO_3$, where the B site is occupied by two different cations with differing valences [22–25].

Why might a relaxor ferroelectric be an interesting material to study? These materials display exceptional dielectric and electromechanical properties [26] making them of particular interest for industrial applications. Some unusual opportunities exist in piezoelectric devices made from relaxor ferroelectrics such as the possibility to switch the device on and off by the application of a bias voltage. In general, the large response of the crystal to the applied electric fields means that devices made from relaxor ferroelectrics can have a large degree of electromechanical control. The electromechanical properties can further be enhanced by adding small amounts of other compounds to a pure relaxor ceramic. For instance, when lead titanate, PT, is added to PMN making the PMN-PT system it produces a large degree of electrostrictive strains close to room temperature, which is not observed for pure perovskite ceramics [27].

In the absence of an applied electric field [29], PMN and PZN do not undergo spatially long-range structural transitions [30–32] and instead only show short-range polar [33, 34] order characterized by momentum broadened diffuse scattering [35] near the Bragg peaks at low temperatures [36–39]. Unlike typical ferroelectric transitions that exhibit a single critical temperature in the dielectric response, the temperature dependence is governed by two distinct



[redacted]

FIGURE 1.11: Proposed [28] crystal structure of $PbMg_{1/3}Nb_{2/3}O_3$ (Pm-3m, space group number 221) showing the Mg^{2+}/Nb^{5+} cations at the centre of the cubic unit cell (a=b=c=4.04Å, α, β, γ=90°), the Pb^+ cation positioned on each of the corners and the O^{2-} anions located on the face-centres.

temperature scales. A high temperature T_d (referred to as as the Burns temperature [40, 41]) defines where the diffuse scattering is onset and where polar nano-regions are formed resulting in an elastic diffuse neutron cross section. A second lower temperature T_c defines where a ferroelectric phase can be induced with an electric field and where the diffuse scattering cross section can develop a corresponding anisotropy. These two temperature scales are reflected in the lattice dynamics and are particularly clear in high resolution neutron spin-echo measurements [42, 43] probing the dynamics associated with the diffuse scattering cross section. At high temperatures, where a transverse optic mode softens [44, 45], diffuse scattering appears with timescales on the scale of ~GHz. At a lower temperature T_c, where ferroelectric order can be induced with an electric field, the dynamics become static on the GHz frequency scale, while lower frequency measurements still observe some dynamics. The two temperature scales have been heuristically understood in terms of a random field model in analogy to model magnets [46–48].

Indeed, it may represent an experimental case of the Heisenberg model in the presence of a random field [49]. Over this region between T_d and T_c, the transverse optic phonon becomes energy broadened and overdamped with the optic mode appearing to precipitously collapse into the acoustic mode with energy [50–52]. The origin of this has been a matter of debate with it originally being suggested that dampening due to spatially localised polar nano-regions being the cause [53]. Indeed, these spatially localised polar nano-regions have a measurable response on the acoustic phonon linewidths and lifetimes [54, 55]. However, work on PMN doped with 60% PT [56] found nearly an identical lineshape for the low energy transverse optic mode despite PMN-60PT being a ferroelectric. This indicated that spatially localised regions of polar order was not the cause of the unusual energy broadening of the transverse optic mode near the Brillouin zone centre. Further work implicated symmetry allowed mode coupling [57, 58] as reported in other perovskites such as $KTaO_3$ [59], $BaTiO_3$ [60], and $SrTiO_3$ [61]. However, investigation of the temperature dependence of the acoustic phonon structure factor reported no evidence of strong phonon coupling [62] in PMN. It has been recently proposed that the unusual phonon lifetime effects originates from a kinematically allowed phonon decay mechanism analogous to that reported in low dimensional quantum magnets [63].

The phonon dynamics are further complicated by the observation of new modes in Raman spectroscopy (see Ref. [64] for complete of discussion and origin). This combined with the unusual phonon response discussed above has led to an investigation of the higher energy lattice dynamics. Initial work applying neutron spectroscopy with a triple-axis found a complete lack of underdamped phonons in PMN above ∼25–30 meV [65]. This is in contrast to expectations based on measurements in $SrTiO_3$ and $PbTiO_3$. Raman scattering has since been applied to PMN and has observed a higher energy response analogous to $PbTiO_3$ [66–68], but no new modes were observed at low energies.

1.4.2 Brief introduction to $Fe_{1+y}Te$

In this section, the reader will be introduced to another solid-state compound that was studied for this thesis, namely the iron-based chalcogenide and parent-phase of iron-based superconductors, $Fe_{1+y}Te$. The structural and magnetic phase diagram is complicated by the fact that the interstitial iron concentration, y, which is thought to populate the vacuum space within the lattice structure, can alter the physics of the crystal and engender a change of either structural or magnetic phase at a specific temperature. Fig. 1.12, which have been based on neutron scattering, x-ray diffraction and electronic resistivity data [69, 70], shows the structural, magnetic and electronic phases of $Fe_{1+y}Te$ plotted in the $y - T$ plane, i.e., interstitial iron concentration vs temperature. At higher temperatures, e.g., around room temperature and pressure, $Fe_{1+y}Te$ has a paramagnetic state and is a layered material belonging to the tetragonal (space group P4/nmm) crystal system. Fig. 1.13 shows the high temperature phase of $Fe_{1+y}Te$ when its structure belongs to this tetragonal crystal system. On the other hand, as the temperature of the crystal is reduced there are structural, magnetic and electronic phase transitions that are closely related with the final phase depending on the interstitial iron concentration. If the interstitial iron concentration is below a critical concentration, y∼0.12, $Fe_{1+y}Te$ undergoes a phase transition to a monoclinic (space group $P2_1/m$) crystal system with a double striped bicollinear antiferromagnetic structure where the magnetic moments are aligned along the b-axis. Conversely, if the interstitial iron concentration is above this critical concentration, the crystal has a transition to an orthorhombic (space group Pmmn) crystal system with a helical magnetic structure. Furthermore, the low temperature phases differ in their electronic states because the former is metallic whereas the latter is a semiconductor as revealed by resistivity data.

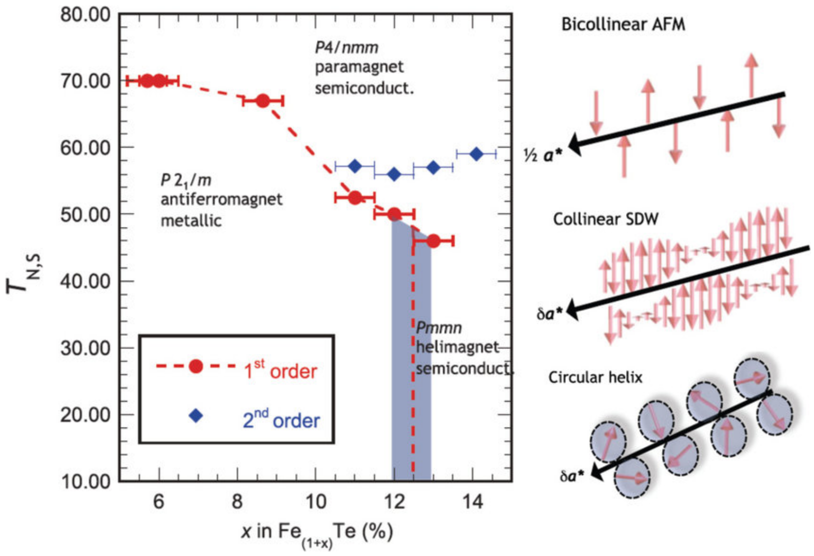


FIGURE 1.12: The structural, magnetic and electronic phase diagram of $Fe_{1+y}Te$ based on neutron scattering, x-ray diffraction and electronic resistivity data. A schematic of the magnetic structures is also shown for the antiferromagnetic and helical magnetic phases as well as the phase found in the region of phase space separating low-y and high-y [69, 70].

It is not known exactly how the interstitial iron atoms are positioned within the lattice and it is often assumed, when studying non-stoichiometric iron telluride, that these interstitial atoms are disorderly located throughout the space not occupied by the lattice formed by stoichiometric FeTe, i.e., when $y = 0$. In the tetragonal phase, a tetrahedron formed by the four nearest Te atoms form around each Fe atom in stoichiometric iron telluride such that the layers of Fe atoms form a plane parallel to both the a- and b-axes, as shown in Fig. 1.13.

Previous studies in the literature have tried to understand what is causing the emergence of the bicollinear double stripe antiferromagnetic order for the low-y configuration of $Fe_{1+y}Te$. For instance, one study has investigated whether the magnetic exchange between the Fe atoms in the layers of $Fe_{1+y}Te$ are being driven by the spin degrees of freedom [72]. The magnetic exchanges that are included within the Heisenberg spin model are those depicted in Fig. 1.14. There are two nearest neighbour exchange terms (J_{1a} and J_{1b}), two next nearest exchange terms (J_{2a} and J_{2b}) and one next-next nearest exchange term (J_3). Furthermore, there is an inter-layer coupling term (J_z) for the magnetic exchange between layers of $Fe_{1+y}Te$ with their normal axis parallel to the c-axis of the single crystal.

This spin-only model of $Fe_{1+y}Te$ below the critical interstitial iron concentration was tested on a sample of $Fe_{1.05}Te$ with an experiment involving the use of inelastic neutron scattering [73]. A Holstein-Primakoff transformation was performed of the Heisenberg coupling terms on the Hamiltonian to obtain a model of the spin wave dispersion. The model of the spin waves was then fitted to the neutron scattering data and the exchange parameters were optimised such that $J_{1a} = -17.5 \pm 5.7eV$, $J_{1b} = -51.0 \pm 3.4eV$, $J_{2a} = J_{2b} = 21.7 \pm 3.5eV$

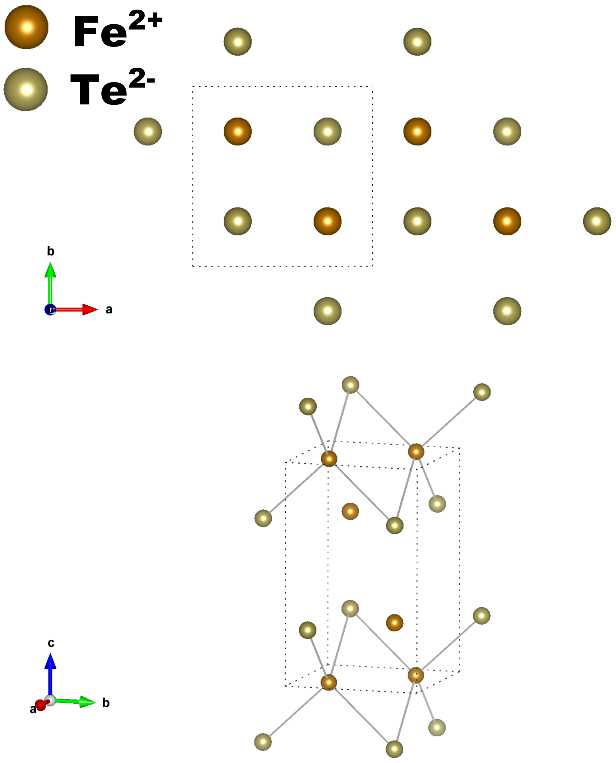


FIGURE 1.13: $Fe_{1+y}Te$ in the higher temperature tetragonal phase viewed in the a-b plane and in an orientation such that the layers of Fe atoms are easily visible. The atomic positions and lattice parameters associated with this crystal structure were determined experimentally by refining X-ray diffraction data [71].

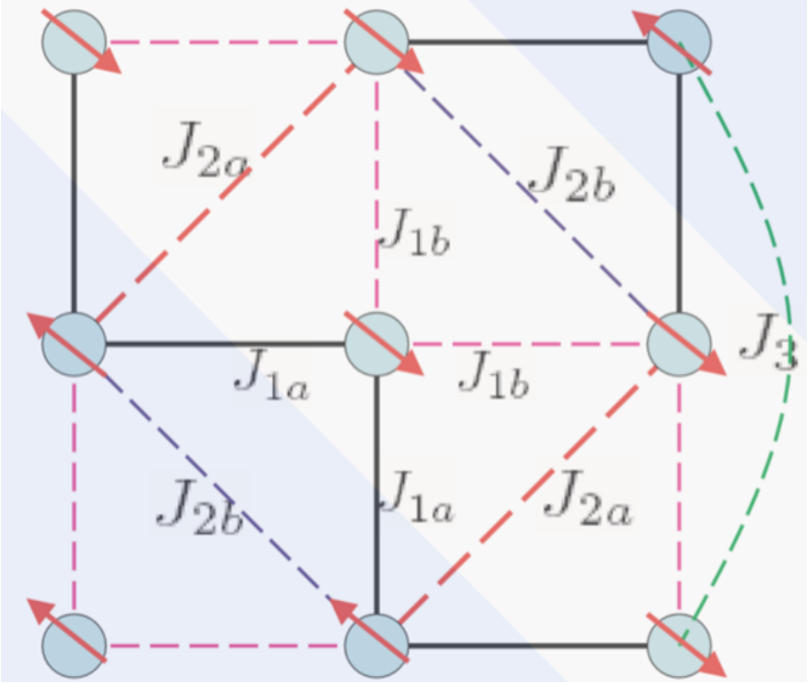


FIGURE 1.14: The Heisenberg model of spins in $Fe_{1+y}Te$ have been modelled with nearest-neighbour, next-nearest neighbour and next-next-nearest neighbour exchange coupling terms for the intra-layer magnetic interactions [72].

and $J_z = 6.8 \pm 2.8eV$ (assuming S=1). By comparing their findings with a related iron pnictide, $CaFe_2As_2$, they observe that although their nearest-neighbour exchange couplings are very different their next-nearest neighbour exchange couplings are remarkably similar. They suggest that the iron pnictides and chalcogenides' superconductivity has a magnetic origin that is very much connected to the next-nearest neighbour exchange couplings between the iron moments.

A further theoretical work focused on the alternative source of the magnetic order in that the magnetic interactions could be driven by the orbital degrees of freedom in the 3d atomic orbitals of the Fe atoms [74] and the interplay between structural distortions in the tetragonal crystal structure and the emergence of the magnetic order. It can explain the commensurate, i.e., that the periodicity of the spin orientations matches the lattice sites, $(\frac{\pi}{2}, \frac{\pi}{2})$ magnetic order of the $Fe_{1+y}Te$ layers by making several assumptions about the crystal. Firstly, the model assumes that FeTe is a correlated insulator with localised electronic charge around the Fe atoms such that their spin has a state of S=1. The model starts from the FeTe lattice having a perfect tetragonal structure such that, if it is visualised from above or below the plane of each Fe layer, the Te atoms are at the centre of four Fe atoms if projected onto that plane but alternate above and below the plane. If the basis vectors of the Fe atoms are $\hat{\mathbf{x}} \pm \hat{\mathbf{y}}$, and the d-orbitals are defined by $d_{Xz}/d_{Yz} = \frac{1}{\sqrt{2}}(d_{xz} \pm d_{yz})$, then the tetragonal crystal field formed by the Te tetrahedron lifts the d-orbital degeneracy such that only the d_{Xz}/d_{Yz} remain degenerate (they have the highest energy followed by, in order of descending energy, the d_{xy}, d_{z^2} and $d_{x^2-y^2}$ orbitals). The S=1 state

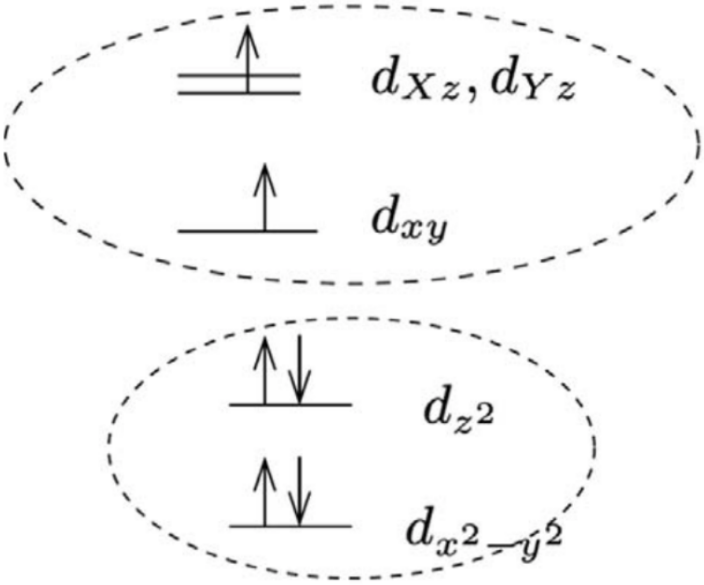


FIGURE 1.15: The d-orbital degeneracy is lifted by the tetragonal crystal field around each Fe atom in FeTe. The S=1 state is due to the dominance of Hund's coupling over the crystal field splitting between the d_{xy} and the degenerate d_{Xz} and d_{Yz} orbitals. [74].

arises because there are six electrons to fill the d-orbitals in each Fe atom and Hund's coupling dominates over the crystal field splitting between the d_{xy} and d_{Xz}/d_{Yz}, which leads to the d_{z^2} and $d_{x^2-y^2}$ being fully occupied and the d_{xy} and either one of d_{Xz}/d_{Yz} being partially occupied (see Fig. 1.15).

The single Fe spin state is frustrated due to the degeneracy of the d_{Xz}/d_{Yz} orbitals. However, this degeneracy is lifted by a Jahn-Teller distortion that reduces the symmetry and therefore splits these d-orbitals, hence making one of the orbitals energetically favourable. The exact geometric distortion resulting from the Jahn-Teller effect is challenging to predict, but the model assumes that the distortion is such as to realise the configuration that was observed experimentally, i.e., the Te atoms are alternatively displaced toward and away from Fe atoms along one axis and the Fe atoms along the other axis both move toward it forming a rhombus shape (the latter distortion is shown in Fig. 1.16).

But why does this orbital order lead to the bicollinear commensurate antiferromagnetic order in $Fe_{1+y}Te$? The theoretical explanation is that this is due to both the orbital ordering and the excess, i.e., y, of Fe atoms, and thus the presence of these electrons in the lattice. This is also what is believed to be the cause of the semi-metallic electronic order in $Fe_{1+y}Te$ [74]. The virtual electron hopping between orbitals is anisotropic, and due to the d_{Xz} orbital having a lower energy, it is energetically favourable to form one-dimensional chains of electronic charge along the projection of the d_{Xz}'s lobes. The next-nearest neighbour coupling favours an antiferromagnetic order of spins along each of these one-dimensional chains in the x-direction, whilst excess electrons

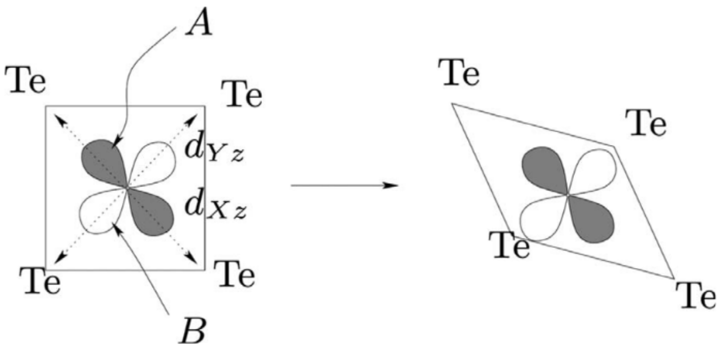


FIGURE 1.16: A Janh-Teller distortion lifts the degeneracy of the d_{Xz} and d_{Yz} making the former energetically favoured for a single Fe atom. [74].

donated from interstitial Fe atoms in the lattice dope the d_{Yz} and form one-dimensional metals that favours ferromagnetic ordering. The combination of these two effects produces the $(\frac{\pi}{2}, \frac{\pi}{2})$ magnetic order of the $Fe_{1+y}Te$. The predicted anisotropy in the next-nearest neighbour magnetic interactions does not agree with the experimental findings which instead found the nearest neighbour couplings to be anisotropic between orbitals [73].

Another interesting experimental finding is that there is competition between two configurations of magnetic order with the same magnetic order vector, i.e., $(\frac{\pi}{2}, \frac{\pi}{2})$ [75]. These two magnetic orders are the plaquette magnetic order and the bicollinear antiferromagnetic order with both having an overall C_2 rotational symmetry but the former has short range C_4 rotational symmetry owing to the ferromagnetic ordering within the plaquettes. At low inelastic neutron scattering energies, it was found that the magnetic spectrum has C_2 symmetry, but as it approaches an energy of E=26meV C_4 symmetry emerges. This can be explained by the possible frustration between the plaquette and the bicollinear antiferromagnetic order with the former winning out at energies above the aforementioned critical energy. It is interesting to note that this energy is suspiciously close to the assumed energy scale of the Jahn-Teller distortion in $Fe_{1+y}Te$. Unfortunately, there have not been any experimental studies investigating the energy scales associated with the Jahn-Teller distortion in $Fe_{1+y}Te$, but there has been a study that evaluated this energy to be E∼22meV in a similar iron-based compound called magnetite, $Fe^{2+}Fe_2^{3+}O_4$ [76].

In this thesis, it is therefore assumed that the Jahn-Teller distortion is the critical driver of the magnetic interactions within $Fe_{1+y}Te$. At low energies (much below E∼26meV), the Jahn-Teller effect dominates leading to the occupation of the d_{Xz} orbital in the Fe atoms with S=1 spin state. This is believed to be the source of the anisotropy in the magnetic interactions and as the energy increases more virtual exchange pathways are opened up owing to the greater occupation of the d_{Yz} orbitals. This orbital degree of freedom, along with the inclusion of the spin projection degrees of freedom in our model, is assumed to drive the magnetic ordering as will be explained in more detail in Sec. 5.

The phase transitions within $\mathrm{Fe}_{1+x}\mathrm{Te}$ can also be observed by neutron scattering because of the "softening" of phonon modes, which means that a phonon mode's energy within a small or broad range of $\mathbf{Q}$ values has been reduced. This happens because a change in real space alters the lattice structure or electronic charge order, which induces a change in the potentials around the atoms within a lattice. This leads to a different dynamical matrix and therefore to a different phonon dispersion for one or more modes depending on the symmetry changes. The softening of phonon modes can be categorised into those that cause the reduction in energy of the branch close to the zone centre (called zone centre soft phonons) and that which causes softening close to the zone boundary (called zone boundary soft phonons). As is self-evident in this terminology, the origin of the zone centre and zone boundary soft phonons differs and it is often difficult to understand precisely what is causing the softening, especially in complex solid-state compounds like $\mathrm{Fe}_{1+y}\mathrm{Te}$.

The acoustic transverse phonon in $\mathrm{Fe}_{1+y}\mathrm{Te}$ softens across a range of wavevectors close to the coincidental structural and magnetic phase transitions [77, 78]. Phonon softening in another iron-based system, $\mathrm{Ba}(\mathrm{Fe}_{1-y}\mathrm{Co}_y)_2 As_2$, has already been studied [79]. It looked at the anomalous softening of the transverse acoustic phonons in this solid-state system close to the zone centre and determined that nematic correlation lengths due to the electronic order within the crystal can account for the phonon softening rather than other possible explanations such as lattice instability or density waves. Lowering of the symmetry from four-fold to two-fold due to the electronic order can further induce the crystal to undergo a lattice distortion, hence why the two can be interlinked and often occur at the same or at close temperatures. The phonons within the system can couple to the electronic order and lead to decay pathways that dampen the phonons and concomitantly increase their lifetimes. The phonon-electron coupling can also be seen as introducing anharmonicity to the atomic potentials and thus leading to the phonon softening. This thesis provides some new experimental results from inelastic neutron scattering experiments that look to verify or refute the possibility that the phonon softening in this other iron-based system is being driven by the nematicity resulting from the orbital disorder.

1.4.3 Soft phonons and phase transitions

In Sec. 1.4.2, it was stated that phonon mode softening has been observed in iron-based systems close to phase transitions and that the latter induces the former because of changes in the potentials that go into the dynamical matrix that governs the physical equations for phonon dispersion. The origin of the phonon softening can come from many different phenomena such as lattice instabilities, changes to the electronic or even magnetic order. In effect any change to the internal forces between atoms within the crystal can change the lattice dynamics and collective excitations of the crystal, thus either softening or hardening the phonon modes and producing different phonon dispersions.

Phonons can also soften due to scattering interactions since pathways allowing for phonon decay lead to finite phonon lifetimes (this is measured as the inverse of the phonon linewidth, $\tau \sim \frac{1}{\Gamma}$) and a shift in the energy of the phonon peak from $E = \hbar\omega$ to $E = \hbar\omega - \Gamma$.

What theoretical framework is there to include the phenomenon of softening in more simplistic treatments of phonons? There are two frameworks for treating softening with one relying on a Landau phenomenological treatment of phase transitions and the other treating the phonons quantum mechanically and including higher order terms within a second quantisation approach. The mathematical beauty of the former approach is that all the fiendishly complicated physics of this many-body problem can be reduced to the fundamental symmetries of the system with the physics being condensed to the study of a free energy functional (called the Landau free energy) that is a function of so-called order parameters. This phenomenological theory is not therefore a microscopic theory and can only be applied to phase transitions associated with symmetry changes. The beauty is also in the fact that distant branches of physics can be governed by the same functional form due to similar symmetries in their free energies, which introduces the idea of universality.

The Landau phenomenological theory is governed by the observation that the stable state of a system is the order parameter(s) that minimise(s) the Landau free energy functional, $\mathcal{F}(\phi_i), i \in \{\phi_1, \phi_2, ..., \phi_n\}$. In the high-symmetry phase, which is usually the high temperature phase, the order parameter(s) is/are zero and the free energy can be decomposed into two parts:

$$\mathcal{F}(\phi) = \mathcal{F}' + \Delta\mathcal{F}(\phi), \tag{1.25}$$

where $\mathcal{F}'$ is a a constant that is not dependent on the order parameter and $\Delta\mathcal{F}(\phi)$ is a small perturbation of the free energy around the phase transition. Since the perturbation to the Landau free energy is small close to the phase transition, it can be Taylor expanded in powers of the order parameter around the point in phase space represented by $\phi = 0$:

$$\Delta\mathcal{F}(\phi) = -\phi H + \frac{a}{2}\phi^2 + \frac{b}{3}\phi^3 + \frac{c}{4}\phi^4 + O(\phi), \tag{1.26}$$

where H is an external field that interacts with the system. $\Delta\mathcal{F}(\phi)$ is invariant to all symmetry elements in the symmetry group of the high-symmetry phase, which often results in some of the powers in the order parameter not appearing in this functional.

The choice of a free energy in Landau's theory depends on which natural variables are specified. All thermodynamic potentials, ψ, are constructed from Legendre transformations of the internal energy:

$$\psi = U - \sum_i \frac{\partial U}{\partial x_i} x_i, \tag{1.27}$$

where $x_i \in \{V, P, T, N\}$, i.e., the volume, pressure, temperature and particle number of the system. If natural variables are specified, then the thermodynamic potential is a minimum when the system reaches equilibrium. If the pressure and temperature is specified, then the Landau free energy is of the form:

$$\mathcal{F} = U - TS, \tag{1.28}$$

and the equilibrium state is found by minimising this thermodynamic potential (the entropy, S, is given by the partial derivative of the internal energy with respect to the temperature, T, $\frac{\partial U}{\partial T}$). The internal energy of a crystal is the sum of the kinetic energies and potential energies, hence, for a non-magnetic crystal, this is the sum of the electrostatic potential between the ions within the lattice and the energy of the vibrational modes, i.e., the internal energy of the phonons in a second quantisation :

$$U = U_{eq} + \sum_{k} \sum_{s=1}^{3N_{cell}} \hbar\omega_{k,s}\left(\frac{1}{\exp\frac{\hbar\omega_{k,s}}{K_B T} - 1} + \frac{1}{2}\right) \tag{1.29}$$

where U_{eq} is the potential energy of the crystal when it is at equilibrium, the sum includes the wavevector, k, and the normal modes, s, and the parenthesis includes a contribution to the internal energy of the crystal owing to thermal excitations and another from the zero point energy. There are 3N$_{cell}$ independent normal modes where N$_{cell}$ is the number of atoms per unit cell. Moreover, there are N permitted k's where N is the number of unit cells in the lattice—this comes from the periodic boundary conditions imposed by the finite extent of the crystal.

The energy of the phonon modes, $\hbar\omega_{k,s}$, will depend on the forces intrinsic to the lattice structure, electronic and magnetic order of the crystal. For instance, the Coulombic forces on an ion will differ if there is a tetrahedron or octahedron of other ions surrounding it. It will also change if the valence electrons are filling one orbital rather than another, e.g., filling the d_{Xz} instead of the d$_{Yz}$ orbitals, and therefore creating a nematic order within the crystal that favours a certain electronic charge density. In order to include the possibility of changes to the order, be it a lattice distortion or an electronic order change, the energy of the phonon mode can be parameterised with order parameters. Taylor expanding the free energy about the point at which these are zero will allow for finding the ground state at a specific temperature and to predict any phase transition temperatures. As an example, if there is an arbitrary order parameter, ϕ, e.g., for the lattice distortion from a tetragonal to monoclinic phase, then both the the fixed atom potential energy, U_{eq}, and the internal energy of the phonons, $\omega_{k,s}$, would depend on this parameter. The internal energy could then be written as:

$$U(\phi) = U_{eq}(\phi) + \sum_{k}^{N} \sum_{s=1}^{3N_{cell}} \hbar\omega_{k,s}(\phi)\left(\frac{1}{\exp\frac{\hbar\omega_{k,s}(\phi)}{K_B T} - 1} + \frac{1}{2}\right). \tag{1.30}$$

The free energy functional in eq. 1.28 can then be Taylor expanded about $\phi = 0$ and minimised for a given temperature to find the phase with the minimum energy.

Eq. 1.29 assumes that the lattice vibrations are harmonic but in reality most crystals have non-negligible anharmonic forces that ought to be included. This is evident from inelastic neutron scattering because the linewidth of phonon modes is larger than the energy resolution of the instrument, which indicates the presence of scattering pathways for the phonons, i.e., phonon-phonon coupling, phonon-electron coupling, etc.

Doing this kind of theoretical analysis is beyond the scope of this thesis, but it is certainly useful to note that the softening of phonon modes could be induced by one or a combination of lattice distortions of a crystal, the emergence of electronic order or magnetic phase changes—all of which change the forces acting on the ions within the lattice and, hence, inducing changes to the lattice dynamical behaviour of the crystal.

Chapter 2

Crystal Synthesis

2.1 Introduction to single crystal and polycrystalline samples

In this chapter, the reader will be introduced to the process for synthesising polycrystalline and single crystal samples, which is essential for carrying out any inelastic neutron scattering experiment. This thesis comprises experimental studies of the phonon modes in $PbMg_{1/3}Nb_{2/3}O_3$ and non-stoichiometric $Fe_{1+y}Te$ meaning that sufficiently large single crystals were required for observing well-defined features in the neutron scattering intensities carried out on the TAS and ToF instruments respectively. Having grain boundaries in the sample (i.e., having lots of small crystallites form a polycrystalline sample) would have meant that the same phonons would propagate along different directions owing to the crystallite axes not being collinear. This would create difficulties in carrying out post-experiment analysis of the inelastic neutron scattering data, which would have made it difficult to associate parts of the Brillouin zone with features of the lattice dynamical behaviour, i.e., the collective excitation of the crystal that is quantised as phonons. The scattering length of nuclear scattering and magnetic scattering are approximately the same magnitude so in both cases there needs to be consideration for the size of single crystal that is synthesised if the purpose is to use them to study either of these two types of phenomena. For studying the lattice dynamical behaviour of $PbMg_{1/3}Nb_{2/3}O_3$, a single crystal sample with a mass of $\sim$192g was used in the inelastic neutron scattering experiments undertaken on MERLIN at ISIS (see Sec. 1.3.3 for an introduction to this neutron scattering instrument). This ceramic single crystal had previously been synthesised for use in other inelastic neutron scattering experiments and was of sufficient quality to be used for studying higher energy phonon modes ($E \gtrsim 40$meV) that have weaker scattering intensities owing to the reduced scattering cross sections at higher energies (the double differential cross section for one-phonon scattering drops off in proportion to $1/\omega$ as shown in Sec. 1.2.3).

For the studies that focused on the softening of the transverse acoustic phonon modes in $Fe_{1+y}Te$, single crystals were synthesised such that the amount, or concentration, of interstitial iron put the crystal in the monoclinic phase at low temperatures (the phase diagram for $Fe_{1+y}Te$ is shown in Fig. 1.12). As with $PbMg_{1/3}Nb_{2/3}O_3$, there is a requirement that the single crystal be sufficiently large in dimensions to make sure the scattering of phonons by neutrons is not weak. However, there is a greater level of difficulty owing to the requirements to be precise in the stoichiometry and to understand the difference between powder weights of the constituent reactants and the final interstitial concentration in the single crystal samples.

Although the single crystal samples were used for the inelastic neutron scattering experiments, both polycrystalline and single crystal samples play a critical role in the study of condensed matter. One of the advantages of polycrystalline samples is that they are much easier to produce in larger quantities, which is especially useful if the flux of the probe is low, and at higher phase purities. The crystallites within the polycrystalline sample all form from separate nucleation sites and grow into grains with crystal axes that are randomly oriented. The random orientation means that any scattering (e.g., from X-rays or neutrons) from an experimental probe is an average over all the possible orientations of the individual grains or crystallites. This is called "powder averaging", which, amongst other effects, such as grain boundary scattering, prevent the probe from studying the complex and emergent behaviour of the system if it were a whole single crystal formed from a single nucleation site. An example of emergent behaviour is the collective excitations in crystals that are studied in this thesis, i.e., phonons. If the grain size of the crystallites is reduced, the effect of grain boundaries on the phonons becomes non-negligible and it disrupts the collective motion of the normal modes of vibration of the crystal. This is the reason why studies of phonons using inelastic neutron or X-ray scattering require the use of large single crystal samples. Nonetheless, a lot of information can be obtained by using polycrystalline samples, such as the crystal structure, and they are often still used for checking the phase and purity of the samples before embarking on the more difficult task of synthesising a large single crystal sample.

Within the following sections of this chapter, a rudimentary introduction to the growth of single crystal samples will be presented. Furthermore, it will also detail the specific growing method for the synthesis of both polycrystalline and single crystal samples of $\mathrm{Fe}_{1+y}\mathrm{Te}$ (it will also mention how another perovskite, CsPbBr_3, that is similar to $\mathrm{PbMg}_{1/3}\mathrm{Nb}_{2/3}\mathrm{O}_3$ was synthesised) and some X-ray diffraction data produced for verifying the quality of the samples that were produced.

2.1.1 Basic theory of crystal growth

This section will provide a short introduction to the theory of crystal synthesis. It will help the reader to understand the procedure involved in the synthesis process and how different parameters affect the growth and quality of the samples produced. As discussed in Sec. 2.1, this is critically important because growing sufficiently large and pure single crystal samples is critical for being able to study the collective behaviour of complex systems that would otherwise be significantly affected by grain boundaries, impurities and other types of crystal defects.

The growth of single crystals starts from the embryonic emergence of nuclei formed when atoms or molecules stochastically self-assemble to form a larger structure. In a melt the constituent atoms and molecules have a large amount of kinetic energy. They undergo Brownian motion and thus regularly collide with other atoms and molecules. In this stochastic collision process, the atoms and molecules may form incipient crystal structures that either re-dissolve into the melt or even continue to grow via the accumulation of other colliding particles. Whether these nuclei re-dissolve or continue to grow in size depends on whether the gradient in the free energy favours the disintegration of the crystal structure or its growth. A very simple, but helpful, theoretical explanation for the physics involved in the stability of the incipient crystal structure can be explained by including the difference in free energy per unit volume when a liquid becomes a solid, ΔG_{l-s}, and the difference in free energy per unit area associated with the creation of an interface between the solid material and the surrounding liquid melt, ΔG_i. Assuming for simplicity that the incipient crystal structure is spherical the difference in free energy (whereby a positive value defines an increase with respect to the disaggregation of the structure) is:

$$\Delta G = \frac{4}{3}\pi r^3 \Delta G_{l-s} + 4\pi r^2 \Delta G_i, \tag{2.1}$$

where r is the radius of the incipient crystal structure. $\Delta G_{l-s} < 0$ because liquids release latent heat when they change phase to solids, whereas $\Delta G_i > 0$ since the surface tension is always a positive value. ΔG is plotted against r in Fig. 2.1. It shows that there is a critical value of the radius, r_c, beyond which it is energetically favourable for the incipient crystal to continue to grow in size. Below this critical value it is instead energetically favourable for the crystal to re-dissolve into the surrounding melt.

The critical radius, r_c, can be determined by finding the maximum value of ΔG and is shown to be the following:

$$r_c = -\frac{2\Delta G_i}{\Delta G_{l-s}}. \tag{2.2}$$

ΔG_{l-s} depends on the temperature of the surrounding melt: at the temperature of phase coexistence, i.e., the melting point, T_m, this is zero, but below this temperature it becomes negative. Although it is negative the melt will not immediately solidify if there is no “seed crystal” or a strong probability of nucleation. Slowly cooling the melt below T_m without causing it to solidify is called supercooling or undercooling and it is this effect that reduces (making it become more negative) the value of ΔG_{l-s}, and consequently reduces the critical radius for which an incipient nuclei is stable to re-desolving. For small amounts of undercooling, it can be shown that $\Delta G_{l-s} \sim -\frac{\Delta T \Delta H_f}{T_m}$, where $\Delta T = (T_m - T)$ is the magnitude of the undercooling, ΔH_f is the latent heat of fusion and T_m is the equilibrium phase transition temperature for melting assuming that there are no surfaces. Substituting this into both eq. 2.1 and eq. 2.2, their values at the maximal point become:

$$\Delta G_c = \frac{16\pi \Delta G_i T_m^2}{3\Delta H_f \Delta T^2}, \tag{2.3}$$

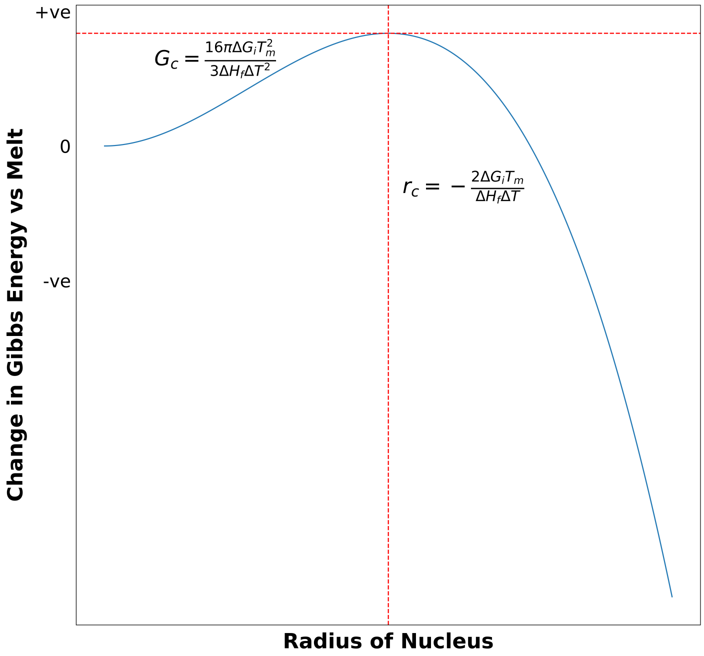


FIGURE 2.1: A plot of the difference in free energy, ΔG, between an emergent crystal structure and a melt composed of a stoichiometric mixture of atoms and molecules constituting the target single crystal.

$$r_c = \frac{2\Delta G_i T_m}{\Delta H_f \Delta T}. \tag{2.4}$$

This means that the more the melt is undercooled the smaller the critical radius, r_c, and the smaller the potential free energy barrier for the stochastic emergence of a stable incipient nuclei from which a crystal can grow. With a simple potential barrier assumption, the probability that nucleation will happen should be proportional to the Boltzmann distribution, $\exp\left(-\frac{16\pi\Delta G_i T_m^2}{3K_B \Delta H_f \Delta T^2} T\right)$, which emphasises the role played by the undercooling in the formation of nuclei within the melt.

The difference between the synthesis of single crystals and polycrystalline samples is that in the former it is desired that the concentration of nuclei that are formed is low and the growth rate of these into crystals is large. If the concentration of nuclei is large, then they will have little to grow before meeting other crystallites to form a plurality of grain boundaries within a polycrystalline sample. To reduce the rate of nucleation and encourage the formation of large single crystal samples, it is therefore preferential to reduce the degree of undercooling. This is achieved in the synthesis of $\mathrm{Fe}_{1+y}\mathrm{Te}$ by reducing the temperature in the furnace slowly as will be explained in more detail in the following sections.

The nucleation process is often aided by the presence of impurities within the melt, e.g., suspended dust particles, or by having other solids in contact with the liquid phase. It can further be helped by there being cracks and dislocations on those solid surfaces, for instance, on the surface of the ampoules being used to hold the melt. The desired result is that an irregularity of the surface at the base of the ampoule (e.g., through re-moulding the bottom using a hot flame) will increase the nucleation rate at this point in space and the growth rate of the crystal will be such that this single site of nucleation would produce a single crystal sample that grows linearly along the ampoule. This is the hoped for result, but it is often difficult to achieve owing to multiple real world experimental factors that influence the nucleation rate and induce the formation of polycrystalline samples with a plurality of individual grains or crystallites with boundaries between them. The reason why the nucleation rate is increased by introducing exogenous surfaces is that the potential barrier for the incipient nuclei to overcome and become stable to dissolution is reduced owing to a reduction in the peak free energy, ΔG_c.

How can the crystal growth be encouraged along the linear axis of the ampoule to achieve large single crystals? Since the interface between the solid incipient crystal and the melt is constantly changing, to achieve a constant growth of the crystal the temperature profile in the ampoule needs to move to keep apace with that boundary moving. On the melt side of that boundary undercooling of the melt ought to be maintained to ensure the continual favourability of the crystal to enlarge. This can be achieved by having a temperature gradient within the ampoule created by a source of heat and as the temperature is reduced the point along the ampoule where crystal growth is stimulated moves. The difficulty of this operation is that the temperature profile must not change too rapidly beyond the actual growing rates of the incipient crystal otherwise it would induce other nuclei to form and create polycrystalline samples with grain boundaries. In simple terms, however, a temperature gradient is an essential feature of the procedure for growing single crystal ingots. Some of the most common techniques include the Bridgman, Stockbarger and Czochralski methods. The first two have subtle differences in how the temperature gradient is maintained with the former achieving the temperature profile using the gradient present at the exit of a furnace, whilst the latter uses two coupled furnaces that maintain the temperature below the melting point on the side of the ampoule with the growing ingot and above the melting point on the other side of the ampoule closest to the melt. The advantage of the Bridgman technique is that it is much easier to set up, but at the same time because of this feature it is more difficult to control the temperature gradient and can achieve worse results for single crystal growth. The Czochralski method instead dips a mounted seed crystal into a melt with the seed crystal slowly being extracted vertically from the melt and simultaneously rotated about the vertical axis. A single crystal ingot grows from this seed crystal if the temperature gradient, angular velocity of rotation and extraction speed are carefully adjusted to encourage single crystal growth (Jan Czochralski, who serendipitously discovered the method when he dipped a pen into molten tin in a crucible rather than an inkwell and saw the formation of a solidified thin strip of metal attached to the nip, noted that the rate of withdrawal must be equal to the velocity of crystallisation which was different for different metallic substances [80]).

In this thesis, the horizontal Bridgman technique was used for growing the $\mathrm{Fe}_{1+y}\mathrm{Te}$ single crystals indicating that the temperature gradient, ∇T, was horizontal to the floor of the furnace. As was mentioned in the preceding short description, the temperature gradient was achieved by having a heating unit within the furnace acting as a source of heat and the door of the furnace acting as a sink of heat. In the following sections, the specific procedure for growing the $\mathrm{Fe}_{1+y}\mathrm{Te}$ single crystal and polycrystalline samples will be described with reference to specific instrumentation that was used in preparation of the samples to enable an experimentalist to repeat the procedure and grow their own samples for experimentation.

2.2 Preparation of the pre-melt stoichiometric mixture of $\mathrm{Fe}_{1+y}\mathrm{Te}$

The objective of the crystal synthesis was to prepare single crystal samples of non-stoichiometric $\mathrm{Fe}_{1+y}\mathrm{Te}$ with interstitial concentrations of Fe atoms, y, such that the low temperature phase of the crystal was monoclinic rather than orthorhombic. There is a critical interstitial concentration of Fe atoms below which the phase is monoclinic, which can be seen from Fig. 1.12 found in Sec. 1.4.2 of this thesis. Below $y \sim 0.12$ the phase is monoclinic at low temperatures meaning that to grow a single crystal of $\mathrm{Fe}_{1+y}\mathrm{Te}$ for studying the tetragonal–monoclinic phase transition the target interstitial concentration must be less than this amount. Hence, to ensure the growth of low-interstitial single crystals of $\mathrm{Fe}_{1+y}\mathrm{Te}$ a non-stoichiometric mixture of Fe and Te powders needs to be prepared. One issue is that the dry weight ratios may not exactly correspond to the final stoichiometry of the single crystals owing to a multitude of factors during the preparation and growing stages (e.g., the growth of phases other than $\mathrm{Fe}_{1+y}\mathrm{Te}$, such as $\mathrm{Fe}_2\mathrm{Te}_3$ etc.). The presence of other phases and the interstitial iron concentration can be determined with X-ray diffraction data, which will be discussed in subsequent sections.

The stoichiometric mixture of iron and tellurium powder was prepared in batches of up to 5g based on the assumption that the ratio of iron to tellurium in the dry powder was a reasonably close indicator of their final stoichiometries in the resultant single crystals of $\mathrm{Fe}_{1+y}\mathrm{Te}$. Depending on the target interstitial iron concentration, y^*, this meant that the mixture contained the following masses of iron, m_{Fe}, and tellurium, m_{Te}, powder given that their molar masses are 55.85u and 127.60u respectively:

$$m_{\mathrm{Fe}} = \frac{55.85 \times (1+y)}{55.85 \times (1+y) + 127.60} \times 5g, \tag{2.5}$$

$$m_{\mathrm{Te}} = \frac{126.70}{55.85 \times (1+y) + 127.60} \times 5g. \tag{2.6}$$

The stoichiometric mixture was then poured into a mortar and fastidiously ground into a fine, homogeneous powder using a pestle. As with all subsequent instruments and surfaces that must come into contact with the samples, the mortar and pestle were rigorously cleaned with appropriate chemicals (hydrochloric acid that was heavily diluted with distilled water was used for cleaning the mortar and pestle) to substantially reduce the likelihood that impurities are introduced into the sample. For instance, any dust particles introduced into the sample at this early stage would provide for ideal sites of nucleation when it is placed in the furnace leading to polycrystalline samples with small individual single crystals.

Once the stoichiometric mixture had been prepared the batch would be subdivided between a plurality of pre-cleaned ampoules and placed on a vacuum sealing line, as shown in Fig. 2.2. The powder mixture was carefully placed in the ampoules through a conical funnel to ensure that little of the powder stuck to the side of the quartz. The reason for this was to ensure that when the ampoules were sealed with the flame no powder obstructed the open end of the ampoule creating leaks to the ambient environment and to ensure that only one homogeneous melt was present at the bottom of the ampoule, thus providing a single starting point for crystallisation to occur.

After the powder was placed in the quartz ampoules, they were connected to the vacuum sealing line. To prevent any leaks occurring during the sealing process, the o-rings were pre-greased and the correct diameters for the connections were established. The atmospheric pressure was driven down to a pressure of ∼1Torr using a mechanical vacuum pump (this is depicted in Fig. 2.2). All vacuum pumps work on the basic principle of creating a pressure differential between an input and output inlet (with one of those connecting to the ampoule in this specific case) such that a force displaces the fluid from the input to the output inlet, thus driving the pressure in the vessel connected to the

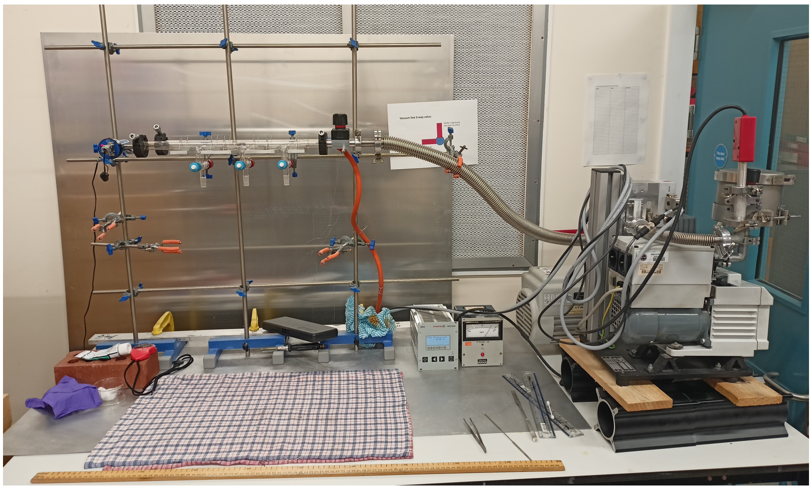


Figure 2.2: The vacuum sealing line is shown with its connections to the mechanical and turbo-pumps on the right of the image. The pressure gauge used to measure the pressure within the system is on the left and there is space for three ampoules, which allows for the evacuation of multiple quartz ampoules at the same time.

input inlet down and eventually creating an imperfect vacuum. Once a pressure of ∼1Torr inside the ampoule had been achieved it was checked for leaks. This was carried out by dispensing ethanol on the outside of the ampoule and verifying that there was not a rapid increase in the pressure as measured by a pressure gauge. Having checked that the quartz ampoules did not have any leaks, the vacuum pumping was switched from a mechanical to a turbo-pump system. The turbo-pump was able to drop the pressure inside the quartz ampoules below ∼1mTorr, but this procedure was carried out gingerly to ensure that the rate of de-pressurisation inside the vessel was not so large and would lead to the powder being sucked up into the vacuum sealing line. The rate of de-pressurisation could be controlled with the degree that the valve connecting the quartz ampoule to the vacuum sealing line was opened. There is sometimes remnant gas stuck within the powder mixture at the bottom of the ampoule that can be displaced with a simple tap of the side of the vessel. If the measured pressure using the gauge is still not declining, then it could imply that there is a leak within the system that ought to be searched for and rectified so as to continue the vacuuming process. A large proportion of the vacuuming can be carried out within ∼15 minutes, but to achieve the optimal vacuum pressures the sealing line is often left to run for several more hours or ideally overnight. The final reading on the pressure gauge is 0.0mTorr indicating that the pressure inside the quartz ampoule was below the sensitivity of the gauge, i.e., 1mTorr.

The next stage in the procedure was to flame seal the quartz ampoule. To ensure that the powder did not immediately vaporise on contact with the heat provided by the flame, an aerosol can was turned upside down and used to spray fluid on the ampoule that quickly solidified into ice, thus providing some buffer of protection for the powder. The valve connecting to the quartz ampoule was closed which allowed for the ampoule to be taken off the vacuum sealing

FIGURE 2.3: The three-zone furnace used for growing the single crystal samples. It had a temperature gradient due to the leakage of heat through the furnace door, which allowed for the horizontal Bridgman technique to be used for the synthesis. The heating run could be automatically controlled using the interface on the bottom right of the furnace.

line without the vacuum being lost. The combustion of propane gas is used to produce a flame and the flow rate of both the propane gas and oxygen is used for controlling the flame's strength. The tips of the flame torch can be changed to suit the operator's desired degree of concentration of the flame. The quartz ampoule is gripped such that it lies in an almost horizontal position but with a slight tilt for gravity to keep the powder from being displaced from the position at the closed end of the ampoule. At this stage of the procedure the objective is not to completely seal the ampoule but indent it somewhere along the cylindrical axis and far enough away from the powder so it is not disturbed by either the heat or the change to the morphology of the ampoule. The ampoule is indented at the centre so that a disc shaped piece of quartz can be placed inside the ampoule and supported by the indentation. Since it is necessary to open up the valve that is connected to the ampoule, and thus re-pressurising the vacuum held therein, the ampoule needs to be re-sealed and vacuumed using the aforementioned procedure for the vacuum sealing line. When the ampoule has been re-sealed and a vacuum has been created within it, the ampoule is completely sealed at the point of the indentation where the quartz disc is located. Each ampoule that is sealed is checked for any holes that may have formed during the flame sealing process. If none are found, then that ampoule, with the powder mixture intact inside the vessel, can be used in the next stage of the crystal growth procedure.

The next stage includes the first heating regime to grow single crystals using the Bridgman technique, which was broached before in this section and as mentioned uses the relatively uncontrolled temperature gradient created by the door of the furnace to grow crystals along ∇T. A picture of the three-zone furnace used for this crystal synthesis and the measured temperature profile therein are shown in Fig. 2.3-2.4. The temperature was measured using a thermocouple that uses the Seebeck effect to gauge the temperature: this is the electromotive

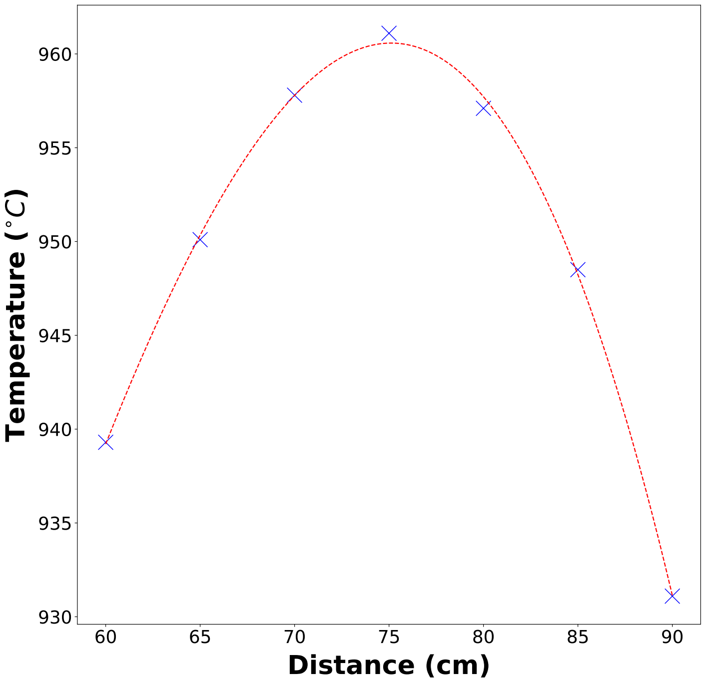


FIGURE 2.4: The temperature at specific points within the three-zone furnace that was used for the growth of single crystals of $Fe_{1+y}Te$. The distance is measured from the end of the three-zone furnace and the temperature was determined using a thermocouple.

force generated between two points of an electrically conducting material because of the temperature difference between them. An ideal temperature profile is a large temperature gradient from the door of the furnace to the heat source, which is conducive for more localised undercooling of the powder mixture and the growth of larger single crystal samples.

Having verified that the temperature gradients within the three-zone furnace suffice for use in the Bridgman technique of single crystal growth, it was necessary to construct a heating regime for the stoichiometric powders contained within the quartz ampoules. These ampoules were placed horizontally along the floor of the three-zone furnace but raised slightly at the end closest to the door using some fire-proof mineral wool (aluminium oxide, Al_2O_3). A substantial quantity of further mineral wool was placed around each ampoule to reduce the risks associated with an ampoule exploding (the pressure difference between the inside of the ampoule and the environment was a hazard) and potentially creating a fire. A guide to the required heating regime for the $Fe_{1+y}Te_{1-x}Se_x$ system had been written by [81]. It provided a lot of instructive information about the temperature regimes that successfully produced large single crystal samples and was therefore a valuable starting point for the synthesis of single crystals of $Fe_{1+y}Te$ for this thesis. It suggests a two-stage heating regime with a pre-melting stage and a final stage for single crystal growth. The pre-melting stage is not essential, but it is suggested that following it would lead to improved success rates for the production of large single crystals. It involves melting the stoichiometric powder and solidifying the resultant melt by cooling it, regrinding the ingot, which might contain multiple phases with constituent Fe and Te atoms, until it is an homogeneous powder, before proceeding to another heating phase for the growth of $Fe_{1+y}Te$ single crystal samples. Furthermore, the quartz ampoules can often crack in the furnace and to prevent this being an issue it was decided to doubly seal them with another evacuated tube for extra preventative measures- this reduces the risk of contaminating the melt with the ambient gases inside the furnace, which would hinder the growth of large single crystal ingots.

The pre-set pre-melting heat regime for the non-stoichiometric $Fe_{1+y}Te$ powder was as follows: the temperature was ramped up to 900°C at a rate of ∼50°C/hr; it was held at this temperature for 12hrs; the temperature was then reduced to 300°C at a rate of ∼1°C/hr. Once the temperature within the three-zone furnace had cooled below ∼300°C it is not completely necessary to wait for the temperature to further fall to room temperature, instead it is possible to turn off the heating element and allow it to cool down on its own. The same heating regime was used for the second heating schedule after the pre-melting had been completed. A schematic of the temperature-time plots for the heating regime are shown in Fig. 2.5.

The quartz ampoules, which were assumed to contain ingots of polycrystalline, rather than single crystal, $Fe_{1+y}Te$, were taken out of the three-zone furnace once they had fully cooled down. The doubly sealed quartz ampoules were broken into, taking care to not split up the ingots, using a heavy instrument, such as a mallet or hammer. Often the ingot will cleave naturally into smaller parts along the faces of constituent grains. These smaller pieces can be

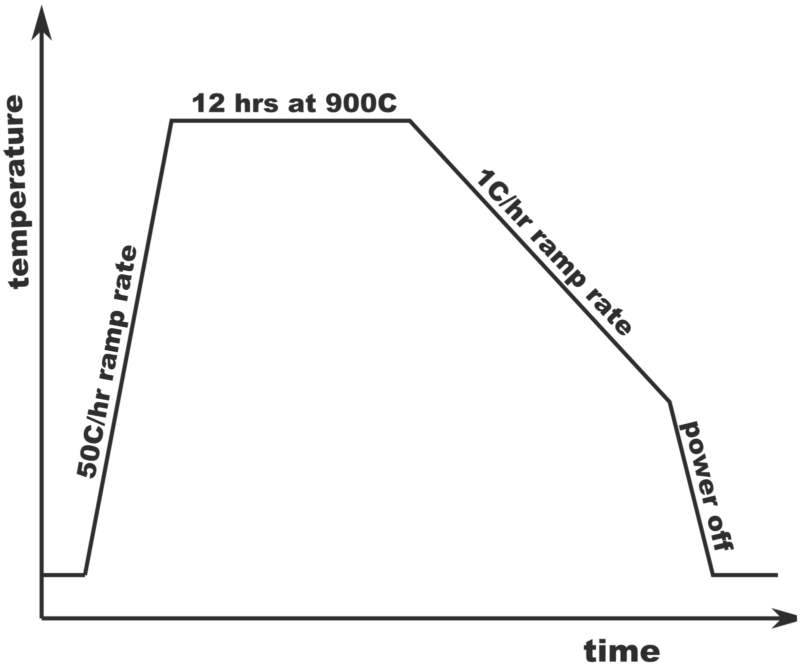


FIGURE 2.5: The heating regime (used for both the pre-melting and the final stages of the synthesis process) for producing the $Fe_{1+y}Te$ single crystal samples using the horizontal Bridgman technique. The natural gradient of the three-zone furnace provided necessary undercooling close to the interface between the melt and the growing crystal.

checked for single crystals using the natural eye (i.e., if there are any faces that have a particular shine indicative of the face of a single crystal of $Fe_{1+y}Te$) or through the lens of a microscope and using an optical polariser. The polariser lens produces a different hue for different grains within a polycrystalline sample because the optical properties of the crystallites depend on the direction of electromagnetic wave propagation through them relative to the crystallite axes.

2.3 X-ray powder and Laue diffraction of $Fe_{1+y}Te$

In the preceding section, the methodology for growing polycrystalline and single crystal samples of $Fe_{1+y}Te$ was discussed in depth. Our understanding of the physics and chemistry behind the growth of these crystals is well developed but due to the relative uncontrolled nature of the crystal synthesis in the laboratory, and especially the relative uncontrollability of the temperature profile within the three-zone furnace, it is difficult to predict with certainty whether a sample will be prepared that suffices for use in an inelastic neutron scattering experiment. There are a few experimental tools that can be utilised to verify the quality of the samples produced. One of those tools is to use X-ray diffraction of a powder sample to check for the phases present in the samples and other such useful information: Rietveld refinement of the X-ray diffraction pattern is used to refine a theoretical powder diffraction pattern against the experimentally

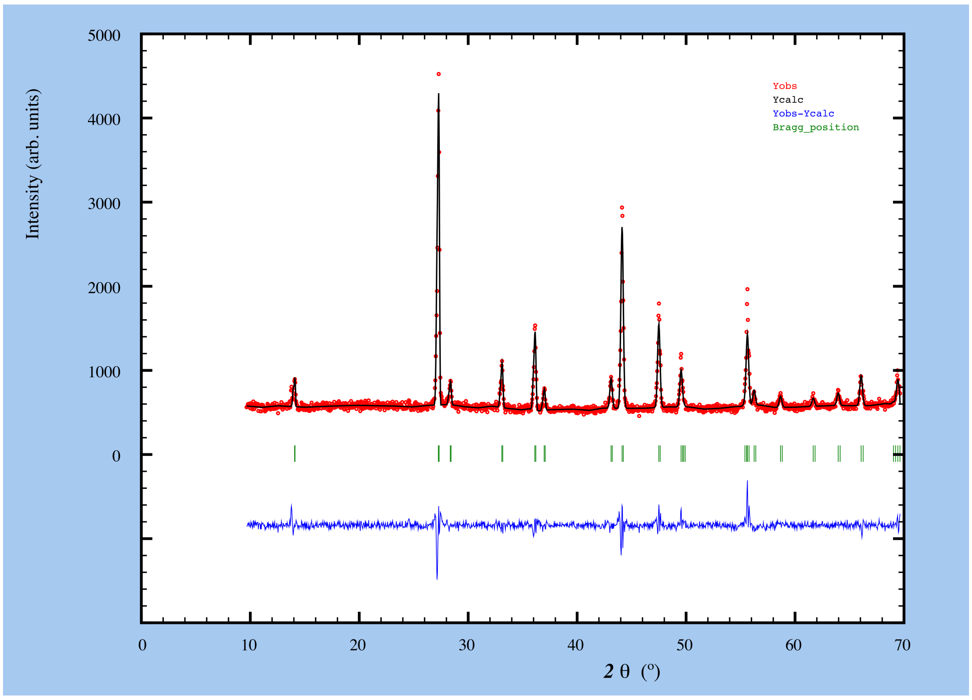


FIGURE 2.6: An X-ray powder diffraction pattern of the $Fe_{1+y}Te$ sample produced using the horizontal Bridgman technique. The experimentally observed data has been fitted to a theoretical diffraction pattern of the crystal using Rietveld refinement and shows the expected Bragg reflections (green lines). The refinement analysis reveals the interstitial iron concentration to be $y = 0.146(7)$ (i.e., in the orthorhombic phase in the low temperature regime), which is greater than the expected concentration based on the ratios of powder used in the recipe, namely $y = 0.095$.

observed pattern and use that to optimise several parameters that describe the sample. Since it is a powder diffraction instrument, a part of the sample needs to be re-ground in a mortar with a pestle and the measured diffraction pattern uses the "powder averaging" as described in Sec. 2.1.

The X-ray powder diffractometer used for this experiment, which was sourced from Oxford Diffraction Ltd., produces a collimated, monochromatic beam of X-rays from the filtered emission spectra of a target material in a cathode ray tube. The geometry of the X-ray powder diffractometer is such that the powder sample is placed on top of a goniometer that rotates by an angle of θ while at the same time an X-ray detector rotates at double this angle, i.e., 2θ. The powder sample is spread on top of a plate with the aid of some drops of ethanol from a pipette (the ethanol diffuses the powder sample across the diameter of the plate and quickly evaporates) which helps to cover the entire plate with a thin layer of the sample. The diffraction pattern was then measured for the sample covering detector angles ranging from 10° to 70°. The X-ray powder diffraction pattern observed for the $Fe_{1+y}Te$ samples is shown in Fig. 2.6 (which also plots the theoretical diffraction pattern fitted after Rietveld refinement was performed using FullProf [82]).

The Rietveld refinement analysis optimised several parameters related to the lattice, such as the positions of the Fe and Te ions, the occupancy of the constituent atoms (which gives an estimate of the interstitial iron concentration), the Debye-Waller factor (also called the B-factor), the lengths of the lattice axes and the angles between them using the tetragonal phase symmetry. The analysis minimises the following metric:

$$M = \sum_i W_i \{y_i^{\mathrm{exp}} - c y_i^{\mathrm{calc}}\}^2, \tag{2.7}$$

where W_i is the weight applied to the i^{th} value in the sum of intensities, y_i^{exp} is the experimentally determined intensity, y_i^{calc} is the theoretically calculated intensity given a particular crystal model, and c is a scale factor. There are several metrics used to gauge how well the crystal model fits to the X-ray powder diffraction pattern including R_{p}, R_{wp}, R_{exp}, R_{B} and χ^2, each with their own formula. The one that is most commonly used as a metric for the goodness of the fit to the data is χ^2 with a value of one equating to a perfect fit to the model. Tab. 2.1 shows the refined parameters for the $\mathrm{Fe}_{1+y}\mathrm{Te}$ sample produced using the horizontal Bridgman technique. The overall fit had a $\chi^2 = 2.99$ indicating that the crystal model was a reasonably good fit for the X-ray powder diffraction data. The refinement did give a higher R_{B} value, but this could be ignored given that the Bragg reflections of the calculated crystal model would depend on the structure of the interstitial Fe ions, which is hitherto a scientific question with an unknown answer (it is not known whether the interstitial Fe atoms form their own structure or whether they are randomly dispersed throughout the stoichiometric FeTe structure). For similar reasons, the Rietveld refinement analysis should not be assumed to provide a highly accurate determination of the occupancy (i.e., the interstitial concentration) of the second Fe atom. The analysis gives an interstitial iron concentration above the critical concentration, $y \sim 0.12$, even though the dry powder ratios should give a value below this threshold. It is accepted that the chemistry involved in the crystal growth is highly complex and the final interstitial iron concentration could be different to what would be expected from the dry weight ratios of the Fe and Te powder, but it is more likely that the analysis is overestimating the parameter in this case and that the low temperature regime would be in the monoclinic phase rather than the orthorhombic phase.

The Debye-Waller factors (or B-factors) were also determined in the Rietveld refinement analysis. However, they are often very inaccurate and should not be a taken to be a reliable guide to the thermal motion of the atoms within the crystal. The fact that some are negative further reinforces this argument and explains why this parameter is often simply a collective term for other corrections to the calculated diffraction pattern.

In summary, the Rietveld refinement analysis demonstrated that the crystal growth using the horizontal Bridgman technique had successfully produced $\mathrm{Fe}_{1+y}\mathrm{Te}$ samples with little evidence of the presence of phases such as FeTe_2 or other combinations of the Fe and Te species.

Lattice Parameter			Values		
Lattice lengths (Å)			a=3.82544(0)	b=3.82544(0)	c=6.28949(0)
Lattice angles			90.0°	90.0°	90.0°
Species	x (a)	y (a)	z (c)	B-factor	occupancy
Fe 1	0.75000(0)	0.25000(0)	0.00000(0)	0.706(0)	1.000(0)
Fe 2	0.25000(0)	0.25000(0)	0.69801(0)	-4.693(0)	0.146(7)
Te	0.25000(0)	0.25000(0)	0.28396(0)	-0.327(0)	1.000(0)
χ^2			2.99		

TABLE 2.1: Tabulated data from the Rietveld refinement of the X-ray powder diffraction pattern of the $Fe_{1+y}Te$ sample produced using the horizontal Bridgman technique.

Having confirmed that the growth method produced $Fe_{1+y}Te$ using the X-ray powder diffraction data, the samples were placed under a microscope with the optical polariser lens in order to locate candidates of potential single crystals. The ingots of $Fe_{1+y}Te$ were mostly polycrystalline samples, but some surfaces looked at under the microscope gave hope that there were some larger single crystals amongst them. Some of the potential larger single crystal samples were collected so that X-ray Laue single crystal diffraction experiments could be carried out.

A Laue diffraction experiment was carried out on samples of $Fe_{1+y}Te$ that were believed to contain large enough single crystals within their polycrystalline structure. They were placed on a sample mount in front of the X-ray beam as indicated by a red laser point. Since the samples were multi-grained it was critically important that the X-ray beam targeted a point on the sample where it was believed the larger single crystal was located. The samples were then exposed to a polychromatic X-ray beam for a period of 15 minutes in order to increase the contrast in the Laue diffraction patterns. Notwithstanding the lack of resolution of the instrument—which has recently been upgraded and used for the Laue experiment on the $CsPbBr_3$ samples, as shown in Fig. 2.9—some of the Laue diffraction patterns indicated the presence of small grains of pure $Fe_{1+y}Te$ single crystals. This could be seen by the fact that the symmetry of the Laue patterns fitted with the tetragonal structure of $Fe_{1+y}Te$. However, owing to the fact that the size of these single crystals was smaller than previously synthesised single crystal samples by Dr Chris Stock, it was decided that for the experiments it would be better to use those samples.

2.4 Miscellaneous: growth of $CsPbBr_3$ single crystals

This chapter has principally focused on the growth of $Fe_{1+y}Te$ polycrystalline samples and single crystals using the horizontal Bridgman technique. This is due to it being the condensed matter system that was to be studied for its lattice dynamical behaviour, especially the softening of the transverse acoustic phonons. However, there was also time to carry out another crystal synthesis using the same horizontal Bridgman technique: this was the growth of a perovskite crystal called caesium lead bromide, $CsPbBr_3$. Being a perovskite crystal it has the same crystal structure as $PbMg_{1/3}Nb_{2/3}$, thus it is interesting in itself for being in the same category of solid state system as the one studied for this thesis.

The stages of the crystal growth began in a similar way to that described for the growth of $Fe_{1+y}Te$ with a stoichiometric mixture of CsBr and $PbBr_2$ (both from Alfa Aesar with their crystalline purities being reported as 99% and 98% respectively) powders being ground in the same mortar with a pestle until they formed an homogeneous mixture. The molar masses of the CsBr and $PbBr_2$ were 212.809g/mol and 367.01g/mol respectively meaning that to prepare a 5g stoichiometric powder ready for the synthesis of $CsPbBr_3$ samples the following masses of these constituent crystalline powders were required:

$$m_{\mathrm{CsBr}} = \frac{212.809}{212.809 + 367.01} \times 5g, \tag{2.8}$$

$$m_{\mathrm{PbBr_2}} = \frac{367.01}{212.809 + 367.01} \times 5g. \tag{2.9}$$

An aid to the mixing process was that when the two substances were ground together they underwent a solid state reaction and produced a powder with a pale orange hue, which is very different from their original pale cream or beige colour. When the entire mixture had this hue it was known that the grinding process had been completed and the two substances were combined into one homogeneous mixture.

The next stage in the process was to doubly seal the samples in quartz ampoules using the methodology described in Sec. 2.2. To summarise, the samples were divided between several quartz ampoules that were evacuated on a vacuum sealing line using a mechanical and turbo-pump. These were then sealed closed using the flame from a torch and the process was repeated again to enclose the samples within two quartz ampoules, i.e., a smaller one within a larger one like a Matryoshka doll. The evacuated quartz ampoules were then placed in an Elite Thermal Systems box furnace and heated according to the following pre-programmed heating regime: the furnace was heated to 650°C/hr at a ramping rate of 50°C and then held at this temperature for a period of 24 hours before being cooled to room temperature at a rate of 1°C/hr (see Fig. 2.7). Once the temperature had been cooled to below 300°C there was also the option of shutting down the furnace and allowing the ampoules to cool more rapidly because it was unlikely at this stage that the crystals would undergo another solid state reaction. This heating regime was used for both the pre-melting stage and the final heating regime after the polycrystalline samples were re-ground into an homogeneous powder and re-sealed inside two quartz ampoules apiece. Fig. 2.8 shows the ingots of $CsPbBr_3$ produced by this crystal synthesis with the bright orange hues indicative of the successful growth of, at a minimum, polycrystalline samples of $CsPbBr_3$. There were also sections of the ingots that had darker hues, with some brown and black sections clearly visible, meaning that some other phases were present in the samples.

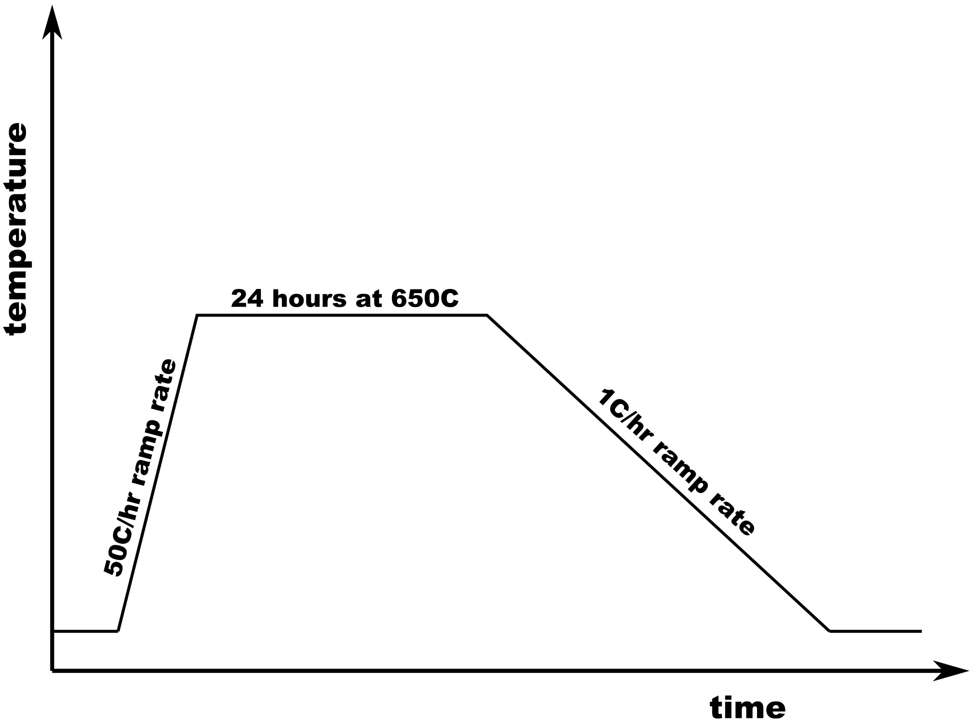


FIGURE 2.7: The pre-programmed heating regime (used for both the pre-melting and the final heating schedule) for producing the $CsPbBr_3$ single crystal samples using the horizontal Bridgman technique. The temperature gradient for localised undercooling of the melt of $CsPbBr_3$ was given by the gradient within the furnace due to natural heat sinks.

FIGURE 2.8: The $CsPbBr_3$ samples produced via the horizontal Bridgman technique that used the relatively uncontrolled environment of the furnace to generate a sufficient temperature gradient in order to cause localised undercooling of the stoichiometric melt in a crucible.

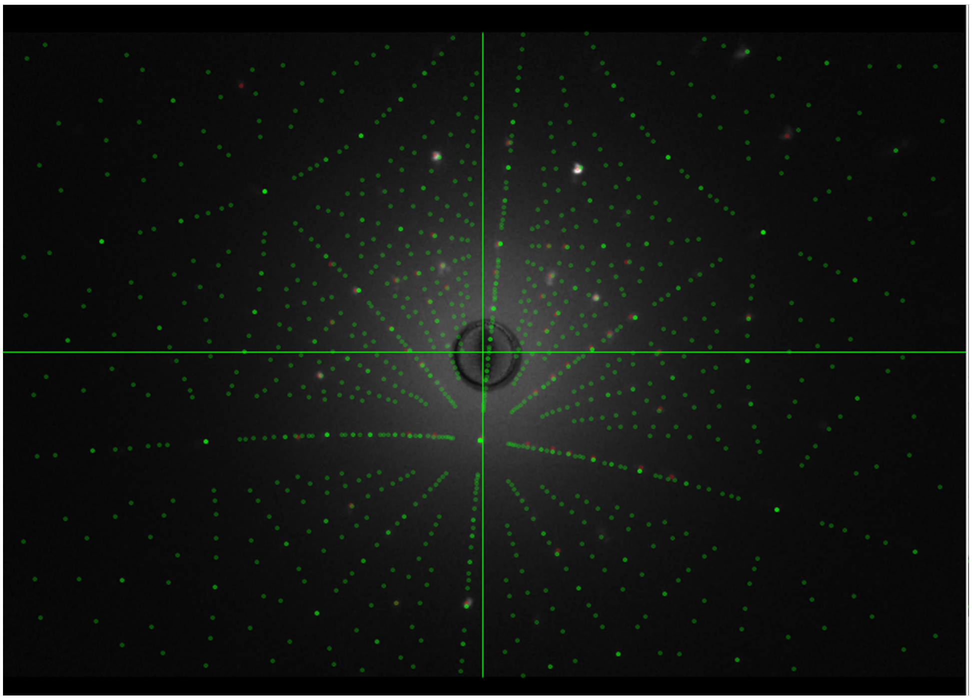


FIGURE 2.9: The Laue pattern for one of the $CsPbBr_3$ samples produced via the horizontal Bridgman technique. There was a good fit between the experimental data produced using the polychromatic X-ray beam and the theoretical diffraction pattern that was expected given the crystal symmetry and positions of the atoms within the perovskite unit cell of $CsPbBr_3$.

In order to verify the presence of single crystals of $CsPbBr_3$, samples of potential single crystals were collected (these were chosen based on their visual appearance when viewed under a microscope using an optical polariser lens) and probed using a Laue single crystal X-ray diffraction instrument. The samples were exposed to the polychromatic X-ray beam and the resultant diffraction spots produced by the diffraction of the X-rays from the parallel arrays of atoms in the crystal structure of $CsPbBr_3$ are shown in Fig. 2.9. The regular pattern of spots and their symmetry about the central spot were indicative of the presence of single crystals of $CsPbBr_3$ meaning that the horizontal Bridgman technique was successful in growing this compound. The growth of these crystals could be improved so as to remove the likely impurities present in the ingots (as shown by the dark hues) by more rigorous grinding of the constituent crystalline substances going into the stoichiometric mixture. An inhomogeneous mixture is likely to have caused the nucleation of different phases at different locations within the ampoules and leading to various species growing and forming grain boundaries with one another. This human error could be improved by the use of an automated mortar and pestle that would have the ability to both exert more pressure when grinding the powders and operate for much longer, thus ensuring the production of more homogeneous stoichiometric mixtures in future crystal syntheses.

Chapter 3

Phonon softening of transverse acoustic phonons in $Fe_{1+y}Te$

This chapter will detail the inelastic neutron scattering experiment carried out on the EIGER triple-axis spectrometer at PSI, Switzerland, which focused on the softening of the transverse acoustic phonons within a single crystal of $Fe_{1+y}Te$ during a structural phase transition that brought the crystal from the tetragonal phase to the monoclinic phase (from space group P4/nmm to $P2_1/m$). As has been stated in Sec. 1.4.2, the physics of the $Fe_{1+y}Te$ system is highly complicated owing to the coincidental structural, magnetic and electronic phase changes whose origins are quite likely to be related to some degree. It is very possible that one of these phase changes is driving the other through a coupling mechanism, which is something that has already been researched in other iron-based systems, including FeSe, underdoped $Ba(Fe_{0.94}Co_{0.03})_2As_2$ and optimally doped $Ba(Fe_{0.94}Co_{0.06})_2As_2$ [83]. In these systems, the acoustic phonon softening pointed towards the nematic-elastic coupling (the coupling mechanism that acts between the nematic electronic order, which is generated by the double striped spin structure, and the elastic constants due to the electrostatic potentials) being responsible for the softening at long wavelengths. Specifically, the anomalous softening that created a kink in the acoustic phonon dispersion curve close to the centre of the Brillouin zone ($q < 0.1$) could be used to determine the nematic correlation length scale within the electronic order of the iron-based system, thus revealing the interconnectedness of the crystal's structural, magnetic and electronic order.

The objective of this inelastic neutron scattering experiment is likewise to study the lattice dynamical behaviour of the $Fe_{1+y}Te$ system as a function of temperature and understand whether the system undergoes anomalous phonon softening of the transverse acoustic modes. It would be useful to observe whether this system undergoes the same anomalous softening of the modes for $q < 0.1$, and if so, to determine the corresponding nematic correlation lengths within the system's electronic order. If this type of anomalous softening is not seen in the experimental data, then the task is to observe how the transverse acoustic phonons soften and what might be causing it to do so in that way. For instance, a critical difference would be to observe anomalous phonon softening close to the Brillouin zone edge rather than at the Brillouin zone centre, which might be an indication that the physical mechanism behind the softening during the phase transition is different in the $Fe_{1+y}Te$ system than the other iron-based systems that have been mentioned.

It is evident from this short introduction to the acoustic phonon softening within the iron-based systems that a lot of information about them can be extracted by studying the dispersion curves (i.e., the energy of the phonon modes as a function of the position in reciprocal space within the Brillouin zone). It would thus be useful to give the reader a basic introduction to the link between the features of an acoustic phonon's dispersion curve and the inherent order of a crystal.

3.1 Elastic constants and long-wavelength acoustic phonon softening

Long wavelength acoustic phonon modes can be used to determine the elastic constants of a crystal. By understanding the relation between the dynamical matrix and the elastic tensor, it is possible to use the dispersion curves along various high symmetry directions of the crystal to calculate the elements of this tensor- the so-called elastic constants. In an experiment, sometimes only one of these elastic constants is required meaning that an inelastic neutron scattering study can restrict its search in momentum space to specific high symmetry directions. As an example, for a crystal with cubic symmetry the C_{44} elastic constant can be ascertained by finding the gradient of the acoustic phonon mode along $q \parallel [010]$. The relationships between the other elastic constants and the gradients of phonon dispersion curves along different high symmetry directions is rather complex, but in the following it is shown how this can be done for any periodic solid state system, including both the low and high temperature phases in $Fe_{1+y}Te$.

The important starting point, as demonstrated in reference guides to linear elastic theory (including both the discrete and continuous cases), is to appreciate that the elastic constants are related to the dynamical matrix through the strain components:

$$e_{ij} = \frac{\partial u_i}{\partial r_j}, \tag{3.1}$$

where $\mathbf{u} = (u_x, u_y, u_z)$ is the displacement of a volume element in the crystal and $\mathbf{r} = (x, y, z)$ is the displacement of this element from the origin of the system. However, this is usually rewritten as the classical pure strain components, ϵ_{ij}:

$$\epsilon_{ij} = \frac{1}{2}(e_{ij} + e_{ji}). \tag{3.2}$$

Distortions from an equilibrium volume bring about a change with respect to its energy such that the potential energy per unit volume, U, is given by the following formula in terms of these classical strain coordinates:

$$U = \frac{1}{2}\Sigma_{ijkl} C_{ijkl} \epsilon_{ij} \epsilon^{*}_{kl}, \tag{3.3}$$

which is a generalised extension of Hooke's law to include both the normal and shear strains. The former is when the strain is perpendicular to the face of the volume element, whereas shear strain is defined as when it is parallel to the face of the element. This formula for the potential energy of the system when undergoing strain can be used to form a system of Newtonian equations of motion which is likewise simply an extension of the one-dimensional Hooke's law equation of motion for a spring (or any linearly elastic object). First, we note that a conservative force is associated with the energy of a system by taking the gradient in vector calculus form:

$$\mathbf{F} = -\nabla U, \tag{3.4}$$

and applying this to the form of the potential energy in eq. 4.2 we get the following form for the classical equations of motion:

$$\rho \frac{\partial^2 u_i}{\partial t^2} dV = -\frac{\partial U}{\partial u_i} dV. \tag{3.5}$$

Treating the volume elements as having motion associated with travelling waves (in the context of studying the lattice dynamical response of $\mathrm{Fe}_{1+x}\mathrm{Te}$ they would be acoustic modes) the coordinates evolve in time such that:

$$\mathbf{u} = \tilde{\mathbf{u}} \exp i(\mathbf{k} \cdot r - \omega \mathbf{k}). \tag{3.6}$$

The left hand side of eq. 3.5 can then be rewritten as the following:

$$\rho \frac{\partial^2 u_i}{\partial t^2} dV = -\rho\omega^2 [\tilde{\mathbf{u}} \exp i(\mathbf{k} \cdot r - \omega \mathbf{k})]_i = -\rho\omega^2 u_i. \tag{3.7}$$

The right hand side becomes:

$$\begin{aligned} -\frac{\partial U}{\partial u_i} dV = -\frac{1}{2} \Sigma_{ijkl} C_{ijkl} \frac{\partial \epsilon_{ij}}{\partial u_i} \epsilon_{kl} dV \\ = -\frac{1}{2} \Sigma_{jkl} C_{ijkl} k_j (u_k k_l + u_l k_k) dV \\ = \Sigma_{jkl} C_{ijkl} k_j k_l u_k dV. \end{aligned} \tag{3.8}$$

Putting eq. 3.7 and eq. 3.8 together we get the equation of motion for the volume element:

$$\rho\omega^2 u_i = \Sigma_{jkl} C_{ijkl} k_j k_l u_k, \tag{3.9}$$

which can be rewritten in terms of the dynamical matrix thus forming a eigenvalue problem to solve:

$$\rho\omega^2 \tilde{\mathbf{u}} = \mathbf{M} \cdot \tilde{\mathbf{u}}. \tag{3.10}$$

And the dynamical matrix, **M**, can be defined in the following way by comparing eq. 3.9 with eq. 3.10:

$$M_{ik} = \Sigma_{jl} C_{ijkl} k_j k_l. \tag{3.11}$$

This can be solved using standard linear algebra techniques and demonstrates how the eigenvalues of the equation of motion, i.e., $\rho\omega^2$, are linked to the elements of the fourth rank elastic tensor, C_{ijkl}. Solving this eigenvalue problem can be substantially simplified by incorporating the symmetries inherent in both the strain and stress tensors. The elastic tensor can be reduced to 36 independent elements meaning that it is suitable to write it as a 6×6 matrix (a second rank tensor). This matrix is expressed below with the Voigt or Nye notation (second matrix):

$$\begin{aligned}\mathbf{C} = &\\ &\begin{pmatrix} C_{1111} & C_{1122} & C_{1133} & C_{1123} & C_{1131} & C_{1112} \\ C_{2211} & C_{2222} & C_{2233} & C_{2223} & C_{2231} & C_{2212} \\ C_{3311} & C_{3322} & C_{3333} & C_{3323} & C_{3331} & C_{3312} \\ C_{2311} & C_{2322} & C_{2333} & C_{2323} & C_{2331} & C_{2312} \\ C_{3111} & C_{3122} & C_{3133} & C_{3123} & C_{3131} & C_{3112} \\ C_{1211} & C_{1222} & C_{1233} & C_{1223} & C_{1231} & C_{1212} \end{pmatrix} \\ &= \begin{pmatrix} C_{11} & C_{12} & C_{13} & C_{14} & C_{15} & C_{16} \\ C_{21} & C_{22} & C_{23} & C_{24} & C_{25} & C_{26} \\ C_{31} & C_{32} & C_{33} & C_{34} & C_{35} & C_{36} \\ C_{41} & C_{42} & C_{43} & C_{44} & C_{45} & C_{46} \\ C_{51} & C_{52} & C_{53} & C_{54} & C_{55} & C_{56} \\ C_{61} & C_{62} & C_{63} & C_{64} & C_{65} & C_{66} \end{pmatrix}\end{aligned} \tag{3.12}$$

This matrix can be simplified even further by studying the symmetries of the crystal systems. In the case of Fe$_{1+y}$Te, it has a tetragonal phase at higher temperatures and undergoes a structural phase transition to an orthorhombic phase at approximately 70K [84]. Depending on the class of the tetragonal phase the matrix can be reduced to either 6 ($4mm$, $\bar{4}2m$, 422, $4/mmm$) or 7 (4, $\bar{4}$, $4/m$) independent elements. Likewise, when in the monoclinic phase this matrix can be reduced to 13 independent elements with the structure of the matrix differing slightly when the diad is parallel to different axes [85]. In the high temperature regime, Fe$_{1+x}$Te has a tetragonal structure and has 6 independent elastic constants:

$$\mathbf{C}_{tetragonal} = \begin{pmatrix} C_{11} & C_{12} & C_{13} & 0 & 0 & 0 \\ C_{12} & C_{11} & C_{13} & 0 & 0 & 0 \\ C_{13} & C_{13} & C_{33} & 0 & 0 & 0 \\ 0 & 0 & 0 & C_{44} & 0 & 0 \\ 0 & 0 & 0 & 0 & C_{44} & 0 \\ 0 & 0 & 0 & 0 & 0 & C_{66} \end{pmatrix}, \tag{3.13}$$

and in the low temperature regime with low interstitial iron concentration, after the crystal has undergone a structural transition to a monoclinic phase, the elastic tensor has the following form:

$$\mathbf{C}_{monoclinic} = \begin{pmatrix} C_{11} & C_{12} & C_{13} & 0 & C_{15} & 0 \\ C_{12} & C_{22} & C_{23} & 0 & C_{25} & 0 \\ C_{13} & C_{23} & C_{33} & 0 & C_{35} & 0 \\ 0 & 0 & 0 & C_{44} & 0 & C_{46} \\ C_{15} & C_{25} & C_{35} & 0 & C_{55} & 0 \\ 0 & 0 & 0 & C_{46} & 0 & C_{66} \end{pmatrix}. \tag{3.14}$$

These matrices, along with eq. 3.11, can then be used to determine the relationship between the gradients of the acoustic phonon modes in the long wavelength limit (i.e., q→0). Hence, by using the inelastic neutron scattering data to determine the dispersion curves of the acoustic phonon modes it is possible to understand how strongly the atoms within the crystal are coupled to one another, which is indicative of weaker or stronger coupled "springs" (the three-dimensional equivalent of when Hooke's law is used to study the dynamics of an infinite line of masses coupled with springs).

In later sections of this chapter, more details will be given about the experimental set-up associated with the inelastic neutron scattering experiment undertaken on EIGER at the PSI. For this section, the most relevant detail is that the spectrometer was set to look along q ∥ [010] direction, which could be achieved by choosing a scattering vector close to a reciprocal lattice vector but displaced along this axis, such as $\mathbf{Q} = (2, q, 0)$ (the energy transfer would depend on the choice of $\mathbf{k}_i$ and $\mathbf{k}_f$). Thus, for phonons along $[0, \zeta, 0]$ the dynamical matrix in eq. 3.11 reduces to the following:

$$M_{ik}^{[0,\zeta,0]} = C_{i2k2} k_2^2, \tag{3.15}$$

which in matrix form can be shown (keeping only non-zero entries as revealed in eq. 3.12) to be equivalent to the following:

$$\mathbf{M}^{[0,\zeta,0]} = \begin{pmatrix} C_{66} & C_{62} & C_{64} \\ C_{26} & C_{22} & C_{24} \\ C_{46} & C_{42} & C_{44} \end{pmatrix} k_1^2, \tag{3.16}$$

and applying the symmetries inherent in a tetragonal crystal structure (the high temperature phase of the $\mathrm{Fe}_{1+y}\mathrm{Te}$ single crystals) this becomes:

$$\mathbf{M}_{tetragonal}^{[0,\zeta,0]} = \begin{pmatrix} C_{66} & 0 & 0 \\ 0 & C_{11} & 0 \\ 0 & 0 & C_{44} \end{pmatrix} k_1^2. \tag{3.17}$$

The acoustic phonon dispersion relation can then be determined by finding the solution to the characteristic equation:

$$\det(\mathbf{M}_{tetragonal}^{[0,\zeta,0]} - \rho\omega^2 \mathbf{I}) = 0 \tag{3.18}$$

.

Due to the dynamical matrix being diagonal the solutions to the characteristic equation are straight forward with $\rho\omega_{[1,0,0]}^2 = C_{66}\zeta^2$, $\rho\omega_{[0,1,0]}^2 = C_{11}\zeta^2$, and $\rho\omega_{[0,0,1]}^2 = C_{44}\zeta^2$. Hence, the gradient of the dispersion curve for the acoustic phonon mode extracted from the inelastic neutron scattering experiments along one of those directions is equal to $\hbar\sqrt{\frac{C}{\rho}}$ where C is one of C_{11}, C_{44} or C_{66}. The same derivation can be carried out to find the gradient of the acoustic phonon dispersion curves of $Fe_{1+y}Te$ in the monoclinic phase, but since the degree of distortion is rather small it is possible to approximate that the equations governing the lattice dynamics are the same after going through the tetragonal–monoclinic temperature-dependent phase transition.

The softening of the acoustic phonons close to the Brillouin zone centre would therefore be indicative of a weakening of the "springs" (represented by smaller C_{11}, C_{44} and C_{66} elastic constants) coupling the Fe and Te atoms together in the unit cell of $Fe_{1+y}Te$. As we are interested in the TA mode along q ∥ [010], we will be probing the C_{44} and C_{66} elastic constants. The smaller of the two (meaning the lower energy TA mode) is the C_{44} elastic constant and is the one that the inelastic neutron scattering experiments on the EIGER TAS instrument will be studying.

3.2 Nematic order and anomalous phonon softening

As was mentioned in the introduction to this chapter, similar iron-based systems have revealed acoustic phonon softening due to the nematic electronic ordering. One question for this study on $Fe_{1+y}Te$ is whether the same type of phonon softening is occurring due to the orbital degrees of freedom in this condensed matter system, which is due to the different occupancies of the d_{Xz} and d_{Yz} orbitals in the valence shell of the Fe atoms. How would this nematic order of the electronic charge be observed in the transverse acoustic phonon dispersion? Following the same analysis of the study undertaken on the FeSe, underdoped $Ba(Fe_{0.94}Co_{0.03})_2As_2$ and optimally doped $Ba(Fe_{0.94}Co_{0.06})_2As_2$ systems [83], the nematic order disturbs the long-wavelength limit (q →0) of the phonon dispersion curve (see Sec. 3.1) such that it depends, in the long wavelength limit, on the nematic susceptibility at $q = 0$, χ_{nem}, correlation length of the nematic ordering, ζ, the bare shear elastic constant, C^0_{66}, and a nemato-elastic coupling constant, λ:

$$E(q) = |q|\sqrt{\frac{C^0_{66}(1+\zeta^2 q^2)}{\rho(1+\lambda^2\chi_{nem}/C^0_{66}+\zeta^2 q^2)}}. \tag{3.19}$$

This correction to the acoustic phonon dispersion curve close to the Brillouin zone centre seen in these iron-based systems should likewise be observed in the inelastic neutron scattering data for $Fe_{1+y}Te$ in the high interstitial iron concentration regime, i.e., for the tetragonal-orthorhombic structural phase transition. However, even if nematic electronic order is present in the $Fe_{1+y}Te$ single crystals, there would not be a large amount of difference between the undisturbed and disturbed phonon dispersion curves if $\zeta q \ll 1$ and $\lambda^2\chi_{nem} \ll C^0_{66}$. In order to fit this model to a wider range of the Brillouin zone, it is necessary, as it is suggested in the referenced study [83], to modify eq. 3.19 to take into account the periodic nature of the phonon dispersion curves:

$$E(q) = |\frac{\sin(D\pi q)}{D\pi}|\sqrt{\frac{C^0_{66}(1+\zeta^2q^2)}{\rho(1+\lambda^2\chi_{nem}/C^0_{66}+\zeta^2q^2)}}, \tag{3.20}$$

where D is a pre-factor that ensures the correct periodicity along the chosen high symmetry direction. The inherent challenge in this approach for finding the nematic correlation length through the distortions of the acoustic transverse phonon dispersion curve is the plurality of unfixed parameters which make fitting to a limited amount of experimental data very difficult. The approach taken by the study into these iron-based systems was to use previous studies that published data for the temperature dependence of the elastic constants of similarly doped compounds (e.g., $BaFe_2As_2$ and $BaFe_{1.84}Co_{0.16}As_2$ [86]) versus the bare elastic constant equivalent, i.e., the elastic constant without the presence of nematicity. The temperature dependence of the elastic constants for these iron-based systems follow a Landau phenomenological form, and are thus indicative of a phase transition to emergent nematic order, such that:

$$\frac{C_{66}}{C^0_{66}} \propto (T-T_0)^{-1/2} \tag{3.21}$$

where T_0 is a phase transition temperature and the pre-factor differs on either side of the phase transition (Landau's phenomenological theory of phase transitions explains that this is double on the high temperature side compared to the low temperature side).

In the FeSe, underdoped $Ba(Fe_{0.94}Co_{0.03})_2As_2$ and optimally doped $Ba(Fe_{0.94}Co_{0.06})_2As_2$ systems, the nematicity in the electronic order induces the acoustic phonon dispersion curves to soften close to the Brillouin zone centre but to re-harden towards the middle point of the Brillouin zone such that the dispersion approximately returns to where it would have been if there was no nemato-elastic coupling. The effect of the nematicity on the acoustic phonons in other single crystals, such as $Fe_{1+y}Te$ would depend on the relative magnitudes of the uniform nematic susceptibility, χ_{nem}, and the nematic correlation length, ζ. In a continuous phase transition, both of these can be described by their critical behaviour close to the phase transition temperature:

$$\chi_{nem} = \begin{cases} a^-(1-\frac{T}{T_0})^{-|\alpha^-|} & \text{if T} < \text{T}_0 \\ a^+(\frac{T}{T_0}-1)^{-|\alpha^+|} & \text{if T} > \text{T}_0 \end{cases}$$

$$\zeta = \begin{cases} b^-(1-\frac{T}{T_0})^{-|\beta^-|} & \text{if T} < \text{T}_0 \\ b^+(\frac{T}{T_0}-1)^{-|\beta^+|} & \text{if T} > \text{T}_0 \end{cases}$$

where the pre-factors and critical exponents may differ below and above the phase transition temperature. Substituting these expressions into eq. 3.20, it becomes:

$$E(q) = |\frac{\sin{(D\pi q)}}{D\pi}| \times \begin{cases} \sqrt{\frac{C^0_{66}(1+(b^-q)^2(1-\frac{T}{T_0})^{-2|\beta^-|})}{\rho(1+\lambda^2 a^-(1-\frac{T}{T_0})^{-|\alpha^-|}/C^0_{66}+(b^-q)^2(1-\frac{T}{T_0})^{-2|\beta^-|})}} & \text{if T} < \text{T}_0 \\ \sqrt{\frac{C^0_{66}(1+(b^+q)^2(1-\frac{T}{T_0})^{-2|\beta^+|})}{\rho(1+\lambda^2 a^+(1-\frac{T}{T_0})^{-|\alpha^+|}/C^0_{66}+(b^+q)^2(1-\frac{T}{T_0})^{-2|\beta^+|})}} & \text{if T} < \text{T}_0 \end{cases}$$

Fig. 3.1 shows the temperature dependence of the acoustic phonon dispersion using the above equation for a fixed set of parameters. This should simply be taken to show the effect that the nematic order has on the degree of softening on the acoustic phonons at given parts of the Brillouin zone.

The above example of how nematic electronic order can induce acoustic phonons to soften as the temperature approaches the phase transition from above and below the critical temperature is a useful pedagogical illustration of the phenomenon. However, due to the complexity of the physics within $Fe_{1+y}Te$ (there are two possible drivers of the structural phase transition, i.e., the magnetic and nematic fluctuations) and the fact that below the critical interstitial iron concentration the structural distortion is from a tetragonal–monoclinic crystal structure, the exact form of the softening of the transverse acoustic phonons within this iron-based system might differ from those previously studied. In order to derive a more appropriate model of the phonon energies, it would be necessary to derive the free energy of the $Fe_{1+y}Te$ as a function of the relevant order parameters (e.g., the shear strain, nematic order parameter, etc.) for this system.

3.3 Neutron scattering study of phonons

The previous sections of this chapter have explained how information from inelastic neutron spectroscopy can be used to determine physical parameters which describe both the structure and nematic order of a periodic crystal structure. It has shown how the gradient of the acoustic phonon dispersion curve can be used to determine the elastic constants and phonons can soften as they approach a continuous phase transition temperature from below and above, and that this can be explained through a Landau phenomenological approach that considers the symmetries in the system. The remaining sections of this chapter will instead detail the inelastic neutron scattering study performed on single crystals of $Fe_{1+y}Te$ (with $y < 0.12$) and the methodology for analysing the data collected on the EIGER TAS instrument at PSI, Switzerland.

3.3.1 Methodology for analysing neutron scattering data

The inelastic neutron scattering experiments were performed using the EIGER TAS spectrometer at the SINQ spallation source (details about this instrument can be found in Sec. 1.3.4). The spectrometer was operated in constant momentum transfer mode, $\mathbf{Q} = (2, q, 0)$, with the temperature of the $Fe_{1+y}Te$ single crystals being varied from 2–280K in order to measure the softening of

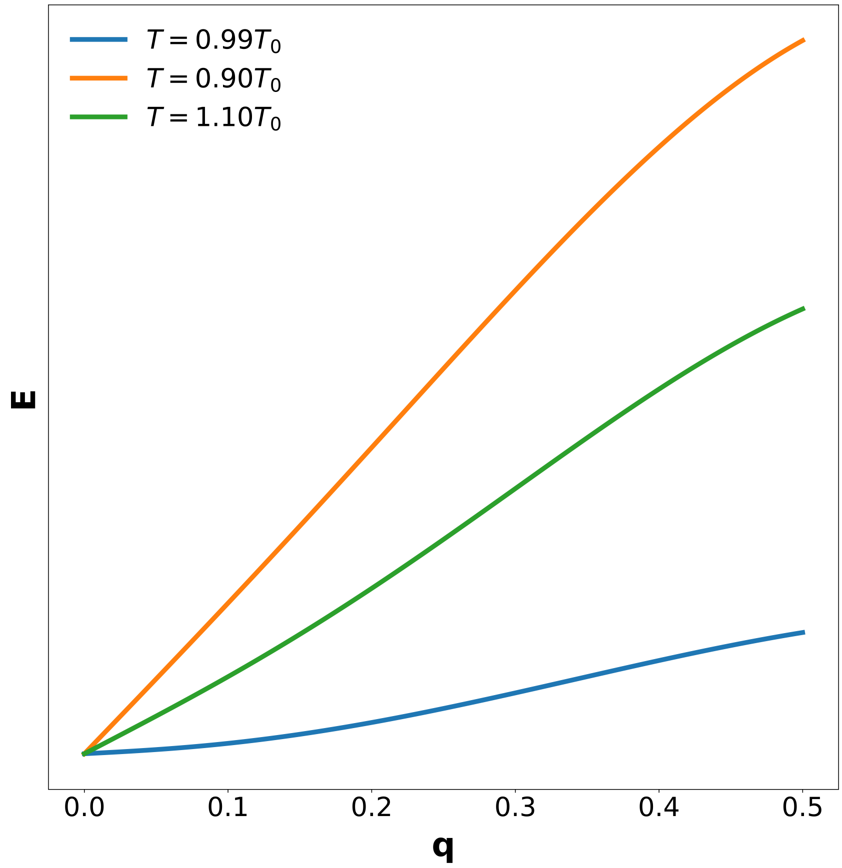


FIGURE 3.1: An illustration of the temperature dependence of the acoustic phonons across one half of the Brillouin zone ($0 < q < 0.5$) due to nematic fluctuations in a tetragonal-orthorhombic phase transition. The pre-factors and critical exponents for the uniform nematic susceptibility and the nematic correlation length have been fixed such that they are equal on either side of the phase transition temperature. The critical exponents used were $\alpha = 1$ and $\beta = 0.5$.

the TA mode on both in the vicinity of the tetragonal–monoclinic phase transition. The objective of the experiment was to use the inelastic scattering data to understand whether the softening was similar or different to the anomalous phonon softening described in Sec. 3.2, and if it behaves differently then ideally to suggest possible causes of its critical behaviour.

The neutron scattering intensities measured in EIGER, $\frac{d^2\sigma}{d\Omega d\omega} d\Omega d\omega$, provide access to the scattering function, $S(\mathbf{Q}, \omega)$, which contains all the static and dynamic physics of the system:

$$\frac{d^2\sigma}{d\Omega d\omega} = \left(\frac{\gamma r_0}{2}\right)^2 \frac{k_f}{k_i} S(\mathbf{Q}, \omega), \tag{3.22}$$

where γ is the gyromagnetic factor of the neutron, r_0 is the classical radius of the electron, and k_i and k_f are the incoming and outgoing momentum of the neutrons which are related to the momentum transfer by $\mathbf{Q} = \mathbf{k_f} - \mathbf{k_i}$. There is a further link between this scattering function and the dynamic susceptibility of the crystal, $\chi(\mathbf{Q}, \omega)$: it can be shown that the former is related to the imaginary part of this dynamic susceptibility function:

$$S(\mathbf{Q}, \omega) = \frac{\hbar}{\pi} \frac{1}{1 - \exp\left(-\beta\hbar\omega\right)} Im\chi(\mathbf{Q}, \omega). \tag{3.23}$$

This can be modelled by Lorentzian peaks in both the up and down scattering regimes such that the scattering function obeys the fluctuation-dissipation theorem, which is given below, but has also been described in Sec. 1.2.2:

$$S(\mathbf{Q}, \omega) = \exp\left(\beta\hbar\omega\right) S(-\mathbf{Q}, -\omega). \tag{3.24}$$

The following Lorentzian lineshape not only obeys this fundamental result in inelastic neutron scattering theory but accurately models the lattice dynamical response of the single crystals:

$$Im\chi(\mathbf{Q}, \omega) = \frac{\Gamma_0}{\hbar^2(\omega - \omega_0(\mathbf{Q}}))^2 + \Gamma_0^2 + \frac{\Gamma_0}{\hbar^2(\omega + \omega_0(\mathbf{Q}}))^2 + \Gamma_0^2, \tag{3.25}$$

where $\omega_0(\mathbf{Q})$ is the energy of the phonon mode at momentum transfer $\mathbf{Q}$ and Γ_0 is its associated linewidth- a measure of the broadening of the mode that physically signifies the finite lifetime of the phonons. An example of how eq. 3.23 is used to fit the inelastic neutron scattering data for the transverse acoustic phonon peaks in $Fe_{1+y}Te$ is shown in Fig. 3.2.

3.3.2 Analysis of the softening of the phonons close to the phase transition

This section will present the inelastic neutron scattering data collected on the EIGER instrument during October and November 2021. Since the same single crystal of $Fe_{1+y}Te$ was used and the alignment of their crystal axes were the same for both of these experiments, the data collected on both of these occasions is merged in the analysis that is to follow. In order to characterise the softening

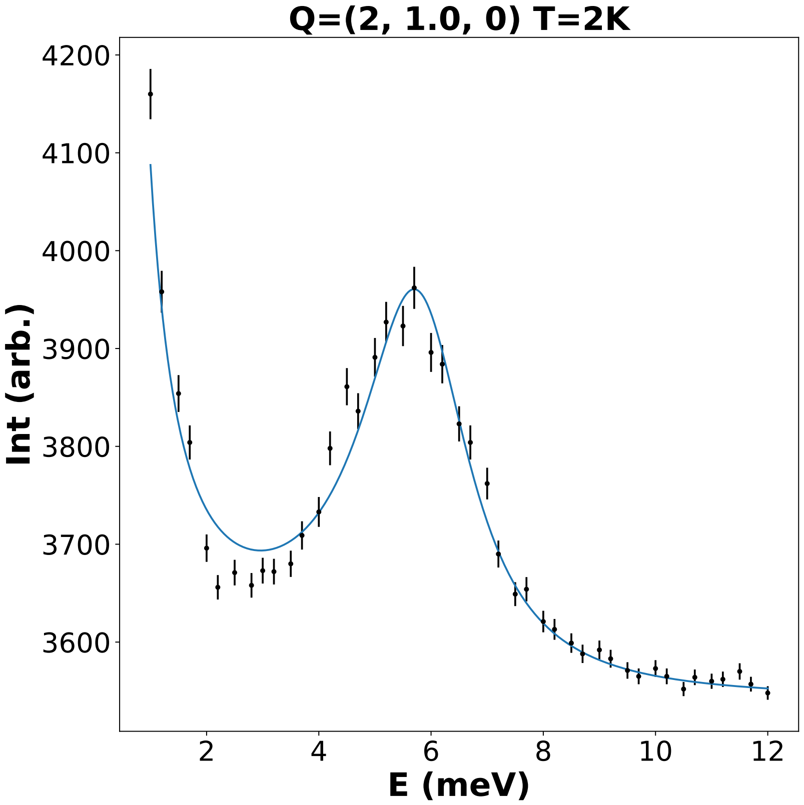


FIGURE 3.2: A constant momentum scan at $\mathbf{Q}$=(2, 1, 0) showing a close fit between the inelastic neutron scattering data and the model of the scattering as stated in eq. 3.23.

of the transverse acoustic phonons in the single crystal of Fe_{1+y}Te, a plurality of constant energy and constant momentum scans were carried out on the EIGER instrument for a range of temperatures ($2K < T < 280K$). The Fe_{1+y}Te single crystals had a low concentration of interstitial iron meaning that during this temperature range a structural phase transition taking the crystal from a tetragonal to monoclinic phase would be observed, and due to the symmetry breaking having an impact on the lattice dynamical behaviour it was hoped that this would be seen in the phonon energies. The form of the phonon softening could also aid in determining the driver of the structural phase transition, with one hypothesis being that the emergence of nematic electronic order is the cause of this structural distortion. As has been mentioned in Sec. 3.2, however, due to the complexity of the Fe_{1+y}Te system, which has structural, electronic and magnetic phase transitions with close transition temperatures, it is difficult to know which one is a principle driver of the others (the classic “chicken and the egg” dilemma). A point of reference is the softening observed for the acoustic phonons in the FeSe, underdoped $Ba(Fe_{0.94}Co_{0.03})_2As_2$ and optimally doped $Ba(Fe_{0.94}Co_{0.06})_2As_2$ [83] systems, which is why constant momentum scans were carried out at $\mathbf{Q} = (2, 0.3, 0)$ and $\mathbf{Q} = (2, 1, 0)$, i.e., one scan close to the centre of the Brillouin zone and another close to the edge of the Brillouin zone.

The reciprocal lattice vectors used were for Fe_{1+y}Te in its tetragonal phase since the lattice distortion taking the crystal from the tetragonal to the monoclinic unit cell is small, hence for the purposes of this experiment it was not necessary to renormalise the cells- $\mathbf{Q}$=(2, 1, 0) using the monoclinic lattice vectors would be equal to $\mathbf{Q}$=(2.01, 0.99, 0) with the tetragonal lattice vectors. These two constant Q-scans were chosen to study how the dynamics of the crystal behaved as the structural phase transition temperature was approached and whether any dynamical changes were being driven more by zone centre phonons or zone boundary phonons. Furthermore, the (200) Brillouin zone was chosen to take advantage of the Q^2-dependence of the coherent neutron scattering cross section for phonons. Although the cited study researched the softening of the acoustic phonons across a tetragonal-orthorhombic structural phase transition, it might be expected that the degree of phonon softening for those systems and for Fe_{1+y}Te with a low interstitial iron concentration would be similar. If this turns out to not be the case, then it will not entirely rule out the hypothesis that the nematic electronic correlations are the cause of the acoustic transverse phonon softening in Fe_{1+y}Te. This is because the form of the free energy functional that is used to derive eq. 3.20 could be very different owing to the different symmetries involved in the tetragonal-orthorhombic and tetragonal–monoclinic structural phase transitions. It is simply that in this analysis, without re-deriving from first principles the free energy functional and the associated theoretical phonon dispersion curve, it is assumed that the forms share a large degree of similarities, especially in where the softening occurs across the Brillouin zone.

The scattering function, $S(\mathbf{Q}, E)$, was assumed to be dominated by a coherent phonon peak for the transverse acoustic phonon mode and incoherent scattering. This incoherent phonon scattering was modelled such that it included a constant background and a Lorentzian peak centred at the elastic line,

$E = 0$.. Moreover, the resolution function of the instruments also contribute to the broadening of the phonon peaks and in general it produces a smearing of the entire inelastic neutron scattering data. The coherent phonon scattering was modelled by the expression given by eq. 3.23. It was assumed to also consist of Lorentzian peaks centred at the phonon energies in the up and down scattering regimes (the pre-factor is to ensure the term obeys detailed balance, as explained in Sec. 1.2.2 and 3.2). The lineshape is a Lorentzian since the lattice dynamics around the equilibrium positions of the atoms are quasi-harmonic with a damping factor producing this form of lineshape rather than a Dirac delta function.

Combining together all the contributions to the neutron scattering the model for the inelastic neutron scattering data for constant momentum scans was the following:

$$\begin{aligned} S(\mathbf{Q}_0, E) =& \frac{\hbar}{\pi}\frac{1}{1-\exp\left(-\beta\hbar\omega\right)}\left(\frac{\Gamma_{ph}}{\hbar^2(\omega-\omega_0(\mathbf{Q}_0))^2+\Gamma_{ph}^2}+\frac{\Gamma_{ph}}{\hbar^2(\omega+\omega_0(\mathbf{Q}_0))^2+\Gamma_{ph}^2}\right) \\ &+ \exp\left(\frac{1}{2}\beta E\right)\frac{\Gamma_0}{E^2+\Gamma_0^2}+c, \end{aligned} \tag{3.26}$$

where $\beta = K_B T$, Γ_{ph} is the linewidth of the coherent phonon peaks centred at E_{ph}, Γ_0 is the linewidth of the incoherent phonon peak centred at the elastic line, and c is the constant addition to the background. Fig. 3.3 shows fits of constant momentum scans at $\mathbf{Q} = (2, 0.3, 0)$ and $\mathbf{Q} = (2, 1, 0)$ for a range of temperatures crossing the continuous phase transition that takes the $\mathrm{Fe}_{1+y}\mathrm{Te}$ from the tetragonal (high temperature) to monoclinic (low temperature) crystal phase. The statistics are Poissonic owing to the binning of the neutron data at the detectors meaning that the intensity errors are given by $\delta I \approx \frac{1}{\sqrt{I}}$. As can be seen from the plots, the optimised fits are in strong agreement with the inelastic neutron scattering data. The relevant data to extract from these optimised fits for understanding the degree of softening of the acoustic transverse mode is the temperature dependence of the phonon peak energies, E_{ph}, and the phonon linewidths, Γ_{ph}, which, assuming the broadening is mostly explained by scattering mechanisms, i.e., phonon-phonon and phonon-electron scattering, gives an idea about the lifetimes of these acoustic phonons through the formula $\tau_{ph} \sim \frac{1}{\Gamma_{ph}}$. The temperature dependence of the phonon peak energies and linewidths for the zone centre and zone boundary are shown in the subplots of Fig. 3.4.

It is observed that both zone centre and zone boundary phonons soften as the structural phase transition is approached; however, the magnitude of the softening is much greater for the zone boundary phonons than the zone centre phonons, which is significant because in FeSe, underdoped $\mathrm{Ba(Fe_{0.94}Co_{0.03})_2As_2}$ and optimally doped $\mathrm{Ba(Fe_{0.94}Co_{0.06})_2As_2}$ there was a significant degree of softening at the zone centre due to the nematic correlation lengths. In this case, the softening appears to be caused by an increasing number of channels opening up for zone boundary phonon scattering close to the phase transition temperature with the scattering causing the phonon lifetime to be increased and the phonon

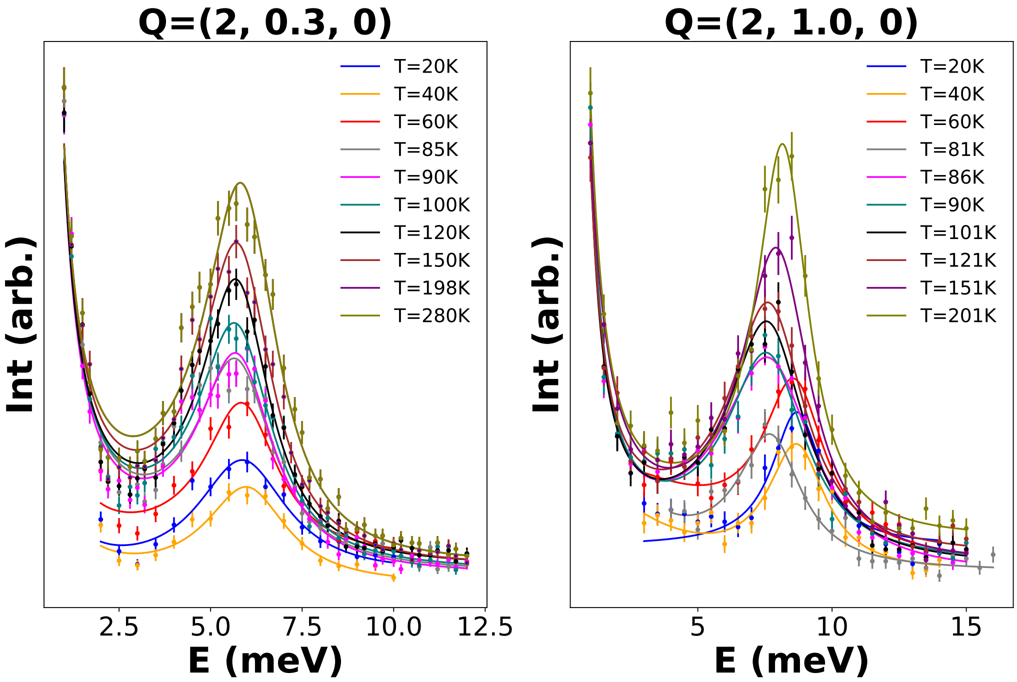
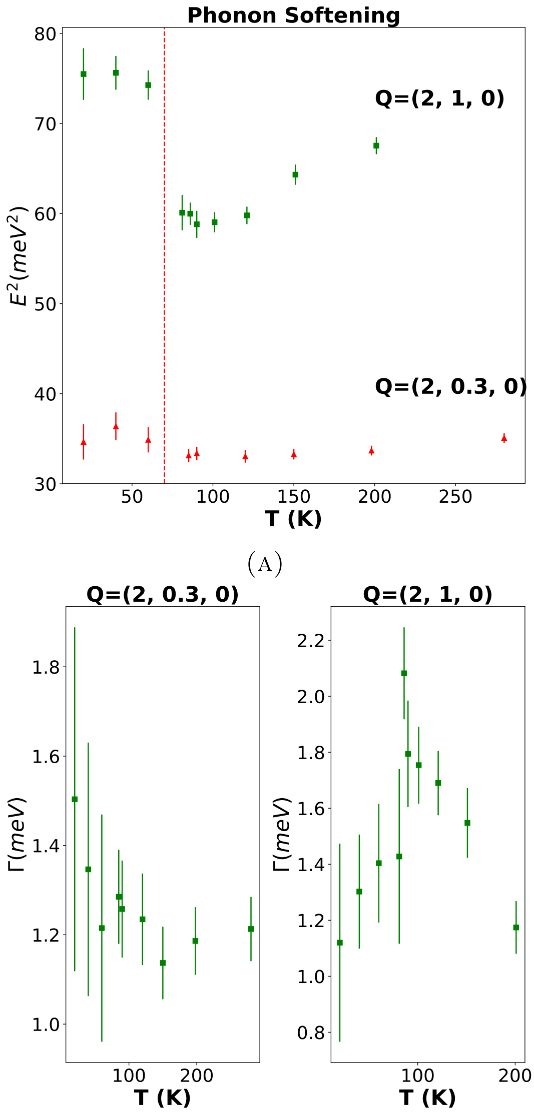


FIGURE 3.3: The model for the coherent and incoherent scattering has been fit to zone centre and zone boundary phonons for the transverse acoustic phonons in $Fe_{1+y}Te$. The temperatures cover the $Fe_{1+y}Te$ crystal in the tetragonal and monoclinic phases.

energy to be concomitantly reduced (a standard result for phonon dampening). The difference in the observed acoustic phonon softening for $Fe_{1+y}Te$ compared with these other iron-based systems could be due to the nematic order acting in a different way for the tetragonal–monoclinic phase transition in this condensed matter system as opposed to the tetragonal-orthorhombic phase transition in the latter systems. On the other hand, the scattering could possibly be due to the combined presence of the interstitial Fe atoms or other fluctuations (e.g., magnetic fluctuations) in the structure of the crystal close to the phase transition temperature. There is no statistically significant change in the phonon lifetimes close to the zone centre meaning that the anharmonicity of the potentials around the atoms does not change much close to the zone centre. The softening of the zone centre phonons is therefore likely to be due to the entire transverse acoustic phonon branch being pulled down by the zone boundary softening.

3.3.3 Bragg peak emergence

In the preceding sections of this chapter it has been stated that the $Fe_{1+y}Te$ system (in the low intersitial iron concentration regime) undergoes a structural phase transition from tetragonal (at high temperatures) to monoclinic (at low temperatures), i.e., from a high to low symmetry phase. It is possible to verify the existence of this structural phase transition in the emergence, or disappearance, of certain Bragg peaks associated with only one of the crystal structures



(A)

(B)

FIGURE 3.4: (a) Softening of the zone centre and zone boundary phonons. The phonon softening has characteristics of the Landau phenomenological theory of phonon softening with the gradient of the softening below the structural phase transition being greater than the gradient above the transition (the red dotted line indicates the approximate tetragonal–monoclinic phase transition temperature, T_S=70K, in $Fe_{1+y}Te$ [69]). (b) Temperature dependence of the phonon linewidths at the zone centre (left) and zone boundary (right). An increase in the phonon linewidth for the zone boundary corresponds with phonon softening around the structural phase transition.

and its symmetries. Since this inelastic neutron scattering experiment focused on the (210) Brillouin zone, it would be pertinent to find out whether there exists a Bragg peak at $\mathbf{Q} = (2, 1, 0)$ for any of the known structural phases of the $Fe_{1+y}Te$ system, which are the tetragonal, monoclinic and orthorhombic crystal structures. A software package called VESTA was used to visualise in three-dimensions the crystal structure of the $Fe_{1+y}Te$ single crystals in all three structure phases and to predict which Bragg reflections are allowed by each of their symmetries [87]. It demonstrated that a Bragg reflection is only permitted at $\mathbf{Q} = (2, 1, 0)$ for the tetragonal phase and neither of the monoclinic nor orthorhombic phases. The significance of this is that for both the tetragonal–monoclinic and tetragonal-orthorhombic structural phase transitions passing through the structural phase transition temperature should be accompanied by the disappearance of the (210) Bragg reflection (when the sample is being cooled).

Since there are multiple contributions to the scattering, i.e., Bragg scattering and incoherent quasi-inelastic scattering, it is often difficult to immediately analyse phenomena, such as a structural phase transition, using inelastic neutron scattering data. Moreover, the inelastic neutron scattering data collected on the EIGER instrument unfortunately did not cover the same energy ranges for all temperatures in the range studied (see Fig. 3.3). Ideally, the lower energies, i.e., E<4meV, would have been included in all of the constant momentum scans at the Brillouin zone edge in order to check whether any Bragg reflections existed and, most saliently, over which temperature range they existed. For analysing the inelastic neutron scattering data it is only possible to use the data that has been collected. Since the lowest energy that is included within all of the constant momentum scans at (210) are those at E=4meV, the scattering at this neutron energy transfer was used to check for the emergence of a Bragg peak and study the symmetry of the crystal as it goes through an expected structural phase transition. Due to the energy resolution of the EIGER instrument and other contributions to the broadening of Bragg peaks if a Bragg reflection emergences this ought to be visible in increased inelastic neutron scattering further away from the elastic line, i.e., E=0. The higher the energy the less effect the Bragg reflection has on the inelastic neutron scattering, but it is patently possible to see a structural phase transition at some distance from the elastic line. In the case of this study, the scattering intensity from constant momentum scans at the (210) Brillouin zone edge for E=4meV is plotted in Fig. 3.5 for a range of temperatures ($20K < T < 201K$). It is possible to observe some patterns within this inelastic scattering data: the higher temperature scattering is greater than the lower temperature scattering and, except for one anomalous data point, it appears that a Bragg peak emerges in the tetragonal phase at approximately the correct temperature. One reason why there might be the anomalous data point is that at this energy, i.e., away from the elastic line, there are a multitude of possible contributions to the inelastic neutron scattering, including incoherent inelastic scattering. Notwithstanding, there is clearly a difference in the scattering at low and high temperatures, and it is highly likely that responsibility for this difference in intensity comes from the symmetry change of the $Fe_{1+y}Te$ single crystal as it goes through the

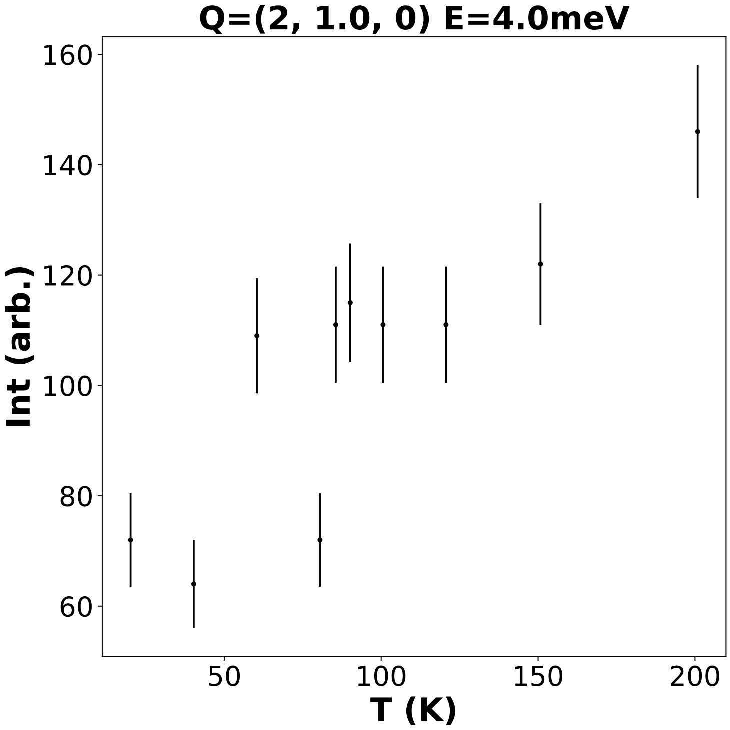


FIGURE 3.5: The scattering intensity close to the elastic line, E=4meV, for the $Fe_{1+y}Te$ single crystals for a range of temperatures. This range of temperature is suspected to comprise the tetragonal–monoclinic phase transition for the low interstitial iron concentration regime.

structural phase transition. However, it is not possible, from just this data, to conclude that the $Fe_{1+y}Te$ single crystal sample goes from a tetragonal crystal structure to a monoclinic crystal structure. However, having seen that the softening of the acoustic transverse phonon in the $Fe_{1+y}Te$ system is very different from that observed in the other iron-based systems with orthorhombic crystal structures (softening at the Brillouin zone edge as opposed to the Brillouin zone centre), it could be surmised that the difference in phonon softening due to nematic correlations is due to the different structural distortions that take place in the tetragonal–monoclinic phase transition, thus indicating that the $Fe_{1+y}Te$ sample in this study was in the monoclinic phase at low temperatures.

3.3.4 Extraction of the shear stress elastic constants

To extract the elastic constants from the data it would be necessary to include the effects of the nematic electronic order in the phonon softening. However, it is possible to ignore this effect in order to obtain a lower order approximation for the shear stress elastic constant. There is limited data covering the transverse acoustic phonon dispersion curve across the entire Brillouin zone, but this is available for the temperatures T=80K and T=280K. A simplified form of eq.3.20 that ignores the nematic order is used for extracting the shear stress elastic constant C_{44}:

$$E(q) = |\frac{2\sin\left(\frac{\pi q}{2}\right)}{\pi}|\sqrt{\frac{C_{44}}{\rho}}. \tag{3.27}$$

This equation has been used to extract a zeroth order approximation for the shear elastic constant associated with the transverse acoustic phonon mode in the $Fe_{1+y}Te$ system. As can be seen in Fig. 3.6, it provides a reasonable fit at the Brillouin zone centre (albeit a little bit under-fitting the inelastic neutron scattering data), whilst being a very poor fit for the Brillouin zone edge—indicative of possible nematic order causing the softening of zone edge transverse acoustic phonons. Notwithstanding the poor fit to the model at the zone edge, the model leads to C_{44} ~30GPa for both the T=81K and T=280K inelastic neutron scattering data.

How does this compare with theoretical predictions of the elastic constants in $Fe_{1+y}Te$ in its tetragonal (high temperature) phase using density functional perturbation theory? CASTEP, a leading software package for calculating the properties of materials from first principles [88], was used to give an approximate theoretical calculation for the elastic constants of stoichiometric FeTe in the higher symmetry phase. As usual all parameters were optimised to make sure a minimum of the free energy was found. Following this step the atoms were perturbed a small amount in all directions and the gradients of the free energy were used to determine all the elastic constants for the tetragonal phase—these are tabulated in the matrix below:

$$\mathbf{C}_{tetragonal} = \begin{pmatrix} 182.7\pm4.6 & 112.4\pm4.5 & 110.2\pm4.2 & 0 & 0 & 0 \\ 112.4\pm4.5 & 182.7\pm4.6 & 110.2\pm4.2 & 0 & 0 & 0 \\ 110.2\pm4.2 & 110.2\pm4.2 & 106.7\pm6.1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 43.7\pm0.6 & 0 & 0 \\ 0 & 0 & 0 & 0 & 43.7\pm0.6 & 0 \\ 0 & 0 & 0 & 0 & 0 & 59.9\pm0.7 \end{pmatrix} \tag{3.28}$$

The calculated results for the C_{44} elastic constant are slightly higher than those extracted from the inelastic neutron scattering data collected on the EIGER instrument. The difference between the two could be explained by the poor fit of the “no nematicity” model to the data or due to the presence of the interstitial iron atoms within the crystal structure of the sample.

3.4 Concluding the study of $Fe_{1+y}Te$

In summary, this chapter has set out to use the inelastic neutron scattering data collected on the EIGER instrument at the PSI to investigate the softening of transverse acoustic phonon modes in the $Fe_{1+y}Te$ system. This condensed matter system has a complicated structural, magnetic and electronic order meaning that it was challenging to pinpoint the exact origin of the softening and which one of these phenomena is driving the overall softening close to the phase transition temperature. The FeSe, underdoped $Ba(Fe_{0.94}Co_{0.03})_2As_2$ and optimally

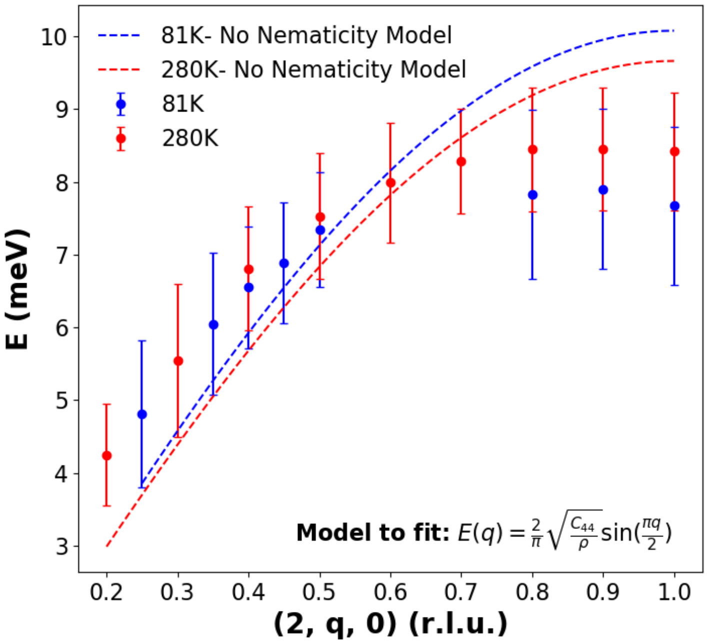


FIGURE 3.6: A fit of the data from separate constant momentum scans on Fe$_{1+y}$Te using the data points $q < 0.5$ and extrapolated towards the edge of the Brillouin zone. The model, which assumed there to be no nematic electronic order, shows a reasonable fit at the zone centre but a poor fit close to the zone edge. This could indicate that nematic order in Fe$_{1+y}$Te acts to bring down the entire acoustic transverse phonon mode curve from the Brillouin zone edge.

doped $Ba(Fe_{0.94}Co_{0.06})_2As_2$ systems have been used as a comparison in the analysis because of the assumed similarities of these iron-base systems with $Fe_{1+y}Te$, but it is clear that the softening of the acoustic transverse phonons in the latter are being driven from the Brillouin zone edge rather than the zone centre as in the former cases. The cause of this softening at the phase transition might still be due to the nematic electronic order (possibly due to the occupancies of the d_{Xz} and d_{Yz} orbitals in the Fe atoms), but another explanation is the opening of scattering channels created by the presence of the interstitial iron within the otherwise stoichiometric FeTe crystal structure. To verify whether the latter is responsible for the type of softening seen in these experiments, it would be necessary to repeat the inelastic neutron scattering experiments on different single crystal samples with different interstitial iron concentrations—this is beyond the scope of this thesis but could be researched further in future neutron experiments.

Chapter 4

Rayleigh-Schrödinger perturbative model of FeTe

As was stated in Section. 1.4.2, a bicollinear double stripe antiferromagnetic order emerges for the low-y configuration of the $\mathrm{Fe}_{1+y}\mathrm{Te}$ system. Several research papers have attempted to investigate the origin of this exotic magnetic structure including modelling the Hamiltonian as a Heisenberg spin model (hypothesising that the emergence of this magnetic order comes solely from the spin degrees of freedom) or a model that theorises that the magnetic interactions are being driven by orbital degrees of freedom in the Fe atoms and the interplay between the structural distortions due to the Jahn-Teller effect. Inelastic neutron scattering experiments carried out on FeTe [75] provide a strong motivation for intuiting that the Jahn-Teller effect is important for its magnetic interactions and concomitantly the magnetic structure and spin waves observed in the data.

In this chapter, which is the work produced via a collaboration with Dr Chris Hooley of the St. Andrews University and Janika Reichstetter of the University of Heidelberg, a Rayleigh-Schrödinger perturbative model for the virtual electron hoppings between the Fe and Te atoms in FeTe single crystals is developed in order to produce an effective Hamiltonian that includes both the spin and orbital degrees of freedom and the interplay between them. The magnetic interactions within the system are therefore governed by spin-only and orbital pseudospin-only (representing the d_{Xz} and d_{Yz} orbital physics on the Fe atoms) operators with multiplications of these two types of operators producing the interplay between the spin and orbital physics. Since the model is developed from first principles, lots of explanatory notes are provided within this chapter as an aid to the reader and to increase the knowledge necessary to comprehend how an input parameter may affect the final effective Hamiltonian.

The first section details information about the FeTe crystal structure and the single-ion spin and orbital degrees of freedom. This is important for understanding how the parameters are obtained for the Rayleigh-Schrödinger perturbative model and to give the reader a sense of the energy scales involved in the problem. Following on from this section, there is a detailed explanation of how the crystal field splitting is calculated given the symmetry of the single crystal structure and how the Hund's rule coupling parameters (the terms giving the energies associated with having the electron spins up or down within valence orbitals given a pre-existing electron configuration) are also determined. An entire section is devoted to explaining the Rayleigh-Schrödinger virtual hopping model, which is taken from the theoretical work carried out by Ingvar Lindgren

Species	x (a)	y (a)	z (c)	Wyckoff letter	Multiplicity
Fe	0.750	0.250	0.000	2a	2
Te	0.250	0.250	0.285	2c	2

TABLE 4.1: The refined Wyckoff positions and site symmetries of the Fe and Te atoms within the $Fe_{1+y}Te$ system for the tetragonal phase [90].

[89], and it is explained how this abstract theoretical model can be applied to the FeTe system. Some understanding of this model is required since without a basic understanding it is impossible to be able to replicate the effective Hamiltonian that has been developed for modelling the FeTe single crystals. The latter sections of this chapter use the theoretical tools that have been described and numerically apply them to the FeTe system using a combination of known structural data and educated guesses about parameters, such as the lattice distortion associated with the Jahn-Teller effect and effective nuclear shielding parameters to name only a few.

4.1 Crystal structure and single-ion spin and orbital physics

An important starting point to developing an effective Hamiltonian for the FeTe system is to understand its crystal structure. The reason for this is that the crystal structure indicates the symmetry of the crystal field around the constituent ions within the crystal and how the degeneracy of a valence shell may be lifted, thus producing a particular single-ion ground state. What does X-ray diffraction data tell us about the crystal structure of FeTe? This system is commonly synthesised producing non-stoichiometric $Fe_{1+y}Te$, where y, as described in Section. 1.4.2, is the interstitial concentration of Fe atoms. Taking experimental data that analysed $Fe_{1+y}Te$ with a low interstitial iron concentration, and therefore one that would produce little effect on the lattice constants and atomic positions, the crystal structure is revealed to be tetragonal at room temperature and atmospheric pressure with a space group of P4/nmm [90]. A refinement of the experimental X-ray diffraction data leads to lattice constants of a=3.829Å, c=6.288Å and α=β=γ=90°. If the refined positions of the interstitial Fe atoms are ignored (there is a large degree of uncertainty as to whether they have structure or are completely disordered throughout the crystal) the positions of the atoms in the stoichiometric version of the crystal can be obtained. The Wyckoff positions and symmetries of the Fe and Te atoms are presented in Tab. 4.1.

As shown in Tab. 4.1, the Fe and Te atoms have the Wyckoff letters 2a and 2c respectively. What this implies is that each of these species has another atom at position $(0.250, 0.750, 0.00)$ and $(0.750, 0.750, -0.285)$ respectively in each unit cell and linearly translated throughout the crystal structure. Hence, for stoichiometric iron telluride four Te atoms form a parallelogram around a single Fe atom. This is demonstrated in Tab. 4.2 which shows the positions of

Species	x (a)	y (a)	z (c)
Fe	0.000	0.000	0.000
Te1	-0.500	0.000	0.285
Te2	0.000	0.5000	-0.285
Te3	0.500	0.000	0.285
Te4	0.000	-0.500	-0.285

TABLE 4.2: The parallelogram of Te atoms around a central Fe atom within tetragonal iron telluride.

the four Te atoms in a reference frame generated by translating the unit cell such that the Fe atom is at the origin. The tetrahedron crystal field generated by the parallelogram of Te ions causes the valence shell of the Fe to split into two degenerate energy levels, a lower energy e_g level and a higher energy t_{2g} level. Using a cartesian grid that is co-linear with the crystal axes the e_g subgroup comprises the $d_{x^2-y^2}$ and d_{z^2} orbitals and the t_{2g} subgroup comprise the d_{xy}, d_{xz} and d_{yz} orbitals. The relative energies of these degenerate subgroups can be understood by visualising the fact that the orbital lobes in the e_g orbitals point along an axis that is more distant from the negative Te ions than the equivalent dominant axes in the t_{2g} orbitals. The electronic configuration of the crystal leads to a geometrical frustration (an energetic frustration of the electronic order) that can only be broken by a distortion of the lattice. This is the Jahn-Teller effect which tells us that a crystal with a spatially degenerate ground state will undergo a geometrical distortion in order to lower the overall energy (the entropy of the system is important owing to the multiple configurations available for the frustrated system). The effect of this geometrical distortion of the lattice is to lift the degeneracy of the 3d orbitals in the two subgroups, which breaks the frustration due to the degeneracy of different electronic configurations of the Fe atoms. The form of the Jahn-Teller distortion is difficult to predict with certainty but it is hypothesised that in the FeTe system one pair of Te atoms on opposite sides of the parallelogram move towards each other along an axis that is colinear with each of them and the other pair moves further apart. At the same time the Fe atoms are alternately shifted (the translation is along the same axis for each unit cell of the FeTe crystal) towards one Te atom in each distorted tetrahedron. This composite distortion of the tetragonal structure of FeTe is shown in Fig. 4.1, which might aid the reader's comprehension of this complicated three-dimensional geometrical distortion.

What effect does this geometrical distortion of the crystal structure have on the order of the 3d orbitals in the Fe atoms? If a new basis is chosen whereby $d_{Xz}=\frac{1}{\sqrt{2}}(d_{xz}+d_{yz})$ and $d_{Yz}=\frac{1}{\sqrt{2}}(d_{xz}\text{-}d_{yz})$, then these new orbitals would have their lobes (the space where they are most likely to be found if measured) facing the Fe atoms in the undistorted structure (assuming that the x and y axes bisect the line joining neighbouring Te atoms in Fig. 4.1). Even with the Jahn-Teller distortion both of these orbitals would have one lobe that points substantially towards one pair of Te atoms giving them the Coulomb repulsion energy. They will not have the same energy owing to the small translation of the Fe atom within each tetrahedron. The other 3d orbital in the t_{2g} subgroub,

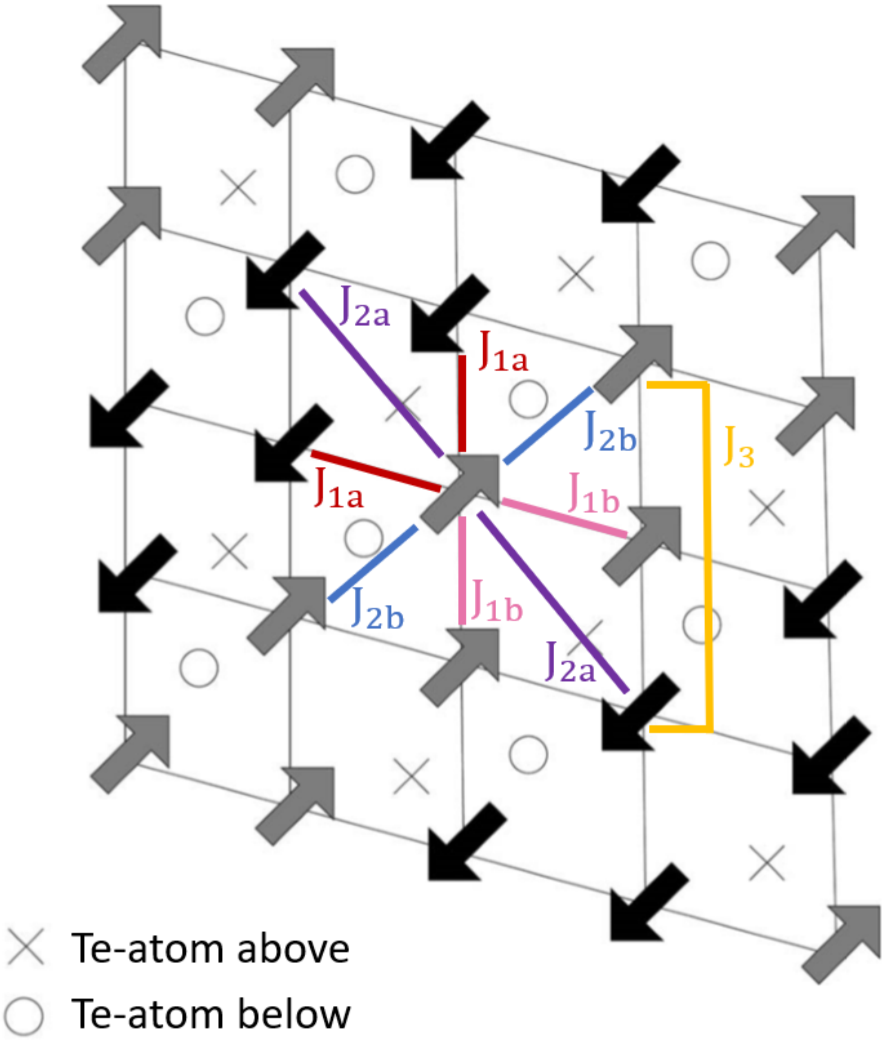
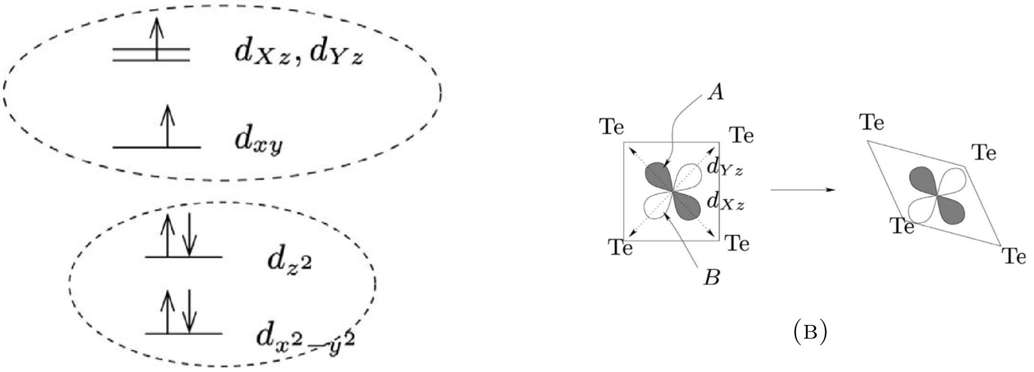


Figure 4.1: The tetrahedral structure formed by the Te atoms around each Fe atom in FeTe is distorted by the Jahn-Teller effect. The Jahn-Teller distortion lifts the degeneracy of the 3d orbitals in the Fe atoms and leads to the S=1 spin state [72].



(A)

(B)

FIGURE 4.2: (A) The relative energies of the 3d orbitals on the Fe atoms is changed by a combination of the compression and elongation of the tetrahedron and a Jahn-Teller distortion with the FeTe crystal structure. (B) The crystal field around the d$_{Xz}$ and d$_{Yz}$ orbitals is altered by the Jahn-Teller distortion, which lifts the degeneracy of their respective energies [72].

d$_{xy}$, has both lobes pointing between the Te atoms in the tetrahedra meaning that it will have a lower Coulomb repulsion energy than either of the other aforementioned orbitals. The degeneracy of the e$_g$ subgroup is also lifted with the d$_{x^2-y^2}$ orbital having a lower energy compared with the d$_{z^2}$ orbital. The relative energies of all of the orbitals in both the t$_{2g}$ and e$_g$ are shown in Fig. 4.2.

In Fig. 4.2, the electronic configuration of the Fe ion produces an S=1 spin state meaning each Fe ion has a magnetic moment that is consistent with the $\sim 2\mu_B$ ordered magnetic moment observed in the commensurate magnetic phase of Fe$_{1+y}$Te [91]. This is due to the exchange coupling term, $-J_H \mathbf{S}_1 \cdot \mathbf{S}_2$, between the d$_{xy}$ and d$_{Xz}$ orbitals (or the d$_{Yz}$ orbital since the order of the d$_{Xz}$ and d$_{Yz}$ orbitals is only an arbitrary choice of axis labelling) being dominant over the crystal field splitting between the latter and the d$_{xy}$ orbital. This energy comes from the fact that the expected value of the distance between two identical particles, such as the electrons filling these 3d orbitals, is increased with respect to electrons with opposite spins, which decreases the energy associated with the Coulomb repulsion between the charged particles. It ought to be noted that the symmetry of the crystal field, i.e., the field created by the tetrahedra of Te ions around each of the Fe ions, leads to the quenching of the orbital angular momentum, $\langle L_z \rangle = 0$, which can be seen by writing the 3d orbitals in the Fe, i.e., {d$_{xy}$, d$_{xz}$, d$_{yz}$, d$_{x^2-y^2}$, d$_{z^2}$}, in terms of the complex hydrogenic-like orbitals instead of these real orbitals. Using Dirac notation for the complex orbital basis, $|n, l, m_l\rangle$, where n is the principle quantum number, l is the angular momentum number and m$_l$ is the magnetic number:

$$
\begin{aligned}
d_{xy} &= \frac{i}{\sqrt{2}}(|3,2,-1\rangle - |3,2,1\rangle) \\
d_{xz} &= \frac{1}{\sqrt{2}}(|3,2,-1\rangle - |3,2,1\rangle) \\
d_{yz} &= \frac{1}{\sqrt{2}}(|3,2,-1\rangle + |3,2,1\rangle) \\
d_{x^2-y^2} &= \frac{1}{\sqrt{2}}(|3,2,-2\rangle + |3,2,2\rangle) \\
d_{z^2} &= |3,2,0\rangle
\end{aligned}
\tag{4.1}
$$

Since the 3d orbitals in the Fe ions are linear combinations of orbitals with $m_l = \pm m$ where $m \in \{0, 1, 2\}$, the expectation of $\mathbf{L}_z$ is zero. Furthermore, in the presence of a magnetic field the orbital angular momenta are not orientable because they are strongly coupled to the lattice by the crystal field, which explains the quenching and the negation of the orbital angular momentum when calculating the magnetic moment of the Fe ions, i.e., $\mu \sim 2\sqrt{S(S+1)}\mu_B$. This explains why the electronic configuration has a magnetic moment consistent with the experimental data [91] ($\mu \sim 2\sqrt{2}\mu_B$ being comparable with $2\mu_B$).

4.2 Crystal field splitting and Hund's rule coupling parameters

In the previous section, the symmetry of the crystal field in FeTe and its effect on the electronic configuration and relative energies of the 3d orbitals in the Fe atoms was described. This section will delve further into the physics and attempt to calculate the exact crystal field splitting of the energy levels and the Hund's rule coupling parameters using the real basis for the hydrogenic-like orbitals in eq. 4.1. Calculating these is critical to developing a first-principles theoretical framework for how the spin and orbital degrees of freedom have a role to play in the magnetic exchange physics of the FeTe crystal. Taking a first order perturbative approach the matrix of crystal field splitting parameters, $V_{\alpha\beta}$, is given by:

$$V_{\alpha\beta} = \int d^3\mathbf{r}\psi^{\dagger}_{Z\alpha}(\mathbf{r})V(\mathbf{r})\psi_{Z\beta}(\mathbf{r}), \tag{4.2}$$

where α and β are any of the five 3d orbitals listed in eq. 4.1 and $+$Ze is the core nuclear charge felt by the valence electrons in the 3d sub-shell of the Fe ions. However, seeing that the FeTe system is metallic there will be another contribution to the screening of the electric field due to conduction electrons that are hypothesised to originate from the interstitial Fe atoms donating some of their valence electrons to the solid. The effect of these conduction electrons on the electric field observed by the valence electrons can be modelled by the so-called Jellium model. In this model, the solid is comprised of the static electric charges due to the atomic nuclei in the crystal and a uniformly distributed sea of electric charge. It can be demonstrated that the effect of the sea of conduction electrons is to attenuate the electric field observed at a distance r from one of the static charges producing the screened Coulomb potential:

$$V(r) = \frac{q^2}{4\pi\epsilon r} \exp{(-\frac{r}{\zeta})}, \tag{4.3}$$

where ζ is the screening length due to the sea of conduction electrons in the crystal. The choice of Z and ζ can be guided by our knowledge of the structure of the system being studied, but the utility of this can only take the calculations so far because the objective is to reproduce the S=1 spin state of the Fe atoms with the electron orbital scheme shown in Fig. 4.2. Nonetheless, it is appropriate to start from such a point when doing theoretical calculations and deviations from the expected results can lead to optimisation of the variable parameters. A good starting point for the effective core nuclear charge felt by the valence electrons in the Fe atoms is to consult Slater's rule, which gives an estimate in the case of a many-electron atom. Slater's rule associates a different degree of shielding or screening for core electrons that differ depending on the principal quantum number and angular momentum number of the core and valence electrons. The effective nuclear charge is then calculated by the formula:

$$Z_{\text{eff}} = Z - s, \tag{4.4}$$

where Z_{eff} is the effective core nuclear charge, Z is the nuclear charge and s is the sum of the contributions of all the core electrons to screening the valence electrons. In the case of the valence electrons in the Fe^{2+} ions, the contribution to s for the six valence electrons in the 3d sub-shell is 19.75 giving an effective core nuclear charge of Z_{eff}=6.25 (the nuclear charge of the Fe^{2+} ion is 26).

What would be an appropriate starting point for the screening length of the Te^{2-} charges due to the conduction electrons contributed to the solid by the interstitial iron concentration? Since we are dealing with a metallic solid at low temperatures, the sea of conduction electrons needs to be treated using the Fermi-Thomas approximation. In this theoretical treatment the system of electrons is maintained at a constant electron chemical potential (i.e., the Fermi level) and at a low temperature. The former means that the system is in dynamical equilibrium (if there is spatial variation in the chemical potential the system would respond by diffusion of electrons from regions of higher to lower chemical potential until the chemical potential is constant), whilst the latter implies that the system acts as a Fermi gas. It can be shown that under the Fermi-Thomas approximations the screening length due to the conduction electrons in the metal is:

$$\zeta = \sqrt{\frac{2\epsilon_0 E_F}{3e^2 n_0}}, \tag{4.5}$$

where n_0 is the density of conduction electrons in the metallic solid. To give a decent approximation for the Fermi-Thomas screening length in the FeTe system it is assumed that the Fermi energy is in the order of $\mathrm{E}_F \sim 1\mathrm{eV}$ and $\mathrm{n}_0 \sim 1 \times 10^{28}\mathrm{m}^{-3}$. The latter comes from the fact that the interstitial Fe atoms are assumed to contribute all of their valence electrons in the 4s and 3d subshells and the interstitial iron concentration is low, i.e., y ~0.10. Using the formula in eq. 4.5, the Thomas-Fermi screening length due to the conduction electrons in FeTe is predicted to be 0.6Å, which appears to be a reasonable value given the bond lengths in the crystal.

Using the parameters described in this model of the crystal field splitting, the energies of the 3d orbitals in Fe can therefore be determined. The parameters are not fixed—they are just an educated guess about the system—and are optimised in order to produce the same spin state (S=1) as has been observed in experiments carried out on the $\mathrm{Fe}_{1+y}\mathrm{Te}$ crystals.

The crystal field splitting is, however, only one part of the calculations for determining the single-ion ground state and the energy scale inputs to the Rayleigh-Schrödinger model for the magnetic interactions in the FeTe system. The obverse to this is calculating the Hund's rule coupling parameters that come about due to the Coulomb repulsion between electrons in the same atom. There are two contributions to the Hund's coupling due to the wavefunction of the electrons being comprised of a spatial part and a spin part: it energetically favours electrons to occupy different electronic orbitals but also favours electrons to have their spins aligned, i.e., be parallel to each other. It is possible to derive these two different contributions by considering the Hartree equation (a Hamiltonian for a multi-electron atom that includes the repulsion between orbital electrons) and taking a first order perturbation in the electronic Coulomb repulsion between electrons in the valence orbitals:

$$U_{\alpha\beta} = \int\int d^3\mathbf{r} d^3\mathbf{r}' |\psi_\alpha(\mathbf{r})|^2 \frac{e^2}{4\pi\epsilon_0|\mathbf{r}-\mathbf{r}'|} |\psi_\beta(\mathbf{r}')|^2, \tag{4.6}$$

where $\mathrm{U}_{\alpha\beta}$ is the matrix of overlap integrals indexed by the same orbitals as in eq. 4.2 and the integration is over both $\mathbf{r}$ and $\mathbf{r}'$. This formula reveals that if the two orbitals are the same, then the energy is larger due to the increased overlap between the electron wavefunctions. Thus, having a pair of electrons in different orbitals is electronically favoured compared to having them fill one orbital. However, this only captures one element of Hund's rule and does not lead to the electrons in different orbitals favouring an aligned spin projection between two electrons. This can be extracted from the equation if the problem is treated quantum mechanically in the second quantisation with field operators, $\phi^\dagger_\sigma(\mathbf{r})$, that create an electron with a spin projection σ at the position $\mathbf{r}$. These field operators can be expanded as a product of the hydrogenic-like wavefunctions and an annihilation operator that destroys a single-particle in an orbital γ with spin projection σ:

$$\phi_\sigma(\mathbf{r}) = \Sigma_\gamma \psi_\gamma(\mathbf{r}) c_{\gamma\sigma}. \tag{4.7}$$

With normal ordering of the creation and annihilation operators (also called Wick ordering), the Hund's rule coupling becomes a sum over the field operators:

$$U = \Sigma_{\sigma\rho} \int \int d^3\mathbf{r} d^3\mathbf{r}' \frac{e^2}{4\pi\epsilon_0 |\mathbf{r} - \mathbf{r}'|} \phi^\dagger_\sigma(\mathbf{r}) \phi^\dagger_\rho(\mathbf{r}') \phi_\sigma(\mathbf{r}) \phi_\rho(\mathbf{r}'), \tag{4.8}$$

which can be expanded into the single-particle spatial wavefunctions and the creation and annihilation operators to produce:

$$U = \Sigma_{\sigma\rho\alpha\beta\delta\gamma} \int \int d^3\mathbf{r} d^3\mathbf{r}' \frac{e^2}{4\pi\epsilon_0 |\mathbf{r} - \mathbf{r}'|} \psi^*_\alpha(\mathbf{r}) \psi^*_\beta(\mathbf{r}') \psi_\delta(\mathbf{r}') \psi_\gamma(\mathbf{r}) c^\dagger_{\alpha\sigma} c^\dagger_{\beta\sigma} c_{\delta\sigma} c_{\gamma\sigma}. \tag{4.9}$$

Taking the expectation of this Coulomb interaction operator for states in which electrons are in different orbitals and spins are either aligned parallel or anti-parallel, it can be demonstrated that:

$$\langle \mu \uparrow, \lambda \downarrow | U | \mu \uparrow, \lambda \downarrow \rangle = 2U_{\mu\lambda\lambda\mu}, \tag{4.10}$$

$$\langle \mu \uparrow, \lambda \uparrow | U | \mu \uparrow, \lambda \uparrow \rangle = 2U_{\mu\lambda\lambda\mu} - 2U_{\mu\lambda\mu\lambda}. \tag{4.11}$$

The energy difference between the electrons' spins being parallel and anti-parallel is therefore $2\mathrm{U}_{\mu\lambda\lambda\mu}$. This is what one calls the "exchange term" and can be compared with the Hund's rule coupling from the Heisenberg model of quantum spins:

$$-J_H \mathbf{S}_1 \cdot \mathbf{S}_2, \tag{4.12}$$

where the J_H can be equated with the above to arrive at:

$$J_H = \int \int d^3\mathbf{r} d^3\mathbf{r}' \frac{e^2}{2\pi\epsilon_0 |\mathbf{r} - \mathbf{r}'|} \psi^*_\mu(\mathbf{r}) \psi^*_\lambda(\mathbf{r}') \psi_\mu(\mathbf{r}') \psi_\lambda(\mathbf{r}). \tag{4.13}$$

What is the energy splitting due to the exchange term for the singlet and triplet states, i.e., states with total spin S=0 and S=1 respectively? The singlet and triplet states for a two-electron system are the following:

$$\begin{aligned} |0,0\rangle &= \frac{1}{\sqrt{2}}(|\uparrow\downarrow\rangle - |\downarrow\uparrow\rangle) \\ |1,-1\rangle &= |\downarrow\downarrow\rangle \\ |1,0\rangle &= \frac{1}{\sqrt{2}}(|\uparrow\downarrow\rangle + |\downarrow\uparrow\rangle) \\ |1,1\rangle &= |\uparrow\uparrow\rangle \end{aligned} \tag{4.14}$$

and so taking the expectation value of eq. 4.12 would reveal the energy splitting of the singlet and triplet spin states. This can be seen by using the fact that the total spin is $\mathbf{S} = \mathbf{s}_1 + \mathbf{s}_2$ is the sum of the two spins on the two electronic orbitals and the operator relation that $\mathbf{S}^2 = \mathbf{s}_1^2 + \mathbf{s}_2^2 + 2\mathbf{s}_1 \cdot \mathbf{s}_2$. The eigenvalue of $\mathbf{S}^2$ is 0 and 1 for the singlet and triplet states respectively. Meanwhile the eigenvalue of the individual spin operators, $\mathbf{s}_1$ and $\mathbf{s}_2$ is $-\frac{1}{4}$ when the electron is in the state with spin down and $\frac{3}{4}$ when the electron is in the state with spin up. This leads to the energy of the singlet state being increased by $\frac{J_H}{4}$ and the energy of the triplet state being reduced by $\frac{3J_H}{4}$. The effect of this is to energetically favour the triplet state over the singlet state in the Fe^{2+} ion. Ideally, the Coulomb screening from other electrons in the many-electron atom should also be included in eq. 4.13, but this is quite complicated and it is assumed that this formula gives at least an approximate numerical value to the exchange interaction due to the symmetry of the singlet and triplet electronic wavefunctions in the crystal.

This section has so far demonstrated how to calculate the relative energy shifts associated with the crystal field splitting and Hund's rule coupling, but it has neglected to determine the absolute energy scales associated with the atomic orbitals on the Fe and Te atoms. A crude model has been used for calculating the crystal field splitting that parameterises the amount of shielding by a single parameter, Z_{eff}, that does not take into account the spatial dependence of the effects of inner core electrons. This simple model of nuclear core shielding can also be used for calculating the absolute energies of the atomic orbitals in the 3d and 5p valence subshells of the Fe and Te atoms respectively. Using a simplistic assumption that the hydrogen-like Schrödinger equation can well-describe the Coulomb interactions between the nucleus of the Fe and Te atoms and their valence electrons, the energy of a subshell can be expressed as:

$$E_{Z_{\text{eff}},n} = -\frac{\mu Z_{\text{eff}}^2 e^4}{32\pi^2 \epsilon_0 \hbar^2 n^2}, \tag{4.15}$$

where n is the principle quantum number (n=3 and 5 for the valence electrons in Fe and Te respectively) and μ is the effective mass of the nucleus-electron system. Since the nucleus is much more massive than the electrons, it is often possible to assume that $\mu \sim m_e$. As was aforementioned, the effective nuclear charge is an input parameter to the model and so its value could have a big impact on the magnitude given to the virtual electron hopping processes in the Rayleigh-Schrödinger perturbative model. The relative energies of the orbitals on both the Fe and Te atoms will, as will be shown in Sec. 4.3, be introduced into the Hamiltonian via the denominator of a resolvent operator meaning that a larger energy difference between the orbitals will reduce the magnitude of the elements of the Hamiltonian matrix that describes the magnetic interactions within the FeTe system. Hence, the values given for s in eq. 4.4 for either of the two species in FeTe ought to be optimised such that the single-ion spin state reflects the experimentally observed value (i.e., it produces a crystal field splitting that leads to the S=1 spin state for the Fe^{2+} ions) and leads to the expected magnetic ground state. Nonetheless, as was suggested in Sec. 4.3 an appropriate starting point would be to use Slater's rules for calculating an estimate of Z_{eff} for both Fe and Te. In the case of Fe^{2+} this was shown to be 6.25, but what about the shielding for the electrons in the 5p orbital of Te^{2-} (the atom accepts two electrons from the Fe atoms to form a negative ion)? The nuclear core shielding of the electrons in the fully occupied 5p orbitals of Te^{2-} leads to s=47.25. The nuclear charge of the Te atom is Z=52 which leads to the effective nuclear core charge as seen by an electron occupying the 5p orbital of Z_{eff}=4.75. Putting these values into eq. 4.15 leads to E=-59.0eV for the 3d orbitals in Fe^{2+} and E=-12.27eV for the 5p orbitals in Te^{2-}.

All the tools for determining the energies of different single-ion electronic configurations have now been explained including how to include the Jahn-Teller distortion in the crystal field splitting and the favourability of having electrons in different 3d orbitals whilst also having an energy incentive to have their spin projections aligned. These tools will be utilised in later sections in order to determine the single-ion ground state and can also be parameterised such that the degree of the lattice distortion can be taken into account when calculating these energy scales, which is important given that the emergence of the bicollinear double stripe antiferromagnetism happens at or close to the structural phase transition temperature indicating the possible link between the two phenomena.

4.3 Rayleigh-Schrödiner virtual hopping model

In the preceding section, the first order perturbative overlap integrals for calculating both the crystal field splitting and Hund's rule coupling were explained in detail. This was critical because it helps the reader to understand how inputs related to the structural geometry and the electronic order lead to different energies and therefore different single-ion ground states for the Fe^{2+} ions. In developing a perturbative model for the magnetic interactions these are also important as the zeroth order eigenstates and eigenvalues, which are used in subsequent contributions to higher order energy corrections to the zeroth order eigenstates and eigenvalues. In this section, it will describe how the Rayleigh-Schrödinger perturbative model for the virtual electron hoppings between the Fe and Te ions in FeTe crystals is constructed. This is an application of the theoretical framework that was developed by Ingvar Lindgren who developed this perturbative approach for understanding interactions within quantum mechanical multi-configurational systems [89]. It can be applied to many different quantum mechanical systems and is not restricted to its use for the theoretical prediction of the magnetic interactions present in a condensed matter system, such as a single crystal with localised magnetic order.

How can the Rayleigh-Schrödinger perturbative model developed by Ingvar Lindgren be applied to the FeTe system? In order to understand how this can be achieved, it is apt to first understand what it means by a multi-configurational model. This means that the so-called "model space" contains multiple states with energies within reach of the expected zeroth order ground state of the quantum mechanical system. In the case of the present work, the model space includes the configuration of states with the e_g orbitals fully occupied and two electrons in the t_{2g} orbitals such that one of those electrons is occupying the lower energy d_{xy} orbital, which is energetically favoured by crystal field splitting, and the other electron is occupying either of the d_{Xz} or d_{Yz} orbitals which have had their degeneracy lifted by a Jahn-Teller distortion as aforementioned in this chapter. Higher order terms in the Rayleigh-Schrödinger model's Hamiltonian involve the second and fourth order virtual hopping processes that exchange electrons between neighbouring Fe and Te atoms in the crystal structure. Intermediary states involve a configuration of electrons in an Fe-Te-Te-Fe quartet that is outwith the model space of the multi-configurational model. The virtual electron hopping could therefore be imagined as involving an operator that projects the states in the model space into another set of accessible states called the orthogonal space and thereafter return it to states to the model space. The difference between the second order and fourth order virtual hopping processes is the number of virtual electron hopping processes within which the state must return to the model space for the system.

The effective Hamiltonian operator in the Rayleigh-Schrödinger perturbative model is constructed to include the plurality of possible second and fourth order virtual hopping processes as well as the zeroth order Hamiltonian acting on the model space that gives the zeroth order eigenvalues. Given a model space with a dimensionality of n, the effective Hamiltonian produced by the Rayleigh-Schrödinger model is an $n \times n$ matrix that is not necessarily Hermitian and whose eigenvalues are always real (the effective Hamiltonian only acts on the states within the model space). However, owing to the fact that the Hamiltonian for this system is expected to have time-reversal symmetry (the Hamiltonian ought to be invariant to flipping all the spins whilst maintaining the same orbital occupations) this effective Hamiltonian can be transformed to an Hermitian Hamiltonian with the same eigenvalues as the original Hamiltonian but with different eigenstates using the theoretical framework developed by des Cloizeaux [92].

The choice of model space for the FeTe system included the six S=1 spin states on each Fe atom (three triplet states multiplied by two degrees of freedom for occupying either the d_{Xz} or d_{Yz} orbital within the t_{2g} subgroup) with a Hilbert product space spanning two Fe atoms which share two neighbouring Te atoms, thus forming a space of thirty-six states (Fig 4.3 shows the electron configurations of the single Fe^{2+} ions in the model space). As the name suggests, the orthogonal space includes all those states in the Hilbert space but orthogonal to those in the model space. They can be reached by virtual hopping processes from the model space that can be represented by an operator acting on the model space and projecting it into the orthogonal space, $\hat{\Omega}$, or vice versa in projecting from the orthogonal space into the model space, $\hat{P}$.

In this work, both the second order and fourth order virtual hopping processes are considered: the former annihilates a valence electron in one of the orbitals in the 5p subshell on either of the two neighbouring Te atoms and creates an electron with the same spin in either one of the t_{2g} orbitals in a neighbouring Fe atom, which is followed by one of the electrons in the Fe t_{2g} orbitals hopping back to the vacated Te orbital bringing the final state back into the model space; the latter includes two pairs of virtual hopping processes which also brings the final state back inside the model space (the intermediary states in the fourth order virtual hopping processes may include a return to the model space after two virtual hops). Only including the non-zero terms in the effective real Hamiltonian from Ingvar Lindgren's work [89], the effective Hamiltonian can be written as:

$$\hat{H}_{\text{eff}} = \hat{P}\hat{H}_0\hat{P} + \hat{P}\hat{V}\hat{\Omega}, \tag{4.16}$$

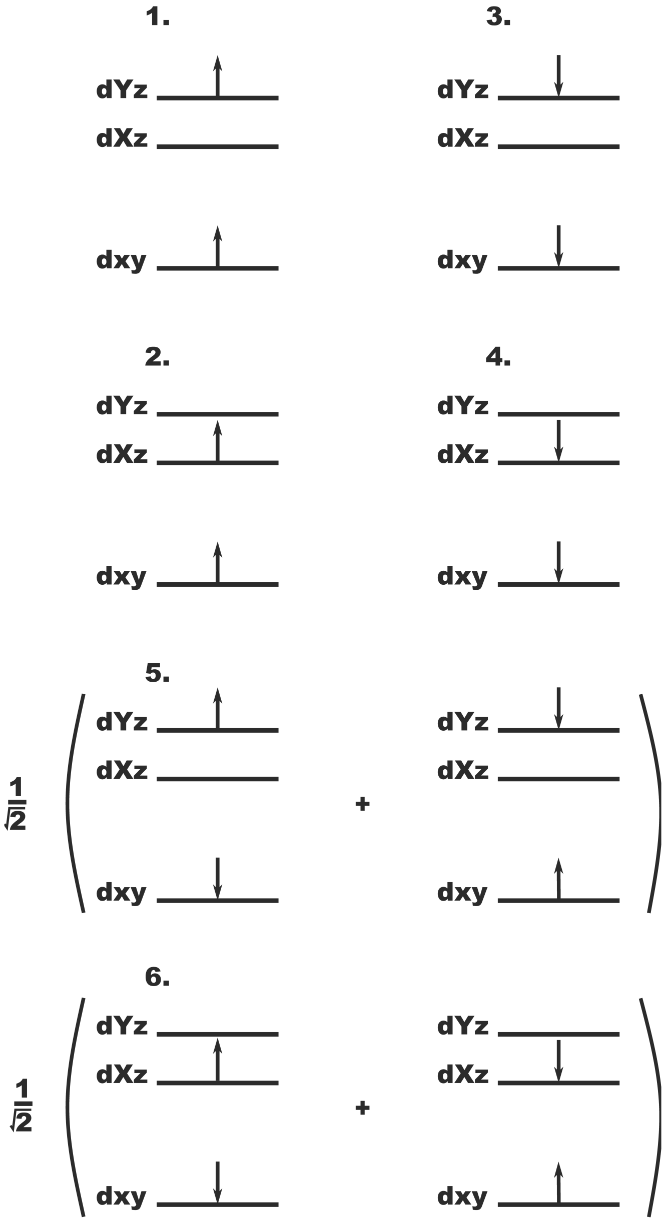


FIGURE 4.3: The single-ion S=1 spin states for the Fe^{2+} ions in the FeTe system. These triplet spin states have m $\in \{-1, 0, 1\}$.

where $\hat{\Omega}$ includes terms that produce second and fourth order virtual hopping processes:

$$\hat{\Omega} = \hat{R}\hat{V}\hat{P} + \hat{R}\hat{V}\hat{R}\hat{V}\hat{R}\hat{V} - \hat{R}^2\hat{V}\hat{R}\hat{V}\hat{P}\hat{V}\hat{P}. \tag{4.17}$$

$\hat{V}$ is an operator that annihilates an electron in one orbital and creates an electron of the same spin projection in another orbital. $\hat{R}$ is a resolvent operator that acts on the multi-configurational states in the orthogonal space and is defined as:

$$\hat{R} = \sum_{r \in D} \frac{|r\rangle \langle r|}{E_0 - E_r}, \tag{4.18}$$

where D are the set of states in the orthogonal space and E_0 is the energy of the degenerate states within the model space. Since the resolvent operator only ever acts on the states within the orthogonal space, which have different energies compared with those of the model space, it will never explode due to a vanishing denominator. The fourth order terms are split into two different processes depending on whether any of the intermediate states are degenerate with one of the states in the orthogonal space. Any process with such degeneracy of intermediate states leads to the effective Hamiltonian having the square of the resolvent of the final intermediate state. Hence, the Rayleigh-Schrödinger effective Hamiltonian is essentially one of both counting up the numbers of virtual hopping processes allowed by the symmetry inherent in the hopping integrals and by Pauli's exclusion principle for each orbital in both the Fe and Te atoms. It is important to also understand that a sign change can take place owing to the ordering of the creation operators when creating a state from a vacuum state (see Sec. 4.3.1 for detailed discussion).

However, there is a problem associated with eq. 4.16 and 4.17 in that the effective Hamiltonian is not Hermitian and therefore does not respect the time-reversal symmetry of the physics of this quantum system. des Cloizeaux deals with this by transforming the effective Hamiltonian in eq. 4.16 such that the eigenstates of this new transformed Hamiltonian are as close to the original eigenstates and produces the same eigenvalues. The following will briefly detail the basics of the theoretical framework developed by des Cloizeaux and how it was shown to be able to transform a non-Hermitian Brillouin-Wigner perturbative Hamiltonian (the difference between the Brillouin-Wigner and the Rayleigh-Schrödinger perturbative theories is that the perturbed energies in the former approach are calculated via an iterative method) into an Hermitian Hamiltonian with orthogonal eigenstates and real energy eigenvalues. It starts from asking whether a constant operator, h, can have the same eigenvectors and eigenvalues as the Hamiltonian, $H(E)$, to the following Schrödinger equation:

$$[E - e - H(E)] |\phi\rangle = 0, \tag{4.19}$$

where $|\phi\rangle$ is the eigenvector, e is a constant and $H(E)$ is an Hermitian operator that is a function of E. In certain cases, there does exist an Hermitian operator, h, for which the eigenvector and eigenvalues are the same as for the Hamiltonian, $H(E)$, but in general this is not the case. It shows that there is a transformation, g, to an orthogonal basis set, $|\psi\rangle$:

$$|\psi\rangle = g\,|\phi\rangle\,, \tag{4.20}$$

and that this transforms the eq. 4.19 into the following:

$$[E - e - K]\,|\psi\rangle = 0, \tag{4.21}$$

where the transformed operator, $K = ghg^{-1}$, is a function of the original constant operator. The operator g can be chosen in such a way to ensure that the transformed operator, K, is Hermitian, and therefore has both orthogonal eigenvectors and real eigenvalues. Moreover, K is the Hamiltonian used to describe the physics of the multi-level quantum mechanical system. It can be demonstrated that for a Brillouin-Wigner perturbation of the Hamiltonian, H_0, by a small perturbative potential, V, the operator, h, expanded to third order in the perturbative series is written as:

$$h = \hat{P}\hat{V}\hat{P} + \hat{P}\hat{V}\hat{R}\hat{V}\hat{P} + \hat{P}\hat{V}\hat{R}\hat{V}\hat{R}\hat{V}\hat{P} - \hat{P}\hat{V}\hat{R}^2\hat{V}\hat{P}\hat{V}\hat{P} + O(\hat{V}^4), \tag{4.22}$$

and the transformation leading to orthogonal eigenvectors (the non-Hermiticity of h comes from the last term shown in the expansion) leads to the following form for K:

$$K = \hat{P}\hat{V}\hat{P} + \hat{P}\hat{V}\hat{R}\hat{V}\hat{P} + \hat{P}\hat{V}\hat{R}\hat{V}\hat{R}\hat{V}\hat{P} - \frac{1}{2}\hat{P}\hat{V}\hat{R}^2\hat{V}\hat{P}\hat{V}\hat{P} - \frac{1}{2}\hat{P}\hat{V}\hat{P}\hat{V}\hat{R}^2\hat{V}\hat{P} + O(\hat{V}^4). \tag{4.23}$$

This operator is clearly Hermitian owing to an effective averaging of the terms with their Hermitian conjugates. It should be noted that the form of h in eq. 4.22 is the $\hat{P}\hat{V}\hat{\Omega}$ in eq. 4.17 with a truncation to the third order in the perturbation expansion but including terms that would be forbidden by symmetry in the FeTe model. If h is instead constructed to include all expansion terms up to fourth order, then it can be demonstrated by taking the Hermitian conjugates of all operators that the transformed Hermitian operator is:

$$\begin{aligned}K =&\hat{P}\hat{V}\hat{P}+\hat{P}\hat{V}\hat{R}\hat{V}\hat{P}+\hat{P}\hat{V}\hat{R}\hat{V}\hat{R}\hat{V}\hat{P}-\frac{1}{2}\hat{P}\hat{V}\hat{R}^2\hat{V}\hat{P}\hat{V}\hat{P}\\ &-\frac{1}{2}\hat{P}\hat{V}\hat{P}\hat{V}\hat{R}^2\hat{V}\hat{P}+\hat{P}\hat{V}\hat{R}^3\hat{V}\hat{P}\hat{V}\hat{P}\hat{V}\hat{P}\\ &-\frac{1}{2}\hat{P}\hat{V}\hat{R}^2\hat{V}\hat{P}\hat{V}\hat{R}\hat{V}\hat{P}-\frac{1}{2}\hat{P}\hat{V}\hat{R}^2\hat{V}\hat{R}\hat{V}\hat{P}\hat{V}\hat{P}\\ &-\frac{1}{2}\hat{P}\hat{V}\hat{R}\hat{V}\hat{R}^2\hat{V}\hat{P}\hat{V}\hat{P}+\hat{P}\hat{V}\hat{R}\hat{V}\hat{R}\hat{V}\hat{R}\hat{V}\hat{P}\\ &-\frac{1}{2}\hat{P}\hat{V}\hat{P}\hat{V}\hat{R}^2\hat{V}\hat{R}\hat{V}\hat{P}-\frac{1}{2}\hat{P}\hat{V}\hat{P}\hat{V}\hat{R}\hat{V}\hat{R}^2\hat{V}\hat{P}\\ &-\frac{1}{2}\hat{P}\hat{V}\hat{R}\hat{V}\hat{P}\hat{V}\hat{R}^2\hat{V}\hat{P}+\hat{P}\hat{V}\hat{R}\hat{V}\hat{P}\hat{V}\hat{R}^3\hat{V}\hat{P}+O(\hat{V}^5)\end{aligned}\tag{4.24}$$

Keeping the perturbative expansion to fourth order terms but removing those that are zero because of the symmetry of the problem (terms with the sequence of operators $\hat{P}\hat{V}\hat{P}$ are necessarily zero because it is impossible for an electron to virtually hop from one of the model space states and remain inside the model space), an Hermitian effective Hamiltonian, H, describing the same physics as the non-Hermitian effective Hamiltonian in eq. 4.16, can be formed and is expressed as:

$$\begin{aligned}H =&\hat{P}\hat{H}_0\hat{P}+\hat{P}\hat{V}\hat{R}\hat{V}\hat{P}-\frac{1}{2}\hat{P}\hat{V}\hat{R}^2\hat{V}\hat{P}\hat{V}\hat{R}\hat{V}\hat{P}\\ &+\hat{P}\hat{V}\hat{R}\hat{V}\hat{R}\hat{V}\hat{R}\hat{V}\hat{P}-\frac{1}{2}\hat{P}\hat{V}\hat{R}\hat{V}\hat{P}\hat{V}\hat{R}^2\hat{V}\hat{P}+O(\hat{V}^5)\end{aligned}\tag{4.25}$$

However, this effective Hermitian Hamiltonian is still not appropriate for modelling the virtual exchange of electrons in the FeTe system owing to the non-degeneracy of the states in the model space due to both the crystal field splitting and Jahn-Teller effect (the first lifting the degeneracy of the 3d orbitals in the Fe^{2+} ions generating the t_{2g} and e_g subgroup and the latter splitting these subgroups further) and the Hund's rule coupling including the exchange interaction. The non-degenerate model space is important because this case needs to be treated differently in the Rayleigh-Schrödinger perturbative approach (as opposed to the Brillouin-Wigner perturbative method which does not require a separate treatment). Using the non-degenerate case of the Rayleigh-Schrödinger perturbative model as presented by Ingvar Lindgren and applying the transformation as developed by des Cloiseaux, it can be demonstrated that the Hermitian effective Hamiltonian describing the virtual exchange of electrons between neighbouring Fe and Te atoms in the Fe-Te-Te-Fe quartets can be expressed as:

$$\begin{aligned}H =&\hat{P}\hat{H}_0\hat{P}+\hat{P}\hat{V}\hat{S}(\hat{V}\hat{P})+(\hat{P}\hat{V}\hat{S}(\hat{V}\hat{P}))^\dagger\\ &-\frac{1}{2}\hat{P}\hat{V}\hat{S}^2(\hat{V}\hat{P}\hat{V}\hat{S}(\hat{V}\hat{P}))-\frac{1}{2}(\hat{P}\hat{V}\hat{S}^2(\hat{V}\hat{P}\hat{V}\hat{S}(\hat{V}\hat{P}))^\dagger\\ &+\hat{P}\hat{V}\hat{S}(\hat{V}\hat{S}(\hat{V}\hat{S}(\hat{V}\hat{P})))+(\hat{P}\hat{V}\hat{S}(\hat{V}\hat{S}(\hat{V}\hat{S}(\hat{V}\hat{P}))))^\dagger\\ &-\frac{1}{2}\hat{P}\hat{V}\hat{S}(\hat{V}\hat{P}\hat{V}\hat{S}^2(\hat{V}\hat{P}))-\frac{1}{2}(\hat{P}\hat{V}\hat{S}(\hat{V}\hat{P}\hat{V}\hat{S}^2(\hat{V}\hat{P})))^\dagger+O(\hat{V}^5)\end{aligned}\tag{4.26}$$

where $\hat{S}^n(...)$ is the resolvent for the non-degenerate model space and is defined such that:

$$\langle r| \hat{S}^n(\hat{A}) |a\rangle = \frac{\langle r| \hat{A} |a\rangle}{(E_a - E_r)^n}, \tag{4.27}$$

where $|a\rangle$ is a state in the orthogonal space. This is the perturbative expansion used to model the second and fourth order virtual electron hopping within the FeTe single crystals and to understand what is driving the magnetic interactions (and ultimately whether the bicollinear double stripe antiferromagnetic order can be explained by the interaction between the spin and orbital degrees of freedom involving the d_{Xz} and d_{Yz} orbitals).

4.3.1 Convention for the ordering of creation operators

In this section, it will describe the construction of the quantum mechanical states in the model space and orthogonal space in a Fock basis and how the effective Hamiltonian in eq. 4.26 when using the canonical number representation (i.e., the operators are defined using fermionic creation and annihilation operators) can be operated on these states. Since it is a many-body electron quantum mechanical problem, the multiple particle states, which are fermions, must lead to indistinguishability of the individual particles and have an anti-symmetric wavefunction because it must obey Pauli's exclusion principle—that no two fermions can occupy the same quantum state. The basis states in the model space are comprised of three valence electrons in the t_{2g} subgroup for each of the two Fe atoms and fully occupied 5p subshells in each of the two neighbouring Te atoms. Using Slater's determinant for an anti-symmetric wavefunction, a many-body wavefunction can be constructed from the product of the individual electron wavefunctions, $|\lambda_m\rangle$:

$$|n^{\lambda_1}, n^{\lambda_2}, ..., n^{\lambda_{23}}, n^{\lambda_{24}}\rangle = \frac{1}{\sqrt{18!}} \sum_{\mathcal{P}} (-1)^{\mathcal{P}} |\lambda_{\mathcal{P}(1)}\rangle \otimes |\lambda_{\mathcal{P}(2)}\rangle ... \otimes |\lambda_{\mathcal{P}(17)}\rangle \otimes |\lambda_{\mathcal{P}(18)}\rangle, \tag{4.28}$$

where $\mathcal{P}$ is the number of transpositions from a normal ordering of the eighteen individual fermionic wavefunctions, $\mathcal{P}(i)$ is the i^{th} element of the set formed by a permutation of individual electron states, and n^{λ_m} is the number of particles in the state $|\lambda_m\rangle$.

It ought to be noted that there are 24 possible individual quantum states (6 possible quantum states for electrons to occupy in the t_{2g} subgroup in the Fe atoms and the same number of states in the 5p subshell in the Te atoms) but there are only a maximum of 18 total electrons occupying these states. This explains why there are 24 numbers representing the number of states occupying a particular quantum state on the left hand side of this equation but only 18 numbers representing the states occupied by the available electrons to the quartet of Fe-Te-Te-Fe on the right hand side. The convenience of the second quantisation is that the many-body electron state can be defined by a product of creation operators, $a^\dagger_{\lambda_m}$, acting on a quantum vacuum state, Ω:

$$|n^{\lambda_1}, n^{\lambda_2}, ..., n^{\lambda_{23}}, n^{\lambda_{24}}\rangle = (a^\dagger_{\lambda_1})^{n^{\lambda_1}}(a^\dagger_{\lambda_2})^{n^{\lambda_2}}...(a^\dagger_{\lambda_{17}})^{n^{\lambda_{17}}}(a^\dagger_{\lambda_{18}})^{n^{\lambda_{18}}}|\Omega\rangle \,. \tag{4.29}$$

The creation operator physically creates a particle in the state $|\lambda_m\rangle$ whilst its conjugate, the annihilation operator, a_{λ_m}, destroys the particle, and to prevent any contradiction with the anti-symmetry of the wavefunction they must obey the fermionic commutation relations:

$$\begin{aligned} \{a_{\lambda_m}, a^\dagger_{\lambda_n}\} &= \delta_{mn} \\ \{a_{\lambda_m}, a_{\lambda_n}\} &= 0 \\ \{a^\dagger_{\lambda_m}, a^\dagger_{\lambda_n}\} &= 0 \end{aligned} \tag{4.30}$$

where $\{A, B\} = AB + BA$. These creation and annihilation operators can also be used to describe the virtual exchange of electrons of states in both the model and orthogonal space with the perturbative potential in eq. 4.26 representing the tight-binding hopping potentials. This particle exchange requires the simultaneous annihilation of an electron in one individual quantum state, $|\lambda_m\rangle$, and the creation of an electron in a different state, $|\lambda_m\rangle$:

$$\hat{V}_{nm} = a^\dagger_{\lambda_n} V_{nm} a_{\lambda_m}, \tag{4.31}$$

where V_{nm} is the tight-binding hopping integrals defined in eq. 4.36. For instance, to describe the virtual exchange of electrons in the FeTe system it would be convenient to use the following notation for the creation operators:

$$\begin{aligned} &c^\dagger_{\alpha\beta\sigma}, \quad \alpha \in \{1,\, 2\},\, \beta \in \{\text{xy, Xz, Yz}\},\, \sigma \in \{\uparrow, \downarrow\} \\ &d^\dagger_{\alpha\beta\sigma}, \quad \alpha \in \{1,\, 2\},\, \beta \in \{\text{x, y, z}\},\, \sigma \in \{\uparrow, \downarrow\} \end{aligned} \tag{4.32}$$

where $c^\dagger_{\alpha\beta\sigma}$ creates a spin-up or spin-down electron in one of the t_{2g} orbitals in either of the Fe atoms and $d^\dagger_{\alpha\beta\sigma}$ creates a spin-up or spin-down electron in of the 5p orbitals in either of the Te atoms. When combining higher order virtual electron hopping processes the ordering of the creation and annihilation operators is not important, but it is critical to be consistent with a chosen convention because of the anti-commutative property of fermionic operators. In the FeTe model, the convention for ordering the creation and annihilation operators follows the general rules set out below:

- from left to right they act on Fe atom 1 followed by Te atom 1, then Te atom 2 and then Fe atom 2;
- within an Fe atom the operators act on d_{xy} orbitals first and d_{Yz} orbitals last;
- within a Te atom the operators act on p_x orbitals first and p_z orbitals last;
- a spin-down state is created before a spin-up state.

To demonstrate this with an example of one of the multi-electron quantum states situated within the model space in which a spin-up electron occupies the d_{xy} and d_{Xz} orbitals in both Fe atoms and both 2p subshells in the two Te atoms are fully occupied by six electrons, this can be represented by the following product of creation operators acting on the vacuum state, $|\Omega\rangle$:

$$\begin{aligned}&|S_1 = 1, m_1 = 1; S_2 = 1, m_2 = 1; d_{Xz}, d_{Xz}\rangle = \hat{c}^\dagger_{1,Xz,\uparrow}\hat{c}^\dagger_{1,xy,\uparrow}\hat{c}^\dagger_{1,xy,\downarrow}\\ &\hat{d}^\dagger_{1,z,\uparrow}\hat{d}^\dagger_{1,z,\downarrow}\hat{d}^\dagger_{1,y,\uparrow}\hat{d}^\dagger_{1,y,\downarrow}\hat{d}^\dagger_{1,x,\uparrow}\hat{d}^\dagger_{1,x,\downarrow}\hat{d}^\dagger_{2,z,\downarrow}\hat{d}^\dagger_{2,y,\uparrow}\hat{d}^\dagger_{2,y,\downarrow}\hat{d}^\dagger_{2,x,\uparrow}\hat{d}^\dagger_{2,x,\downarrow}\\ &\hat{c}^\dagger_{2,Xz,\uparrow}\hat{c}^\dagger_{2,xy,\uparrow}\hat{c}^\dagger_{2,xy,\downarrow}\,|\Omega\rangle\,,\end{aligned} \tag{4.33}$$

where $|S_1 = 1, m_1 = 1; S_2 = 1, m_2 = 1; d_{Xz}, d_{Xz}\rangle$ indicates that this many-electron quantum state is comprised of two Fe^{2+} ions with S=1 spins, spin projections of m=1 and with an electron on each of the ions occupying the d_{Xz} orbitals. This notation helps reveal the fact that the effective Hamiltonian acting on the states within the model space can be constructed as a 32×32 matrix (each 16×16 sub-matrix represents all the spin states for a fixed combined orbital state): 3 degrees of freedom for the spin projection and 2 degrees of freedom for the orbital occupancy for each Fe^{2+} in the pair giving 32 separate spin and orbital states.

It is also possible to consider how a fourth order virtual hopping process can be represented by a combination of creation and annihilation operators for the two Fe and Te atoms. Restricting the virtual hopping processes to interatomic electron hops between neighbouring Fe and Te atoms, this fourth order virtual hopping process can be represented by an operator with four creation and four annihilation operators. The sign change introduced by acting this virtual exchange operator on a many-electron quantum state will then depend on the ordering of the individual creation and annihilation operators. For instance, consider the following fourth order virtual electron hopping operator (ignoring the magnitudes of the tight-binding integrals that would explain the processes' contribution to the overall effective Hamiltonian):

$$\hat{V}^4 = \hat{d}^\dagger_{1,x,\uparrow}\hat{c}_{2,Xz,\uparrow}\hat{d}^\dagger_{1,y,\uparrow}\hat{c}_{1,Xz,\uparrow}\hat{c}^\dagger_{2,Yz,\uparrow}\hat{d}_{1,y,\uparrow}\hat{c}^\dagger_{1,Yz,\uparrow}\hat{d}_{1,x,\uparrow}, \tag{4.34}$$

which can be described as comprising the following sequence of virtual electron exchanges:

- a spin-up electron hops from the p_x orbital in one Te atom into the d_{Yz} orbital of the first Fe atom;
- another spin-up electron hops from the p_y orbital in the same Te atom into the d_{Yz} of the second Fe atom;
- the spin-up electron in the d_{Xz} orbital in the first Fe atom hops into the partially occupied p_y orbital of the Te atom just vacated;
- the spin-up electron in the d_{Xz} orbital in the second Fe atom hops into the partially occupied p_x orbital of the Te atom similarly just vacated.

The overall effect of this sequence of virtual electron hopping is that the total spin of this many-electron quantum system remains the same, but that both of the Fe atoms have spin-up electrons occupying the d_{Yz} (which is a higher energy level compared with the d_{Xz} orbital owing to the Jahn-Teller effect) instead of the d_{Xz} orbital. If this fourth order electron hopping operator is acted on the many-electron quantum state shown in eq. 4.33, a minus sign is introduced owing to the order in which the creation and annihilation operators are in the sequence and having to move the operators around using the anti-commutativity rules for fermionic particles:

$$
\begin{aligned}
&\hat{V}^4 \left|S_1 = 1, m_1 = 1; S_2 = 1, m_2 = 1; d_{Xz}, d_{Xz}\right\rangle = \\
&\quad \hat{d}^\dagger_{1,x,\uparrow}\hat{c}_{2,Xz,\uparrow}\hat{d}^\dagger_{1,y,\uparrow}\hat{c}_{1,Xz,\uparrow}\hat{c}^\dagger_{2,Yz,\uparrow}\hat{d}_{1,y,\uparrow}\hat{c}^\dagger_{1,Yz,\uparrow}\hat{d}_{1,x,\uparrow} \\
&\quad \hat{c}^\dagger_{1,Xz,\uparrow}\hat{c}^\dagger_{1,xy,\uparrow}\hat{d}^\dagger_{1,z,\uparrow}\hat{d}^\dagger_{1,z,\downarrow}\hat{d}^\dagger_{1,y,\uparrow}\hat{d}^\dagger_{1,y,\downarrow}\hat{d}^\dagger_{1,x,\uparrow}\hat{d}^\dagger_{1,x,\downarrow}\hat{d}^\dagger_{2,z,\downarrow}\hat{d}^\dagger_{2,y,\uparrow}\hat{d}^\dagger_{2,y,\downarrow}\hat{d}^\dagger_{2,x,\uparrow}\hat{d}^\dagger_{2,x,\downarrow}\hat{c}^\dagger_{2,Xz,\uparrow}\hat{c}^\dagger_{2,xy,\uparrow}\left|\Omega\right\rangle \\
&\quad = \hat{d}^\dagger_{1,x,\uparrow}\hat{c}_{2,Xz,\uparrow}\hat{d}^\dagger_{1,y,\uparrow}\hat{c}_{1,Xz,\uparrow}\hat{c}^\dagger_{2,Yz,\uparrow}\hat{d}_{1,y,\uparrow}\hat{c}^\dagger_{1,Yz,\uparrow}\hat{c}^\dagger_{1,Xz,\uparrow} \\
&\quad \hat{c}^\dagger_{1,xy,\uparrow}\hat{d}^\dagger_{1,z,\uparrow}\hat{d}^\dagger_{1,z,\downarrow}\hat{d}^\dagger_{1,y,\uparrow}\hat{d}^\dagger_{1,y,\downarrow}\hat{d}_{1,x,\uparrow}\hat{d}^\dagger_{1,x,\uparrow}\hat{d}^\dagger_{1,x,\downarrow}\hat{d}^\dagger_{2,z,\downarrow}\hat{d}^\dagger_{2,y,\uparrow}\hat{d}^\dagger_{2,y,\downarrow}\hat{d}^\dagger_{2,x,\uparrow}\hat{d}^\dagger_{2,x,\downarrow}\hat{c}^\dagger_{2,Xz,\uparrow}\hat{c}^\dagger_{2,xy,\uparrow}\left|\Omega\right\rangle \\
&\quad = -\hat{d}^\dagger_{1,x,\uparrow}\hat{c}_{2,Xz,\uparrow}\hat{d}^\dagger_{1,y,\uparrow}\hat{c}_{1,Xz,\uparrow}\hat{c}^\dagger_{2,Yz,\uparrow}\hat{c}^\dagger_{1,Yz,\uparrow}\hat{c}^\dagger_{1,Xz,\uparrow}\hat{c}^\dagger_{1,xy,\uparrow} \\
&\quad \hat{d}^\dagger_{1,z,\uparrow}\hat{d}^\dagger_{1,z,\downarrow}\hat{d}_{1,y,\uparrow}\hat{d}^\dagger_{1,y,\uparrow}\hat{d}^\dagger_{1,y,\downarrow}\hat{d}_{1,x,\uparrow}\hat{d}^\dagger_{1,x,\uparrow}\hat{d}^\dagger_{1,x,\downarrow}\hat{d}^\dagger_{2,z,\downarrow}\hat{d}^\dagger_{2,y,\uparrow}\hat{d}^\dagger_{2,y,\downarrow}\hat{d}^\dagger_{2,x,\uparrow}\hat{d}^\dagger_{2,x,\downarrow}\hat{c}^\dagger_{2,Xz,\uparrow}\hat{c}^\dagger_{2,xy,\uparrow}\left|\Omega\right\rangle \\
&\quad = -\hat{d}^\dagger_{1,x,\uparrow}\hat{c}_{2,Xz,\uparrow}\hat{d}^\dagger_{1,y,\uparrow}\hat{c}_{1,Xz,\uparrow}\hat{c}^\dagger_{1,Yz,\uparrow}\hat{c}^\dagger_{1,Xz,\uparrow}\hat{c}^\dagger_{1,xy,\uparrow}\hat{d}^\dagger_{1,z,\uparrow} \\
&\quad \hat{d}^\dagger_{1,z,\downarrow}\hat{d}_{1,y,\uparrow}\hat{d}^\dagger_{1,y,\uparrow}\hat{d}^\dagger_{1,y,\downarrow}\hat{d}_{1,x,\uparrow}\hat{d}^\dagger_{1,x,\uparrow}\hat{d}^\dagger_{1,x,\downarrow}\hat{d}^\dagger_{2,z,\downarrow}\hat{d}^\dagger_{2,y,\uparrow}\hat{d}^\dagger_{2,y,\downarrow}\hat{d}^\dagger_{2,x,\uparrow}\hat{d}^\dagger_{2,x,\downarrow}\hat{c}^\dagger_{2,Yz,\uparrow}\hat{c}^\dagger_{2,Xz,\uparrow}\hat{c}^\dagger_{2,xy,\uparrow}\left|\Omega\right\rangle \\
&\quad = \hat{d}^\dagger_{1,x,\uparrow}\hat{c}_{2,Xz,\uparrow}\hat{d}^\dagger_{1,y,\uparrow}\hat{c}^\dagger_{1,Yz,\uparrow}\hat{c}_{1,Xz,\uparrow}\hat{c}^\dagger_{1,Xz,\uparrow}\hat{c}^\dagger_{1,xy,\uparrow}\hat{d}^\dagger_{1,z,\uparrow} \\
&\quad \hat{d}^\dagger_{1,z,\downarrow}\hat{d}_{1,y,\uparrow}\hat{d}^\dagger_{1,y,\uparrow}\hat{d}^\dagger_{1,y,\downarrow}\hat{d}_{1,x,\uparrow}\hat{d}^\dagger_{1,x,\uparrow}\hat{d}^\dagger_{1,x,\downarrow}\hat{d}^\dagger_{2,z,\downarrow}\hat{d}^\dagger_{2,y,\uparrow}\hat{d}^\dagger_{2,y,\downarrow}\hat{d}^\dagger_{2,x,\uparrow}\hat{d}^\dagger_{2,x,\downarrow}\hat{c}^\dagger_{2,Yz,\uparrow}\hat{c}^\dagger_{2,Xz,\uparrow}\hat{c}^\dagger_{2,xy,\uparrow}\left|\Omega\right\rangle \\
&\quad = \hat{d}^\dagger_{1,x,\uparrow}\hat{c}_{2,Xz,\uparrow}\hat{c}^\dagger_{1,Yz,\uparrow}\hat{c}_{1,Xz,\uparrow}\hat{c}^\dagger_{1,Xz,\uparrow}\hat{c}^\dagger_{1,xy,\uparrow}\hat{d}^\dagger_{1,z,\uparrow}\hat{d}^\dagger_{1,z,\downarrow} \\
&\quad \hat{d}^\dagger_{1,y,\uparrow}\hat{d}_{1,y,\uparrow}\hat{d}^\dagger_{1,y,\uparrow}\hat{d}^\dagger_{1,y,\downarrow}\hat{d}_{1,x,\uparrow}\hat{d}^\dagger_{1,x,\uparrow}\hat{d}^\dagger_{1,x,\downarrow}\hat{d}^\dagger_{2,z,\downarrow}\hat{d}^\dagger_{2,y,\uparrow}\hat{d}^\dagger_{2,y,\downarrow}\hat{d}^\dagger_{2,x,\uparrow}\hat{d}^\dagger_{2,x,\downarrow}\hat{c}^\dagger_{2,Yz,\uparrow}\hat{c}^\dagger_{2,Xz,\uparrow}\hat{c}^\dagger_{2,xy,\uparrow}\left|\Omega\right\rangle \\
&\quad = -\hat{d}^\dagger_{1,x,\uparrow}\hat{c}^\dagger_{1,Yz,\uparrow}\hat{c}_{1,Xz,\uparrow}\hat{c}^\dagger_{1,Xz,\uparrow}\hat{c}^\dagger_{1,xy,\uparrow}\hat{d}^\dagger_{1,z,\uparrow}\hat{d}^\dagger_{1,z,\downarrow}\hat{d}^\dagger_{1,y,\uparrow} \\
&\quad \hat{d}_{1,y,\uparrow}\hat{d}^\dagger_{1,y,\uparrow}\hat{d}^\dagger_{1,y,\downarrow}\hat{d}_{1,x,\uparrow}\hat{d}^\dagger_{1,x,\uparrow}\hat{d}^\dagger_{1,x,\downarrow}\hat{d}^\dagger_{2,z,\downarrow}\hat{d}^\dagger_{2,y,\uparrow}\hat{d}^\dagger_{2,y,\downarrow}\hat{d}^\dagger_{2,x,\uparrow}\hat{d}^\dagger_{2,x,\downarrow}\hat{c}^\dagger_{2,Yz,\uparrow}\hat{c}_{2,Xz,\uparrow}\hat{c}^\dagger_{2,Xz,\uparrow}\hat{c}^\dagger_{2,xy,\uparrow}\left|\Omega\right\rangle \\
&\quad = -\hat{c}^\dagger_{1,Yz,\uparrow}\hat{c}_{1,Xz,\uparrow}\hat{c}^\dagger_{1,Xz,\uparrow}\hat{c}^\dagger_{1,xy,\uparrow}\hat{d}^\dagger_{1,z,\uparrow}\hat{d}^\dagger_{1,z,\downarrow}\hat{d}^\dagger_{1,y,\uparrow}\hat{d}_{1,y,\uparrow} \\
&\quad \hat{d}^\dagger_{1,y,\uparrow}\hat{d}^\dagger_{1,y,\downarrow}\hat{d}^\dagger_{1,x,\uparrow}\hat{d}_{1,x,\uparrow}\hat{d}^\dagger_{1,x,\uparrow}\hat{d}^\dagger_{1,x,\downarrow}\hat{d}^\dagger_{2,z,\downarrow}\hat{d}^\dagger_{2,y,\uparrow}\hat{d}^\dagger_{2,y,\downarrow}\hat{d}^\dagger_{2,x,\uparrow}\hat{d}^\dagger_{2,x,\downarrow}\hat{c}^\dagger_{2,Yz,\uparrow}\hat{c}_{2,Xz,\uparrow}\hat{c}^\dagger_{2,Xz,\uparrow}\hat{c}^\dagger_{2,xy,\uparrow}\left|\Omega\right\rangle \\
&\quad = -\hat{c}^\dagger_{1,Yz,\uparrow}\hat{c}^\dagger_{1,xy,\uparrow}\hat{d}^\dagger_{1,z,\uparrow}\hat{d}^\dagger_{1,z,\downarrow}\hat{d}^\dagger_{1,y,\uparrow}\hat{d}_{1,y,\uparrow} \\
&\quad \hat{d}^\dagger_{1,y,\uparrow}\hat{d}^\dagger_{1,y,\downarrow}\hat{d}^\dagger_{1,x,\uparrow}\hat{d}_{1,x,\uparrow}\hat{d}^\dagger_{1,x,\uparrow}\hat{d}^\dagger_{1,x,\downarrow}\hat{d}^\dagger_{2,z,\downarrow}\hat{d}^\dagger_{2,y,\uparrow}\hat{d}^\dagger_{2,y,\downarrow}\hat{d}^\dagger_{2,x,\uparrow}\hat{d}^\dagger_{2,x,\downarrow}\hat{c}^\dagger_{2,Yz,\uparrow}\hat{c}^\dagger_{2,xy,\uparrow}\left|\Omega\right\rangle \\
&\quad = -\left|S_1 = 1, m_1 = 1; S_2 = 1, m_2 = 1; d_{Yz}, d_{Yz}\right\rangle .
\end{aligned}
\tag{4.35}
$$

Thus, what this demonstrates is that it is not only important for the Rayleigh-Schrödinger perturbative model to calculate the sign and magnitude of the tight-binding hopping integrals but to also determine the sign introduced by the conventional ordering of the creation and annihilation operators. This means that in some cases two virtual hopping processes could have the same magnitude of contribution to the effective Hamiltonian operator but because they have opposite signs their effect cancels each other out to produce no net effect on the effective Hamiltonian for the virtual electron exchange.

4.3.2 Calculations for tight-binding hopping integrals

In the previous section, the Rayleigh-Schrödinger perturbative approach for developing an effective Hamiltonian for the magnetic interactions within the FeTe system was discussed. It was shown that up to fourth order electron hopping the effective Hamiltonian contained virtual hopping operators, $\hat{V}$, that describe the annihilation of an electron with either spin-up or spin-down on one species to be followed by the creation of an electron with the same spin projection on a neighbouring atom. The only thing that is so far missing from the calculation is any means for determining the hopping integrals using a tight-binding model approach. it is an assumption of this work that the the electrons in the solid are tightly bound to the constituent Fe and Te atoms meaning that they should have limited interaction with the electrons on neighbouring atoms. The consequence of being tightly bound is that the wavefunction of the electron will be close to that of an isolated single atom in a vacuum and the interaction with other neighbouring atoms can be treated as a small perturbation to this system. The interaction between neighbouring atomic sites can be described by a well-known tight-binding hopping integral, $t_{i\alpha,j\beta}$, that gives an energy cost associated with a virtual electron hop from one atomic orbital to another on a different atomic site:

$$t_{i\alpha,j\beta} = \int d^3\mathbf{r}[\psi_{i\alpha}(\mathbf{r}-\mathbf{R}_i)]^* V_{\text{eff}}^{ij}(\mathbf{r})[\psi_{j\beta}(\mathbf{r}-\mathbf{R}_i)], \tag{4.36}$$

where the $V_{\text{eff}}^{ij}(\mathbf{r})$ is an effective Coulomb potential defined as:

$$V_{\text{eff}}^{ij}(\mathbf{r}) = -\frac{1}{2}\Big[\frac{\mathrm{Z}_i e^2}{4\pi\epsilon_0|\mathbf{r}-\mathbf{R}_i|} + \frac{\mathrm{Z}_j e^2}{4\pi\epsilon_0|\mathbf{r}-\mathbf{R}_j|}\Big]. \tag{4.37}$$

In these expressions, i and j represent a pair of atomic sites whilst α and β represent a pair of atomic orbitals. An important relation for these integrals is that the reversal of the virtual hopping direction gives the complex conjugate of the hopping integrals:

$$[t_{j\beta,i\alpha}]^* = t_{i\alpha,j\beta}, \tag{4.38}$$

which provides a simple means for verifying that the calculated values make physical sense. Since the virtual hopping processes in this work go up to fourth order, their contributions to the effective Hamiltonian operator will be the product of the tight-binding hopping integrals for the sites and orbitals involved in the sequence of virtual electron exchange. For instance, reintroducing the tight-binding hopping integrals into the fourth order electron hopping operator in eq. 4.34 gives:

$$\hat{V}^4 = t_{1,x,2,Xz} t_{1,y,1,Xz} t_{2,Yz,1,y} t_{1,Yz,1,x} \hat{d}^\dagger_{1,x,\uparrow} \hat{c}_{2,Xz,\uparrow} \hat{d}^\dagger_{1,y,\uparrow} \hat{c}_{1,Xz,\uparrow} \hat{c}^\dagger_{2,Yz,\uparrow} \hat{d}_{1,y,\uparrow} \hat{c}^\dagger_{1,Yz,\uparrow} \hat{d}_{1,x,\uparrow}. \tag{4.39}$$

The contributions to the effective Rayleigh-Schrödinger perturbative Hamiltonian is therefore composed of a magnitude given by the absolute value of the product of the tight-binding hopping integrals and the overall sign of each virtual exchange process is composed of both the sign from the hopping integrals and the sign due to the operation of the creation and annihilation operators on states within the model space.

4.3.3 Numerical calculations using Monte Carlo method

Having described the mathematical tools for calculating the plurality of input parameters for the Rayleigh-Schrödinger perturbative model, it will now be shown how the crystal field splitting, Hund's rule coupling and tight-binding hopping integrals were numerically determined using stochastic integration methods. Owing to the multi-dimensionality of some of the integrals the Monte Carlo method still ensures relatively quick convergence and low uncertainties of the outputs. The former guarantees that the computational work to arrive at accurate and reliable numerical estimates of the integrals is not too burdensome, whilst the latter promises that the uncertainties of particular inputs do not propagate to produce much larger fractional uncertainties of the elements of the 32×32 effective Hamiltonian matrix. The propagation of uncertainties to produce large fractional uncertainties in the elements of the matrix comes from the structure of the effective Hamiltonian as shown in eq. 4.26. For instance, each element of the effective Hamiltonian is associated with a large number of virtual electron exchange processes that can often cancel each other out due to the sign introduced by the action of the electron hopping operator, $\hat{V}$, on states within either the model or orthogonal space. The propagated uncertainty of the sum of these virtual electron exchange processes will be quadrature additive meaning that no matter the sign introduced by the action of $\hat{V}$ the uncertainty will always grow. This is not the case for the estimated value of a particular element in the effective Hamiltonian matrix (e.g., $\langle S_1 = 1, m_1 = 1; S_2, m_2; d_{Yz}, d_{Yz}| \hat{H}_{\text{eff}} |S_1 = 1, m_1 = 1; S_2, m_2; d_{Xz}, d_{Xz}\rangle$). The Monte Carlo method for calculating an integral is not so much a specific method but a broad class of algorithms for doing numerical calculations using random sampling. Hence, in the following it will be explained how the Monte Carlo method was set up for numerically calculating the integrals that are a requisite for constructing the Rayleigh-Schrödinger perturbative model of the FeTe system.

The simple idea behind a Monte Carlo method is to determine an integral using a random sampling of N points selected from the space defined by the limits of integration. The Monte Carlo estimator, $\mathbb{E}$, of the integral is then calculated as the average value of the integrand over the space defined by the integration limits:

$$\mathbb{E}(\int_{\mathcal{V}} f(\mathbf{x})dV) = \mathcal{V}\frac{\sum_{i=1}^{N} f(\mathbf{x}_i)}{N}, \tag{4.40}$$

where $\int_{\mathcal{V}} f(\mathbf{x})dV$ is the integral to be evaluated, $\mathcal{V}$ is the integral volume defined by the limits of integration and $f(\mathbf{x}_i)$ is the integrand evaluated at the randomly selected point $\mathbf{x}_i$. The Monte Carlo estimator converges to the actual value at a rate that is proportional to $1/\sqrt{N}$. Convergence of the algorithm can be pre-determined by set criteria including a simple difference between consecutive Monte Carlo estimators or more complicated convergence metrics.

The Monte Carlo algorithm developed for estimating the crystal field splitting, Hund's rule coupling and tight-binding hopping integrals consisted of the following steps:

- an hypercube with dimensionality equal to that of the integral and with sides of length ΔL is chosen as the starting $\mathcal{V}$;
- a random sample of N points within $\mathcal{V}$ (the hypercube is conventionally centred on an atomic site) are used to give an initial Monte Carlo estimate of the integral;
- the number of points is increased by a factor of δ (i.e., N$\rightarrow \delta$N) and a new estimate of the integral is evaluated;
- if the fractional change in the Monte Carlo estimate of the integral using the two different numbers of randomly sampled points is greater than a chosen threshold of convergence, then repeat the previous step. Otherwise proceed to the next step;
- increase the hypercube side lengths by a factor of γ (i.e., $\Delta L \rightarrow \gamma\Delta L$) and a new estimate of the integral is evaluated;
- using the same metric as before check for the convergence of the Monte Carlo estimator after having increased ΔL but with N kept constant;
- if convergence of the Monte Carlo estimate is achieved, then this is the final estimate of the integral. Otherwise, repeat the algorithm from the third step and carry on until there is convergence, which might require multiple increases in both the hypercube side length and the number of sampling points.

The computational load of this Monte Carlo algorithm depends on the choice of the threshold of convergence, ΔL, N, δ and γ. It also depends on the dimensionality of the integral with a higher number of dimensions leading to a longer time to convergence. For instance, since the Hund's rule coupling and tight-binding hopping integrals have a dimensionality of D=6, they will take longer to converge compared with evaluating the crystal field splitting integrals, which only have a dimensionality of D=3. When carrying out a double integral in practice it is often necessary to "soften" the $1/|\mathbf{r}-\mathbf{r}'|$ divergence of the effective Coulomb potential in order to avoid a zero appearing in the denominator of the integrand in the numerical calculations. This can be achieved by including a "softening" length, δ:

$$\frac{e^2}{4\pi\epsilon|\mathbf{r}-\mathbf{r}'|} \rightarrow \frac{e^2}{4\pi\epsilon\sqrt{(\mathbf{r}-\mathbf{r}')^2+\delta^2}}, \tag{4.41}$$

Species	x (a)	y (a)	z (c)
Fe	0.000	0.000	0.000
Te1	0.000	-0.512	0.285
Te2	-0.494	-0.006	-0.285
Te3	0.000	0.500	0.285
Te4	0.494	-0.006	-0.285

Table 4.3: The parallelogram of Te atoms around a re-centred Fe atom within a distorted FeTe lattice.

and incrementally reducing δ until it is small enough that it does not change the result of the numerical integration.

As was mentioned in Sec. 4.1, another complication of the numerical calculations was that the values of the aforementioned integrals should produce states with the experimentally observed ground states. For example, the single-ion Fe ground state observed in experiments is in the S=1 spin state with a fully occupied e_g orbital and an electron in both the d_{xy} and d_{Xz} or d_{Yz} orbital (the latter depends on the definition of the x and y axes and so is an arbitrary choice not relating to the physics of the problem). To achieve the S=1 spin state on the Fe^{2+} ions the geometrical distortion of the lattice, the effective nuclear core charge and the Fermi-Thomas Coulomb screening length due to conduction electrons need to be optimised. The distortion parameter, d is defined as a percentage of the bond length between neighbouring Fe and Te ions in the undistorted lattice as shown in Tab. 4.2. A pair of Te ions on opposite sides of the parallelogram are moved a distance of $2d$ away from each other, whilst the other pair are moved $2d$ towards each other along a line colinear with their original locations. Along the axis colinear with the pair of Te ions that have been moved further apart the Fe ion is displaced by d towards one of either of those Te ions. Using a distortion parameter of $d = 1\%$, the positions of the Te ions forming a parallelogram encasing a re-centred Fe ion are shown in Tab. 4.3. With this distortion parameter fixed the remaining variables (i.e., the effective nuclear core charges and the Fermi-Thomas Coulomb screening length) need to be optimised to give the correct single-ion ground state. The phase space made up of these two parameters might have a plurality of regions that produce the S=1 spin state and so it is often difficult to know that a unique solution to this optimisation problem has been achieved. Using the aforementioned initial guesses for these two parameters as described in Sec. 4.2, it did not produce the ground state that was consistent with the experimental data. A process of trial-and-error lead to a set of parameters that gave the expected single-ion ground state, but due to its ad hoc nature it is not certain that the set of results is within a unique region of phase space or whether there exists other regions that satisfy these conditions. These set of parameters are tabulated in Tab. 4.4 for a lattice distortion parameter of $d = 1\%$.

The crystal field splitting and Hund's rule coupling constants obtained for both species in the FeTe system using the parameters presented in Tab. 4.4 are shown in Tab. 4.5-4.6. Due to the symmetry of the atomic orbitals on the Fe and Te ions some of the Hund's rule coupling constants are the same for different

Parameter	Numerical Value
Z_{eff} (Fe)	1.25
Z_{eff} (Te)	4.75
Fermi-Thomas Coulomb Screening Length (ζ)	2.0Å

TABLE 4.4: A set of parameters given a lattice distortion parameter of $d = 1\%$ that gave the single-ion ground state with S=1 consistent with the refined magnetic moment of the Fe atoms observed in experiments.

TABLE 4.5: Energy shifts due to the crystal field splitting for the Fe 3d orbitals (effective core nuclear charge of Z_{eff}=1.25 on the Fe ions and a Fermi-Thomas Coulomb screening length of 2.0Å.

d orbital (real basis)	Splitting (eV)	Error (eV)
d_{xy}	5.56	0.01
d_{Xz}	6.82	0.01
d_{Yz}	6.79	0.01
$d_{x^2-y^2}$	4.63	0.01
d_{z^2}	5.52	0.01

pairs of orbitals. In order to aid the reader in understanding the tabulated data in Tab. 4.5-4.6, the matrix form of the Hund's rule coupling parameters, $2U_{\mu\lambda\lambda\mu}$ and J, are also shown below using a basis constructed from their real orbitals. The ordering of the basis is $|\psi_\alpha(\mathbf{r})\rangle$ where $\alpha \in \{d_{xy}, d_{Xz}, d_{Yz}, d_{x^2-y^2}, d_{z^2}\}$ for the Fe ions and $\alpha \in \{p_x, p_y, p_z\}$ for the Te ions. Hence, the matrices for $2U_{\mu\lambda\lambda\mu}$ and J are 5×5 matrices for the Fe ions and 3×3 matrices for the Te ions (see below for their structure including any equalities between the numerical value of different elements in the matrices due to symmetries of the atomic orbitals).

$$(2U_{\mu\lambda\lambda\mu}/J)_{\text{Fe}} = \begin{bmatrix} a & b & b & c & d \\ b & a & e & f & g \\ b & e & a & f & g \\ c & f & f & a & d \\ d & g & g & d & a \end{bmatrix} \tag{4.42}$$

$$(2U_{\mu\lambda\lambda\mu}/J)_{\text{Te}} = \begin{bmatrix} a & b & b \\ b & a & b \\ b & b & a \end{bmatrix} \tag{4.43}$$

As was previously mentioned, the chosen values for the effective nuclear core charge and Fermi-Thomas Coulomb screening length are not necessarily the unique solution to this optimisation problem, but the fact that the calculated energy associated with the Jahn-Teller effect (the difference in energy between the d_{Xz} and d_{Yz} orbitals) is 30±1meV suggests that these inputs would suffice for developing a Rayleigh-Schrödinger perturbative model for the FeTe system that accurately reflects its magnetic and orbital physics.

Table 4.6: Hund's rule coupling constants for the Fe ions using the parameters presented in Tab. 4.4.

Species	Parameter	Energy (eV)	Error (eV)
Fe	$2U_{\mu\lambda\lambda\mu}$ (a)	1.22	0.02
	$2U_{\mu\lambda\lambda\mu}$ (b)	0.20	0.01
	$2U_{\mu\lambda\lambda\mu}$ (c)	0.13	0.01
	$2U_{\mu\lambda\lambda\mu}$ (d)	0.22	0.01
	$2U_{\mu\lambda\lambda\mu}$ (e)	0.20	0.01
	$2U_{\mu\lambda\lambda\mu}$ (f)	0.20	0.01
	$2U_{\mu\lambda\lambda\mu}$ (g)	0.15	0.01
	J (a)	1.24	0.01
	J (b)	0.20	0.01
	J (d)	0.22	0.01
	J (e)	0.20	0.01
	J (f)	0.20	0.01
	J (g)	0.16	0.01
Te	$2U_{\mu\lambda\lambda\mu}$ (a)	4.35	0.03
	$2U_{\mu\lambda\lambda\mu}$ (b)	3.19	0.05
	J (a)	4.36	0.06
	J (b)	0.54	0.01

Furthermore, the tight-binding hopping integrals were calculated using the same set of initial parameters and calculated for the virtual electron hopping between Fe^{2+} ions and Te^{2-} both below and above the plane formed by the lattice of the Fe2+ ions in the crystal. As can be seen in Fig. 4.1, the structural distortions of the tetrahedra of Te^{2-} ions around each Fe^{2+} ion breaks the translational symmetry of the crystal and removes the equivalency between some of the Fe-Te bonds. This means that the tight-binding hopping integrals used for different pairs of Fe^{2+} and Te^{2-} are different and need to be carefully labelled when constructing the Rayleigh-Schrödinger perturbative model for the FeTe system as a whole. It is possible to identify eight different Fe-Te bonds in Fig. 4.1 meaning that in order to obtain a full picture of the magnetic and orbital interactions in FeTe it is necessary to calculate 72 tight-binding hopping integrals: between each pair of Fe and Te atoms there are 9 tight-binding integrals for the virtual electron hopping between the d_{xy}, d_{Xz} and d_{Yz} orbitals in the e_g subgroup of the Fe atoms and the p_x, p_y and p_z orbitals in the Te atoms. The numerical calculations for these tight-binding integrals have been carried out and for simplicity the equalities between pairs of bonds are illustrated in Tab. 4.7. In the numerical calculations used the same initial parameters as had been used for determining the crystal field splitting and Hund's rule coupling constants that are shown in Tab. 4.5 and Tab. 4.6. However, whatever values are used for the calculations the equalities presented below will not be broken.

TABLE 4.7: The symmetries of the tight-binding hopping integrals determined from the parameters presented in Tab. 4.4. The lobes of the d_{Xz} and d_{Yz} point towards the Te atoms forming the tetrahedron in the undistorted crystal structure. The labels for the eight Fe atoms are illustrated for clarity in Fig. 4.5.

Fe-Te Bond	orbital pairs	p_x	p_y	p_z
1	d_{xy}	c	d	g
	d_{Xz}	i	ll	oo
	d_{Yz}	s	v	y
2	d_{xy}	b	e	h
	d_{Xz}	k	n	q
	d_{Yz}	t	w	z
3	d_{xy}	a	f	i
	d_{Xz}	l	o	r
	d_{Yz}	u	x	zz
4	d_{xy}	a	-f	-i
	d_{Xz}	-l	o	r
	d_{Yz}	-u	x	zz
5	d_{xy}	-c	-d	g
	d_{Xz}	i	-ll	-oo
	d_{Yz}	s	-v	-y
6	d_{xy}	-b	-e	h
	d_{Xz}	k	-n	-q
	d_{Yz}	t	-w	-z
7	d_{xy}	-a	f	i
	d_{Xz}	l	-o	-r
	d_{Yz}	u	-x	-zz
8	d_{xy}	-a	-f	-i
	d_{Xz}	-l	-o	-r
	d_{Yz}	-u	-x	-zz

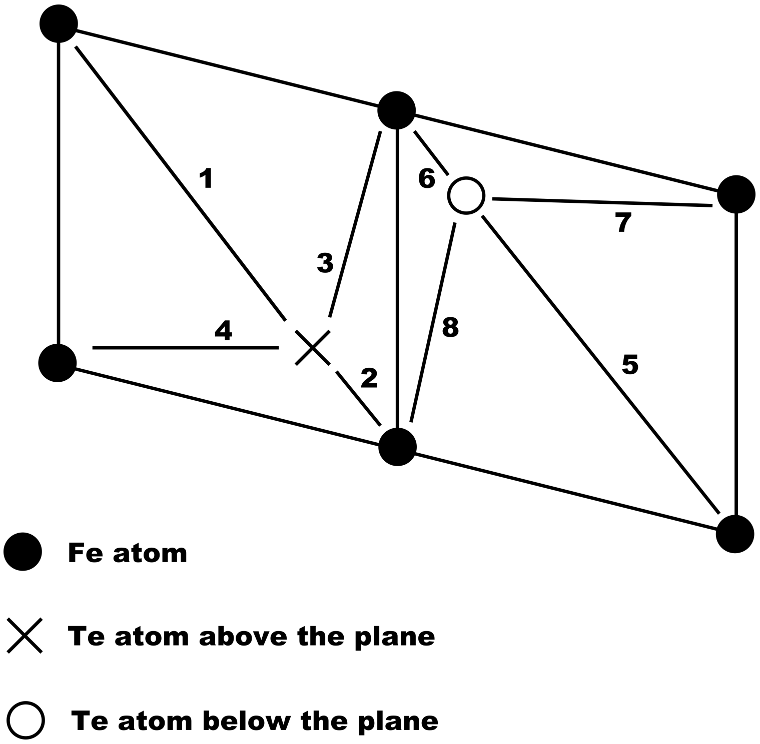


FIGURE 4.4: The labelling scheme for the Fe-Te bonds in the FeTe system that has undergone a structural distortion owing to the Jahn-Teller effect. Note that the overlap of the orbitals in the Fe and Te atoms will depend on whether the Te atom is above or below the plane of Fe atoms in the lattice.

The virtual electron exchange between neighbouring Fe atoms is mediated by different Te atoms. Hence, it is important when constructing the effective Hamiltonian for different pairs of Fe atoms to choose the appropriate tight-binding integrals as inputs to the Rayleigh-Schrödinger perturbative model. As an example, in constructing an effective Hamiltonian for the coupling labelled J_{1a} in Fig. 4.1 the tight-binding hopping integrals for bond 2, 3, 6 and 8 ought to be used.

The equality between pairs of the bonds in Tab. 4.7 is due to the symmetry of the tight-binding hopping integral in eq. 4.36. This is due to the fact their respective tight-binding hopping integrals can be related by an inversion, i.e., $(x, y, z) \to (-x, -y, -z)$, of the integration axes, which is the same as a rotation about the z-axis by 180° followed by a reflection in the x-y plane.

4.4 Structure of the FeTe perturbative model

The preceding sections have described how various inputs for the Rayleigh-Schrödinger perturbative model for the FeTe system have been calculated from first principles using our understanding of the lattice and orbital structure of the FeTe single crystals. To summarise, the FeTe single crystals are assumed to undergo a structural distortion that lifts the degeneracy of the e_g and t_{2g} orbitals followed by a Jahn-Teller distortion that further splits the d_{Xz} and d_{Yz} orbitals in the t_{2g} orbitals. The latter occurs since the Jahn-Teller distortion reduces the symmetry of the crystal, thus breaking the geometrical frustration of the S=1 spin state, leading to a lower energy for the ground state, which can be partly explained by the reduction in entropy within the unfrustrated system due to there being fewer degenerate configurations. The result of these structural distortions is that the tetrahedra of Te ions surrounding each stoichiometric Fe ion is changed such that a pair of Te ions on opposite sides of a tetrahedron move further away from each other along the axis defined by the line colinear with their original positions, whilst the other pair of Te ions move close together along with own similarly defined axis. At the same time the Fe ion moves closer to one of the Te ions in the pair that moved further away from each other. The degree of this lattice distortion is described by a lattice distortion parameter as described in Sec. 4.3.3. Using a set of input parameters, including this lattice distortion parameter, the effective nuclear core charges of the two species and the Fermi-Thomas Coulomb screening length, the absolute energies, crystal field splitting, Hund's rule coupling and tight-binding hopping integrals can be determined. The first three can be used to determine the zeroth order energies of the states in the model and orthogonal spaces, whilst the latter enables the higher order terms in the Rayleigh-Schrödinger perturbative model's effective Hamiltonian to be calculated (see eq. 4.26 in Sec. 4.3).

Since the model space has degenerate states (time-reversal symmetry of the Hamiltonian leads to pairs of states with the same energy), the Rayleigh-Schrödinger perturbative expansion was developed using the degenerate case—this prevents any term in the higher order terms having a vanishing denominator leading to an infty in the expansion. An effective Hamiltonian was constructed separately for different neighbouring pairs of Fe ions with virtual electron hopping permitted for the two closest Te ions in the crystal. The single-ion basis states for the Fe ions were the triplet states (S=1) in which either the d_{Xz} or d_{Yz} orbital in the t_{2g} subgroup is occupied. The effective Hamiltonian describing the coupling between two neighbouring Fe ions (with the ability to exchange electrons with the fully occupied 5p orbitals in two Te ions) is therefore constructed from a set of 36 basis states. In matrix form, the elements of this effective Hamiltonian are given by:

$$\mathbf{H}_{ij}^{ab} = \langle\psi_i|\, \hat{H}_{\text{eff}}^{ab}\, |\psi_j\rangle\,, \tag{4.44}$$

where $\mathbf{H}_{ij}^{ab}$ is the ijth element of the effective Hamiltonian matrix describing the coupling between the two Fe ions labelled a and b, $|\psi_i\rangle$ are the basis states and $\hat{H}$ is the effective Hamiltonian defined in eq. 4.26. $|\psi_i\rangle$ are written as a product of creation operators for the t_{2g} subgroup in the two Fe ions and the 5p orbitals in the two Te ions acting on a vacuum state, $|\Omega\rangle$. This is described in Sec. 4.3.1. The reason why the elements of the effective Hamiltonian matrix are different for different pairs of Fe ions is that the crystal environment around each Fe ion is no longer identical owing to the lattice distortion comprising the Jahn-Teller effect.

It is desirable to find a physically meaningful basis in which to decompose the 36×36 effective Hamiltonian matrix, $\mathbf{H}$ (labels have been removed for simplicity), describing the magnetic and orbital interactions between the neighbouring Fe ions. It is claimed that the entries in that basis can be written in the following form:

$$\mathbf{Q}^{cdef} = \mathbf{S}_1^c \otimes \mathbf{T}_1^d \otimes \mathbf{S}_2^e \otimes \mathbf{T}_2^f. \tag{4.45}$$

Here $c, e \in \{0,1,2,3,4,5,6,7,8\}$ and $d, f \in \{0,1,2,3\}$. Hence the string $cdef$ has $4 \times 9 \times 4 \times 9 = (36)^2$ possible values, which matches the number of independent real numbers encoded in a 36×36 Hermitian matrix such as $\mathbf{H}$ (the property of the trace for the spin, $\mathbf{S}$, and pseudo-spin, $\mathbf{T}$, will be shown to provide for a complete orthogonal basis set for the effective Hamiltonian matrix) 4 The matrices $\mathbf{S}_1^a$ and $\mathbf{S}_2^a$ are spin matrices acting on the first and second spins respectively. Let us choose the following nine:

$$\mathbf{S}^0 = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix} \qquad \mathbf{S}^1 = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & -1 \end{pmatrix} \qquad \mathbf{S}^2 = \begin{pmatrix} 0 & \sqrt{2} & 0 \\ 0 & 0 & \sqrt{2} \\ 0 & 0 & 0 \end{pmatrix}$$

$$\mathbf{S}^3 = \begin{pmatrix} 0 & 0 & 0 \\ \sqrt{2} & 0 & 0 \\ 0 & \sqrt{2} & 0 \end{pmatrix} \qquad \mathbf{S}^4 = \begin{pmatrix} 1 & 0 & 0 \\ 0 & -2 & 0 \\ 0 & 0 & 1 \end{pmatrix} \qquad \mathbf{S}^5 = \begin{pmatrix} 0 & 0 & 2 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix}$$

$$\mathbf{S}^6 = \begin{pmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 2 & 0 & 0 \end{pmatrix} \qquad \mathbf{S}^7 = \begin{pmatrix} 0 & \sqrt{2} & 0 \\ 0 & 0 & -\sqrt{2} \\ 0 & 0 & 0 \end{pmatrix} \qquad \mathbf{S}^8 = \begin{pmatrix} 0 & 0 & 0 \\ \sqrt{2} & 0 & 0 \\ 0 & -\sqrt{2} & 0 \end{pmatrix} \tag{4.46}$$

Note that:

- $\mathbf{S}^0$ is the identity matrix;
- $\mathbf{S}^1$, $\mathbf{S}^2$, and $\mathbf{S}^3$ are respectively $\mathbf{S}^z$, $\mathbf{S}^+$, and $\mathbf{S}^-$;
- $\mathbf{S}^5 = (\mathbf{S}^2)^2$;
- $\mathbf{S}^6 = (\mathbf{S}^3)^2$;
- $\mathbf{S}^7 = \{\mathbf{S}^1, \mathbf{S}^2\}$; and
- $\mathbf{S}^8 = \{\mathbf{S}^1, \mathbf{S}^3\}$.

These are a convenient choice because they obey the following orthogonality relation:

$$\mathrm{Tr}\left(\left[\mathbf{S}^i\right]^\dagger \mathbf{S}^j\right) = C_{ij}\delta_{ij}, \tag{4.47}$$

where C_{ij} are constants. Some of these matrices alter the spin projection of the S=1 spin states: $\mathbf{S}^2$ and $\mathbf{S}^7$ lower the spin once; $\mathbf{S}^3$ and $\mathbf{S}^8$ raise the spin once; $\mathbf{S}^5$ raises the spin twice; $\mathbf{S}^6$ lowers the spin twice.

The matrices $\mathbf{T}_1^d$ and $\mathbf{T}_2^f$ are orbital pseudo-spin matrices acting on the first and second orbital pseudo-spins respectively. Let us choose the following four:

$$\mathbf{T}^0 = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}, \quad \mathbf{T}^1 = \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix}, \quad \mathbf{T}^2 = \begin{pmatrix} 0 & 1 \\ 0 & 0 \end{pmatrix}, \quad \mathbf{T}^3 = \begin{pmatrix} 0 & 0 \\ 1 & 0 \end{pmatrix}. \tag{4.48}$$

$\mathbf{T}^0$ and $\mathbf{T}^1$ count the pseudo-spins; $\mathbf{T}^2$ and $\mathbf{T}^3$ are the raising and lowering operators respectively, i.e., the reduce or increase the pseudo-spin projection. These pseudo-spin matrices obey a similar relation for their traces as described for the spin matrices in eq. 4.47.

In this orthogonal basis set the effective Hamiltonian matrix can be written as:

$$\mathbf{H} = \sum_{cdef} h_{cdef} \mathbf{Q}^{cdef}, \tag{4.49}$$

where h_{cdef} are the 36^2 weights. Having calculated all the elements of the effective Hamiltonian matrix, the objective is to calculate the weights of the Kronecker decomposition, which can be achieved by multiplying both sides by $\mathbf{Q}^{abcd\dagger}$ from the left-hand side:

$$\mathbf{Q}^{ghij\dagger}\mathbf{H} = \sum_{cdef} h_{cdef} \mathbf{Q}^{ghij\dagger}\mathbf{Q}^{cdef}, \tag{4.50}$$

and taking the trace of the Kronecker product and using the identity $Tr(\mathbf{A} \otimes \mathbf{B}) = Tr(\mathbf{A})Tr(\mathbf{B})$:

$$\begin{aligned} Tr(\mathbf{Q}^{ghij\dagger}\mathbf{H}) &= \sum_{cdef} h_{cdef} Tr(\mathbf{Q}^{ghij\dagger}\mathbf{Q}^{cdef}) \\ &= \sum_{cdef} h_{cdef} Tr([\mathbf{S}_1^g]^\dagger \mathbf{S}_1^c \otimes [\mathbf{T}_1^h]^\dagger \mathbf{T}_1^d \otimes [\mathbf{S}_2^i]^\dagger \mathbf{S}_2^e \otimes [\mathbf{T}_2^j]^\dagger \mathbf{T}_2^f) \\ &= \sum_{cdef} h_{cdef} Tr([\mathbf{S}_1^g]^\dagger \mathbf{S}_1^c) Tr([\mathbf{T}_1^h]^\dagger \mathbf{T}_1^d) Tr([\mathbf{S}_2^i]^\dagger \mathbf{S}_2^e) Tr([\mathbf{T}_2^j]^\dagger \mathbf{T}_2^f) \\ &= \sum_{cdef} h_{cdef} C_{gc} C_{hd} C_{ie} C_{jf} \delta_{gc} \delta_{hd} \delta_{ie} \delta_{hd} \\ &= h_{ghij} C_{gg} C_{hh} C_{ii} C_{jj} \end{aligned} \tag{4.51}$$

and making the weights the subject of the equation by dividing through by the constants gives:

$$h_{ghij} = \frac{Tr(\mathbf{Q}^{ghij\dagger}\mathbf{H})}{C_{gg} C_{hh} C_{ii} C_{jj}}. \tag{4.52}$$

These are the unique weights of the effective Hamiltonian in the basis of the product of spins and pseudo-spins on the neighbouring Fe ions. This proves that $\mathbf{Q}^{cdef}$ forms a complete orthogonal basis set for the effective Hamiltonian matrix. Some of the weights must be zero due to the conservation of spin for tight-binding electron hopping processes. The reason why is because the effective Coulomb potential, $V_{\text{eff}}^{ij}(\mathbf{r})$, in eq. 4.37 is spin-independent, which only leads to non-zero tight-binding hopping integrals if the spins of the initial and final state are either both up or down. This can be seen if the spin part of the electronic wavefunctions, χ_σ, is introduced into the expression for the tight-binding hopping integrals in eq. 4.36:

$$\begin{aligned} t_{i\alpha\sigma,j\beta\gamma} &= \sum_\sigma \sum_\gamma \int\int \mathbf{r} d^3\mathbf{r}' [\psi_{i\alpha}(\mathbf{r}-\mathbf{R}_i)]^* V_{\text{eff}}^{ij}(\mathbf{r}) [\psi_{i\alpha}(\mathbf{r}-\mathbf{R}_i)]_{\sigma\gamma} \\ &= \int\int \mathbf{r} d^3\mathbf{r}' [\psi_{i\alpha}(\mathbf{r}-\mathbf{R}_i)]^* V_{\text{eff}}^{ij}(\mathbf{r}) [\psi_{i\alpha}(\mathbf{r}-\mathbf{R}_i)] \delta_{\sigma\gamma}, \end{aligned} \tag{4.53}$$

implying that the tight-binding hopping conserves the spin of the electrons. The effective Hamiltonian can reflect this conservation of spin by only having non-zero weights for the subset of the orthogonal basis set that conserves the overall spin of the system: if the spin on one of the Fe ions is decreased it must be increased by the same amount on the neighbouring Fe ion, or if the spin is not changed on the former it must also remain the same on the latter. The subset of spin matrices that increase the spin by one, i.e., $\{\mathbf{S}^2, \mathbf{S}^7\}$, on one Fe ion may appear in the effective Hamiltonian with the spin matrices that decrease the spin of the neighbouring Fe ion by one, i.e., $\{\mathbf{S}^3, \mathbf{S}^8\}$. $\mathbf{S}^5$ may appear with $\mathbf{S}^6$ since these raise the spin twice on one Fe ion and lower it twice on the other. Moreover, the subset of spin matrices that lead to neither the raising nor lowering of the spin on either Fe ion, i.e., $\{\mathbf{S}^0, \mathbf{S}^1, \mathbf{S}^4\}$, may be used with another matrix within the same subset. There is no restriction on which pseudo-spin matrices appear in the effective Hamiltonian due to the fact that the tight-binding hopping integrals are dependent on the atomic orbitals and that channels for virtual electron hopping are open for processes that lead to the orbital occupations on the Fe ions changing.

It is entirely possible that other weights in the effective Hamiltonian are zero, but predicting this from the structure of the Rayleigh-Schrödinger perturbative expansion of the effective Hamiltonian is much more difficult. This can only practically be done by evaluating the effective Hamiltonian using eq. 4.26, decomposing the 36×36 matrix using the orthogonal basis set defined in eq. 4.45, and then trying to understand whether a pattern exists by analysing the weights, h_{ghij}.

4.4.1 Weights for the effective Hamiltonian using the orthogonal basis set

In this section, the weights, h_{cdef}, for the effective Hamiltonian in the orthogonal basis set, $\mathbf{Q}^{cdef}$, are evaluated by following the methods set out in the preceding description of the Rayleigh-Schrödinger perturbative model. All methods, including evaluating the elements of the 36×36 effective Hamiltonian and the Kronecker decomposition of this matrix, have been programmed in python with the scripts presented in Appendix. B.7.

After evaluating the weights for the effective Hamiltonian using the aforementioned parameters, it was noted that more were zero than were required by the conservation of spin due to the spin-independence of the effective Coulomb potential, $V_{\text{eff}}^{ij}(\mathbf{r})$. This was observed by splitting the 36^2 weights into those associated with either no change to the spin or pseudo-spin on either Fe ion, a change of spin, a change of pseudo-spin or a change of both the spin and pseudospin. For instance, it was observed that raising the spin projection by two on one Fe ion and lowering it by two on the neighbouring Fe ion could never be combined with pseudospin matrices that changed the pseudo-spin projection, i.e., $\mathbf{S}^5$ and $\mathbf{S}^6$ could never be combined with one or both of $\mathbf{T}^2$ and $\mathbf{T}^3$. Moreover, it was also noted that in the subgroup of weights associated with only spin changes $\mathbf{S}^5$ and $\mathbf{S}^6$ could also not be combined with $\mathbf{T}^2$ and $\mathbf{T}^3$, which, in effect, meant that in this model there is no possibility for the spin projection to be simultaneously raised by two on one Fe ion and lowered by the same amount on the neighbouring Fe ion irrespective of what happens to the pseudo-spin projections. Even though a lot of weights were non-zero because they were several orders of magnitude it was decided to ignore them in order to construct a more simplistic model of the magnetic and orbital interactions in the FeTe system. Moreover, due to some of the weights having an equal magnitude, with perhaps a change of sign, some of the spin and pseudo-spin matrices were combined to form new matrices to simplify the model so as to more easily understand what is happening to the spin and pseudo-spin projections in some of the virtual electron hopping processes. These spin matrices were $\mathbf{S}^2$, $\mathbf{S}^3$, $\mathbf{S}^7$ and $\mathbf{S}^8$ that could be combined to form new matrices defined by the following matrices defined by:

$$\mathbf{S}^m = \mathbf{S}^3 - \mathbf{S}^8, \tag{4.54}$$

$$\mathbf{S}^p = \mathbf{S}^2 + \mathbf{S}^7. \tag{4.55}$$

It was also possible to combine a pair of the pseudospin matrices, $\mathbf{T}^2$ and $\mathbf{T}^3$, such that:

$$\mathbf{T}^x = \mathbf{T}^2 + \mathbf{T}^3, \tag{4.56}$$

which has eigenvectors pointing along the positive or negative direction of the x-axis of a Bloch sphere. Hence, the effective Hamiltonian can therefore be written in terms of the following spin operators:

- an identity, $\mathbf{I} \otimes ... \otimes \mathbf{I} \otimes ...$;
- a term representing the single-ion anisotropy, $\mathbf{I} \otimes ... \otimes \mathbf{S}^4 \otimes ... + \mathbf{S}^4 \otimes ... \otimes \mathbf{I} \otimes ...$;
- an Ising term that correlates spin-projections, $\mathbf{S}^1 \otimes ... \otimes \mathbf{S}^1 \otimes ...$;
- an Ising-like term that correlates mid-planeness, $\mathbf{S}^4 \otimes ... \otimes \mathbf{S}^4 \otimes ...$;
- and a spin exchange term, $\mathbf{S}^p \otimes ... \otimes \mathbf{S}^m \otimes ... + \mathbf{S}^m \otimes ... \otimes \mathbf{S}^p \otimes ...$;

and orbital pseudo-spin operators:

- an identity, $\ldots \otimes \mathbf{I} \otimes \ldots \otimes \mathbf{I}$;
- a term representing the single-site 'orbital transverse field', $\ldots \otimes \mathbf{I} \otimes \ldots \otimes \mathbf{T}^x + \ldots \otimes \mathbf{T}^x \otimes \ldots \otimes \mathbf{I}$;
- and a two-site 'orbital transverse field' term, $\ldots \otimes \mathbf{T}^x \otimes \ldots \otimes \mathbf{T}^x$,

where the “orbital transverse field" acts like an external field transverse to the poles of the Bloch sphere. Multiplying the five spin terms with the three orbital pseudo-spin terms gives a total of fifteen coefficients for the effective Hamiltonian matrix in the orthogonal basis set, $\mathbf{Q}^{cdef}$. However, it was found during the course of developing the model that there are different coefficients for when the $\mathbf{S}^4$ and $\mathbf{T}^x$ appear on the same Fe site or on different sites, i.e., $\mathbf{I} \otimes \mathbf{I} \otimes \mathbf{S}^4 \otimes \mathbf{T}^x + \mathbf{S}^4 \otimes \mathbf{T}^x \otimes \mathbf{I} \otimes \mathbf{I}$ and $\mathbf{S}^4 \otimes \mathbf{I} \otimes \mathbf{I} \otimes \mathbf{T}^x + \mathbf{I} \otimes \mathbf{T}^x \otimes \mathbf{S}^4 \otimes \mathbf{I}$. Hence, there are actually 16 coefficients in the effective Hamiltonian.

In order to construct an effective Hamiltonian for each layer of the FeTe system (it is assumed that the inter-layer coupling is much less than the intra-layer coupling), it is necessary to calculate the fifteen coefficients for each pair of the neighbouring Fe ions in the repeating 2D unit cells for FeTe. The unit cell for FeTe, which contains four Fe ions each, is shown in Fig. 4.5. The effective Hamiltonian for the layers of FeTe would therefore include couplings between neighbouring pairs of Fe ions within the same unit cell and also couplings between pairs in adjacent unit cells. The couplings included in the effective Hamiltonian for FeTe include nearest and next-nearest neighbours (i.e., diagonal bonds between Fe ions) in Fig. 4.5. Each Fe ion in the unit cells would have bonds with two nearest neighbours and two next-nearest neighbours meaning that for each unit cell there would be a total of 16 bonds. Each of these bonds has an effective Hamiltonian that can be described by the aforementioned 16 coefficients leading to the entire effective Hamiltonian for the FeTe system being fully described by a total of 240 coefficients. However, if the periodic structure of the Te ions above and below the plane of the Fe ions were put into Fig. 4.5, the crystal's translational symmetry would imply that these 16 bonds are double the number that actually repeats across unit cells. This suggests that the effective Hamiltonian for the FeTe system can instead be described by 120 coefficients.

If the Fe ions in each unit cell are labelled a, b, c and d (a being the Fe ion located at the farthest site to the left and d being located at the farthest site to the right), then the effective Hamiltonian for the intra-layer coupling terms can be written as:

$$
\begin{aligned}
\mathbf{H}_{\mathrm{FeTe}} = \sum_{\mathbf{R}} \big(& \mathbf{H}^{aa}_{\mathbf{R},\mathbf{R}+\mathbf{a}^*} + \mathbf{H}^{ab}_{\mathbf{R},\mathbf{R}+\mathbf{a}^*} + \mathbf{H}^{ac}_{\mathbf{R},\mathbf{R}+\mathbf{a}^*} + \mathbf{H}^{bb}_{\mathbf{R},\mathbf{R}+\mathbf{a}^*} \\
& + \mathbf{H}^{bc}_{\mathbf{R},\mathbf{R}+\mathbf{a}^*} + \mathbf{H}^{bd}_{\mathbf{R},\mathbf{R}+\mathbf{a}^*} + \mathbf{H}^{cc}_{\mathbf{R},\mathbf{R}+\mathbf{a}^*} + \mathbf{H}^{cd}_{\mathbf{R},\mathbf{R}+\mathbf{a}^*} \\
& + \mathbf{H}^{dd}_{\mathbf{R},\mathbf{R}+\mathbf{a}^*} + \mathbf{H}^{ab}_{\mathbf{R},\mathbf{R}} + \mathbf{H}^{bc}_{\mathbf{R},\mathbf{R}} + \mathbf{H}^{cd}_{\mathbf{R},\mathbf{R}} \\
& + \mathbf{H}^{ca}_{\mathbf{R},\mathbf{R}+\mathbf{a}^*+\mathbf{b}^*} + \mathbf{H}^{da}_{\mathbf{R},\mathbf{R}+\mathbf{a}^*+\mathbf{b}^*} + \mathbf{H}^{db}_{\mathbf{R},\mathbf{R}+\mathbf{a}^*+\mathbf{b}^*} + \mathbf{H}^{da}_{\mathbf{R},\mathbf{R}+\mathbf{R}+\mathbf{b}^*} \big)
\end{aligned}
\tag{4.57}
$$

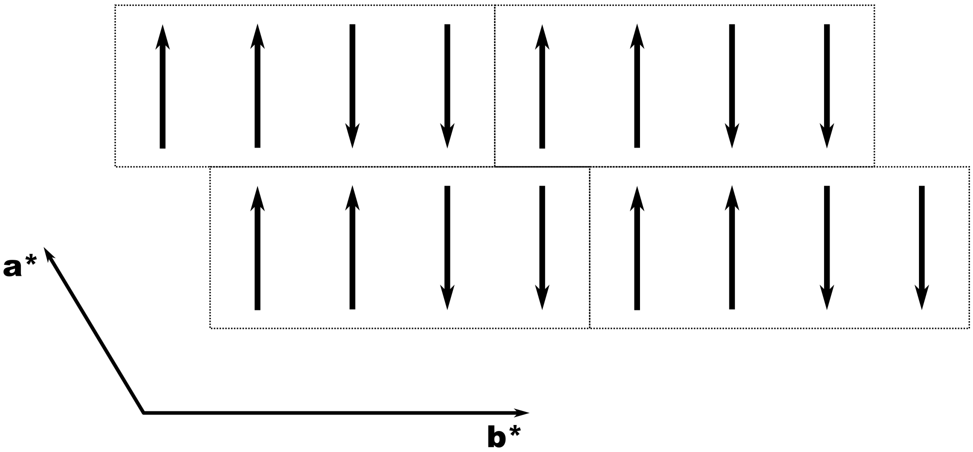


FIGURE 4.5: A suitable unit cell for FeTe in the bicollinear antiferromagnetic phase is a linear cell of four Fe ions with the unit vectors $\mathbf{a}^*$=(-1, 1) and $\mathbf{b}^*$=(4, 0) measured by the number of Fe ions moved along the parallelogram (note that the figure does not depict the structural distortions and simplifies the geometrical structure).

where the sum, $\sum_{\mathbf{R}}$, is over all unit cells in an infinite lattice. The effective Hamiltonian for the crystal, $\mathbf{H}_{\mathrm{FeTe}}$, is therefore the summation over 16 individual effective Hamiltonians for the bonds associated with each unit cell, which can themselves be expanded in terms of the orthogonal basis set, $\mathbf{Q}^{ijkl}$. The following pairs of individual effective Hamiltonians for the bonds between neighbouring Fe ions share the same coefficients: $\mathbf{H}^{aa}_{\mathbf{R},\mathbf{R}+\mathbf{a}^*}$ and $\mathbf{H}^{cc}_{\mathbf{R},\mathbf{R}+\mathbf{a}^*}$, $\mathbf{H}^{ab}_{\mathbf{R},\mathbf{R}+\mathbf{a}^*}$ and $\mathbf{H}^{cd}_{\mathbf{R},\mathbf{R}+\mathbf{a}^*}$, $\mathbf{H}^{ac}_{\mathbf{R},\mathbf{R}+\mathbf{a}^*}$ and $\mathbf{H}^{ca}_{\mathbf{R},\mathbf{R}+\mathbf{a}^*+\mathbf{b}^*}$, $\mathbf{H}^{ab}_{\mathbf{R},\mathbf{R}}$ and $\mathbf{H}^{cd}_{\mathbf{R},\mathbf{R}}$, $\mathbf{H}^{bb}_{\mathbf{R},\mathbf{R}+\mathbf{a}^*}$ and $\mathbf{H}^{dd}_{\mathbf{R},\mathbf{R}+\mathbf{a}^*}$, $\mathbf{H}^{bc}_{\mathbf{R},\mathbf{R}+\mathbf{a}^*}$ and $\mathbf{H}^{da}_{\mathbf{R},\mathbf{R}+\mathbf{a}^*+\mathbf{b}^*}$, $\mathbf{H}^{bd}_{\mathbf{R},\mathbf{R}+\mathbf{a}^*} + \mathbf{H}^{db}_{\mathbf{R},\mathbf{R}+\mathbf{a}^*+\mathbf{b}^*}$, and $\mathbf{H}^{bc}_{\mathbf{R},\mathbf{R}}$ and $\mathbf{H}^{da}_{\mathbf{R},\mathbf{R}+\mathbf{b}^*}$.

This effective Hamiltonian is the basis upon which the magnetic ground state of the FeTe system can be determined. Moreover, it is possible to calculate the spin waves of the system using a Holstein-Primakoff and Jordan-Wigner transformation that maps the spin and pseudo-spin operators onto boson and fermion creation and annihilation operators. Before proceeding with these quantum mechanical it would be informative to determine what the ground state of the system is by taking a classical approach to the spin and pseudo-spin operators, i.e., assuming that they are classical vectors.

4.5 Classical spin and pseudo-spin model

This section will explain how the ground state magnetic and orbital order pattern of the FeTe system can be found by assuming that the spins and pseudospins act like classical magnets. All the nine spin and four pseudo-spin operators on a site j and in unit cell at position $\mathbf{R}$ which appear in the Rayleigh-Schrödinger perturbative model's effective Hamiltonian can be written in terms of one of the components of the following operators:

$$\mathbf{S}_{j,\mathbf{R}} = \begin{pmatrix} \mathbf{S}^x_{j,\mathbf{R}} \\ \mathbf{S}^y_{j,\mathbf{R}} \\ \mathbf{S}^z_{j,\mathbf{R}} \end{pmatrix} \tag{4.58}$$

$$\mathbf{T}_{j,\mathbf{R}} = \begin{pmatrix} \mathbf{T}^x_{j,\mathbf{R}} \\ \mathbf{T}^y_{j,\mathbf{R}} \\ \mathbf{T}^z_{j,\mathbf{R}} \end{pmatrix} \tag{4.59}$$

The commutation relation between the spin and pseudo-spin operators on sites j and k in unit cells $\mathbf{R}$ and $\mathbf{R}'$ are given by:

$$[\mathbf{S}^a_{j,\mathbf{R}}, \mathbf{S}^b_{k,\mathbf{R}'}] = i\epsilon^{abc}\delta_{jk}\delta_{\mathbf{R}\mathbf{R}'}\mathbf{S}^c_{j,\mathbf{R}} \tag{4.60}$$

$$[\mathbf{T}^a_{j,\mathbf{R}}, \mathbf{T}^b_{k,\mathbf{R}'}] = i\epsilon^{abc}\delta_{jk}\delta_{\mathbf{R}\mathbf{R}'}\mathbf{T}^c_{j,\mathbf{R}} \tag{4.61}$$

where ϵ^{abc} is the Levi-Civita symbol, which is defined in the usual sense. Hence, both spin and pseudo-spin operators on different sites (within the same or in different unit cells) commute, whilst those on the same site and within the same unit cell do not commute. Furthermore, the magnitude of the spins and pseudo-spins on all sites is S and T, which leads to the following relations:

$$\mathbf{S}_{j,\mathbf{R}} \cdot \mathbf{S}_{j,\mathbf{R}} = S(S+1), \tag{4.62}$$

$$\mathbf{T}_{j,\mathbf{R}} \cdot \mathbf{T}_{j,\mathbf{R}} = T(T+1). \tag{4.63}$$

In the classical limit $S \to \infty$ and $T \to \infty$, the spin and pseudo-spin operators may be approximated by a form that includes their magnitudes and orientations:

$$\mathbf{S}_{j,\mathbf{R}} = S s_{j,\mathbf{R}}, \tag{4.64}$$

$$\mathbf{T}_{j,\mathbf{R}} = T t_{j,\mathbf{R}}, \tag{4.65}$$

and substituting these equations into the commutation relations for the spin and pseudo-spin operators in eq. 4.60 and 4.61 gives the following relations between the orientation operators, $s_{j,\mathbf{R}}$ and $t_{j,\mathbf{R}}$:

$$[\mathbf{s}^a_{j,\mathbf{R}}, \mathbf{s}^b_{k,\mathbf{R}'}] = \frac{1}{S} i\epsilon^{abc}\delta_{jk}\delta_{\mathbf{R}\mathbf{R}'}\mathbf{t}^c_{j,\mathbf{R}}, \tag{4.66}$$

$$[\mathbf{t}^a_{j,\mathbf{R}}, \mathbf{t}^b_{k,\mathbf{R}'}] = \frac{1}{T} i\epsilon^{abc}\delta_{jk}\delta_{\mathbf{R}\mathbf{R}'}\mathbf{t}^c_{j,\mathbf{R}}, \tag{4.67}$$

which means that in the classical limit of large magnitudes for the spins and pseudo-spins the right-hand side of the commutation relations tends to zero. The fact that these operators no longer commute indicates that they have lost their quantum properties and behave as classical vectors that have unit magnitude, i.e., $\mathbf{s}_{j,\mathbf{R}} \cdot \mathbf{s}_{j,\mathbf{R}} = 1$ and $\mathbf{t}_{j,\mathbf{R}} \cdot \mathbf{t}_{j,\mathbf{R}} = 1$. The spin operators used in the orthogonal basis set of the effective Hamiltonian can therefore be written in terms of these classical vectors:

$$\begin{gathered}\mathbf{S}^0 = S\mathbf{s}^2, \qquad \mathbf{S}^1 = S\mathbf{s}^z, \qquad \mathbf{S}^2 = S(\mathbf{s}^x + i\mathbf{s}^y), \\ \mathbf{S}^3 = S(\mathbf{s}^x - i\mathbf{s}^y), \qquad \mathbf{S}^4 = S^2(2\mathbf{s}^{z2} - \mathbf{s}^{x2} - \mathbf{s}^{y2}), \qquad \mathbf{S}^5 = S^2(\mathbf{s}^x + i\mathbf{s}^y)^2, \\ \mathbf{S}^6 = S^2(\mathbf{s}^x - i\mathbf{s}^y)^2, \qquad \mathbf{S}^7 = S^2\{\mathbf{s}^z, \mathbf{s}^x + i\mathbf{s}^y\}, \qquad \mathbf{S}^8 = S^2\{\mathbf{s}^z, \mathbf{s}^x - i\mathbf{s}^y\},\end{gathered} \tag{4.68}$$

and the pseudo-spin operators can also be written in a similar way:

$$\mathbf{T}^0 = T\mathbf{t}^2, \quad \mathbf{T}^1 = T\mathbf{t}^z, \quad \mathbf{T}^2 = T(\mathbf{t}^x + i\mathbf{t}^y), \quad \mathbf{T}^3 = T(\mathbf{t}^x - i\mathbf{t}^y). \tag{4.69}$$

Using these expressions for the spin and pseudo-spin operators, it is possible to formulate the effective Hamiltonian in the classical regime. The large spin and pseudo-spin approximations to the combinations of spin and pseudo-spin operators found in the effective Hamiltonian are presented below:

$$\begin{aligned}\mathbf{I} \otimes ... \otimes \mathbf{I} \otimes ... &\rightarrow S^2, \\ \mathbf{I} \otimes ... \otimes \mathbf{S}^4_j \otimes ... + \mathbf{S}^4_i \otimes ... \otimes \mathbf{I} \otimes ... &\rightarrow S^2(2\mathbf{s}_i^{z2} - \mathbf{s}_i^{x2} - \mathbf{s}_i^{y2}) + S^2(2\mathbf{s}_j^{z2} - \mathbf{s}_j^{x2} - \mathbf{s}_j^{y2}), \\ \mathbf{S}^1_i \otimes ... \otimes \mathbf{S}^1_j \otimes ... &\rightarrow S^2\mathbf{s}_i^z\mathbf{s}_j^z, \\ \mathbf{S}^4 \otimes ... \otimes \mathbf{S}^4 \otimes ... &\rightarrow S^2(2\mathbf{s}_i^{z2} - \mathbf{s}_i^{x2} - \mathbf{s}_i^{y2})(2\mathbf{s}_j^{z2} - \mathbf{s}_j^{x2} - \mathbf{s}_j^{y2}), \\ \mathbf{S}^p \otimes ... \otimes \mathbf{S}^m \otimes ... + \mathbf{S}^m \otimes ... \otimes \mathbf{S}^p \otimes ... &\rightarrow S^2(\mathbf{s}_i^x - i\mathbf{s}_i^y)(1 - 2S\mathbf{s}_i^z)(\mathbf{s}_j^x + i\mathbf{s}_j^y)(1 + 2S\mathbf{s}_j^z) \\ +S^2(\mathbf{s}_i^x + i\mathbf{s}_i^y)(1 + 2S\mathbf{s}_i^z)(\mathbf{s}_j^x - i\mathbf{s}_j^y)(1 - 2S\mathbf{s}_j^z),\end{aligned} \tag{4.70}$$

$$\begin{gathered}... \otimes \mathbf{I} \otimes ... \otimes \mathbf{I} \rightarrow T^2, \\ ... \otimes \mathbf{I} \otimes ... \otimes \mathbf{T}^x + ... \otimes \mathbf{T}^x \otimes ... \otimes \mathbf{I} \rightarrow T(\mathbf{t}_i^x + \mathbf{t}_j^x), \\ ... \otimes \mathbf{T}^x \otimes ... \otimes \mathbf{T}^x \rightarrow T^2\mathbf{t}_i^x\mathbf{t}_j^x.\end{gathered} \tag{4.71}$$

The effective Hamiltonian for a single Fe-Fe bond in the classical limit $S \rightarrow \infty$ and $T \rightarrow \infty$ is therefore comprised of the multiplication of these spin and pseudo-spin terms together (remembering that there are two terms associated with $\mathbf{S}^4$ and $\mathbf{T}^x$ appearing on the same site or on different sites) with the associated 16 weights. Labelling the weights $h_k \in \{1, 2, ..., 16\}$, the effective Hamiltonian for a single bond can be as:

$$
\begin{aligned}
\mathbf{H}_{ij} =& h_1 S^2T^2 + h_2 S^2T(\mathbf{t}_i^x + \mathbf{t}_j^x) + h_3 S^2T^2 \mathbf{t}_i^x \mathbf{t}_j^x \\
&+ h_4 S^2T^2\{(2\mathbf{s}_i^{z2} - \mathbf{s}_i^{x2} - \mathbf{s}_i^{y2}) + S^2(2\mathbf{s}_j^{z2} - \mathbf{s}_j^{x2} - \mathbf{s}_j^{y2})\} \\
&+ h_5 S^2T\{(2\mathbf{s}_i^{z2} - \mathbf{s}_i^{x2} - \mathbf{s}_i^{y2})\mathbf{t}_i^x + S^2(2\mathbf{s}_j^{z2} - \mathbf{s}_j^{x2} - \mathbf{s}_j^{y2})\mathbf{t}_j^x\} \\
&+ h_6 S^2T\{(2\mathbf{s}_i^{z2} - \mathbf{s}_i^{x2} - \mathbf{s}_i^{y2})\mathbf{t}_j^x + S^2(2\mathbf{s}_j^{z2} - \mathbf{s}_j^{x2} - \mathbf{s}_j^{y2})\mathbf{t}_i^x\} \\
&+ h_7 S^2T^2\{(2\mathbf{s}_i^{z2} - \mathbf{s}_i^{x2} - \mathbf{s}_i^{y2}) + S^2(2\mathbf{s}_j^{z2} - \mathbf{s}_j^{x2} - \mathbf{s}_j^{y2})\}\mathbf{t}_i^x \mathbf{t}_j^x \\
&+ h_8 S^2T^2 \mathbf{s}_i^z \mathbf{s}_j^z + h_9 S^2T \mathbf{s}_i^z \mathbf{s}_j^z (\mathbf{t}_i^x + \mathbf{t}_j^x) + h_{10} S^2T^2 \mathbf{s}_i^z \mathbf{s}_j^z \mathbf{t}_i^x \mathbf{t}_j^x \\
&+ h_{11} S^2T^2 (2\mathbf{s}_i^{z2} - \mathbf{s}_i^{x2} - \mathbf{s}_i^{y2})(2\mathbf{s}_j^{z2} - \mathbf{s}_j^{x2} - \mathbf{s}_j^{y2}) \\
&+ h_{12} S^2T (2\mathbf{s}_i^{z2} - \mathbf{s}_i^{x2} - \mathbf{s}_i^{y2})(2\mathbf{s}_j^{z2} - \mathbf{s}_j^{x2} - \mathbf{s}_j^{y2})(\mathbf{t}_i^x + \mathbf{t}_j^x) \qquad (4.72) \\
&+ h_{13} S^2T^2 (2\mathbf{s}_i^{z2} - \mathbf{s}_i^{x2} - \mathbf{s}_i^{y2})(2\mathbf{s}_j^{z2} - \mathbf{s}_j^{x2} - \mathbf{s}_j^{y2})\mathbf{t}_i^x \mathbf{t}_j^x \\
&+ h_{14} S^2T^2\{(\mathbf{s}_i^x - i\mathbf{s}_i^y)(1 - 2S\mathbf{s}_i^z)(\mathbf{s}_j^x + i\mathbf{s}_j^y)(1 + 2S\mathbf{s}_j^z) \\
&(\mathbf{s}_i^x + i\mathbf{s}_i^y)(1 + 2S\mathbf{s}_i^z)(\mathbf{s}_j^x - i\mathbf{s}_j^y)(1 - 2S\mathbf{s}_j^z)\} \\
&+ h_{15} S^2T\{(\mathbf{s}_i^x - i\mathbf{s}_i^y)(1 - 2S\mathbf{s}_i^z)(\mathbf{s}_j^x + i\mathbf{s}_j^y)(1 + 2S\mathbf{s}_j^z) \\
&(\mathbf{s}_i^x + i\mathbf{s}_i^y)(1 + 2S\mathbf{s}_i^z)(\mathbf{s}_j^x - i\mathbf{s}_j^y)(1 - 2S\mathbf{s}_j^z)\}(\mathbf{t}_i^x + \mathbf{t}_j^x) \\
&+ h_{16} S^2T^2\{(\mathbf{s}_i^x - i\mathbf{s}_i^y)(1 - 2S\mathbf{s}_i^z)(\mathbf{s}_j^x + i\mathbf{s}_j^y)(1 + 2S\mathbf{s}_j^z) \\
&(\mathbf{s}_i^x + i\mathbf{s}_i^y)(1 + 2S\mathbf{s}_i^z)(\mathbf{s}_j^x - i\mathbf{s}_j^y)(1 - 2S\mathbf{s}_j^z)\}\mathbf{t}_i^x \mathbf{t}_j^x.
\end{aligned}
$$

It is noted that the effective Hamiltonian contains terms that involve the interplay between the spin and orbitals physics. Whether the magnetic and orbital interactions are dominated by the interplay between the two is dependent on the magnitude of the weights.

4.5.1 Concluding the Study of the Magnetic and Orbital Order of FeTe

In summary, this study of $Fe_{1+y}Te$ has attempted to develop a theoretical framework for investigating and understanding the magnetic and orbital order of the FeTe system. It differs from previous studies that tried to explain the magnetic interactions of this condensed matter system in terms of either the spin degrees of freedom [72] or the orbital degrees of freedom in the 3d orbitals of the Fe atoms [74]. The theoretical frameworks within these two studies were unsuccessful in explaining the magnetic interactions within a sample of $Fe_{1+y}Te$ with a low concentration of interstitial iron, $y = 0.05$, as both of their spin or orbital degree of freedom Hamiltonians fitted poorly with the inelastic neutron scattering data of the system's spin waves [73]. Since the inelastic neutron scattering data revealed that the FeTe system has C2 rotational order at lower temperatures but short range C4 rotational order emerges above an energy of E=26meV (which is similar to the energy associated with the Jahn-Teller geometrical distortion in another iron-based system called magnetite [76]), it was evident that orbital degrees of freedom must play a role in the magnetic interactions.

The hypothesis for our study was that an effective Hamiltonian for the FeTe system that combined both spin and orbital degrees of freedom would better describe the magnetic interactions of the compound. The effective Hamiltonian for FeTe was developed using a Rayleigh-Schrödinger perturbative model that captured the magnetic interactions between pairs of nearest and next-nearest neighbour Fe atoms mediated by the 5p orbitals on Te atoms. Virtual electron hopping between each bond was extended to fourth order processes allowing for the exchange of electrons between neighbouring Fe atoms while maintaining the S=1 spin state of each of the Fe atoms. The effective Hamiltonian was expanded in terms of an orthogonal basis set that included 9 spin and 4 orbital pseudo-spin operators to give terms that involved spin-only, orbital pseudo-spin only or an interplay between the spin and orbital physics of the system. The terms involving the interplay between the spin and orbital degrees of freedom were non-negligible indicating that a full understanding of the magnetic interactions of the FeTe system must include these terms. The theoretical framework developed for this project has a high degree of complexity with lots of possible magnetic states dependent on the input variables including the degree of geometrical distortion and shielding parameters.

It is recommended that with further time it could be used to calculate the magnetic and orbital ground state of the FeTe system and calculate its spin wave dispersion. A fit of the spin waves to the experimental inelastic neutron scattering data [73] could then be used to refine the inputs to the model and reveal more information about the electronic shielding within the crystal. The beauty of this model is that it is essentially an insulator model with some small adjustments for the shielding of conduction elections. Hence, if a fit to the inelastic neutron scattering data is possible, it would mean that notwithstanding the metallic state of the FeTe system it can be described by an insulator model.

Chapter 5

Study of the disorder in PMN

In the preceding sections, the reader has been introduced to the basics of inelastic neutron scattering techniques and the horizontal Bridgman technique for the synthesis of condensed matter systems, including iron chalcogenides and perovskite semiconductors. This chapter of the thesis will detail the analysis of inelastic neutron scattering data on a similar perovskite structure, $PbMg_{1/3}Nb_{2/3}O_3$ (PMN), that was collected on the MERLIN direct geometry ToF chopper instrument at ISIS in Oxford, UK. One of the interesting features of $PbMg_{1/3}Nb_{2/3}O_3$ is that it is a disordered perovskite that shares a lot of common traits with more ordered perovskites, including strontium titanate, $SrTiO_3$. For this reason, the analysis also included theoretical calculations of the phonon dispersion spectrum along a high symmetry direction of the $SrTiO_3$ perovskite crystal using a software package called Quantum Espresso [93]. These calculations formed the basis for understanding the degree to which disorder is important for the lattice dynamical behaviour, particularly for the higher energy phonons in $PbMg_{1/3}Nb_{2/3}O_3$. Some of the important reference studies undertaken in the past on the lattice dynamics of $PbMg_{1/3}Nb_{2/3}O_3$ that are used to make a comparison with this analysis of the inelastic neutron scattering experiment include [94–96].

5.1 Neutron scattering and sample details

As has been mentioned the inelastic neutron scattering experiment studying phonons within $PbMg_{1/3}Nb_{2/3}O_3$ was carried out on the MERLIN chopper spectrometer instrument (STFC-ISIS, UK). The single crystals were grown by Dr Chris Stock using a modified Bridgman technique similar to those described in Sec. 2.2 and 2.4. The sample used for the experiment consisted of three single crystals that produced an effective 192g single crystal (see Fig. 5.1). The reason for combining the three single crystals together was to take advantage of the larger magnitude of scattering that would take place given the same incident flux of neutrons from the source. The only requirement was that the three single crystals have all their crystal axes aligned in order to produce the effective single crystal, otherwise the result would be something similar to that described for polycrystalline samples that are formed by crystallites meeting at grain interfaces with their crystal axes randomly misaligned due to the formation of nuclei at several different sites within the melt during synthesis.

Each sample was aligned such that Bragg reflections of the form (HK0) lay within the horizontal plane (that is the plane where the sample and the detector array are co-planar). Two different incident energies were used (E_i=75 and 150meV) and the Fermi chopper frequency was set to 300Hz. The multi-repetition rate was not utilised for this ToF experiment meaning that two separate experiments were carried out sequentially with these control parameters. The incident energies afforded an energy resolution (full-width at half maximum) at the elastic line, i.e., $\Delta E=0$, of approximately 5meV and 12meV respectively. The MERLIN direct geometry ToF spectrometer was chosen since the resolution of direct geometry ToF spectrometers increases with increasing energy transfer—$\Delta E \sim 2$meV at $E \sim 70$meV when using the 75meV incident neutrons and $\Delta E \sim 5$meV at $E \sim 100$meV when using the 150meV incident neutrons.

A wide range of energy and momentum transfers are accessible with the MERLIN chopper spectrometer, but this study mainly focuses on the lattice dynamical behaviour for the higher energy phonon modes along the Γ-X high symmetry direction (see Fig. 5.1). Due to the cubic symmetry of $PbMg_{1/3}Nb_{2/3}O_3$ this meant that there were three symmetry directions along which to observe the same lattice dynamical behaviour, i.e., scattering along the [100], [010] and [001] directions in reciprocal space. Furthermore, to take advantage of the higher neutron cross sections at higher momentum transfers (the double differential neutron scattering cross section is proportional to $|\mathbf{Q}\cdot\mathbf{e}|^2$) the focus of the experimental analysis was on the (H00), (0K0), (00L), (H40), (4K0) and (40L) axes in momentum space. Since the scattering of phonons depends on the dot product of the scattering vector, $\mathbf{Q}$, and the atomic displacements within the crystal, $\mathbf{e}$, and the [100], [010] and [001] axes are high symmetry directions in cubic structures, the (H00), (0K0) and (00L) momentum transfers should only contain pure longitudinal modes, whilst the (H40), (4K0) and (40L) could contain both pure transverse and pure longitudinal modes in ordered perovskites. However, any disorder within the crystal that disrupts the cubic symmetry could lead to quasi-transverse modes being present in the scattering of both the longitudinal and transverse channels.

5.2 Theoretical calculations using density functional perturbation theory

Density functional theory calculations of the phonon bandstructure in $SrTiO_3$ were completed using Quantum Espresso. This is a suite for ab initio quantum chemistry methods for electronic structure calculations and other material property modelling [97, 98]. The ordered chemical structure of the $SrTiO_3$ crystals, as opposed to the disordering in the B-sublattice of PMN, meant that it was far easier to theoretically model the phonon band structure of $SrTiO_3$. However, due to their similar overall chemical structure it was supposed that using $SrTiO_3$ as an input to the ab initio code would be an aid for our study of the higher energy phonon modes for PMN, whose dispersions have so far eluded all recent attempts at studying them. This assumption may break down if the electronic structure of the two crystals differ substantially owing to the different orbital

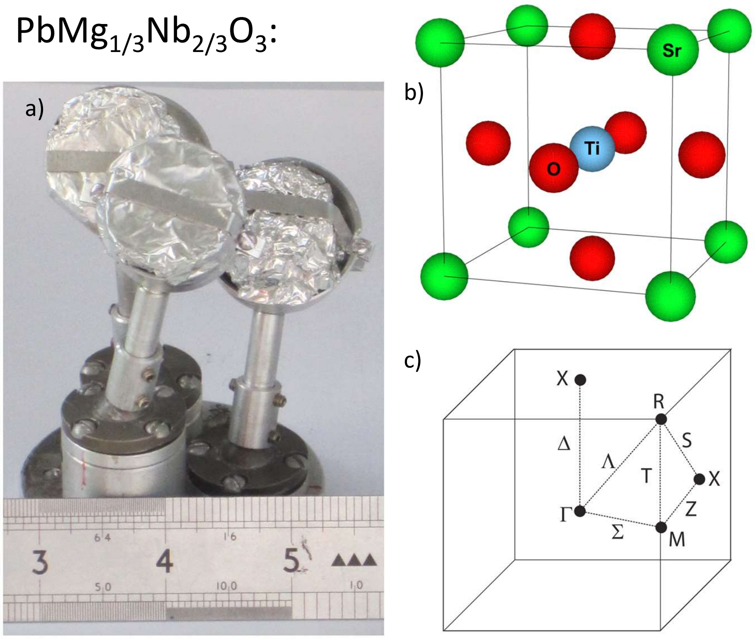


Figure 5.1: (a) Three separate single crystals of PMN with a total mass of 192g were used in this inelastic neutron scattering experiment. A vanadium sample holder is used owing to its large incoherent scattering cross section compared with its coherent scattering cross section. (b) The unit cell of SrTiOs with titanium atoms occupying the B-site of the perovskite structure. This is used as an ordered proxy for doing calculations of the lattice dynamical behaviour of PMN crystals. (c) shows the high symmetry paths in reciprocal space- this study focuses on the Γ-X path.

physics of the Ti atoms in $SrTiO_3$ and the Nb and Mg atoms in PMN. This is due to the fact that the orbital structure plays a non-negligent role in determining the overall interatomic forces between the constituent atoms in condensed matter systems. The interatomic forces then determine the dispersion curves of the normal modes of vibration. In the past, neutron scattering experiments have failed to reveal the structure to the higher energy phonon modes and have only managed to find a wide band of continuous excitations between 27–50meV, an excitation with both longitudinal and transverse character at 55meV, a further weak signal at 66meV and a couple bands of phonon excitations above 80meV [65].

The model developed here for $SrTiO_3$ can, moreover, be compared with models **I**–**V** (see Fig. 5.2 and Fig. 5.3) for the same perovskite structure in the 1960s by R. A. Cowley [95]. In this study, the crystal dynamics were studied experimentally using neutron spectroscopy at the Chalk River Laboratories of Atomic Energy of Canada Ltd. Some of the normal modes propagating along the [100] direction were characterised. The neutron spectroscopy data confirmed, along with other infrared spectroscopy measurements, the results of the phonon band structure models thus developed; however, a scarcity of data above 40meV made it difficult to confirm, if any, of the models could accurately predict the higher energy longitudinal and transverse optical branches.

All the models predicted a pair of optical phonon branches at higher energies in the longitudinal configuration, which are separated by a large band gap. They put the lower energy optical branch at around 66meV, whereas the higher energy branch is situated at approximately 90meV. In contrast, there is substantial difference between the predicted phonon band structure for $SrTiO_3$ in the transverse configuration. Models **III**–**V** predict that a transverse optical branch will exist at approximately 70meV, whilst model **I**'s results diverge substantially from this prediction. Instead, it predicts that another higher energy optical phonon branch will be present close in energy, and likewise flat in energy, to the highest energy longitudinal optical phonon branch. This latter model is in contradiction with the density functional theory calculations (see Fig 5.4) since the latter does not predict the presence of a transverse optical mode with energies close to the highest energy longitudinal optical mode. There is also substantial disagreement between the theoretical models in R. A. Cowley's paper [95] and our present predictions: ours show a further set of optical phonon branches between those situated below 40meV and the higher energy LO branch. As will be demonstrated the inelastic neutron scattering data collected for this experiment is in agreement to a greater extent with this new analysis than any particular model developed in R. A. Cowley's paper on SrTiO3 [95].

The phonon band structure along directions [100] was calculated using Quantum Espresso making use of the PWscf and PHonon packages for the self-consistent calculations and phonon density functional theory calculations respectively. The default inputs for both the convergence threshold for the self-consistency calculation and the mixing beta were used since changing them did not lead to significantly different end results. Owing to the cubic symmetry in the perovskite crystal lattice this direction reproduces the same phonon band

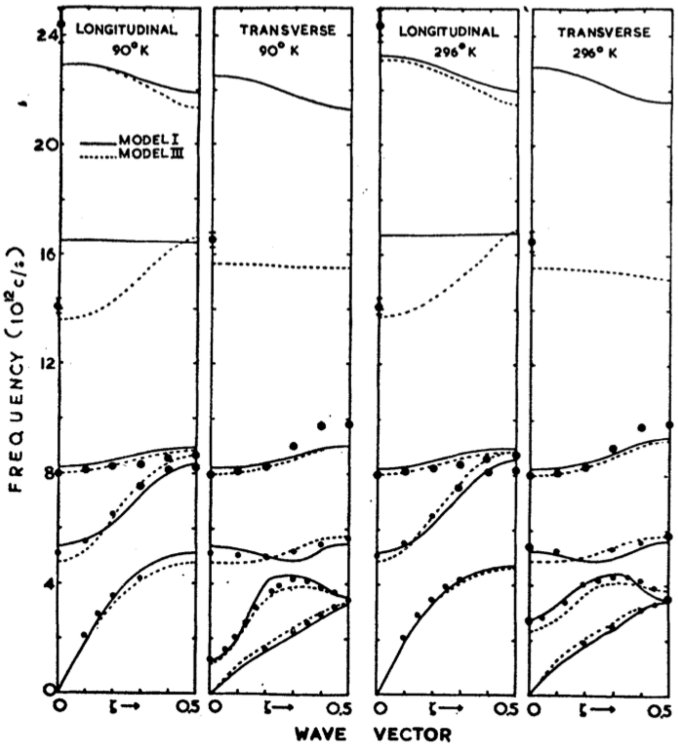


FIGURE 5.2: The phonon dispersions for $SrTiO_3$ using models **I** and **III** showing the transverse and longitudinal models propagating along the direction [1, 0, 0] [95]. In each of these models there is only one longitudinal mode present above a frequency of $20{\times}10^{12}c/sec$.

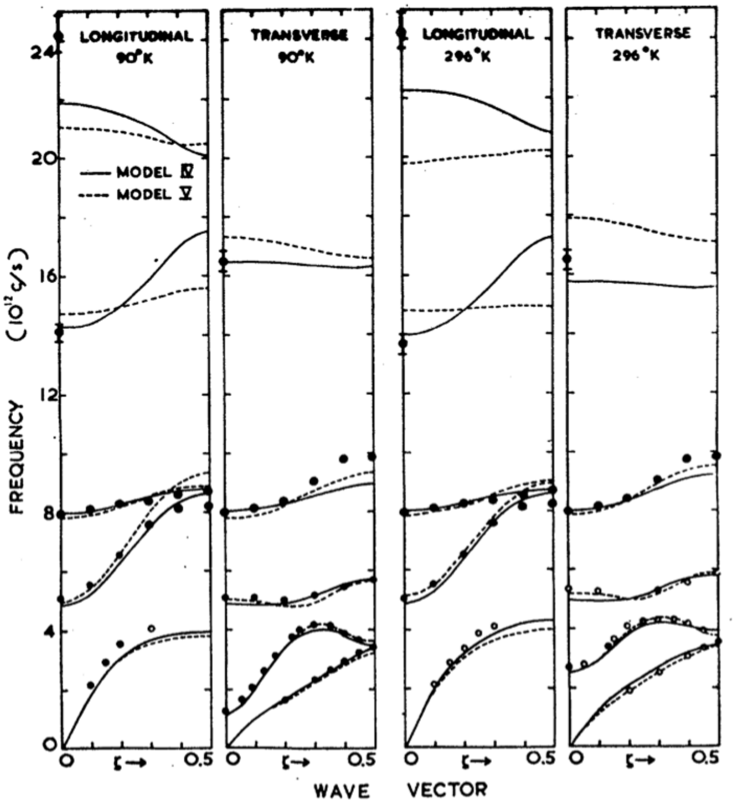


FIGURE 5.3: The phonon dispersions for $SrTiO_3$ using models **IV**–**V** showing the transverse and longitudinal models propagating along the direction [1, 0, 0] [95]. In contrast with models **I** and **III**, models **IV**–**V** both have one transverse and one longitudinal model above a frequency of $20{\times}10^{12} c/sec$.

structure as in the other symmetric directions, viz., along the [010] and [001] directions. By comparing the output of this density functional theory calculation with the previous results obtained [65], it is possible that the second and third highest energy modes that were seen correspond to the interweaving optical phonon modes between 55meV and 62meV. Furthermore, it was not possible to experimentally determine the character of the highest energy mode, which according to this calculation is meant to exist at approximately 93meV and be flat in energy. One of the main objectives of this research project is thus to utilise the recent advancements in inelastic neutron scattering experiments to detect this highest energy mode and also to verify some of the results for the middle phonon branches. It should be noted that the calculations are unstable for one of the branches, as seen in Fig. 5.4, which is most likely a result of the real instability in these perovskites that is associated with the presence of the softened transverse optical mode—an analysis of the eigenvectors for the modes outputted from the Quantum Espresso calculations does in fact confirm that this unstable mode belongs to a transverse optic mode.

5.3 Background subtraction algorithm

5.3.1 Motivation and development of the background subtraction algorithm

The study of phonons in crystals plays an important role in understanding how some of the microscopic properties of materials manifest themselves in the bulk properties of solids. This is most relevant in the study of a crystal's thermal and electrical conductivity as both are intricately tied to the existence of phonons—the former because phonons can carry heat from one side of the crystal to another and the latter because electron-phonon scattering interactions give rise to higher electrical resistivity and, ergo, lower electrical conductivity. In condensed matter physics, one of the objectives for an experiment involving phonon scattering is to calculate the phonon dispersion curve of a crystal as this gives vital information about the way phonon modes behave as a function of their energy. However, one of the inherent difficulties in determining this dispersion curve in real scattering experiments is that the scattering intensity, $I(\mathbf{Q}, \omega)$, is broadened due to many factors related to the experimental setup (i.e, $\mathbf{Q}$-dependent resolution function of the detectors, the mosaicity of the crystals used as monochromators giving rise to peak broadening of the incident neutrons, etc.) and, further, that the scattering intensity is a superposition of many different types of interactions that are either elastic or inelastic and coherent or incoherent in nature. Isolating for the type of scattering that one is interested in is therefore of vital importance for experimental condensed matter scientists, and it is here that this investigation hopes to provide a systematic method for the removal of elastic and incoherent scattering contributions from the overall

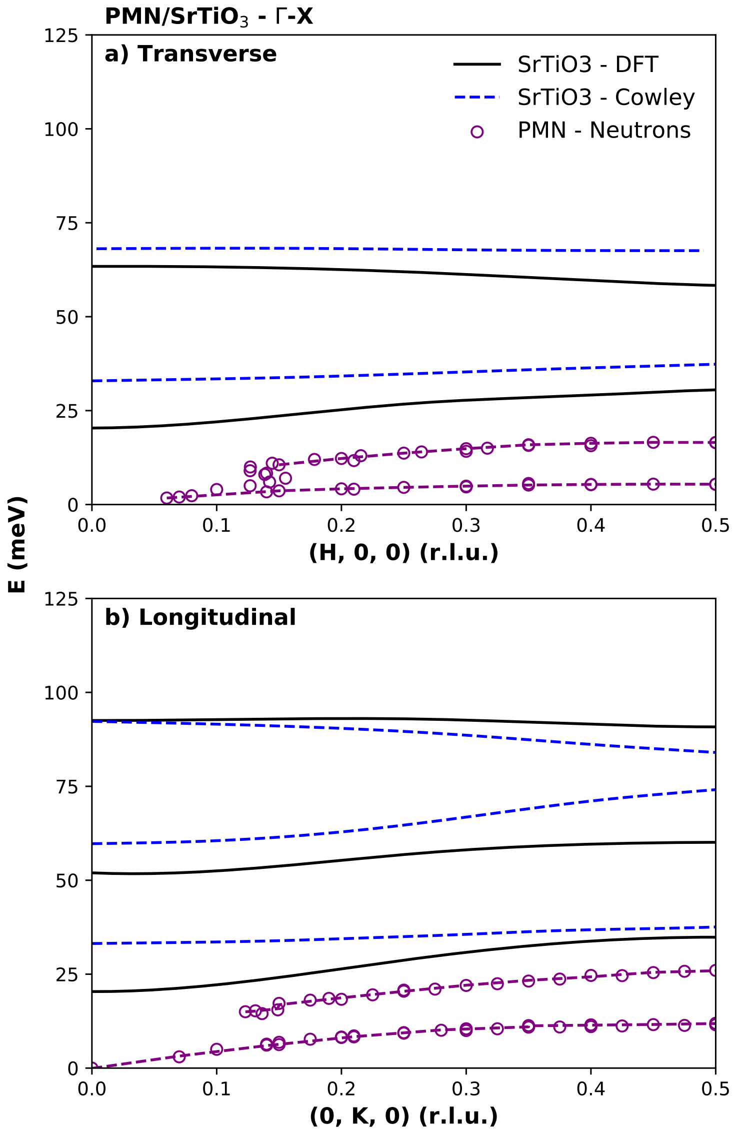


FIGURE 5.4: Density functional perturbation theory calculations were performed on SrTiO$_3$ so as to determine the phonon band structure of an ordered equivalent of PMN along the [100] direction (Γ–X). The presence of only one longitudinal optic mode above 80meV is one theoretical prediction of our calculations that is in agreement with models **III–V** of the R. A. Cowley paper [95]. Our density function theory calculations also predict that there are two degenerate transverse optical modes and one longitudinal optical mode between 50meV and 70meV. In the transverse channel (a) and the longitudinal channel (b), the plots include the DFT calculations from this present work (black line), some predictions from the R. A. Cowley paper [95] (dashed blue line) and some inelastic neutron scattering data for PMN from Fig. 5 of Swainson et al. [99] that looked at soft phonons at the edge of the Brillouin zone.

intensity profile, $I(\mathbf{Q},\omega)$. This background subtraction algorithm was successfully applied to the study of phonon modes in lead magnesium niobate, PMN, which, as has been partly stated, is an interesting ferroelectric relaxor ceramic with piezoelectric properties that is used in lots of optoelectronics and acousto-optics applications in industry.

5.3.2 Description of the algorithm

When doing any kind of scattering experiment involving a direct geometry ToF instrument the detectors indirectly measure the energy and momentum of scattered particles and quasiparticles, and, due to the electronics involved in the post-processing of the scattered particles used for probing the sample, the scattering intensity, $I(\mathbf{Q},\omega)$, is effectively a 4-dimensional histogram with the $\mathbf{Q}$ and ω-widths determined by the electronics. Hence, the value given to each bin in the histogram is actually the integrated intensity, $\tilde{I}(\mathbf{Q},\omega)$:

$$\tilde{I}(\mathbf{Q},\omega) = \int_{\Delta} d\mathbf{Q} d\omega I(\mathbf{Q}',\omega'), \tag{5.1}$$

where Δ is the 4-dimensional hypervoxel centred at $(\mathbf{Q},\ \omega)$ and with widths $\Delta\mathbf{Q}$ and $\Delta\omega$ defined such that the volume of the hypervoxel is $16|\Delta\mathbf{Q}|^2\Delta\omega^2$. The experimental results are often then plotted with either two or three of the coordinates in the 5-dimensional space, Ω, removed, thus reducing the dimensionality for data visualisation purposes.

5.3.3 Modelling the phonon scattering of neutrons

In Sec. 1.2.3 there was a detailed explanation of the form that neutron scattering takes for interactions with phonons within a single crystal, specifically the double differential cross section for one-phonon interactions. The form of incoherent and coherent scattering by neutrons of phonons will be used to develop a background subtraction algorithm that extracts more information from the scattering intensities obtained from the experiments undertaken on the MERLIN direct geometry ToF spectrometer instrument. Since more of the structure and dynamics are contained within the coherent scattering, the principal aim of the algorithm is to eliminate as much of the incoherent scattering as possible. Some of the assumptions made in developing the algorithm, which are often made to keep a degree of computational simplicity, will introduce a small amount of error in the analysis, thus all assumptions that are used in the computation will be explained and put into context by justifying why it is possible to make such assumptions based on the magnitude of certain experimental parameters.

The important contributions to the overall intensity profile, $I(\mathbf{Q},\omega)$, are the coherent and incoherent elastic scattering interactions, centred at $\omega = 0$, and the coherent and incoherent single and multi-phonon inelastic scattering interactions, $\omega \neq 0$. Understanding how these behave as a function of both $\mathbf{Q}$ and ω helped to produce this algorithm for isolating the coherent inelastic phonon peaks in the neutron scattering data.

If we assume, without loss of generality to the algorithm, that all of the scattering centres have the same mass, then, from G. L. Squires, the differential cross section of coherent inelastic single-phonon scattering is given by:

$$\left(\frac{d^2\sigma}{d\Omega dE'}\right)_{\text{coh},\Delta n=1} \propto \frac{Q-k}{k}\exp^{-2W}\Sigma_s\Sigma_{\mathbf{G}}\frac{(\mathbf{Q}\cdot\mathbf{e}_s)^2}{\omega_s} \langle n_s+1\rangle\,\delta(\mathbf{Q}-\mathbf{q}-\mathbf{G})\delta(\omega-\omega_s) \tag{5.2}$$

$$\left(\frac{d^2\sigma}{d\Omega dE'}\right)_{\text{coh},\Delta n=-1} \propto \frac{Q-k}{k}\exp^{-2W}\Sigma_s\Sigma_{\mathbf{G}}\frac{(\mathbf{Q}\cdot\mathbf{e}_s)^2}{\omega_s} \langle n_s\rangle\,\delta(\mathbf{Q}+\mathbf{q}-\mathbf{G})\delta(\omega+\omega_s) \tag{5.3}$$

where k is the incident momentum of the neutrons, W is the Debye-Waller factor, $\mathbf{e}_s$ is the polarisation vector of phonon mode, and $\hbar\omega_s$ is the energy of mode s. Moreover, Δn is the change in the number of phonons caused by the scattering interaction with Δn=1 indicating that the number of phonons has increased by one (absorption) and Δn=-1 indicating the reverse, namely the emission of a phonon. In the direct geometry ToF spectrometer the incident neutron momentum is kept constant. Assuming that the crystal is isotropic and therefore that the polarisation vector does not favour one particular direction in reciprocal space, then this leads to a contribution of the intensity profile, $I(\mathbf{Q},\omega)$, that can be modelled as:

$$I(\mathbf{Q},\omega)_{\text{coh}} = B(Q,\omega)Q^2, \tag{5.4}$$

where B is some function to be determined. The same analysis can be done for the incoherent inelastic phonon scattering contribution, with the differential cross section of this going as:

$$\left(\frac{d^2\sigma}{d\Omega dE'}\right)_{\text{incoh},\Delta n=\pm 1} \propto \frac{Q-k}{k}\exp^{-2W}\Sigma_s\Sigma_{\mathbf{G}}\frac{(\mathbf{Q}\cdot\mathbf{e}_s)^2}{\omega_s} [\langle n_s+1\rangle\,\delta(\omega-\omega_s)+\langle n_s\rangle\,\delta(\omega+\omega_s)]. \tag{5.5}$$

Hence, this can also be modelled using the same form as that given by eq. 5.4.

Lastly, the coherent elastic scattering contribution has the following differential cross section:

$$\left(\frac{d^2\sigma}{d\Omega dE'}\right)_{\text{el-coh}} \propto \exp^{-2W}\Sigma_\tau\delta(\mathbf{Q}-\mathbf{G}). \tag{5.6}$$

This clearly lacks the quadratic dependence in $\mathbf{Q}$ that is present in both the coherent and incoherent inelastic cross sections. Instead of modelling the elastic contribution in the form of eq. 5.4, this is modelled with the following function:

$$I(\mathbf{Q},\omega) = A(\mathbf{Q},\omega). \tag{5.7}$$

Combining the two expressions for the different contributions to the phonon scattering, the overall intensity profile can be modelled as:

$$I(\mathbf{Q},\omega) = A(\mathbf{Q},\omega) + B(\mathbf{Q},\omega)Q^2. \tag{5.8}$$

The benefit of this analysis is that the terms can contain information about the physics of the scattering interactions and valuable information about the broadening of peaks (the latter of which can be due to a multitude of reasons). For instance, a non-zero value for $A(\mathbf{Q}, \omega)$ at $\omega \neq 0$ indicates that there is broadening of the elastic line, which is otherwise known as quasi-elastic neutron scattering. The same can also be said for the $B(\mathbf{Q}, \omega)$ term, viz., that some broadening of the coherent peaks will be observed for $\omega \neq \omega_s$.

5.3.4 Isolating the coherent inelastic phonon peaks

The expression for the phonon scattering given by eq. 5.8 can be used to subtract the elastic line and incoherent inelastic phonon contributions from the overall intensity profile, $I(\mathbf{Q}, \omega)$. If the elastic and incoherent single and multi-phonon inelastic scattering contributions are now given by eq. 5.8, then subtracting this from the overall intensity profile found via neutron scattering experiments, $I(\mathbf{Q}, \omega)_{\text{exp}}$, should give the contribution from just the coherent inelastic phonon scattering interactions, $I(\mathbf{Q}, \omega)_{\text{in-coh}}$:

$$I(\mathbf{Q}, \omega)_{\text{in-coh}} = I(\mathbf{Q}, \omega)_{\text{exp}} - I(\mathbf{Q}, \omega)_{\text{back}}, \tag{5.9}$$

where $I(\mathbf{Q}, \omega)_{\text{back}}$ now represents the elastic and incoherent inelastic scattering contributions to the overall intensity profile. The background subtraction algorithm outlined in this paper is mostly concerned with accurately, and without too much computational labour, determining $I(\mathbf{Q}, \omega)_{\text{back}}$. Doing this well is conducive to having neutron scattering plots with a strong contrast between the coherent phonon scattering peaks and other contributions to the phonon scattering that is less relevant for the determination of the dispersions of the normal modes of a crystalline solid.

5.3.5 Assumptions in development of the algorithm

One of the difficulties in confronting this computational problem is that the intensities are given for each hypervoxel. This makes the problem multi-dimensional and dramatically increases the computational load. The philosophy of the background subtraction algorithm is thus to reduce the dimensionality of the problem without drastically reducing the accuracy of the results. The main assumption that is made is that the expressions $A(\mathbf{Q}, \omega)$ and $B(\mathbf{Q}, \omega)$, which contain important information about the dynamics of the solid, can be reduced, without loss of too much information, to functions of the form $A(E)$ and $B(E)$, thus reducing the optimisation problem from four to only one dimension. The elastic and incoherent phonon scattering background now take the following form, which is updated from eq. 5.8:

$$I(\mathbf{Q}, E)_{\text{back}} = A(E) + B(E)Q^2. \tag{5.10}$$

There are two sources of inaccuracy that this introduces into the analysis: one is that the coherent elastic phonon scattering contribution to the intensity profile, i.e., Bragg Scattering, is centred at the reciprocal lattice vectors, $\mathbf{G}$, which this expression ignores; and secondly, that the quadratic term in eq. 5.10

ignores the $\frac{k-Q}{k}$ pre-factor to the double summation of eq. 5.5. The former introduces greater inaccuracies at lower energies since this is closer to the elastic line and further away the broadening of peaks dampens the inaccuracy. The latter leads to momentum-transfer dependent inaccuracies that are greater at larger Q values.

5.3.6 Computational tasks of the algorithm

As the algorithm's main objective is to find the form of both $A(E)$ and $B(E)$ as a function of the energy transfer, the first step, after having produced .sqw files using Horace (a suite of programmes for the analysis of datasets from time-of-flight neutron inelastic scattering spectrometers [100]) is to produce constant energy cuts from E=0 to some upper energy threshold, which is limited by the incident energy of the neutrons in the beam. The width of the integration over E in the cuts is chosen to counterweight the requirements of having both large statistics and precision in the energy scale. Each of the constant energy cuts produces a file containing the following data: position in reciprocal space of the Q-voxel, intensity and the uncertainty in intensity.

The constant energy cuts are then put into equally spaced bins according to the magnitude of their momentum transfer vector, $|\mathbf{Q}|$. Plotting the intensity as a function of $|\mathbf{Q}|$ aids in the determination of both functions in eq. 5.10. These are calculated using a form of regression analysis that uses a penalised cost function so as to exclude the peaks arising due to coherent inelastic phonon scattering, but include the broadened elastic line and also the continuous background due to incoherent inelastic phonon scattering. This cost function is of the following form:

$$S = \sum_i w_i (y_i - z_i)^2 + \lambda \sum_i (\Delta^2 z_i)^2, \tag{5.11}$$

where $\Delta z_i = z_i - 2z_{i-1} + z_{i-2}$. The first summation is a measurement of the fit of the regression curve, z_i, to the experimental data, y_i, whilst the second term is a penalty on the non-smoothness of the data. The weights, w_i, can also be introduced to give an asymmetry in the cost function so that there is a greater penalty arising when the regression curve is above the data than when it is below it. Minimisation of the cost function in eq. 5.11 leads to the following set of algebraic equations:

$$(\mathbf{W} + \lambda \mathbf{D}^\intercal \mathbf{D})\mathbf{z} = \mathbf{y}, \tag{5.12}$$

where $\mathbf{W}$ is a matrix with diagonals equal to the weights and zero off-diagonals and $\mathbf{D}$ is called the difference matrix in that $Dz = \Delta^2 z$. The parameters used in the regression analysis were optimised and the values used in this study of PMN were chosen to be $\lambda = 10^2$ and w_i such that:

$$w_i = \begin{cases} p, & \text{if } y_i > z_i \\ 1-p, & \text{otherwise} \end{cases} \tag{5.13}$$

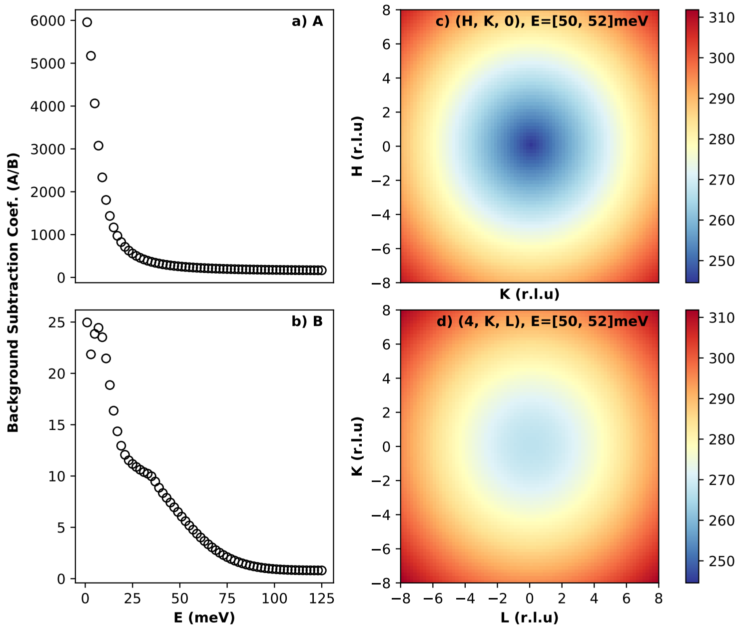


Figure 5.5: The background polynomial coefficients, $A(E)$ and $B(E)$, are shown up to 125meV in subplots (a) and (b) respectively. They were fitted to a combination of Gaussian and Lorentzian functions to make the background more smooth as a function of the energy transfer. Subplots (c) and (d) show the background for constant energy scans ([50, 52]meV) for $\mathbf{Q} = (H, K, 0)$ and $\mathbf{Q} = (4, K, L)$ respectively. The background is meant to eliminate as much as possible the contributions to the scattering emanating from elastic scattering and incoherent inelastic neutron scattering processes.

where $p = 10^{-2}$. As in previous studies of this asymmetric least squares analysis, only ten iterations were found to be necessary for sufficient convergence of the z_i values to the baseline of the data. These z_i values were then used to fit, using standard regression analysis methods, a curve of the form in eq. 5.10. The $A(E)$ and $B(E)$ functions were thus optimised using the discrete data from the constant energy cuts. They were subsequently fitted to Lorentzian and Gaussian functions so as to give a smooth background for each of the hypervoxels in the time-of-flight neutron scattering dataset. Fig 5.5 shows the fits for the $A(E)$ and $B(E)$ curves when applied to the 75meV incident neutron data. Using eq. 5.10 these polynomial coefficients were subsequently used by the algorithm to calculate the background as a function of both the energy and momentum transfer. Fig 5.6 gives an example of how the algorithm succeeds in removing the background from the neutron scattering plots and makes the phonon excitations much more discernible irrespective of whether it is at low or high momentum scattering or energy transfer.

In order to further improve the statistics of the scattering intensity data, it was also decided to combine all data along symmetric axes owing to the 4-fold symmetry inherent in the cubic crystal structure. This meant that the [100], [010] and [001] directions could be combined to form one $S(\mathbf{Q}, \omega)$ plot. In addition, the resultant plots were then mirrored about the zero momentum transfer axis to produce a final scattering plot that is ready to be used as an input to the background subtraction algorithm. This process of increasing the statistics of the scattering intensities is shown in Fig 5.7, which includes $S(\mathbf{Q}, \omega)$ plots in the (000) and (400) Brillouin zones and along the [100], [010] and [001] directions.

5.4 Analysis of the neutron scattering data

The inelastic neutron scattering data was collected using the up-scattering regime (neutron energy-gain mode), operating with an incident wavevector 6.02Å^{-1} (E$_i$=75meV) in one experiment and an incident wavevector 8.51Å^{-1} (E$_i$=150meV) in another. The former offered a maximum Q range of ∼6.7a* along the [100] direction and other symmetric axes starting from the (000) Brillouin zone and ∼5.9a* along the [010] direction starting from the (400) Brillouin zone. The higher incident energy configuration offered a maximum Q range of ∼8.8a* in the [100] direction and other symmetric axes starting from the (000) Brillouin zone and ∼8.2a* along the [010] direction originating in the (500) Brillouin zone. These maximum Q ranges are defined as the range of available data close to the elastic line in the up-scattering regime and in the positive momentum transfer quadrant and come from the fact that the neutron scattering events conserve both energy and momentum. This Q range was sufficient for studying the whole range of lattice dynamical properties of the perovskite crystal and especially for making use of the quadratic dependence of the coherent inelastic phonon scattering cross section. The wide range in Q allowed for an accurate determination of the phonon density of states (DOS) (see Fig. 5.8 and 5.9). This helped to locate the positions of the optic phonon branches above 40meV and to compare these experimentally observed phonon branches with those theoretically determined from density functional perturbation theory calculations and previous models published in the literature [95]. As will be explained in this section, there is some agreement between the dispersions of the higher energy optic phonon branches in this experiment and models III–V developed by R. A. Cowley, but there is substantial disagreement with some of the continuous regions of neutron scattering seen by B. Dorner et al. (see Fig. 5.11).

Fig. 5.8 shows the scattering intensity, $S(\mathbf{Q}, \omega)$, for the E$_i$=75meV and E$_i$=150meV incident neutrons along with the constant energy integrated scattering intensities. These integrated intensities give an approximate calculation for the density of states of the phonon branches and help detect branches with strong dependence on $\mathbf{Q}$. The pseudo-DOS plots along the [100] direction starting at the (000) Brillouin zone substantially agrees with the density functional theory calculations for $SrTiO_3$: the peaks in the pseudo-DOS around 48–60meV is evidence of the presence of middle energy optic phonon branches in

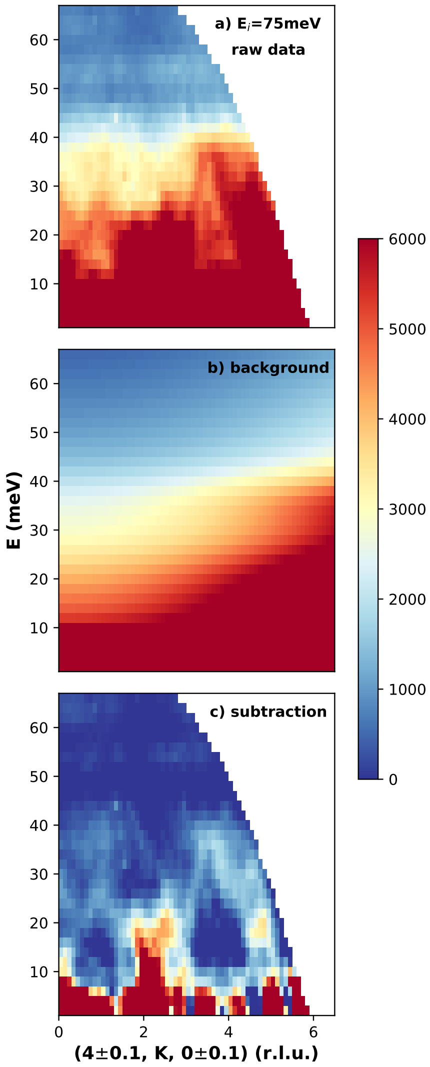


FIGURE 5.6: Neutron scattering intensities along $(4, K, 0)$. Top: the original scattering intensities without applying the background subtraction. Middle: background intensities calculated for each pixel in the 2D plane with the $\mathbf{Q}$-dependence clearly visible. Bottom: the resultant scattering intensities when the background has been removed- this attempts to isolate only the coherent inelastic neutron scattering processes that are important for our analysis of the phonons.

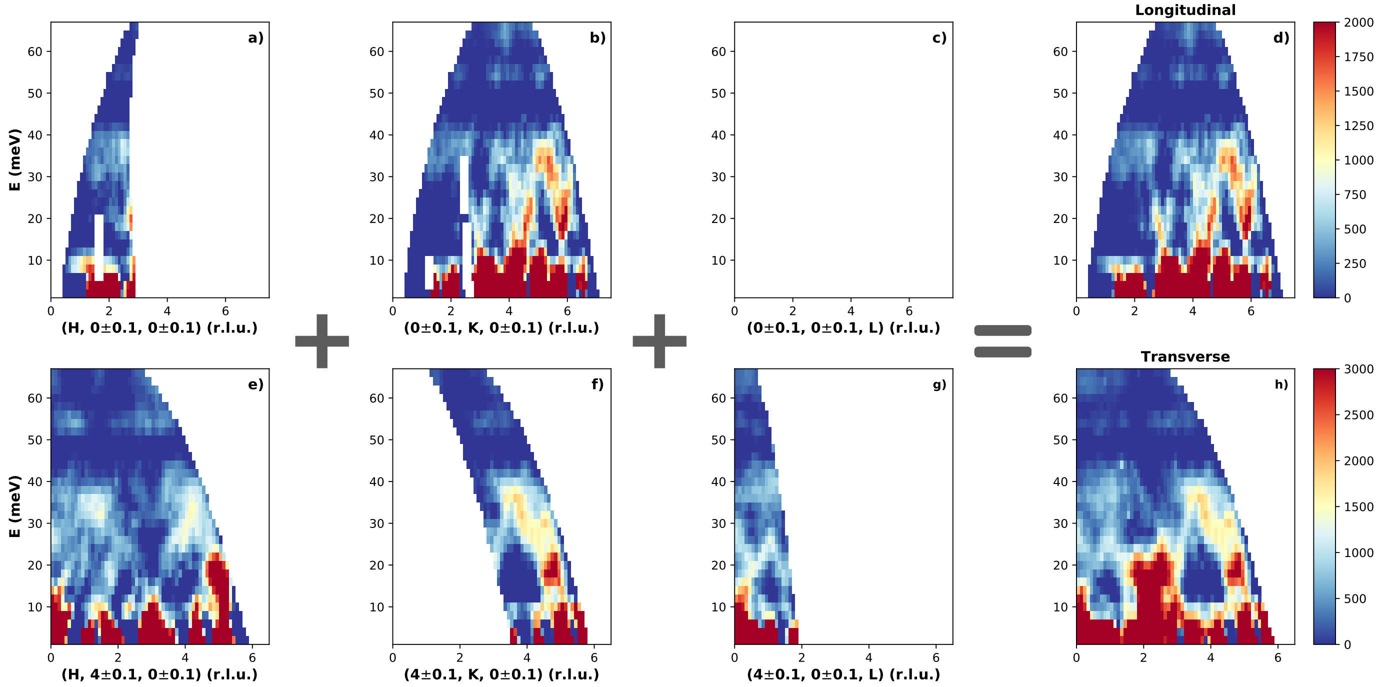


FIGURE 5.7: Improving statistics of data by combining symmetric directions and mirroring about zero momentum transfer. Left subplots: transverse phonon scattering along $(4, K, 0)$, $(H, 4, 0)$ and $(4, 0, L)$ and respectively. The first three columns of subplots, (a)–(c) and (e)–(g), are the inelastic neutron scattering data for three different $\mathbf{Q}$ planes, namely from left to right: $(H, 0, 0)$, $(4, K, 0)$ and $(4, 0, L)$. The columns on the far right represent the resultant scattering after adding together the scattering in the three equivalent datasets- this is possible because of the cubic symmetry of the PMN crystals.

PMN, which leads to the conclusion that the disorder inherent in the perovskite structure does not produce any noticeable differences in the lattice dynamical behaviour of the crystal for these energy scales. Unfortunately, owing to the closeness of the dispersions of these optic phonon branches and the inherent difficulties of analysing inelastic neutron scattering data, it is an extremely challenging task to be able to isolate the phonon scattering for individual phonon modes and determine their energy-momentum dispersions. Nonetheless, all subplots of Fig. 5.8 show three peaks in the DOS within the range of energies that were expected to contain, according to the density function theory perturbation calculations, three optic phonon branches. Hence, it is possible to conclude that the inelastic neutron scattering data agrees with the density functional theory calculations to the extent that the phonon branches do exist and correspond closely with the energy scales observed from the output of the Quantum Espresso calculations. However, it is very difficult to determine from the neutron scattering data which of the atoms in the unit cell of PMN contribute to the lattice dynamics. It can only be assumed that the close agreement of the neutron scattering with the theoretical calculations lends weight to the hypothesis that the dynamics of these modes are dominated by motion of oxygen atoms: analysis of the atom displacements of the phonon modes from the Quantum Espresso calculations show that oxygen atoms contribute to most of the dynamics of the transverse optic mode, but the two longitudinal modes are driven by the motion of both oxygen atoms and the cations on the B-sublattice.

Fig. 5.9 shows the inelastic neutron scattering data along the [010] direction starting from the Γ-point of the (400) Brillouin zone. The data taken using the E_i=75meV incident neutrons depicts the same two peaks in the pseudo-DOS plots: these again point to the presence of at least two non-degenerate optic phonon branches at this energy scale. These energy scales are slightly different from those found in the aforementioned data as the two peaks in this pseudo-DOS occur at $E \approx 53$meV and $E \approx 63$meV. Owing to the same difficulties in analysing this low-resolution inelastic neutron scattering data it was impossible to determine the full lattice dynamical response of these phonon modes.

The subplot in Fig. 5.9 depicting the inelastic neutron scattering data for the E_i=150meV incident neutrons shows four peaks between 58meV and 76meV; however, owing to the statistics of the data, it is difficult to determine rigorously whether this is because there are more phonon branches than were expected from the calculations or whether this is just a statistical phenomenon. The phonon excitations corresponded to islands centred at integer **Q**-points, which substantiates the original hypothesis that the lattice dynamical response of the PMN single crystals is exactly what would be expected from more ordered perovskites, such as $KTaO_3$, $BaTiO_3$ and $SrTiO_3$.

Fig. 5.10 subplots (a) and (b) show the dispersions of these islands of phonon excitations, which were obtained by taking a constant energy cut centred on the peak scattering intensities in both Brillouin zone configurations and finding the **Q**-point peaks and their associated FWHM's along the [100] and isotropic directions in reciprocal space. The peaks in scattering intensity were found to be slightly off the Brillouin zone edges which could be related to the crystal's

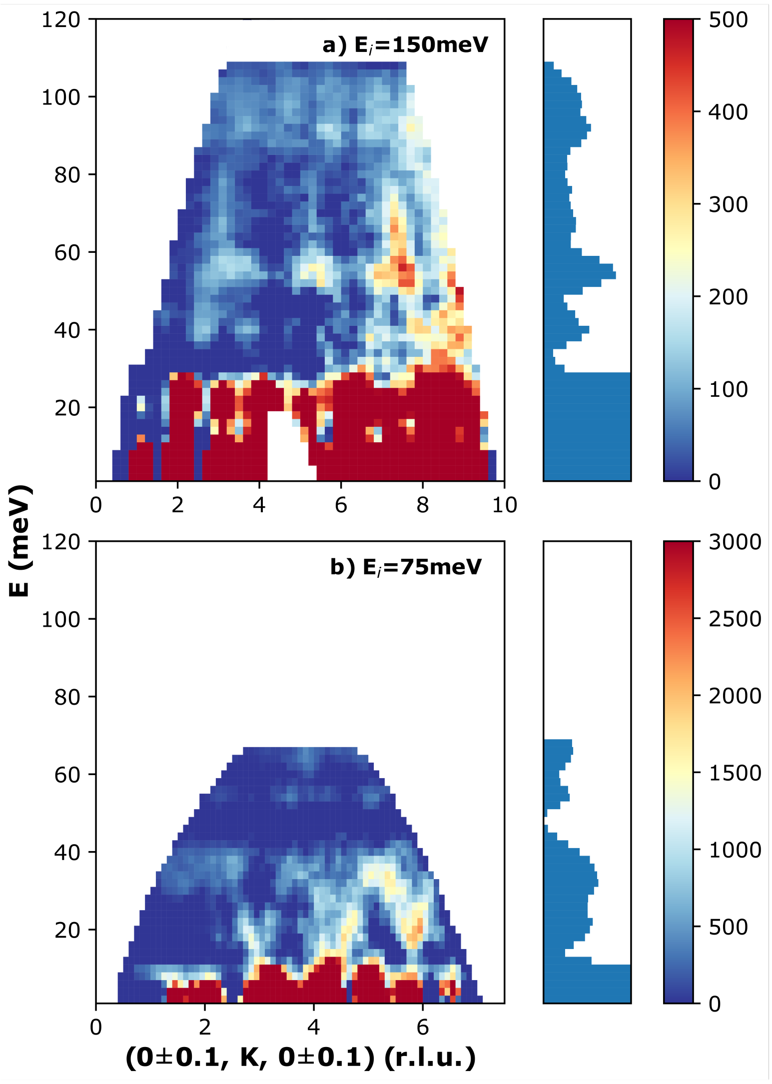


FIGURE 5.8: The inelastic neutron scattering plots along the [010] in the (000) Brillouin zone (left) for the E_i=150meV (a) and E_i=75meV (b) incident neutrons. On the right side is plotted the integrated intensities along lines of constant energies: this is representative of the density of states of the phonon branches in PMN crystal. It shows a lattice dynamical response with phonon branches between 40–60meV and a broad phonon mode higher in energy at 90–100meV. The side panels plot the normalised integrated intensities, $\frac{\int S(\mathbf{Q},\omega)d\mathbf{Q}}{\int d\mathbf{Q}}$, which help to locate the positions in energy of the phonon modes.

structure or some non-ideal experimental phenomenon: one hypothesised explanation for these islands of scattering intensity was that the high absorption cross section for the niobium atoms was contributing to reduced transmission through the crystal and causing the dips in $S(\mathbf{Q},\omega)$ due to the anisotropic morphology of the sample; however, it was determined that the expected drop in intensity would be too small to account for the structure seen in Fig. 5.10. It is impossible to determine the phonon lifetimes of these optic modes since their phonon scattering intensities cannot be resolved; nonetheless, it is possible to use the energy widths of these islands to calculate a lower limit for the phonon lifetimes of the separate optic phonon modes. Using the quantum uncertainty principle, $\Delta E \Delta t \sim \frac{\hbar}{2}$, with an energy width of $\Delta E \sim$ 10meV a lower limit to the phonon lifetimes is $\tau > 0.1$ps.

The experiment using an incident wavevector 8.51Å^{-1} (E$_i$ =150meV) was used for the purpose of studying the dynamical behaviour of the highest energy phonon branch (or branches as predicted by previous models). The density functional theory calculations predicted that there would be a single longitudinal optic phonon branch located at 93meV. In contrast, the models developed by B. Dorner predicted that there would be two non-degenerate modes at this energy scale, one of which is longitudinal and the other transverse in character. Inelastic neutron scattering studies on the lattice dynamical response of PMN at the higher energy scales were also undertaken to test this model[65]: using neutron spectroscopy data a continuum of phonon excitations was found above 20THz (E∼100meV). Unfortunately, owing to the weak signal for the higher energy excitations it was impossible to experimentally verify the model for the lattice dynamical response that predicted one LO and one TO mode at this energy scale. Moreover, this study was unable to determine whether the continuum of phonon excitations found above E∼100meV was due to the disordering of the perovskite structure of PMN or simply due to the poor statistics inherent in a lot of neutron spectroscopy experiments.

In this experiment, phonon excitations were observed at 93meV along the [100] directions at various regions of reciprocal space starting from the Γ-points of the (000) and (500) Brillouin zones. The translational invariance of the Γ-point energy and the fact that the crystal obeys Bloch's theorem was a strong indication that the lattice dynamical response of PMN was similar to the other ordered perovskites, such as $KTaO_3$, $BaTiO_3$ and $SrTiO_3$, for energies above E∼40meV. The statistics of the data sufficed for characterisation of the phonon dispersion and a fit was made to a function obeying Bloch's theorem for translationally invariant crystals on the assumption, as aforementioned, that at the higher energies the lattice dynamical response behaved like an ordered perovskite crystal. The following methodology was used to obtain the fit: constant Q scans were taken from the data collected using the E$_i$=150meV neutrons, and a Gaussian fit was refined for the peak in the scattering intensity that corresponded with the highest energy optic phonon mode. These peaks in Q were subsequently fitted to the following function to characterise the dispersion of the optic mode:

$$E = \big(A + B\sin\big(\frac{\pi Q}{2}\big)\big)^{\frac{1}{2}}. \tag{5.14}$$

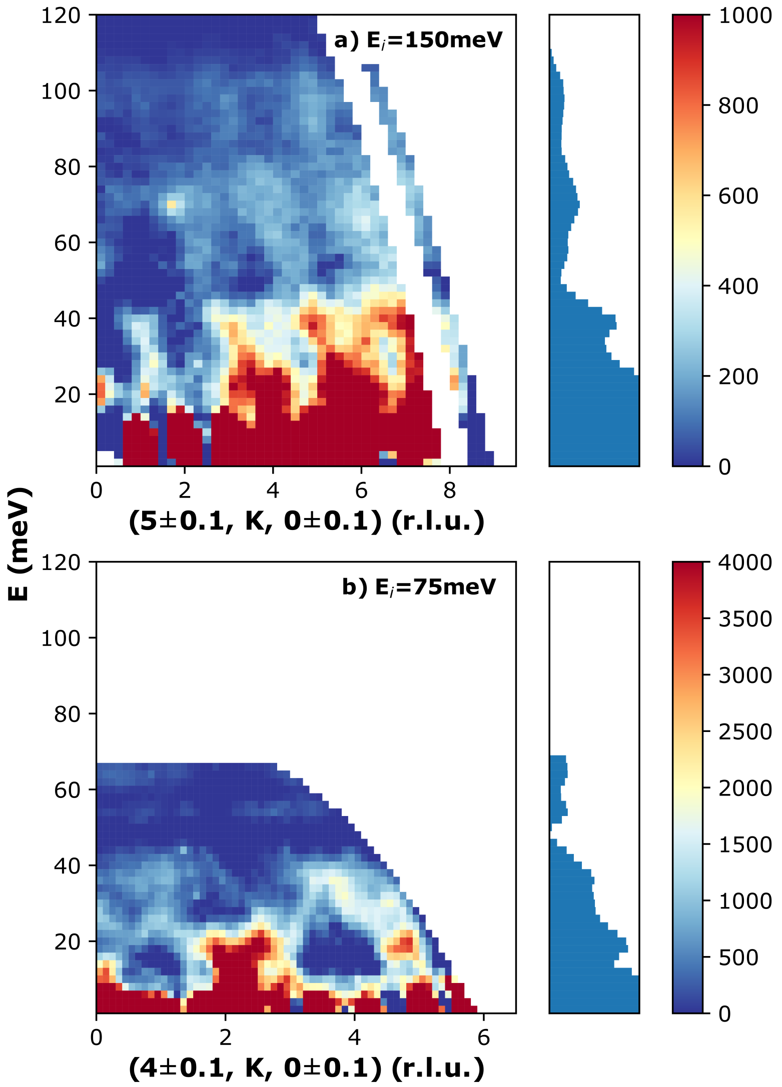


FIGURE 5.9: A plot of $S(\mathbf{Q},\omega)$ along the [010] direction in the (400) and (500) Brillouin zones taking advantage of the $\mathbf{Q}^2$ scaling for phonon inelastic neutron cross sections. E=75meV neutrons are used for the (400) Brillouin zone and E=150meV neutrons are used for the (500) Brillouin zones. There is the same lattice dynamical response at the higher energies as there is in the (000) Brillouin zone. Furthermore, the agreement between the experimental data and the density function theory calculations for $SrTiO_3$ reveals that the lattice dynamical response at high energies closely resembles the chemically ordered classical perovskites. The side panels plot the normalised integrated intensities, $\frac{\int S(\mathbf{Q},\omega)d\mathbf{Q}}{\int d\mathbf{Q}}$.

This fit was refined separately for the higher energy (E∼100meV) optic phonon, as shown in Fig. 5.8 and Fig. 5.9, in order to characterise the dispersions at different Brillouin zones. In the (000) Brillouin zone, the refined parameters for the dispersion curve were 9270meV^2 and -260meV^2 respectively, whereas in the (500) Brillouin zone the same parameters were refined differently as 9240meV^2 and 817meV^2 respectively. Using the values for these refined parameters to extrapolate the Γ-point energy of the optical phonon branch gives 96.3meV for the (000) Brillouin zone and 96.1meV for the (500) Brillouin zone. These energies show strong agreement with the position of the highest energy excitation peak in Raman spectroscopy data [101] (see Fig. 5.11). The translational invariance of the Γ-point energy implies that the dynamics of the phonons at higher energies follows that of a perfect crystal rather than the expected disordered behaviour associated with perovskites. This is a very interesting result since it splits the lattice dynamical behaviour into two ill-defined regimes: one at lower energies where the crystal behaves as an imperfect crystal and one at higher energies at which the behaviour becomes closer to that of a perfect crystal. This conclusion does not simply rest on the aforementioned behaviour since it also is due to the close resemblance between the calculated dispersions for the three phonon branches between 48–60meV in SrTiO3 (a perfect crystal with the space group Pm3m) and the observed peaks in the DOS plots for both the (000) and (500) Brillouin zones for PMN. Another similarity between the density functional theory calculations for $SrTiO_3$ and the experiments can be seen from Fig. 5.10: in the calculations, the highest energy phonon branch was longitudinal with the lattice dynamics dominated by the movement of the oxygen atom at position (0.5, 0, 0.5) along the [010] direction. The dependence of the coherent inelastic neutron scattering cross section on $(\mathbf{Q}\cdot\mathbf{e})^2$, where $\mathbf{e}$ is the eigen-displacement vector for the constituent atoms taking part in the lattice vibrations, means that close to Q=0 within the (000) Brillouin zone (this is where Q and q are equal to one another in neutron scattering events) any structure observed along the [100] direction in the first Brillouin zone would evidence the longitudinal character of the mode. This is because for a transverse mode in the first Brillouin zone, i.e., $\tau = 0$, $\mathbf{e}$ is perpendicular to the scattering vector, $\mathbf{Q}$, and so incident neutrons would not scatter off these modes. As can be seen from the inelastic neutron scattering data in Fig. 5.10, there is clear structure to the constant energy scan that is indicative of coherent scattering of a longitudinal phonon mode.

The analytical objective for Fig. 5.11 was to understand whether the inelastic neutron scattering data were in agreement with the understanding of the lattice dynamical response from the DFT calculations: that the motion of the crystal is dominated by oxygen atoms and to a lesser extend the atoms occupying the disordered B-lattice site. If this is true, it would mean that the disordering of the B-site does not affect the principal driving features of the lattice dynamical behaviour and would be a fascinating scientific result given the degree to which PMN is a disordered crystalline structure in so many ways. To estimate the degree to which the oxygen, magnesium and niobium atoms contribute to the lattice dynamics for the highest energy mode, an estimate of the ratio of the structure factors at $\mathbf{Q} = (3, 0, 0)$ and $(4, 0, 0)$ was calculated for the neutron data

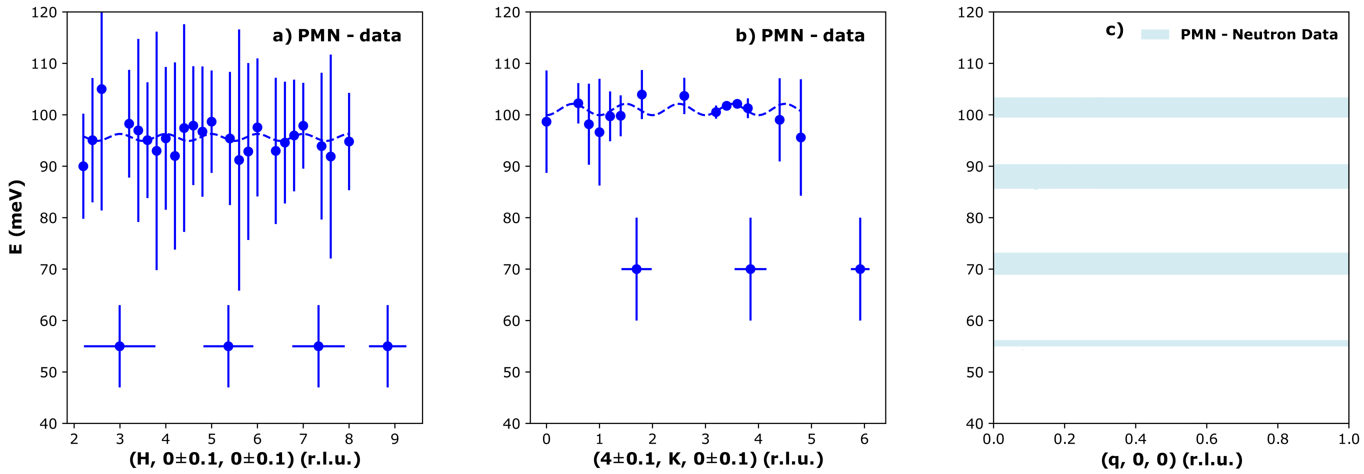


FIGURE 5.10: (a) Phonon dispersion of the highest energy mode (E∼100meV) starting from the Γ-point in the (000) Brillouin zone and the islands of coherent phonon scattering lower in energy. (b) The same highest energy mode characterised within the (400) Brillouin zone and the islands of scattering corresponding to the predicted LO and TO phonon modes. (c) The light blue shaded zones are the continuous neutron scattering excitations found in previous research by Dorner et al. [65].

and using the eigen-displacements from the DFT calculations. For the latter, the normalised eigen-displacements at a reduced wavevector of $\mathbf{q} = 0$ was used since we are at the Brillouin zone centres, and the ratio of the structure factors was calculated assuming that we can approximately neglect the contributions to the scattering from the lead atoms. A simple expression was developed for the ratio of structure factors in $SrTiO_3$, $|F_{H00}|^2$, that included a single parameter, S, that described the fraction of the normalised eigen-displacement of the B-site atoms compared with the oxygen atoms:

$$|F_{H00}|^2 = |H \times \frac{b_O}{\sqrt{M_O}} - H \times \frac{S \times b_{MN \times \exp{(i\pi H)}}}{\sqrt{M_{MN}}}|^2, \tag{5.15}$$

$$S = -\frac{\delta_{MN}}{\delta_O}, \tag{5.16}$$

where b_O and b_{MN} are the inelastic coherent scattering lengths for the oxygen atoms and titanium atoms respectively (the subscript "MN" indicates that both magnesium and niobium atoms are on the B-site in PMN rather than titanium atoms as in $SrTiO_3$). From the DFT calculations, it is found that $S \approx 0.23$ in $SrTiO_3$ (represented by the black vertical line in Fig. 5.11), which is predicted to produce a ratio of the two Brillouin zones, according to the expression above, of slightly above 0.8. Assuming that the lattice dynamics for $SrTiO_3$ well describes the motion of the atoms for the highest energy mode in PMN, the ratio of structure factors should be similar in the latter. In fact, this is exactly what is found from the inelastic neutron scattering data. Using the integrated intensities as a proxy for the structure factors and integrating over E=[90, 110]meV, the data shows that there is a similar ratio of structure factors between the (300) and (400) Brillouin zones in PMN (as represented by the horizontal line with 1σ uncertainty bars) as in the more ordered perovskite crystals, which was an unexpected result given the multiple degrees of disorder in the PMN crystal. This is further emphasised when extending the same analysis to other Brillouin zone centres further away from the centre of reciprocal space. In Fig. 5.11 subplot (b), the predicted structure factors using $S = 0.23$ along with the integrated intensities from the PMN neutron scattering experiments are plotted together. Given the difficulties inherent in using the scattering function to determine the atomic positions and, to an even greater extent, the atomic displacements (the phase problem in crystallography), it is rather reassuring that the general variability of the scattering data is encapsulated by the simple lattice dynamical model in Eq. 5.15 (the analysis gives a fitting parameter of $\chi^2 = 3.25$). With this degree of fit it is not possible to make a conclusion about the precise motion of this phonon mode with certainty, but it is apt to conclude that the higher energy phonon modes have a high degree of structure indicative of a more perfect non-disordered crystalline material.

Lastly, we note that in both the (000) and (500) Brillouin zone configurations, the coherent phonon scattering for this optic phonon branch is considerably broad in comparison to most of the peaks in the pseudo-DOS for the lower energy modes. With a FWHM on the order of ΔE ~20meV, these phonons decay more quickly than those in branches with lower energies and have a lifetime on the order of $\tau \sim 0.01$ps. This is due to more decay channels being open for the higher energy branches, which can be understood as that these phonon modes have more branches into which they can decay or scatter.

5.5 Concluding the study of PMN

Neutron spectroscopy has been applied to PMN to study the lattice dynamical response of the crystal at low temperatures and at energies above E$\sim$40meV. An algorithm for the subtraction of the incoherent phonon scattering from the overall scattering function, $S(\mathbf{Q},\omega)$, has been detailed and applied to inelastic neutron scattering data collected on MERLIN at ISIS. The lattice dynamical response observed at these higher energies showed well-defined modes similar to those predicted to exist in another classically ordered perovskite, $SrTiO_3$. This study therefore appears to invalidate the hypothesis made in previous research on this crystal [65] that says that the band of scattering at higher energies is due to the chemical disordering of the B-sublattice in PMN. Rather, it is highly likely that the neutron spectroscopy data in that study had too weak a signal to make such a conclusion and the resolution of the data was affected by the lower resolving power of previous neutron spectroscopy experiments.

Advances at neutron facilities has therefore made it possible to study the lattice dynamical response to much greater precision and allow us to conclude that in some respects PMN does, unexpectedly, behave like a classically ordered perovskite. By measuring the scattering intensity at multiple reciprocal lattice points—thus taking advantage of the $\vec{\mathbf{Q}}^2$-dependence of the neutron scattering of phonons—and comparing these with expected values from a simplistic optic phonon model, it is also possible to infer from the data that the dynamics are mostly dominated by the oxygen atoms and, to a lesser extent, those occupying the B-site of the perovskite unit cell. This result is certainly not intuitive given that the disordered structure of PMN, which breaks the perfect translational symmetry of the crystal, would be expected to lead to a concomitant disordering of the normal modes of vibration of the crystal, including the highest energy phonon. This is especially true given the large differences in the valences and orbital structure of the Mg and Nb atoms that occupy the B-site of PMN. Nonetheless, the inelastic neutron scattering data presented does suggest the lattice dynamics are unaffected by the valence disorder. A logical explanation is that the mode is mostly driven by the motion of oxygen atoms meaning that the phonon mode can propagate through the lattice without being affected by the aforementioned disorder.

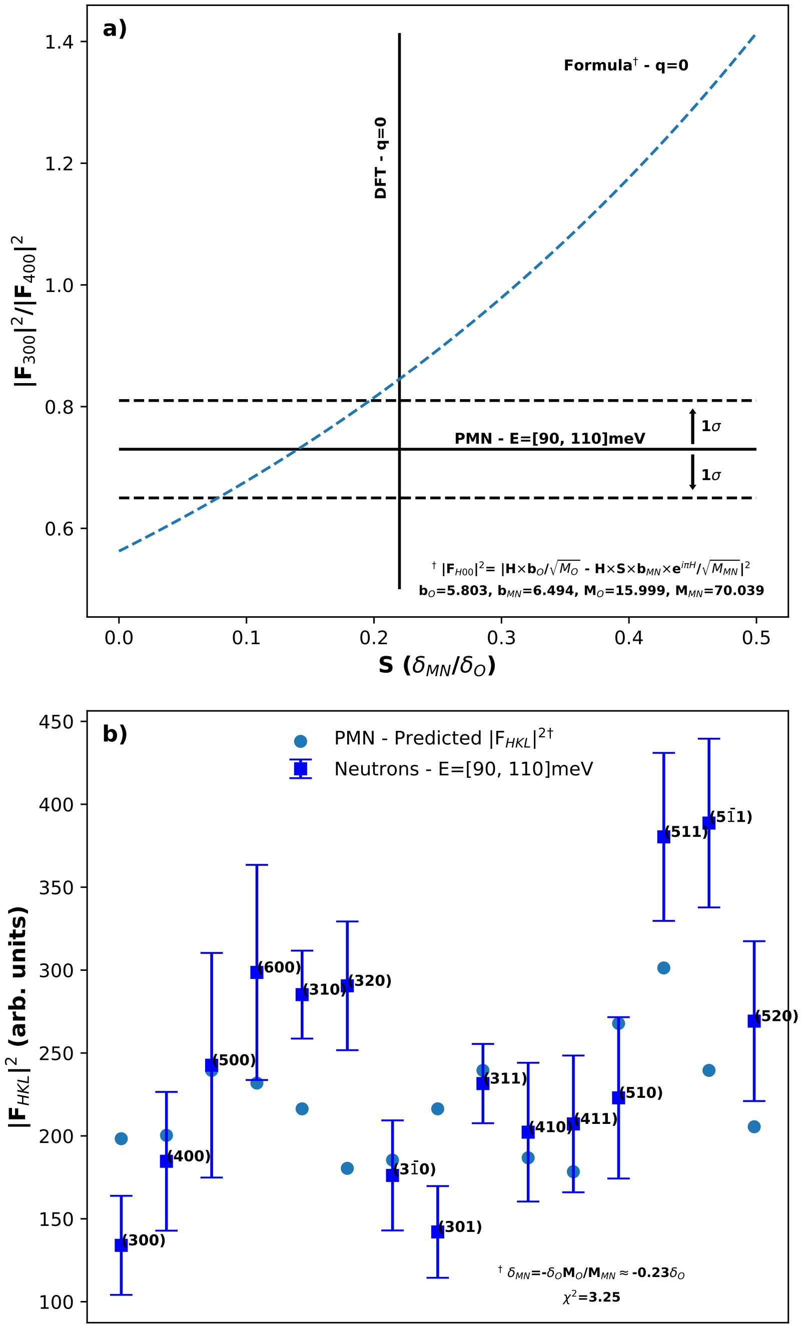


Figure 5.11: (a) Constant energy scan integrated over E=[90meV, 110meV] of the inelastic neutron scattering data for PMN. (b) Theoretical structure factor of PMN assuming that it has the same ordered perovskite structure of others in the same class, i.e., $KTaO_3$, $BaTiO_3$ and $SrTiO_3$. Similarities between the constant energy scan from the neutron scattering data and the theoretical structure factor for PMN show that the lattice dynamical response is like the other ordered perovskites with the chemical formula ABO_3.

Chapter 6

Thesis summary

This thesis outlines in detail the research projects that were undertaken over the course of 2018–2022 at the University of Edinburgh. The objective of the research projects was to improve our understanding of the lattice dynamical behaviour of perovskites and iron chalcogenides and to provide a fuller picture of the origin of magnetic interactions within a system with complicated magnetic and orbital physics. An invaluable technique that was used to study these compounds, which included $PbMg_{1/3}Nb_{2/3}O_3$ and $Fe_{1+y}Te$, was inelastic neutron scattering. This uses neutrons produced by a spallation reaction or from a nuclear reactor to probe the structural and dynamical properties of solid state substances—the technique is also regularly utilised in various other fields of scientific research, including in soft condensed matter and in biophysics to name only a few. Each chapter of this thesis has given an overview of the scientific literature that has been published about these compounds that pertains to studying the same types of phenomena. These include papers focusing on the behaviour of phonon modes in both crystals and the role that structural distortions are playing in the emergence of the double stripe bicollinear antiferromagnetic order for the low-y configuration of $Fe_{1+y}Te$. For the inelastic neutron scattering studies, they further explain the experiments that have been carried out, which includes the preparation of any single crystal samples, the methodology for analysing the neutron scattering data and how the form of scattering can reveal information about the structure and dynamics of the periodic crystals. The neutron scattering comprises both coherent and incoherent scattering with the former probing correlations between the relative positions of atoms within the crystal, and therefore providing information about the structure of the compound and the collective motion of atoms in normal modes of vibration, and the latter supplying important information about the density of phonon states. Although the final project involved developing a theoretical framework using a Rayleigh-Schrödinger perturbative approach for studying the interplay between spin and orbital degrees of freedom within FeTe crystals, the objective was still to be able to re-construct the spin waves in order to compare their structure with those published in previous neutron scattering studies. Hence, the overriding emphasis of the thesis is on the use of neutron scattering techniques to probe various physical phenomena within hard condensed matter. For the benefit of the reader, the following sections of this concluding chapter will briefly summarise the individual projects that were carried out for this thesis for the award of Doctor of Philosophy.

6.1 Summaries of research projects

6.1.1 Order and disorder within $PbMg_{1/3}Nb_{2/3}O_3$

The objective of this research project was to study the lattice dynamical behaviour of a prototypical lead-based relaxor ferroelectric, $PbMg_{1/3}Nb_{2/3}O_3$, in order to understand whether the structural disorder on the B site combined with the differing valence has an effect on the high energy phonons. Relaxor ferroelectrics differ from normal ferroelectrics in that they also have exceptional electromechanical properties, including large electrostriction and high dielectric constants. This makes them a focus of research due to the fact that these materials have properties that are favourable for using in electromechanically controlled devices.

$PbMg_{1/3}Nb_{2/3}O_3$ has two critical transition temperatures rather than one ferroelectric transition that results in the temperature and frequency broadening of the peak in the dielectric response of the crystal. There is a high temperature T_d, the Burns temperature, that is associated with the onset of short-range polar nano-regions and a lower temperature T_c which defines the emergence of the ferroelectric phase that can be induced by an electric field—this is observed in diffuse scattering as an anisotropy in the cross section. Previous research has suggested that the energy broadening and overdampening of the lower energy transverse optic phonon mode over this region from T_c to T_d leading to its precipitous collapse at long wavelengths (q$\sim$0) is due to these polar nano-regions [53]. Hence, it was already well appreciated that the disorder within this perovskite has an effect on the lattice dynamical response of the crystal at lower energies, i.e., E<40meV. However, using the inelastic neutron scattering data collected on the MERLIN triple-axis spectrometer at ISIS it was hoped that more could be revealed about the dynamics at higher energies. In order to have a counterfactual of the lattice dynamical response of an ordered equivalent perovskite, the inelastic neutron scattering data was compared with theoretical calculations of the lattice dynamics of the higher energy longitudinal and transverse optic modes in strontium titanate, $SrTiO_3$. The dynamic response of this ordered perovskite crystal along the $\Gamma - X$ path in reciprocal space was determined from density functional perturbation theory calculations using the Quantum Espresso software package [93].

To investigate the lattice dynamics of the crystal along the Γ–X high symmetry direction, an analysis of inelastic neutron scattering data using a composite $\sim$192g single crystal sample was carried out for the (H00), (0K0), (00L), (H40), (4K0) and (40L) directions with initial neutron energies of E=75meV and E=150meV. The statistics were improved by removing background and incoherent inelastic neutron scattering and combining the scattering in symmetric directions owing to the cubic symmetry of the $PbMg_{1/3}Nb_{2/3}O_3$ crystals. The dynamic response at high energies revealed by this analysis demonstrated that the high energy transverse and longitudinal optic modes were well-defined and closely resembled the phonon dispersions of the ordered perovskite $SrTiO_3$. Moreover, comparisons of the integrated intensities of $PbMg_{1/3}Nb_{2/3}O_3$ at reciprocal lattice vectors with theoretical calculations of the structure factor using a simplistic model of the atomic displacements revealed a close similarity with the experimental data implying that, notwithstanding the known disorder of the crystal at lower energies, the lattice dynamical response was mostly typical of an ordered perovskite rather than being influenced by the polar nano-regions that exist between the two critical temperatures of this relaxor ferroelectric. At the same time, however, the appearance of the transverse optic phonon mode in both the longitudinal and tranverse channels indicates that there is some breakdown of the cubic symmetry of the crystal owing to an anisotropy in the dynamical response that could be due to these polar nano-regions. Nonetheless, it is clear that there are well-defined normal modes of vibration at higher energies rather than the previously observed continuous bands of scattering which were assumed to be due to the chemical disordering on the B site of the perovskite lattice [65].

6.1.2 Softening of transverse acoustic phonons in $Fe_{1+y}Te$

This was another research project that used inelastic neutron scattering techniques with the aim of probing the lattice dynamical behaviour of a solid-state compound, which in this case was an iron-based chalcogenide, $Fe_{1+y}Te$. It shares structural and magnetic properties with other iron-based systems, including FeSe, underdoped $Ba(Fe_{0.94}Co_{0.03})_2As_2$ and optimally doped $Ba(Fe_{0.94}Co_{0.06})_2As_2$, and has a complicated structural, magnetic and electronic phase space that is dependent on the concentration of interstitial iron, y, in the compound. The phase space also reveals coincidental phase changes that could point to a coupling between the lattice, spins and the electronic order. In previous studies of FeSe, underdoped $Ba(Fe_{0.94}Co_{0.03})_2As_2$ and optimally doped $Ba(Fe_{0.94}Co_{0.06})_2As_2$, the anomalous softening of the acoustic phonons close to the Brillouin zone centre ($q<0.1$) suggested that the nematic electronic order is responsible for changes to the energy of the modes at long wavelengths and that the degree of softening depends on the static nematic susceptibility, the nematic correlation length and a nematic-elastic coupling constant.

Neutron scattering data was collected on the EIGER instrument using a single crystal of $Fe_{1+y}Te$ with an interstitial iron concentration below the critical concentration, $y \sim 0.12$, at a plurality of temperatures that included both the lower temperature monoclinic phase and the higher temperature tetragonal phase. The objective of the inelastic neutron scattering experiments was to characterise the softening of the transverse acoustic phonons and use constant momentum scans at the Brillouin zone centre, $\mathbf{Q}=(2,\ 0.3,\ 0)$, and Brillouin zone edge, $\mathbf{Q}=(2,\ 1,\ 0)$, in order to test the hypothesis that, as is the case with the other iron-based systems, nematic order is responsible for the softening of the phonons in this iron-based chalcogenide. Analysis of the neutron scattering data revealed that, unlike the acoustic phonons in the other iron-based system, the softening of the transverse acoustic phonons in $Fe_{1+y}Te$ were being driven by the zone-edge phonons and that close to the zone centre there was very little softening that could be observed. The softening at the Brillouin zone edge appears to be due to an increasing number of scattering channels opening up as the structural phase transition temperature is approached as evidenced by the temperature-dependent broadening of these phonons. It is difficult to conclude whether the softening of the transverse acoustic phonons at the Brillouin zone edge is due to the presence of interstitial Fe atoms or the onset of magnetic order close to the structural transition temperature. Moreover, it is not possible to rule out the involvement of the electronic order since the tetragonal–monoclinic phase transition that occurs in $Fe_{1+y}Te$ may produce a different type of phonon softening compared with that observed during the tetragonal–orthorhombic phase transition in the other iron-based systems.

6.1.3 Interplay between spin and orbital physics in FeTe and the emergence of exotic magnetic order

$Fe_{1+y}Te$ has the unusual double stripe antiferromagnetic order at low temperatures and low interstitial iron concentrations as seen in Fig. 1.12 of Sec. 1.4.2, which depicts the phase diagram of the crystal in the $y - T$ plane. Previous studies have tried to describe the magnetic interactions that lead to this magnetic order by developing theoretical frameworks that include either the spin or orbital degrees of freedom (the latter comes from the valence occupancies of both the d_{Xz} and d_{Yz} orbitals on the Fe atoms). However, so far these have been unable to give a completely satisfactory description of the magnetic interactions and a spin-only Heisenberg model of the spin waves in the low-y configuration of $Fe_{1+y}Te$ did not produce an optimal fit of the inelastic magnetic neutron scattering data [72, 73]. At low energies in the neutron scattering, the data revealed a magnetic spectrum that had C_2 symmetry but as the energy is increased beyond E=26meV a spectrum with C_4 symmetry emerges. As this energy scale is close to those associated with the Jahn-Teller effect in other iron-based systems including magnetite, $Fe^{2+}Fe_2^{3+}O_4$, it was suggested that the magnetic interactions could be explained by a model that included both the spin and orbital physics and the interplay between them.

The objective of this research project was to develop a Rayleigh-Schrödinger perturbative model of the FeTe system that included both spin and orbital degrees of freedom in order to explain the magnetic interactions within the crystal and the bicollinear antiferromagnetic order observed at low temperatures and low interstitial iron concentrations. A starting point of the model was to consider the structure in the tetragonal phase and introduce a lattice distortion owing to the crystal field splitting from the tetrahedra of Te ions surrounding each Fe ion and that split the 3d orbitals into a lower energy t_{2g} subgroup and a higher energy e_g subgroup. This was further split by a Jahn-Teller distortion of the lattice such that the Fe ions moved towards one of the Te ions along an axis colinear with this ion and another Te ion on the opposite side of the parallelogram of Te ions. The model space was then constructed by pairs of Fe ions in the S=1 spin state and neighbouring Te ions with fully occupied 5p orbitals. The perturbative model included virtual electron hopping up to fourth order hopping with states in an orthogonal space. An 36×36 effective Hamiltonian matrix was constructed out of the 6 S=1 spin states on each Fe ion giving a basis set of 36 states. This effective Hamiltonian matrix was decomposed into an orthogonal basis set formed by the sum of products of spin (S=1) and orbital pseudo-spin (S=1/2) operators multiplied by their weights. Some of these weights were zero by construction due to the spin-independence of the tight-binding hopping integrals, whilst others were zero due to the restrictions imposed on the virtual electron hopping. It was found that the entire effective Hamiltonian could be described by 16 weights that included terms for spin-only processes, orbital pseudo-spin only processes and processes that involved the interplay between the spin and orbital physics.

6.2 Prospects for continuing research

This section will explain some of the limitations of the research carried out for this thesis and what could reasonably be achieved if there was more time to complete them. This is an important part of assessing the state of current research and would help someone to continue to investigate both $PbMg_{1/3}Nb_{2/3}O_3$ and $Fe_{1+y}Te$ in order to better understand their structural, magnetic or electronic properties. The limitations can be categorised into those associated with experimental constraints and theoretical barriers to further investigations. For instance, in the case of the study investigating the high energy longitudinal and transverse optic phonon modes in $PbMg_{1/3}Nb_{2/3}O_3$, the project was limited by the low nuclear inelastic scattering cross section at higher energies (the cross section is proportional to $1/\omega_s$), which meant that it was a challenge to use the neutron scattering data to accurately compare it with models of the structure factor. One way around this physical limitation imposed by the neutron flux of the instrument is that a more complex model be constructed of the atomic displacements. If the model contains a large number of degrees of freedom, and therefore with a large multi-dimensional parameter space, an optimisation algorithm could have been used to find the best fit with the neutron scattering data. This would have given a much richer understanding of the lattice dynamical behaviour of $PbMg_{1/3}Nb_{2/3}O_3$ at higher energies, i.e., E$\sim$100meV.

One of the principle limitations to the study of the anomalous softening of the transverse acoustic phonons in $Fe_{1+y}Te$ is that the model used for the nematic driven softening was from a previous study that observed a tetragonal–orthorhombic structural phase transition rather than the tetragonal–monoclinic phase transition that occurs in $Fe_{1+y}Te$ single crystals with an interstitial iron concentration that is below the critical value of $y \sim 0.12$. As described in Sec. 4, the softening in $Fe_{1+y}Te$ was being driven by phonons at the Brillouin zone edge rather than those at the zone centre, but it is challenging to work out, with only the current model of the phonon softening, whether this is due to other factors that might influence the number of scattering channels available, such as the interstitial Fe atoms or the effect of the emergence of the double stripe bicollinear antiferromagnetic order, or also being caused by the nematic electronic order. Hence, only by having this theoretical model of the nematic softening of the transverse acoustic phonons would it be possible to finally answer the question the research project posed: what is causing the anomalous phonon softening in $Fe_{1+y}Te$?

Finally, if more time was available for continuing the development of the theoretical framework of the magnetic interactions in $Fe_{1+y}Te$, it would have been instructive to optimise the weights in the effective Hamiltonian that described the entire system (i.e., including all the bonds within the translationally invariant lattice) so as to reproduce the double stripe bicollinear antiferromagnetic ground state. This could be achieved by initialising an $N \times N$ cell of the single crystal (ignoring the interlayer magnetic interactions) and writing a program to find the weights that lead to the experimentally observed magnetic ground state by brute force. Using the ground state, it would then be possible to use the Holstein-Primakoff and Jordan-Wigner transformations of the effective Hamiltonian in order to characterise the spin wave dispersion. By comparing these spin waves with the inelastic neutron scattering data [75], it would be possible to test the hypothesis that the interplay between both the spin and orbital physics has some role to play in the origin of the double stripe bicollinear antiferromagnetic order and the magnetic dynamics of this iron-based solid state compound.

Appendix A

Quantum Perturbation Series

This appendix is devoted to explaining the general approach of the Rayleigh-Schrödinger perturbation series, which is important because it is applied in Sec. 5 to form the effective Hamiltonian of the $\mathrm{Fe}_{1+y}\mathrm{Te}$ system. The difference between this perturbation series and that given by the Brilllouin-Wigner perturbation theory is that in the former the zeroth-order eigenvalues appear in the series expansion, while in the latter the perturbed eigenvalues appear on both sides of the equation meaning that the problem needs to be solved by an iterative method. In the following, the Rayleigh-Schrödinger perturbation theory is used to solve the time-independent Schrödinger equation with a model space that contains degenerate states and generates an effective Hamiltonian that only acts on this model space. For the more general case, which could include the model space being completely degenerate, it is recommended to consult Ingvar Lindgren's paper on the use of the Rayleigh-Schrödinger perturbative theory [89]. The following is a summary of what has been outlined in that paper and is included here because of its importance to this thesis.

A.1 Rayleigh-Schrödinger Perturbation Theory

The Rayleigh-Schrödinger perturbation theory is trying to solve the time-independent Schrödinger equation:

$$\mathbf{H}\mathbf{\Psi}^{\alpha} = \mathbf{E}^{\alpha}\mathbf{\Psi}^{\alpha}, \tag{A.1}$$

where $\mathbf{H}$ is the full Hamiltonian, $\mathbf{\Psi}^{\alpha}$ is the wavefunction and $\mathbf{E}^{\alpha}$ is the associated eigenvalue. The full Hamiltonian can be decomposed into the unperturbed Hamiltonian, $\mathbf{H}_0$, and the perturbation potential, $\mathbf{V}$:

$$\mathbf{H} = \mathbf{H}_0 + \mathbf{V}. \tag{A.2}$$

The unperturbed Hamiltonian has a complete set of orthogonal eigenstates, ϕ^a, and their eigenvalues, $\mathbf{E}_0^a$:

$$\mathbf{H}_0\mathbf{\Phi}^{a} = \mathbf{E}_0^{a}\mathbf{\Phi}^{a}. \tag{A.3}$$

The model space is defined as a subset of these eigenstates, D, and has the projection operator, $\mathbf{P}$:

$$\mathbf{P} = \sum_{a \in D} |\boldsymbol{\Phi}\rangle^{a} \langle \boldsymbol{\Phi}^{a}| \, . \tag{A.4}$$

It is also possible to define another operator, $\mathbf{Q}$, that projects onto states outside of the model space (often called the orthogonal space):

$$\mathbf{Q} = \sum_{a \notin D} |\boldsymbol{\Phi}^{a}\rangle \langle \boldsymbol{\Phi}^{a}| \, , \tag{A.5}$$

with the relation between the two operators being $\mathbf{Q} = 1 - \mathbf{P}$. These operators obey some important relations, which are useful for the derivation of the Rayleigh-Schrödinger effective Hamiltonian:

$$\mathbf{P} = \mathbf{P}^{\dagger} = \mathbf{P}\mathbf{P}^{\dagger}, \tag{A.6}$$

$$\mathbf{Q} = \mathbf{Q}^{\dagger} = \mathbf{Q}\mathbf{Q}^{\dagger}, \tag{A.7}$$

$$\mathbf{P}\mathbf{Q} = \mathbf{Q}\mathbf{P} = 0, \tag{A.8}$$

$$[\mathbf{P}, \mathbf{H}_0] = [\mathbf{Q}, \mathbf{H}_0] = 0. \tag{A.9}$$

If there are d eigenstates within the model space of the unperturbed Hamiltonian, then there is a corresponding d eigenstates of the full Hamiltonian that can be projected into the model space by the operator $\mathbf{P}$:

$$\boldsymbol{\Psi}_0^{\alpha} = \mathbf{P}\boldsymbol{\Psi}^{\alpha}. \tag{A.10}$$

This is due to the fact that as the perturbation is switched on the distinct eigenstates within the model space are expected to mix but the perturbed eigenstates are likewise going to be comprised of the same number of orthogonal and distinct functions. Another operator can be defined, $\boldsymbol{\Omega}$, that projects the eigenstates in the model space back out of the model space:

$$\boldsymbol{\Omega}\boldsymbol{\Psi}_0^{\alpha} = \boldsymbol{\Psi}^{\alpha}, \tag{A.11}$$

which for completeness can be assumed to obey the relation $\boldsymbol{\Omega}\mathbf{Q} = 0$. It can be demonstrated that the d eigenvalues of the full Hamiltonian, $\mathbf{E}^{a}$, can be determined by the expectation value of an effective Hamiltonian acting only on the model space by their associated eigenstates which are projected into the model space, $\boldsymbol{\Psi}_0^{\alpha}$:

$$\mathbf{E}^{a} = \langle \boldsymbol{\Psi}_0^{a} | \, \mathbf{H}_{\text{eff}} \, | \boldsymbol{\Psi}_0^{a} \rangle \, . \tag{A.12}$$

The benefit of this is that the unperturbed eigenstates within the model space can be used to determine their energies after the introduction of the small perturbation, which in the case of the $\mathrm{Fe}_{1+y}\mathrm{Te}$ involves the switching on of the virtual electron hopping processes that connects the model space with the orthogonal space. It can be demonstrated by applying $\boldsymbol{\Omega}$ to both sides of the Schrödinger equation that the effective Hamiltonian can be written as:

$$\mathbf{H}_{\text{eff}} = \mathbf{P}\mathbf{H}_0\mathbf{P} + \mathbf{P}\mathbf{V}\boldsymbol{\Omega}. \tag{A.13}$$

The Rayleigh-Schrödinger perturbation series then comes from the expansion of the $\mathbf{P}\mathbf{V}\boldsymbol{\Omega}$ term in the effective Hamiltonian. In order to expand this term it is necessary to write Ω as a function of increasing powers of the perturbation, $\mathbf{V}$. This can be achieved by applying the $\boldsymbol{\Omega} = \boldsymbol{\Omega}\mathbf{P}$ operator to both sides of the Schrödinger equation in eq. A.1:

$$\boldsymbol{\Omega}\mathbf{P}(\mathbf{H}_0 + \mathbf{V})\boldsymbol{\Psi}^{\alpha} = \mathbf{E}^{\alpha}\boldsymbol{\Omega}\mathbf{P}\boldsymbol{\Psi}^{\alpha}, \tag{A.14}$$

which can be rearranged to produce the following expression:

$$(\boldsymbol{\Omega}\mathbf{H}_0 - \mathbf{H}_0\boldsymbol{\Omega})\boldsymbol{\Psi}_0^{\alpha} = (\mathbf{V}\boldsymbol{\Omega} - \boldsymbol{\Omega}\mathbf{V}\boldsymbol{\Omega})\boldsymbol{\Psi}_0^{\alpha}, \tag{A.15}$$

and hence:

$$(\mathbf{E}_0^{\alpha} - \mathbf{H}_0)\boldsymbol{\Omega} = \mathbf{V}\boldsymbol{\Omega} - \boldsymbol{\Omega}\mathbf{V}\boldsymbol{\Omega}. \tag{A.16}$$

By defining a resolvent operator as:

$$\mathbf{R}(\mathbf{E}_0^{\alpha} - \mathbf{H}_0) = \mathbf{Q}, \tag{A.17}$$

this can be applied to eq. A.16 in order to isolate an expression for $\boldsymbol{\Omega}$:

$$\mathbf{Q}\boldsymbol{\Omega} = \mathbf{R}(\mathbf{V}\boldsymbol{\Omega} - \boldsymbol{\Omega}\mathbf{V}\boldsymbol{\Omega}), \tag{A.18}$$

and using the relation $\mathbf{Q} = 1 - \mathbf{P}$:

$$\boldsymbol{\Omega} = \mathbf{P} + \mathbf{R}(\mathbf{V}\boldsymbol{\Omega} - \boldsymbol{\Omega}\mathbf{V}\boldsymbol{\Omega}). \tag{A.19}$$

Solving this iteratively it can be shown that the terms in the expansion of $\boldsymbol{\Omega}$ up to second order in $\mathbf{V}$ are:

$$\boldsymbol{\Omega} = \mathbf{P} + \mathbf{R}\mathbf{V}\mathbf{P} + \mathbf{R}\mathbf{V}\mathbf{R}\mathbf{V}\mathbf{P} - \mathbf{R}^2\mathbf{V}\mathbf{P}\mathbf{V}\mathbf{P} + O(\mathbf{V}^3). \tag{A.20}$$

Substituting this expansion of $\boldsymbol{\Omega}$ into the expression for the effective Hamiltonian in eq. A.13 leads to:

$$\mathbf{H}_{\text{eff}} = \mathbf{P}\mathbf{H}_0\mathbf{P} + \mathbf{P}\mathbf{V}\mathbf{P} + \mathbf{P}\mathbf{V}\mathbf{R}\mathbf{V}\mathbf{P} + O(\mathbf{V}^3). \tag{A.21}$$

Appendix B

Programs for calculating the elements of the effective Hamiltonian matrices

The research project that studied the interplay between the spin and orbital degrees of freedom in the magnetic interactions of the FeTe system lead to several scripts being written to carry out both small and larger computationally difficult tasks. These tasks included the calculation of the crystal field splitting, Hund's rule coupling terms and the tight-binding hopping integrals, as well as determining the elements of the 36×36 effective Hamiltonian matrix using the Rayleigh-Schrödinger perturbative model, which was discussed in full in Sec. 5. Some of the most important scripts are listed below with a brief description of their purpose.

B.1 Real orbital basis for the Fe^{2+} and Te^{2-} ions

The following script was used to model the functional forms of the valence shell atomic orbitals in the 3d and 5p orbitals in Fe and Te respectively. They were used in calculating the crystal field splitting, Hund's rule coupling terms and the tight-binding hopping integrals.

```
import numpy as np

class HydrogenLikeOrbitals:
    def __init__(self, Z):
        self.Z = Z
        self.bohrRadius = 5.29177210903E-11
        self.bohrLikeRadius = self.bohrRadius / Z

    def hydrogenicOrbitals3d(self, state, rho, theta, psi):
        def complexHydrogenicOrbitals3d(self, m):
            if (m == 0):
                prefactor = (1 / 486) * (6 / (np.pi * self.
                                                bohrLikeRadius
                                                ** 3)) ** 0.5
                rhoFactor = (rho / self.bohrLikeRadius) ** 2 *
                                                np.exp(- ((rho /
                                                 self.
                                                bohrLikeRadius)
                                                / 3))
```

```
            thetaFactor = (3 * np.cos(theta) ** 2 - 1)
            return prefactor * rhoFactor * thetaFactor
        elif (abs(m) == 1):
            prefactor = np.sign(m) * (1 / 81) * (1 / (np.pi
                                            * self.
                                            bohrLikeRadius
                                            ** 3)) ** 0.5
            rhoFactor = (rho / self.bohrLikeRadius) ** 2 *
                                            np.exp(- ((rho /
                                             self.
                                            bohrLikeRadius)
                                            / 3))
            thetaFactor = np.sin(theta) * np.cos(theta)
            psiFactor = np.cos(m * psi) + (np.sign(m) * np.
                                            sin(abs(m) * psi
                                            ))*1j
            return prefactor * rhoFactor * thetaFactor *
                                            psiFactor
        elif (abs(m) == 2):
            prefactor = (1 / 162) * (1 / (np.pi * self.
                                            bohrLikeRadius
                                            ** 3)) ** 0.5
            rhoFactor = (rho / self.bohrLikeRadius) ** 2 *
                                            np.exp(- ((rho /
                                             self.
                                            bohrLikeRadius)
                                            / 3))
            thetaFactor = np.sin(theta) ** 2
            psiFactor = np.cos(m * psi) + (np.sign(m) * np.
                                            sin(abs(m) * psi
                                            ))*1j
            return prefactor * rhoFactor * thetaFactor *
                                            psiFactor
    if (state == "z2"):
        return complexHydrogenicOrbitals3d(self, 0)
    elif (state == "xz"):
        return (1 / np.sqrt(2)) *
          (complexHydrogenicOrbitals3d(self, -1) -
complexHydrogenicOrbitals3d(self, 1))
    elif (state == "yz"):
        return (complex(0, 1) / np.sqrt(2)) *
          (complexHydrogenicOrbitals3d(self, -1) +
          complexHydrogenicOrbitals3d(self, 1))
    elif (state == "xy"):
        return (1 / (np.sqrt(2) * complex(0, 1))) *
          (complexHydrogenicOrbitals3d(self, 2) -
          complexHydrogenicOrbitals3d(self, -2))
    elif (state == "x2-y2"):
        return (1 / np.sqrt(2)) *
          (complexHydrogenicOrbitals3d(self, 2) +
          complexHydrogenicOrbitals3d(self, -2))
    elif (state == "Xz"):
        return (1 / np.sqrt(2)) * ((1 / np.sqrt(2)) *
          (complexHydrogenicOrbitals3d(self, -1) -
          complexHydrogenicOrbitals3d(self, 1)) +
          (complex(0, 1) / np.sqrt(2)) *
          (complexHydrogenicOrbitals3d(self, -1) +
```

```
              complexHydrogenicOrbitals3d(self, 1)))
        elif (state == "Yz"):
            return (1 / np.sqrt(2)) * ((1 / np.sqrt(2)) *
              (complexHydrogenicOrbitals3d(self, -1) -
              complexHydrogenicOrbitals3d(self, 1)) -
              (complex(0, 1) / np.sqrt(2)) *
              (complexHydrogenicOrbitals3d(self, -1) +
              complexHydrogenicOrbitals3d(self, 1)))

    def hydrogenicOrbitals5p(self, state, rho, theta, psi):
        # this is a wrapper function to output the orbital
                                        wavefunctions using a
                                        complex basis
        def complexHydrogenicOrbitals5p(self, m):
            if (m == 0):
                radialFunction = (1 / (150 * np.sqrt(30))) * (1
                                                / (self.
                                                bohrLikeRadius
                                                ** 3)) ** 0.5 *
                                                (120 - 90 * ((0.
                                                4 * rho) / self.
                                                bohrLikeRadius)
                                                + 18 * ((0.4 *
                                                rho) / self.
                                                bohrLikeRadius)
                                                ** 2 - ((0.4 *
                                                rho) / self.
                                                bohrLikeRadius)
                                                ** 3) * ((0.4 *
                                                rho) / self.
                                                bohrLikeRadius)
                                                * np.exp(- ((0.4
                                                 * rho) / self.
                                                bohrLikeRadius)
                                                / 2)
                sphericalFunction = np.sqrt(3 / (4 * np.pi)) *
                                                np.cos(theta)
                return radialFunction * sphericalFunction
            elif (abs(m) == 1):
```

```
                radialFunction = (1 / (150 * np.sqrt(30))) * (1
                                                / (self.
                                                bohrLikeRadius
                                                ** 3)) ** 0.5 *
                                                (120 - 90 * ((0.
                                                4 * rho) / self.
                                                bohrLikeRadius)
                                                + 18 * ((0.4 *
                                                rho) / self.
                                                bohrLikeRadius)
                                                ** 2 - ((0.4 *
                                                rho) / self.
                                                bohrLikeRadius)
                                                ** 3) * ((0.4 *
                                                rho) / self.
                                                bohrLikeRadius)
                                                * np.exp(- ((0.4
                                                 * rho) / self.
                                                bohrLikeRadius)
                                                / 2)
                sphericalFunction = np.sqrt(3 / (8 * np.pi)) *
                                                np.sin(theta) *
                                                (np.cos(psi) + (
                                                np.sign(m) * np.
                                                sin(psi))*1j)
                return radialFunction * sphericalFunction
        # we now convert to the real basis
        if (state == "x"):
            realWavefunction = (1 / np.sqrt(2)) *
              (complexHydrogenicOrbitals5p(self, -1) +
              complexHydrogenicOrbitals5p(self, 1))
            return realWavefunction
        elif (state == "y"):
            return (complex(0, 1) / np.sqrt(2)) *
              (complexHydrogenicOrbitals5p(self, -1) -
              complexHydrogenicOrbitals5p(self, 1))
        elif (state == "z"):
            return complexHydrogenicOrbitals5p(self, 0)
```

B.2 Script for mathematical transformations and integrands

This script was used to transform between cartesian and spherical coordinates, which was especially helpful for carrying out the numerical integrations using the Monte Carlo method.

```
import numpy as np

class Mathematics:
    def sphericalToCartesian(rho, theta, psi):
        x = rho * np.cos(theta) * np.sin(psi)
        y = rho * np.sin(theta) * np.sin(psi)
        z = rho * np.cos(psi)
        return x, y, z
```

```
    # function to convert cartesian to spherical coordinates.
    def cartesianToSpherical(x, y, z):
        rho = np.sqrt(x ** 2 + y ** 2 + z ** 2)
        with np.errstate(divide="ignore"):
            theta = np.where(rho == 0, 0, np.arccos(z / rho))
        psi = np.arctan2(y, x)
        return rho, theta, psi
```

Moreover, the script presented below gives the integrands used for calculating the constants used as inputs to the Rayleigh-Schrödinger perturbative model.

```
from BuildFeTeLattice import FeTe
from HydrogenLikeOrbitals import HydrogenLikeOrbitals
from Mathematics import Mathematics
import numpy as np

electronCharge = 1.602E-19
epsilon = 8.854187812813E-12
myScreeningLength = 2E-10
myZeta = 0.2E-14

FeOrbitals = HydrogenLikeOrbitals(1.25)
TeOrbitals = HydrogenLikeOrbitals(4.75)
FeTeLattice = FeTe(distortion=0.01).getTeAtoms()

print("FeTeLattice: ", FeTeLattice)
bondLengths = [np.linalg.norm(FeTeLattice[0]), np.linalg.norm(
                                FeTeLattice[1]), np.linalg.norm(
                                FeTeLattice[2]), np.linalg.norm(
                                FeTeLattice[3])]

def rotation_matrix_xy_plane(angle):
    matrix = np.array([[np.cos(angle), -np.sin(angle), 0],[np.
                                    sin(angle), np.cos(angle), 0
                                    ],[0, 0, 1]])
    return matrix

FeTeLattice = [np.matmul(FeTeLattice[i],
                                rotation_matrix_xy_plane(np.pi/4
                                )) for i in range(len(
                                FeTeLattice))]

print("Rotated FeTe: ", FeTeLattice)

def mirror_in_xy_plane(my_list):
    mirrored_list = []
    for i in range(len(my_list)):
        element = []
        for j in range(3):
            if j == 0:
                element.append(np.array(my_list[i][j]))
            elif j == 1:
                element.append(np.array(my_list[i][j]))
            elif j == 2:
                element.append(-np.array(my_list[i][j]))
```

```
        mirrored_list.append(element)
    return mirrored_list

def potentialEnergyDoubleOccupancy(x, y, z, u, v, w, zeta,
                                screeningLength):
    distance = np.sqrt((x - u) ** 2 + (y - v) ** 2 + (z - w) **
                                    2 + zeta ** 2)
    potential = (electronCharge ** 2) / (4 * np.pi * epsilon *
                                    distance) * np.exp(-
                                    distance / screeningLength)
    return potential

def crystalField(x, y, z, posIons, ZIons, zeta, screeningLength)
                                :
    num = len(posIons)
    potential = 0
    for i in range(num):
        distance = np.sqrt((x - posIons[i][0]) ** 2 + (y -
                                        posIons[i][1]) ** 2 + (z
                                         - posIons[i][2]) ** 2 +
                                         zeta ** 2)
        potential += (- ZIons[i] * electronCharge ** 2) / (4 *
                                        np.pi * epsilon *
                                        distance) * np.exp(-
                                        distance /
                                        screeningLength)
    return potential

def effectiveCrystalField(x, y, z, posFe, FeCharge, TeCharge,
                                zeta, screeningLength):
    distanceTe = np.sqrt(x ** 2 + y ** 2 + z ** 2 + zeta ** 2)
    distanceFe = np.sqrt((x - posFe[0]) ** 2 + (y - posFe[1]) **
                                     2 + (z - posFe[2]) ** 2 +
                                    zeta ** 2)
    effectiveV = - ((1 / 8) * np.pi * epsilon) * electronCharge
                                    * ((FeCharge / distanceFe) *
                                     np.exp(- distanceFe /
                                    screeningLength) + (TeCharge
                                     / distanceTe) * np.exp(-
                                    distanceTe / screeningLength
                                    ))
    return effectiveV

crystal_field_splitting_dxy = lambda x, y, z: FeOrbitals.
                                hydrogenicOrbitals3d("xy", *
                                Mathematics.cartesianToSpherical
                                (x, y, z)) * np.conjugate(
                                FeOrbitals.hydrogenicOrbitals3d(
                                "xy", *Mathematics.
                                cartesianToSpherical(x, y, z)))
                                * crystalField(x, y, z,
                                FeTeLattice, [-2, -2, -2, -2],
                                myZeta, myScreeningLength)
```

```
crystal_field_splitting_dXz = lambda x, y, z: FeOrbitals.
                                hydrogenicOrbitals3d("Xz", *
                                Mathematics.cartesianToSpherical
                                (x, y, z)) * np.conjugate(
                                FeOrbitals.hydrogenicOrbitals3d(
                                "Xz", *Mathematics.
                                cartesianToSpherical(x, y, z)))
                                * crystalField(x, y, z,
                                FeTeLattice, [-2, -2, -2, -2],
                                myZeta, myScreeningLength)
crystal_field_splitting_dYz = lambda x, y, z: FeOrbitals.
                                hydrogenicOrbitals3d("Yz", *
                                Mathematics.cartesianToSpherical
                                (x, y, z)) * np.conjugate(
                                FeOrbitals.hydrogenicOrbitals3d(
                                "Yz", *Mathematics.
                                cartesianToSpherical(x, y, z)))
                                * crystalField(x, y, z,
                                FeTeLattice, [-2, -2, -2, -2],
                                myZeta, myScreeningLength)
crystal_field_splitting_dx2y2 = lambda x, y, z: FeOrbitals.
                                hydrogenicOrbitals3d("x2-y2", *
                                Mathematics.cartesianToSpherical
                                (x, y, z)) * np.conjugate(
                                FeOrbitals.hydrogenicOrbitals3d(
                                "x2-y2", *Mathematics.
                                cartesianToSpherical(x, y, z)))
                                * crystalField(x, y, z,
                                FeTeLattice, [-2, -2, -2, -2],
                                myZeta, myScreeningLength)
crystal_field_splitting_dz2 = lambda x, y, z: FeOrbitals.
                                hydrogenicOrbitals3d("z2", *
                                Mathematics.cartesianToSpherical
                                (x, y, z)) * np.conjugate(
                                FeOrbitals.hydrogenicOrbitals3d(
                                "z2", *Mathematics.
                                cartesianToSpherical(x, y, z)))
                                * crystalField(x, y, z,
                                FeTeLattice, [-2, -2, -2, -2],
                                myZeta, myScreeningLength)

test_func_Fe_dxy = lambda x, y, z: FeOrbitals.
                                hydrogenicOrbitals3d("xy", *
                                Mathematics.cartesianToSpherical
                                (x, y, z)) * np.conjugate(
                                FeOrbitals.hydrogenicOrbitals3d(
                                "xy", *Mathematics.
                                cartesianToSpherical(x, y, z)))
                                * 1.609E-19

test_func_Te_x = lambda x, y, z: TeOrbitals.hydrogenicOrbitals5p
                                ("x", *Mathematics.
                                cartesianToSpherical(x, y, z)) *
                                 np.conjugate(TeOrbitals.
                                hydrogenicOrbitals5p("x", *
                                Mathematics.cartesianToSpherical
                                (x, y, z))) * 1.609E-19
```

```
test_func_Te_y = lambda x, y, z: TeOrbitals.hydrogenicOrbitals5p
                                ("y", *Mathematics.
                                cartesianToSpherical(x, y, z)) *
                                 np.conjugate(TeOrbitals.
                                hydrogenicOrbitals5p("y", *
                                Mathematics.cartesianToSpherical
                                (x, y, z))) * 1.609E-19
test_func_Te_z = lambda x, y, z: TeOrbitals.hydrogenicOrbitals5p
                                ("z", *Mathematics.
                                cartesianToSpherical(x, y, z)) *
                                 np.conjugate(TeOrbitals.
                                hydrogenicOrbitals5p("z", *
                                Mathematics.cartesianToSpherical
                                (x, y, z))) * 1.609E-19

U_pz_pz_func = lambda x, y, z, u, v, w: 2 *
                                potentialEnergyDoubleOccupancy(x
                                , y, z, u, v, w, zeta=myZeta,
                                screeningLength=
                                myScreeningLength) * np.
                                conjugate(TeOrbitals.
                                hydrogenicOrbitals5p("z", *
                                Mathematics.cartesianToSpherical
                                (x, y, z))) * np.conjugate(
                                TeOrbitals.hydrogenicOrbitals5p(
                                "z", *Mathematics.
                                cartesianToSpherical(u, v, w)))
                                * TeOrbitals.
                                hydrogenicOrbitals5p("z", *
                                Mathematics.cartesianToSpherical
                                (u, v, w)) * TeOrbitals.
                                hydrogenicOrbitals5p("z", *
                                Mathematics.cartesianToSpherical
                                (x, y, z))
U_py_py_func = lambda x, y, z, u, v, w: 2 *
                                potentialEnergyDoubleOccupancy(x
                                , y, z, u, v, w, zeta=myZeta,
                                screeningLength=
                                myScreeningLength) * np.
                                conjugate(TeOrbitals.
                                hydrogenicOrbitals5p("y", *
                                Mathematics.cartesianToSpherical
                                (x, y, z))) * np.conjugate(
                                TeOrbitals.hydrogenicOrbitals5p(
                                "y", *Mathematics.
                                cartesianToSpherical(u, v, w)))
                                * TeOrbitals.
                                hydrogenicOrbitals5p("y", *
                                Mathematics.cartesianToSpherical
                                (u, v, w)) * TeOrbitals.
                                hydrogenicOrbitals5p("y", *
                                Mathematics.cartesianToSpherical
                                (x, y, z))
```

```
U_py_pz_func = lambda x, y, z, u, v, w: 2 *
                                potentialEnergyDoubleOccupancy(x
                                , y, z, u, v, w, zeta=myZeta,
                                screeningLength=
                                myScreeningLength) * np.
                                conjugate(TeOrbitals.
                                hydrogenicOrbitals5p("y", *
                                Mathematics.cartesianToSpherical
                                (x, y, z))) * np.conjugate(
                                TeOrbitals.hydrogenicOrbitals5p(
                                "z", *Mathematics.
                                cartesianToSpherical(u, v, w)))
                                * TeOrbitals.
                                hydrogenicOrbitals5p("z", *
                                Mathematics.cartesianToSpherical
                                (u, v, w)) * TeOrbitals.
                                hydrogenicOrbitals5p("y", *
                                Mathematics.cartesianToSpherical
                                (x, y, z))
U_px_px_func = lambda x, y, z, u, v, w: 2 *
                                potentialEnergyDoubleOccupancy(x
                                , y, z, u, v, w, zeta=myZeta,
                                screeningLength=
                                myScreeningLength) * np.
                                conjugate(TeOrbitals.
                                hydrogenicOrbitals5p("x", *
                                Mathematics.cartesianToSpherical
                                (x, y, z))) * np.conjugate(
                                TeOrbitals.hydrogenicOrbitals5p(
                                "x", *Mathematics.
                                cartesianToSpherical(u, v, w)))
                                * TeOrbitals.
                                hydrogenicOrbitals5p("x", *
                                Mathematics.cartesianToSpherical
                                (u, v, w)) * TeOrbitals.
                                hydrogenicOrbitals5p("x", *
                                Mathematics.cartesianToSpherical
                                (x, y, z))
U_px_py_func = lambda x, y, z, u, v, w: 2 *
                                potentialEnergyDoubleOccupancy(x
                                , y, z, u, v, w, zeta=myZeta,
                                screeningLength=
                                myScreeningLength) * np.
                                conjugate(TeOrbitals.
                                hydrogenicOrbitals5p("x", *
                                Mathematics.cartesianToSpherical
                                (x, y, z))) * np.conjugate(
                                TeOrbitals.hydrogenicOrbitals5p(
                                "y", *Mathematics.
                                cartesianToSpherical(u, v, w)))
                                * TeOrbitals.
                                hydrogenicOrbitals5p("y", *
                                Mathematics.cartesianToSpherical
                                (u, v, w)) * TeOrbitals.
                                hydrogenicOrbitals5p("x", *
                                Mathematics.cartesianToSpherical
                                (x, y, z))
```

```
U_px_pz_func = lambda x, y, z, u, v, w: 2 *
                                potentialEnergyDoubleOccupancy(x
                                , y, z, u, v, w, zeta=myZeta,
                                screeningLength=
                                myScreeningLength) * np.
                                conjugate(TeOrbitals.
                                hydrogenicOrbitals5p("x", *
                                Mathematics.cartesianToSpherical
                                (x, y, z))) * np.conjugate(
                                TeOrbitals.hydrogenicOrbitals5p(
                                "z", *Mathematics.
                                cartesianToSpherical(u, v, w)))
                                * TeOrbitals.
                                hydrogenicOrbitals5p("z", *
                                Mathematics.cartesianToSpherical
                                (u, v, w)) * TeOrbitals.
                                hydrogenicOrbitals5p("x", *
                                Mathematics.cartesianToSpherical
                                (x, y, z))

U_dxy_dxy_func = lambda x, y, z, u, v, w: 2 * FeOrbitals.
                                hydrogenicOrbitals3d("xy", *
                                Mathematics.cartesianToSpherical
                                (x, y, z)) * np.conjugate(
                                FeOrbitals.hydrogenicOrbitals3d(
                                "xy", *Mathematics.
                                cartesianToSpherical(x, y, z)))
                                * potentialEnergyDoubleOccupancy
                                (x, y, z, u, v, w, zeta=myZeta,
                                screeningLength=
                                myScreeningLength) * FeOrbitals.
                                hydrogenicOrbitals3d("xy", *
                                Mathematics.cartesianToSpherical
                                (u, v, w)) * np.conjugate(
                                FeOrbitals.hydrogenicOrbitals3d(
                                "xy", *Mathematics.
                                cartesianToSpherical(u, v, w)))
U_dxy_dXz_func = lambda x, y, z, u, v, w: 2 * FeOrbitals.
                                hydrogenicOrbitals3d("xy", *
                                Mathematics.cartesianToSpherical
                                (x, y, z)) * np.conjugate(
                                FeOrbitals.hydrogenicOrbitals3d(
                                "xy", *Mathematics.
                                cartesianToSpherical(x, y, z)))
                                * potentialEnergyDoubleOccupancy
                                (x, y, z, u, v, w, zeta=myZeta,
                                screeningLength=
                                myScreeningLength) * FeOrbitals.
                                hydrogenicOrbitals3d("Xz", *
                                Mathematics.cartesianToSpherical
                                (u, v, w)) * np.conjugate(
                                FeOrbitals.hydrogenicOrbitals3d(
                                "Xz", *Mathematics.
                                cartesianToSpherical(u, v, w)))
```

```
U_dxy_dYz_func = lambda x, y, z, u, v, w: 2 * FeOrbitals.
                                hydrogenicOrbitals3d("xy", *
                                Mathematics.cartesianToSpherical
                                (x, y, z)) * np.conjugate(
                                FeOrbitals.hydrogenicOrbitals3d(
                                "xy", *Mathematics.
                                cartesianToSpherical(x, y, z)))
                                * potentialEnergyDoubleOccupancy
                                (x, y, z, u, v, w, zeta=myZeta,
                                screeningLength=
                                myScreeningLength) * FeOrbitals.
                                hydrogenicOrbitals3d("Yz", *
                                Mathematics.cartesianToSpherical
                                (u, v, w)) * np.conjugate(
                                FeOrbitals.hydrogenicOrbitals3d(
                                "Yz", *Mathematics.
                                cartesianToSpherical(u, v, w)))
U_dxy_dx2y2_func = lambda x, y, z, u, v, w: 2 * FeOrbitals.
                                hydrogenicOrbitals3d("xy", *
                                Mathematics.cartesianToSpherical
                                (x, y, z)) * np.conjugate(
                                FeOrbitals.hydrogenicOrbitals3d(
                                "xy", *Mathematics.
                                cartesianToSpherical(x, y, z)))
                                * potentialEnergyDoubleOccupancy
                                (x, y, z, u, v, w, zeta=myZeta,
                                screeningLength=
                                myScreeningLength) * FeOrbitals.
                                hydrogenicOrbitals3d("x2-y2", *
                                Mathematics.cartesianToSpherical
                                (u, v, w)) * np.conjugate(
                                FeOrbitals.hydrogenicOrbitals3d(
                                "x2-y2", *Mathematics.
                                cartesianToSpherical(u, v, w)))
U_dxy_dz2_func = lambda x, y, z, u, v, w: 2 * FeOrbitals.
                                hydrogenicOrbitals3d("xy", *
                                Mathematics.cartesianToSpherical
                                (x, y, z)) * np.conjugate(
                                FeOrbitals.hydrogenicOrbitals3d(
                                "xy", *Mathematics.
                                cartesianToSpherical(x, y, z)))
                                * potentialEnergyDoubleOccupancy
                                (x, y, z, u, v, w, zeta=myZeta,
                                screeningLength=
                                myScreeningLength) * FeOrbitals.
                                hydrogenicOrbitals3d("z2", *
                                Mathematics.cartesianToSpherical
                                (u, v, w)) * np.conjugate(
                                FeOrbitals.hydrogenicOrbitals3d(
                                "z2", *Mathematics.
                                cartesianToSpherical(u, v, w)))
```

```
U_dXz_dXz_func = lambda x, y, z, u, v, w: 2 * FeOrbitals.
                                hydrogenicOrbitals3d("Xz", *
                                Mathematics.cartesianToSpherical
                                (x, y, z)) * np.conjugate(
                                FeOrbitals.hydrogenicOrbitals3d(
                                "Xz", *Mathematics.
                                cartesianToSpherical(x, y, z)))
                                * potentialEnergyDoubleOccupancy
                                (x, y, z, u, v, w, zeta=myZeta,
                                screeningLength=
                                myScreeningLength) * FeOrbitals.
                                hydrogenicOrbitals3d("Xz", *
                                Mathematics.cartesianToSpherical
                                (u, v, w)) * np.conjugate(
                                FeOrbitals.hydrogenicOrbitals3d(
                                "Xz", *Mathematics.
                                cartesianToSpherical(u, v, w)))
U_dXz_dYz_func = lambda x, y, z, u, v, w: 2 * FeOrbitals.
                                hydrogenicOrbitals3d("Xz", *
                                Mathematics.cartesianToSpherical
                                (x, y, z)) * np.conjugate(
                                FeOrbitals.hydrogenicOrbitals3d(
                                "Xz", *Mathematics.
                                cartesianToSpherical(x, y, z)))
                                * potentialEnergyDoubleOccupancy
                                (x, y, z, u, v, w, zeta=myZeta,
                                screeningLength=
                                myScreeningLength) * FeOrbitals.
                                hydrogenicOrbitals3d("Yz", *
                                Mathematics.cartesianToSpherical
                                (u, v, w)) * np.conjugate(
                                FeOrbitals.hydrogenicOrbitals3d(
                                "Yz", *Mathematics.
                                cartesianToSpherical(u, v, w)))
U_dXz_dx2y2_func = lambda x, y, z, u, v, w: 2 * FeOrbitals.
                                hydrogenicOrbitals3d("Xz", *
                                Mathematics.cartesianToSpherical
                                (x, y, z)) * np.conjugate(
                                FeOrbitals.hydrogenicOrbitals3d(
                                "Xz", *Mathematics.
                                cartesianToSpherical(x, y, z)))
                                * potentialEnergyDoubleOccupancy
                                (x, y, z, u, v, w, zeta=myZeta,
                                screeningLength=
                                myScreeningLength) * FeOrbitals.
                                hydrogenicOrbitals3d("x2-y2", *
                                Mathematics.cartesianToSpherical
                                (u, v, w)) * np.conjugate(
                                FeOrbitals.hydrogenicOrbitals3d(
                                "x2-y2", *Mathematics.
                                cartesianToSpherical(u, v, w)))
```

```
U_dXz_dz2_func = lambda x, y, z, u, v, w: 2 * FeOrbitals.
                                 hydrogenicOrbitals3d("Xz", *
                                 Mathematics.cartesianToSpherical
                                 (x, y, z)) * np.conjugate(
                                 FeOrbitals.hydrogenicOrbitals3d(
                                 "Xz", *Mathematics.
                                 cartesianToSpherical(x, y, z)))
                                 * potentialEnergyDoubleOccupancy
                                 (x, y, z, u, v, w, zeta=myZeta,
                                 screeningLength=
                                 myScreeningLength) * FeOrbitals.
                                 hydrogenicOrbitals3d("z2", *
                                 Mathematics.cartesianToSpherical
                                 (u, v, w)) * np.conjugate(
                                 FeOrbitals.hydrogenicOrbitals3d(
                                 "z2", *Mathematics.
                                 cartesianToSpherical(u, v, w)))
U_dYz_dYz_func = lambda x, y, z, u, v, w: 2 * FeOrbitals.
                                 hydrogenicOrbitals3d("Yz", *
                                 Mathematics.cartesianToSpherical
                                 (x, y, z)) * np.conjugate(
                                 FeOrbitals.hydrogenicOrbitals3d(
                                 "Yz", *Mathematics.
                                 cartesianToSpherical(x, y, z)))
                                 * potentialEnergyDoubleOccupancy
                                 (x, y, z, u, v, w, zeta=myZeta,
                                 screeningLength=
                                 myScreeningLength) * FeOrbitals.
                                 hydrogenicOrbitals3d("Yz", *
                                 Mathematics.cartesianToSpherical
                                 (u, v, w)) * np.conjugate(
                                 FeOrbitals.hydrogenicOrbitals3d(
                                 "Yz", *Mathematics.
                                 cartesianToSpherical(u, v, w)))
U_dYz_dx2y2_func = lambda x, y, z, u, v, w: 2 * FeOrbitals.
                                 hydrogenicOrbitals3d("Yz", *
                                 Mathematics.cartesianToSpherical
                                 (x, y, z)) * np.conjugate(
                                 FeOrbitals.hydrogenicOrbitals3d(
                                 "Yz", *Mathematics.
                                 cartesianToSpherical(x, y, z)))
                                 * potentialEnergyDoubleOccupancy
                                 (x, y, z, u, v, w, zeta=myZeta,
                                 screeningLength=
                                 myScreeningLength) * FeOrbitals.
                                 hydrogenicOrbitals3d("x2-y2", *
                                 Mathematics.cartesianToSpherical
                                 (u, v, w)) * np.conjugate(
                                 FeOrbitals.hydrogenicOrbitals3d(
                                 "x2-y2", *Mathematics.
                                 cartesianToSpherical(u, v, w)))
```

```
U_dYz_dz2_func = lambda x, y, z, u, v, w: 2 * FeOrbitals.
                                hydrogenicOrbitals3d("Yz", *
                                Mathematics.cartesianToSpherical
                                (x, y, z)) * np.conjugate(
                                FeOrbitals.hydrogenicOrbitals3d(
                                "Yz", *Mathematics.
                                cartesianToSpherical(x, y, z)))
                                * potentialEnergyDoubleOccupancy
                                (x, y, z, u, v, w, zeta=myZeta,
                                screeningLength=
                                myScreeningLength) * FeOrbitals.
                                hydrogenicOrbitals3d("z2", *
                                Mathematics.cartesianToSpherical
                                (u, v, w)) * np.conjugate(
                                FeOrbitals.hydrogenicOrbitals3d(
                                "z2", *Mathematics.
                                cartesianToSpherical(u, v, w)))
U_dx2y2_dx2y2_func = lambda x, y, z, u, v, w: 2 * FeOrbitals.
                                hydrogenicOrbitals3d("x2-y2", *
                                Mathematics.cartesianToSpherical
                                (x, y, z)) * np.conjugate(
                                FeOrbitals.hydrogenicOrbitals3d(
                                "x2-y2", *Mathematics.
                                cartesianToSpherical(x, y, z)))
                                * potentialEnergyDoubleOccupancy
                                (x, y, z, u, v, w, zeta=myZeta,
                                screeningLength=
                                myScreeningLength) * FeOrbitals.
                                hydrogenicOrbitals3d("x2-y2", *
                                Mathematics.cartesianToSpherical
                                (u, v, w)) * np.conjugate(
                                FeOrbitals.hydrogenicOrbitals3d(
                                "x2-y2", *Mathematics.
                                cartesianToSpherical(u, v, w)))
U_dx2y2_dz2_func = lambda x, y, z, u, v, w: 2 * FeOrbitals.
                                hydrogenicOrbitals3d("x2-y2", *
                                Mathematics.cartesianToSpherical
                                (x, y, z)) * np.conjugate(
                                FeOrbitals.hydrogenicOrbitals3d(
                                "x2-y2", *Mathematics.
                                cartesianToSpherical(x, y, z)))
                                * potentialEnergyDoubleOccupancy
                                (x, y, z, u, v, w, zeta=myZeta,
                                screeningLength=
                                myScreeningLength) * FeOrbitals.
                                hydrogenicOrbitals3d("z2", *
                                Mathematics.cartesianToSpherical
                                (u, v, w)) * np.conjugate(
                                FeOrbitals.hydrogenicOrbitals3d(
                                "z2", *Mathematics.
                                cartesianToSpherical(u, v, w)))
```

```
U_dz2_dz2_func = lambda x, y, z, u, v, w: 2 * FeOrbitals.
                                hydrogenicOrbitals3d("z2", *
                                Mathematics.cartesianToSpherical
                                (x, y, z)) * np.conjugate(
                                FeOrbitals.hydrogenicOrbitals3d(
                                "z2", *Mathematics.
                                cartesianToSpherical(x, y, z)))
                                * potentialEnergyDoubleOccupancy
                                (x, y, z, u, v, w, zeta=myZeta,
                                screeningLength=
                                myScreeningLength) * FeOrbitals.
                                hydrogenicOrbitals3d("z2", *
                                Mathematics.cartesianToSpherical
                                (u, v, w)) * np.conjugate(
                                FeOrbitals.hydrogenicOrbitals3d(
                                "z2", *Mathematics.
                                cartesianToSpherical(u, v, w)))

J_pz_pz_func = lambda x, y, z, u, v, w: 2 *
                                potentialEnergyDoubleOccupancy(x
                                , y, z, u, v, w, zeta=myZeta,
                                screeningLength=
                                myScreeningLength) * np.
                                conjugate(TeOrbitals.
                                hydrogenicOrbitals5p("z", *
                                Mathematics.cartesianToSpherical
                                (x, y, z))) * np.conjugate(
                                TeOrbitals.hydrogenicOrbitals5p(
                                "z", *Mathematics.
                                cartesianToSpherical(u, v, w)))
                                * TeOrbitals.
                                hydrogenicOrbitals5p("z", *
                                Mathematics.cartesianToSpherical
                                (u, v, w)) * TeOrbitals.
                                hydrogenicOrbitals5p("z", *
                                Mathematics.cartesianToSpherical
                                (x, y, z))
J_py_py_func = lambda x, y, z, u, v, w: 2 *
                                potentialEnergyDoubleOccupancy(x
                                , y, z, u, v, w, zeta=myZeta,
                                screeningLength=
                                myScreeningLength) * np.
                                conjugate(TeOrbitals.
                                hydrogenicOrbitals5p("y", *
                                Mathematics.cartesianToSpherical
                                (x, y, z))) * np.conjugate(
                                TeOrbitals.hydrogenicOrbitals5p(
                                "y", *Mathematics.
                                cartesianToSpherical(u, v, w)))
                                * TeOrbitals.
                                hydrogenicOrbitals5p("y", *
                                Mathematics.cartesianToSpherical
                                (u, v, w)) * TeOrbitals.
                                hydrogenicOrbitals5p("y", *
                                Mathematics.cartesianToSpherical
                                (x, y, z))
```

```
J_py_pz_func = lambda x, y, z, u, v, w: 2 *
                                potentialEnergyDoubleOccupancy(x
                                , y, z, u, v, w, zeta=myZeta,
                                screeningLength=
                                myScreeningLength) * np.
                                conjugate(TeOrbitals.
                                hydrogenicOrbitals5p("y", *
                                Mathematics.cartesianToSpherical
                                (x, y, z))) * np.conjugate(
                                TeOrbitals.hydrogenicOrbitals5p(
                                "z", *Mathematics.
                                cartesianToSpherical(u, v, w)))
                                * TeOrbitals.
                                hydrogenicOrbitals5p("y", *
                                Mathematics.cartesianToSpherical
                                (u, v, w)) * TeOrbitals.
                                hydrogenicOrbitals5p("z", *
                                Mathematics.cartesianToSpherical
                                (x, y, z))
J_px_px_func = lambda x, y, z, u, v, w: 2 *
                                potentialEnergyDoubleOccupancy(x
                                , y, z, u, v, w, zeta=myZeta,
                                screeningLength=
                                myScreeningLength) * np.
                                conjugate(TeOrbitals.
                                hydrogenicOrbitals5p("x", *
                                Mathematics.cartesianToSpherical
                                (x, y, z))) * np.conjugate(
                                TeOrbitals.hydrogenicOrbitals5p(
                                "x", *Mathematics.
                                cartesianToSpherical(u, v, w)))
                                * TeOrbitals.
                                hydrogenicOrbitals5p("x", *
                                Mathematics.cartesianToSpherical
                                (u, v, w)) * TeOrbitals.
                                hydrogenicOrbitals5p("x", *
                                Mathematics.cartesianToSpherical
                                (x, y, z))
J_px_py_func = lambda x, y, z, u, v, w: 2 *
                                potentialEnergyDoubleOccupancy(x
                                , y, z, u, v, w, zeta=myZeta,
                                screeningLength=
                                myScreeningLength) * np.
                                conjugate(TeOrbitals.
                                hydrogenicOrbitals5p("x", *
                                Mathematics.cartesianToSpherical
                                (x, y, z))) * np.conjugate(
                                TeOrbitals.hydrogenicOrbitals5p(
                                "y", *Mathematics.
                                cartesianToSpherical(u, v, w)))
                                * TeOrbitals.
                                hydrogenicOrbitals5p("x", *
                                Mathematics.cartesianToSpherical
                                (u, v, w)) * TeOrbitals.
                                hydrogenicOrbitals5p("y", *
                                Mathematics.cartesianToSpherical
                                (x, y, z))
```

```
J_px_pz_func = lambda x, y, z, u, v, w: 2 *
                                potentialEnergyDoubleOccupancy(x
                                , y, z, u, v, w, zeta=myZeta,
                                screeningLength=
                                myScreeningLength) * np.
                                conjugate(TeOrbitals.
                                hydrogenicOrbitals5p("x", *
                                Mathematics.cartesianToSpherical
                                (x, y, z))) * np.conjugate(
                                TeOrbitals.hydrogenicOrbitals5p(
                                "z", *Mathematics.
                                cartesianToSpherical(u, v, w)))
                                * TeOrbitals.
                                hydrogenicOrbitals5p("x", *
                                Mathematics.cartesianToSpherical
                                (u, v, w)) * TeOrbitals.
                                hydrogenicOrbitals5p("z", *
                                Mathematics.cartesianToSpherical
                                (x, y, z))

J_dz2_dz2_func = lambda x, y, z, u, v, w: 2 *
                                potentialEnergyDoubleOccupancy(x
                                , y, z, u, v, w, zeta=myZeta,
                                screeningLength=
                                myScreeningLength) * np.
                                conjugate(FeOrbitals.
                                hydrogenicOrbitals3d("z2", *
                                Mathematics.cartesianToSpherical
                                (x, y, z))) * np.conjugate(
                                FeOrbitals.hydrogenicOrbitals3d(
                                "z2", *Mathematics.
                                cartesianToSpherical(u, v, w)))
                                * FeOrbitals.
                                hydrogenicOrbitals3d("z2", *
                                Mathematics.cartesianToSpherical
                                (u, v, w)) * FeOrbitals.
                                hydrogenicOrbitals3d("z2", *
                                Mathematics.cartesianToSpherical
                                (x, y, z))
```

```
J_dx2y2_dx2y2_func = lambda x, y, z, u, v, w: 2 *
                                potentialEnergyDoubleOccupancy(x
                                , y, z, u, v, w, zeta=myZeta,
                                screeningLength=
                                myScreeningLength) * np.
                                conjugate(FeOrbitals.
                                hydrogenicOrbitals3d("x2-y2", *
                                Mathematics.cartesianToSpherical
                                (x, y, z))) * np.conjugate(
                                FeOrbitals.hydrogenicOrbitals3d(
                                "x2-y2", *Mathematics.
                                cartesianToSpherical(u, v, w)))
                                * FeOrbitals.
                                hydrogenicOrbitals3d("x2-y2", *
                                Mathematics.cartesianToSpherical
                                (u, v, w)) * FeOrbitals.
                                hydrogenicOrbitals3d("x2-y2", *
                                Mathematics.cartesianToSpherical
                                (x, y, z))
J_dx2y2_dz2_func = lambda x, y, z, u, v, w: 2 *
                                potentialEnergyDoubleOccupancy(x
                                , y, z, u, v, w, zeta=myZeta,
                                screeningLength=
                                myScreeningLength) * np.
                                conjugate(FeOrbitals.
                                hydrogenicOrbitals3d("x2-y2", *
                                Mathematics.cartesianToSpherical
                                (x, y, z))) * np.conjugate(
                                FeOrbitals.hydrogenicOrbitals3d(
                                "z2", *Mathematics.
                                cartesianToSpherical(u, v, w)))
                                * FeOrbitals.
                                hydrogenicOrbitals3d("x2-y2", *
                                Mathematics.cartesianToSpherical
                                (u, v, w)) * FeOrbitals.
                                hydrogenicOrbitals3d("z2", *
                                Mathematics.cartesianToSpherical
                                (x, y, z))
J_dYz_dYz_func = lambda x, y, z, u, v, w: 2 *
                                potentialEnergyDoubleOccupancy(x
                                , y, z, u, v, w, zeta=myZeta,
                                screeningLength=
                                myScreeningLength) * np.
                                conjugate(FeOrbitals.
                                hydrogenicOrbitals3d("Yz", *
                                Mathematics.cartesianToSpherical
                                (x, y, z))) * np.conjugate(
                                FeOrbitals.hydrogenicOrbitals3d(
                                "Yz", *Mathematics.
                                cartesianToSpherical(u, v, w)))
                                * FeOrbitals.
                                hydrogenicOrbitals3d("Yz", *
                                Mathematics.cartesianToSpherical
                                (u, v, w)) * FeOrbitals.
                                hydrogenicOrbitals3d("Yz", *
                                Mathematics.cartesianToSpherical
                                (x, y, z))
```

```
J_dYz_dx2y2_func = lambda x, y, z, u, v, w: 2 *
                                potentialEnergyDoubleOccupancy(x
                                , y, z, u, v, w, zeta=myZeta,
                                screeningLength=
                                myScreeningLength) * np.
                                conjugate(FeOrbitals.
                                hydrogenicOrbitals3d("Yz", *
                                Mathematics.cartesianToSpherical
                                (x, y, z))) * np.conjugate(
                                FeOrbitals.hydrogenicOrbitals3d(
                                "x2-y2", *Mathematics.
                                cartesianToSpherical(u, v, w)))
                                * FeOrbitals.
                                hydrogenicOrbitals3d("Yz", *
                                Mathematics.cartesianToSpherical
                                (u, v, w)) * FeOrbitals.
                                hydrogenicOrbitals3d("x2-y2", *
                                Mathematics.cartesianToSpherical
                                (x, y, z))
J_dYz_dz2_func = lambda x, y, z, u, v, w: 2 *
                                potentialEnergyDoubleOccupancy(x
                                , y, z, u, v, w, zeta=myZeta,
                                screeningLength=
                                myScreeningLength) * np.
                                conjugate(FeOrbitals.
                                hydrogenicOrbitals3d("Yz", *
                                Mathematics.cartesianToSpherical
                                (x, y, z))) * np.conjugate(
                                FeOrbitals.hydrogenicOrbitals3d(
                                "z2", *Mathematics.
                                cartesianToSpherical(u, v, w)))
                                * FeOrbitals.
                                hydrogenicOrbitals3d("Yz", *
                                Mathematics.cartesianToSpherical
                                (u, v, w)) * FeOrbitals.
                                hydrogenicOrbitals3d("z2", *
                                Mathematics.cartesianToSpherical
                                (x, y, z))
J_dXz_dXz_func = lambda x, y, z, u, v, w: 2 *
                                potentialEnergyDoubleOccupancy(x
                                , y, z, u, v, w, zeta=myZeta,
                                screeningLength=
                                myScreeningLength) * np.
                                conjugate(FeOrbitals.
                                hydrogenicOrbitals3d("Xz", *
                                Mathematics.cartesianToSpherical
                                (x, y, z))) * np.conjugate(
                                FeOrbitals.hydrogenicOrbitals3d(
                                "Xz", *Mathematics.
                                cartesianToSpherical(u, v, w)))
                                * FeOrbitals.
                                hydrogenicOrbitals3d("Xz", *
                                Mathematics.cartesianToSpherical
                                (u, v, w)) * FeOrbitals.
                                hydrogenicOrbitals3d("Xz", *
                                Mathematics.cartesianToSpherical
                                (x, y, z))
```

```
J_dXz_dYz_func = lambda x, y, z, u, v, w: 2 *
                                potentialEnergyDoubleOccupancy(x
                                , y, z, u, v, w, zeta=myZeta,
                                screeningLength=
                                myScreeningLength) * np.
                                conjugate(FeOrbitals.
                                hydrogenicOrbitals3d("Xz", *
                                Mathematics.cartesianToSpherical
                                (x, y, z))) * np.conjugate(
                                FeOrbitals.hydrogenicOrbitals3d(
                                "Yz", *Mathematics.
                                cartesianToSpherical(u, v, w)))
                                * FeOrbitals.
                                hydrogenicOrbitals3d("Xz", *
                                Mathematics.cartesianToSpherical
                                (u, v, w)) * FeOrbitals.
                                hydrogenicOrbitals3d("Yz", *
                                Mathematics.cartesianToSpherical
                                (x, y, z))
J_dXz_dx2y2_func = lambda x, y, z, u, v, w: 2 *
                                potentialEnergyDoubleOccupancy(x
                                , y, z, u, v, w, zeta=myZeta,
                                screeningLength=
                                myScreeningLength) * np.
                                conjugate(FeOrbitals.
                                hydrogenicOrbitals3d("Xz", *
                                Mathematics.cartesianToSpherical
                                (x, y, z))) * np.conjugate(
                                FeOrbitals.hydrogenicOrbitals3d(
                                "x2-y2", *Mathematics.
                                cartesianToSpherical(u, v, w)))
                                * FeOrbitals.
                                hydrogenicOrbitals3d("Xz", *
                                Mathematics.cartesianToSpherical
                                (u, v, w)) * FeOrbitals.
                                hydrogenicOrbitals3d("x2-y2", *
                                Mathematics.cartesianToSpherical
                                (x, y, z))
J_dXz_dz2_func = lambda x, y, z, u, v, w: 2 *
                                potentialEnergyDoubleOccupancy(x
                                , y, z, u, v, w, zeta=myZeta,
                                screeningLength=
                                myScreeningLength) * np.
                                conjugate(FeOrbitals.
                                hydrogenicOrbitals3d("Xz", *
                                Mathematics.cartesianToSpherical
                                (x, y, z))) * np.conjugate(
                                FeOrbitals.hydrogenicOrbitals3d(
                                "z2", *Mathematics.
                                cartesianToSpherical(u, v, w)))
                                * FeOrbitals.
                                hydrogenicOrbitals3d("Xz", *
                                Mathematics.cartesianToSpherical
                                (u, v, w)) * FeOrbitals.
                                hydrogenicOrbitals3d("z2", *
                                Mathematics.cartesianToSpherical
                                (x, y, z))
```

```
J_dxy_dxy_func = lambda x, y, z, u, v, w: 2 *
                                potentialEnergyDoubleOccupancy(x
                                , y, z, u, v, w, zeta=myZeta,
                                screeningLength=
                                myScreeningLength) * np.
                                conjugate(FeOrbitals.
                                hydrogenicOrbitals3d("xy", *
                                Mathematics.cartesianToSpherical
                                (x, y, z))) * np.conjugate(
                                FeOrbitals.hydrogenicOrbitals3d(
                                "xy", *Mathematics.
                                cartesianToSpherical(u, v, w)))
                                * FeOrbitals.
                                hydrogenicOrbitals3d("xy", *
                                Mathematics.cartesianToSpherical
                                (u, v, w)) * FeOrbitals.
                                hydrogenicOrbitals3d("xy", *
                                Mathematics.cartesianToSpherical
                                (x, y, z))
J_dxy_dXz_func = lambda x, y, z, u, v, w: 2 *
                                potentialEnergyDoubleOccupancy(x
                                , y, z, u, v, w, zeta=myZeta,
                                screeningLength=
                                myScreeningLength) * np.
                                conjugate(FeOrbitals.
                                hydrogenicOrbitals3d("xy", *
                                Mathematics.cartesianToSpherical
                                (x, y, z))) * np.conjugate(
                                FeOrbitals.hydrogenicOrbitals3d(
                                "Xz", *Mathematics.
                                cartesianToSpherical(u, v, w)))
                                * FeOrbitals.
                                hydrogenicOrbitals3d("xy", *
                                Mathematics.cartesianToSpherical
                                (u, v, w)) * FeOrbitals.
                                hydrogenicOrbitals3d("Xz", *
                                Mathematics.cartesianToSpherical
                                (x, y, z))
J_dxy_dYz_func = lambda x, y, z, u, v, w: 2 *
                                potentialEnergyDoubleOccupancy(x
                                , y, z, u, v, w, zeta=myZeta,
                                screeningLength=
                                myScreeningLength) * np.
                                conjugate(FeOrbitals.
                                hydrogenicOrbitals3d("xy", *
                                Mathematics.cartesianToSpherical
                                (x, y, z))) * np.conjugate(
                                FeOrbitals.hydrogenicOrbitals3d(
                                "Yz", *Mathematics.
                                cartesianToSpherical(u, v, w)))
                                * FeOrbitals.
                                hydrogenicOrbitals3d("xy", *
                                Mathematics.cartesianToSpherical
                                (u, v, w)) * FeOrbitals.
                                hydrogenicOrbitals3d("Yz", *
                                Mathematics.cartesianToSpherical
                                (x, y, z))
```

```
J_dxy_dx2y2_func = lambda x, y, z, u, v, w: 2 *
                                  potentialEnergyDoubleOccupancy(x
                                  , y, z, u, v, w, zeta=myZeta,
                                  screeningLength=
                                  myScreeningLength) * np.
                                  conjugate(FeOrbitals.
                                  hydrogenicOrbitals3d("xy", *
                                  Mathematics.cartesianToSpherical
                                  (x, y, z))) * np.conjugate(
                                  FeOrbitals.hydrogenicOrbitals3d(
                                  "x2-y2", *Mathematics.
                                  cartesianToSpherical(u, v, w)))
                                  * FeOrbitals.
                                  hydrogenicOrbitals3d("xy", *
                                  Mathematics.cartesianToSpherical
                                  (u, v, w)) * FeOrbitals.
                                  hydrogenicOrbitals3d("x2-y2", *
                                  Mathematics.cartesianToSpherical
                                  (x, y, z))
J_dxy_dz2_func = lambda x, y, z, u, v, w: 2 *
                                  potentialEnergyDoubleOccupancy(x
                                  , y, z, u, v, w, zeta=myZeta,
                                  screeningLength=
                                  myScreeningLength) * np.
                                  conjugate(FeOrbitals.
                                  hydrogenicOrbitals3d("xy", *
                                  Mathematics.cartesianToSpherical
                                  (x, y, z))) * np.conjugate(
                                  FeOrbitals.hydrogenicOrbitals3d(
                                  "z2", *Mathematics.
                                  cartesianToSpherical(u, v, w)))
                                  * FeOrbitals.
                                  hydrogenicOrbitals3d("xy", *
                                  Mathematics.cartesianToSpherical
                                  (u, v, w)) * FeOrbitals.
                                  hydrogenicOrbitals3d("z2", *
                                  Mathematics.cartesianToSpherical
                                  (x, y, z))

t_1_px_dxy_func = lambda x, y, z: TeOrbitals.
                                  hydrogenicOrbitals5p("x", *
                                  Mathematics.cartesianToSpherical
                                  (x - FeTeLattice[0][0], y -
                                  FeTeLattice[0][1], z -
                                  FeTeLattice[0][2])) *
                                  effectiveCrystalField(x, y, z,
                                  FeTeLattice[0], 2, -2, myZeta,
                                  myScreeningLength) * FeOrbitals.
                                  hydrogenicOrbitals3d("xy", *
                                  Mathematics.cartesianToSpherical
                                  (x, y, z))
```

```
t_1_px_dXz_func = lambda x, y, z: TeOrbitals.
                                hydrogenicOrbitals5p("x", *
                                Mathematics.cartesianToSpherical
                                (x - FeTeLattice[0][0], y -
                                FeTeLattice[0][1], z -
                                FeTeLattice[0][2])) *
                                effectiveCrystalField(x, y, z,
                                FeTeLattice[0], 2, -2, myZeta,
                                myScreeningLength) * FeOrbitals.
                                hydrogenicOrbitals3d("Xz", *
                                Mathematics.cartesianToSpherical
                                (x, y, z))
t_1_px_dYz_func = lambda x, y, z: TeOrbitals.
                                hydrogenicOrbitals5p("x", *
                                Mathematics.cartesianToSpherical
                                (x - FeTeLattice[0][0], y -
                                FeTeLattice[0][1], z -
                                FeTeLattice[0][2])) *
                                effectiveCrystalField(x, y, z,
                                FeTeLattice[0], 2, -2, myZeta,
                                myScreeningLength) * FeOrbitals.
                                hydrogenicOrbitals3d("Yz", *
                                Mathematics.cartesianToSpherical
                                (x, y, z))
t_1_py_dxy_func = lambda x, y, z: TeOrbitals.
                                hydrogenicOrbitals5p("y", *
                                Mathematics.cartesianToSpherical
                                (x - FeTeLattice[0][0], y -
                                FeTeLattice[0][1], z -
                                FeTeLattice[0][2])) *
                                effectiveCrystalField(x, y, z,
                                FeTeLattice[0], 2, -2, myZeta,
                                myScreeningLength) * FeOrbitals.
                                hydrogenicOrbitals3d("xy", *
                                Mathematics.cartesianToSpherical
                                (x, y, z))
t_1_py_dXz_func = lambda x, y, z: TeOrbitals.
                                hydrogenicOrbitals5p("y", *
                                Mathematics.cartesianToSpherical
                                (x - FeTeLattice[0][0], y -
                                FeTeLattice[0][1], z -
                                FeTeLattice[0][2])) *
                                effectiveCrystalField(x, y, z,
                                FeTeLattice[0], 2, -2, myZeta,
                                myScreeningLength) * FeOrbitals.
                                hydrogenicOrbitals3d("Xz", *
                                Mathematics.cartesianToSpherical
                                (x, y, z))
```

```
t_1_py_dYz_func = lambda x, y, z: TeOrbitals.
                                  hydrogenicOrbitals5p("y", *
                                  Mathematics.cartesianToSpherical
                                  (x - FeTeLattice[0][0], y -
                                  FeTeLattice[0][1], z -
                                  FeTeLattice[0][2])) *
                                  effectiveCrystalField(x, y, z,
                                  FeTeLattice[0], 2, -2, myZeta,
                                  myScreeningLength) * FeOrbitals.
                                  hydrogenicOrbitals3d("Yz", *
                                  Mathematics.cartesianToSpherical
                                  (x, y, z))
t_1_pz_dxy_func = lambda x, y, z: TeOrbitals.
                                  hydrogenicOrbitals5p("z", *
                                  Mathematics.cartesianToSpherical
                                  (x - FeTeLattice[0][0], y -
                                  FeTeLattice[0][1], z -
                                  FeTeLattice[0][2])) *
                                  effectiveCrystalField(x, y, z,
                                  FeTeLattice[0], 2, -2, myZeta,
                                  myScreeningLength) * FeOrbitals.
                                  hydrogenicOrbitals3d("xy", *
                                  Mathematics.cartesianToSpherical
                                  (x, y, z))
t_1_pz_dXz_func = lambda x, y, z: TeOrbitals.
                                  hydrogenicOrbitals5p("z", *
                                  Mathematics.cartesianToSpherical
                                  (x - FeTeLattice[0][0], y -
                                  FeTeLattice[0][1], z -
                                  FeTeLattice[0][2])) *
                                  effectiveCrystalField(x, y, z,
                                  FeTeLattice[0], 2, -2, myZeta,
                                  myScreeningLength) * FeOrbitals.
                                  hydrogenicOrbitals3d("Xz", *
                                  Mathematics.cartesianToSpherical
                                  (x, y, z))
t_1_pz_dYz_func = lambda x, y, z: TeOrbitals.
                                  hydrogenicOrbitals5p("z", *
                                  Mathematics.cartesianToSpherical
                                  (x - FeTeLattice[0][0], y -
                                  FeTeLattice[0][1], z -
                                  FeTeLattice[0][2])) *
                                  effectiveCrystalField(x, y, z,
                                  FeTeLattice[0], 2, -2, myZeta,
                                  myScreeningLength) * FeOrbitals.
                                  hydrogenicOrbitals3d("Yz", *
                                  Mathematics.cartesianToSpherical
                                  (x, y, z))
```

```
t_2_px_dxy_func = lambda x, y, z: TeOrbitals.
                                hydrogenicOrbitals5p("x", *
                                Mathematics.cartesianToSpherical
                                (x - FeTeLattice[1][0], y -
                                FeTeLattice[1][1], z -
                                FeTeLattice[1][2])) *
                                effectiveCrystalField(x, y, z,
                                FeTeLattice[1], 2, -2, myZeta,
                                myScreeningLength) * FeOrbitals.
                                hydrogenicOrbitals3d("xy", *
                                Mathematics.cartesianToSpherical
                                (x, y, z))
t_2_px_dXz_func = lambda x, y, z: TeOrbitals.
                                hydrogenicOrbitals5p("x", *
                                Mathematics.cartesianToSpherical
                                (x - FeTeLattice[1][0], y -
                                FeTeLattice[1][1], z -
                                FeTeLattice[1][2])) *
                                effectiveCrystalField(x, y, z,
                                FeTeLattice[1], 2, -2, myZeta,
                                myScreeningLength) * FeOrbitals.
                                hydrogenicOrbitals3d("Xz", *
                                Mathematics.cartesianToSpherical
                                (x, y, z))
t_2_px_dYz_func = lambda x, y, z: TeOrbitals.
                                hydrogenicOrbitals5p("x", *
                                Mathematics.cartesianToSpherical
                                (x - FeTeLattice[1][0], y -
                                FeTeLattice[1][1], z -
                                FeTeLattice[1][2])) *
                                effectiveCrystalField(x, y, z,
                                FeTeLattice[1], 2, -2, myZeta,
                                myScreeningLength) * FeOrbitals.
                                hydrogenicOrbitals3d("Yz", *
                                Mathematics.cartesianToSpherical
                                (x, y, z))
t_2_py_dxy_func = lambda x, y, z: TeOrbitals.
                                hydrogenicOrbitals5p("y", *
                                Mathematics.cartesianToSpherical
                                (x - FeTeLattice[1][0], y -
                                FeTeLattice[1][1], z -
                                FeTeLattice[1][2])) *
                                effectiveCrystalField(x, y, z,
                                FeTeLattice[1], 2, -2, myZeta,
                                myScreeningLength) * FeOrbitals.
                                hydrogenicOrbitals3d("xy", *
                                Mathematics.cartesianToSpherical
                                (x, y, z))
```

```
t_2_py_dXz_func = lambda x, y, z: TeOrbitals.
                                hydrogenicOrbitals5p("y", *
                                Mathematics.cartesianToSpherical
                                (x - FeTeLattice[1][0], y -
                                FeTeLattice[1][1], z -
                                FeTeLattice[1][2])) *
                                effectiveCrystalField(x, y, z,
                                FeTeLattice[1], 2, -2, myZeta,
                                myScreeningLength) * FeOrbitals.
                                hydrogenicOrbitals3d("Xz", *
                                Mathematics.cartesianToSpherical
                                (x, y, z))
t_2_py_dYz_func = lambda x, y, z: TeOrbitals.
                                hydrogenicOrbitals5p("y", *
                                Mathematics.cartesianToSpherical
                                (x - FeTeLattice[1][0], y -
                                FeTeLattice[1][1], z -
                                FeTeLattice[1][2])) *
                                effectiveCrystalField(x, y, z,
                                FeTeLattice[1], 2, -2, myZeta,
                                myScreeningLength) * FeOrbitals.
                                hydrogenicOrbitals3d("Yz", *
                                Mathematics.cartesianToSpherical
                                (x, y, z))
t_2_pz_dxy_func = lambda x, y, z: TeOrbitals.
                                hydrogenicOrbitals5p("z", *
                                Mathematics.cartesianToSpherical
                                (x - FeTeLattice[1][0], y -
                                FeTeLattice[1][1], z -
                                FeTeLattice[1][2])) *
                                effectiveCrystalField(x, y, z,
                                FeTeLattice[1], 2, -2, myZeta,
                                myScreeningLength) * FeOrbitals.
                                hydrogenicOrbitals3d("xy", *
                                Mathematics.cartesianToSpherical
                                (x, y, z))
t_2_pz_dXz_func = lambda x, y, z: TeOrbitals.
                                hydrogenicOrbitals5p("z", *
                                Mathematics.cartesianToSpherical
                                (x - FeTeLattice[1][0], y -
                                FeTeLattice[1][1], z -
                                FeTeLattice[1][2])) *
                                effectiveCrystalField(x, y, z,
                                FeTeLattice[1], 2, -2, myZeta,
                                myScreeningLength) * FeOrbitals.
                                hydrogenicOrbitals3d("Xz", *
                                Mathematics.cartesianToSpherical
                                (x, y, z))
```

```
t_2_pz_dYz_func = lambda x, y, z: TeOrbitals.
                                hydrogenicOrbitals5p("z", *
                                Mathematics.cartesianToSpherical
                                (x - FeTeLattice[1][0], y -
                                FeTeLattice[1][1], z -
                                FeTeLattice[1][2])) *
                                effectiveCrystalField(x, y, z,
                                FeTeLattice[1], 2, -2, myZeta,
                                myScreeningLength) * FeOrbitals.
                                hydrogenicOrbitals3d("Yz", *
                                Mathematics.cartesianToSpherical
                                (x, y, z))

t_3_px_dxy_func = lambda x, y, z: TeOrbitals.
                                hydrogenicOrbitals5p("x", *
                                Mathematics.cartesianToSpherical
                                (x - FeTeLattice[2][0], y -
                                FeTeLattice[2][1], z -
                                FeTeLattice[2][2])) *
                                effectiveCrystalField(x, y, z,
                                FeTeLattice[2], 2, -2, myZeta,
                                myScreeningLength) * FeOrbitals.
                                hydrogenicOrbitals3d("xy", *
                                Mathematics.cartesianToSpherical
                                (x, y, z))
t_3_px_dXz_func = lambda x, y, z: TeOrbitals.
                                hydrogenicOrbitals5p("x", *
                                Mathematics.cartesianToSpherical
                                (x - FeTeLattice[2][0], y -
                                FeTeLattice[2][1], z -
                                FeTeLattice[2][2])) *
                                effectiveCrystalField(x, y, z,
                                FeTeLattice[2], 2, -2, myZeta,
                                myScreeningLength) * FeOrbitals.
                                hydrogenicOrbitals3d("Xz", *
                                Mathematics.cartesianToSpherical
                                (x, y, z))
t_3_px_dYz_func = lambda x, y, z: TeOrbitals.
                                hydrogenicOrbitals5p("x", *
                                Mathematics.cartesianToSpherical
                                (x - FeTeLattice[2][0], y -
                                FeTeLattice[2][1], z -
                                FeTeLattice[2][2])) *
                                effectiveCrystalField(x, y, z,
                                FeTeLattice[2], 2, -2, myZeta,
                                myScreeningLength) * FeOrbitals.
                                hydrogenicOrbitals3d("Yz", *
                                Mathematics.cartesianToSpherical
                                (x, y, z))
```

```
t_3_py_dxy_func = lambda x, y, z: TeOrbitals.
                                   hydrogenicOrbitals5p("y", *
                                   Mathematics.cartesianToSpherical
                                   (x - FeTeLattice[2][0], y -
                                   FeTeLattice[2][1], z -
                                   FeTeLattice[2][2])) *
                                   effectiveCrystalField(x, y, z,
                                   FeTeLattice[2], 2, -2, myZeta,
                                   myScreeningLength) * FeOrbitals.
                                   hydrogenicOrbitals3d("xy", *
                                   Mathematics.cartesianToSpherical
                                   (x, y, z))
t_3_py_dXz_func = lambda x, y, z: TeOrbitals.
                                   hydrogenicOrbitals5p("y", *
                                   Mathematics.cartesianToSpherical
                                   (x - FeTeLattice[2][0], y -
                                   FeTeLattice[2][1], z -
                                   FeTeLattice[2][2])) *
                                   effectiveCrystalField(x, y, z,
                                   FeTeLattice[2], 2, -2, myZeta,
                                   myScreeningLength) * FeOrbitals.
                                   hydrogenicOrbitals3d("Xz", *
                                   Mathematics.cartesianToSpherical
                                   (x, y, z))
t_3_py_dYz_func = lambda x, y, z: TeOrbitals.
                                   hydrogenicOrbitals5p("y", *
                                   Mathematics.cartesianToSpherical
                                   (x - FeTeLattice[2][0], y -
                                   FeTeLattice[2][1], z -
                                   FeTeLattice[2][2])) *
                                   effectiveCrystalField(x, y, z,
                                   FeTeLattice[2], 2, -2, myZeta,
                                   myScreeningLength) * FeOrbitals.
                                   hydrogenicOrbitals3d("Yz", *
                                   Mathematics.cartesianToSpherical
                                   (x, y, z))
t_3_pz_dxy_func = lambda x, y, z: TeOrbitals.
                                   hydrogenicOrbitals5p("z", *
                                   Mathematics.cartesianToSpherical
                                   (x - FeTeLattice[2][0], y -
                                   FeTeLattice[2][1], z -
                                   FeTeLattice[2][2])) *
                                   effectiveCrystalField(x, y, z,
                                   FeTeLattice[2], 2, -2, myZeta,
                                   myScreeningLength) * FeOrbitals.
                                   hydrogenicOrbitals3d("xy", *
                                   Mathematics.cartesianToSpherical
                                   (x, y, z))
```

```
t_3_pz_dXz_func = lambda x, y, z: TeOrbitals.
                                 hydrogenicOrbitals5p("z", *
                                 Mathematics.cartesianToSpherical
                                 (x - FeTeLattice[2][0], y -
                                 FeTeLattice[2][1], z -
                                 FeTeLattice[2][2])) *
                                 effectiveCrystalField(x, y, z,
                                 FeTeLattice[2], 2, -2, myZeta,
                                 myScreeningLength) * FeOrbitals.
                                 hydrogenicOrbitals3d("Xz", *
                                 Mathematics.cartesianToSpherical
                                 (x, y, z))
t_3_pz_dYz_func = lambda x, y, z: TeOrbitals.
                                 hydrogenicOrbitals5p("z", *
                                 Mathematics.cartesianToSpherical
                                 (x - FeTeLattice[2][0], y -
                                 FeTeLattice[2][1], z -
                                 FeTeLattice[2][2])) *
                                 effectiveCrystalField(x, y, z,
                                 FeTeLattice[2], 2, -2, myZeta,
                                 myScreeningLength) * FeOrbitals.
                                 hydrogenicOrbitals3d("Yz", *
                                 Mathematics.cartesianToSpherical
                                 (x, y, z))

t_4_px_dxy_func = lambda x, y, z: TeOrbitals.
                                 hydrogenicOrbitals5p("x", *
                                 Mathematics.cartesianToSpherical
                                 (x - FeTeLattice[3][0], y -
                                 FeTeLattice[3][1], z -
                                 FeTeLattice[3][2])) *
                                 effectiveCrystalField(x, y, z,
                                 FeTeLattice[3], 2, -2, myZeta,
                                 myScreeningLength) * FeOrbitals.
                                 hydrogenicOrbitals3d("xy", *
                                 Mathematics.cartesianToSpherical
                                 (x, y, z))
t_4_px_dXz_func = lambda x, y, z: TeOrbitals.
                                 hydrogenicOrbitals5p("x", *
                                 Mathematics.cartesianToSpherical
                                 (x - FeTeLattice[3][0], y -
                                 FeTeLattice[3][1], z -
                                 FeTeLattice[3][2])) *
                                 effectiveCrystalField(x, y, z,
                                 FeTeLattice[3], 2, -2, myZeta,
                                 myScreeningLength) * FeOrbitals.
                                 hydrogenicOrbitals3d("Xz", *
                                 Mathematics.cartesianToSpherical
                                 (x, y, z))
```

```
t_4_px_dYz_func = lambda x, y, z: TeOrbitals.
                                  hydrogenicOrbitals5p("x", *
                                  Mathematics.cartesianToSpherical
                                  (x - FeTeLattice[3][0], y -
                                  FeTeLattice[3][1], z -
                                  FeTeLattice[3][2])) *
                                  effectiveCrystalField(x, y, z,
                                  FeTeLattice[3], 2, -2, myZeta,
                                  myScreeningLength) * FeOrbitals.
                                  hydrogenicOrbitals3d("Yz", *
                                  Mathematics.cartesianToSpherical
                                  (x, y, z))
t_4_py_dxy_func = lambda x, y, z: TeOrbitals.
                                  hydrogenicOrbitals5p("y", *
                                  Mathematics.cartesianToSpherical
                                  (x - FeTeLattice[3][0], y -
                                  FeTeLattice[3][1], z -
                                  FeTeLattice[3][2])) *
                                  effectiveCrystalField(x, y, z,
                                  FeTeLattice[3], 2, -2, myZeta,
                                  myScreeningLength) * FeOrbitals.
                                  hydrogenicOrbitals3d("xy", *
                                  Mathematics.cartesianToSpherical
                                  (x, y, z))
t_4_py_dXz_func = lambda x, y, z: TeOrbitals.
                                  hydrogenicOrbitals5p("y", *
                                  Mathematics.cartesianToSpherical
                                  (x - FeTeLattice[3][0], y -
                                  FeTeLattice[3][1], z -
                                  FeTeLattice[3][2])) *
                                  effectiveCrystalField(x, y, z,
                                  FeTeLattice[3], 2, -2, myZeta,
                                  myScreeningLength) * FeOrbitals.
                                  hydrogenicOrbitals3d("Xz", *
                                  Mathematics.cartesianToSpherical
                                  (x, y, z))
t_4_py_dYz_func = lambda x, y, z: TeOrbitals.
                                  hydrogenicOrbitals5p("y", *
                                  Mathematics.cartesianToSpherical
                                  (x - FeTeLattice[3][0], y -
                                  FeTeLattice[3][1], z -
                                  FeTeLattice[3][2])) *
                                  effectiveCrystalField(x, y, z,
                                  FeTeLattice[3], 2, -2, myZeta,
                                  myScreeningLength) * FeOrbitals.
                                  hydrogenicOrbitals3d("Yz", *
                                  Mathematics.cartesianToSpherical
                                  (x, y, z))
```

```
t_4_pz_dxy_func = lambda x, y, z: TeOrbitals.
                                hydrogenicOrbitals5p("z", *
                                Mathematics.cartesianToSpherical
                                (x - FeTeLattice[3][0], y -
                                FeTeLattice[3][1], z -
                                FeTeLattice[3][2])) *
                                effectiveCrystalField(x, y, z,
                                FeTeLattice[3], 2, -2, myZeta,
                                myScreeningLength) * FeOrbitals.
                                hydrogenicOrbitals3d("xy", *
                                Mathematics.cartesianToSpherical
                                (x, y, z))
t_4_pz_dXz_func = lambda x, y, z: TeOrbitals.
                                hydrogenicOrbitals5p("z", *
                                Mathematics.cartesianToSpherical
                                (x - FeTeLattice[3][0], y -
                                FeTeLattice[3][1], z -
                                FeTeLattice[3][2])) *
                                effectiveCrystalField(x, y, z,
                                FeTeLattice[3], 2, -2, myZeta,
                                myScreeningLength) * FeOrbitals.
                                hydrogenicOrbitals3d("Xz", *
                                Mathematics.cartesianToSpherical
                                (x, y, z))
t_4_pz_dYz_func = lambda x, y, z: TeOrbitals.
                                hydrogenicOrbitals5p("z", *
                                Mathematics.cartesianToSpherical
                                (x - FeTeLattice[3][0], y -
                                FeTeLattice[3][1], z -
                                FeTeLattice[3][2])) *
                                effectiveCrystalField(x, y, z,
                                FeTeLattice[3], 2, -2, myZeta,
                                myScreeningLength) * FeOrbitals.
                                hydrogenicOrbitals3d("Yz", *
                                Mathematics.cartesianToSpherical
                                (x, y, z))

t_5_px_dxy_func = lambda x, y, z: TeOrbitals.
                                hydrogenicOrbitals5p("x", *
                                Mathematics.cartesianToSpherical
                                (x - FeTeLattice[0][0], y -
                                FeTeLattice[0][1], z +
                                FeTeLattice[0][2])) *
                                effectiveCrystalField(x, y, z,
                                mirror_in_xy_plane(FeTeLattice)[
                                0], 2, -2, myZeta,
                                myScreeningLength) * FeOrbitals.
                                hydrogenicOrbitals3d("xy", *
                                Mathematics.cartesianToSpherical
                                (x, y, z))
```

```
t_5_px_dXz_func = lambda x, y, z: TeOrbitals.
                                hydrogenicOrbitals5p("x", *
                                Mathematics.cartesianToSpherical
                                (x - FeTeLattice[0][0], y -
                                FeTeLattice[0][1], z +
                                FeTeLattice[0][2])) *
                                effectiveCrystalField(x, y, z,
                                mirror_in_xy_plane(FeTeLattice)[
                                0], 2, -2, myZeta,
                                myScreeningLength) * FeOrbitals.
                                hydrogenicOrbitals3d("Xz", *
                                Mathematics.cartesianToSpherical
                                (x, y, z))
t_5_px_dYz_func = lambda x, y, z: TeOrbitals.
                                hydrogenicOrbitals5p("x", *
                                Mathematics.cartesianToSpherical
                                (x - FeTeLattice[0][0], y -
                                FeTeLattice[0][1], z +
                                FeTeLattice[0][2])) *
                                effectiveCrystalField(x, y, z,
                                mirror_in_xy_plane(FeTeLattice)[
                                0], 2, -2, myZeta,
                                myScreeningLength) * FeOrbitals.
                                hydrogenicOrbitals3d("Yz", *
                                Mathematics.cartesianToSpherical
                                (x, y, z))
t_5_py_dxy_func = lambda x, y, z: TeOrbitals.
                                hydrogenicOrbitals5p("y", *
                                Mathematics.cartesianToSpherical
                                (x - FeTeLattice[0][0], y -
                                FeTeLattice[0][1], z +
                                FeTeLattice[0][2])) *
                                effectiveCrystalField(x, y, z,
                                mirror_in_xy_plane(FeTeLattice)[
                                0], 2, -2, myZeta,
                                myScreeningLength) * FeOrbitals.
                                hydrogenicOrbitals3d("xy", *
                                Mathematics.cartesianToSpherical
                                (x, y, z))
t_5_py_dXz_func = lambda x, y, z: TeOrbitals.
                                hydrogenicOrbitals5p("y", *
                                Mathematics.cartesianToSpherical
                                (x - FeTeLattice[0][0], y -
                                FeTeLattice[0][1], z +
                                FeTeLattice[0][2])) *
                                effectiveCrystalField(x, y, z,
                                mirror_in_xy_plane(FeTeLattice)[
                                0], 2, -2, myZeta,
                                myScreeningLength) * FeOrbitals.
                                hydrogenicOrbitals3d("Xz", *
                                Mathematics.cartesianToSpherical
                                (x, y, z))
```

```
t_5_py_dYz_func = lambda x, y, z: TeOrbitals.
                                hydrogenicOrbitals5p("y", *
                                Mathematics.cartesianToSpherical
                                (x - FeTeLattice[0][0], y -
                                FeTeLattice[0][1], z +
                                FeTeLattice[0][2])) *
                                effectiveCrystalField(x, y, z,
                                mirror_in_xy_plane(FeTeLattice)[
                                0], 2, -2, myZeta,
                                myScreeningLength) * FeOrbitals.
                                hydrogenicOrbitals3d("Yz", *
                                Mathematics.cartesianToSpherical
                                (x, y, z))
t_5_pz_dxy_func = lambda x, y, z: TeOrbitals.
                                hydrogenicOrbitals5p("z", *
                                Mathematics.cartesianToSpherical
                                (x - FeTeLattice[0][0], y -
                                FeTeLattice[0][1], z +
                                FeTeLattice[0][2])) *
                                effectiveCrystalField(x, y, z,
                                mirror_in_xy_plane(FeTeLattice)[
                                0], 2, -2, myZeta,
                                myScreeningLength) * FeOrbitals.
                                hydrogenicOrbitals3d("xy", *
                                Mathematics.cartesianToSpherical
                                (x, y, z))
t_5_pz_dXz_func = lambda x, y, z: TeOrbitals.
                                hydrogenicOrbitals5p("z", *
                                Mathematics.cartesianToSpherical
                                (x - FeTeLattice[0][0], y -
                                FeTeLattice[0][1], z +
                                FeTeLattice[0][2])) *
                                effectiveCrystalField(x, y, z,
                                mirror_in_xy_plane(FeTeLattice)[
                                0], 2, -2, myZeta,
                                myScreeningLength) * FeOrbitals.
                                hydrogenicOrbitals3d("Xz", *
                                Mathematics.cartesianToSpherical
                                (x, y, z))
t_5_pz_dYz_func = lambda x, y, z: TeOrbitals.
                                hydrogenicOrbitals5p("z", *
                                Mathematics.cartesianToSpherical
                                (x - FeTeLattice[0][0], y -
                                FeTeLattice[0][1], z +
                                FeTeLattice[0][2])) *
                                effectiveCrystalField(x, y, z,
                                mirror_in_xy_plane(FeTeLattice)[
                                0], 2, -2, myZeta,
                                myScreeningLength) * FeOrbitals.
                                hydrogenicOrbitals3d("Yz", *
                                Mathematics.cartesianToSpherical
                                (x, y, z))
```

```
t_6_px_dxy_func = lambda x, y, z: TeOrbitals.
                                hydrogenicOrbitals5p("x", *
                                Mathematics.cartesianToSpherical
                                (x - FeTeLattice[1][0], y -
                                FeTeLattice[1][1], z +
                                FeTeLattice[1][2])) *
                                effectiveCrystalField(x, y, z,
                                mirror_in_xy_plane(FeTeLattice)[
                                1], 2, -2, myZeta,
                                myScreeningLength) * FeOrbitals.
                                hydrogenicOrbitals3d("xy", *
                                Mathematics.cartesianToSpherical
                                (x, y, z))
t_6_px_dXz_func = lambda x, y, z: TeOrbitals.
                                hydrogenicOrbitals5p("x", *
                                Mathematics.cartesianToSpherical
                                (x - FeTeLattice[1][0], y -
                                FeTeLattice[1][1], z +
                                FeTeLattice[1][2])) *
                                effectiveCrystalField(x, y, z,
                                mirror_in_xy_plane(FeTeLattice)[
                                1], 2, -2, myZeta,
                                myScreeningLength) * FeOrbitals.
                                hydrogenicOrbitals3d("Xz", *
                                Mathematics.cartesianToSpherical
                                (x, y, z))
t_6_px_dYz_func = lambda x, y, z: TeOrbitals.
                                hydrogenicOrbitals5p("x", *
                                Mathematics.cartesianToSpherical
                                (x - FeTeLattice[1][0], y -
                                FeTeLattice[1][1], z +
                                FeTeLattice[1][2])) *
                                effectiveCrystalField(x, y, z,
                                mirror_in_xy_plane(FeTeLattice)[
                                1], 2, -2, myZeta,
                                myScreeningLength) * FeOrbitals.
                                hydrogenicOrbitals3d("Yz", *
                                Mathematics.cartesianToSpherical
                                (x, y, z))
t_6_py_dxy_func = lambda x, y, z: TeOrbitals.
                                hydrogenicOrbitals5p("y", *
                                Mathematics.cartesianToSpherical
                                (x - FeTeLattice[1][0], y -
                                FeTeLattice[1][1], z +
                                FeTeLattice[1][2])) *
                                effectiveCrystalField(x, y, z,
                                mirror_in_xy_plane(FeTeLattice)[
                                1], 2, -2, myZeta,
                                myScreeningLength) * FeOrbitals.
                                hydrogenicOrbitals3d("xy", *
                                Mathematics.cartesianToSpherical
                                (x, y, z))
```

```
t_6_py_dXz_func = lambda x, y, z: TeOrbitals.
                                hydrogenicOrbitals5p("y", *
                                Mathematics.cartesianToSpherical
                                (x - FeTeLattice[1][0], y -
                                FeTeLattice[1][1], z +
                                FeTeLattice[1][2])) *
                                effectiveCrystalField(x, y, z,
                                mirror_in_xy_plane(FeTeLattice)[
                                1], 2, -2, myZeta,
                                myScreeningLength) * FeOrbitals.
                                hydrogenicOrbitals3d("Xz", *
                                Mathematics.cartesianToSpherical
                                (x, y, z))
t_6_py_dYz_func = lambda x, y, z: TeOrbitals.
                                hydrogenicOrbitals5p("y", *
                                Mathematics.cartesianToSpherical
                                (x - FeTeLattice[1][0], y -
                                FeTeLattice[1][1], z +
                                FeTeLattice[1][2])) *
                                effectiveCrystalField(x, y, z,
                                mirror_in_xy_plane(FeTeLattice)[
                                1], 2, -2, myZeta,
                                myScreeningLength) * FeOrbitals.
                                hydrogenicOrbitals3d("Yz", *
                                Mathematics.cartesianToSpherical
                                (x, y, z))
t_6_pz_dxy_func = lambda x, y, z: TeOrbitals.
                                hydrogenicOrbitals5p("z", *
                                Mathematics.cartesianToSpherical
                                (x - FeTeLattice[1][0], y -
                                FeTeLattice[1][1], z +
                                FeTeLattice[1][2])) *
                                effectiveCrystalField(x, y, z,
                                mirror_in_xy_plane(FeTeLattice)[
                                1], 2, -2, myZeta,
                                myScreeningLength) * FeOrbitals.
                                hydrogenicOrbitals3d("xy", *
                                Mathematics.cartesianToSpherical
                                (x, y, z))
t_6_pz_dXz_func = lambda x, y, z: TeOrbitals.
                                hydrogenicOrbitals5p("z", *
                                Mathematics.cartesianToSpherical
                                (x - FeTeLattice[1][0], y -
                                FeTeLattice[1][1], z +
                                FeTeLattice[1][2])) *
                                effectiveCrystalField(x, y, z,
                                mirror_in_xy_plane(FeTeLattice)[
                                1], 2, -2, myZeta,
                                myScreeningLength) * FeOrbitals.
                                hydrogenicOrbitals3d("Xz", *
                                Mathematics.cartesianToSpherical
                                (x, y, z))
```

```
t_6_pz_dYz_func = lambda x, y, z: TeOrbitals.
                                hydrogenicOrbitals5p("z", *
                                Mathematics.cartesianToSpherical
                                (x - FeTeLattice[1][0], y -
                                FeTeLattice[1][1], z +
                                FeTeLattice[1][2])) *
                                effectiveCrystalField(x, y, z,
                                mirror_in_xy_plane(FeTeLattice)[
                                1], 2, -2, myZeta,
                                myScreeningLength) * FeOrbitals.
                                hydrogenicOrbitals3d("Yz", *
                                Mathematics.cartesianToSpherical
                                (x, y, z))

t_7_px_dxy_func = lambda x, y, z: TeOrbitals.
                                hydrogenicOrbitals5p("x", *
                                Mathematics.cartesianToSpherical
                                (x - FeTeLattice[2][0], y -
                                FeTeLattice[2][1], z +
                                FeTeLattice[2][2])) *
                                effectiveCrystalField(x, y, z,
                                mirror_in_xy_plane(FeTeLattice)[
                                2], 2, -2, myZeta,
                                myScreeningLength) * FeOrbitals.
                                hydrogenicOrbitals3d("xy", *
                                Mathematics.cartesianToSpherical
                                (x, y, z))
t_7_px_dXz_func = lambda x, y, z: TeOrbitals.
                                hydrogenicOrbitals5p("x", *
                                Mathematics.cartesianToSpherical
                                (x - FeTeLattice[2][0], y -
                                FeTeLattice[2][1], z +
                                FeTeLattice[2][2])) *
                                effectiveCrystalField(x, y, z,
                                mirror_in_xy_plane(FeTeLattice)[
                                2], 2, -2, myZeta,
                                myScreeningLength) * FeOrbitals.
                                hydrogenicOrbitals3d("Xz", *
                                Mathematics.cartesianToSpherical
                                (x, y, z))
t_7_px_dYz_func = lambda x, y, z: TeOrbitals.
                                hydrogenicOrbitals5p("x", *
                                Mathematics.cartesianToSpherical
                                (x - FeTeLattice[2][0], y -
                                FeTeLattice[2][1], z +
                                FeTeLattice[2][2])) *
                                effectiveCrystalField(x, y, z,
                                mirror_in_xy_plane(FeTeLattice)[
                                2], 2, -2, myZeta,
                                myScreeningLength) * FeOrbitals.
                                hydrogenicOrbitals3d("Yz", *
                                Mathematics.cartesianToSpherical
                                (x, y, z))
```

```
t_7_py_dxy_func = lambda x, y, z: TeOrbitals.
                                hydrogenicOrbitals5p("y", *
                                Mathematics.cartesianToSpherical
                                (x - FeTeLattice[2][0], y -
                                FeTeLattice[2][1], z +
                                FeTeLattice[2][2])) *
                                effectiveCrystalField(x, y, z,
                                mirror_in_xy_plane(FeTeLattice)[
                                2], 2, -2, myZeta,
                                myScreeningLength) * FeOrbitals.
                                hydrogenicOrbitals3d("xy", *
                                Mathematics.cartesianToSpherical
                                (x, y, z))
t_7_py_dXz_func = lambda x, y, z: TeOrbitals.
                                hydrogenicOrbitals5p("y", *
                                Mathematics.cartesianToSpherical
                                (x - FeTeLattice[2][0], y -
                                FeTeLattice[2][1], z +
                                FeTeLattice[2][2])) *
                                effectiveCrystalField(x, y, z,
                                mirror_in_xy_plane(FeTeLattice)[
                                2], 2, -2, myZeta,
                                myScreeningLength) * FeOrbitals.
                                hydrogenicOrbitals3d("Xz", *
                                Mathematics.cartesianToSpherical
                                (x, y, z))
t_7_py_dYz_func = lambda x, y, z: TeOrbitals.
                                hydrogenicOrbitals5p("y", *
                                Mathematics.cartesianToSpherical
                                (x - FeTeLattice[2][0], y -
                                FeTeLattice[2][1], z +
                                FeTeLattice[2][2])) *
                                effectiveCrystalField(x, y, z,
                                mirror_in_xy_plane(FeTeLattice)[
                                2], 2, -2, myZeta,
                                myScreeningLength) * FeOrbitals.
                                hydrogenicOrbitals3d("Yz", *
                                Mathematics.cartesianToSpherical
                                (x, y, z))
t_7_pz_dxy_func = lambda x, y, z: TeOrbitals.
                                hydrogenicOrbitals5p("z", *
                                Mathematics.cartesianToSpherical
                                (x - FeTeLattice[2][0], y -
                                FeTeLattice[2][1], z +
                                FeTeLattice[2][2])) *
                                effectiveCrystalField(x, y, z,
                                mirror_in_xy_plane(FeTeLattice)[
                                2], 2, -2, myZeta,
                                myScreeningLength) * FeOrbitals.
                                hydrogenicOrbitals3d("xy", *
                                Mathematics.cartesianToSpherical
                                (x, y, z))
```

```
t_7_pz_dXz_func = lambda x, y, z: TeOrbitals.
                                hydrogenicOrbitals5p("z", *
                                Mathematics.cartesianToSpherical
                                (x - FeTeLattice[2][0], y -
                                FeTeLattice[2][1], z +
                                FeTeLattice[2][2])) *
                                effectiveCrystalField(x, y, z,
                                mirror_in_xy_plane(FeTeLattice)[
                                2], 2, -2, myZeta,
                                myScreeningLength) * FeOrbitals.
                                hydrogenicOrbitals3d("Xz", *
                                Mathematics.cartesianToSpherical
                                (x, y, z))
t_7_pz_dYz_func = lambda x, y, z: TeOrbitals.
                                hydrogenicOrbitals5p("z", *
                                Mathematics.cartesianToSpherical
                                (x - FeTeLattice[2][0], y -
                                FeTeLattice[2][1], z +
                                FeTeLattice[2][2])) *
                                effectiveCrystalField(x, y, z,
                                mirror_in_xy_plane(FeTeLattice)[
                                2], 2, -2, myZeta,
                                myScreeningLength) * FeOrbitals.
                                hydrogenicOrbitals3d("Yz", *
                                Mathematics.cartesianToSpherical
                                (x, y, z))

t_8_px_dxy_func = lambda x, y, z: TeOrbitals.
                                hydrogenicOrbitals5p("x", *
                                Mathematics.cartesianToSpherical
                                (x - FeTeLattice[3][0], y -
                                FeTeLattice[3][1], z +
                                FeTeLattice[3][2])) *
                                effectiveCrystalField(x, y, z,
                                mirror_in_xy_plane(FeTeLattice)[
                                3], 2, -2, myZeta,
                                myScreeningLength) * FeOrbitals.
                                hydrogenicOrbitals3d("xy", *
                                Mathematics.cartesianToSpherical
                                (x, y, z))
t_8_px_dXz_func = lambda x, y, z: TeOrbitals.
                                hydrogenicOrbitals5p("x", *
                                Mathematics.cartesianToSpherical
                                (x - FeTeLattice[3][0], y -
                                FeTeLattice[3][1], z +
                                FeTeLattice[3][2])) *
                                effectiveCrystalField(x, y, z,
                                mirror_in_xy_plane(FeTeLattice)[
                                3], 2, -2, myZeta,
                                myScreeningLength) * FeOrbitals.
                                hydrogenicOrbitals3d("Xz", *
                                Mathematics.cartesianToSpherical
                                (x, y, z))
```

```
t_8_px_dYz_func = lambda x, y, z: TeOrbitals.
                                hydrogenicOrbitals5p("x", *
                                Mathematics.cartesianToSpherical
                                (x - FeTeLattice[3][0], y -
                                FeTeLattice[3][1], z +
                                FeTeLattice[3][2])) *
                                effectiveCrystalField(x, y, z,
                                mirror_in_xy_plane(FeTeLattice)[
                                3], 2, -2, myZeta,
                                myScreeningLength) * FeOrbitals.
                                hydrogenicOrbitals3d("Yz", *
                                Mathematics.cartesianToSpherical
                                (x, y, z))
t_8_py_dxy_func = lambda x, y, z: TeOrbitals.
                                hydrogenicOrbitals5p("y", *
                                Mathematics.cartesianToSpherical
                                (x - FeTeLattice[3][0], y -
                                FeTeLattice[3][1], z +
                                FeTeLattice[3][2])) *
                                effectiveCrystalField(x, y, z,
                                mirror_in_xy_plane(FeTeLattice)[
                                3], 2, -2, myZeta,
                                myScreeningLength) * FeOrbitals.
                                hydrogenicOrbitals3d("xy", *
                                Mathematics.cartesianToSpherical
                                (x, y, z))
t_8_py_dXz_func = lambda x, y, z: TeOrbitals.
                                hydrogenicOrbitals5p("y", *
                                Mathematics.cartesianToSpherical
                                (x - FeTeLattice[3][0], y -
                                FeTeLattice[3][1], z +
                                FeTeLattice[3][2])) *
                                effectiveCrystalField(x, y, z,
                                mirror_in_xy_plane(FeTeLattice)[
                                3], 2, -2, myZeta,
                                myScreeningLength) * FeOrbitals.
                                hydrogenicOrbitals3d("Xz", *
                                Mathematics.cartesianToSpherical
                                (x, y, z))
t_8_py_dYz_func = lambda x, y, z: TeOrbitals.
                                hydrogenicOrbitals5p("y", *
                                Mathematics.cartesianToSpherical
                                (x - FeTeLattice[3][0], y -
                                FeTeLattice[3][1], z +
                                FeTeLattice[3][2])) *
                                effectiveCrystalField(x, y, z,
                                mirror_in_xy_plane(FeTeLattice)[
                                3], 2, -2, myZeta,
                                myScreeningLength) * FeOrbitals.
                                hydrogenicOrbitals3d("Yz", *
                                Mathematics.cartesianToSpherical
                                (x, y, z))
```

```
t_8_pz_dxy_func = lambda x, y, z: TeOrbitals.
                                hydrogenicOrbitals5p("z", *
                                Mathematics.cartesianToSpherical
                                (x - FeTeLattice[3][0], y -
                                FeTeLattice[3][1], z +
                                FeTeLattice[3][2])) *
                                effectiveCrystalField(x, y, z,
                                mirror_in_xy_plane(FeTeLattice)[
                                3], 2, -2, myZeta,
                                myScreeningLength) * FeOrbitals.
                                hydrogenicOrbitals3d("xy", *
                                Mathematics.cartesianToSpherical
                                (x, y, z))
t_8_pz_dXz_func = lambda x, y, z: TeOrbitals.
                                hydrogenicOrbitals5p("z", *
                                Mathematics.cartesianToSpherical
                                (x - FeTeLattice[3][0], y -
                                FeTeLattice[3][1], z +
                                FeTeLattice[3][2])) *
                                effectiveCrystalField(x, y, z,
                                mirror_in_xy_plane(FeTeLattice)[
                                3], 2, -2, myZeta,
                                myScreeningLength) * FeOrbitals.
                                hydrogenicOrbitals3d("Xz", *
                                Mathematics.cartesianToSpherical
                                (x, y, z))
t_8_pz_dYz_func = lambda x, y, z: TeOrbitals.
                                hydrogenicOrbitals5p("z", *
                                Mathematics.cartesianToSpherical
                                (x - FeTeLattice[3][0], y -
                                FeTeLattice[3][1], z +
                                FeTeLattice[3][2])) *
                                effectiveCrystalField(x, y, z,
                                mirror_in_xy_plane(FeTeLattice)[
                                3], 2, -2, myZeta,
                                myScreeningLength) * FeOrbitals.
                                hydrogenicOrbitals3d("Yz", *
                                Mathematics.cartesianToSpherical
                                (x, y, z))
```

B.3 Script for constructing the FeTe lattice

This script was used to construct the undistorted FeTe lattice with a tetragonal crystal structure and apply to this structure the geometrical distortion described in Sec. 5. The distortion leads to the lifting of the degeneracy of the d_{Xz} and d_{Yz} orbitals in the t_{2g} subgroup.

```
import numpy as np
import pandas as pd

class FeTe():
    def __init__(self, distortion=0.0):
        self.distortion = distortion
        self.a = 3.829*10**(-10)
        self.c = 6.288*10**(-10)
```

```
        self.bondLength = np.sqrt((self.a*0.5)**2+(self.c*0.285)
                                        **2)
        self.unitCell = np.array([self.a, self.a, self.c])
        self.delta = self.distortion * self.bondLength
        self.FeCentralAtom = np.array([0.25, 0.75, 0.0]) * self.
                                        unitCell
        self.Te1 = np.array([0.25, 0.25, 0.285]) * self.unitCell
        self.Te2 = np.array([0.25, 1.25, 0.285]) * self.unitCell
        self.Te3 = np.array([-0.25, 0.75, -0.285]) * self.
                                        unitCell
        self.Te4 = np.array([0.75, 0.75, -0.285]) * self.
                                        unitCell

    def distortParallelogram(self):
        self.Te1 = self.Te1 + np.array([0, -1, 0]) * self.delta
        self.Te2 = self.Te2 + np.array([0, 1, 0]) * self.delta
        self.Te3 = self.Te3 + np.array([1, 0, 0]) * self.delta
        self.Te4 = self.Te4 + np.array([-1, 0, 0]) * self.delta
        self.FeCentralAtom = self.FeCentralAtom + np.array([0, 1
                                        , 0]) * self.delta
        return self.FeCentralAtom, self.Te1, self.Te2, self.Te3,
                                         self.Te4

    def recenterParallelogram(self):
        self.Te1 = self.Te1 - self.FeCentralAtom
        self.Te2 = self.Te2 - self.FeCentralAtom
        self.Te3 = self.Te3 - self.FeCentralAtom
        self.Te4 = self.Te4 - self.FeCentralAtom
        self.FeCentralAtom = np.array([0, 0, 0])
        return self.FeCentralAtom, self.Te1, self.Te2, self.Te3,
                                         self.Te4

    def getTeAtoms(self):
        self.distortParallelogram()
        self.recenterParallelogram()
        TeList = [self.Te1.tolist(), self.Te2.tolist(), self.Te3
                                        .tolist(), self.Te4.
                                        tolist()]
        return TeList

model = FeTe(distortion=0.01)
atoms = model.getTeAtoms()
```

B.4 Script for numerical integration using the Monte Carlo methodology

This script was used to numerically approximate the integrals used to calculate the crystal field splitting, Hund's rule coupling constants and the tight-binding hopping integrals. The Monte Carlo class accepts any function as the integrand so long as it is has dimensions of D=3 or D=6. The full methodology of the Monte Carlo method used for calculating the requisite parameters for the Rayleigh-Schrödinger perturbative model is discussed in Sec. 4.3.3.

```
import numpy as np
from inspect import signature
import time

class MonteCarlo:
    def __init__(self, function, effectiveNuclearCharge=1,
                                    startWidthMultiple=20,
                                    zetaMultiple=1E-3,
                                    pointsMultiplier=1.2,
                                    startPoints=100, outputFile=
                                    "output.txt"):
        self.function = function
        self.effectiveNuclearCharge = effectiveNuclearCharge
        self.startWidthMultiple = startWidthMultiple
        self.zetaMultiple = zetaMultiple
        self.pointsMultiplier = pointsMultiplier
        self.currentPoints = startPoints
        self.outputFile = outputFile
        self.batchSize = 100
        self.bohrRadius = 5.29177210903E-11
        self.radius = self.bohrRadius / self.
                                        effectiveNuclearCharge
        self.zeta = self.zetaMultiple * self.radius
        self.electronCharge = 1.602E-19
        self.halfWidth = self.startWidthMultiple * self.radius
        self.convergenceThreshold = 0.01
        self.results = []
        self.errors = []
        self.isSizeConverged = False
        self.isPointsConverged = False

    def changeConvergenceThreshold(self, convergenceThreshold):
        self.convergenceThreshold = convergenceThreshold

    def doMonteCarlo(self):
        sig = signature(self.function)
        params = sig.parameters
        numParams = len(params)
        if (numParams == 3):
            if (self.currentPoints < self.batchSize):
                X = np.random.uniform(-self.halfWidth, self.
                                                halfWidth, self.
                                                currentPoints)
                Y = np.random.uniform(-self.halfWidth, self.
                                                halfWidth, self.
                                                currentPoints)
                Z = np.random.uniform(-self.halfWidth, self.
                                                halfWidth, self.
                                                currentPoints)
                result = ((2 * self.halfWidth) ** 3 / self.
                                                currentPoints) *
                                                 sum(self.
                                                function(X, Y, Z
                                                ).real)
                mean = (1 / self.currentPoints) * sum(self.
                                                function(X, Y, Z
                                                ).real)
```

```
                variance = (1 / (self.currentPoints - 1)) * sum
                                                ((self.function(
                                                X, Y, Z).real -
                                                mean) ** 2)
                error = ((2 * self.halfWidth) ** 3) * (np.sqrt(
                                                variance) / np.
                                                sqrt(self.
                                                currentPoints))
                return result / self.electronCharge, error /
                                                self.
                                                electronCharge
            else:
                checkNum = 0
                iterBatch = self.currentPoints // self.batchSize
                partialResult = 0
                for i in range(iterBatch + 1):
                    if (i != iterBatch):
                        X = np.random.uniform(-self.halfWidth,
                            self.halfWidth, self.batchSize)
                        Y = np.random.uniform(-self.halfWidth,
                             self.halfWidth, self.batchSize)
                        Z = np.random.uniform(-self.halfWidth,
                            self.halfWidth, self.batchSize)
                        partialResult += sum(self.function(X, Y,
                                                         Z).real
                                                        )
                        checkNum += self.batchSize
                    else:
                        remainingBatch = self.currentPoints -
                            iterBatch * self.batchSize
                        X = np.random.uniform(-self.halfWidth,
                            self.halfWidth, remainingBatch)
                        Y = np.random.uniform(-self.halfWidth,
                            self.halfWidth, remainingBatch)
                        Z = np.random.uniform(-self.halfWidth,
                            self.halfWidth, remainingBatch)
                        partialResult += sum(self.function(X, Y,
                                                         Z).real
                                                        )
                        checkNum += remainingBatch
                result = ((2 * self.halfWidth) ** 3 / self.
                                                currentPoints) *
                                                 partialResult
                mean = (1 / self.currentPoints) * partialResult
                checkNum2 = 0
                iterBatch2 = self.currentPoints // self.
                                                batchSize
                partialResult2 = 0
    for i in range(iterBatch2 + 1):
                    if (i != iterBatch2):
                        X = np.random.uniform(-self.halfWidth,
                            self.halfWidth, self.batchSize)
                        Y = np.random.uniform(-self.halfWidth,
                            self.halfWidth, self.batchSize)
                        Z = np.random.uniform(-self.halfWidth,
                            self.halfWidth, self.batchSize)
```

```
                        partialResult2 += sum((self.function(X,
                                                        Y, Z).
                                                        real -
                                                        mean) **
                                                         2)
                        checkNum2 += self.batchSize
                    else:
                        remainingBatch = self.currentPoints -
                            iterBatch2 * self.batchSize
                        X = np.random.uniform(-self.halfWidth,
                            self.halfWidth, remainingBatch)
                        Y = np.random.uniform(-self.halfWidth,
                            self.halfWidth, remainingBatch)
                        Z = np.random.uniform(-self.halfWidth,
                            self.halfWidth, remainingBatch)
                        partialResult2 += sum((self.function(X,
                                                        Y, Z).
                                                        real -
                                                        mean) **
                                                         2)
                        checkNum2 += remainingBatch
                variance = (1 / (self.currentPoints - 1)) *
                                                partialResult2
                error = ((2 * self.halfWidth) ** 3) * (np.sqrt(
                                                variance) / np.
                                                sqrt(self.
                                                currentPoints))
                return result / self.electronCharge, error /
                                                self.
                                                electronCharge
        elif (numParams == 6):
            if (self.currentPoints < self.batchSize):
                X = np.random.uniform(-self.halfWidth,
                    self.halfWidth, self.currentPoints)
                Y = np.random.uniform(-self.halfWidth,
                    self.halfWidth, self.currentPoints)
                Z = np.random.uniform(-self.halfWidth,
                    self.halfWidth, self.currentPoints)
                U = np.random.uniform(-self.halfWidth,
                    self.halfWidth, self.currentPoints)
                V = np.random.uniform(-self.halfWidth,
                    self.halfWidth, self.currentPoints)
                W = np.random.uniform(-self.halfWidth,
                    self.halfWidth, self.currentPoints)
                result = ((2 * self.halfWidth) ** 6 / self.
                                                currentPoints) *
                                                 sum(self.
                                                function(X, Y, Z
                                                , U, V, W).real)
                mean = (1 / self.currentPoints) * sum(self.
                                                function(X, Y, Z
                                                , U, V, W).real)
                variance = (1 / (self.currentPoints - 1)) * sum
                                                ((self.function(
                                                X, Y, Z, U, V, W
                                                ).real - mean)
                                                ** 2)
```

```
            error = ((2 * self.halfWidth) ** 6) * (np.sqrt(
                                            variance) / np.
                                            sqrt(self.
                                            currentPoints))
            return result / self.electronCharge, error /
                                            self.
                                            electronCharge
        else:
            checkNum = 0
            iterBatch = self.currentPoints // self.batchSize
            partialResult = 0
            for i in range(iterBatch + 1):
                if (i != iterBatch):
                    X = np.random.uniform(-self.halfWidth,
                        self.halfWidth, self.batchSize)
                    Y = np.random.uniform(-self.halfWidth,
                        self.halfWidth, self.batchSize)
                    Z = np.random.uniform(-self.halfWidth,
                        self.halfWidth, self.batchSize)
                    U = np.random.uniform(-self.halfWidth,
                        self.halfWidth, self.batchSize)
                    V = np.random.uniform(-self.halfWidth,
                        self.halfWidth, self.batchSize)
                    W = np.random.uniform(-self.halfWidth,
                        self.halfWidth, self.batchSize)
                    partialResult += sum(self.function(X, Y,
                                                     Z, U, V
                                                    , W).
                                                    real)
                    checkNum += self.batchSize
                else:
                    remainingBatch = self.currentPoints -
                        iterBatch * self.batchSize
                    X = np.random.uniform(-self.halfWidth,
                        self.halfWidth, remainingBatch)
                    Y = np.random.uniform(-self.halfWidth,
                        self.halfWidth, remainingBatch)
                    Z = np.random.uniform(-self.halfWidth,
                        self.halfWidth, remainingBatch)
                    U = np.random.uniform(-self.halfWidth,
                        self.halfWidth, remainingBatch)
                    V = np.random.uniform(-self.halfWidth,
                        self.halfWidth, remainingBatch)
                    W = np.random.uniform(-self.halfWidth,
                        self.halfWidth, remainingBatch)
                    partialResult += sum(self.function(X, Y,
                                                     Z, U, V
                                                    , W).
                                                    real)
                    checkNum += remainingBatch
            result = ((2 * self.halfWidth) ** 6 / self.
                                            currentPoints) *
                                             partialResult
            mean = (1 / self.currentPoints) * partialResult
            checkNum2 = 0
            iterBatch2 = self.currentPoints // self.
                                            batchSize
```

```
                partialResult2 = 0
                for i in range(iterBatch2 + 1):
                    if (i != iterBatch2):
                        X = np.random.uniform(-self.halfWidth,
                            self.halfWidth, self.batchSize)
                        Y = np.random.uniform(-self.halfWidth,
                            self.halfWidth, self.batchSize)
                        Z = np.random.uniform(-self.halfWidth,
                            self.halfWidth, self.batchSize)
                        U = np.random.uniform(-self.halfWidth,
                            self.halfWidth, self.batchSize)
                        V = np.random.uniform(-self.halfWidth,
                            self.halfWidth, self.batchSize)
                        W = np.random.uniform(-self.halfWidth,
                            self.halfWidth, self.batchSize)
                        partialResult2 += sum((self.function(X,
                                                        Y, Z, U,
                                                         V, W).
                                                        real -
                                                        mean) **
                                                         2)
                        checkNum2 += self.batchSize
                    else:
                        remainingBatch = self.currentPoints -
                            iterBatch2 * self.batchSize
                        X = np.random.uniform(-self.halfWidth,
                            self.halfWidth, remainingBatch)
                        Y = np.random.uniform(-self.halfWidth,
                            self.halfWidth, remainingBatch)
                        Z = np.random.uniform(-self.halfWidth,
                            self.halfWidth, remainingBatch)
                        U = np.random.uniform(-self.halfWidth,
                            self.halfWidth, remainingBatch)
                        V = np.random.uniform(-self.halfWidth,
                            self.halfWidth, remainingBatch)
                        W = np.random.uniform(-self.halfWidth,
                            self.halfWidth, remainingBatch)
                        partialResult2 += sum((self.function(X,
                                                        Y, Z, U,
                                                         V, W).
                                                        real -
                                                        mean) **
                                                         2)
                        checkNum2 += remainingBatch
                variance = (1 / (self.currentPoints - 1)) *
                                                partialResult2
                error = ((2 * self.halfWidth) ** 6) * (np.sqrt(
                                                variance) / np.
                                                sqrt(self.
                                                currentPoints))
                return result / self.electronCharge, error /
                                                self.
                                                electronCharge
        else:
            raise Exception("this Monte Carlo integrator only
                                            works with 3 or 6
                                            variables.")
```

```
    def getConvergence(self):
        convergence = abs(self.results[-1] - self.results[-2]) /
                                         abs(self.results[-2])
        return convergence

    def isResultConverged(self):
        if len(self.results <= 1):
            return False
        else:
            if (self.getConvergence < self.convergenceThreshold)
                                             :
                return True
            else:
                return False

    def getConvergedResult(self):
        self.results.append(self.doMonteCarlo()[0])
        self.errors.append(self.doMonteCarlo()[1])
        self.currentPoints = int(self.currentPoints * self.
                                        pointsMultiplier)
        while (self.isPointsConverged == False):
            print(self.currentPoints)
            start_time = time.time()
            result, error = self.doMonteCarlo()
            end_time = time.time()
            self.results.append(result)
            self.errors.append(error)
            if (self.isResultConverged):
                self.isPointsConverged = True
                print("number of points: ", self.currentPoints,
                    " half cube width: ", self.halfWidth, " --->
                                                     ",
                    self.results[-1], self.errors[-1], "
                                                    convergence:
                                                     ",
                    self.getConvergence(), file=open(self.
                                                    outputFile,
                                                    "a"))
                print("check:", "Run time =
                {}".format(end_time - start_time),
                file=open(self.outputFile, "a"))
            else:
                print("number of points: ", self.currentPoints,
                " half cube width: ", self.halfWidth, " ---> ",
                self.results[-1], self.errors[-1], " convergence
                                                : ",
                self.getConvergence(), file=open(self.outputFile
                                                , "a"))
                print("check:", "Run time =
                {}".format(end_time - start_time),
                file=open(self.outputFile, "a"))
                self.currentPoints = round(
                    self.currentPoints * self.pointsMultiplier
                                                    ** 3)

        while (self.isSizeConverged == False):
```

```
            self.currentPoints = int(self.currentPoints * self.
                                            pointsMultiplier **
                                            6)
            self.halfWidth = self.halfWidth * self.
                                            pointsMultiplier
            start_time = time.time()
            result, error = self.doMonteCarlo()
            end_time = time.time()
            self.results.append(result)
            self.errors.append(error)
            if (self.getConvergence() < self.
                                            convergenceThreshold
                                            ):
                self.isSizeConverged = True
                print("number of steps: ", self.currentPoints, "
                                                 half cube width
                                                : ", self.
                                                halfWidth, "
                                                ---> ", self.
                                                results[-1],
                                                self.errors[-1],
                                                 " convergence:
                                                ", self.
                                                getConvergence()
                                                , file=open(self
                                                .outputFile, "a"
                                                ))
                print("check:", "Run time = {}".format(end_time
                                                - start_time),
                                                file=open(self.
                                                outputFile, "a")
                                                )
            else:
                print("number of steps: ", self.currentPoints, "
                                                 half cube width
                                                : ", self.
                                                halfWidth, "
                                                ---> ", self.
                                                results[-1],
                                                self.errors[-1],
                                                 " convergence:
                                                ", self.
                                                getConvergence()
                                                , file=open(self
                                                .outputFile, "a"
                                                ))
                print("check:", "Run time = {}".format(end_time
                                                - start_time),
                                                file=open(self.
                                                outputFile, "a")
                                                )
                self.isPointsConverged = False
                while (self.isPointsConverged == False):
                    self.currentPoints = round(
                        self.currentPoints *
                        self.pointsMultiplier ** 6)
                    start_time = time.time()
```

```
                    result, error = self.doMonteCarlo()
                    end_time = time.time()
                    self.results.append(result)
                    self.errors.append(error)
                    if (self.getConvergence() <
                        self.convergenceThreshold):
                        self.isPointsConverged = True
                        print("number of steps: ",
                            self.currentPoints,
                            " half cube width: ",
                            self.halfWidth, " ---> ",
                            self.results[-1], self.errors[-1],
                            " convergence: ",
                            self.getConvergence(),
                            file=open(self.outputFile, "a"))
                        print("check:", "Run time =
                            {}".format(end_time - start_time),
                            file=open(self.outputFile, "a"))
                        print("success: integration has
                            converged for the number of steps.",
                            file=open(self.outputFile, "a"))
                    else:
                        print("number of steps: ",
                            self.currentPoints,
                            " half cube width: ",
                            self.halfWidth, " --->
                            ", self.results[-1], self.errors[-1]
                                                             ,
                            " convergence: ",
                            self.getConvergence(),
                            file=open(self.outputFile, "a"))
                        print("check:", "Run time =
                            {}".format(end_time - start_time),
                            file=open(self.outputFile, "a"))

        print("success: integration has converged for the size
                                        of the integration box",
                                         file=open(self.
                                        outputFile, "a"))
        return self.results[-1], self.errors[-1]
```

B.5 Program for the Rayleigh-Schrödinger perturbative model

A parameter file was written so as to keep the inputs to the Rayleigh-Schrödinger perturbative model separate from the main file for calculating the elements of the effective 36×36 Hamiltonian matrix. This is for ease of access and to prevent any any accidental corruption of the main program.

```
[hamiltonian_model_section]
fe_1 = 1
fe_2 = 2
d_xy = 5.56
d_Xz = 6.86
```

```
d_Yz = 6.81
d_x2y2 = 4.64
d_z2 = 5.52
fe_2U_a = 1.23
fe_2U_b = 0.18
fe_2U_c = 0.13
fe_2U_d = 0.22
fe_2U_e = 0.20
fe_2U_f = 0.20
fe_2U_g = 0.15
fe_J_a = 1.24
fe_J_b = 0.20
fe_J_c = 0.13
fe_J_d = 0.22
fe_J_e = 0.20
fe_J_f = 0.20
fe_J_g = 0.15
te_2U_a = 4.35
te_2U_b = 3.23
te_J_a = 4.33
te_J_b = 0.56
e_fe = -12.27
e_te = -2.36
```

The following program, which takes the input parameters from the above file, calculates a sub-matrix of the effective Hamiltonian matrix for a chosen pair of Fe atoms in the FeTe lattice. The virtual electron hopping processes can be mediated by either one or two intermediary Te atoms.

```
[hamiltonian_model_section]
import numpy as np
from ast import literal_eval
import sys

class FeTeHamiltonian:

    def __init__(self, initial_Fe_states, number_of_Te,
                                   parameter_file_path):
        self.initial_Fe_states = initial_Fe_states
        self.number_of_Te = number_of_Te
        self.parameter_file_path = parameter_file_path
        self.zeroth_order_state_space = []
        self.first_order_state_space = []
        self.second_order_state_space = []
        self.Fe_U_antiparallel = []
        self.Fe_J = []
        self.Te_U_antiparallel = []
        self.Te_J = []
        self.zeroth_order_transition_matrix = []
        self.first_order_transition_matrix = []
        self.second_order_transition_matrix = []
        self.zerothOrdereHamiltonian = []
        self.secondOrderHamiltonian = []
        self.fourthOrderHamiltonian = []
        self.Hamiltonian = []
        self.overlap_integrals = np.zeros((6, 12))
```

```
    def getParameters(self):
        try:
            from configparser import ConfigParser
        except ImportError:
            from ConfigParser import ConfigParser  # ver. < 3.0
        config = ConfigParser()
        config.read(self.parameter_file_path)

        self.fe_1 = int(config.get(
            "hamiltonian_model_section", "fe_1"))
        self.fe_2 = int(config.get(
            "hamiltonian_model_section", "fe_2"))
        self.d_xy = float(config.get(
            "hamiltonian_model_section", "d_xy"))
        self.d_Xz = float(config.get(
            "hamiltonian_model_section", "d_Xz"))
        self.d_Yz = float(config.get(
            "hamiltonian_model_section", "d_Yz"))
        self.d_x2y2 = float(config.get(
            "hamiltonian_model_section", "d_x2y2"))
        self.d_z2 = float(config.get(
            "hamiltonian_model_section", "d_z2"))
        self.fe_2U_a = float(config.get(
            "hamiltonian_model_section", "fe_2U_a"))
        self.fe_2U_b = float(config.get(
            "hamiltonian_model_section", "fe_2U_b"))
        self.fe_2U_c = float(config.get(
            "hamiltonian_model_section", "fe_2U_c"))
        self.fe_2U_d = float(config.get(
            "hamiltonian_model_section", "fe_2U_d"))
        self.fe_2U_e = float(config.get(
            "hamiltonian_model_section", "fe_2U_e"))
        self.fe_2U_f = float(config.get(
            "hamiltonian_model_section", "fe_2U_f"))
        self.fe_2U_g = float(config.get(
            "hamiltonian_model_section", "fe_2U_g"))
        self.fe_J_a = float(config.get(
            "hamiltonian_model_section", "fe_J_a"))
        self.fe_J_b = float(config.get(
            "hamiltonian_model_section", "fe_J_b"))
        self.fe_J_c = float(config.get(
            "hamiltonian_model_section", "fe_J_c"))
        self.fe_J_d = float(config.get(
            "hamiltonian_model_section", "fe_J_d"))
        self.fe_J_e = float(config.get(
            "hamiltonian_model_section", "fe_J_e"))
        self.fe_J_f = float(config.get(
            "hamiltonian_model_section", "fe_J_f"))
        self.fe_J_g = float(config.get(
            "hamiltonian_model_section", "fe_J_g"))
        self.te_2U_a = float(config.get(
            "hamiltonian_model_section", "te_2U_a"))
        self.te_2U_b = float(config.get(
            "hamiltonian_model_section", "te_2U_b"))
        self.te_J_a = float(config.get(
            "hamiltonian_model_section", "te_J_a"))
        self.te_J_b = float(config.get(
```

```
            "hamiltonian_model_section", "te_J_b"))
        self.e_fe = float(config.get(
            "hamiltonian_model_section", "e_fe"))
        self.e_te = float(config.get(
            "hamiltonian_model_section", "e_te"))
        self.Fe_U_antiparallel = [[self.fe_2U_a, self.fe_2U_b,
                                        self.fe_2U_b, self.
                                        fe_2U_c, self.fe_2U_d],
                                        [self.fe_2U_b, self.
                                        fe_2U_a, self.fe_2U_e,
                                        self.fe_2U_f, self.
                                        fe_2U_g], [self.fe_2U_b,
                                         self.fe_2U_e, self.
                                        fe_2U_a, self.fe_2U_f,
                                        self.fe_2U_g], [self.
                                        fe_2U_c, self.fe_2U_f,
                                        self.fe_2U_f, self.
                                        fe_2U_a, self.fe_2U_d],
                                        [self.fe_2U_d, self.
                                        fe_2U_g, self.fe_2U_g,
                                        self.fe_2U_d, self.
                                        fe_2U_a]]
        self.Fe_J = [[self.fe_J_a, self.fe_J_b, self.fe_J_b,
                                        self.fe_J_c, self.fe_J_d
                                        ], [self.fe_J_b, self.
                                        fe_J_a, self.fe_J_e,
                                        self.fe_J_f, self.fe_J_g
                                        ], [self.fe_J_b, self.
                                        fe_J_e, self.fe_J_a,
                                        self.fe_J_f, self.fe_J_g
                                        ], [self.fe_J_c, self.
                                        fe_J_f, self.fe_J_f,
                                        self.fe_J_a, self.fe_J_d
                                        ], [self.fe_J_d, self.
                                        fe_J_g, self.fe_J_g,
                                        self.fe_J_d, self.fe_J_a
                                        ]]
        self.Te_U_antiparallel = [[self.te_2U_a, self.te_2U_b,
                                        self.te_2U_b], [self.
                                        te_2U_b, self.te_2U_a,
                                        self.te_2U_b], [self.
                                        te_2U_b, self.te_2U_b,
                                        self.te_2U_a]]
        self.Te_J = [[self.te_2U_a, self.te_2U_b, self.te_2U_b],
                                         [self.te_2U_b, self.
                                        te_2U_a, self.te_2U_b],
                                        [self.te_2U_b, self.
                                        te_2U_b, self.te_2U_a]]
        with open("overlap_matrix_allFe_with0.txt", "r") as f:
            searchlines = f.readlines()
        if (self.number_of_Te == 1):
            pass
        elif (self.number_of_Te == 2):
            for i, line in enumerate(searchlines):
                if "Te-atom below" in line:
```

```
                    self.overlap_integrals[0][:3] = literal_eval
                                                    (searchlines
                                                    [i+2])
                    self.overlap_integrals[1][:3] = literal_eval
                                                    (searchlines
                                                    [i+3])
                    self.overlap_integrals[2][:3] = literal_eval
                                                    (searchlines
                                                    [i+4])
                    self.overlap_integrals[0][3:6] =
                                                    literal_eval
                                                    (searchlines
                                                    [i+6])
                    self.overlap_integrals[1][3:6] =
                                                    literal_eval
                                                    (searchlines
                                                    [i+7])
                    self.overlap_integrals[2][3:6] =
                                                    literal_eval
                                                    (searchlines
                                                    [i+8])
                    self.overlap_integrals[0][6:9] =
                                                    literal_eval
                                                    (searchlines
                                                    [i+10])
                    self.overlap_integrals[1][6:9] =
                                                    literal_eval
                                                    (searchlines
                                                    [i+11])
                    self.overlap_integrals[2][6:9] =
                                                    literal_eval
                                                    (searchlines
                                                    [i+12])
                    self.overlap_integrals[0][9:12] =
                                                    literal_eval
                                                    (searchlines
                                                    [i+14])
                    self.overlap_integrals[1][9:12] =
                                                    literal_eval
                                                    (searchlines
                                                    [i+15])
                    self.overlap_integrals[2][9:12] =
                                                    literal_eval
                                                    (searchlines
                                                    [i+16])
                elif "Te-atom above" in line:
                    self.overlap_integrals[3][:3] = literal_eval
                                                    (searchlines
                                                    [i+2])
                    self.overlap_integrals[4][:3] = literal_eval
                                                    (searchlines
                                                    [i+3])
                    self.overlap_integrals[5][:3] = literal_eval
                                                    (searchlines
                                                    [i+4])
```

```
                    self.overlap_integrals[3][3:6] =
                                                    literal_eval
                                                    (searchlines
                                                    [i+6])
                    self.overlap_integrals[4][3:6] =
                                                    literal_eval
                                                    (searchlines
                                                    [i+7])
                    self.overlap_integrals[5][3:6] =
                                                    literal_eval
                                                    (searchlines
                                                    [i+8])
                    self.overlap_integrals[3][6:9] =
                                                    literal_eval
                                                    (searchlines
                                                    [i+10])
                    self.overlap_integrals[4][6:9] =
                                                    literal_eval
                                                    (searchlines
                                                    [i+11])
                    self.overlap_integrals[5][6:9] =
                                                    literal_eval
                                                    (searchlines
                                                    [i+12])
                    self.overlap_integrals[3][9:12] =
                                                    literal_eval
                                                    (searchlines
                                                    [i+14])
                    self.overlap_integrals[4][9:12] =
                                                    literal_eval
                                                    (searchlines
                                                    [i+15])
                    self.overlap_integrals[5][9:12] =
                                                    literal_eval
                                                    (searchlines
                                                    [i+16])
            print(self.overlap_integrals)

    def generateStateSpace(self):
        if (self.number_of_Te == 1):
            for i in self.initial_Fe_states[:]:
                for j in self.initial_Fe_states[:]:
                    element = [*i,1,1,1,1,1,1,*j]
                    self.zeroth_order_state_space.append(element
                                                    )
            for i in range(len(self.zeroth_order_state_space)):
                vector = self.zeroth_order_state_space[i][:]
                for j in range(6):
                    rem = j % 2
                    electron_annihilated = vector.copy()
                    electron_annihilated[j + 6] = 0
                    # if the remainder is equal to zero then
                      # this is considering hopping of up spin
                                                    electrons
                    if (rem == 0):
                        for k in [0, 2, 4, 12, 14, 16]:
                            # if the state is unoccupied then
```

```
                          # add an electron to it
                        if (electron_annihilated[k] == 0):
                            parser =
                                electron_annihilated.copy()
                            parser[k] = 1
                            electron_created = parser.copy()
                            self.first_order_state_space.
                                append(electron_created)
                        else:
                            pass
                # if the remainder is equal to 1 then this
                  # is considering hopping of down spin
                                                electrons
                else:
                    for k in [1, 3, 5, 13, 15, 17]:
                        # if the state is unoccupied then
                          # add an electron to it
                        if (vector[k] == 0):
                            parser =
                                electron_annihilated.copy()
                            parser[k] = 1
                            electron_created = parser.copy()
                            self.first_order_state_space.
                                append(electron_created)
                        else:
                            pass
        self.first_order_state_space =
          self.zeroth_order_state_space +
self.first_order_state_space
        print("size of first hop set: ",
          len(self.first_order_state_space))
        for i in range(len(self.first_order_state_space)):
            vector = self.first_order_state_space[i][:]
            for j in range(12):
                rem = j % 2
                electron_annihilated = vector.copy()
                # check whether the state is in the first or
                                                 second Fe
                if j in range(0, 6):
                    if (electron_annihilated[j] == 1):
                        electron_annihilated[j] = 0
                        if (rem == 0):
                            for k in [6, 8, 10]:
                                if (electron_annihilated[k]
                                  == 0):
                                    parser =
                                        electron_annihilated
                                          .copy()
                                    parser[k] = 1
                                    electron_created =
                                        parser.copy()
                                    self.
                                        second_
                                  order_
                                  state_
                                  space.
                                        append(
```

```
                                            electron_created)
                                    else:
                                        pass
                            else:
                                for k in [7, 9, 11]:
                                    if (electron_annihilated[k]
                                      == 0):
                                        parser =
                                            electron_annihilated
                                            .copy()
                                        parser[k] = 1
                                        electron_created =
                                            parser.copy()
                                        self.
                                            second_
                                            order_
                                            state_
                                            space.
                                            append(
                                            electron_created)
                                    else:
                                        pass
                        else:
                            pass
                    elif j in range(6, 12):
                        if (electron_annihilated[j + 6] == 1):
                            electron_annihilated[j + 6] = 0
                            if (rem == 0):
                                for k in [6, 8, 10]:
                                    if (electron_annihilated[k]
                                    == 0):
                                        parser =
                                            electron_annihilated
                                            .copy()
                                        parser[k] = 1
                                        electron_created =
                                            parser.copy()
                                        self.
                                            second_
                                            order_
                                            state_
                                            space.
                                            append(
                                            electron_created)
                                    else:
                                        pass
                            else:
                                for k in [7, 9, 11]:
                                    if (electron_annihilated[k]
                                    == 0):
                                        parser =
                                            electron_annihilated
                                            .copy()
                                        parser[k] = 1
                                        electron_created =
                                            parser.copy()
                                        self.
```

```
                                  second_
                                  order_
                                  state_
                                  space.
                                  append(
                                  electron_created)
                          else:
                              pass
              else:
                  pass
  for i in range(len(self.first_order_state_space)):
      vector = self.first_order_state_space[i][:]
      for j in range(6):
          rem = j % 2
          electron_annihilated = vector.copy()
          if (electron_annihilated[j + 6] == 1):
              electron_annihilated[j + 6] = 0
              # if the state is unoccupied then
                  # add an electron to it
              if (rem == 0):
                  for k in [0, 2, 4, 12, 14, 16]:
                      # if the state is unoccupied
                          # then add an electron to it
                      if (electron_annihilated[k]
                          == 0):
                          parser =
                              electron_annihilated
                              .copy()
                          parser[k] = 1
                          electron_created =
                            parser.copy()
                          self.
                              second_order_state_space
                                                               .

                              append(electron_created)
                      else:
                          pass
              # if the remainder is equal to 1 then
                                              this
                  # is considering hopping of down
                      # spin electrons
              else:
                  for k in [1, 3, 5, 13, 15, 17]:
                      # if the state is unoccupied
                          # then add an electron to it
                      if (vector[k] == 0):
                          parser =
                            electron_annihilated
.copy()
                          parser[k] = 1
                          electron_created =
                            parser.copy()
                          self.
                                second_
                                order_
                                state_
```

```
                                          space.
                                      append(
                                    electron_created)
                                else:
                                    pass
                    else:
                        pass
            self.second_order_state_space =
            self.first_order_state_space +
            self.second_order_state_space
            print("size of second set with non-uniques: ",
            len(self.
            second_order_state_space))
            self.second_order_state_space =
            [list(item) for item in set(tuple(row)
            for row in self.second_order_state_space)]
            print("size of second set only with uniqueness: ",
            len(self.
            second_order_state_space))
        elif (self.number_of_Te == 2):
            for i in self.initial_Fe_states[:]:
                    print(self.initial_Fe_states[:])
                    print(i)
                    element = [*i[0],1,1,1,1,1,1,1,1,1,1,1,1,*i[
                                                    1]]
                    print("element: ", element)
                    self.zeroth_order_state_space.append(element
                                                    )
            for i in range(len(self.zeroth_order_state_space)):
                vector = self.zeroth_order_state_space[i][:]
                for j in range(6):
                    rem = j % 2
                    electron_annihilated = vector.copy()
                    electron_annihilated[j + 6] = 0
                    # if the remainder is equal to zero
                        # then this is considering hopping of up
                                                         spin
                                                        electrons

                    if (rem == 0):
                        for k in [0, 2, 4, 18, 20, 22]:
                            # if the state is unoccupied
                                # then add an electron to it
                            if (electron_annihilated[k]
                            == 0):
                                parser = electron_annihilated
                                .copy()
                                parser[k] = 1
                                electron_created = parser.copy()
                                self.first_order_state_space.
                                append(electron_created)
                            else:
                                pass
                    # if the remainder is equal to 1 then
                        # this is considering hopping of
                            # down spin electrons
                    else:
```

```
                    for k in [1, 3, 5, 19, 21, 23]:
                        # if the state is unoccupied
                            # then add an electron to it
                        if (vector[k] == 0):
                            parser = electron_annihilated
                                .copy()
                            parser[k] = 1
                            electron_created =
                                parser.copy()
                            self.first_order_state_space.
                            append(electron_created)
                        else:
                            pass
        self.first_order_state_space = self.
          zeroth_order_state_space +
self.first_order_state_space
        print("size of first hop set: ",
          len(self.first_order_state_space))
        for i in range(len(self.first_order_state_space)):
            vector = self.first_order_state_space[i][:]
            for j in range(12):
                rem = j % 2
                electron_annihilated = vector.copy()
                # check whether the state is in the first or
                                                 second Fe
                if j in range(0, 6):
                    if (electron_annihilated[j] == 1):
                        electron_annihilated[j] = 0
                        if (rem == 0):
                            for k in [6, 8, 10, 12, 14, 16]:
                                if (electron_annihilated[k]
                                == 0):
                                    parser =
                                    electron_annihilated
                                    .copy()
                                    parser[k] = 1
                                    electron_created =
                                    parser.copy()
                                    self.
                                    second_order_state_space
                                    .append(
                                    electron_created)
                                else:
                                    pass
                        else:
                            for k in [7, 9, 11, 13, 15, 17]:
                                if (electron_annihilated[k]
                                == 0):
                                    parser =
                                    electron_annihilated
                                    .copy()
                                    parser[k] = 1
                                    electron_created =
                                    parser.copy()
                                    self.
                                    second_order_state_space
                                    .append(
```

```
                                        electron_created)
                                    else:
                                        pass
                        else:
                            pass
                    elif j in range(6, 12):
                        if (electron_annihilated[j + 12] == 1):
                            electron_annihilated[j + 12] = 0
                            if (rem == 0):
                                for k in [6, 8, 10, 12, 14, 16]:
                                    if (electron_annihilated[k]
                                    == 0):
                                        parser =
                                        electron_annihilated
                                        .copy()
                                        parser[k] = 1
                                        electron_created =
                                        parser.copy()
                                        self.
                                        second_order_state_space
                                        .append(
                                        electron_created)
                                    else:
                                        pass
                            else:
                                for k in [7, 9, 11, 13, 15, 17]:
                                    if (electron_annihilated[k]
                                    == 0):
                                        parser =
                                        electron_annihilated
                                        .copy()
                                        parser[k] = 1
                                        electron_created =
                                        parser.copy()
                                        self.
                                        second_order_state_space
                                        .append(
                                        electron_created)
                                    else:
                                        pass
                        else:
                            pass
            for i in range(len(self.first_order_state_space)):
                vector = self.first_order_state_space[i][:]
                for j in range(12):
                    rem = j % 2
                    electron_annihilated = vector.copy()
                    if (electron_annihilated[j + 6] == 1):
                        electron_annihilated[j + 6] = 0
                        # if the state is unoccupied then add an

                                                        electron
                                                         to it
                        if (rem == 0):
                            for k in [0, 2, 4, 18, 20, 22]:
                                # if the state is unoccupied
                                    # then add an
```

```
                                        # electron to it
                                if (electron_annihilated[k]
                                == 0):
                                    parser =
                                    electron_annihilated
                                    .copy()
                                    parser[k] = 1
                                    electron_created =
                                    parser.copy()
                                    self.
                                    second_order_state_space.
                                    append(
                                    electron_created)
                                else:
                                    pass
                        # if the remainder is equal to 1
                            # then this is considering hopping
                                # of down spin electrons
                        else:
                            for k in [1, 3, 5, 19, 21, 23]:
                                # if the state is unoccupied
                                    # then add an
                                        #electron to it
                                if (vector[k] == 0):
                                    parser =
                                    electron_annihilated
                                    .copy()
                                    parser[k] = 1
                                    electron_created =
                                    parser.copy()
                                    self.
                                    second_order_state_space
                                    .append(
                                    electron_created)
                                else:
                                    pass
                    else:
                        pass
            self.second_order_state_space =
                self.first_order_state_space +
                self.second_order_state_space
            print("size of second set with non-uniques: ",
                len(self.second_order_state_space))
            self.second_order_state_space =
                [list(item) for item in set(tuple(row)
                    for row in self.second_order_state_space)]
            print("size of second set only with uniqueness: ",
                len(self.second_order_state_space))
        else:
            raise Exception("There should be either one
                or two neighboring Te atoms included in the
                                                model")

    def calculateZerothOrderEnergy(self, state):
        energy = 0
        number_of_Te = 2
        if (number_of_Te == 1):
```

```
            pass
        elif (number_of_Te == 2):
            """
            energy += (np.sum(state[:6]) + np.sum(state[18:])) *
                                             self.epsilon
            energy += (np.sum(state[2:6]) + np.sum(state[20:]))
                                            * self.delta_one
            energy += (np.sum(state[4:6]) + np.sum(state[22:]))
                                            * self.delta_two
            """
            energy += 2 * self.d_xy
            energy += (np.sum(state[4:6]) + np.sum(state[22:24])
                                            ) * self.d_Xz
            energy += (np.sum(state[2:4]) + np.sum(state[20:22])
                                            ) * self.d_Yz
            energy += (np.sum(state[0:2]) + np.sum(state[18:20])
                                            ) * self.d_x2y2
            energy += 2 * self.d_z2
            energy += 8 * self.e_fe
            energy += (np.sum(state[0:6]) + np.sum(state[18:24])
                                            ) * self.e_fe
            energy += np.sum(state[6:18]) * self.e_te
            total_state = [1, 1, 1, 1, *state[:-6], 1, 1, 1, 1,
                                            *state[-6:]]
            for i in range(10):
                for j in range(10):
                    if (i != j):
                        if (i % 2 == 0) & (j % 2 == 0):
                            energy +=
                            total_state[i] *
                            total_state[j] *
                            0.5 * (self.Fe_U_antiparallel[i // 2
                                                            ][j
                                                            // 2
                                                            ] -
                            self.Fe_J[i // 2][j // 2])
                        elif (i % 2 == 1) & (j % 2 == 1):
                            energy +=
                            total_state[i] *
                            total_state[j] *
                            0.5 * (self.Fe_U_antiparallel[i // 2
                                                            ][j
                                                            // 2
                                                            ] -
                            self.Fe_J[i // 2][j // 2])
                        else:
                            energy +=
                            total_state[i] *
                            total_state[j] *
                            0.5 * self.Fe_U_antiparallel[i // 2]
                                                            [j
                                                            // 2
                                                            ]
            for i in range(22, 32):
                for j in range(22, 32):
                    if (i != j):
                        if (i % 2 == 0) & (j % 2 == 0):
```

```
                            energy +=
                            total_state[i] *
                            total_state[j] *
                            0.5 * (self.Fe_U_antiparallel[(i -
                                                            22)
                                                            // 2
                                                            ][(j
                                                             -
                                                            22)
                                                            // 2
                                                            ] -
                            self.Fe_J[(i - 22) // 2][(j - 22) //
                                                             2])
                        elif (i % 2 == 1) & (j % 2 == 1):
                            energy +=
                            total_state[i] *
                            total_state[j] *
                            0.5 * (self.Fe_U_antiparallel[(i -
                                                            22)
                                                            // 2
                                                            ][(j
                                                             -
                                                            22)
                                                            // 2
                                                            ] -
                            self.Fe_J[(i - 22) // 2][(j - 22) //
                                                             2])
                        else:
                            energy +=
                            total_state[i] *
                            total_state[j] *
                            0.5 * self.Fe_U_antiparallel[(i - 22
                                                            ) //
                                                             2][
                                                            (j -
                                                             22)
                                                             //
                                                            2]
            for i in range(10, 16):
                for j in range(10, 16):
                    if (i != j):
                        if (i % 2 == 0) & (j % 2 == 0):
                            energy +=
                            total_state[i] *
                            total_state[j] *
                            0.5 * (self.Te_U_antiparallel[(i -
                                                            10)
                                                            // 2
                                                            ][(j
                                                             -
                                                            10)
                                                            // 2
                                                            ] -
                            self.Te_J[(i - 10) // 2][(j - 10) //
                                                             2])
                        elif (i % 2 == 1) & (j % 2 == 1):
                            energy +=
```

```
                            total_state[i] * total_state[j] *
                            0.5 * (self.Te_U_antiparallel[(i -
                                                            10)
                                                            // 2
                                                            ][(j
                                                             -
                                                            10)
                                                            // 2
                                                            ] -
                            self.Te_J[(i - 10) // 2][(j - 10) //
                                                             2])
                        else:
                            energy +=
                            total_state[i] *
                            total_state[j] *
                            0.5 * self.Te_U_antiparallel[(i - 10
                                                            ) //
                                                             2][
                                                            (j -
                                                             10)
                                                             //
                                                            2]
            for i in range(16, 22):
                for j in range(16, 22):
                    if (i != j):
                        if (i % 2 == 0) & (j % 2 == 0):
                            energy +=
                            total_state[i] *
                            total_state[j] *
                            0.5 * (self.Te_U_antiparallel[(i -
                                                            16)
                                                            // 2
                                                            ][(j
                                                             -
                                                            16)
                                                            // 2
                                                            ] -
                            self.Te_J[(i - 16) // 2][(j - 16) //
                                                             2])
                        elif (i % 2 == 1) & (j % 2 == 1):
                            energy +=
                            total_state[i] *
                            total_state[j] *
                            0.5 * (self.Te_U_antiparallel[(i -
                                                            16)
                                                            // 2
                                                            ][(j
                                                             -
                                                            16)
                                                            // 2
                                                            ] -
                            self.Te_J[(i - 16) // 2][(j - 16) //
                                                             2])
                        else:
                            energy +=
                            total_state[i] *
                            total_state[j] *
```

```
                            0.5 * self.Te_U_antiparallel[(i - 16
                                                            ) //
                                                             2][
                                                            (j -
                                                             16)
                                                             //
                                                            2]
        else:
            pass
        return energy

    def generateOrderingSign(self, start, end):
        diff = np.array(start) - np.array(end)
        for i in range(len(diff.tolist())):
            if (diff[i] == 1):
                num_to_left = i
            else:
                pass
        sign = (-1) ** np.sum(np.array(start[:num_to_left]))
        temp = []
        for i in range(len(start)):
            if (diff[i] != 1):
                temp.append(start[i])
            else:
                temp.append(0)
        diff = np.array(temp) - np.array(end)
        for i in range(len(diff.tolist())):
            if (diff[i] == -1):
                num_to_left = i
            else:
                pass
        sign = sign * (-1) ** np.sum(np.array(temp[:num_to_left]
                                        ))
        return sign

    def canElectronHop(self, start, end):
        if (self.number_of_Te == 1):
            diff = np.array(start) - np.array(end)
            diff_Fe_one = list(filter(lambda num: num != 0, diff
                                            [:6].tolist()))
            diff_Te = list(filter(lambda num: num != 0, diff[6:
                                            12].tolist()))
            diff_Fe_two = list(filter(lambda num: num != 0, diff
                                            [12:18].tolist()))
            list_changes = [len(diff_Fe_one), len(diff_Te), len(
                                            diff_Fe_two)]
            list_changes_sorted = sorted(list_changes)
            if (sorted(list_changes) != [0, 1, 1]):
                return 0
            else:
                if (sorted([*diff_Fe_one, *diff_Te, *diff_Fe_two
                                                ]) == [-1, 1]) &
                                                 (list_changes[1
                                                ] != 0):
                    return 1
                else:
                    return 0
```

```
        elif (self.number_of_Te == 2):
            diff = np.array(start) - np.array(end)
            diff_Fe_one = list(filter(lambda num: num != 0, diff
                                            [:6].tolist()))
            diff_Te = list(filter(lambda num: num != 0, diff[6:
                                            18].tolist()))
            diff_Fe_two = list(filter(lambda num: num != 0, diff
                                            [18:24].tolist()))
            list_changes = [len(diff_Fe_one), len(diff_Te), len(
                                            diff_Fe_two)]
            list_changes_sorted = sorted(list_changes)
            change_total_spin_projection = sum(diff[::2]) - sum(
                                            diff[1::2])
            if (change_total_spin_projection != 0):
                return 0
            else:
                if (sorted(list_changes) != [0, 1, 1]):
                    return 0
                else:
                    if (sorted([*diff_Fe_one, *diff_Te, *
                                                    diff_Fe_two]
                                                    ) == [-1, 1]
                                                    ) & (
                                                    list_changes
                                                    [1] != 0):
                        return 1
                    else:
                        return 0

    def getOverlapIntegral(self, start, end):
        if (self.number_of_Te == 1):
            pass
        elif (self.number_of_Te == 2):
            if (self.canElectronHop(start, end) == 1):
                diff = np.array(start) - np.array(end)
                starting_orbital = diff.tolist().index(-1)
                ending_orbital = diff.tolist().index(1)
                if (starting_orbital in [0, 1]):
                    col = 1
                elif (starting_orbital in [2, 3]):
                    col = 2
                elif (starting_orbital in [4, 5]):
                    col = 3
                elif (starting_orbital in [6, 7]):
                    row = 1
                elif (starting_orbital in [8, 9]):
                    row = 2
                elif (starting_orbital in [10, 11]):
                    row = 3
                elif (starting_orbital in [12, 13]):
                    row = 4
                elif (starting_orbital in [14, 15]):
                    row = 5
                elif (starting_orbital in [16, 17]):
                    row = 6
                elif (starting_orbital in [18, 19]):
                    col = 4
```

```
                elif (starting_orbital in [20, 21]):
                    col = 5
                elif (starting_orbital in [22, 23]):
                    col = 6
                if (ending_orbital in [0, 1]):
                    col = 1
                elif (ending_orbital in [2, 3]):
                    col = 2
                elif (ending_orbital in [4, 5]):
                    col = 3
                elif (ending_orbital in [6, 7]):
                    row = 1
                elif (ending_orbital in [8, 9]):
                    row = 2
                elif (ending_orbital in [10, 11]):
                    row = 3
                elif (ending_orbital in [12, 13]):
                    row = 4
                elif (ending_orbital in [14, 15]):
                    row = 5
                elif (ending_orbital in [16, 17]):
                    row = 6
                elif (ending_orbital in [18, 19]):
                    col = 4
                elif (ending_orbital in [20, 21]):
                    col = 5
                elif (ending_orbital in [22, 23]):
                    col = 6
                if col in [1, 2, 3]:
                    col = 3 * (self.fe_1 - 1) + col
                elif col in [4, 5, 6]:
                    col = 3 * (self.fe_2 - 1) + col - 3
                return self.overlap_integrals[row-1][col-1]
            else:
                return 0

    def calculateEnergyDifference(self, start, end):
        if (self.number_of_Te == 1):
            pass
        if (self.number_of_Te == 2):
            ENERGY_DIFF = self.calculateZerothOrderEnergy(start)
                                             -

            self.calculateZerothOrderEnergy(end)
        else:
            pass
        return ENERGY_DIFF

    def calculateUnconstrainedTransitionMatrix(self):
        dim = len(self.second_order_state_space)
        for power in range(1):
            dim = len(self.second_order_state_space)
            transition_matrix = np.zeros((dim, dim))
            for i in range(dim):
                for j in range(dim):
                    if (i != j):
                        numerator = self.
                        generateOrderingSign(
```

```
                        self.second_order_state_space[j],
                        self.second_order_state_space[i]) *
                        self.canElectronHop(self.
                        second_order_state_space[j],
                        self.second_order_state_space[i]) *
                        self.getOverlapIntegral(
                        self.second_order_state_space[j],
                        self.second_order_state_space[i])
                        denominator = 1.
                        if (denominator != 0):
                            prefactor = numerator /
                            (denominator ** power)
                            transition_matrix[i][j] = prefactor
                            if (transition_matrix[i][j] != 0):
                                pass
                        else:
                            pass
                    else:
                        pass
            if (power == 0):
                self.zeroth_order_transition_matrix =
                transition_matrix
            elif (power == 1):
                self.first_order_transition_matrix =
                transition_matrix
            elif (power == 2):
                self.second_order_transition_matrix =
                transition_matrix

    def calculateZerothOrderHamiltonian(self):
        print("---------0th--------------------------")
        self.zerothOrderHamiltonian =
        np.zeros((int(len(self.zeroth_order_state_space) / 2),
        int(len(self.zeroth_order_state_space) / 2)))
        for i in range(int(len(self.zeroth_order_state_space) /
                                    2)):
            for j in range(int(len(self.zeroth_order_state_space
                                        ) / 2),
            len(self.zeroth_order_state_space)):
                if (self.zeroth_order_state_space[i] == self.
                zeroth_order_state_space[j]):
                    self.zerothOrderHamiltonian[i][j-int(len(
                                                self.
                    zeroth_order_state_space) / 2)] =
                    self.calculateZerothOrderEnergy(
                    self.zeroth_order_state_space[i])
                else:
                    self.zerothOrderHamiltonian[i][j-int(len(
                    self.zeroth_order_state_space) / 2)] = 0.0
                print(i, j, j-int(len(self.
                zeroth_order_state_space) / 2),
                self.zerothOrderHamiltonian[i][j-int(len(self.
                zeroth_order_state_space) / 2)],
                self.zeroth_order_state_space[i],
                self.zeroth_order_state_space[j])

    def calculateSecondOrderHamiltonian(self):
```

```
        self.secondOrderHamiltonian = np.zeros((int(len(self.
                                        zeroth_order_state_space
                                        ) / 2), int(len(self.
                                        zeroth_order_state_space
                                        ) / 2)))
        print("---------2nd--------------------------")
        for i in range(int(len(self.zeroth_order_state_space) /
                                        2)):
            for j in range(int(len(self.zeroth_order_state_space
                                            ) / 2),
                len(self.zeroth_order_state_space)):
                # check whether the spin is conserved
                spin_init = np.sum(self.zeroth_order_state_space
                                                [j][::2]) -
                np.sum(self.zeroth_order_state_space[j][1::2])
                spin_final = np.sum(self.
                zeroth_order_state_space[i][::2]) -
                np.sum(self.zeroth_order_state_space[i][1::2])
                spin_diff = spin_init - spin_final
                if (spin_diff == 0):
                    initial_state_column_vec = np.zeros((len(
                    self.second_order_state_space), 1))
                    initial_state_column_vec[self.
                    second_order_state_space.index(self.
                    zeroth_order_state_space[j])] = 1
                    denominator = []
                    for num in range(len(
                        self.second_order_state_space)):
                        energy_difference = self.
                        calculateEnergyDifference(self.
                        zeroth_order_state_space[j], self.
                        second_order_state_space[num])
                        if (energy_difference != 0.0):
                            value = [1. / energy_difference]
                            denominator.append(value)
                        else:
                            denominator.append([0.0])
                    final_state = np.matmul(np.multiply(
                                                    denominator,
                    self.zeroth_order_transition_matrix),
                    initial_state_column_vec)
                    second_order_vector = np.matmul(self.
                    zeroth_order_transition_matrix, final_state)
                    self.secondOrderHamiltonian[i][j-int(len(
                    self.zeroth_order_state_space) / 2)] +=
                    0.5 * second_order_vector[self.
                    second_order_state_space.index(self.
                    zeroth_order_state_space[i])]
                    denominator = []
                    for num in range(len(self.
                        second_order_state_space)):
                        energy_difference =
                        self.calculateEnergyDifference(self.
                        zeroth_order_state_space[i],
                        self.second_order_state_space[num])
                        if (energy_difference != 0.0):
                            value = [1. / energy_difference]
```

```
                            denominator.append(value)
                        else:
                            denominator.append([0.0])
                    final_state = np.matmul(np.multiply(
                                                    denominator,
                    self.zeroth_order_transition_matrix),
                    initial_state_column_vec)
                    second_order_vector = np.matmul(self.
                    zeroth_order_transition_matrix,
                    final_state)
                    self.secondOrderHamiltonian[i][j-int(len(
                    self.zeroth_order_state_space) / 2)] +=
                    0.5 * second_order_vector[self.
                    second_order_state_space.index(self.
                    zeroth_order_state_space[i])]
self.zeroth_order_state_space[j])
                else:
                    pass
                print(i, j, j-int(len(self.
                zeroth_order_state_space) / 2),
                self.secondOrderHamiltonian[i][j-int(len(
                self.zeroth_order_state_space) / 2)],
                self.zeroth_order_state_space[i],
                self.zeroth_order_state_space[j])

    def calculateFourthOrderHamiltonian(self):
        self.fourthOrderHamiltonian = np.zeros((int(len(
        self.zeroth_order_state_space) / 2),
        int(len(self.zeroth_order_state_space) / 2)))
        for i in range(int(len(self.
            zeroth_order_state_space) / 2)):
            for j in range(int(len(self.
                zeroth_order_state_space) / 2),
            len(self.zeroth_order_state_space)):
                # check whether the spin is conserved
                spin_init = np.sum(self.
                zeroth_order_state_space[j][::2]) -
                np.sum(self.zeroth_order_state_space[j][1::2])
                spin_final = np.sum(self.
                zeroth_order_state_space[i][::2]) -
                np.sum(self.
                zeroth_order_state_space[i][1::2])
                spin_diff = spin_init - spin_final
                if (spin_diff == 0):
                    # start with the terms with the initial
                                                    state
                                                    included in
                                                    the
                                                    denominator
                    print("------4th-----------")
                    denominator = []
                    denominator_squared = []
                    for num in range(len(self.
                        second_order_state_space)):
                        energy_difference =
                        self.calculateEnergyDifference(self.
                        zeroth_order_state_space[j],
```

```
    self.second_order_state_space[num])
    if (energy_difference != 0.0):
        value = [1. / energy_difference]
        value_squared = [1. /
        (energy_difference ** 2)]
        denominator.append(value)
        denominator_squared.append(value)
    else:
        denominator.append([0.0])
        denominator_squared.append([0.0])
initial_state_column_vec = np.zeros((len(
self.second_order_state_space), 1))
initial_state_column_vec[self.
second_order_state_space.index(self.
zeroth_order_state_space[j])] = 1
first_intermediary_column_vec =
np.matmul(np.multiply(denominator,
self.zeroth_order_transition_matrix),
initial_state_column_vec)
second_intermediary_column_vec =
np.matmul(np.multiply(denominator,
self.zeroth_order_transition_matrix),
first_intermediary_column_vec)
third_intermediary_column_vec =
np.matmul(np.multiply(denominator,
self.zeroth_order_transition_matrix),
second_intermediary_column_vec)
final_state_column_vec =
np.matmul(self.
zeroth_order_transition_matrix,
third_intermediary_column_vec)
self.fourthOrderHamiltonian[i][j-int(len(
self.zeroth_order_state_space) / 2)] +=
0.5 * final_state_column_vec[self.
second_order_state_space.index(
self.zeroth_order_state_space[i])]
second_intermediary_column_vec_back_to_init
                                =
np.matmul(self.
zeroth_order_transition_matrix,
first_intermediary_column_vec)
second_intermediary_column_vec_reinitialize
                                =
np.zeros((len(self.second_order_state_space)
                                , 1))
second_intermediary_column_vec_reinitialize[
                                self.
second_order_state_space.index(self.
zeroth_order_state_space[j])] =
second_intermediary_column_vec_back_to_init[
                                self.
second_order_state_space.index(self.
zeroth_order_state_space[j])]
third_intermediary_column_vec =
np.matmul(np.multiply(denominator_squared,
self.zeroth_order_transition_matrix),
second_intermediary_column_vec_reinitialize)
```

```
final_state_column_vec =
np.matmul(self.
zeroth_order_transition_matrix,
third_intermediary_column_vec)
self.fourthOrderHamiltonian[i][j-int(len(
self.zeroth_order_state_space) / 2)] += - 0.
                                5 *
final_state_column_vec[self.
second_order_state_space.index(self.
zeroth_order_state_space[i])]
# now with the terms with the final state
                                included in
                                the
                                denominator
denominator = []
denominator_squared = []
for num in range(len(self.
    second_order_state_space)):
    energy_difference =
    self.calculateEnergyDifference(self.
    zeroth_order_state_space[i],
    self.second_order_state_space[num])
    if (energy_difference != 0.0):
        value = [1. / energy_difference]
        value_squared = [1. /
        (energy_difference ** 2)]
        denominator.append(value)
        denominator_squared.append(value)
    else:
        denominator.append([0.0])
        denominator_squared.append([0.0])
initial_state_column_vec =
np.zeros((len(self.second_order_state_space)
                                , 1))
initial_state_column_vec[self.
second_order_state_space.index(self.
zeroth_order_state_space[j])] = 1
first_intermediary_column_vec =
np.matmul(np.multiply(denominator,
self.zeroth_order_transition_matrix),
initial_state_column_vec)
second_intermediary_column_vec =
np.matmul(np.multiply(denominator,
self.zeroth_order_transition_matrix),
first_intermediary_column_vec)
third_intermediary_column_vec =
np.matmul(np.multiply(denominator,
self.zeroth_order_transition_matrix),
second_intermediary_column_vec)
final_state_column_vec =
np.matmul(self.
zeroth_order_transition_matrix,
third_intermediary_column_vec)
self.fourthOrderHamiltonian[i][j-int(len(
self.zeroth_order_state_space) / 2)] += 0.5
                                *
final_state_column_vec[self.
```

```
                    second_order_state_space.index(self.
                    zeroth_order_state_space[i])]
                    second_intermediary_column_vec_back_to_init
                                                    =
                    np.matmul(self.
                    zeroth_order_transition_matrix,
                    first_intermediary_column_vec)
                    second_intermediary_column_vec_reinitialize
                                                    =
                    np.zeros((len(self.second_order_state_space)
                                                    , 1))
                    second_intermediary_column_vec[self.
                    second_order_state_space.index(self.
                    zeroth_order_state_space[j])] =
                    second_intermediary_column_vec_back_to_init[
                                                    self.
                    second_order_state_space.index(self.
                    zeroth_order_state_space[j])]
                    third_intermediary_column_vec =
                    np.matmul(np.multiply(denominator_squared,
                    self.zeroth_order_transition_matrix),
                    second_intermediary_column_vec_reinitialize)
                    final_state_column_vec =
                    np.matmul(self.
                    zeroth_order_transition_matrix,
                    third_intermediary_column_vec)
                    self.fourthOrderHamiltonian[i][j-int(len(
                                                    self.
                    zeroth_order_state_space) / 2)] += - 0.5 *
                    final_state_column_vec[self.
                    second_order_state_space.index(self.
                    zeroth_order_state_space[i])]
                    print(self.fourthOrderHamiltonian[i][j-int(
                                                    len(
                    self.zeroth_order_state_space) / 2)],
                    self.zeroth_order_state_space[i],
                    self.zeroth_order_state_space[j])
                else:
                    print("spin not conserved: ", spin_diff)
                    pass

    def getZerothOrderHamiltonian(self):
        return self.zerothOrdereHamiltonian

    def getSecondOrderHamiltonian(self):
        return self.secondOrderHamiltonian

    def getFourthOrderHamiltonian(self):
        return self.fourthOrderHamiltonian

class SubHamiltonian:

    def __init__(self, spin_block):
```

```
        self.states = [[1, 0, 0, 1, 0, 0], [1, 0, 0, 0, 0, 1], [
                                        0, 1, 1, 0, 0, 0], [0, 1
                                        , 0, 0, 1, 0], [0, 1, 0,
                                         1, 0, 0], [0, 1, 0, 0,
                                        0, 1], [1, 0, 1, 0, 0, 0
                                        ], [1, 0, 0, 0, 1, 0]]
        self.spin_block = spin_block
        self.total_states = []
        for i in range(len(self.states)):
            for j in range(len(self.states)):
                self.total_states.append([self.states[i], self.
                                                states[j]])
        self.FeTe = FeTeHamiltonian([self.total_states[self.
                                        spin_block[0]], self.
                                        total_states[self.
                                        spin_block[1]], self.
                                        total_states[self.
                                        spin_block[2]], self.
                                        total_states[self.
                                        spin_block[3]], self.
                                        total_states[self.
                                        spin_block[4]], self.
                                        total_states[self.
                                        spin_block[5]], self.
                                        total_states[self.
                                        spin_block[6]], self.
                                        total_states[self.
                                        spin_block[7]]], 2, "./
                                        params.ini")

    def initialiseFeTe(self):
        self.FeTe.getParameters()
        progressfile = "check_progress-" + str(self.spin_block[0
                                        ]) + "-" + str(self.
                                        spin_block[1]) + "-" +
                                        str(self.spin_block[2])
                                        + "-" + str(self.
                                        spin_block[3]) + "-" +
                                        str(self.spin_block[4])
                                        + "-" + str(self.
                                        spin_block[5]) + "-" +
                                        str(self.spin_block[6])
                                        + "-" + str(self.
                                        spin_block[7]) + ".txt"
        f = open(progressfile,"ab")
        self.FeTe.generateStateSpace()
        for row in self.FeTe.zeroth_order_state_space:
            np.savetxt(f, row)
        self.FeTe.calculateZerothOrderHamiltonian()
        f.close()
```

```
        outputfile1 = "zero-order-calculation-herm-" + str(self.
                                        spin_block[0]) + "-" +
                                        str(self.spin_block[1])
                                        + "-" + str(self.
                                        spin_block[2]) + "-" +
                                        str(self.spin_block[3])
                                        + "-" +  str(self.
                                        spin_block[4]) + "-" +
                                        str(self.spin_block[5])
                                        + "-" + str(self.
                                        spin_block[6]) + "-" +
                                        str(self.spin_block[7])
                                        + ".txt"
        g = open(outputfile1, "ab")
        for row in self.FeTe.zerothOrderHamiltonian:
            np.savetxt(g, row)
        g.close()

        self.FeTe.calculateUnconstrainedTransitionMatrix()

        self.FeTe.calculateSecondOrderHamiltonian()
        print(self.FeTe.secondOrderHamiltonian)

        outputfile2 = "second-order-calculation-herm-" + str(
                                        self.spin_block[0]) + "-
                                        " + str(self.spin_block[
                                        1]) + "-" + str(self.
                                        spin_block[2]) + "-" +
                                        str(self.spin_block[3])
                                        + "-" +  str(self.
                                        spin_block[4]) + "-" +
                                        str(self.spin_block[5])
                                        + "-" + str(self.
                                        spin_block[6]) + "-" +
                                        str(self.spin_block[7])
                                        + ".txt"
        g = open(outputfile2, "ab")
        for row in self.FeTe.secondOrderHamiltonian:
            np.savetxt(g, row)
        g.close()

        self.FeTe.calculateFourthOrderHamiltonian()
        print(self.FeTe.fourthOrderHamiltonian)

        outputfile3 = "fourth-order-calculation-herm-" + str(
                                        self.spin_block[0]) + "-
                                        " + str(self.spin_block[
                                        1]) + "-" + str(self.
                                        spin_block[2]) + "-" +
                                        str(self.spin_block[3])
                                        + "-" +  str(self.
                                        spin_block[4]) + "-" +
                                        str(self.spin_block[5])
                                        + "-" + str(self.
                                        spin_block[6]) + "-" +
                                        str(self.spin_block[7])
                                        + ".txt"
```

```
        g = open(outputfile3, "ab")
        for row in self.FeTe.fourthOrderHamiltonian:
            np.savetxt(g, row)
        g.close()

sub = SubHamiltonian([int(sys.argv[1]), int(sys.argv[2]), int(
                                sys.argv[3]), int(sys.argv[4]),
                                int(sys.argv[5]), int(sys.argv[6
                                ]), int(sys.argv[7]), int(sys.
                                argv[8])])
sub.initialiseFeTe()

FeTe = FeTeHamiltonian([[1,0,0,1,0,0], [0,1,1,0,0,0]], 2, "./
                                params.ini")
FeTe.getParameters()
progressfile = "check_progress.txt"
f = open(progressfile,"ab")
FeTe.generateStateSpace()
for row in FeTe.zeroth_order_state_space:
    np.savetxt(f, row)
FeTe.calculateUnconstrainedTransitionMatrix()
FeTe.calculateZerothOrderHamiltonian()
FeTe.calculateSecondOrderHamiltonian()
FeTe.calculateFourthOrderHamiltonian()
f.close()

outputfile1 = "zero-order-calculation.txt"
g = open(outputfile1, "ab")
for row in FeTe.zerothOrderHamiltonian:
    np.savetxt(g, row)
g.close()
outputfile2 = "second-order-calculation.txt"
g = open(outputfile2, "ab")
for row in FeTe.secondOrderHamiltonian:
    np.savetxt(g, row)
g.close()
outputfile3 = "fourth-order-calculation.txt"
g = open(outputfile3, "ab")
for row in FeTe.fourthOrderHamiltonian:
    np.savetxt(g, row)
g.close()
```

B.6 Script for collating all sub-matrices to construct the 36×36 effective Hamiltonian matrix

The effective Hamiltonian matrix is not immediately outputted by the previous script, but is constructed by the following scripts that collate the elements from the 4×4 sub-matrices and input them into the full effective Hamiltonian matrix. It is the nature of the program that the $\frac{1}{2}(|\uparrow\downarrow\rangle + |\downarrow,\uparrow\rangle)$ single-ion states are treated as two individual single-ion states $|\uparrow\downarrow\rangle$ and $|\downarrow\uparrow\rangle$ with the superposition carried out ex post facto of the Rayleigh-Schrödinger perturbative calculations.

This means that the initial set of single-ion states contains 8 quantum states and produces an effective 64×64 Hamiltonian matrix, which needs to be transformed into the desired 36×36 matrix by superposition of elements of the former matrix. Hence, the following script collates those elements outputted by the previous script and produces the 64×64 matrix:

```
import numpy as np

S_matrix_zeroth = np.zeros(shape=(64, 64))
S_matrix_second = np.zeros(shape=(64, 64))
S_matrix_fourth = np.zeros(shape=(64, 64))

for row in range(16):
    for col in range(16):
        i, j, k, l = row * 4, row * 4 + 1, row * 4 + 2, row * 4
                                            + 3
        m, n, o, p = col * 4, col * 4 + 1, col * 4 + 2, col * 4
                                            + 3
        print(i, j, k, l, m, n, o, p)
        try:
            filename = "zero-order-calculation-herm-" + str(i) +
                                                "-" + str(j) + "-"
                                               + str(k) + "-" + str
                                               (l) + "-" + str(m) +
                                                "-" + str(n) + "-"
                                               + str(o) + "-" + str
                                               (p) + ".txt"
            print(filename, "file does exist")
            f = open(filename)
            content = f.readlines()
            for iter in range(len(content)):
                content[iter] = float(content[iter][:-2])
            content = np.array(content)
            arr = content.reshape((int(len(content) / 4), int(
                                               len(content) / 4)))
            print(arr)
            S_matrix_zeroth[i, m] = arr[0, 0]
            S_matrix_zeroth[i, n] = arr[0, 1]
            S_matrix_zeroth[i, o] = arr[0, 2]
            S_matrix_zeroth[i, p] = arr[0, 3]
            S_matrix_zeroth[j, m] = arr[1, 0]
            S_matrix_zeroth[j, n] = arr[1, 1]
            S_matrix_zeroth[j, o] = arr[1, 2]
            S_matrix_zeroth[j, p] = arr[1, 3]
            S_matrix_zeroth[k, m] = arr[2, 0]
            S_matrix_zeroth[k, n] = arr[2, 1]
            S_matrix_zeroth[k, o] = arr[2, 2]
            S_matrix_zeroth[k, p] = arr[2, 3]
            S_matrix_zeroth[l, m] = arr[3, 0]
            S_matrix_zeroth[l, n] = arr[3, 1]
            S_matrix_zeroth[l, o] = arr[3, 2]
            S_matrix_zeroth[l, p] = arr[3, 3]
        except Exception as e:
            print(e)
        try:
```

```
            filename = "second-order-calculation-herm-" + str(i)
                                             + "-" + str(j) + "-
                                            " + str(k) + "-" +
                                            str(l) + "-" + str(m
                                            ) + "-" + str(n) + "
                                            -" + str(o) + "-" +
                                            str(p) + ".txt"
            print(filename, "file does exist")
            f = open(filename)
            content = f.readlines()
            for iter in range(len(content)):
                content[iter] = float(content[iter][:-2])
            content = np.array(content)
            arr = content.reshape((int(len(content) / 4), int(
                                            len(content) / 4)))
            print(arr)
            S_matrix_second[i, m] = arr[0, 0]
            S_matrix_second[i, n] = arr[0, 1]
            S_matrix_second[i, o] = arr[0, 2]
            S_matrix_second[i, p] = arr[0, 3]
            S_matrix_second[j, m] = arr[1, 0]
            S_matrix_second[j, n] = arr[1, 1]
            S_matrix_second[j, o] = arr[1, 2]
            S_matrix_second[j, p] = arr[1, 3]
            S_matrix_second[k, m] = arr[2, 0]
            S_matrix_second[k, n] = arr[2, 1]
            S_matrix_second[k, o] = arr[2, 2]
            S_matrix_second[k, p] = arr[2, 3]
            S_matrix_second[l, m] = arr[3, 0]
            S_matrix_second[l, n] = arr[3, 1]
            S_matrix_second[l, o] = arr[3, 2]
            S_matrix_second[l, p] = arr[3, 3]
        except:
            print("file does not exist")
        try:
            filename = "fourth-order-calculation-herm-" + str(i)
                                             + "-" + str(j) + "-
                                            " + str(k) + "-" +
                                            str(l) + "-" + str(m
                                            ) + "-" + str(n) + "
                                            -" + str(o) + "-" +
                                            str(p) + ".txt"
            print(filename, "file does exist")
            f = open(filename)
            content = f.readlines()
            for iter in range(len(content)):
                content[iter] = float(content[iter][:-2])
            content = np.array(content)
            arr = content.reshape((int(len(content) / 4), int(
                                            len(content) / 4)))
            print(arr)
            S_matrix_fourth[i, m] = arr[0, 0]
            S_matrix_fourth[i, n] = arr[0, 1]
            S_matrix_fourth[i, o] = arr[0, 2]
            S_matrix_fourth[i, p] = arr[0, 3]
            S_matrix_fourth[j, m] = arr[1, 0]
            S_matrix_fourth[j, n] = arr[1, 1]
```

```
            S_matrix_fourth[j, o] = arr[1, 2]
            S_matrix_fourth[j, p] = arr[1, 3]
            S_matrix_fourth[k, m] = arr[2, 0]
            S_matrix_fourth[k, n] = arr[2, 1]
            S_matrix_fourth[k, o] = arr[2, 2]
            S_matrix_fourth[k, p] = arr[2, 3]
            S_matrix_fourth[l, m] = arr[3, 0]
            S_matrix_fourth[l, n] = arr[3, 1]
            S_matrix_fourth[l, o] = arr[3, 2]
            S_matrix_fourth[l, p] = arr[3, 3]
        except:
            print("file does not exist")

np.savetxt("S_matrix_zeroth.txt", S_matrix_zeroth)
np.savetxt("S_matrix_second.txt", S_matrix_second)
np.savetxt("S_matrix_fourth.txt", S_matrix_fourth)

S_matrix = S_matrix_zeroth + S_matrix_second + S_matrix_fourth

np.savetxt("S_matrix.txt", S_matrix)
```

The script for converting the 64×64 matrix into the 36×36 effective Hamiltonian matrix is also shown below:

```
import numpy as np
import pandas as pd

S_matrix_zeroth_64 = np.loadtxt("S_matrix_zeroth.txt")
S_matrix_second_64 = np.loadtxt("S_matrix_second.txt")
S_matrix_fourth_64 = np.loadtxt("S_matrix_fourth.txt")
S_matrix_64 = np.loadtxt("S_matrix.txt")

print(S_matrix_zeroth_64.shape)

S_matrix_zeroth_36 = np.zeros(shape=(36, 36))
S_matrix_second_36 = np.zeros(shape=(36, 36))
S_matrix_fourth_36 = np.zeros(shape=(36, 36))
S_matrix_36 = np.zeros(shape=(36, 36))

def conversion(x):
    if (x == 0):
        y = [6]
    elif (x == 1):
        y = [4]
    elif (x == 2):
        y = [7]
    elif (x == 3):
        y = [5]
    elif (x == 4):
        y = [0, 2]
    elif (x == 5):
        y = [1, 3]
    return y

for i in range(36):
    for j in range(36):
        print("i and j: ", i, j)
```

```
        Fe1_row_36 = i // 6
        Fe2_row_36 = i % 6
        Fe1_col_36 = j // 6
        Fe2_col_36 = j % 6
        Fe1_row_64 = conversion(Fe1_row_36)
        Fe2_row_64 = conversion(Fe2_row_36)
        Fe1_col_64 = conversion(Fe1_col_36)
        Fe2_col_64 = conversion(Fe2_col_36)
        i_64 = []
        j_64 = []
        for m in range(len(Fe1_row_64)):
            for n in range(len(Fe2_row_64)):
                i_64.append(Fe1_row_64[m] * 8 + Fe2_row_64[n])
        for o in range(len(Fe1_col_64)):
            for p in range(len(Fe2_col_64)):
                j_64.append(Fe1_col_64[o] * 8 + Fe2_col_64[p])
        print("i_64 and j_64: ", i_64, j_64)
        value_zeroth = 0
        value_second = 0
        value_fourth = 0
        value_total = 0
        for q in range(len(i_64)):
            for r in range(len(j_64)):
                print("q and r: ", q, r)
                print("S_matrix_zeroth_64[i_64[q], j_64[r]]: ",
                S_matrix_zeroth_64[i_64[q], j_64[r]])
                value_zeroth += ((1 / (np.sqrt(len(i_64) * len(
                j_64)))) * S_matrix_zeroth_64[i_64[q], j_64[r]])
                value_second += ((1 / (np.sqrt(len(i_64) * len(
                j_64)))) * S_matrix_second_64[i_64[q], j_64[r]])
                value_fourth += ((1 / (np.sqrt(len(i_64) * len(
                j_64)))) * S_matrix_fourth_64[i_64[q], j_64[r]])
                value_total += ((1 / (np.sqrt(len(i_64) * len(
                j_64)))) * S_matrix_64[i_64[q], j_64[r]])
                print("value_zeroth: ", value_zeroth)
                print("value_second: ", value_second)
                print("value_fourth: ", value_fourth)
                print("value_total: ", value_total)
        S_matrix_zeroth_36[i, j] = value_zeroth
        S_matrix_second_36[i, j] = value_second
        S_matrix_fourth_36[i, j] = value_fourth
        S_matrix_36[i, j] = value_total
        print("S_matrix_zeroth_36[i, j]: ", S_matrix_zeroth_36[i
                                        , j])

df_zeroth_36 = pd.DataFrame(S_matrix_zeroth_36)
df_second_36 = pd.DataFrame(S_matrix_second_36)
df_fourth_36 = pd.DataFrame(S_matrix_fourth_36)
df_36 = pd.DataFrame(S_matrix_36)

df_zeroth_36.to_csv("S_matrix_zeroth_36.txt", sep=" ", index=
                                True, header=True)
df_second_36.to_csv("S_matrix_second_36.txt", sep=" ", index=
                                True, header=True)
df_fourth_36.to_csv("S_matrix_fourth_36.txt", sep=" ", index=
                                True, header=True)
```

```
df_36.to_csv("S_matrix_36.txt", sep=" ", index=False, header=
                                 False)
```

The effective 36×36 Hamiltonian matrix must have the correct order of orthogonal basis states in order for the decomposition according to Sec. 4.4 to work. The following script transforms the Hamiltonian so that it has the requisite form:

```
import numpy as np
import pandas as pd

original_matrix = pd.read_csv("S_matrix_36.txt", header=None,
                                 index_col=False, sep=" ").values

print(original_matrix.shape)

def old_ordering():
    old_basis_states = []
    single_ion_states = [[1, -0.5], [-1, -0.5], [1, 0.5], [-1, 0
                             .5], [0, -0.5], [0, 0.5]]
    combined_states = []
    for i in range(6):
        for j in range(6):
            basis_state = single_ion_states[i] +
                                    single_ion_states[j]
            combined_states.append(basis_state)
    return combined_states

def new_ordering():
    new_basis_states = []
    def spin(x):
        if x in [0]:
            y = 1
        elif x in [1]:
            y = 0
        elif x in [2]:
            y = -1
        else:
            raise Exception("input must be between 0 and 5")
        return y
    def pseudospin(x):
        if (x==0):
            y = 0.5
        elif (x==1):
            y = -0.5
        else:
            raise Exception("input must be between 0 and 1")
        return y
    for i in range(3):
        for j in range(2):
            for k in range(3):
                for l in range(2):
                    print(i, j, k, l)
                    new_basis_state = [spin(i), pseudospin(j),
                                                spin(k),
                                                pseudospin(l
                                                )]
```

```
                    new_basis_states.append(new_basis_state)
    return new_basis_states

old_order = old_ordering()
new_order = new_ordering()

print(old_order)
print(new_order)

new_indices = [x for x in range(36)]
old_indices = []

for i in range(len(new_indices)):
    new_basis_state = new_order[i]
    old_index = old_order.index(new_basis_state)
    old_indices.append(old_index)

print(new_indices)
print(old_indices)

permutation_matrix = np.zeros([36, 36])
print(permutation_matrix)

for i in range(36):
    permutation_matrix[i, old_indices[i]] = 1

print(permutation_matrix[0, :])

transformed_matrix = np.matmul(permutation_matrix, np.matmul(
                                original_matrix, np.linalg.inv(
                                permutation_matrix)))

print(transformed_matrix)

df = pd.DataFrame(transformed_matrix)

df.to_csv("transformed_matrix.txt", sep=" ", index=False, header
                                =False)
```

B.7 Script for decomposing the effective Hamiltonian into a sum of Kronecker products

The effective 36×36 Hamiltonian matrix is decomposed into a sum of Kronecker products of 9 spin and 4 orbital pseudo-spin operators that each individually act on either of the two coupled Fe atoms partaking in the bond. The weights for each of the resultant 1296 weights are computed using a script that follows the procedure outlined in Sec.4.4.

```
import numpy as np
import pandas as pd

hamiltonian = pd.read_csv("transformed_matrix.txt", header=None,
                                 index_col=False, sep=" ").
                                values
```

```
S_0 = np.array([[1, 0, 0], [0, 1, 0], [0, 0, 1]])
S_1 = np.array([[1, 0, 0], [0, 0, 0], [0, 0, -1]])
S_2 = np.array([[0, np.sqrt(2), 0], [0, 0, np.sqrt(2)], [0, 0, 0
                                ]])
S_3 = np.array([[0, 0, 0], [np.sqrt(2), 0, 0], [0, np.sqrt(2), 0
                                ]])
S_4 = np.array([[1, 0, 0], [0, -2, 0], [0, 0, 1]])
S_5 = np.matmul(S_2, S_2)
S_6 = np.matmul(S_3, S_3)
S_7 = np.matmul(S_1, S_2) - np.matmul(S_2, S_1)
S_8 = np.matmul(S_1, S_3) - np.matmul(S_3, S_1)

T_0 = np.array([[1, 0], [0, 1]])
T_1 = np.array([[1, 0], [0, -1]])
T_2 = np.array([[0, 1], [0, 0]])
T_3 = np.array([[0, 0], [1, 0]])

def decomposition(a, b, c, d):
    def spin(num):
        if (num==0):
            return S_0
        elif (num==1):
            return S_1
        elif (num==2):
            return S_2
        elif (num==3):
            return S_3
        elif (num==4):
            return S_4
        elif (num==5):
            return S_5
        elif (num==6):
            return S_6
        elif (num==7):
            return S_7
        elif (num==8):
            return S_8
        else:
            pass
    def pseudospin(num):
        if (num==0):
            return T_0
        elif (num==1):
            return T_1
        elif (num==2):
            return T_2
        elif (num==3):
            return T_3
    Q_matrix = np.kron(spin(a), np.kron(pseudospin(b), np.kron(
                                    spin(c), pseudospin(d))))
    coefficient = (1 / np.trace(np.matmul(Q_matrix.transpose(),
                                    Q_matrix))) * np.trace(np.
                                    matmul(Q_matrix.transpose(),
                                     hamiltonian))
    return coefficient
```

```
for i in range(9):
    for j in range(4):
        for k in range(9):
            for l in range(4):
                string = "A_{ " + str(i) + ", " + str(j) + ", "
                                                + str(k) + ", "
                                                + str(l) + "}: "
                                                 + str(
                                                decomposition(i,
                                                 j, k, l))
                print(string)
```

Bibliography

1. Chadwick, J. The neutron and its properties. *The British Journal of Radiology* **6,** 24–32 (1933).
2. De Broglie, L. *Recherches sur la théorie des Quanta* Theses (Migration - université en cours d'affectation, Nov. 1924). `https://tel.archives-ouvertes.fr/tel-00006807`.
3. Boothroyd, A. T. *Principles of neutron scattering from condensed matter / Andrew T. Boothroyd.* First edition. eng. ISBN: 9780191895081 (Oxford University Press, Oxford, 2020).
4. Shull, C. G, Strauser, W. A & Wollan, E. O. Neutron Diffraction by Paramagnetic and Antiferromagnetic Substances. eng. *Physical review* **83,** 333–345. ISSN: 0031-899X (1951).
5. Shull, C. G. Early development of neutron scattering. eng. *Reviews of modern physics* **67,** 753–757. ISSN: 0034-6861 (1995).
6. Pynn, R. in *Neutron Applications in Earth, Energy and Environmental Sciences* (Springer US, Boston, MA). ISBN: 9780387094151.
7. Shirane, G. *Neutron Scattering with a Triple-Axis Spectrometer : Basic Techniques / Gen Shirane, Stephen M. Shapiro, John M. Tranquada.* eng. ISBN: 9780511534881 (Cambridge University Press, Cambridge, 2002).
8. Squires, G. L. G. L. *Introduction to the theory of thermal neutron scattering / G.L. Squires (1924-2010), lecturer in physics at the University of Cambridge and Fellow of Trinity College, Cambridge.* Third edition. eng. ISBN: 9781107644069 (Cambridge University Press, Cambridge, 2012).
9. Sears, V. F. Neutron scattering lengths and cross sections. *Neutron News* **3,** 26–37 (1992).
10. KOSTORZ, G. & LOVESEY, S. in *Neutron Scattering* (ed Kostorz, G.) 1–67 (Elsevier, 1979).
11. *Encyclopedia of condensed matter physics / edited by Franco Bassani, Gerald L. Liedl, Peter Wyder.* eng. ISBN: 9780122276101 (Elsevier, Amsterdam, 2005).
12. Thomason, J. The ISIS Spallation Neutron and Muon Source—The first thirty-three years. *Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment* **917,** 61–67. ISSN: 0168-9002. `https://www.sciencedirect.com/science/article/pii/S0168900218317820` (2019).

13. Mamontov, E. *et al.* A concept of a broadband inverted geometry spectrometer for the Second Target Station at the Spallation Neutron Source. *Review of Scientific Instruments* **93,** 045101. eprint: `https://doi.org/10.1063/5.0086451`. `https://doi.org/10.1063/5.0086451` (2022).
14. Tremsin, A. S *et al.* High Resolution Neutron Resonance Absorption Imaging at a Pulsed Neutron Beamline. *IEEE transactions on nuclear science* **59,** 3272–3277. ISSN: 0018-9499 (2012).
15. Stuhr, U. *et al.* The thermal triple-axis-spectrometer EIGER at the continuous spallation source SINQ. *Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment* **853,** 16–19. ISSN: 0168-9002. `https://www.sciencedirect.com/science/article/pii/S0168900217301833` (2017).
16. MERLIN, a new high count rate spectrometer at ISIS. eng. *Physica. B, Condensed matter* **385,** 1029–1031. ISSN: 0921-4526 (2006).
17. *Isis how Isis Works - In Depth* `https://www.isis.stfc.ac.uk/Pages/How-ISIS-works--in-depth.aspx`.
18. *Description of EIGER* `https://www.psi.ch/en/sinq/eiger/description` (2022).
19. *The SINQ neutron source* `https://www.psi.ch/en/media/the-sinq-neutron-source` (2022).
20. Viehland, D., Jang, S. J. & Cross, L. E. *Phys. Rev. B* **46,** 8003 (1992).
21. Viehland, D., Jang, S. J., Cross, L. E. & Wuttig, M. *J. Appl. Phys.* **68,** 2016 (1990).
22. Ye, Z.-G. *Key Eng. Mater.* **155-156,** 81 (1998).
23. Ye, Z.-G. *MRS Bull.* **34,** 277 (2009).
24. Xu, G. *J. Phys. Soc. Jpn.* **79,** 011011 (2010).
25. Cowley, R. A., Gvasaliya, S. N., Lushnikov, S. G., Roessli, B. & Rotaru, G. M. *Adv. Phys.* **60,** 229 (2011).
26. Park, S. E. & Shrout, T. R. *J. Appl. Phys.* **82,** 1804 (1997).
27. Kelly, J. M. *A study of electromechanical properties of PMN-PT ceramics and analysis of the effects of loss on frequency response of piezoelectric ceramics* 1998.
28. Bhakar, A. *et al.* *Study of Structural Disorder in Pb(Mg1/3Nb2/3)O-3* in. **1512** (Feb. 2013), 512–513.
29. Ye, Z.-G. & Schmid, H. *Ferroelectrics* **145,** 83 (1993).
30. Xu, G. *et al.* *Phys. Rev. B* **67,** 104102 (2003).
31. Xu, G. *et al.* *Phys. Rev. B* **70,** 064107 (2004).
32. Conlon, K. H. *et al.* *Phys. Rev. B* **70,** 172204 (2004).
33. Vakhrushev, S. B., Naberezhnov, A. A., Okuneva, N. M. & Savenko, B. N. *Phys. Solid State* **40,** 1728 (1998).

34. Stock, C. *et al. Phys. Rev. B* **76,** 064122 (2007).
35. You, H. & Zhang, Q. M. *Phys. Rev. Lett.* **79,** 3950 (1997).
36. Vakhrushev, S. B., Naberezhnov, A. A., Okuneva, N. M. & Savenko, B. N. *Phys. Solid State* **37,** 1993 (1995).
37. Hirota, K., Ye, Z.-G., Wakimoto, S., Gehring, P. M. & Shirane, G. *Phys. Rev. B* **65,** 104105 (2002).
38. Hiraka, H., Lee, S.-H., Gehring, P. M., Xu, G. & Shirane, G. *Phys. Rev. B* **70,** 184105 (2004).
39. Xu, G., Shirane, G., Copley, J. R. D. & Gehring, P. M. *Phys. Rev. B* **69,** 064112 (2004).
40. Burns, G. & Dacol, F. H. *Solid State Commun.* **48,** 853 (1983).
41. Gehring, P. M. *et al. Phys. Rev. B* **79,** 224109 (2009).
42. Stock, C. *et al. Phys. Rev. B* **81,** 144127 (2010).
43. Vakhrushev, S. B., Ivanov, A. & Kulda, J. *Phys. Chem. Chem. Phys.* **7,** 2340 (2005).
44. Naberezhnov, A., Vakhrushev, S., Dorner, B., Strauch, D. & Moudden, H. *Eur. Phys. J. B* **11,** 13 (1999).
45. Hlinka, J. & Kempa, M. *Phase Transitions* **81,** 491 (2008).
46. Westphal, V., Kleemann, W. & Glinchuk, M. D. *Phys. Rev. Lett.* **68,** 847 (1992).
47. Stock, C. *et al. Phys. Rev. B* **69,** 094104 (2004).
48. Fisch, R. *Phys. Rev. B* **67,** 094110 (2003).
49. Gvasaliya, S. N., Roessli, B., Cowley, R. A., Huber, P. & Lushnikov, S. G. *J. Phys. Condens. Matter* **17,** 4343 (2005).
50. Gehring, P. M., Park, S.-E. & Shirane, G. *Phys. Rev. B* **63,** 224109 (2001).
51. Gehring, P. M., Wakimoto, S., Ye, Z. G. & Shirane, G. *Phys. Rev. Lett.* **87,** 277601 (2001).
52. Wakimoto, S. *et al. Phys. Rev. B* **65,** 172105 (2002).
53. Gehring, P. M., Park, S.-E. & Shirane, G. *Phys. Rev. Lett.* **84,** 5216–5219 (22 2000).
54. Koo, T. Y. *et al. Phys. Rev. B* **65,** 144113 (2002).
55. Stock, C. *et al. Phys. Rev. B* **86,** 104108 (2013).
56. Stock, C. *et al. Phys. Rev. B* **73,** 064107 (2006).
57. Hlinka, J. *et al. Phys. Rev. Lett.* **91,** 107602 (2003).
58. Wakimoto, S. *et al. Phys. Rev. B* **66,** 224102 (2002).
59. Axe, J. D., Harada, J. & Shirane, G. *Phys. Rev. B* **1,** 1227–1234 (1970).
60. Shirane, G., Axe, J. D., Harada, J. & Linz, A. *Phys. Rev. B* **2,** 3651 (1970).

61. Yamada, Y. & Shirane, G. *J. Phys. Soc. Jpn.* **26,** 396 (1969).
62. Stock, C. *et al. J. Phys. Soc. Jpn.* **74,** 3002 (2005).
63. Stock, C. *et al. Phys. Rev. Materials* **2,** 024404 (2018).
64. Taniguchi, H., Itoh, M. & Fu, D. S. *J. Raman Spectrosc.* **42,** 706 (2011).
65. Dorner, B. *et al. Ferroelectrics* **282,** 9 (2003).
66. Lima, J. A. *et al. J. Raman Spectrosc.* **40,** 1144 (2009).
67. Tomeno, I., Ishii, Y., Tsunoda, Y. & Oka, K. *Phys. Rev. B* **73,** 064116 (2006).
68. Kempa, M. *et al. Phase Transitions* **79,** 351 (2006).
69. Rodriguez, E. E. *et al.* Magnetic and structural properties near the Lifshitz point in $Fe_{1+x}Te$. *Phys. Rev. B* **88,** 165110. `https://link.aps.org/doi/10.1103/PhysRevB.88.165110` (16 2013).
70. Rodriguez, E. E. *et al.* Magnetic-crystallographic phase diagram of the superconducting parent compound $Fe_{1+x}Te$. *Phys. Rev. B* **84,** 064403. `https://link.aps.org/doi/10.1103/PhysRevB.84.064403` (6 2011).
71. Koz, C., Rößler, S., Tsirlin, A. A., Wirth, S. & Schwarz, U. Low-temperature phase diagram of Fe1+yTe studied using x-ray diffraction. eng. *Physical review. B, Condensed matter and materials physics* **88.** ISSN: 1098-0121 (2013).
72. Fang, C., Andrei Bernevig, B & Hu, J. Theory of magnetic order in Fe1+yTe1- xSex. eng. *Europhysics letters* **86,** 67005–. ISSN: 0295-5075 (2009).
73. Lipscombe, O. J *et al.* Spin waves in the (π,0) magnetically ordered iron chalcogenide $Fe_{1.05}Te$. eng. *Physical review letters* **106,** 057004–057004. ISSN: 1079-7114 (2011).
74. Turner, A. M., Wang, F. & Vishwanath, A. Kinetic magnetism and orbital order in iron telluride. *Phys. Rev. B* **80,** 224504. `https://link.aps.org/doi/10.1103/PhysRevB.80.224504` (22 2009).
75. Tam, D. W. *et al.* Plaquette instability competing with bicollinear ground state in detwinned FeTe. *Phys. Rev. B* **100,** 054405. `https://link.aps.org/doi/10.1103/PhysRevB.100.054405` (5 2019).
76. Huang, H. Y *et al.* Jahn-Teller distortion driven magnetic polarons in magnetite. eng. *Nature communications* **8,** 15929–15929. ISSN: 2041-1723 (2017).
77. Trainer, C. *et al.* Magnetic surface reconstruction in the van der Waals antiferromagnet Fe1+xTe. eng (2021-01-06).
78. Kruger, F, Kumar, S, Zaanen, J & Brink, J. v. d. Spin-orbital frustrations and anomalous metallic state in iron-pnictide superconductors. eng. *Physical review. B, Condensed matter and materials physics* **79,** 054504–. ISSN: 1098-0121 (2009).

79. Merritt, A. *et al.* Nematic Correlation Length in Iron-Based Superconductors Probed by Inelastic X-Ray Scattering. *Physical Review Letters* **124.** ISSN: 1079-7114. `http://dx.doi.org/10.1103/PhysRevLett.124.157001` (2020).
80. Tomaszewski, P. E. Jan Czochralski—father of the Czochralski method. *Journal of Crystal Growth* **236,** 1–4. ISSN: 0022-0248. `https://www.sciencedirect.com/science/article/pii/S0022024801021959` (2002).
81. Wen, J., Xu, G., Gu, G., Tranquada, J. M. & Birgeneau, R. J. Interplay between magnetism and superconductivity in iron-chalcogenide superconductors: crystal growth and characterizations. eng. *Reports on progress in physics* **74,** 124503–. ISSN: 0034-4885 (2011).
82. Rodríguez-Carvajal, J. Recent advances in magnetic structure determination by neutron powder diffraction. *Physica.* **192.** ISSN: 0921-4526 (1993-10).
83. Nematic Correlation Length in Iron-Based Superconductors Probed by Inelastic X-Ray Scattering. eng. *Physical review letters* **124,** 157001–157001. ISSN: 0031-9007 (2020).
84. Maheshwari, P. K, Jha, R., Gahtori, B. & Awana, V. P. S. Structural and Magnetic Properties of Flux-Free Large FeTe Single Crystal. eng. *Journal of superconductivity and novel magnetism* **28,** 2893–2897. ISSN: 1557-1939 (2015).
85. Robertson, J. H. Physical properties of crystals: their representation by tensors and matrices by J. F. Nye. eng. *Acta crystallographica. Section A, Foundations of crystallography* **41,** 624–624. ISSN: 0108-7673 (1985).
86. Fernandes, R. M. *et al.* Effects of Nematic Fluctuations on the Elastic Properties of Iron Arsenide Superconductors. *Phys. Rev. Lett.* **105,** 157003. `https://link.aps.org/doi/10.1103/PhysRevLett.105.157003` (15 2010).
87. Momma, K. & Izumi, F. *VESTA*: a three-dimensional visualization system for electronic and structural analysis. *Journal of Applied Crystallography* **41,** 653–658. `https://doi.org/10.1107/S0021889808012016` (2008).
88. S. J. Clark M. D. Segall, C. J. P. P. J. H. M. J. P. K. R. M. C. P. First principles methods using CASTEP. *Zeitschrift fuer Kristallographie* **220,** 567–570 (2005).
89. Lindgren, I. The Rayleigh-Schrodinger perturbation and the linked-diagram theorem for a multi-configurational model space. eng. *Journal of physics. B, Atomic and molecular physics* **7,** 2441–2470. ISSN: 0022-3700 (1974).
90. Leciejewicz, J., Samuelson, O, Holm, A, Nielsen, P. & Munch-Petersen, J. A neutron diffraction study of magnetic ordering in Iron Telluride. *Acta Chem. Scand* **17,** 2593–2599 (1963).
91. Li, S. *et al.* First-order magnetic and structural phase transitions in $\mathrm{Fe}_{1+y}\mathrm{Se}_x\mathrm{Te}_{1-x}$. *Phys. Rev. B* **79,** 054503. `https://link.aps.org/doi/10.1103/PhysRevB.79.054503` (5 2009).

92. Des Cloizeaux, J. Extension d'une formule de Lagrange à des problèmes de valeurs propres. *Nuclear Physics* **20,** 321–346 (1960).
93. QUANTUM ESPRESSO: a modular and open-source software project for quantum simulations of materials. eng. *Journal of physics. Condensed matter* **21,** 395502–395502. ISSN: 0953-8984 (2009).
94. Gvasaliya, S., Strauch, D., Dorner, B., Lushnikov, S. & Vakhrushev, S. Lattice Dynamics of PbMg 1/3 Nb 2/3 O 3 (PMN): Shell-Model Calculations. *Ferroelectrics* **282,** 21–27 (2003).
95. Cowley, R. A. Lattice Dynamics and Phase Transitions of Strontium Titanate. *Phys. Rev.* **134,** A981–A997. `https://link.aps.org/doi/10.1103/PhysRev.134.A981` (4A 1964).
96. Dorner, B *et al.* Phonons in PbMg 1/3 Nb 2/3 O 3 Measured by Inelastic Neutron Scattering. eng. *Ferroelectrics* **282,** 9–19. ISSN: 0015-0193 (2003).
97. Giannozzi, P. *et al. J. Phys. Condens. Matter* **21,** 395502 (2009).
98. Giannozzi, P *et al. J. Phys. Condens. Matter* **29,** 465901 (2017).
99. Swainson, I. P *et al.* Soft phonon columns on the edge of the Brillouin zone in the relaxor PbMg1/3Nb2/3O3. eng. *Physical review. B, Condensed matter and materials physics* **79.** ISSN: 1098-0121 (2009).
100. Ewings, R. *et al.* Horace: Software for the analysis of data from single crystal spectroscopy experiments at time-of-flight neutron instruments. eng. *Nuclear instruments methods in physics research. Section A, Accelerators, spectrometers, detectors and associated equipment* **834,** 132–142. ISSN: 0168-9002 (2016).
101. Prosandeec, S. A *et al.* Lattice dynamics in PbMg1/3Nb2/3O3. rus. *Physical review. B, Condensed matter and materials physics* **70,** 134110.1–134110.11. ISSN: 1098-0121 (2004).