AN INVESTIGATION INTO TENSOR BASED MAGNETIC FIELD FORWARD MODELLING, AND SOURCE DETECTION.

Patrick Cole

A Thesis submitted to the Faculty of Science, University of the Witwatersrand, Johannesburg, in fulfilment of the requirements for the degree of Doctor of Philosophy

Johannesburg, 2018

DECLARATION

I declare that this Theses is my own, unaided work. It is being submitted for the Degree of Doctor of Philosophy at the University of the Witwatersrand, Johannesburg. It has not been submitted before for any degree or examination at any other University.

_____

(Signature of candidate)

24 day of October, 2018, in Pretoria.

ABSTRACT

The modelling of potential field data is often both time consuming and ambiguous. With the use of tensor datasets becoming more commonplace, efficient techniques to model these data are necessary. The two components of modelling, namely forward modelling and inversion are addressed. The derivation of tensor forward modelling equations is illustrated and it is shown that the forward modelling of voxel data, both in conventional and tensor form, is not only viable but also has efficiencies which are as good, if not better (when taking editing into account) than non-voxel techniques. The theoretical basis for this modelling is presented here.

Source distance calculations is a form of inversion that provides a good starting model in an efficient way. New tensor equations utilising analytic signals were derived for both source distance and susceptibility calculations. The tensor forms offer the possibility of lower noise in the calculations when dealing with tensor data. The synthesis of results from these techniques into a final model is semi-automated and includes using cluster analysis methods such as DBSCAN. This allows for automatic determination of relevant features from the depth calculations.

The presence of remanent magnetisation in data often presents problems in forward and inverse modelling. By deriving novel equations to directly calculate magnetic field direction cosines from the tensor magnetic components, it is possible to get an indication of the presence of remanence, direction of the remanent field and the complexity of remanence within bodies. Tests on real tensor data over the Tallawang deposit in Australia showed both strengths and limitations. In spite of not being a perfect dyke, calculations for depth and width produced solutions in the expected range. Direction cosine solutions over the body show a degree of complexity in the remanence, possibly due to the presence of magnetite in lenses, thereby suggesting a complex composition. The low Q-ratio and uncertainty in susceptibility for the area contributed to non-optimal solutions for total magnetisation, remanent magnetisation, inclination and declination. Synthetic modelling demonstrates that should the total magnetisation and susceptibility be accurately known, it is possible to accurately derive remanent magnetisation, inclination and declination.

The use of actual tensor data as well as the derivation of tensor datasets from total magnetic intensity data showed that the process derived in this project not only is viable, but also achieves good results. The extraction of valid source distance solutions from raster data is straightforward and allows fast creation of the 3D starter model for the area, from which improvements can be made through further forward modelling.

Dedicated to my wife and best friend,

Janine Cole

ACKNOWLEDGEMENTS

I would like to acknowledge the Council for Geoscience for providing infrastructure and data with which to do this study.

I would like to acknowledge Dr David Clark at the CSIRO for providing tensor data for the study.

I would like to thank my supervisor, Prof Gordon Cooper for his guidance.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

CHAPTER 1 INTRODUCTION

1.1 General Introduction

Airborne geophysical surveying has resulted in vast quantities of potential field data that are available globally. The ongoing demand to find more resources has meant that the efficient modelling and interpretation of this form of data (especially in three dimensions) is more critical than ever. The modelling ambiguity of potential field data, coupled with the sheer volume collected has created significant delays in an industry that more than ever is demanding not only 2D but also 3D models of geology. The modelling of such data has made great strides since the work of Bhattacharyya (1964), with his derivation of magnetic anomalies from prism shaped bodies, Talwani (1965), who developed magnetic anomalies from arbitrary shapes and Nabighian (1972), with his paper on the analytic signal of two dimensional bodies which paved the way for future studies on analytic signals. More recent examples are Guptasarma and Singh (1999) who derived equations for magnetic anomalies of arbitrary shaped polygons and Cooper (1997), who demonstrated the forward modelling of data within modern software, have also made advances in modelling magnetic and gravity data. Holstein, FitzGerald and Stefanov, (2013) developed optimized formula and code for the gravity and magnetic effect of a homogenous prismatic target, by taking advantage of advances made for the general polyhedral case.

However, there is need for an efficient overall modelling strategy needed to model such data. In addition to the ambiguity of the data, some techniques introduce noise into the modelling process, further exacerbating the inherent modelling difficulties.

The modelling of potential field data, specifically gravity and magnetic data, is broadly subdivided into two subjects, namely forward modelling and inverse modelling. In forward modelling, potential field data is calculated by manually postulating geological parameters such as density, susceptibility and the geometry of the body, which is being modelled. This resultant field is compared with the measured field, and the process is repeated until the calculated and measured fields match. Forward modelling is as reliable as the human input and needs trial and error. The calculation of this model in three dimensions can be time consuming, so this thesis will examine the optimisation of the modelling process.

In inversion, the measured field is accepted as input, and some or all of the susceptibility, density and geometry values are calculated automatically. Due to the inherent ambiguity of potential field data, finding the correct solutions for model values is challenging. One technique used to overcome some of this problem is semiautomatic source detection.

Source detection determines parameters such as the depth to different source types (e.g. dykes, contacts etc) in different ways, often using the derivatives or Hilbert transforms of the data (Cooper and Whitehead, 2016).

Despite the excellent work of authors such as Bhattacharyya (1964), Talwani (1965), Nabighian (1972), Guptasarma and Singh (1999), Cooper (1997) and Holstein, FitzGerald and Stefanov (2013) in examining different aspects of modelling; the holistic gathering of techniques such as source distance and forward modelling, while balancing each technique's strengths and weaknesses against each other, is an area where more work is needed.

Without the development of these strategies and the techniques to be used by the strategies, the full set of possibilities of potential field data will not be achieved and it will remain consigned to the history book of "cheaper datasets to be collected first".

This study will address this gap by looking at the modelling process from a practical point of view, of not only model creation, but also editing. It will examine each of the broad modelling facets – whether forward modelling, inversion or source detection – and develop a strategy to maximise the strengths of each form of modelling while minimising the weaknesses. It will also look into developing modelling techniques involving the newer tensor or gradiometer data. This will take the form of forward modelling and source distance estimation techniques. It will assess and develop new techniques in relation to various forward models – such as dykes, contacts and dipoles.

Ultimately, the benefits will include:

1) More efficient modelling process
2) Reduced sensitivity to noise by using tensor data where possible and improved techniques
3) Better source location accuracy and model accuracy in general.

1.2 Objectives

Aim: To develop an overall strategy in the modelling of potential field data (but more specifically magnetic data) and to examine forward modelling and source detection techniques as applied to magnetic tensor data. This will include the following:

1) The development of a modelling process, which allows for the smooth and efficient creation and editing of geological models.
2) The integration of multiple modelling techniques – whether forward modelling, inversion and source detection – into the modelling pipeline
3) The enhancement of these techniques to use tensor data.
4) The minimisation of the effects of noise in the modelling process (specifically with source detection techniques)

Hypothesis: The overall development of a strategy to model potential field data will improve model accuracy and efficiency. The modelling of magnetic tensor data gives extra insight into the nature of potential fields. Therefore, it should improve inversion through source detection and forward modelling of potential fields.

Questions:

1) Can voxel based forward modelling be as efficient as, or more efficient than, modelling geometrical bodies?
2) Can forward modelling and inversion be integrated to improve each technique's strengths while minimising the weaknesses?
3) Can tensor data overcome some of the ambiguity of forward and/or inverse modelling?
4) Can source detection be improved with tensor data?
5) What insight can the individual tensor components give to forward modelling and source detection?

1.3 Data Availability and Methodology

Magnetic and gravity data is freely available from the Council for Geoscience, for the entire South Africa at a regional scale. High-resolution magnetic data is also available for selected portions of the country. For the purposes of this study, the study area is not as important as a pre-existing knowledge of the area in order to test the validity of any models calculated.

Theoretical tensor data will be used to test algorithms, with a final demonstration of algorithms using a tensor dataset derived from conventional total magnetic intensity data. The process is described by Pedersen, Rasmussen and Dyrelius (1990) and Yin et al.(2016). The algorithms will also be tested independently against the theoretical tensor data published and provided by Dr David Clark (Clark, 2013).

Ideally measured magnetic tensor data should be used, but at the time of the writing of thesis, mining companies such as Anglo and DeBeers had suspended the release of such data to students for studies. After an extensive search, it was realised this would not be possible. Fortunately, it was possible to obtain measured tensor data from Dr David Clark, collected by the GETMAG system over a magnetic skarn deposit at Tallawang, near Gulgong, New South Wales, Australia.

Of critical importance is the methodology. Therefore, each application of tensor data will be examined, coded in Python and evaluated.

The following process will be followed:

1) Examination and coding of voxel based forward modelling of tensor and non-tensor data
2) Examination and coding of voxel-based source-distance inversion of tensor and non-tensor data
3) Examination and coding of source detection techniques
4) Synthesis of the above techniques into a modelling strategy
5) Testing and enhancements to the techniques

Voxel based modelling has been chosen as an effective strategy to simplify model creation and editing. This will also maximise the synergies between forward and inverse modelling. With voxel-based modelling, it is also possible to increase voxel dimensions, with an increase in depth, lowering the model resolution correspondingly. This is a valid consideration since when comparing voxels of equal size, deeper voxels contribute less to the magnetic anomaly and can therefore be made bigger. This opens up more possibilities for speeding up modelling.

Source detection and source distance inversion techniques will be examined as a means to constrain the model further.

The modelling of regular and tensor data will be examined, in particular for source detection techniques. This is in part due to the efficiency of such techniques as a viable alternative to other, more costly, inversion techniques.

Source detection and tensors are a key area for new research. Pedersen and Rasmussen (1990) discussed the gradient tensor with implications for data collection and processing. A considerable amount of work has been done on gravity tensor data, for example Li (2001) developed an algorithm for the 3D inversion of gravity gradiometer data, Beiki (2010) examined analytic signals of the gravity gradient tensor and their application to estimate source location, Cevallos (2016) published the interpretation of the direction of gravity gradient eigenvectors and Christensen et al. (2015) investigated the noise and repeatability of airborne gravity gradiometry. Less has been published for magnetic tensor data. Schmidt and Clark (2006) discussed the properties and uses of the magnetic gradient tensor in source characterisation. Heath, (2003); Heath, Heinson and Greenhalgh (2003) reported on potential field tensor data with respect to an inversion strategy.

The modelling of source distances has been extensively developed for total magnetic intensity (TMI) data by Cooper (2016, 2015, 2014a, 2014b, 2014c); Cooper and Whitehead (2016); Ma and Du (2012).

This work will focus on magnetic tensors. The forward modelling of rectangular prism data will be demonstrated and used for theoretical tests. All source model equations will be rederived into new tensor forms, checked and re-derived into alternate tensor forms where possible. The modelled tensor data is then tested in the new tensor source distance equations.

Conventional source detection techniques often make use of transforms of the potential field, which includes 1^{st} and 2^{nd} order derivatives. The disadvantage of this is the amplification of noise in the data (Cooper, 2015). Another disadvantage is that calculated derivatives (especially in the case of horizontal derivatives) do not represent the instantaneous rate of change of the potential field. Rather, they are an estimate based on adjacent field values, which can be tens to hundreds of meters apart. Measured gradiometer data does not suffer as much from these problems. Any noise is due to the instrument, and not exaggerated from a calculation. The fact that the data is an instantaneous rate of change implies that modelling should be more accurate.

To examine this properly, source-distance equations will be applied to the components of the magnetic field for various forward models (for example dykes and steps). This will also be applied to the gradients of the components similarly, involving the computation of different types of analytic signal.

Remanence relating to tensor data will also be examined. Techniques by authors such as Clark (2014) will be reviewed and equations will be derived from first principles relating to source detection theory.

The tensor-based source detection techniques should have benefits such as better source location accuracy and reduced sensitivity to noise. The increased number of datasets that a tensor provides should allow for less ambiguous modelling.

1.4 Thesis Organisation and Contributions

The organisation and contributions of the thesis are presented here. In general, work presented in this thesis should be regarded as novel unless it has been referenced.

1.4.1 Chapter 1

Chapter 1 gives an overall introduction into the thesis, along with its objectives as well as data availability.

1.4.2 Chapter 2

Chapter 2 introduces forward modelling methodology applied to total magnetic intensity (TMI) data. It advocates an approach to the use of rectangular prisms in forward modelling and details novel strategies which can be used to optimise calculations for speed, making the results and speed comparable to other methods of forward modelling. It also details a novel method of model input which is suited to the use of rectangular prisms in modelling. By allowing the user to draw a model in a manner similar to a paint program, models can be easily input, and more importantly, easily edited.

The chapters discusses pros and cons of rectangular prism modelling and tests the modelling on an interpretation over the Trompsburg Complex.

1.4.3 Chapter 3

Chapter 3 introduces tensor forward modelling. It continues the rectangular prism based modelling, since this is a theme of this thesis. It also gives background into tensor mathematics applied to potential fields, as well as the processing of tensor data. The derivation of tensor modelling equations is demonstrated. The equations are presented with a small novel twist – they allow coordinates to be input in ENU convention (i.e x-axis is east, y-axis is west and z-axis is up) which is compatible with all major GIS and remote sensing software. However, the polarity of results are the same as in END (east north down) and (NED) conventions, making them directly comparable with datasets using modern potential field standards.

1.4.4 Chapter 4

Chapter 4 introduces tensor equations applied to source distance calculations utilising analytic signals. All tensor derivations of the equations are new, as well some new variants of existing equations in the literature. These variants use component definitions

of the analytic signal, which was tested both theoretically and practically to see if the depth relations still held. This has been published in Cole and Cooper (2018). The chapter also examines the effects of noise on the equations, and tests results appropriately.

1.4.5 Chapter 5

Chapter 5 examines remanence in tensor data. It gives an appropriate background on previous work and proceeds to derive novel equations to directly calculate magnetic field direction cosines from the tensor magnetic components. These cosines can easily give an indication of remanence and a starting point for the direction of the remanent field. They also can give an indication of the complexity of remanence within bodies. Equations are also derived to directly calculate remanent inclination, declination and magnetisation, as well as a demonstration of the limitations of such equations.

1.4.6 Chapter 6

Chapter 6 applies the source distance and remanence equations to real data. Two datasets are used – one using tensors derived from TMI data over the Lichtenberg/Zeerust area. These tensors are limited in that they do not account for remanence properly, and proof of this is given through modelled demonstrations. This limitation has not been discussed before in literature, but it emphasizes the advantage of measured tensor data over calculated tensor data, in that only measured data captures the remanent information properly. The results from this interpretation are a first order, 3D interpretation of the dykes over this area using depths and susceptibilities derived from the data.

The second dataset is over the Tallawang magnetite skarn. This tensor data is highly magnetic (with susceptibility of up to 4 SI and remanence of up to 40 A/m). The source distance and remanence techniques are tested on both a simple model of the skarn and the tensor data, and limitations to the processes are demonstrated.

CHAPTER 2 CONVENTIONAL FORWARD MODELLING

2.1 Introduction

Forward modelling is the process of calculating data from model parameters. In this case, a potential field is calculated from parameters such as geometry of source, susceptibility and density. Early work focussed on calculating anomalies from different sources (Nettleton, 1942; Henderson and Zietz, 1948; Reford, 1964; Hjelt, 1972, 1974; Plouff, 1976; Barnett, 1976; Coggon, 1976; Andreasen and Zietz, 1969; Kogbetliantz, 1944; Hall, 1959). Although it was originally calculated along 2D profiles, scientists such as Bhattacharyya (1964) and Talwani (1965) pioneered the calculation of potential fields using bodies comprised of prisms and polygonal facets. More recently, this has been refined by Guptasarma and Singh (1999) and Singh and Guptasarma (2001a; b), who used line integrals to calculate the field component due to each polygonal facet. Holstein (2003) derived formulas from the gravity potential field and field gradient tensor for a polyhedral target body of a spatially linear density medium.

Broadly speaking, there are two basic strategies. One is to calculate a model made up of a matrix of rectangular prisms or rectangular columns (Bhattacharyya, 1964). The other is to form bodies from arbitrary polygonal shapes. Intuitively, it makes sense that the modelling of arbitrary polygons, whether in 2D or 3D, should be more efficient. This is because it may be possible to represent a body with fewer polygons than as a matrix of rectangular prisms or voxels. However, the simple intersection of a 3D geometry with another 3D geometry (say, a sedimentary layer with a dyke) causes the number of polygon facets to increase dramatically and increases the complexity of the modelling process (discussed in section 2.3.4). This thesis will show that voxel or rectangular prism-based modelling can be made efficient, and that such modelling synergises naturally well with inversion techniques.

In addition, the forward modelling equations of a variety of sources – including dykes and steps – form the basis for source detection techniques as well as the testing of those techniques.

2.1.1 Implicit and explicit modelling

Conventional forward modelling where all model parameters and bodies are explicitly defined by the interpreter, is an example of explicit modelling. It is obviously time consuming, and can be prone to user-error in terms of some of the locations of geological bodies as well as model parameters.

Implicit modelling, on the other hand, seeks to assemble a geological model implicitly, taking into account geological information such as boreholes, contact locations and orientations and other information (Lajaunie, Courrioux and Manuel, 1997). Essentially, contact locations and orientation data are used to construct an implicit scalar potential field through interpolation. This field can be updated with the addition of new data. Geological boundaries are then described through isopotential surfaces and the dips of such boundaries are represented by the gradients of the potential (McInerney et al., 2007; Calcagno et al., 2008). Numerous authors have published on this work and made refinements to it (McInerney et al., 2007; Calcagno et al., 2008; Fitzgerald and Milligan, 2013; Husson et al., 2018).

Implicit modelling is arguably an effective way of managing a model and is a good way of defining an initial or ultimate model. It synergizes well with techniques such as forward modelling and inversion (Guillen et al., 2008).

Although this thesis deals with explicit voxel based forward modelling, the nature of both philosophies is that they will work well together and should not be thought of as incompatible.

2.1.2 Axis conventions

Axis conventions are standardized ways of establishing the location and orientation of coordinate axes. They are important as they make it possible to relate different datasets which may come from different forms of axes. These conventions often specify the directions of positive axes. For example, ENU denotes positive axes in the East, North and Up directions. Coordinate systems are said to be left handed or right handed. If the thumb is aligned with the z-axis, and the forefinger coincides with the x-axis, then the direction of the y-axis is obtained by the second finger of the right hand for right handed systems, or the second finger of the left hand for left handed systems. Alternatively, in right handed systems the cross product of any two basis vectors in a left to right cyclic sequence yields the third. They therefore satisfy $\boldsymbol{X} \times \boldsymbol{Y} = \boldsymbol{Z}, \boldsymbol{Y} \times \boldsymbol{Z} = \boldsymbol{X}, \boldsymbol{Z} \times \boldsymbol{X} = \boldsymbol{Y}$ where $\boldsymbol{X}, \boldsymbol{Y}, \boldsymbol{Z}$ are the axis vectors. For left handed systems we have $\boldsymbol{X} \times \boldsymbol{Y} = -\boldsymbol{Z}, \boldsymbol{Y} \times \boldsymbol{Z} = -\boldsymbol{X}, \boldsymbol{Z} \times \boldsymbol{X} = -\boldsymbol{Y}$.

Below are a list of commonly used conventions:

- ENU (East, North, Up) – used in terrain and on land vehicles, right handed
- NED (North, East, Down) – used on aircraft, magnetics, right handed
- END (East, North, Down), left handed
- ESD (East, South, Down) right handed
- NEU (North, East, Up) left handed

The two most common conventions in use for potential field data are NED (North, East, Down) (Henderson and Zietz, 1948; Tarlowski, 1989; Plouff, 1976; Li and Chouteau, 1998; Talwani, 1965; Singh and Guptasarma, 2001b) and ESD (East, South, Down) (Hall, 1959; Hjelt, 1972; Rasmussen and Pedersen, 1979; Kogbetliantz, 1944; Bhattacharyya, 1964; Coggon, 1976).

Nevertheless, other conventions are also in use. For example, NEU (North, East, Up) (Baykiev et al., 2016) and ENU (East, North, Up) (Zhu, 2007).

This thesis advocates both NED and ENU conventions for magnetic data. They are both right handed systems, and while not as commonly in use for potential field data, when modelling, ENU allows for a more intuitive integration between a model and terrain or map data (which follows ENU conventions). Where necessary, the convention associated with an equation will be stated.

2.2 Magnetic Field

The magnetic field has been described using various terminologies (H field, B field, and magnetic potential to name a few) and often these terms are used interchangeably. Therefore, it is necessary to revisit the basic concepts to establish context in this study. This section is intended to be a literature study to establish background, conventions to be used in this thesis, and has been covered in many different sources in literature. For convenience, one may refer to Blakely (1995, chaps 4–5).

2.2.1 Units

This study will follow the International System of Units (abbreviated as SI) (Blakely, 1995, p 67). However, its alternative, the cgs (centimetre-gram-second) system may be referred to occasionally in the text. The cgs system is also called the Gaussian unit system or Gaussian-cgs units. Formulas exist for both cgs and SI units (Sheriff, 1991, p 183). Conversions between these units are well documented in literature (Goldfarb and Fickett, 1985).

2.2.2 The $\boldsymbol{B}$ Field

The Lorentz force law describes the combination of electric and magnetic force on a point charge due to electromagnetic fields. It is defined as follows:

$$\boldsymbol{F} = q(\boldsymbol{E} + \boldsymbol{v} \cdot \boldsymbol{B}) \tag{2.1}$$

where q is the charge, $\boldsymbol{E}$ is the electric field, $\boldsymbol{v}$ is the velocity of the charged particle and $\boldsymbol{B}$ is termed the magnetic field. The units of $\boldsymbol{B}$ are teslas (T) or gauss (G), depending on whether the units are according to SI or cgs conventions. In geophysics, nanotesla (nT) or gamma (γ) are used and are numerically equivalent. The force due to a magnetic field necessary to describe the motion of a charged particle is therefore (Blakely, 1995, p.66):

$$\boldsymbol{F} = q(\boldsymbol{v} \cdot \boldsymbol{B}) \tag{2.2}$$

The $\boldsymbol{B}$ field is the magnetic field measured by magnetometers on surveys.

2.2.3 The H Field, M and k

The measured $\boldsymbol{B}$ field is comprised of two main sources (Blakely, 1995, sec.5.3); that which comes from external field ($\boldsymbol{H}$), and that which comes from nearby materials ($\boldsymbol{M}$). It can be stated as follows (Sheriff, 1991, p 185):

$$\boldsymbol{B} = \mu_0(\boldsymbol{H} + \boldsymbol{M}) = \boldsymbol{B_a} + \boldsymbol{B_m} \tag{2.3}$$

where μ_0 is the permeability in a vacuum (units: newtons per ampere squared or N/A^2) and $\boldsymbol{M}$ is referred to as the magnetisation or magnetic polarization. $\boldsymbol{B_a}$ is the ambient magnetic field and $\boldsymbol{B_m}$ is the field due to the magnetisation. In SI units, both $\boldsymbol{H}$ and $\boldsymbol{M}$ have units of ampere per meter (A/m).

If the field $\boldsymbol{H}$ is small, the response of the magnetisation $\boldsymbol{M}$ in the material is approximately linear:

$$\boldsymbol{M} = k\boldsymbol{H} \tag{2.4}$$

where k is a dimensionless constant of proportionality called the magnetic susceptibility. The magnetisation value shows how strongly a region of material is magnetised and is therefore the magnetic dipole moment per unit volume. The susceptibility can be rewritten as:

$$k = \frac{|\boldsymbol{M}|}{|\boldsymbol{H}|} \tag{2.5}$$

Substituting (2.4) into (2.3) gives:

$$\begin{aligned} \boldsymbol{B} &= \mu_0(\boldsymbol{H} + \boldsymbol{M}) \\ &= \mu_0(1 + k)\boldsymbol{H} \\ &= \mu_0\mu_r\boldsymbol{H} \\ &= \mu\boldsymbol{H} \end{aligned} \tag{2.6}$$

where μ_r is the relative permeability of the material and $\mu_0$is the permeability of free space. This then becomes:

$$\boldsymbol{H} = \frac{\boldsymbol{B}}{\mu} \tag{2.7}$$

Alternatively, from (2.3) again:

$$\boldsymbol{H} = \frac{\boldsymbol{B}}{\mu_0} - \boldsymbol{M} \tag{2.8}$$

The units for the magnetic field strength $\boldsymbol{H}$ can be derived from its relationship to the magnetic field $\boldsymbol{B}$ using equation (2.7). Since the unit of magnetic permeability (μ) is N/A^2, then the unit for the magnetic field strength is $T/(N/A^2) = (N/A \cdot m)/(N/A^2) = A/m$, where T is Tesla, N is Newton, A is Ampere and m is metre.

From all of this it can be seen that the units of $\boldsymbol{H}$ and $\boldsymbol{M}$ are the same, namely Ampere per metre (A/m).

2.2.4 Remanent Magnetisation

This section will deal with basic remanent magnetisation theory. An expanded discussion on remanence calculations in the case of tensors will be covered in CHAPTER 5

Magnetisation can be induced into a material (which by itself would otherwise have no field) or it can originate from the material itself. Remanence is the magnetisation left behind in a ferromagnetic material after an external magnetic field is removed. There are various mechanisms by which this can happen (Butler, 2004). Thermoremanent magnetisation is acquired through the cooling of igneous rocks below the Curie temperature. Chemical remanent magnetisation occurs through magnetic grains precipitating from a circulating solution of chemical reaction. Depositional remanent magnetisation can occur through magnetic grains aligning with the magnetic field soon after deposition. Viscous remanent magnetisation is acquired by ferromagnetic minerals by sitting in a magnetic field for some time.

Remanence is an important concept when modelling magnetic data. It is taken into account in magnetic field calculations as the component of the anomaly field that is not in the direction of the current magnetic field. It forms part of the magnetisation or magnetic polarization parameter.

If $\boldsymbol{M}$ is the total magnetisation of a body, then (Blakely, 1995, p.89; Clark, 1997)

$$\begin{aligned} \boldsymbol{M} &= \boldsymbol{M_i} + \boldsymbol{M_r} \\ &= k\boldsymbol{H} + \boldsymbol{M_r} \end{aligned} \tag{2.9}$$

where $\boldsymbol{M_i}$ and $\boldsymbol{M_r}$ indicate induced and remanent magnetisation respectively. The relative magnitude of remanence is defined as the ratio of $\boldsymbol{M_r}$ to $\boldsymbol{M_i}$ and has been named the Koenigsberger ratio or the Q-ratio (Blakely, 1995, p.90; Clark, 1997):

$$Q = \frac{|\boldsymbol{M_r}|}{|\boldsymbol{M_i}|} \tag{2.10}$$

A large Q-ratio is an indicator that magnetic material will tend to retain significant remanent magnetisation. Strong remanent magnetisation suggests the presence of finely crystalline magnetic mineral grains (Grant, 1985). Clark, (1997) provides a good overview of the relationship between mineral grain domain structure and magnetisation. He noted that small mineral grains (0.05 - 1μm) have a single domain (SD) structure, which means that they are uniformly magnetised. Small mineral grains can retain remanent magnetisation for a long time and are important carriers of remanent magnetisation in many types of rocks. Larger grains are more likely to subdivide into a number of magnetic domains (known as multi domain or MD grains) with different magnetic orientations, thereby reducing the remanent magnetisation strength. Therefore, although these grains can have high magnetic susceptibilities, they do not retain remanent magnetisation as well as single domain grains. Pseudo single domain (PSD) grains have properties intermediate between SD and MD grains. They are generally in the order of a few microns in size (~1 – 20 μm). Clark, (1997) summarised that with increasing grain size, the remanence generally decreases whereas susceptibility increases.

The defining characteristics of magnetic fields and magnetisation are the magnitude and the direction of the fields. The direction of the fields are defined in terms of inclinations and declinations, translated into direction cosines (Talwani, 1965). Direction cosines are a method of translating angular information (inclination and declination) into conventional coordinate space.

If $\boldsymbol{M_r}$ has inclination and declination M_{inc} and M_{dec}, then:

$$\alpha_r = \cos(M_{inc}) \cdot \cos(M_{dec}) \tag{2.11}$$

$$\beta_r = \cos(M_{inc}) \cdot \sin(M_{dec}) \tag{2.12}$$

$$\gamma_r = \sin(M_{inc}) \tag{2.13}$$

Where $\alpha_r, \beta_r, \gamma_r$ are the direction cosines of remanent magnetisation $\boldsymbol{M_r}$.Note that these expressions are valid for NED coordinate systems. If the x-axis does not point north, then the deviation from north must be subtracted from the declination first. This is of critical importance if using the ENU coordinate system where the deviation from north is 90 degrees.

Similarly, if $\boldsymbol{M_i}$ (or $\boldsymbol{H}$ since it will be in this direction) has inclination and declination H_{inc} and H_{dec}, then it has the following direction cosines:

$$\alpha_i = \cos(H_{inc}) \cdot \cos(H_{dec}) \tag{2.14}$$

$$\beta_i = \cos(H_{inc}) \cdot \sin(H_{dec}) \tag{2.15}$$

$$\gamma_i = \sin(H_{inc}) \tag{2.16}$$

From this the expression for $\boldsymbol{M_r}$ becomes:

$$\boldsymbol{M_r} = |\boldsymbol{M_r}| \cdot [\alpha_r, \beta_r, \gamma_r] \tag{2.17}$$

$|\boldsymbol{M_r}|$ is the magnitude of $\boldsymbol{M_r}$. For the case of $\boldsymbol{M_i}$ we can make use of equation (2.4), which relates $\boldsymbol{M_i}$ to $\boldsymbol{H}$ and take advantage of the fact that since we know the ambient magnetic field when modelling, equation (2.3) can be used with $\boldsymbol{M} = 0$. We therefore get

$$\begin{aligned} \boldsymbol{M_i} &= |\boldsymbol{M_i}| \cdot [\alpha_i, \beta_i, \gamma_i] \\ &= k \cdot |\boldsymbol{H}| \cdot [\alpha_i, \beta_i, \gamma_i] \\ &= \frac{k \cdot |\boldsymbol{B_a}| \cdot 10^{-9}}{\mu_0} \cdot [\alpha_i, \beta_i, \gamma_i] \end{aligned} \tag{2.18}$$

$|\boldsymbol{B_a}|$ is the ambient magnetic field in nanotesla, necessitating the need to multiply the equation by 10^{-9} to convert it to tesla.

By substituting (2.17) and (2.18) into (2.9), it follows that:

$$\begin{aligned}\boldsymbol{M} &= \boldsymbol{M_i} + \boldsymbol{M_r} \\ &= \frac{k \cdot |\boldsymbol{B_a}| \cdot 10^{-9}}{\mu_0} \cdot [\alpha_i, \beta_i, \gamma_i] + |\boldsymbol{M_r}| \cdot [\alpha_r, \beta_r, \gamma_r] \\ &= \frac{k \cdot |\boldsymbol{B_a}|}{400\pi} \cdot [\alpha_i, \beta_i, \gamma_i] + |\boldsymbol{M_r}| \cdot [\alpha_r, \beta_r, \gamma_r] \\ &= |\boldsymbol{M}| \cdot [\alpha_t, \beta_t, \gamma_t]\end{aligned} \tag{2.19}$$

Where $\alpha_t, \beta_t, \gamma_t$ are the new resultant direction cosines.To express $\boldsymbol{M}$ in field strength ($\boldsymbol{B}$ with units in tesla), (2.19) is multiplied by $\mu_0/4\pi$. For nanotesla this becomes $\mu_0 \times 10^{-9}/4\pi = 100$. Therefore:

$$\begin{aligned}\boldsymbol{B} &= 100 \cdot \left(\frac{k \cdot |\boldsymbol{B_a}|}{400\pi} \cdot [\alpha_i, \beta_i, \gamma_i] + |\boldsymbol{M_r}| \cdot [\alpha_r, \beta_r, \gamma_r] \right) \\ &= \frac{k \cdot |\boldsymbol{B_a}|}{4\pi} \cdot [\alpha_i, \beta_i, \gamma_i] + 100 \cdot |\boldsymbol{M_r}| \cdot [\alpha_r, \beta_r, \gamma_r] \\ &= |\boldsymbol{B}| \cdot [\alpha_t, \beta_t, \gamma_t]\end{aligned} \tag{2.20}$$

Importantly, for use in forward modelling, the resultant direction cosines for $\boldsymbol{M}$ can be obtained by dividing $\boldsymbol{M}$ by $|\boldsymbol{M}|$ which can be obtained through the following equation:

$$|\boldsymbol{M}| = \sqrt{\boldsymbol{M} \cdot \boldsymbol{M}} \tag{2.21}$$

Where $\boldsymbol{M}$ is defined by equation (2.19).

2.2.5 Magnetic Potential

The magnetic potential (Blakely, 1995, sec.4.3) is the work done by a magnetic particle against the field. It is an important concept in derivations for forward modelling. The term magnetic potential can be used for either of two quantities: the magnetic vector potential ($\boldsymbol{A}$) and the magnetic scalar potential (φ).

Both quantities can be useful in calculating the magnetic field. The vector potential is related to the $\boldsymbol{B}$ field via:

$$\boldsymbol{B} = \nabla \times \boldsymbol{A} \tag{2.22}$$

where the operator "$\nabla \times$" denotes taking the curl of the magnetic vector potential ($\boldsymbol{A}$)

The magnetic scalar potential φ is related to the $\boldsymbol{H}$ field (in cases when there are no free currents) by:

$$\boldsymbol{H} = -\nabla\varphi = \frac{\boldsymbol{B}}{\mu_0} = \frac{\nabla \times \boldsymbol{A}}{\mu_0} \tag{2.23}$$

2.2.6 Total Magnetic Intensity

If B_x, B_y and B_z are defined to be three components of a magnetic field $\boldsymbol{B}$, and α, β, γ are defined to be the direction cosines relating to the direction of the magnetic field, then the total magnetic intensity B_{tmi} (often referred to as f) can be approximately defined as:

$$B_{tmi} = \alpha \cdot B_x + \beta \cdot B_y + \gamma \cdot B_z \tag{2.24}$$

In spite of being an approximation, this expression has useful mathematical properties. It obeys Laplace's equation, and is therefore a true potential field. It can therefore be continued between surfaces at differing levels, if accurately known everywhere over one surface.

When anomalies are too strong, then this relationship is no longer accurate and the full measured total field anomaly should be used. Schmidt and Clark (2006) point out that the difference between the two expressions can be as large as 1,000 nT for a 10,000 nT anomaly in a 50,000 nT regional field. The full total magnetic intensity is actually defined as:

$$B_{tmi} = \sqrt{(B_x + \alpha B_a)^2 + \left(B_y + \beta B_a\right)^2 + (B_z + \gamma B_a)^2} - B_a \tag{2.25}$$

where $B_a = |\boldsymbol{B_a}|$ is the ambient magnetic field. This version of B_{tmi} is not a potential field, since it does not obey Laplace's equation exactly (Schmidt and Clark, 2006).

2.3 Conventional Forward Modelling in 3D

A voxel-based approach, rather than a polygonal approach, is advocated for potential field modelling. This section aims to show that the creation and editing of voxel-based models can be faster than the equivalent facet-based models. While the actual calculation speeds of voxel-based models may not necessarily be faster, the section also aims to demonstrate that they need not be inefficient either, simply by adopting optimised calculation strategies.

2.3.1 Forward Modelling Theory

Firstly, the magnetic field due to a rectangular prism needs to be calculated. There are many techniques which can be considered here. Bhattacharyya (1964) developed the equations for a rectangular prism of infinite depth. Guptasarma and Singh (1999) and Singh and Guptasarma (2001a; b) used line integrals to calculate the field component due to each polygonal facet. Holstein (2003) derived formulas from the gravity potential field and field gradient tensor for a polyhedral target body of a spatially linear density medium. Parker (1973) showed that Fourier transforms can be used to calculate magnetic or gravitational anomalies. More recently, Caratori Tontini, Cocchi and Carmisciano, (2009) also applied Fourier transforms for rapid 3-D forward modelling of potential fields.

In this case the technique by Bhattacharyya (1964) was used. The simplicity of the technique makes it well suited to rectangular prism calculations. It is described and developed into a Fortran routine, named 'mbox' by Blakely (1995, pp. 200-201) based on the work by Bhattacharyya (1964). The dimensions of the prism are given by $x_1 \leq x \leq x_2, y_1 \leq y \leq y_2, z_1 \leq z < \infty$. The prism is observed in a regional field defined by $\boldsymbol{H} = (H_x, H_y, H_z)$ and the magnetisation of the rectangular prism is given as $\boldsymbol{M} = (M_x, M_y, M_z)$. The equations are (in NED convention)

$$\begin{aligned} B = C_m M \Bigg[& \frac{\alpha_{23}}{2} \log\left(\frac{r - x'}{r + x'}\right) + \frac{\alpha_{13}}{2} \log\left(\frac{r - y'}{r + y'}\right) \\ & - \alpha_{12} \log(r + z_1) \\ & - M_x H_x \tan^{-1}\left(\frac{x'y'}{x'^2 + rz_1 + z_1^2}\right) \\ & - M_y H_y \tan^{-1}\left(\frac{x'y'}{r'^2 + rz_1 + x'^2}\right) \\ & + M_z H_z \tan^{-1}\left(\frac{x'y'}{rz_1}\right) \Bigg] \Bigg|_{x' = x_1}^{x' = x_2} \Bigg|_{y' = y_1}^{y' = y_2} \end{aligned} \tag{2.26}$$

where:

$$\alpha_{12} = M_x H_y + M_y H_x \tag{2.27}$$

$$\alpha_{13} = M_x H_z + M_z H_x \tag{2.28}$$

$$\alpha_{23} = M_y H_z + M_z H_y \tag{2.29}$$

$$r^2 = x'^2 + y'^2 + z'^2 \tag{2.30}$$

$$M = \sqrt{M_x^2 + M_y^2 + M_z^2} \tag{2.31}$$

C_m is a constant that depends on the units of measurement. In cgs units, $C_m = 1$ and in SI units, $C_m = \mu_0/4\pi$. Note that M is the total magnetisation of the rectangular prism, as described by (2.9) and the equations following. It is not the remanent magnetisation exclusively.

When equation (2.26) is calculated twice, at different depths, the difference between the solutions gives the solution for a rectangular prism bounded by the two depths.

Other techniques for calculating the magnetic field due to a magnetic body have also been published. For example, Guptasarma and Singh (1999) published a technique to compute the magnetic field resulting from a uniformly magnetised arbitrary polyhedron. It is very well suited to polyhedrons and calculates the x, y and z components of the field as well as the total field. However, to do this it calculates line integrals for each face of the polyhedron. This makes it a more complex calculation than the 'mbox' routine described above. As such it is not as efficient for voxel based calculations.

To demonstrate the versatility and applicability of voxel based calculations, the gravity case is also presented. For the gravity field calculation Blakely (1995, pp. 186-187) developed a FORTRAN routine, based on the work of Plouff (1976), named 'gbox'. In this case, the dimensions of the prism are given by $x_1 \leq x \leq x_2, y_1 \leq y \leq y_2, z_1 \leq z < z_2$. The equations for gravity are (in NED convention):

$$g = \gamma\rho \sum_{i=1}^{2}\sum_{j=1}^{2}\sum_{k=1}^{2} \mu_{ijk}\left[z_k \tan^{-1}\frac{x_i y_i}{z_k R_{ijk}} - x_i \log(R_{ijk} + y_j) - y_i \log(R_{ijk} + x_j)\right] \tag{2.32}$$

where ρ is uniform density is, γ is the gravitation constant, $R_{ijk} = \sqrt{x_i^2 + y_j^2 + z_k^2}$ and $\mu_{ijk} = (-1)^i(-1)^j(-1)^k$.

Unlike the case with calculating magnetic fields, this equation only has to be calculated once per rectangular prism.

2.3.2 Computational strategy for combined rectangular prism-based calculations

A strategy is proposed for the computation of voxel based calculations. Voxels can be modelled as rectangular prisms. The fundamental shape of the magnetic anomaly due to a rectangular prism is dependent on three factors, namely the vertical distance from the observation point, the direction of magnetisation (both remanent and induced) and the magnetic susceptibility of the rectangular prism. If the terrain is flat, the horizontal location of a rectangular prism has no impact on the fundamental shape of the anomaly over a few hundred km^2, since the inducing field does not change significantly over this size area (Pedersen, Rasmussen and Dyrelius, 1990). In addition, susceptibility only affects the vertical scale of the anomaly shape. Therefore, the same calculated anomaly can be used for two different lithologies that only differ by susceptibility.

With this in mind, a lithology model comprised of voxels I_{ijk} is defined, where i, j, k represent rows, columns and layers in the model. For the purposes of programming, the lithology model is made up of lithology indices, which are defined as follows:

$$I_{ijk} = \begin{cases} -1 \text{ for air} \\ 0 \text{ for background lithology} \\ \geq 1 \; and \leq n \text{ for modelled lithology} \end{cases} \quad (2.33)$$

This set of lithology indices represents n different magnetic lithologies in the model. The value of -1 is used as a marker to account for air above terrain, and 0 is background lithology. In this way, all aspects are accounted for, including terrain. For example, equation (2.34) shows a slice of the lithology model with a small hill, made up of voxels I_{ijk} where $i = 1$, and j and k are all columns and layers respectively. For convenience, the number of calculations at observation points is equal to the number of rectangular prisms, with each calculation located horizontally in the centre of its respective rectangular prism.

$$\boldsymbol{I_{i=1}} = \begin{bmatrix} -1 & -1 & -1 & -1 & -1 \\ -1 & -1 & 0 & -1 & -1 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 1 & 1 & 0 \\ 0 & 1 & 1 & 1 & 0 \end{bmatrix} \quad (2.34)$$

$\boldsymbol{I_{i=1}}$ are all the values of I_{ijk} where $i = 1$. For each value of $I_{ijk} > 0$, the procedure is as follows:

1) For a single rectangular prism, calculate the anomaly solutions, covering *twice* the area of interest (to facilitate anomaly shifting later). The anomaly solution for layer k and lithology n is defined as $\boldsymbol{R_{nk}}$ (Figure 1)**.**
2) Repeat point (1) for each layer or depth of the model.

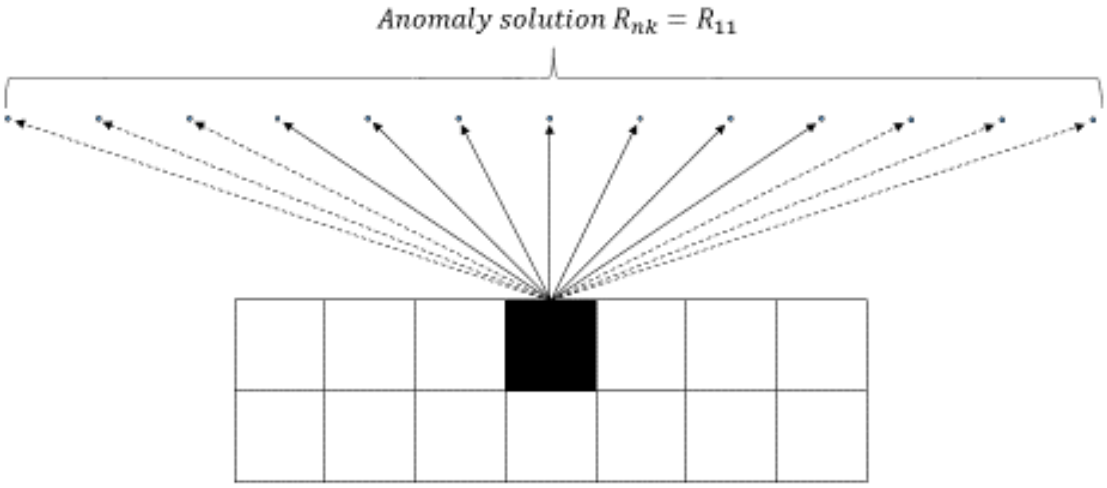

Figure 1 Diagram showing anomaly solution for layer 1 and lithology 1 calculated for a rectangular prism. Solid arrows show which calculations are summed into the total field for this rectangular prism. Dashed arrows indicate solutions outside of the modelled area, but which will be used when summing anomalies.

These points are repeated for each of the *n* lithologies in the model. Then:

3) Iterate through each row, column and layer of the model I_{ijk}. For each rectangular prism with lithology *n* at layer *k* of I_{ijk}, where $n > 0$, shift $\boldsymbol{R_{nk}}$ to be centred at point ij and sum the solutions over the model to the total calculated field (Figure 2).

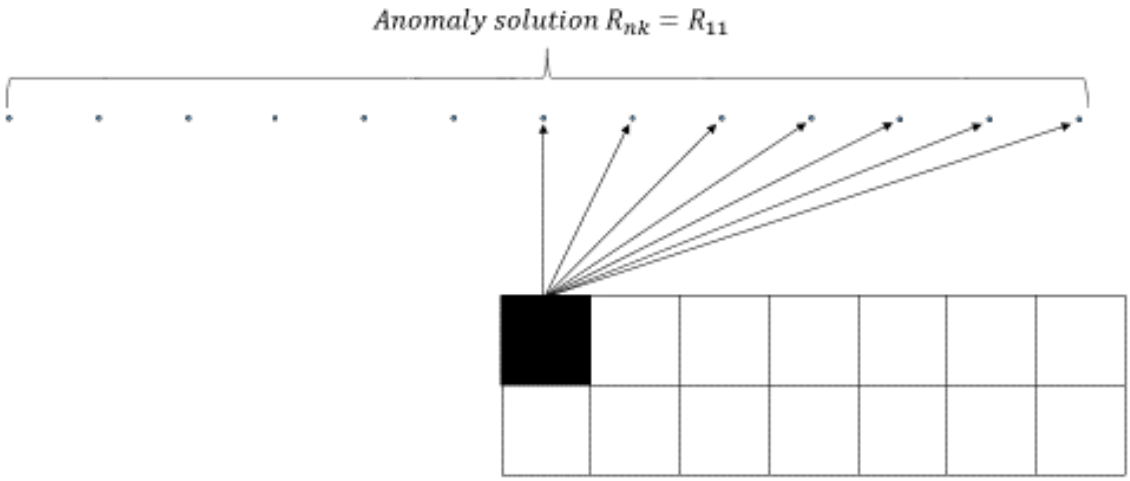

Figure 2 Diagram showing anomaly solution for layer 1 and lithology 1 calculated in Figure 1, then reused for a different rectangular prism.

The above strategy holds for perfectly flat terrain. In order to correct for topography, rectangular prisms that may lie above the observation point have to be accounted for. An example of this is an observation point in a valley that must still take into account adjoining peaks that lie above it (Figure 3).

To account for this, point (2) above must also be done for a range of pseudo layers above the model. This range is equal to the (maximum height – minimum height)/(rectangular prism height). Then at point (3), R_{nk} is replaced with $R_{n(k-d)}$ where *d* is an element of a surface representing the number of rows below the maximum height.

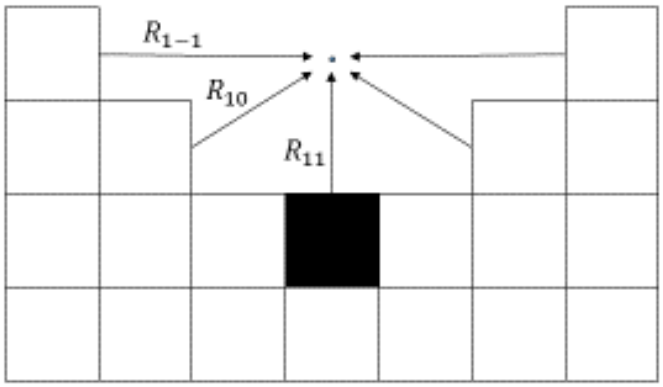

Figure 3 Diagram showing the anomaly solutions which must be used when the topography is not flat. Notice that anomaly solutions are now calculated in the reverse direction (on and under the ground), i.e. R_{10}, R_{1-1}

All of this is mathematically expressed as follows: assume the calculated magnetic field B is for a grid of observation points of r rows and c columns. The model that it will be calculated from also has r rows and c columns and has l layers. The altitude correction D has r rows and c columns and contains the integer layer offsets from the maximum height. Equation (2.35) describes the final summation operation for each final grid point B_{pq}.

$$B_{pq} = \sum_{p=1}^{r} \sum_{q=1}^{c} \sum_{i=p}^{(c+p)} \sum_{j=q}^{(r+q)} \sum_{k=D_{pq}}^{(l+D_{pq})} B_{pqijk} \tag{2.35}$$

where B_{pqijk} is the field contribution from magnetic lithology I_{ijk} at location p, q, taken from $\boldsymbol{R_{nk}}$ (shifted to be centred on position i, j).

2.3.3 Forward modelling software test

The theory outlined in sections 2.3.1 and 2.3.2 were tested against GM-SYS, a major commercial package distributed by Geosoft. A dyke was modelled in a regional field of 30,000 nT, with an inclination of -63 degrees, and a declination of -17 degrees. The susceptibility of the dyke was 0.01 SI. The density was 2.8 g/cm^3. To test remanence, a remanent magnetisation of 0.199 A/m was assigned to the dyke, with an associated inclination of 35 degrees, and declination of 80 degrees.

Figure 4(a) shows the results of the comparison. As can be seen, the voxel based model matches the GM-SYS modelled anomaly perfectly, illustrating that all aspects of the calculation are working perfectly. Note that the voxel rectangular prisms (cyan colour) are also shown in Figure 4(b). Less rectangular prisms could have been used but this was necessary to test the computational strategy outlined in section 2.3.2. Although there are 90 rectangular prisms, only 9 anomalies corresponding to the 9 layers present needed to be calculated.

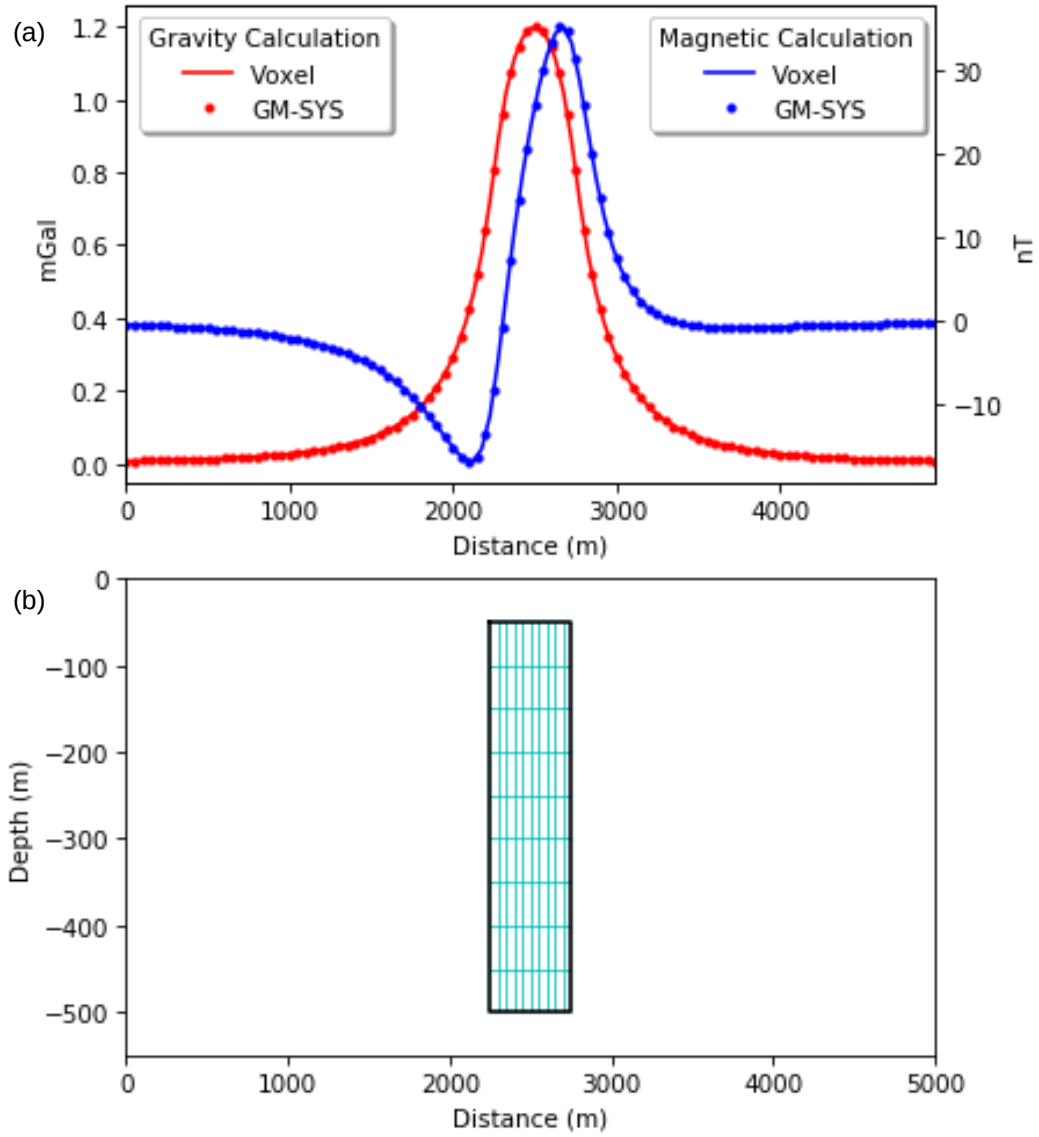

Figure 4 Modelled results of a dyke, showing a gravity and magnetic comparison of results between calculated voxel responses and GM-SYS. Regional geomagnetic field is 30,000 nT, with inclination of -63 degrees and declination of -17 degrees. The susceptibility 0.01 SI. Density is 2.8 g/cm^3. Remanent magnetisation is 0.199 A/m with inclination of 35 degrees and declination of 80 degrees. The anomaly responses are shown in (a) and the body modelled is shown in (b).

2.3.4 Model creation and editing

As mentioned in the introduction, there are two basic strategies for forward modelling in 3D. The conventional approach is to form bodies from arbitrary polygonal shapes.

The time it takes to construct a 3D model can be prohibitive. Conventional polygonal based 3D modelling programs (3DS Max, Maya, Blender) are akin to CAD packages and require a steep learning curve, which is not always possible.

An alternative approach is to use a matrix of rectangular prisms or voxels to form the body. Figure 5 shows a body represented by rectangular prisms and by polygons. Since there are far more rectangular prisms than polygons, calculating a magnetic field from rectangular prisms has fallen out of favour, if only to save calculation time.

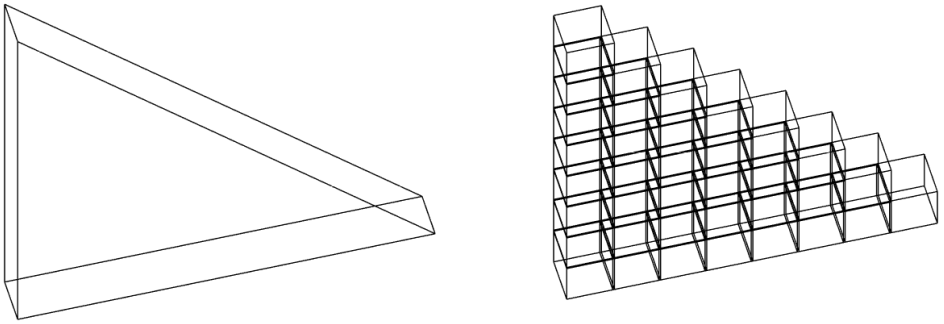

Figure 5 Comparison between polygonal shape (left) and rectangular prism based shape (right)

The major time-sink is in the editing of the 3D models. This problem is definitely present when editing a polygonal style body. Figure 6 shows the increase in complexity from just a few edits. This increase in complexity slows calculation times and may necessitate polygonal simplification algorithms.

Model editing is therefore not trivial and can be extremely time consuming. This is especially the case when directly editing a full 3D mesh of facets, because of the increased geometric complexity of facets versus voxels. An example of this will be demonstrated in section 2.3.9, where model creation and editing were reduced by an order of magnitude (months to days).

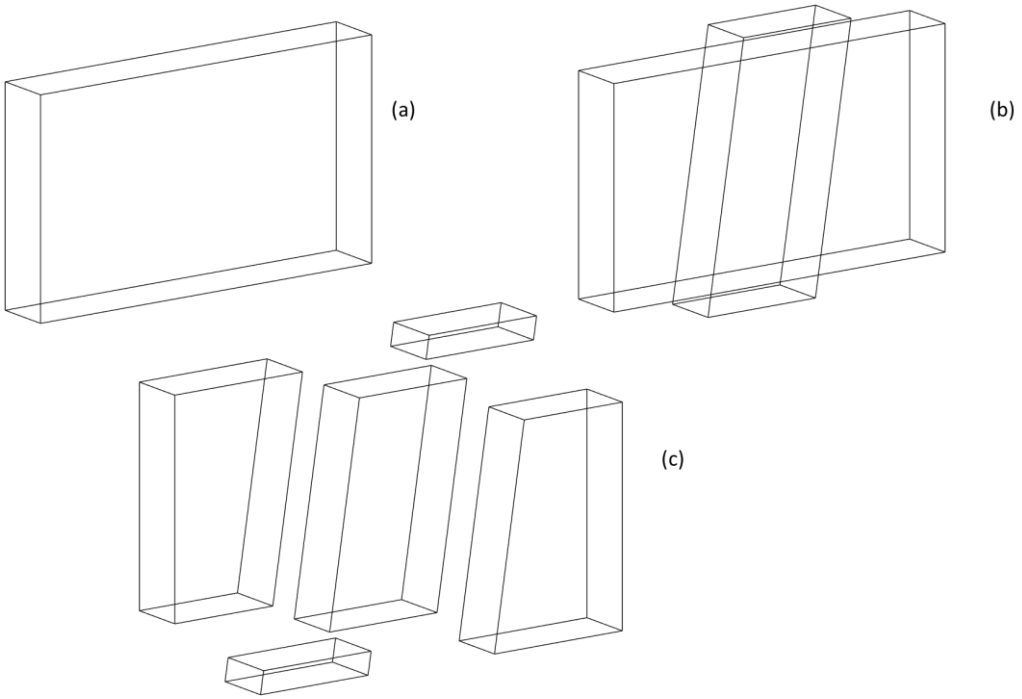

Figure 6 Demonstration of increasing complexity in polygonal modelling. In (a) a simple starting polygon is shown. In (b) the starting polygon is intersected by another shape (say, a dyke). In (c) the increase in polygonal facets if only one edit has been made to the model is shown.

The ideal would be to simply draw a sketch of the model and make changes in a similar manner. The learning curve to do this is low and edits to the model are not as time consuming.

This is possible with rectangular prism (or voxel) based modelling. If a model is defined as having a set number of rows, columns and layers, then it is possible to slice that model along one of its rows, columns or layers. This slice gives a 2D representation of that point in the model. The slice can also be manually drawn on, much like a computer graphics application. The pixels making up the “ink” of the line being drawn would each be a rectangular prism or voxel (Figure 7).

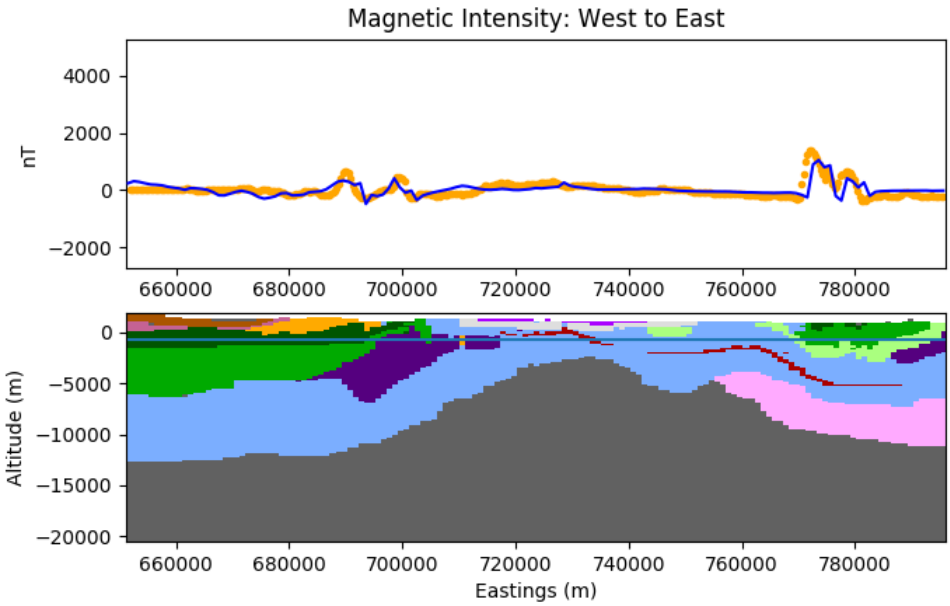

Figure 7 Modelled profile with calculated and observed magnetic data. Each pixel in the model represents a voxel with corresponding magnetic lithology.

Although edits are 2D, they are slices of a 3D model and can be made on both vertical slices (much like conventional 2D software) and horizontal layers (Figure 8), thereby ensuring model continuity.

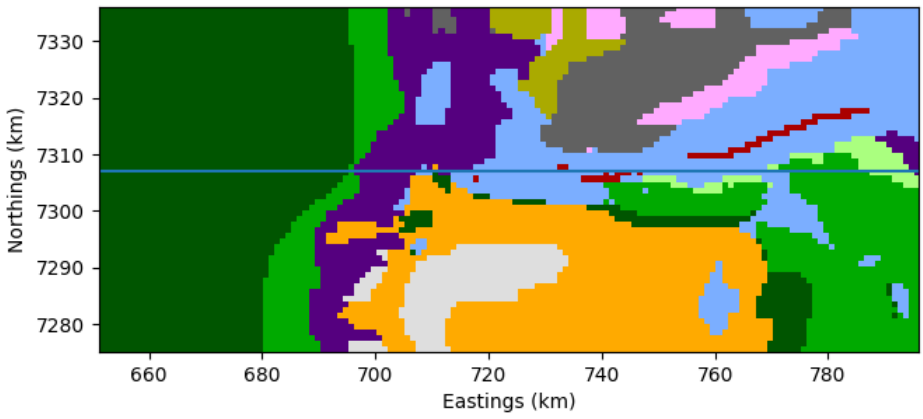

Figure 8 Horizontal slice of a 3D model.

Edits on both vertical and horizontal slices are trivial. Edges of bodies can be quickly redrawn with little to no learning curve required. The copying and pasting of details from one profile to multiple other profiles is also simple to program and use.

In polygonal based modelling packages, each body would get a unique geophysical definition. In a voxel-based design, this is simply not practical. Therefore, voxel-based modelling in this case is defined to be magnetic lithology centric rather than body centric. Each magnetic lithology gets a unique geophysical definition that is applied to all rectangular prisms of that magnetic lithology. The advantage of this is that a magnetic lithology only has to be changed once for all related bodies in the model.

With this in mind, a magnetic lithology is defined to mean all bodies (rectangular prisms) with the same magnetic parameters – i.e. susceptibility, field direction, remanence.

2.3.5 Model accuracy

On the face of it, it makes sense that facets can model certain structures more precisely than voxels. For example, a dipping dyke can be simply modelled with fewer points using facets than with its voxel alternative, which will require potentially many more voxels to be calculated to reach the same level of detail (Figure 9).

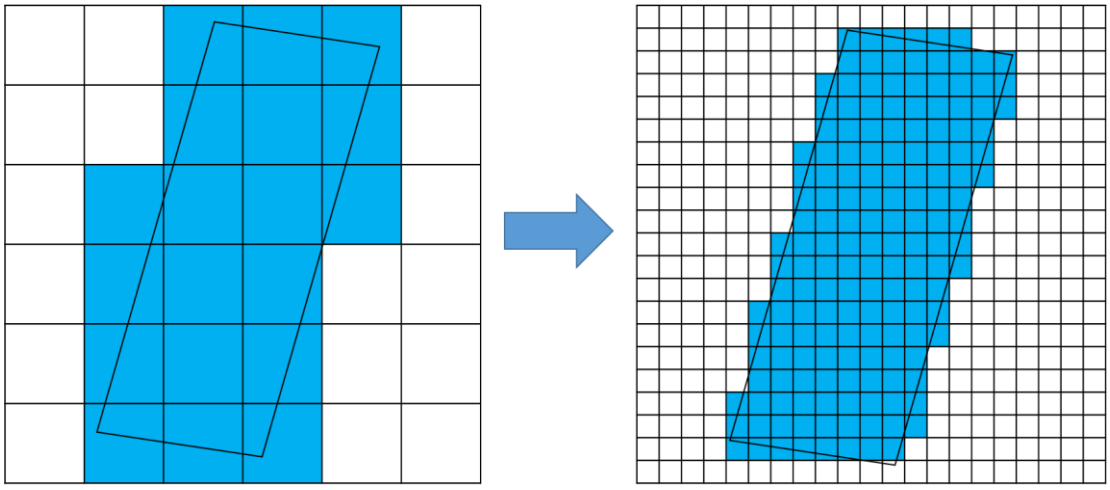

Figure 9 Illustration showing the increase in voxel resolution necessary to approximate dipping dykes.

There are a few points to be noticed. The approximation used in Figure 9 uses 160 rectangular prisms for its calculation as opposed to 1 facet based calculation. However, taking into account redundancies in the rectangular prism calculation, as pointed out earlier, 160 can be reduced to 19 since we only need to calculate anomalies once per layer. If the dyke is more complex, the facet based calculation will slow down (due to

increased facets) while the rectangular prism based calculation will remain similar to before.

Figure 10 shows a comparison between voxel anomalies and polygonal anomalies. It serves to highlight one of the pitfalls of voxel based modelling. Since each voxel has a finite lateral resolution, if the edge of a body does not accurately coincide with the edge of a voxel, an anomaly shift can occur. This is easily fixed, by either shifting the voxel registration point (Figure 11) or by increasing the resolution of the voxet (data set of voxels) (Figure 12)

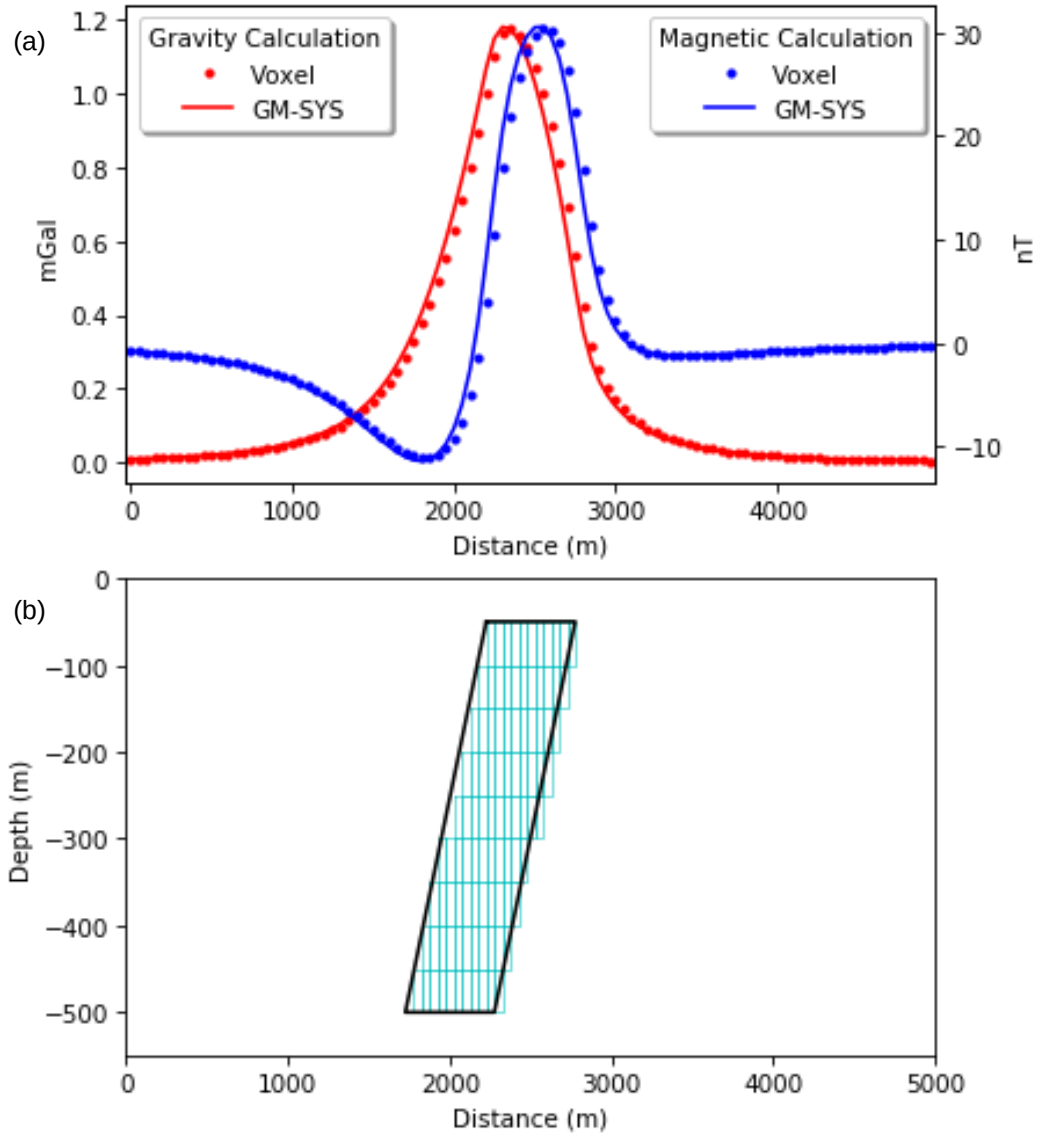

Figure 10 Model of a dipping dyke with a comparison between polygonal (GM-SYS) versus voxel calculations. (a) and (b) show the anomaly and the body respectively. Regional geomagnetic field is 30,000 nT, with inclination of -63 degrees and declination of -17

degrees. The susceptibility 0.01 SI. Density is 2.8 g/cm^3. Remanent magnetisation is 0.199 A/m with inclination of 35 degrees and declination of 80 degrees.

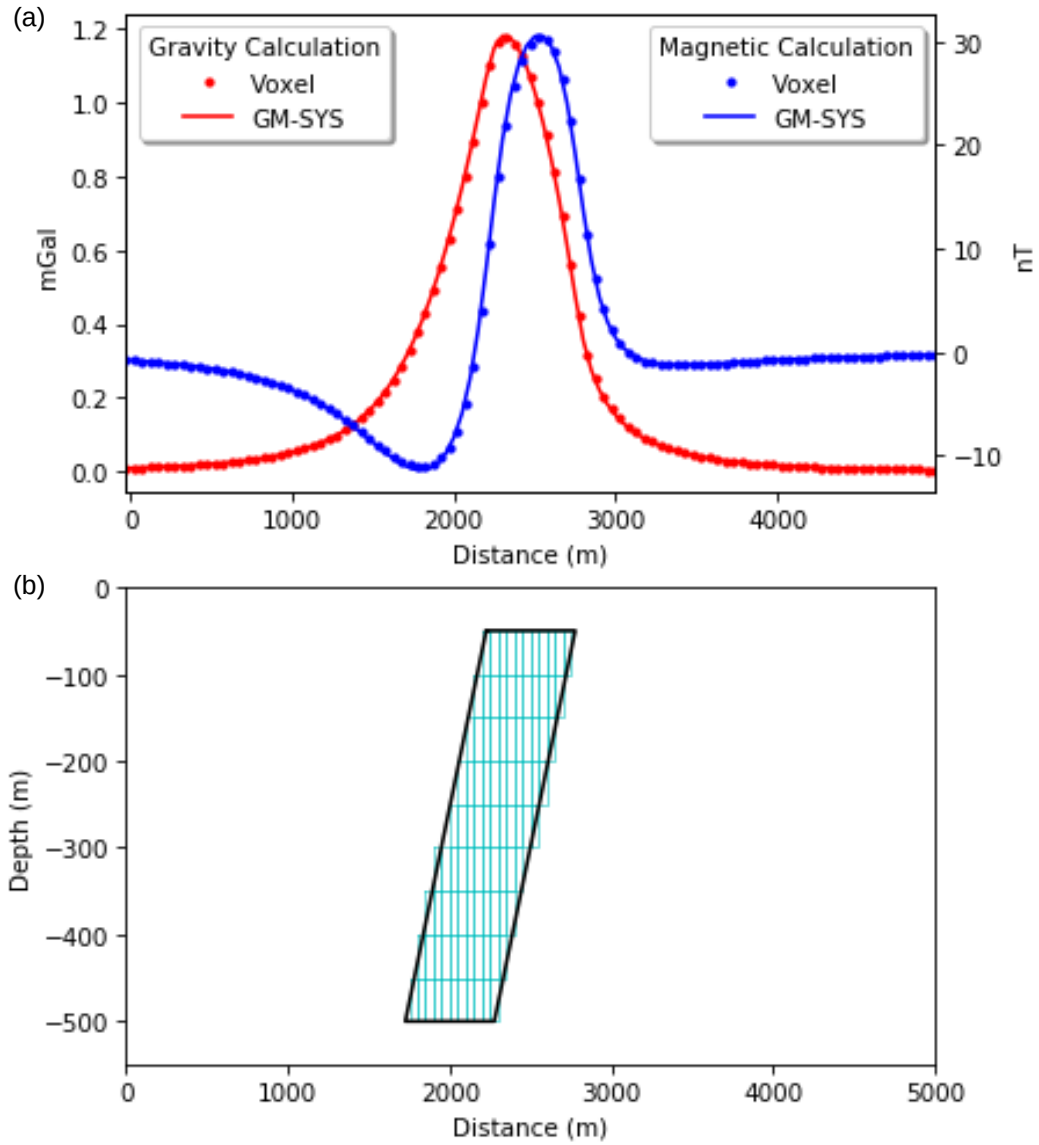

Figure 11 Model of a dipping dyke with a comparison between polygonal (GM-SYS) versus voxel calculations, with the voxels shifted into a slightly more favourable position (half the width of a voxel in this case). (a) and (b) show the anomaly and the body respectively.

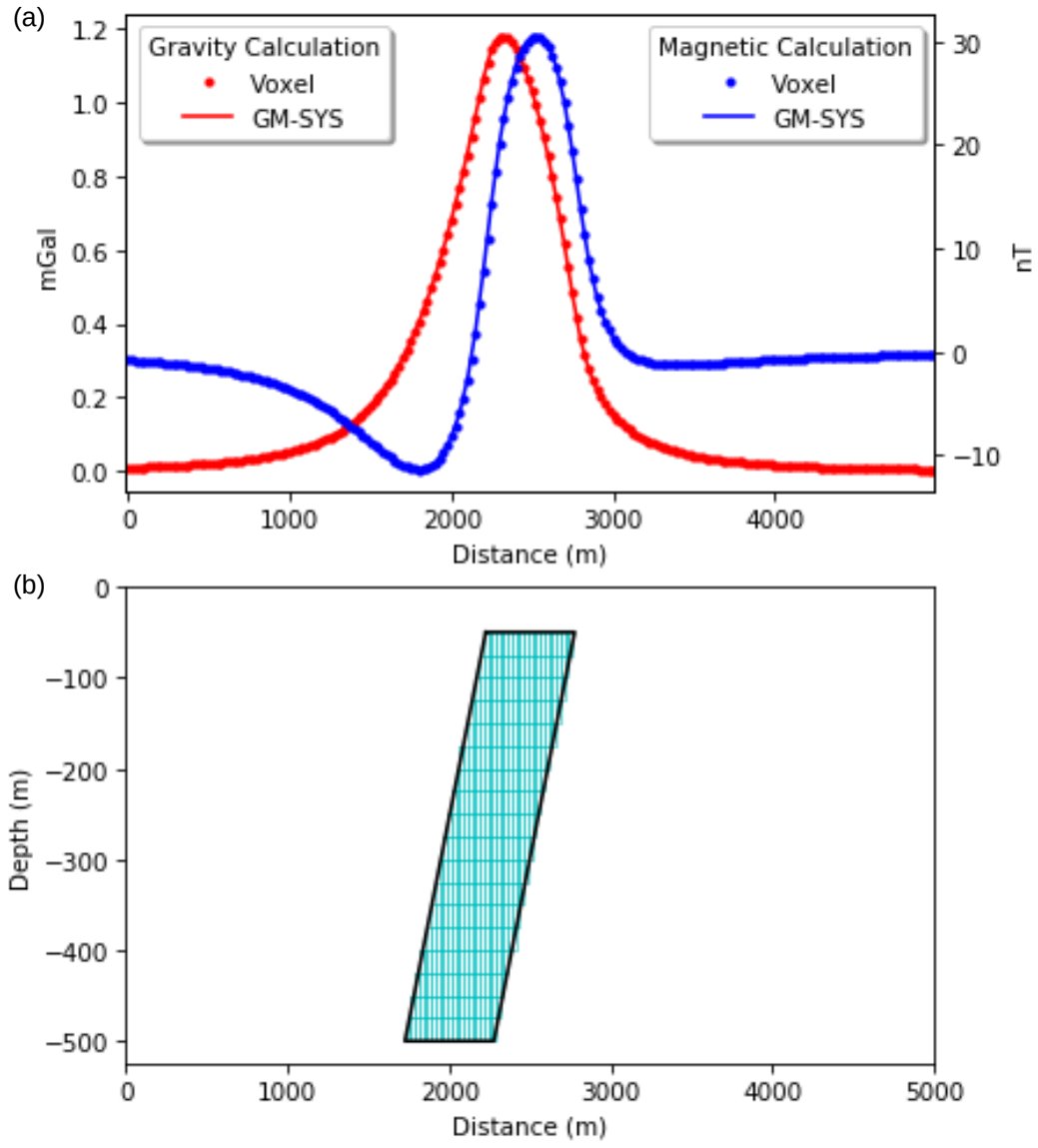

Figure 12 Model of a dipping dyke with a comparison between polygonal (GM-SYS) versus voxel calculations at a higher voxel resolution (twice the resolution in this case). (a) and (b) show the anomaly and the body respectively.

Therefore, in terms of model accuracy, the resolution of the voxel model should be comparable to the resolution of the observed dataset. Increasing the resolution of the voxel model (i.e. decreasing each voxel's dimensions) does allow for enhanced model accuracies to be achieved. This is especially true if one takes into account the inherent modelling error in terms of locations of bodies as input by interpreters and the fact that a simple polygonal dyke is over-simplistic when compared to real geology.

The bottom line is that voxel models are a viable alternative to facet-based models, even with concerns of a model made up essentially of rectangular prisms.

2.3.6 Depth versus resolution and sensitivity

The dipolar and monopolar nature of magnetic and gravity fields result in magnetic fields and gravity fields decaying at a rate of $1/r^3$ and $1/r^2$ respectively (Blakely, 1995, pp.43, 75) . From this, it is clear that the further away geology is from the observation point, the lower its contribution is to the respective potential field. This implies that our effective model resolution is actually lower the further we are from the observation point. This is commonly the case not just in potential field methods, but in all electrical methods.

A side effect of this is that small features in the field tend to relate to shallow sources (i.e. they are close to the observation point) and broad features tend to relate to deeper sources.

As a result of this, optimisations are possible if we take advantage of this property. By lowering the resolution of the model as we move away from the source, less calculations are necessary and the overall calculation time speeds up.

This is straightforward to implement on the strategy outlined in section 2.3.2. Remembering that a rectangular prism is calculated at each depth in the model, the dimensions of the prism can be increased for greater depths. If the horizontal dimension of the rectangular prism is still a multiple of the smallest rectangular prism dimension, there will not need to be any change in any aspect of the strategy.

To illustrate this, Figure 13 shows a simulation of decreasing the model resolution with depth. The z-extent of the model sections were increased in an r^2 manner, with the first layered section being 50 m thick (area between surface and model), the next sections being 150 m and 250 m thick, respectively, with the last section being 50 m thick since the model ends after that point. As is evident, the fit is reasonable, illustrating that depth related optimisations can be done to increase calculation speed. However, from an aesthetic point of view, an interpreter may still prefer seeing (at the very least for conceptual reasons) a fixed voxel resolution throughout the model. The simplification of the model for calculation purposes can be either done behind the scenes or at the control of the interpreter.

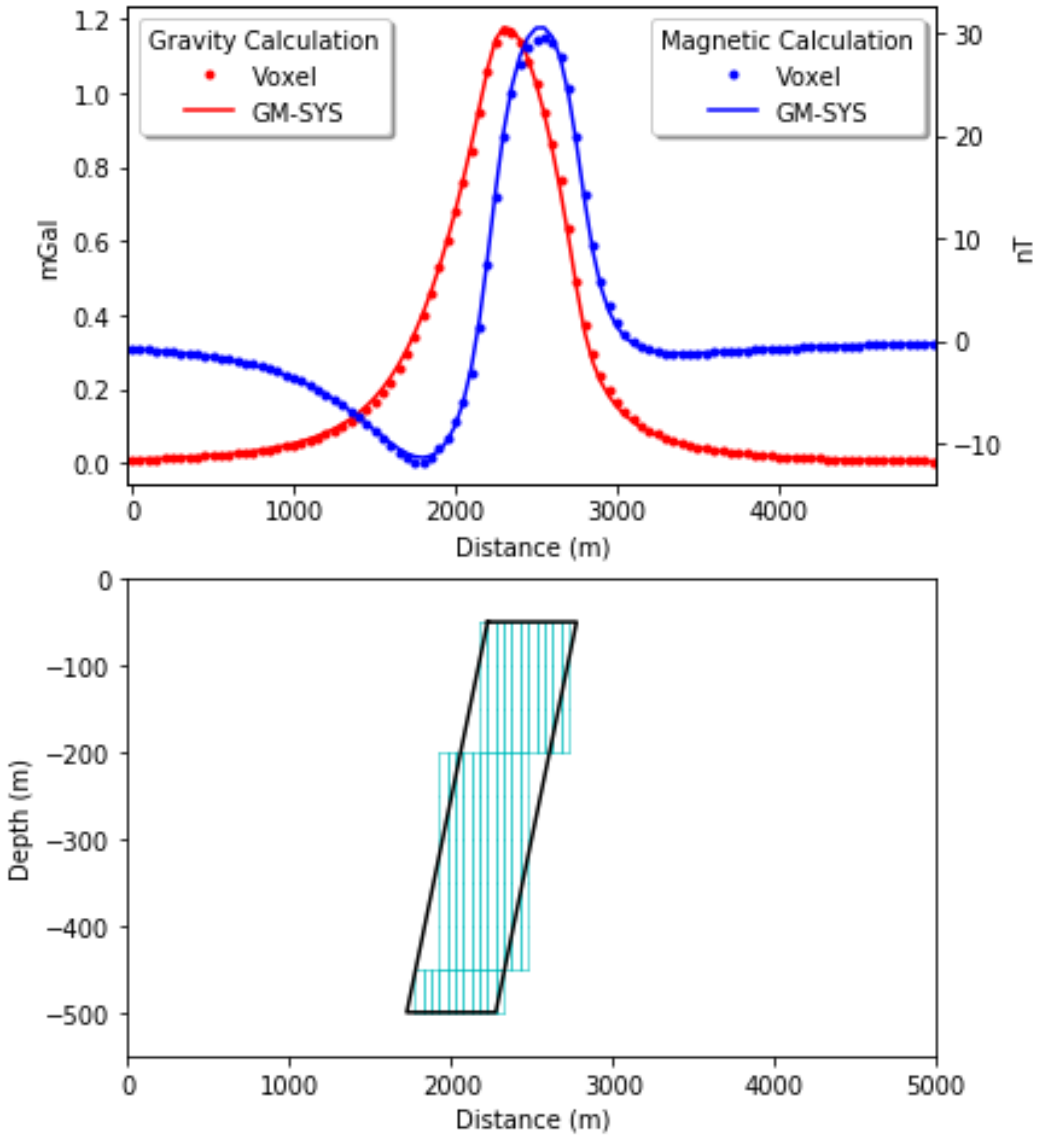

Figure 13 Illustration of the relationship between model depth and anomaly accuracy. As can be seen, the deviation of the voxel dyke at deeper depths has a minor impact on the anomaly.

A final consequence of the importance of features close to the observation point, versus features far away from the observation point, is that the accurate modelling of topography is important. It is the closest feature to the observer and thus the variation of (especially magnetic) topography must be taken seriously and modelled appropriately.

2.3.7 Calculation time

There is no disputing that the calculation time for facets is efficient. Optimisations using Fast Fourier transforms have also been developed to produce rapid modelling (Caratori

Tontini, Cocchi and Carmisciano, 2009). What is in question is whether a traditional voxet based calculation can be optimised so that it is not especially inefficient.

To examine voxel based modelling, the solution for a rectangular prism (Blakely, 1995) was used to calculate the potential field at each observation point. The problem that becomes immediately apparent is that the sheer number of rectangular prisms in any volume quickly becomes too large for any form of efficient calculation.

To illustrate this, since a single calculation of a rectangular prism is necessary to obtain the field from a single observation point, a grid at the surface of the earth of 100 by 100 observations implies 10 000 calculations for one rectangular prism. If the model of the earth volume has 100 rows, 100 columns and 100 layers, this implies 1 000 000 rectangular prisms and 10 000 000 000 calculations.

To optimize this calculation, the components that make up the calculation of a magnetic anomaly need to be understood. There are three basic components

1) A geometric component which is simply due to the distance a magnetic body is from the observation point
2) The field direction which is ultimately comprised of the inducing and remanent fields.
3) The magnetic susceptibility which is in essence a scaling factor to give amplitude to the field.

Point 1 contains no magnetic component. Since this is simply due to distance, the consequence is that if a rectangular prism is at a specific depth, the anomaly from the rectangular prism will be the same as the anomaly of any other rectangular prism at the same depth.

Point 2 is more complex. The components of the magnetic field are woven into the geometric calculation. In spite of this, in the case of two rectangular prisms at the same depth but where one is remanent and the other is not, aspects of the geometric calculation can be reused, saving time for the second rectangular prism calculation.

Point 3 is simply a scaling factor, so two rectangular prisms at the same depth differing by only susceptibility will have final anomalies differing by the magnitude of the susceptibilities only. This is a minor calculation.

Taking advantage of this and remembering that a magnetic lithology means all bodies (rectangular prisms) with the same magnetic parameters – i.e. susceptibility, field

direction, remanence, all anomalies for a rectangular prism of a specific defined lithology at a specific depth will be identical. The resultant overall field is simply the sum of the individual fields from the individual rectangular prisms (Figure 14).

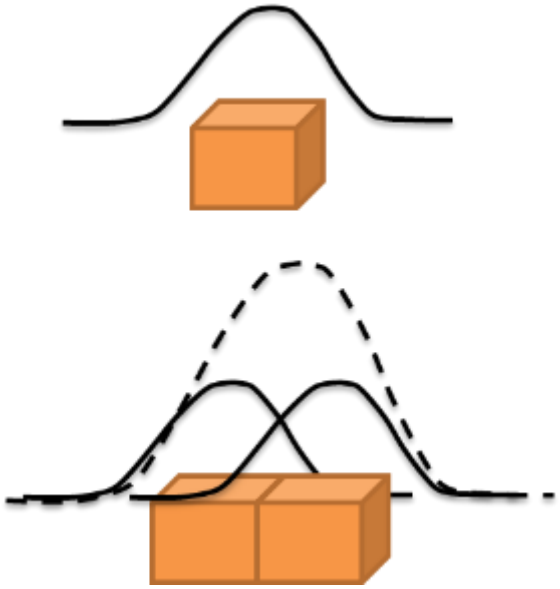

Figure 14 The top image shows the anomaly from a rectangular prism. The bottom image illustrates how these are combined to obtain the resultant anomaly.

In this simplifying case, rectangular prisms are homogenous and there is no topography. Since the anomalies for a rectangular prism at a specific depth are identical, we need only calculate the field of a rectangular prism once, and shift the anomaly results to reflect the locations of other rectangular prisms at the same depth. To account for all possible shifts, the grid of the calculated anomaly must be four times bigger (double the rows and columns) than the desired modelling extents.

A separate anomaly grid must therefore also be calculated for each depth (layer) in the model. From the above calculation, the number of calculations is reduced by the number of rectangular prisms in a layer (10 000) and multiplied by 4 to account for the increased calculations. The new calculation total is 4 000 000 which is 0.04% of the original calculations (Figure 15). This is significantly faster and enables rectangular prisms to be used as a viable alternative for modelling.

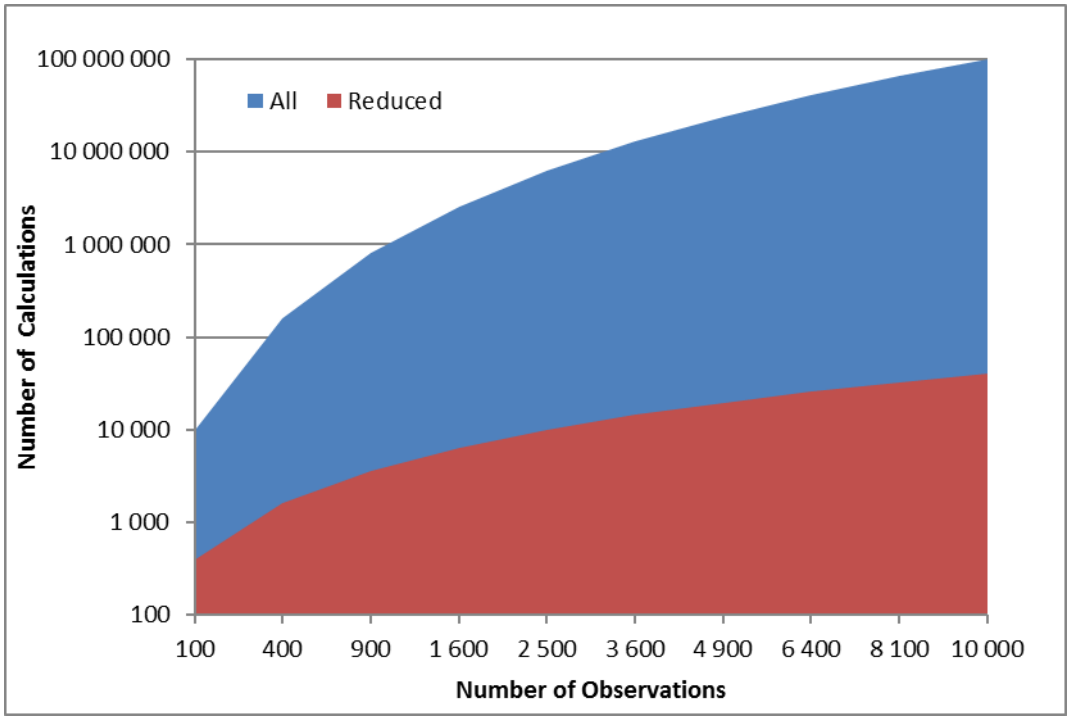

Figure 15 Performance increase using the algorithm

Changes to susceptibility or density do not affect the field shape but merely the amplitude of the anomaly. Therefore, it is not necessary to recalculate the entire field if one of these parameters is changed.

2.3.8 Link to inversion

As mentioned before, in forward modelling, potential field data is calculated by manually postulating geological parameters such as density, susceptibility and the geometry of the body which is being modelled. This resultant field is compared with the measured field, and the process is repeated until the calculated and measured fields match.

In inversion, the measured field is accepted as input, and some or all of the susceptibility, density and geometry values are calculated automatically. Figure 16 illustrates the relationship between forward modelling and inversion.

.

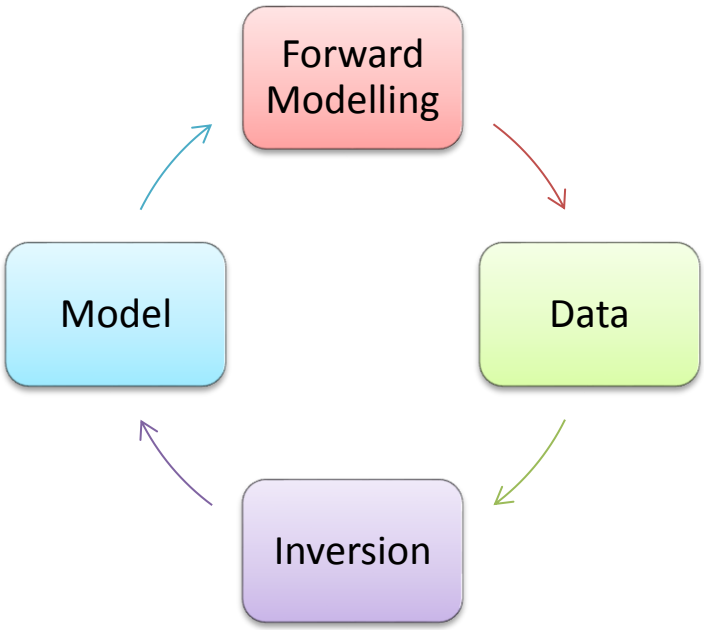

Figure 16 Relationship between forward modelling and inversion. The model is changed (forward modelling) to fit data or the data is used (inversion) to derive a model.

Although inversion can be used to change polygonal geometry, it is often done on voxet models, especially in other techniques such as resistivity and electromagnetic surveys. These voxet models can then be used as input into voxet based potential field forward modelling. These models are therefore more conveniently compatible with voxel based forward modelling. Inversion is discussed in more detail in CHAPTER 4.

2.3.9 Example: Trompsburg Complex

This technique was used to create a very simple 3D model of the Trompsburg Complex, a circular igneous intrusion located in the centre of South Africa (Figure 17(a)). It was forward modelled using both magnetic and gravity data and using the process described in section 2.3.2. Thick Karoo Supergroup sediments cover the Complex and it does not crop out at all (Figure 17(b)). Its presence was only discovered when gravity and later magnetic surveys were conducted over it in the 1940's (Buchmann, 1960; Ortlepp, 1959). Following its discovery seven boreholes were drilled into the north-western part of the Complex where the magnetic intensities are the highest, and the main lithologies encountered were gabbros, olivine gabbro, mineralised gabbro, troctolite and magnetite. The Complex intruded into dolomites at 1915 ± 6 Ma (Maier et al., 2003). A detailed description of the geological setting and physical properties of the lithologies can be found in Maré and Cole (2005). It must be emphasised that the modelling in this report was aimed more at illustrating a new software package, and not to conduct a detailed investigation into the structure and geological development of the Trompsburg Complex.

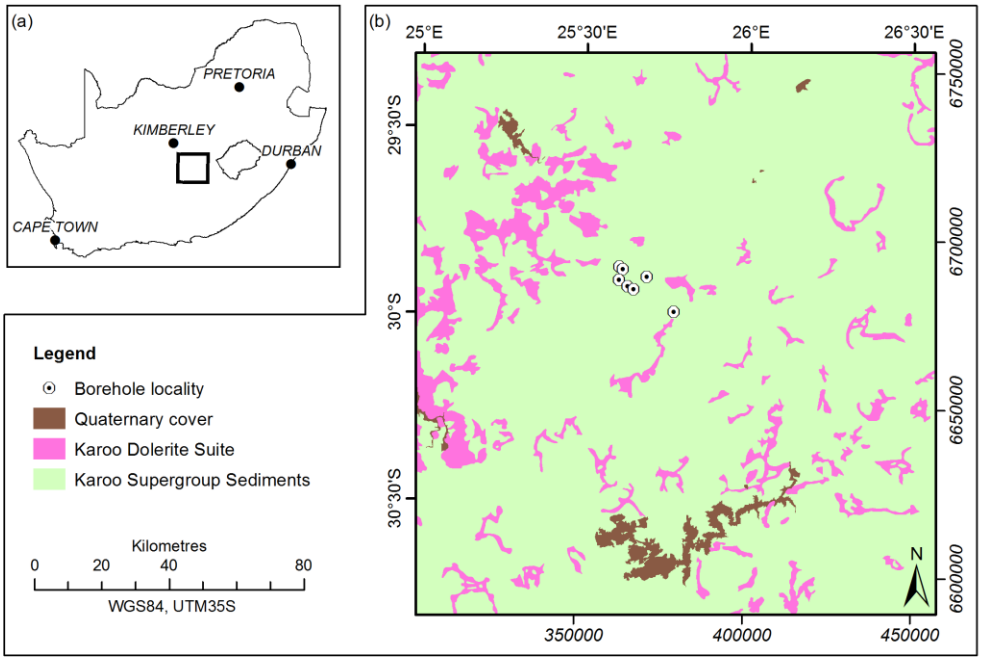

Figure 17 (a) Locality of the Trompsburg Complex (enlarged in (b)); (b) Simplified geological map of the area around the Complex.

Regional gravity and magnetic data collected during the 1970s and 1980s cover the Complex and digital terrain data were extracted from the SRTM (Shuttle Radar Tomography Mission) data set (Farr et al., 2007). The gravity data consists of a measurement roughly every 9 km^2 and the magnetic data were collected along north-south directed flight lines spaced 1 km apart. These data sets were gridded using a minimum curvature algorithm (Briggs, 1974) and using cell sizes of 1 km and 250 m for the gravity and magnetic data respectively. The data sets had extents of 154.7 km in the east-west (x) direction and 166.6 km in the north-south (y) direction. For the model, an x and y cell size of 1000 m and a z cell size of 200 m were chosen, and the total depth (z) extent was specified as 12 km. These settings resulted in a model consisting of 153 columns (x direction), 165 rows (y direction) and 60 layers (z direction).

Figure 18(a) and (b) show the observed Bouguer anomaly and magnetic data respectively over the Trompsburg Complex. The dense, magnetic igneous rocks are responsible for very prominent anomalies in the centre of both data sets. Data were extracted for a larger area around the Complex to avoid edge effects. Linear NNE-SSW and ENE-WSW striking anomalies, to the west and south of the circular anomaly are related to terrain boundaries and were not modelled. A constant regional value of -150

mGal was removed from the Bouguer anomaly data to isolate the anomaly due to the igneous Complex.

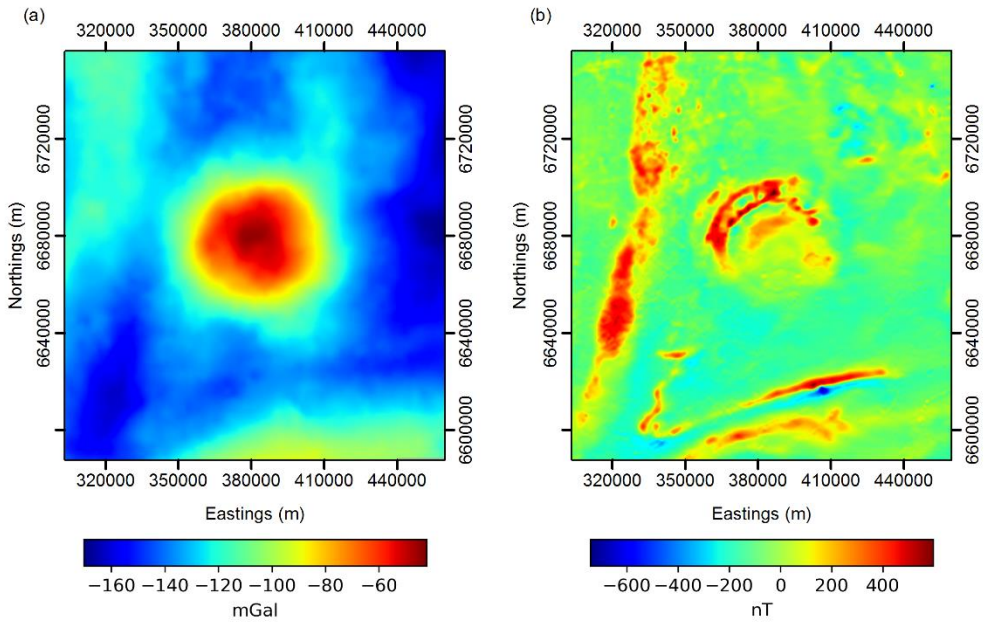

Figure 18 (a) Observed Bouguer anomaly data over the Trompsburg Complex; (b) Observed magnetic data over the Trompsburg Complex. An IGRF has been removed from the magnetic field.

Figure 19(a) shows one of the model layers in plan-view, and Figure 19(b) shows a profile view running in an east-west direction almost through the centre of the model. This profile coincides with two boreholes (positions indicated by black blocks in Figure 19(b)) that were used to constrain the model. Above the model the observed and calculated gravity and magnetic fields along this profile are shown. The major features of the magnetic data are present in the modelled anomaly. The discrepancies can be due to the possibility that remanent magnetisation may be present, but this information was not available. Figure 20 (a) to (e) show perspective views of the model in three dimensions with various lithologies made transparent and opaque. In Figure 20 (a) to (d) the model was exaggerated in the vertical direction, but in (e) no exaggeration was applied. This shows the Complex to be saucer-shaped. The calculated gravity and magnetic field grids are shown in Figure 21 (a) and (b) respectively. For the Complex itself, these grids compare well with the observed grids (Figure 22 (a) and (b)).

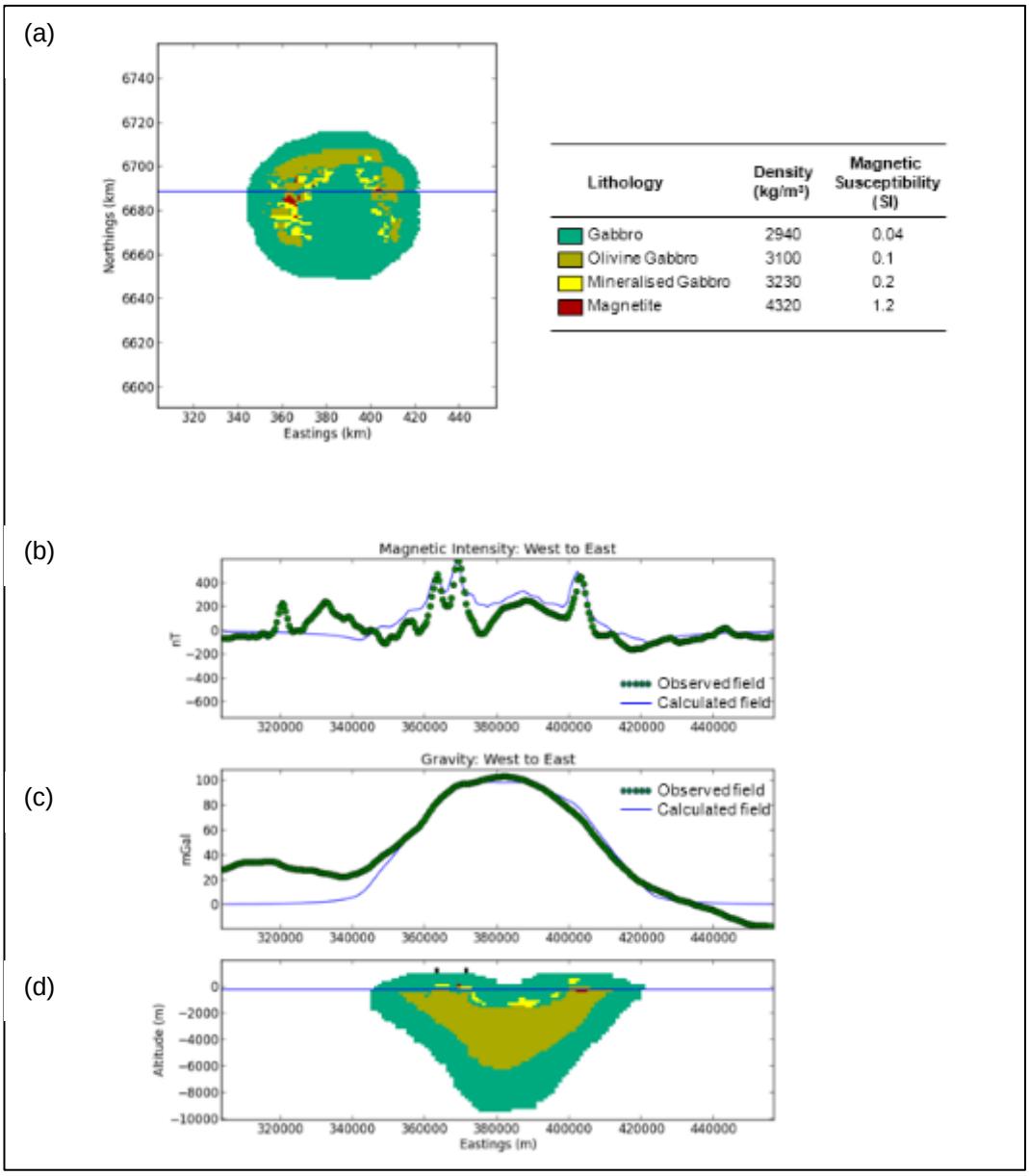

Figure 19 (a) Plan view of layer 11 of the model. The location of profile 11 (shown in (b)) is indicated; (b) Magnetic profile view of profile 98. (c) Gravity profile view of profile 98. The horizontal blue line indicates the layer shown in (a). The vertical exaggeration is roughly 10 times. Black blocks above the model show the locality of two boreholes. (d) Model at profile.

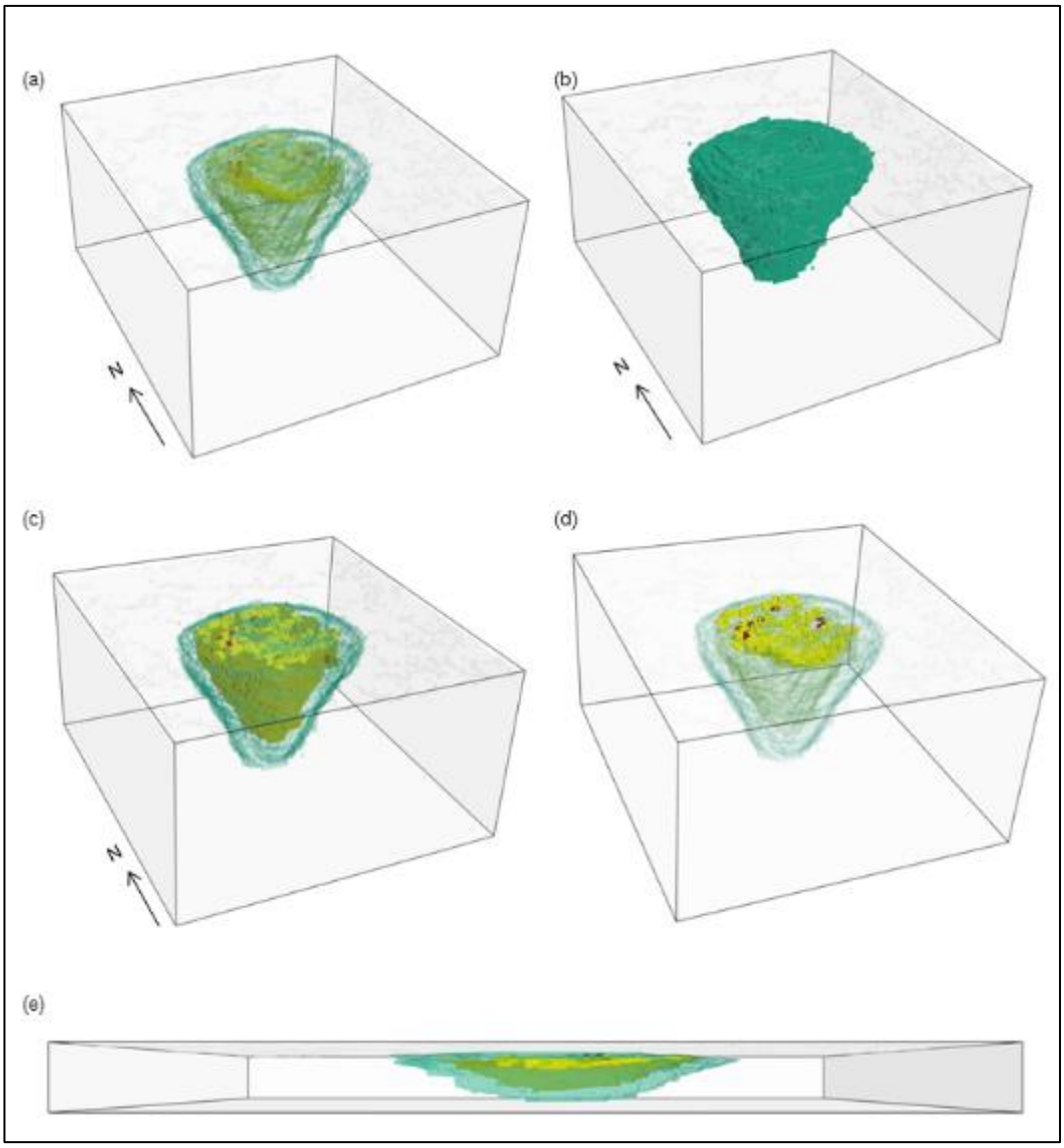

Figure 20 Perspective views of the 3D model. (a) All the lithologies are transparent; (b) Gabbro made opaque to show the extent of the model; (c) Gabbro made transparent, but olivine gabbro, mineralised gabbro and magnetite kept opaque; (d) Gabbro and olivine gabbro made transparent, mineralised gabbro and magnetite kept opaque; (e) View from the south with no vertical exaggeration applied.

Maré and Cole (2005) originally created a 3D model using polyhedral modelling algorithms of Singh and Guptasarma (2001b; 2001a) and Guptasarma and Singh, (1999). They interpreted the Complex as a circular layered intrusion with a feeder in the centre reaching a depth of 16 km. Due to the ambiguity inherent in potential field data, the model shown here (Figure 20) represents one (but not the only) possibility for the geometry of the Trompsburg Complex. A circular layered model was also created, but due to a different distribution of lithologies the body only goes down to a depth of 9 km below the surface and no distinct feeder is visible. However, the primary benefit of this technique is evidenced by the fact that the original model created in 2005 took approximately 3

months to produce using conventional polygonal based techniques. In this case, modelling was done in less than a day. With more time spent, the model accuracy could be improved.

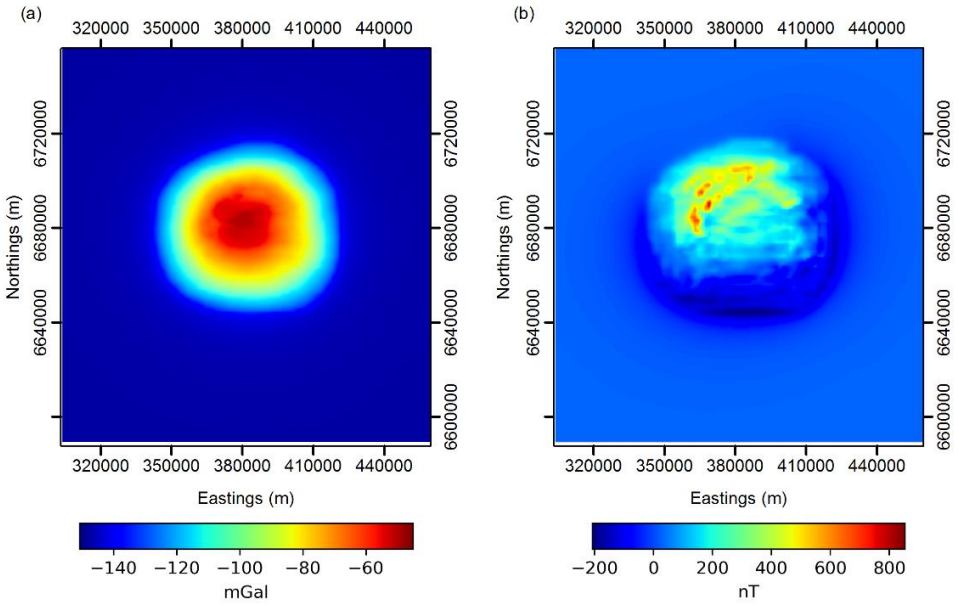

Figure 21 (a) Grid of the calculated gravity field; (b) Grid of the calculated magnetic field.

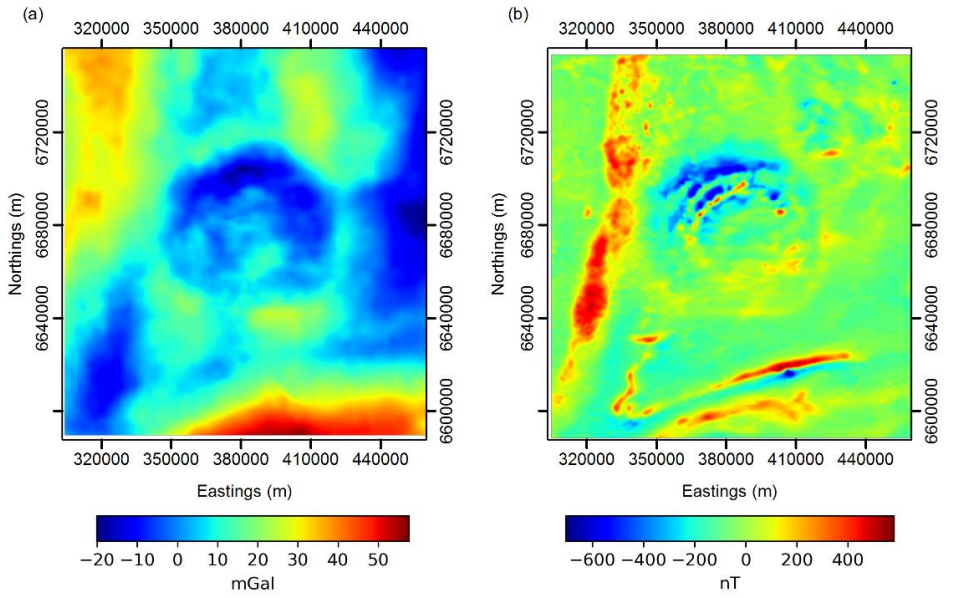

Figure 22 (a) Difference between the observed and calculated gravity fields; (b) Difference between the observed and calculated magnetic fields. The largest differences in the model are associated with the contact between the model and surrounding geology, as well as the surrounding geology. The model itself has a reasonable fit, excluding some areas in the centre which needs additional modelling.

The model was exported to a .kmz file that can be viewed in Google Earth. In Figure 23(a) the model is shown with the lithologies separated. The model is shown above the surface of the earth since it is currently impossible to view below the surface of the earth in Google Earth. The observed and calculated data set can also be viewed in Google Earth, as can be seen in Figure 23 (b) where the observed magnetic data set is shown.

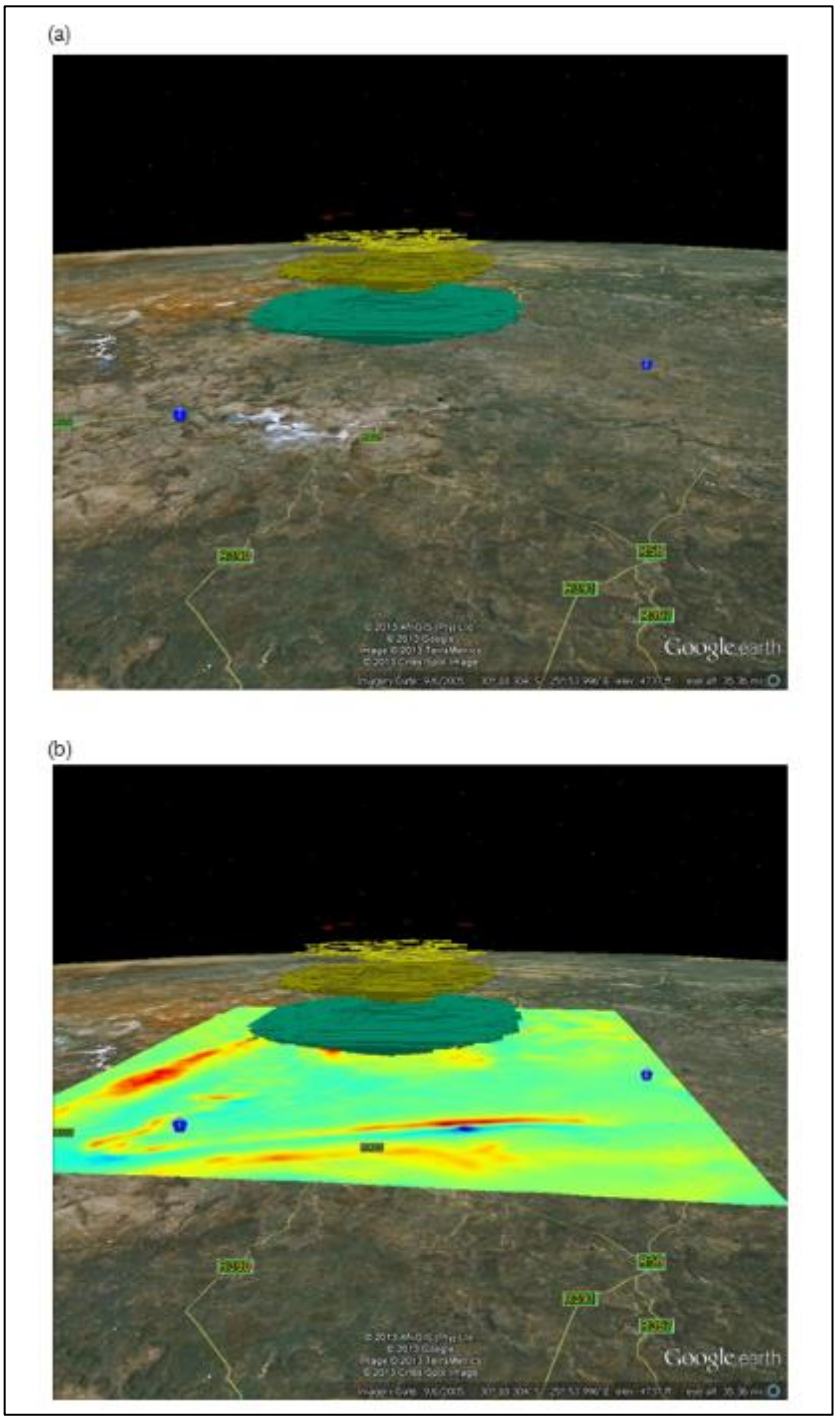

Figure 23 (a) Model imported into Google Earth. The lithologies were separated for better visibility; (b) Observed magnetic data set is also shown

CHAPTER 3 TENSOR FORWARD MODELLING

3.1 Introduction

Forward modelling relations to describe tensor fields due to various sources have been described by numerous authors.

Holstein (2002) found simple relations to express, in tensor form, the gravity and magnetic anomaly solutions for a uniform polyhedron, where the gravity and magnetic solutions are linked through Poisson's differential relation. He derived gravity potential, gravity field and gravity field gradient tensor formulas for a polyhedral target comprising a spatially linear varying density medium as well as the magnetic potential and magnetic field in the case of a medium of spatially linear varying magnetisation (Holstein, 2003). More recently, Holstein, FitzGerald and Stefanov (2013) presented closed formulae for the gravity and magnetic effect due to a homogenous prismatic target. This includes the potential, field and field gradient of gravity and magnetic prismatic targets.

Other work includes forward modelling equations for gravity and magnetic tensors derived for a variety of sources, including rectangular prisms (Heath, 2007), and an interpretation technique that makes use of forward modelling and inversion to construct a realistic 3D model from multiple datasets including geology, physical properties of rocks, topology and tensor data (Guillen et al., 2008).

Tensor measurements hold many advantages over traditional total magnetic intensity surveys (Schmidt et al., 2004; Schmidt and Clark, 2006). Tensors have desirable mathematical properties; tensor elements are true potential fields, allowing, for example, rigorous continuation, RTP (Reduction to the Pole) and magnetization mapping. They have independence from skewing caused by the geomagnetic field direction.

Tensors are measured using superior sensor technology; SQUID sensors have a high sampling rate which allows the unaliased detection of high-frequency aircraft noise. This can be efficiently removed by filtering. Tensor surveys also have all the benefits of vector surveys without the disadvantage of high sensitivity to orientation. It is also possible to perform error correction and noise estimates because of the inherent redundancy in tensor components.

Measured tensors allow the calculation of parameters unaffected by aliasing across flight lines. For example, the determination on which side of a flight line or a drill hole a source

lies. Compact source direction can be defined directly from a single measurement. The calculation of compact source magnetic moments is also possible.

Tensors allow a higher resolution of shallow features and closely spaced sources as well as pipe-like bodies and sources subparallel to flight path. There is also better delineation of N-S elongated sources at low latitudes.

A wider range of new processed quantities unaffected by sensor misorientation is available, including invariants, directional filters, depth slicing, source moments, and dipole locations. These invariant quantities have benefits such as a higher resolving power than the conventional analytic signal.

Tensors allow the direct determination of 3D analytic signal, as well as improved accuracy of Euler deconvolution solutions using true measured gradients along and across lines. Structures can be emphasised in different orientations, since each tensor component represents a directional filter. By rotating the tensor coordinate system, structural orientations can be emphasized. Magnetization direction information can also be obtained via the combination of tensor components.

3.2 Tensors

Although magnetic data is commonly measured as a single value, gradiometer (or tensor) data is becoming more common, especially in airborne gravity surveying. Gradiometer data implies that for each location nine gradient magnetic or gravity values are recorded, instead of simply one overall magnitude of the relevant field (Fitzgerald, Argast and Holstein, 2009). This implies more data is available for modelling, potentially assisting in overcoming modelling ambiguity challenges.

Tensors are an extension to the concepts of scalars, vectors and matrices. A tensor is represented as an organized multidimensional array of numerical values. A practical example of this is the magnetic gradient tensor (Nelson, 1988) :

$$\boldsymbol{B} = \begin{bmatrix} \frac{dB_x}{dx} & \frac{dB_x}{dy} & \frac{dB_x}{dz} \\ \frac{dB_y}{dx} & \frac{dB_y}{dy} & \frac{dB_y}{dz} \\ \frac{dB_z}{dx} & \frac{dB_z}{dy} & \frac{dB_z}{dz} \end{bmatrix} = \begin{bmatrix} B_{xx} & B_{xy} & B_{xz} \\ B_{yx} & B_{yy} & B_{yz} \\ B_{zx} & B_{zy} & B_{zz} \end{bmatrix} \tag{3.1}$$

Where B_x, B_y and B_z are the x, y and z components of the magnetic field. The sheer volume increase in measured data presents many opportunities for new input into source detection and forward modelling algorithms.

3.2.1 Tensor Rank

Tensors may be classified by rank or order (Kolecki, 2002). This classification is reflected in the number of components a tensor possesses in N-dimensional space. Therefore, a tensor of order p has N^p components.

As an example, in a three-dimensional Euclidean space, the number of components of a tensor is 3^p. From this, for example:

- A zero order tensor ($p = 0$) has one component and is called a scalar. Physical quantities possessing magnitude only are represented by scalars.
- A tensor of order one ($p = 1$) has three components and is called a vector. Quantities possessing both magnitude and direction are represented by vectors. B_x is an example of a first rank tensor component.
- A tensor of order two ($p = 2$) has nine components and is typically represented by a matrix. $B_{xx} = \frac{dB_x}{dx}$ is an example of a second rank tensor component

- A tensor of order three ($p = 3$) has twenty seven components. $B_{xxy} = \frac{d^2 B_x}{dxdy}$ is an example of a third rank tensor component.

3.2.2 Structure Tensor

A structure tensor is a matrix derived from the gradient of a function. It is a second order tensor (has components such as B_{xy}), and has 2D and 3D forms (can be represented by either two or three variables). The 3D form is used in gradiometer surveys. Assume that $\boldsymbol{B}$ is a function of three variables (x, y, z). We can recognise that since $\boldsymbol{H} = -\nabla\varphi$, this implies that (in SI units) $B_x = -\mu_0 \frac{d\varphi}{dx}$, $B_y = -\mu_0 \frac{d\varphi}{dy}$, $B_z = -\mu_0 \frac{d\varphi}{dz}$ Therefore, combining (3.1) and (2.23), the structure tensor would be (Heath, Heinson and Greenhalgh, 2003):

$$\begin{aligned}
\boldsymbol{B} &= \nabla \otimes \mu_0 \nabla\varphi \\
&= \begin{bmatrix} \frac{d}{dx} \\ \frac{d}{dy} \\ \frac{d}{dz} \end{bmatrix} \begin{bmatrix} -\mu_0 \frac{d\varphi}{dx} & -\mu_0 \frac{d\varphi}{dy} & -\mu_0 \frac{d\varphi}{dz} \end{bmatrix} \\
&= -\mu_0 \begin{bmatrix} \frac{d^2\varphi}{dx^2} & \frac{d^2\varphi}{dxdy} & \frac{d^2\varphi}{dxdz} \\ \frac{d^2\varphi}{dxdy} & \frac{d^2\varphi}{dy^2} & \frac{d^2\varphi}{dydz} \\ \frac{d^2\varphi}{dxdz} & \frac{d^2\varphi}{dyz} & \frac{d^2\varphi}{dz^2} \end{bmatrix} \\
&= \begin{bmatrix} B_{xx} & B_{xy} & B_{xz} \\ B_{yx} & B_{yy} & B_{yz} \\ B_{zx} & B_{zy} & B_{zz} \end{bmatrix}
\end{aligned} \tag{3.2}$$

Where $\otimes$ is the dyadic product. Since the magnetic field is a potential gradient (see equation (2.23)), the tensor components of (3.2) are second derivatives of the scalar potential (φ). The consequence of this is symmetry of the tensor components. Therefore:

$$B_{xy} = B_{yx}, \qquad B_{yz} = B_{zy}, \qquad B_{xz} = B_{zx} \tag{3.3}$$

According to Laplace's equation:

$$\nabla^2\varphi = 0 \tag{3.4}$$

$$\nabla \times \boldsymbol{B} = 0 \tag{3.5}$$

From this it can be seen that:

$$B_{xx} + B_{yy} + B_{zz} = 0 \tag{3.6}$$

Based on (3.3) and (3.6), (3.2) can be re-written as:

$$\boldsymbol{B} = \begin{bmatrix} B_{xx} & B_{xy} & B_{xz} \\ B_{yx} & B_{yy} & B_{yz} \\ B_{zx} & B_{zy} & B_{zz} \end{bmatrix} = \begin{bmatrix} B_{xx} & B_{xy} & B_{xz} \\ B_{xy} & B_{yy} & B_{yz} \\ B_{xz} & B_{yz} & -B_{xx} - B_{yy} \end{bmatrix} \tag{3.7}$$

This means that there are only five independent tensor components with magnetic and gravity data. For magnetic data these are $B_{xx}, B_{xy}, B_{xz}, B_{yy}, B_{yz}$.

3.2.3 Eigenvector and Eigenvalue Tensor Analysis

An alternative form of tensor representation, based on amplitudes and phases, is discussed by Fitzgerald et al., (2007). Eigenvalues and eigenvectors provide a means to transform a tensor measurement. Each reading is decomposed into the invariant eigenvalue amplitudes and orthogonal rotation matrix with associated eigenvectors local to the survey reference frame. The eigenvalue amplitudes and eigenvector rotations represent the amplitude and phase of the tensor. The amplitude-phase form allows for alternate fast and robust processing of tensor data while respecting the intrinsic physical properties of tensors.

Clark (2012) gives a good overview of eigenvector analysis of the tensor If we define a tensor measurement as the matrix $\boldsymbol{B}$ with a scalar eigenvalue λ and eigenvector $\boldsymbol{v}$ then the relationship between these quantities is:

$$\boldsymbol{B}\boldsymbol{v} = \lambda\boldsymbol{v} \tag{3.8}$$

The eigenvalues are solved by solving the characteristic equation $\det(\boldsymbol{B} - \lambda\boldsymbol{I})$. Expanding this, we get:

$$\lambda^3 + I_1\lambda - I_2 = 0 \tag{3.9}$$

where

$$\begin{aligned} I_1 &= B_{yy}B_{zz} + B_{xx}B_{yy} + B_{zz}B_{xx} - B_{xy}^2 - B_{yy}^2 \\ &= \lambda_1\lambda_2 + \lambda_1\lambda_3 + \lambda_2\lambda_3 \\ &= -(\lambda_1^2 + \lambda_2^2 + \lambda_3^2)/2 \end{aligned} \tag{3.10}$$

$$\begin{aligned} I_2 = \det(\boldsymbol{B}) &= B_{xx}B_{yy}B_{zz} + B_{xx}B_{yz}^2 + B_{zz}B_{xy}^2 - 2B_{xy}B_{xz}B_{yz}B_{xy}^2 \\ &= \lambda_1\lambda_2\lambda_3 \end{aligned} \tag{3.11}$$

Where $\lambda_1 \geq \lambda_2 \geq \lambda_3$. Applying eigenvector analysis to our tensor equation, as shown in equation (3.7), we obtain 3 eigenvalues and 3 eigenvectors. The rotation matrix $\boldsymbol{R}$ which has as its columns the eigenvectors $[\widehat{\boldsymbol{v_1}}, \widehat{\boldsymbol{v_2}}, \widehat{\boldsymbol{v_3}}]$, diagonalises $\boldsymbol{B}$ when applied to it. It is straightforward to verify that the following holds:

$$\boldsymbol{R^T B R} = \boldsymbol{R^T} \begin{bmatrix} B_{xx} & B_{xy} & B_{xz} \\ B_{yx} & B_{yy} & B_{yz} \\ B_{zx} & B_{zy} & B_{zz} \end{bmatrix} \boldsymbol{R} = \begin{bmatrix} \lambda_1 & & \\ & \lambda_2 & \\ & & \lambda_3 \end{bmatrix} \tag{3.12}$$

The eigenvalues, and any combination thereof, are rotational invariants of the tensor. Clark (2012) makes extensive use of one such rotational invariant, namely the normalised source strength. It is defined as:

$$\mu_{nss} = \sqrt{-\lambda_2^2 - \lambda_1\lambda_3} \tag{3.13}$$

Clark (2012) points out that unlike the tensor magnitude (Frobenius norm) $|\boldsymbol{B}|$, μ is completely isotropic around a dipole source. This makes it ideal for homing applications.

3.2.4 Axis Conventions – implications for tensor data

Holstein et al. (2015) discuss at length the differences between different axis convention systems, especially when applied to potential field tensor data. They formulated an intuitive conversion between different reference systems. They further concluded that all these systems give legitimate ways of representing vectors and tensors, but that processing software should give the necessary flexibility to handle each of them, or if necessary, a mixture.

In tensor notation, a NED tensor, which is right handed, may be represented as:

$$\boldsymbol{T} = \begin{bmatrix} T_{NN} & T_{NE} & T_{ND} \\ T_{NE} & T_{EE} & T_{ED} \\ T_{ND} & T_{ED} & T_{DD} \end{bmatrix} = \begin{bmatrix} T_{11} & T_{12} & T_{13} \\ T_{21} & T_{22} & T_{23} \\ T_{31} & T_{32} & T_{33} \end{bmatrix} \tag{3.14}$$

However, the ENU tensor, which is also right handed, would be represented as:

$$\boldsymbol{T} = \begin{bmatrix} T_{EE} & T_{NE} & T_{EU} \\ T_{NE} & T_{NN} & T_{NU} \\ T_{EU} & T_{NU} & T_{UU} \end{bmatrix} = \begin{bmatrix} T_{22} & T_{21} & -T_{23} \\ T_{12} & T_{11} & -T_{13} \\ -T_{32} & -T_{31} & T_{33} \end{bmatrix} \tag{3.15}$$

The END convention, which is a left handed convention, is:

$$\boldsymbol{T} = \begin{bmatrix} T_{EE} & T_{NE} & T_{ED} \\ T_{NE} & T_{NN} & T_{ND} \\ T_{ED} & T_{ND} & T_{DD} \end{bmatrix} = \begin{bmatrix} T_{22} & T_{21} & T_{23} \\ T_{21} & T_{11} & T_{13} \\ T_{32} & T_{31} & T_{33} \end{bmatrix} \tag{3.16}$$

Note that the numerical subscripts show the relation between the tensors, where $N = 1, E = 2$ and U or $D = 3$ Notice that signs only swap between Up and Down when changing between left and right handed systems. Thus, care should be taken to understand the axis convention when examining tensor data.

Conversion between the two systems (NED and ENU) can be achieved by using the following rotation matrix:

$$\boldsymbol{R} = \begin{bmatrix} 0 & 1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & -1 \end{bmatrix} \tag{3.17}$$

And applying it in the following way:

$$\boldsymbol{T'} = \boldsymbol{R^T T R} \tag{3.18}$$

where $\boldsymbol{T'}$ is a new tensor matrix transformed from the old tensor matrix $\boldsymbol{T}$

The specification of axis convention is also important when reporting eigenvalue and eigenvector data derived from tensor data. The eigenvalues represent amplitudes and the eigenvector represents rotations or phase. Although eigenvalues will be the same, the eigenvectors will differ for different conventions.

In tensor notation, a NED set of eigenvectors may be represented as:

$$\boldsymbol{v} = \begin{bmatrix} v_{11} & v_{21} & v_{31} \\ v_{12} & v_{22} & v_{32} \\ v_{13} & v_{23} & v_{33} \end{bmatrix} \tag{3.19}$$

However, the relation from this to ENU eigenvectors would be represented as:

$$\boldsymbol{v} = \begin{bmatrix} -v_{12} & v_{22} & v_{32} \\ -v_{11} & v_{21} & v_{31} \\ -v_{13} & v_{23} & v_{33} \end{bmatrix} \tag{3.20}$$

The relation to the END convention is:

$$\boldsymbol{v} = \begin{bmatrix} v_{12} & v_{22} & v_{32} \\ v_{11} & v_{21} & v_{31} \\ v_{13} & -v_{23} & -v_{33} \end{bmatrix} \tag{3.21}$$

3.3 Tensor Acquisition

Although tensor gradiometry is still in its infancy, Stolz et al. (2006) give a good brief overview as to its history. They report that the first proposals for an airborne magnetic gradiometer system was made by Fromm in 1952. Subsequent to this, Morris and Pedersen achieved this in 1961 with two rigidly connected fluxgates. The challenge in this method is in keeping the sensor axes parallel as well as eliminating airborne noise.

Since then, much effort has been made to improve on this original design, culminating in improved gradiometers based on superconducting quantum interference devices (SQUIDs) cooled with liquid nitrogen (high temperature superconductor, 77 Kelvin) or liquid helium (low temperature superconductor, 4.2 Kelvin). These devices have low intrinsic noise and are extremely sensitive detectors of magnetic field components (Clarke and Braginski, 2004), and gradient tensor components when appropriately configured.

Different configurations of SQUID magnetometers have been developed over the years. Schmidt et al., (2004) describe a system developed by the CSIRO called GETMAG. Keene, Humphrey and Horton (2005) describe a system using four individual SQUID magnetometers. Stolz et al. (2006) report that the development of an airborne Full Tensor Magnetic Gradiometry (FTMG) SQUID system by the Institute of Photonic Technology (IPHT) began in 1997, and was named JeSSY STAR. It measures both the magnetic field vector and the gradient tensor. No scalar magnetometer is assembled to measure the TMI directly (Schiffler et al., 2017). Obtaining TMI for JeSSY STAR is described in Schiffler et al. (2014).

3.3.1 Hardware design

Tensor gradiometer systems typically consist of a series of magnetic gradiometer systems arranged in some configuration. Eschner and Ludwig (1995) filed a patent for a system using planar gradiometers and this has been referenced by both Stolz et al. (2015); and Billings (2012). A more complete description of the hardware design can be obtained from Billings (2012), however it is worth describing the tensor sensor configuration to better understand processing later.

A series of six planar gradiometers as well as three orthogonal magnetometers are arranged in a pyramidal structure, to enable the determination of the full magnetic tensor gradient (Figure 24).

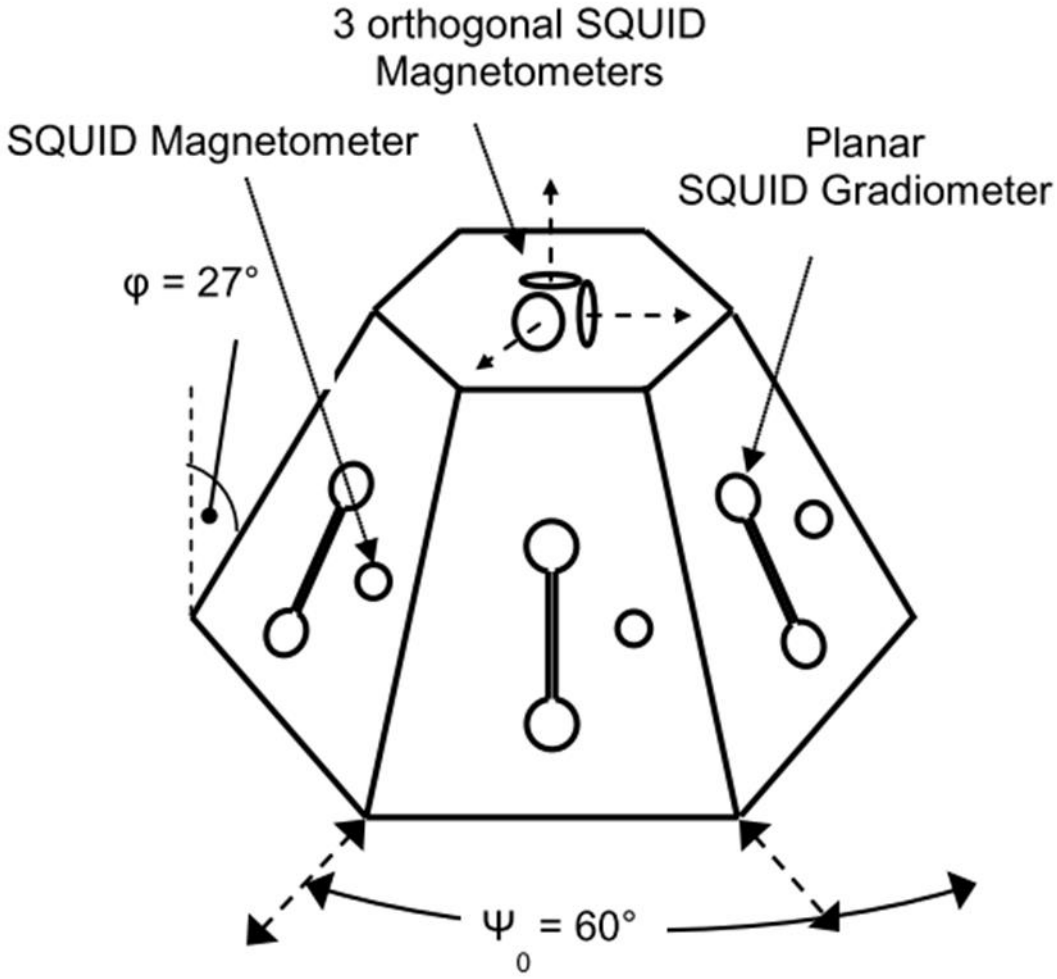

Figure 24 The orientation of the SQUID magnetometer sensors, from Billings (2012)

The configuration of these gradiometers implies that each gradiometer measures a mix of the desired tensor components. However, the geometry of the sensors means that these components can be unmixed. Therefore, the signals measured by gradiometers are not to be confused with the final tensor components.

This pyramidal sensor configuration is then cooled in a Dewar or vacuum flask, using liquid nitrogen. From here, electronics pass the squid signals through to analogue to digital convertors, and from there to the control computer. The system by Billings (2012) also had an external fluxgate magnetometer to provide magnetic vector information.

Of equal importance is an IMU (inertial measurement unit) which consists of three orthogonally oriented gyroscopes and three accelerometers, and a differential GPS receiver. Both the IMU and the GPS are used to correct for the attitude of the instrument.

The GETMAG system is SQUID based system designed by the CSIRO meant for use on the ground. Its principle is slightly different in that it is based on the concept of three rotating axial gradiometers in an umbrella configuration (Schmidt et al., 2004). The initial prototype of this system used one axial gradiometer, which could be manually rotated about a z'-axis (an axis oriented at 45 degrees with respect to the horizon) through eight discrete fixed positions spaced 45 degrees apart. The system can be moved through 120 degree increments around the vertical z-axis. (Figure 25)

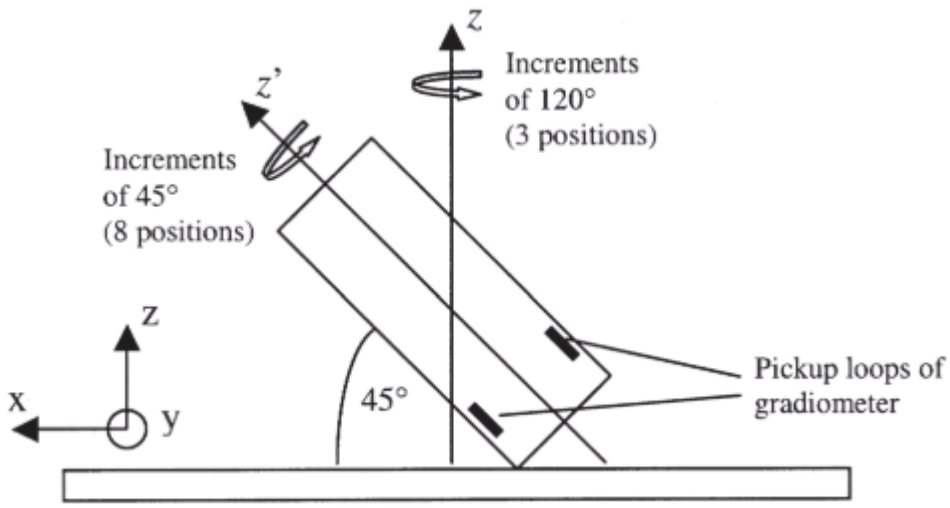

Figure 25 A single axial gradiometer from the GETMAG system, from Schmidt et al., (2004)

This system operates in liquid nitrogen at 77 K. The axial gradiometer measures the first derivative of the magnetic field and consists of a directly coupled SQUID magnetometer, a superconducting flux transformer and a superconducting shield. These components are fixed to the base of a radio-frequency shielded Dewar flask. Electronics are connected to this for the purpose of collecting and filtering the data.

3.3.2 Practical survey issues

Schiffler et al. (2017) report that SQUID recordings are strongly corrupted by motion noise due to the permanent rotation of the sensor during operation while in flight. Noise removal from the magnetic field vector (B_x, B_y, B_z) cannot be done simply by geo-referencing since the attitude data delivered by inertial measurement units (IMUs) is of insufficient quality. However, the quality is adequate for the tensor measurements. Therefore, the vector can be either calculated from the tensor, or the TMI.

One problem of this instrument is the superposition of the components of the magnetic field over the measured gradient components. This effect has been regarded as a parasitic effect (Schiffler et al., 2017) that arises due to fabrication limitations of the sensors. To discriminate these parasitic signals from the desired gradient component, the components of the magnetic field is also measured by three highly sensitive reference magnetometers. The purpose of these magnetometers is only to compensate for the parasitic field, and Schiffler et al. (2017) report that for their system they have an intrinsic noise of 7 $\mathrm{pT/Hz^{1/2}}$ with a dynamic range of $\pm$ 100 μT. The digitized magnetic field has an accuracy of $\pm$ 12 pT.

In reality the main part of the noise is due to the imperfect rotation of the magnetic field vector and the magnetic gradient tensor quantities into a local coordinate frame (Schiffler et al., 2017). Correction of this noise is achieved via the IMU and a differential GPS receiver. This IMU and GPS data is used for geo-referencing, spatial orientation and rotation of the body-frame data.

The attitude is represented by means of Euler angles. Schiffler et al. (2017) mention that their IMU provided an accuracy of $\pm 1°_{RMS}$ for the roll and pitch angles and $\pm 10°_{RMS}$ for the heading angle (yaw). Roll, pitch and yaw will be discussed in section 3.4.4

3.4 Tensor Processing

Tensor processing is described by Argast et al. (2010), Schiffler et al. (2014) and Schiffler et al. (2017), as applied to the JeSSY STAR system. The steps to be followed will be described in sequence in this section.

3.4.1 Balancing of gradiometers

Argast et al. (2010) and later Schiffler et al. (2014) describe a calibration technique (balancing) used to remove the parasitic leakage of the "*B*" field into the measured mixed gradients.

Balancing of the gradiometers is the process whereby the parasitic magnetic influences are compensated for. It consists of the measured gradient component, the parasitic influence and a frequency dependent term originated by eddy currents. The eddy currents can be ignored in the JeSSY STAR system due to electronic design and attitude changes at frequencies below 10 Hz. The measured (corrupted) signal g_k is given by (Schiffler et al., 2017):

$$g_k = \Delta G_k + \left(\tilde{G}_k + \sum_{i=x,y,z} \alpha_{ik} B_i \right) \tag{3.22}$$

where ΔG_k is the uncorrupted gradiometer signal, $\tilde{G}_k$ is the gradiometer offset, B_i is the field component in the direction $i = (x, y, z)$ and α_{ik} represents the size of the three orthogonal parasitic areas, with $k = \{1, \dots, 6\}$.The α_{ik} (balancing coefficients) are estimated by minimising the variance of ΔG_k using points where the magnetic gradient are zero and no regional long wavelength gradient components are present. Argast et al. (2010) state that the balancing coefficients are then averaged over all lines flown in the same direction and are used to remove the parasitic *B* field from the measured mixed gradients.

The remaining noise in the magnetic gradient tensor is due to the estimates being flawed with standard errors in the least squares sense.

3.4.2 Calibration of reference magnetometers

Reference magnetometer calibration converts raw readings into magnetic field values. It uses three misalignments, three sensitivities and three arbitrary offsets, described by (Schiffler et al., 2017):

$$\boldsymbol{F} = \boldsymbol{S}\boldsymbol{D}_{\boldsymbol{dist}}\boldsymbol{B} + \boldsymbol{O} \tag{3.23}$$

where $\boldsymbol{F} = \left(F_x, F_y, F_z\right)^T$ is the raw measurement data, $\boldsymbol{B} = \left(B_x, B_y, B_z\right)^T$ is the magnetic field, $\boldsymbol{S}$ is the sensitivity matrix, $\boldsymbol{D}_{\boldsymbol{dist}}$ is the distortion matrix and $\boldsymbol{O} = \left(O_x, O_y, O_z\right)^T$ is the vector of the three offsets (SQUID magnetometers generally have unknown offsets). The distortion matrix is the transformation matrix between the ideal coordinate systems and the non-orthogonal sensor system and contains the misalignments. A minimisation routine is used to solve for all this in order to achieve calibration.

3.4.3 Decomposition of signals

Signals are decomposed into balanced gradiometer signals using the sensor head and gradiometer mounting geometry. Billings (2012) describe this process applied to the pyramidal tensor sensor design. The six gradiometer outputs corresponding to the six pyramid faces are linear combinations of the gradient tensor in the instrument frame. Therefore, from each face, the calibrated gradiometer outputs $B_{x'_k z'_k}$ are obtained, with the faces designated by $k = 1, \dots, 6$. This can be described by:

$$\boldsymbol{G}' = \left[B_{x'_1 z'_1}, \dots, B_{x'_6 z'_6}\right] = \boldsymbol{M}\left[B_{xx}, B_{xy}, B_{xz}, B_{yy}, B_{zz}\right] \tag{3.24}$$

The equation for $B_{x'_k z'_k}$ is given by (Billings, 2012):

$$\begin{aligned} B_{x'_k z'_k} = \frac{1}{2}\Big(B_{xx}(1 + \cos^2\psi_k) + B_{yy}(1 + \sin^2\psi_k)\Big)\sin 2\varphi \\ + \frac{1}{2}\sin 2\varphi \sin 2\psi_k \, B_{xy} \\ + \cos 2\varphi \cos\psi_k B_{xz} + \cos 2\varphi \sin\psi_k \, B_{yz} \end{aligned} \tag{3.25}$$

where ψ_k denotes the horizontal orientation of the gradiometer on the hexagonal sensor frame (see Figure 24) and φ denotes the slope of the hexagonal pyramid side on which the sensor is mounted. The six equations resulting from equation (3.25) are then inverted to obtain a least squares best fit. This process assumes that the applied gradient is uniform. The challenge with this process is that the rank of the matrix being inverted must be high enough for the inversion to succeed.

3.4.4 Rotation and Euler Angles

Euler angles (Rossberg, 1983, pp.228–230; Diebel, 2006) are three angles used to describe the orientation of a rigid body (sensor or aircraft in this case) with respect to a fixed coordinate system. Any orientation can be achieved by three rotations about the axis of a coordinate system. As an example, assume that the axes of the original frame is defined as (x, y, z) and the rotated frame as (x', y', z'). The rotations work as follows (Figure 26):

1) The first rotation is by an angle ψ about the z axis, creating a new axis (x''', y''', z''').
2) The second rotation is by an angle θ about the x''' axis, creating a new axis (x'', y'', z'') .

3) The third rotation is by an angle ϕ around the z'' getting to final axis (x', y', z').

This particular Euler convention is known as the z-x-z convention, since the rotations take place around those axes. In reality there are six possible conventions for proper Euler angles.

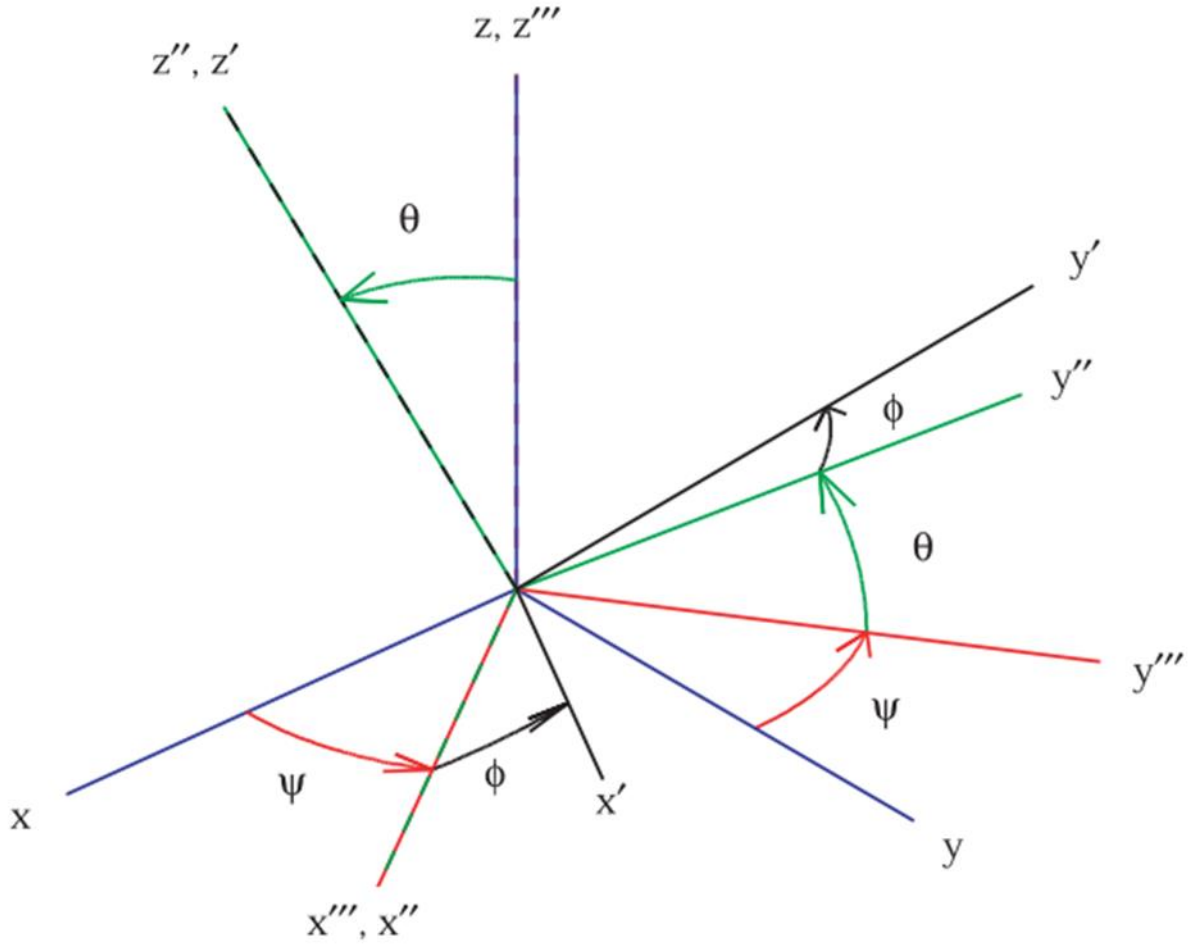

Figure 26 Proper Euler angles where (x, y, z) is shown in blue and (x''', y''', z''') is shown in red. (x'', y'', z'') is shown in green and the final position (x', y', z') is shown in black. (Diebel, 2006)

Euler angles are used to define, and correct for, the attitude of the sensor. The three angles relate to various combinations (depending on which Euler scheme is being applied) of the roll, pitch and yaw of an aircraft. Roll is the rotation about an axis running from nose to tail of the aircraft. Pitch is nose up or down about an axis running from wing to wing (picture climbing or descending). Yaw, nose left or right about an axis running up and down and in the plane defined by the aircraft body and its wings.

Tait-Bryan angles are another form of Euler angle. They are the convention normally used in aerospace applications, which the main difference between them and proper Euler angles being that the Tait-Bryan angles represent rotations about three distinct axes (i.e. x-y-z as opposed to z-x-z). As a consequence, in this case Roll, Pitch and Yaw will represent the three angles. There are also 6 conventions for this type of angle.

The attitude of the instrument as measured by the IMU is represented by Euler angles These angles are necessary to take the five independent magnetic tensor components from the body frame (aircraft) to the local frame (map). Schiffler et al. (2017) use the NED reference frame and give the equation as:

$$\boldsymbol{G}_{NED} = (\boldsymbol{D}^{b}_{NED})^{T} \boldsymbol{G}_{b} \boldsymbol{D}^{b}_{NED} \tag{3.26}$$

where $\boldsymbol{D}^{b}_{NED}$ is a rotation matrix which represents the attitude of the system during mobile operation, and takes into account roll, pitch and yaw, as defined by Euler angles. $\boldsymbol{G}_{NED}$ and $\boldsymbol{G}_{b}$ are the gradient components in the local frame and body frame respectively. If α, β, γ are the roll, pitch and yaw angles, then $\boldsymbol{D}^{b}_{NED}$ is defined as (Cai, Chen and Lee, 2011, p.33):

$$\boldsymbol{D}^{b}_{NED} = \begin{pmatrix} \cos\beta\cos\gamma & \cos\beta\sin\gamma & -\sin\beta \\ \sin\alpha\sin\beta\cos\gamma - \cos\alpha\sin\gamma & \sin\alpha\sin\beta\sin\gamma + \cos\alpha\cos\gamma & \sin\alpha\cos\beta \\ \cos\alpha\sin\beta\cos\gamma + \sin\alpha\sin\gamma & \cos\alpha\sin\beta\sin\gamma - \sin\alpha\cos\gamma & \cos\alpha\cos\beta \end{pmatrix} \tag{3.27}$$

3.4.5 Levelling and the Tensor Mean

The concept of a tensor mean is used in the levelling of airborne acquired tensor data and is achieved by means of a variation of a heading correction (FitzGerald et al., 2009). The calculation of the tensor mean must be discussed since it is not a simple averaging of tensor components. Rather, it must take into account both the amplitude and phase aspects of the tensor (FitzGerald et al., 2009) in order to preserve the intrinsic properties of the tensor (for example the Laplace condition). One way to do so is to convert the raw tensor to its amplitude and phase formulation first. The arithmetic mean can be used for the eigenvalues, but finding the mean of the eigenvector component requires the use of Fisher statistics (Fisher, 1953). To do so, the average rotational (eigenvector) part is determined by gathering the associated directional cosine terms. The following is the process followed:

1) Each rotation matrix is represented in terms of its associated direction cosines

2) Each of the direction cosine terms are squared and added, to give the square of the resultant vector. The resultant direction cosines are calculated from this.
3) The average inclination, declination and angular dispersion are then estimated from the resultant direction cosines.

Once the average tensor is calculated along flight lines and tie lines, this heading correction can be applied. Further levelling can be achieved using the standard loop closure technique (Green, 1983), minimising the Frobenius Norm of the delta misclosure tensor at each crossover point. The Frobenius Norm is a matrix norm defined as the square root of the sum of the absolute squares of its (in this case delta misclosure tensor) elements. FitzGerald et al. (2009) reported good results using this approach.

Noise on the tensor magnetometers, which includes so-called 'flux jumps' needs to be detected and removed during the processing (FitzGerald et al., 2009).

3.4.6 Gridding

The processing of data has also been well described by FitzGerald et al. (2009). The full tensor gradients are decomposed into a structural and a rotational part. Each Full Tensor Gradiometry (FTG) reading is decomposed using principal component analysis into the invariant eigenvalue amplitudes and orthogonal rotation matrix with associated eigenvectors local to the survey reference frame. This rotation represents the "phase" of the signal. FitzGerald et al. (2009) use quaternions (Hamilton, 1853) to handle rotations consistently.

Gridding of tensor data must be done in such a way as to ensure that the Laplace condition (see section 3.2.2) is preserved. This condition defines how tensor components derived from scalar potentials relate to each other. Conventional gridding would grid each tensor component separately, causing a possible breakdown of this condition. To maintain this condition, FitzGerald et al. (2009) and Fitzgerald and Holstein (2006) propose the use of spherical linear interpolation (SLERP) and quaternions as a basis for interpolation of the rotational component of the tensor. Conventional minimum curvature gridding can then safely be used on the amplitude component of the tensor. There is a patent granted in most jurisdictions in 2008 on this development. (Ref: Australian Patent Application No. 2006900346 in the name of Desmond FitzGerald & Associates Pty Ltd "An improved method of interpolation between a plurality of observed tensors").

3.4.7 Errors and Noise

Noise is a problem for any system design and can come from many sources. One source is from the instrument itself. A larger source of noise which needs to be corrected, is from external forms of noise such as the rotation of the sensor while in flight. As an overview, a number of SQUID sensors published in the literature, and their noise levels, are discussed.

Errors can be potentially estimated from the high sampling rate of the gradiometer magnetometers. If the assumption can be made that sources are over sampled, then there exist redundant measurements from which error estimates can be derived.

A high temperature SQUID tensor gradiometer developed by QinetiQ is aimed at magnetic anomaly detection (MAD) from a moving platform (Humphrey, Horton and Keene, 2005; Keene, Humphrey and Horton, 2005). Gradients are calculated by subtraction of the output of two magnetometers. They reported sensitivities of 80 $\mathrm{pT/m/Hz^{1/2}}$ at 1 Hz and 1 $\mathrm{pT/m/Hz^{1/2}}$ in the white noise region, while undergoing rotational motions (pitch, roll and yaw) of ±5º. They detected gradient anomalies of ≥ $1\ \mathrm{nT/m}$ with a 12 dB signal-to noise ratio and measurement bandwidth of 1 kHz.

Billings (2012) reports noise levels for the system developed through SERDP (Strategic Environmental Research and Development Program) of 2 $\mathrm{pT/m/Hz^{1/2}}$ at 10 Hz, under laboratory conditions.

The JeSSY STAR gradiometers have a noise level of $70\ \mathrm{fT/mHz^{1/2}}$, and the digitized signal has a noise level of $200\ \mathrm{fT/mHz^{1/2}}$ (Schiffler et al., 2017). These noise levels are also under laboratory conditions.

Schiffler et al. (2014) report that Euler angles generated by the IMU used with the JeSSY STAR system have inaccuracies in the order of 0.1 degrees. This in turn is responsible for motion noise in the order of 100 nT superimposed on the magnetic vector reference magnetometers. The noise is primarily due to roll, pitch and yaw.

Schiffler et al. (2017) later report an IMU system which had accuracy levels of $\pm\ 1°_{RMS}$ for roll and pitch, and $\pm\ 10°_{RMS}$ for yaw (RMS is short for root mean square, which is a measure of signal standard deviation). This shows the challenge in IMU systems maintaining high accuracy levels.

To demonstrate the impact that these accuracy levels have on the data, equation (3.27) is used and is applied to the following equation (Schiffler et al., 2014):

$$\boldsymbol{B}_{ref,b} = \boldsymbol{D}_{NED}^{b} \cdot \boldsymbol{B}_{ref} \tag{3.28}$$

Here, $\boldsymbol{D}_{NED}^{b}$ is equation (3.27), $\boldsymbol{B}_{ref}$ is the reference field vector and $\boldsymbol{B}_{ref,b}$ are the values rotated into the body frame. To investigate this, we can define various IMU resolutions between 0 and 2 degrees. This can be simulated by calculating a corresponding rotation of the field vector. The difference between the rotated field and the original field would be an estimate of instrument error due to rotation (since a resolution of 1 degree does not imply an error of 1 degree, only that the maximum error could be up to 1 degree). The error is therefore defined as $\mathrm{Error} = \mathrm{B_{ref,b}} - \mathrm{B_{ref}}$.

In this case, the field components over a body with susceptibility 0.1 SI were calculated. The magnetic inclination was -62 degrees, the magnetic declination was -16 degrees and the body was 20 metres below the surface of the earth. The ambient field was set to 28,000 nT. As a result, the magnetic field vector was calculated to be $\boldsymbol{B}_{ref} = (93.6,\ 264.8,\ -921.1)$. Roll, pitch and yaw values are varied between 0 degrees and 2 degrees to see the effect on the data. The results are shown in Figure 27.

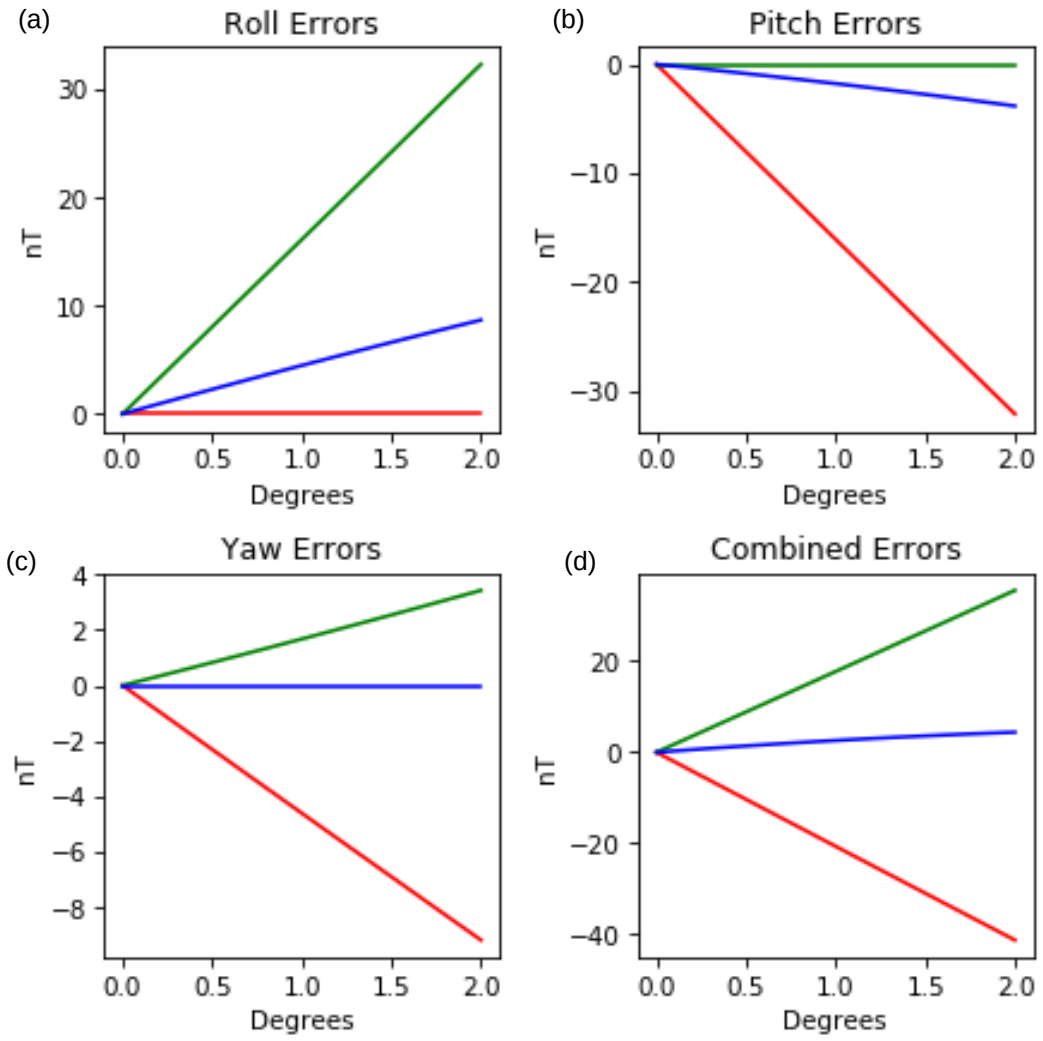

Figure 27 Estimated errors due to changes in roll, pitch and yaw, with B_x, B_y, B_z errors show in red, green and blue respectively. (a) shows the change in field when only roll is varied. (b) shows the change when only pitch is varied. (c) shows the changed when yaw is varied. (d) is the combination of all the changes.

As can be seen, errors can be as high as 40 nT for a roll, pitch and yaw equalling 2 degrees. It must be noted as well that this simulation is for a susceptibility of 0.1 SI. The errors will scale linearly with increases and decreases in susceptibility. Therefore, over a highly magnetic terrain, it is possible to have errors an order of magnitude or higher.

Noise of this form is especially devastating if not corrected appropriately. It creates severe corrugations on the data, making the data very difficult to interpret. Schiffler et al. (2017) have gone further in trying to deal with this, where they propose using Hilbert-like transforms on raw measurement data in order to obtain high quality gradient tensor and field vector quantities.

3.4.8 Grid denoising

Grid de-noising of tensor components can be obtained by making use of a smoothing convolution kernel based on the work by Pajot et al. (2008). The process takes advantage of third order tensor relationships such as $\partial y B_{xx} = \partial x B_{xy}$ in a finite-difference sense, and using this to calculate tensor filters based on minimizing tensor residuals in a least square sense. FitzGerald et al. (2009) term this process MITRE (Minimising Tensor Residual Errors).

The tensor noise residual can be measured in a grid with the following function (Fitzgerald and Paterson, 2013):

$$Noise = \left[\left(\frac{\partial B_{xx}}{\partial y} - \frac{\partial B_{xy}}{\partial x}\right)^2 + \left(\frac{\partial B_{xy}}{\partial y} - \frac{\partial B_{yy}}{\partial x}\right)^2 + \left(\frac{\partial B_{xz}}{\partial y} - \frac{\partial B_{yz}}{\partial x}\right)^2\right] 4\Delta x \Delta y \tag{3.29}$$

Where $\Delta x, \Delta y$ are grid spacing intervals. In central finite differences, this becomes:

$$Noise = \left[(B_{xx}(i, j+1) - B_{xx}(i, j-1) - B_{xy}(i+1, j) - B_{xx}(i-1, j)\right)^2 + \cdots] \tag{3.30}$$

This can be applied to a smoothing kernel, and FitzGerald et al. (2009) illustrate the smoothing process with the equation below:

$$\begin{aligned} B_{xx_{smooth}}[3][3] = \frac{1}{16}\big(&10B_{xx}[3][3] + 3(B_{xx}[3][1] + B_{xx}[3][5]) \\ &+ 2\big(B_{xy}[2][4] - B_{xy}[2][2] + B_{xy}[4][2] \\ &- B_{xy}[4][4]\big) + 2B_{yy}[3][3] - B_{yy}[5][3] \\ &- B_{yy}[1][3]\big) \end{aligned} \tag{3.31}$$

Equation (3.31) is illustrative of a 5x5 smoothing kernel used to calculate $B_{xx_{smooth}}[3][3]$, with B values being measured values. Values in square brackets denote the relative position within the kernel. Therefore, [3][3] denotes the centre of the kernel.

3.5 Tensor Reduction to the Pole

Reduction to the Pole (RTP) is generally done to create a standard anomaly response set for magnetic measurements. This has the defining characteristic that the RTP anomalies delineate bodies well, making for easier interpretation.

Fitzgerald et al. (2009) point out that magnetic tensor data should be reduced to the pole in order to obtain the best spatial registration for known geological bodies. They mention that a candidate means of doing RTP on tensor data is available but is largely untested.

Because tensor data obeys Laplace's equation, it is a true potential field and allows for continuation and RTP (Schmidt and Clark, 2006). Heath (2007) calculated RTP directly on tensor components using the standard FFT equation. The process involves transforming the field data to the Fourier domain and multiplying this new dataset by the term $(\theta_m \theta_f)^{-1}$, where:

$$\theta_m = \hat{M}_z + i\frac{\hat{M}_x k_x + \hat{M}_y k_y}{k} \quad \text{and } \theta_f = \hat{B}_z + i\frac{\hat{B}_x k_x + \hat{B}_y k_y}{k} \tag{3.32}$$

In this case, $(\hat{M}_x, \hat{M}_y, \hat{M}_z)$ is a unit vector in the direction of the magnetisation and $(\hat{B}_x, \hat{B}_y, \hat{B}_z)$ is a unit vector in the direction of the ambient magnetic field, i denotes an imaginary number, k is the wavenumber and (k_x, k_y) are spatial frequencies in the horizontal direction with $k^2 = k_x^2 + {k_y}^2$.

RTP cannot be done at low latitudes due to the RTP operator becoming unbounded along the direction of the magnetic declination. This amplifies noise in this direction resulting in linear features aligned with the declination dominating the RTP field (Li and Oldenburg, 2001). However, Clark (2012) points out that gradient tensor measurements have an advantage over TMI measurements at low latitudes. This is because TMI measurements are insensitive to vertical and easterly components of the anomalous field.

On a related note, the ZZ component of the magnetic tensor corresponds to the ZZ component of the RTP version of the magnetic tensor (Munschy and Fleury, 2011; Clark, 2013). Note that with negative inclinations, the ZZ component needs to be multiplied by negative one for this equivalency to a pole in the northern hemisphere to occur. Similar to standard RTP, this relationship breaks down the closer the field is to the equator. Figure 28 demonstrates this.

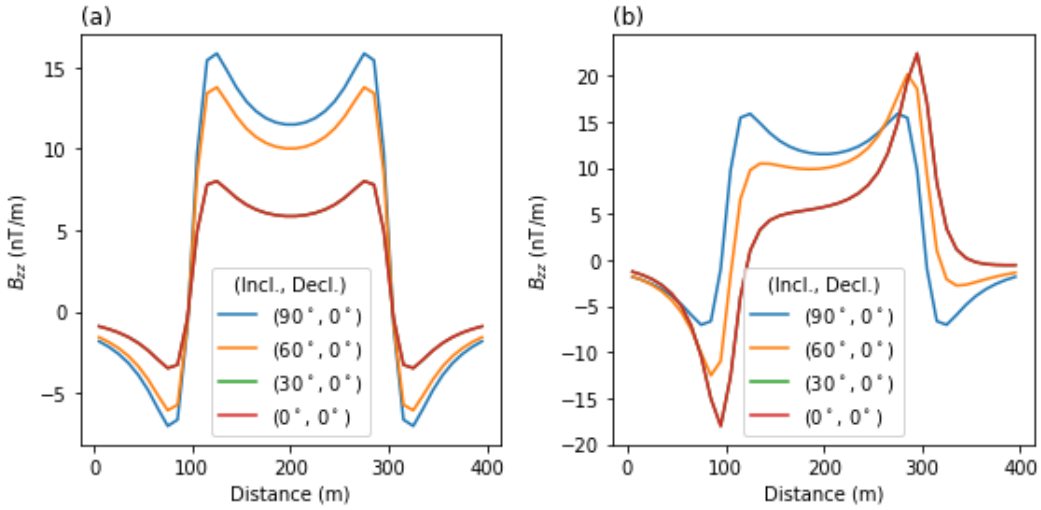

Figure 28 Effect of inclination and declination on an anomaly. A rectangular prism was modelled, with susceptibility 0.1 SI and ambient field of 28,000 nT. Inclinations and declinations are indicated on the graphs. (a) shows the profile intersecting the body perpendicular to the declination, and west to east. (b) shows the same anomaly, but now in the direction of the declination, and therefore north to south Notice the lack of symmetry, indicating that from this direction, B_{zz} is no longer a good approximation for RTP.

Tensor reduction to the pole is potentially a large topic by itself and this section just touches on some aspects of it. Modelling in this thesis does not make use of it, relying on the ambient field instead.

3.6 Derivation of Tensor Magnetic Field Equations

Equations for use in tensors have been derived by other authors and the methodology is well understood. Talwani (1965) and Plouff (1976) both make use of volume integrals which are essentially the gravity tensor equations (3.37) to (3.42) below. Heath (2007) also derived and studied tensor equations. Holstein (2002) found simple relations, in tensor form, for the gravity and magnetic anomaly solutions for a uniform polyhedron. Holstein (2003) derived gravity potential, field and field gradient tensor formulas for a polyhedral target.

Assume both the model and the observation point reside in a Cartesian coordinate system. In keeping with geographic conventions, the x-axis is defined to be west to east, the y-axis to be south to north and the z-axis to be upwards positive (ENU). This was done in order to easily incorporate terrain data, which follows an ENU convention. However, to be most compatible with the NED convention for potential fields, the polarity of the resulting fields will be the same as for NED or END. This allows flexibility in coordinates while remaining comparable to existing standards.

In the case of a rectangular prism, the source of the potential field extends from x_1 to x_2, y_1 to y_2, z_1 to z_2 and thus all equations have the following form:

$$\begin{aligned} B \;&=\; B(x,y,z)\big|_{x1}^{x2}\big|_{y1}^{y2}\big|_{z1}^{z2} \\ &= B(x_2,y_2,z_2) - B(x_1,y_2,z_2) - B(x_2,y_1,z_2) + B(x_1,y_1,z_2) \\ &\qquad - B(x_2,y_2,z_1) + B(x_1,y_2,z_1) + B(x_2,y_1,z_1) \\ &\qquad - B(x_1,y_1,z_1) \end{aligned} \tag{3.33}$$

To simplify the notation, only the $B(x,y,z)$ term is given below for B (magnetic field) and g (gravity field), with the entirety of (3.33) being implied.

Since magnetic relations can be derived through Poisson's relationship with the gravity equations, the gravity equations derived by Talwani (1965) and Plouff (1976) are presented first. Note equation derivations are normally presented in NED convention, but in this case all derivations have been corrected for standard coordinate systems for 3D models in the ENU convention– i.e. x positive eastwards, y positive northwards and z negative downwards. This is for convenience in modelling and to give an alternate derivation (although the differences truly are minor).

$$g_x = G\rho\left[-x\tan^{-1}\left(\frac{yz}{xr}\right) - y\log(r-z) - z\log(r-y)\right] \tag{3.34}$$

$$g_y = G\rho\left[y\tan^{-1}\left(\frac{xz}{yr}\right) + x\log(r-z) - z\log(x+r)\right] \quad (3.35)$$

$$g_z = G\rho\left[z*\tan^{-1}\left(\frac{yx}{zr}\right) + x\log(r-y) - y\log(x+r)\right] \quad (3.36)$$

$$g_{xx} = G\rho\left[-\tan^{-1}\left(\frac{yz}{xr}\right)\right] \quad (3.37)$$

$$g_{yy} = G\rho\left[-\tan^{-1}\left(\frac{xz}{yr}\right)\right] \quad (3.38)$$

$$g_{zz} = G\rho\left[-\tan^{-1}\left(\frac{yx}{zr}\right)\right] \quad (3.39)$$

$$g_{xy} = G\rho[\log(r-z)] \quad (3.40)$$

$$g_{yz} = G\rho[\log(x+r)] \quad (3.41)$$

$$g_{xz} = G\rho[\log(r-y)] \quad (3.42)$$

where ρ is the density, and $G = 6.67 * 10^{-11} Nm^2 kg^{-2}$ is the gravitational constant. The distance between the observation point and the source is described by r.These volume equations or rank two gravity tensor components are then used in Poisson's relation to calculate corresponding magnetic components. Blakely (1995, pp. 91-93) defines it as follows:

$$\begin{aligned} B_m &= -\frac{C_m}{G\rho}\boldsymbol{M}\cdot\nabla\boldsymbol{U} \\ &= -\frac{1}{\mu_0 G\rho}\,\boldsymbol{M}\cdot\nabla\boldsymbol{U} \end{aligned} \quad (3.43)$$

where $\boldsymbol{U}$ is the gravitational potential, $\boldsymbol{M}$ is the uniform magnetisation, $C_m = 1/\mu_0$, $G = 6.67 * 10^{-11}\, Nm^2/kg^2$, ρ is density in kg/m^3 and $\mu_0 = 4\pi * 10^{-7}\ Henry/m$. Equation (3.43) can then be written as:

$$B_m = -\frac{kH}{\mu_0 G\rho}\left(\alpha\frac{\partial}{\partial x} + \beta\frac{\partial}{\partial y} + \gamma\frac{\partial}{\partial z}\right) g_m$$
$$= -\frac{kH}{\mu_0 G\rho}\left[\alpha g_{mx} + \beta g_{my} + \gamma g_{mz}\right] \quad (3.44)$$

where α, β and γ are the direction cosines of the induced magnetic field, k is susceptibility, H is the magnetic field, measured in A/m and g_m is the component of gravity in the direction of magnetisation.

Substituting equations (3.37) to (3.42) into (3.44):

$$B_x = \frac{kH}{\mu_0}\left[-\alpha\tan^{-1}\left(\frac{yz}{xr}\right) + \beta\log(r-z) + \gamma\log(r-y)\right] \quad (3.45)$$

$$B_y = \frac{kH}{\mu_0}\left[\alpha\log(r-z) - \beta\tan^{-1}\left(\frac{xz}{yr}\right) + \gamma\log(x+r)\right] \quad (3.46)$$

$$B_z = \frac{kH}{\mu_0}\left[\alpha\log(r-y) + \gamma\log(x+r) - \gamma\tan^{-1}\left(\frac{yx}{zr}\right)\right] \quad (3.47)$$

The final tensor equations are obtained by taking derivatives, with respect to x, y and z, of (3.45) to (3.47). These are:

$$B_{xx} = \frac{kH}{\mu_0 r}\left[\alpha\frac{yz(r^2 + x^2)}{r^2x^2 + y^2z^2} + \beta\frac{x}{r - z} + \gamma\frac{x}{r-y}\right] \quad (3.48)$$

$$B_{yy} = \frac{kH}{\mu_0 r}\left[-\alpha\frac{y}{r-z} - \gamma\frac{y}{r + x} - \beta\frac{xz(r^2 + y^2)}{r^2y^2 + x^2z^2}\right] \quad (3.49)$$

$$B_{zz} = \frac{kH}{\mu_0 r}\left[-\alpha\frac{z}{r-y} - \beta\frac{z}{r + x} - \gamma\frac{xy(r^2 + z^2)}{r^2z^2 + x^2y^2}\right] \quad (3.50)$$

$$B_{xy} = \frac{kH}{\mu_0 r}\left[\alpha\frac{xz}{x^2 + y^2} - \beta\frac{y}{r - z} + \gamma\right] \quad (3.51)$$

$$B_{yz} = \frac{kH}{\mu_0 r}\left[\alpha + \beta\frac{xy}{y^2 + z^2} - \gamma\frac{z}{r + x}\right] \quad (3.52)$$

$$B_{xz} = \frac{kH}{\mu_0 r}\left[\alpha\frac{xy}{x^2 + z^2} + \beta - \gamma\frac{z}{r - y}\right] \quad (3.53)$$

Synthetic tensor data, calculated for a rectangular prism, is shown in Figure 29 and Figure 30. The magnetic field intensity is 28,000 nT, susceptibility is 0.1 SI, inclination is 45 degrees, declination is 30 degrees. The rectangular prism has a width of 200 m and height of 280 meters, situated between 20 meters and 300 meters. In this case, B_{tmi} is the total magnetic intensity.

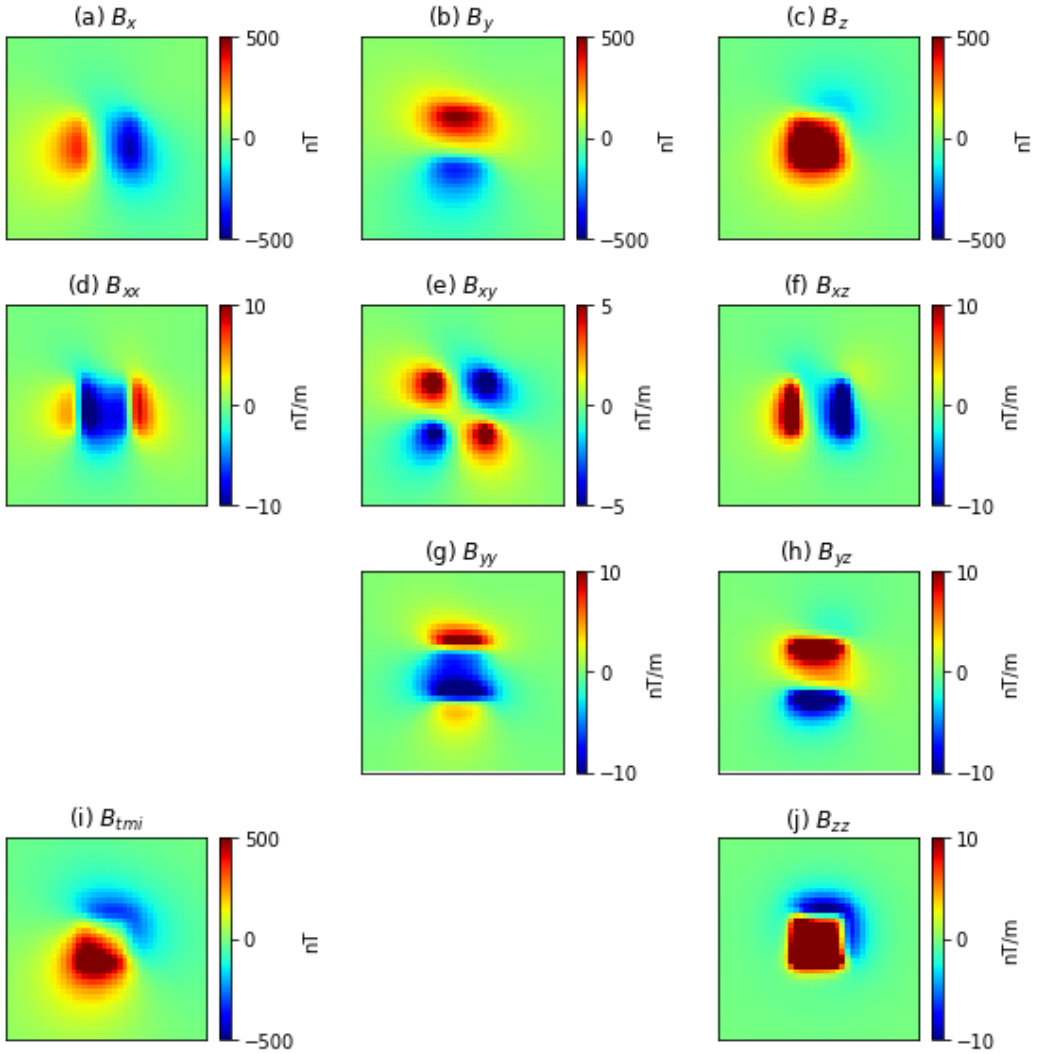

Figure 29 Synthetic magnetic tensor calculations for a rectangular prism at inclination 45 degrees, declination 30 degrees. (a) to (c) are the primary components of the field. (d) to (h), (j) are the tensor components. (i) is the total magnetic intensity.

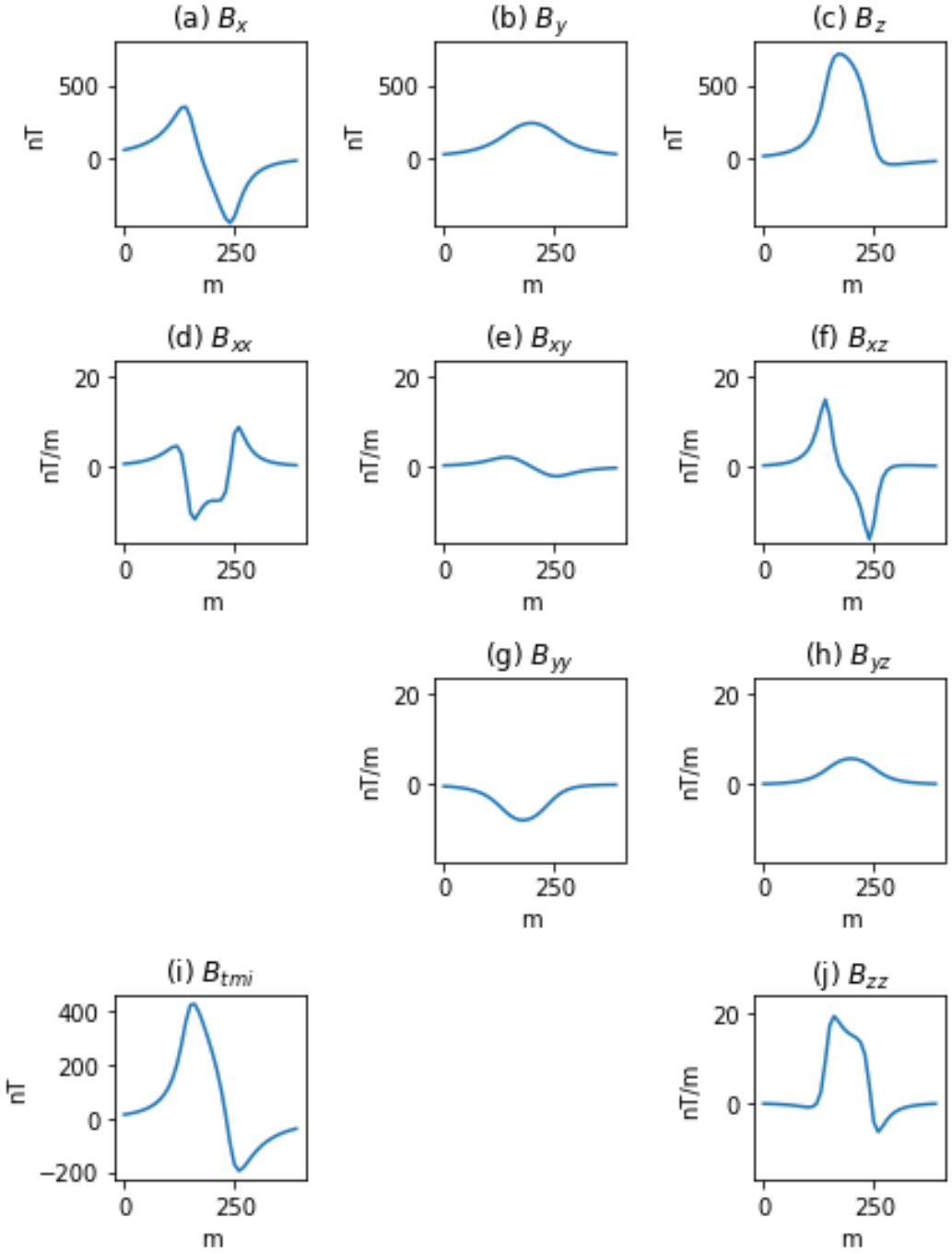

Figure 30 Cross section profiles going west-east through the centre of the rectangular prism at inclination 45 degrees, declination 30 degrees. (a) to (c) are the primary components of the field. (d) to (h), (j) are the tensor components. (i) is the total magnetic intensity.

Another synthetic tensor data set, calculated for the same rectangular prism at the pole, is shown in Figure 31 and Figure 32. The magnetic field intensity is 28,000 nT, susceptibility is 0.1 SI, inclination is 90 degrees, declination is 0 degrees.

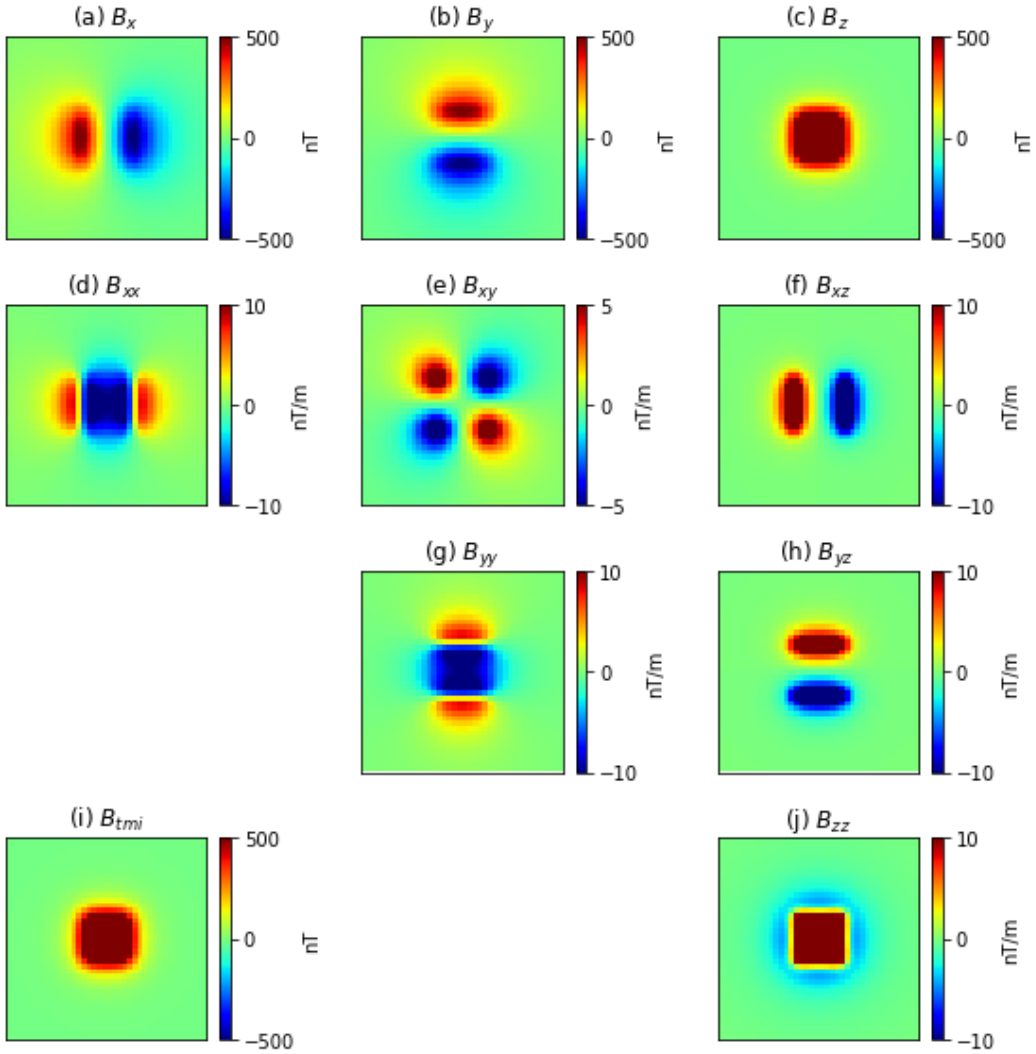

Figure 31 Synthetic magnetic tensor calculations for a rectangular prism at inclination 90 degrees, declination 0 degrees. (a) to (c) are the primary components of the field. (d) to (h), (j) are the tensor components. (i) is the total magnetic intensity.

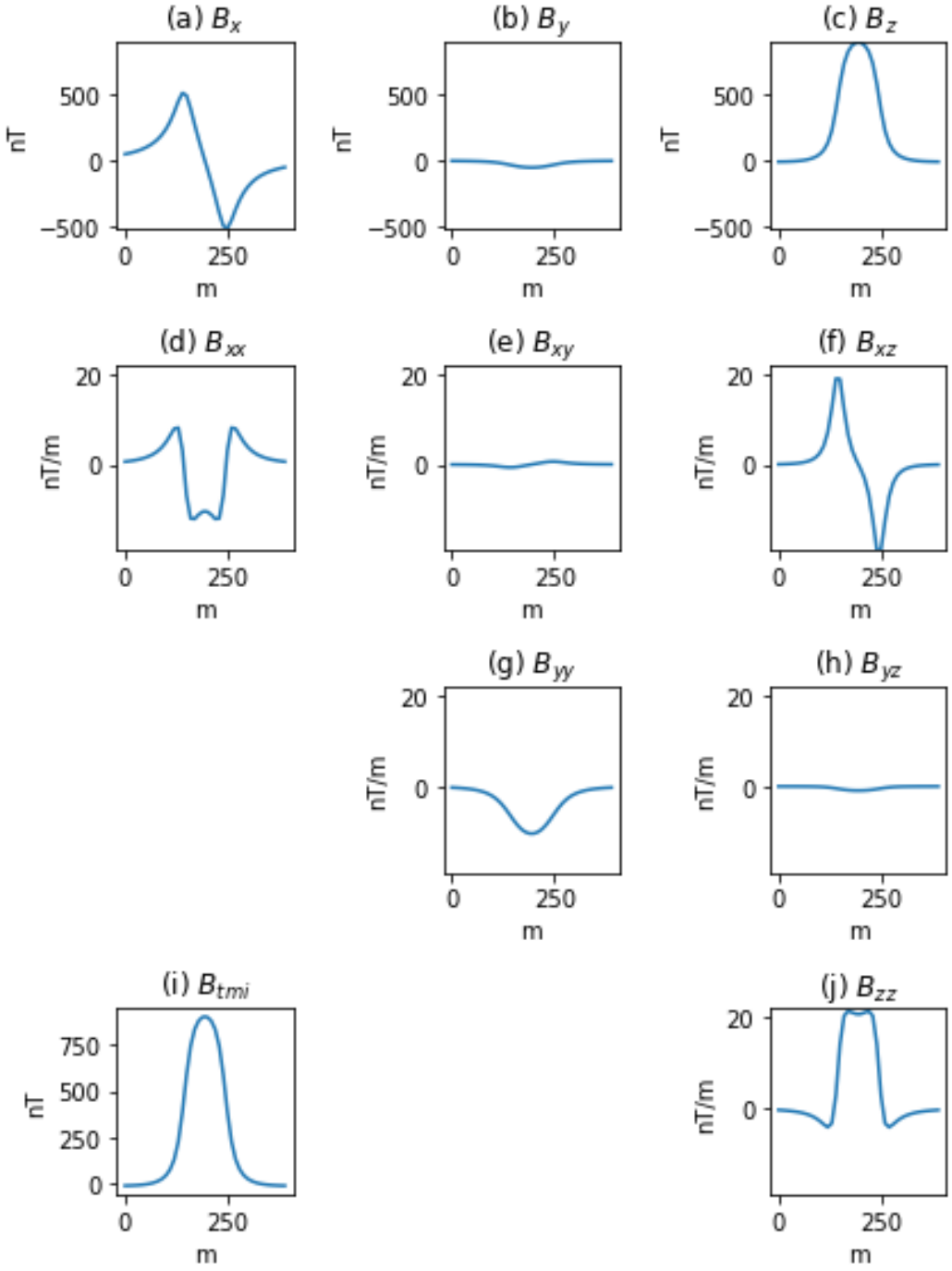

Figure 32 Cross section profiles through the centre of the rectangular prism at inclination 90 degrees, declination 0 degrees. (a) to (c) are the primary components of the field. (d) to (h), (j) are the tensor components. (i) is the total magnetic intensity.

The symmetry of the TMI anomaly is reflected throughout the new tensor components, and is especially obvious when comparing B_{xx} and B_{zz} in Figure 30 to the versions at the pole in Figure 32.

3.6.1 Verifying the equations

Equation verification is an important process when deriving equations. The tensor equations can be verified by understanding the relationships between these equations. The Poisson relation can be used to compare datasets between gravity and magnetic equations and the total magnetic intensity equation can be used to confirm the component data with regularly calculated data.

The full total magnetic intensity is defined as:

$$B_{tmi} = \sqrt{(B_x + \alpha B)^2 + \left(B_y + \beta B\right)^2 + (B_z + \gamma B)^2} - B \tag{3.54}$$

where B is the ambient magnetic field, and α, β, γ are the direction cosines. The process is as follows:

1) Check equations (3.45), (3.46), (3.47) by comparing them with results with alternate calculations for B_x, B_y, B_z (for example, as demonstrated by Guptasarma and Singh, 1999). Figure 33 to Figure 36 show the results of this.
2) Check equations (3.45), (3.46), (3.47) by substituting them into equation (3.50). The results of this can be compared using results calculated with equation (2.26).
3) Check equations (3.48) to (3.53) by calculating the gradients of (3.45), (3.46), (3.47) either in the space domain or the frequency domain. Figure 37 to Figure 41 show the results of this. Minor discrepancies will occur because of the inaccuracies inherent in gradient calculations.

One possible source for error is the calculation of results over nodes or edges of the rectangular prism (i.e. the line connecting two nodes). In such instances, the tensor equations will generate division by zero errors. This problem is especially bad when the observation is also on the surface of the rectangular prism. However, it is easily dealt with. By ensuring that values are calculated away from the edges of the rectangular prism, this error is prevented, even if the observation is on the surface of the rectangular prism. The simplest way to achieve this is to calculate one value per rectangular prism, with the x and y coordinate of that value being in the centre of the rectangular prism. The z coordinate will either be 0 (resting on the top of the rectangular prism) or at the observation height above the rectangular prism. Naturally for this strategy to work, the number of rectangular prisms must equal the number of observations (assuming equal spaced observations, which is the case for a grid). Otherwise there will be too few values calculated to be useful when comparing to observed values.

All results confirm the validity of the tensor equations and the accuracy of the calculations. In the case of point 3 above, it is never necessary to calculate a vertical derivative. This is because of tensor symmetry. Therefore $\frac{dB_x}{dz} = \frac{dB_z}{dx}$, $\frac{dB_y}{dz} = \frac{dB_z}{dy}$ and $\frac{dB_z}{dz} = -\frac{dB_x}{dx} - \frac{dB_y}{dy}$. This simplifies the tests by confining them to horizontal derivatives.

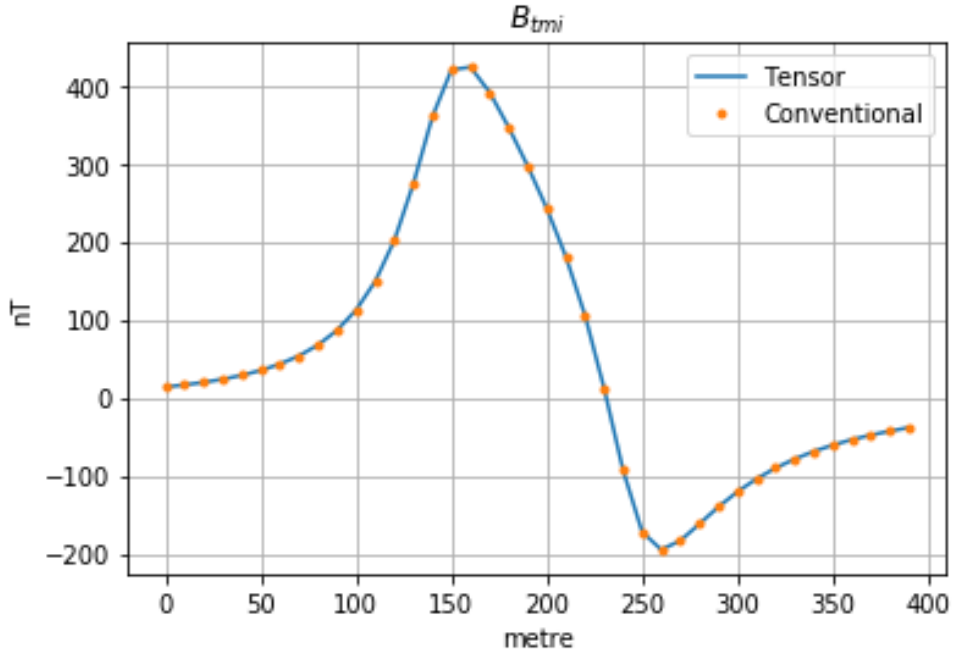

Figure 33 Comparison between tensor and conventional calculations for B_{tmi}. The source is a rectangular prism with dimension 100x100x280 cubic metres, 20 metres below the surface. The ambient field is 28,000 nT, susceptibility is 0.1 SI, Inclination is 45 degrees and declination is 30 degrees.

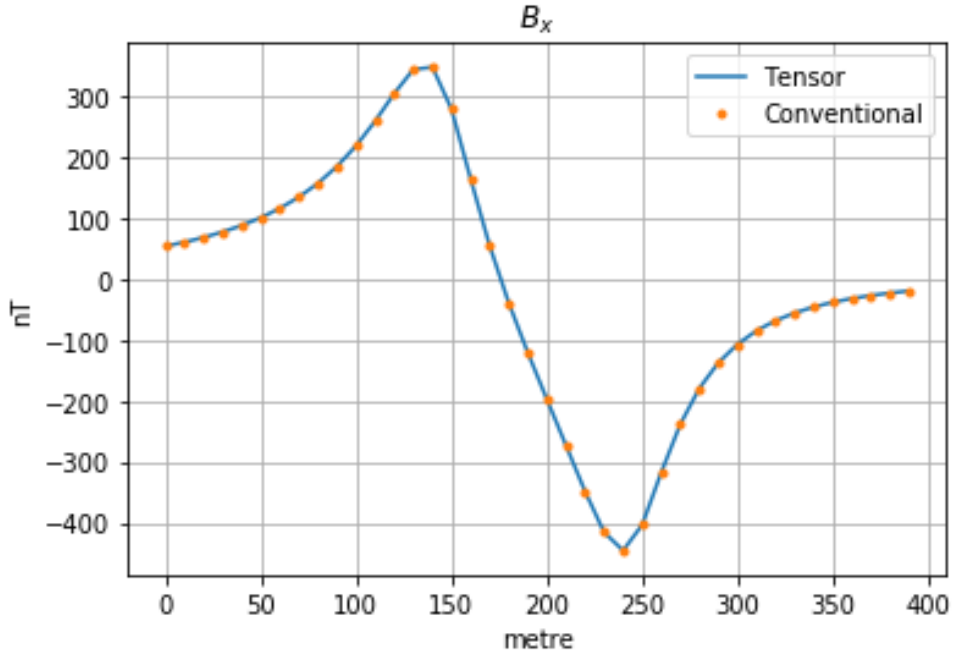

Figure 34 Comparison between tensor and conventional calculations for B_x. The source is a rectangular prism with dimension 100x100x280 cubic metres, 20 metres below the surface. The ambient field is 28,000 nT, susceptibility is 0.1 SI, Inclination is 45 degrees and declination is 30 degrees.

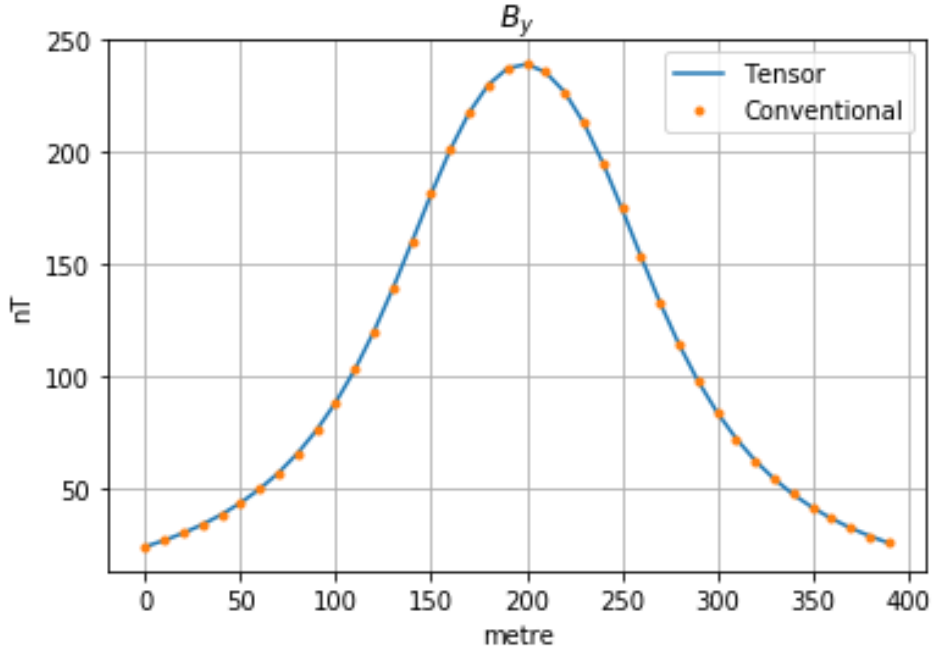

Figure 35 Comparison between tensor and conventional calculations for B_y. The source is a rectangular prism with dimension 100x100x280 cubic metres, 20 metres below the surface. The ambient field is 28,000 nT, susceptibility is 0.1 SI, Inclination is 45 degrees and declination is 30 degrees.

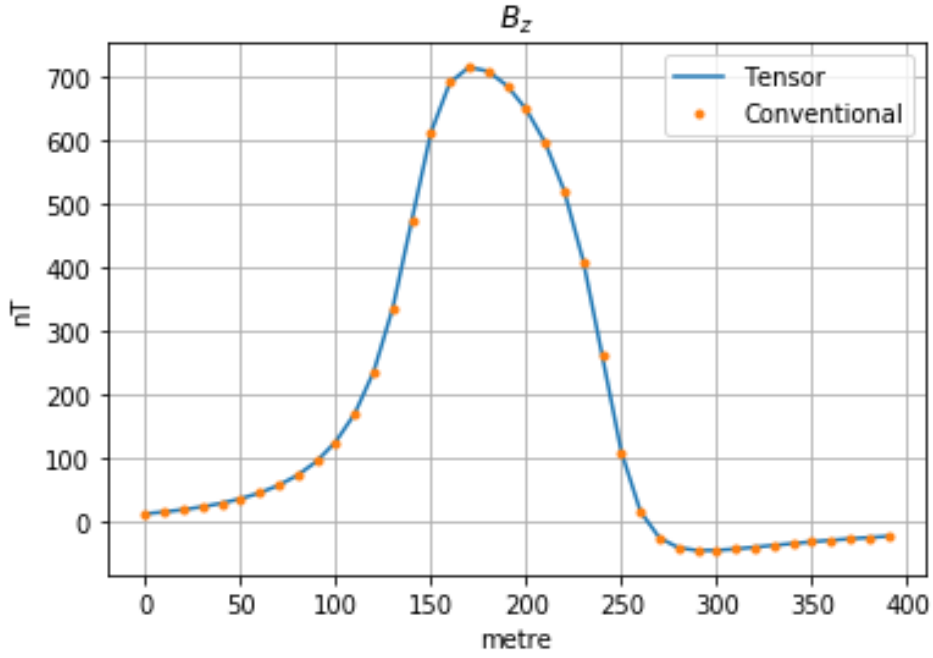

Figure 36 Comparison between tensor and conventional calculations for B_z. The source is a rectangular prism with dimension 100x100x280 cubic metres, 20 metres below the surface. The ambient field is 28,000 nT, susceptibility is 0.1 SI, Inclination is 45 degrees and declination is 30 degrees.

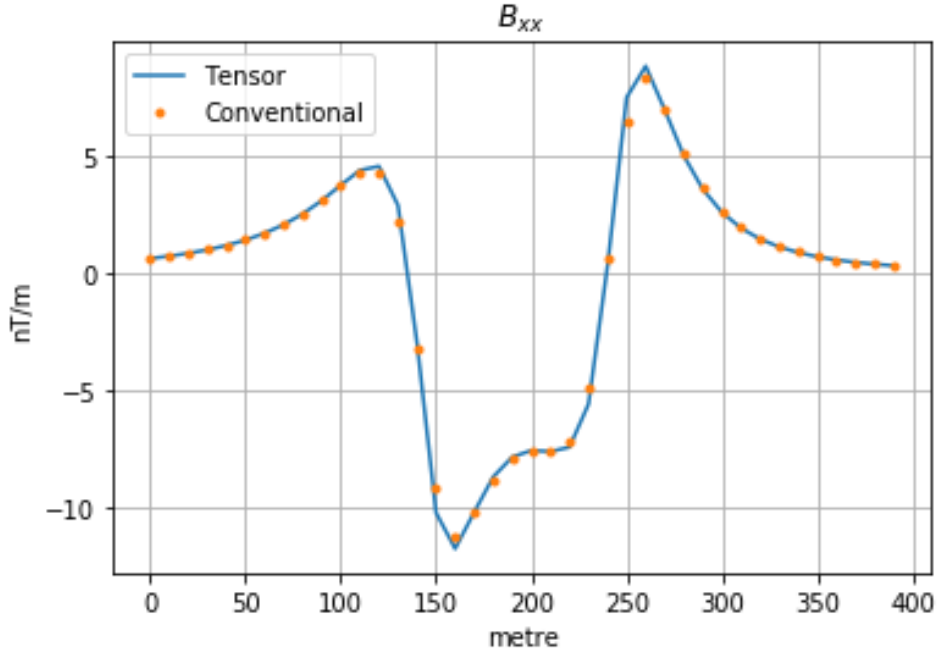

Figure 37 Comparison between tensor and conventional calculations for B_{xx}. The source is a rectangular prism with dimension 100x100x280 cubic metres, 20 metres below the surface. The ambient field is 28,000 nT, susceptibility is 0.1 SI, Inclination is 45 degrees and declination is 30 degrees.

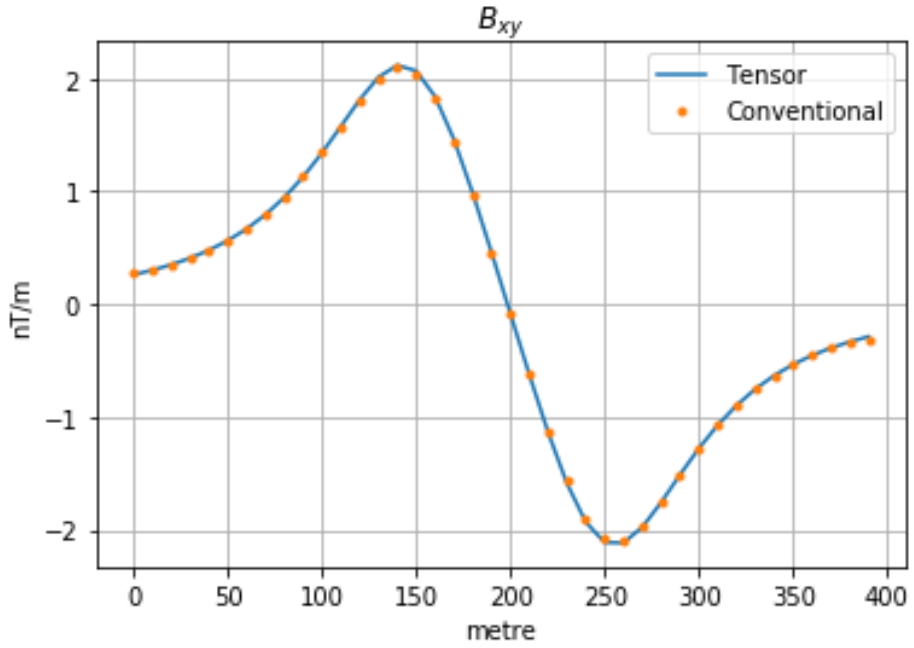

Figure 38 Comparison between tensor and conventional calculations for B_{xy}. The source is a rectangular prism with dimension 100x100x280 cubic metres, 20 metres below the surface. The ambient field is 28,000 nT, susceptibility is 0.1 SI, Inclination is 45 degrees and declination is 30 degrees.

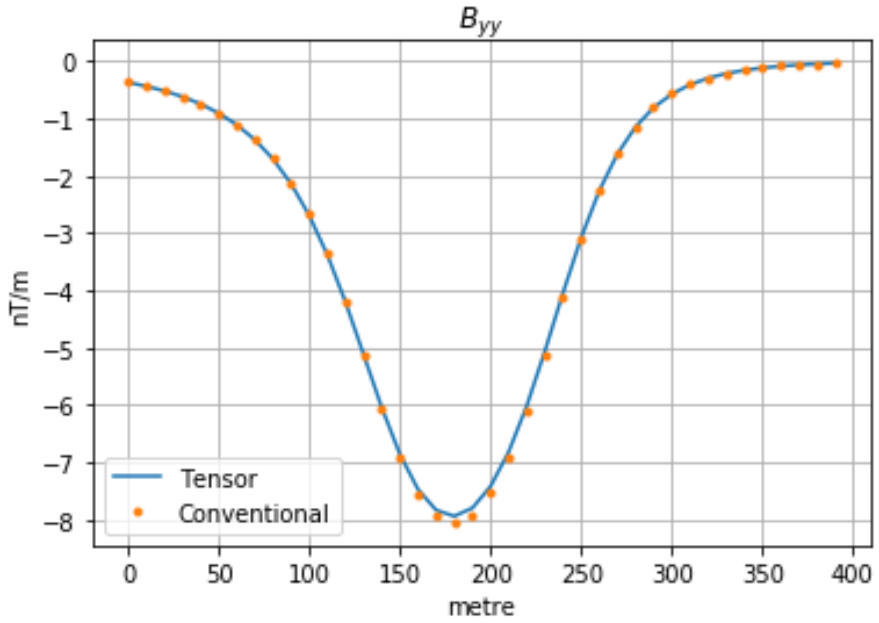

Figure 39 Comparison between tensor and conventional calculations for B_{yy}. The source is a rectangular prism with dimension 100x100x280 cubic metres, 20 metres below the surface. The ambient field is 28,000 nT, susceptibility is 0.1 SI, Inclination is 45 degrees and declination is 30 degrees.

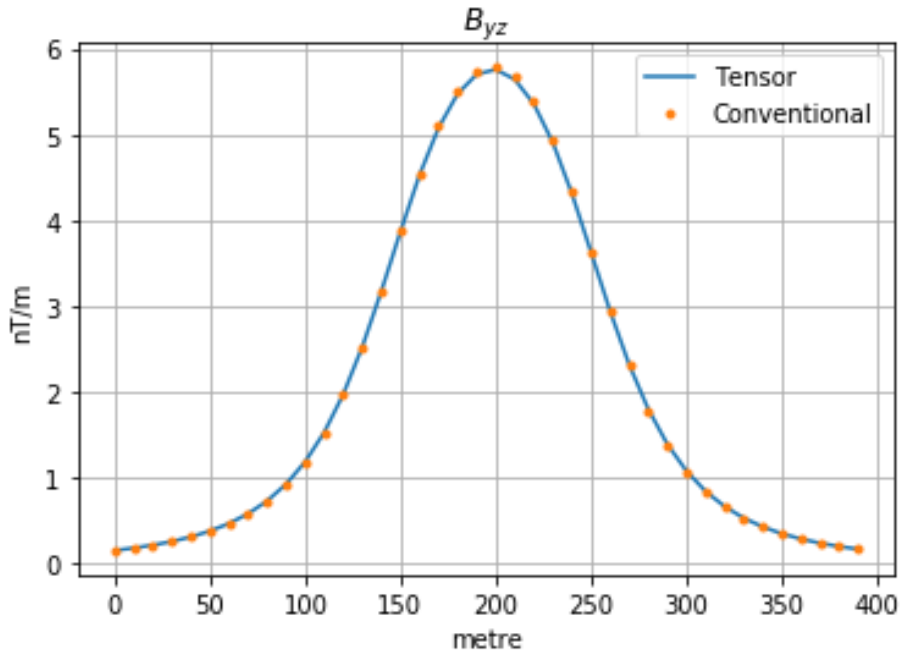

Figure 40 Comparison between tensor and conventional calculations for B_{yz}. The source is a rectangular prism with dimension 100x100x280 cubic metres, 20 metres below the surface. The ambient field is 28,000 nT, susceptibility is 0.1 SI, Inclination is 45 degrees and declination is 30 degrees.

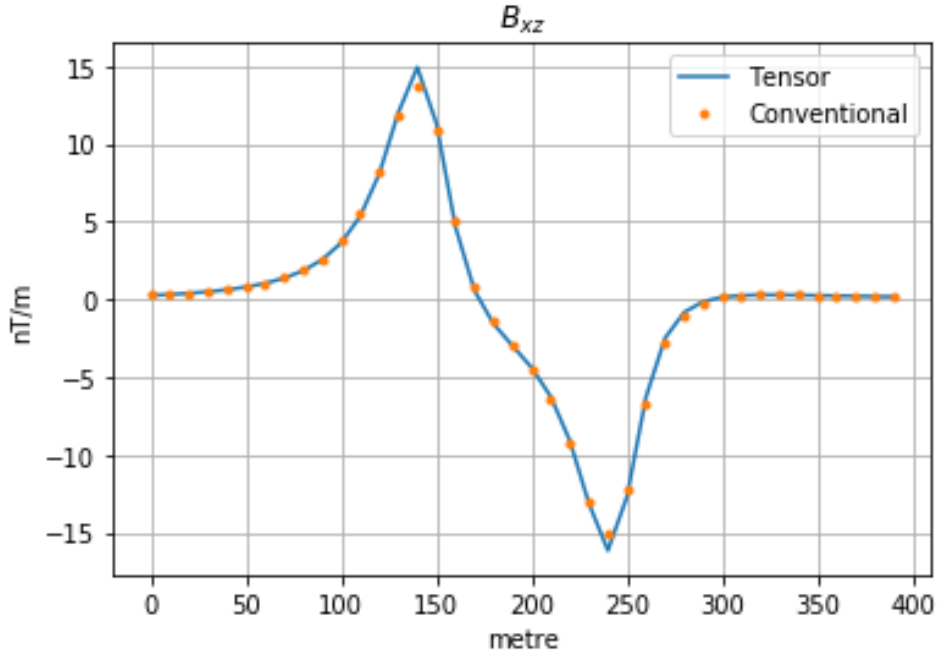

Figure 41 Comparison between tensor and conventional calculations for B_{xz}. The source is a rectangular prism with dimension 100x100x280 cubic metres, 20 metres below the surface. The ambient field is 28,000 nT, susceptibility is 0.1 SI, Inclination is 45 degrees and declination is 30 degrees.

3.7 Tensor Interpretation overview

The interpretation of tensor components has more potential than conventional total magnetic field interpretation, due to the variety of information which is presented through the tensor measurements. It can, of course, be modelled using forward modelling and inversion. Schmidt and Clark (2006) give an overview of the characteristics of the tensor gradient components and derived quantities.

Table 1 Interpretation of tensor components and invariants, from Schmidt and Clark (2006)

Quantity	Interpretation	Comments
B_{xx}	E-W boundary delineation	symmetric anomaly for vertical magnetization, antisymmetric for horizontal magnetization
B_{yy}	N-S boundary delineation	symmetric anomaly for vertical magnetization, antisymmetric for horizontal magnetization
B_{xy}	Body corner delineation	anomaly signs depend on magnetization direction
B_{zz}	Delineates steep boundaries preferentially	symmetric anomaly for vertical magnetization; antisymmetric for horizontal magnetization
B_{xz}	E-W boundary delineation	antisymmetric anomaly for vertical magnetization; symmetric for N-S horizontal magnetization
B_{yz}	N-S boundary delineation	antisymmetric anomaly for vertical magnetization; symmetric for E-W horizontal magnetization
I_1	Resolves source boundaries	better resolving power than analytic signal
I_2	Preferentially resolves shallower features of complex sources.	Due to faster falloff with distance.

I_1 and I_2 are invariants as described in equations (3.10) and (3.11) in section 3.2.3. The normalised source strength (NSS) (Clark, 2012; Beiki et al., 2012) shown in equation (3.13) is another tensor invariant which demonstrates the potential of tensor interpretation. It is proportional to the magnitude of the dipole moment, while being independent of the magnetisation direction. The 2D version of this is equal to the total gradient, or the analytic signal amplitude of either the vertical field component B_z or the strike perpendicular horizontal component B_x. The 2D version is also independent of magnetization direction. Clark (2014) made use of the normalised source strength in conjunction with Helbig analysis (a method to estimate the vector components of the total magnetisation) to determine locations of sources, their depths and magnitudes of magnetic moments.

Both the NSS and tensor components have been used to determine source location through Euler deconvolution (Beiki et al., 2012; Schmidt et al., 2004). Tensor and NSS versions of Euler deconvolution give better estimates than standard Euler deconvolution when anomalies from causative bodies are distorted by horizontally neighbouring sources. Since the tensor Euler deconvolution is applied to all the measured tensor components, it effectively takes advantage of the curvature information inherent in the measurement, thereby improving source detection. If the conventional Euler equation for TMI data is:

$$x\frac{\partial B_{tmi}}{\partial x} + y\frac{\partial B_{tmi}}{\partial y} + z\frac{\partial B_{tmi}}{\partial z} = -nB_{tmi} \tag{3.55}$$

where B_{tmi} is the anomalous field, then the tensor version is expressed as:

$$\begin{bmatrix} B_{xx} & B_{xy} & B_{xz} \\ B_{yx} & B_{yy} & B_{yz} \\ B_{zx} & B_{zy} & B_{zz} \end{bmatrix} \begin{bmatrix} x - x_0 \\ y - y_0 \\ z - z_0 \end{bmatrix} = -n \begin{bmatrix} B_x \\ B_y \\ B_z \end{bmatrix} \tag{3.56}$$

Where x_0, y_0, z_0 is an arbitrary origin and n is the structural index or attenuation rate, which varies typically between 1 and 4 depending on the source type (but can be higher).

3.8 Application to voxel forward modelling

The tensor equations are directly applicable to the voxel based modelling developed in section 2.3.2. To demonstrate the effectiveness of this, a simple model of a dyke and a step has been created. The magnetic field is 28,000 nT, with inclination -62 degrees and declination -16 degrees. The height of observation is 0 metres, and the susceptibility is 0.1 (SI units).

Figure 42 and Figure 43 show 2D and 3D views of the synthetic model, while Figure 44 and Figure 45 show the responses of the profile shown in Figure 42, both for the conventional magnetic components and the tensor components. The 2D profile shown runs from west to east. Since both the dyke and the step are north-south in orientation, it is expected that the largest anomalies with come in the x direction (for ENU convention) and z direction or combinations thereof. This is indeed the case, with the dominant anomalies being B_x, B_z, B_{xx}, B_{xz} in Figure 44 and Figure 45. The anomaly shape is sharper over the dyke than over the step, as is to be expected. The amplitude is also smaller, which is expected since they share the same susceptibility.

The test therefore shows that the voxel based modelling can be successfully applied to tensors.

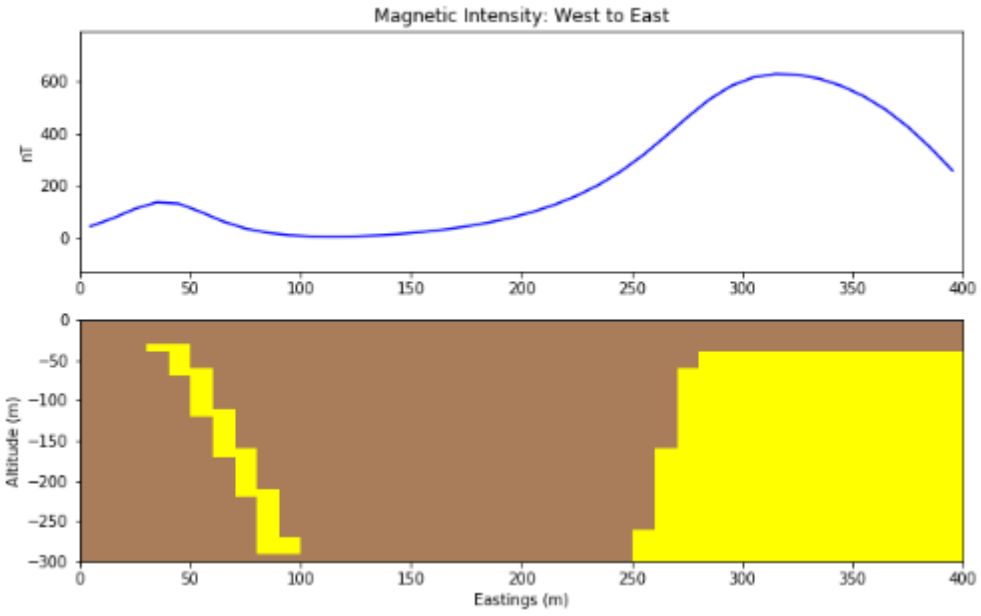

Figure 42 Profile of B_{tmi} of demonstration model, running from west to east.

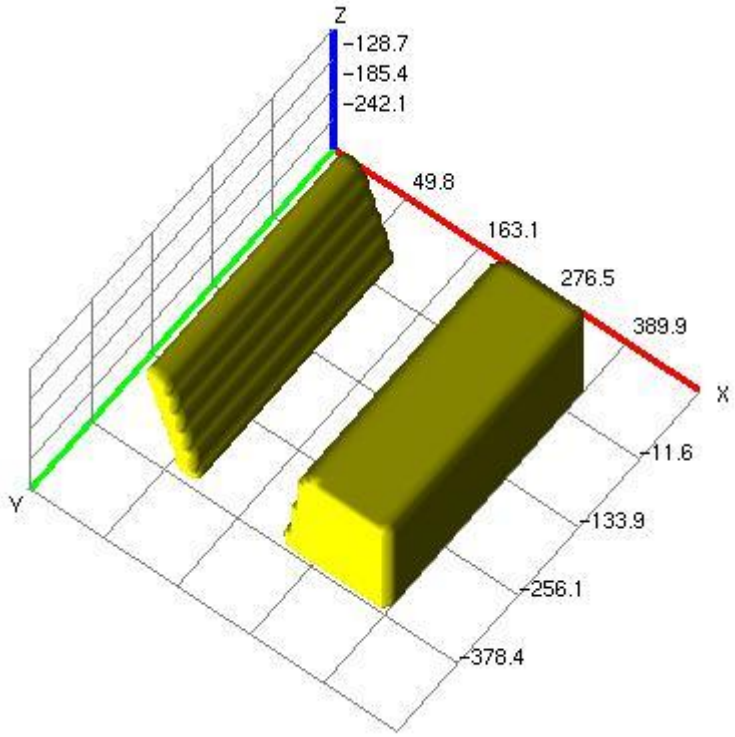

Figure 43 3D view of demonstration model. All coordinates are in metres

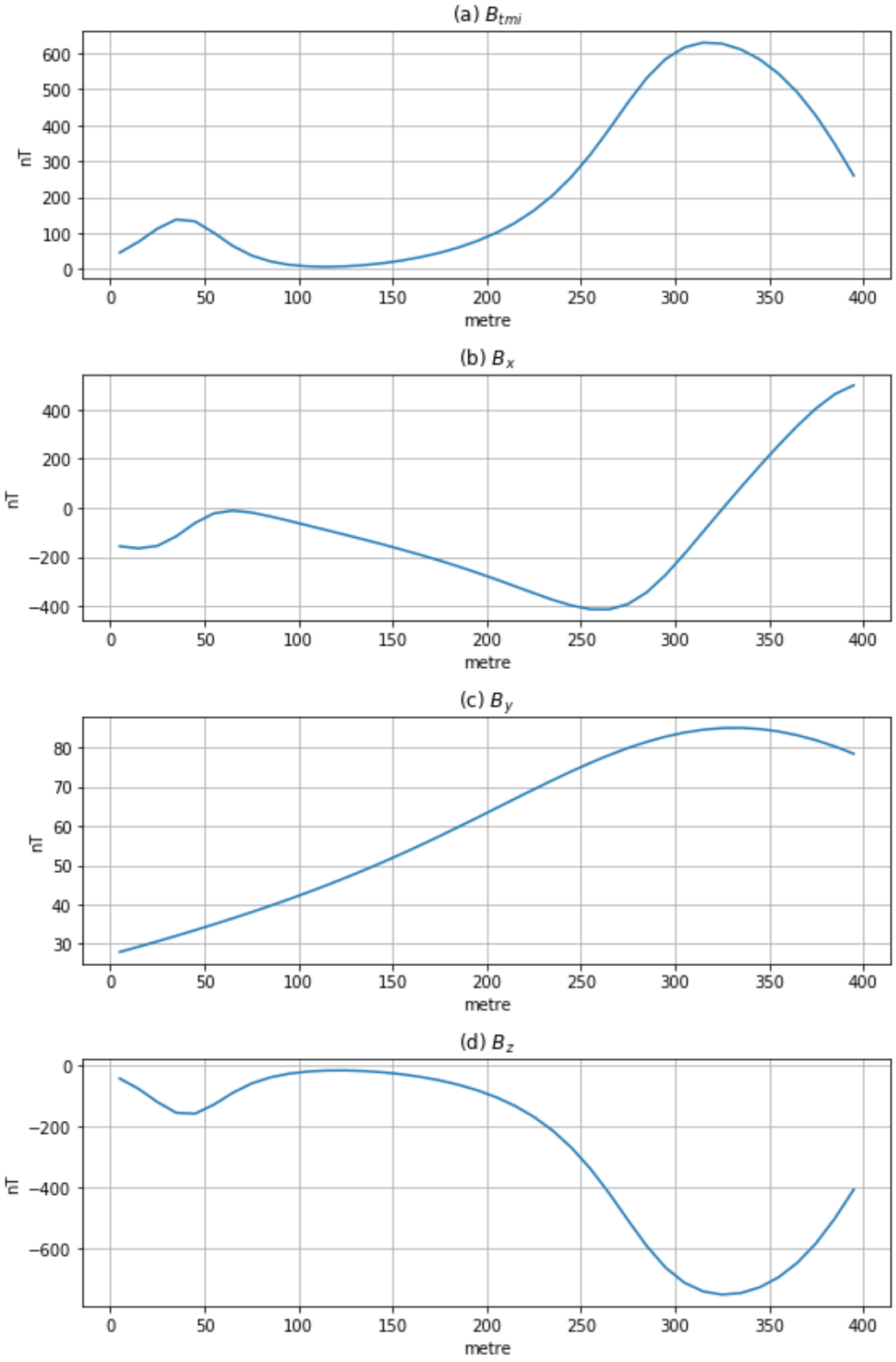

Figure 44 Field Component results for model. (a) to (d) are B_{tmi}, B_x, B_y and B_z respectively.

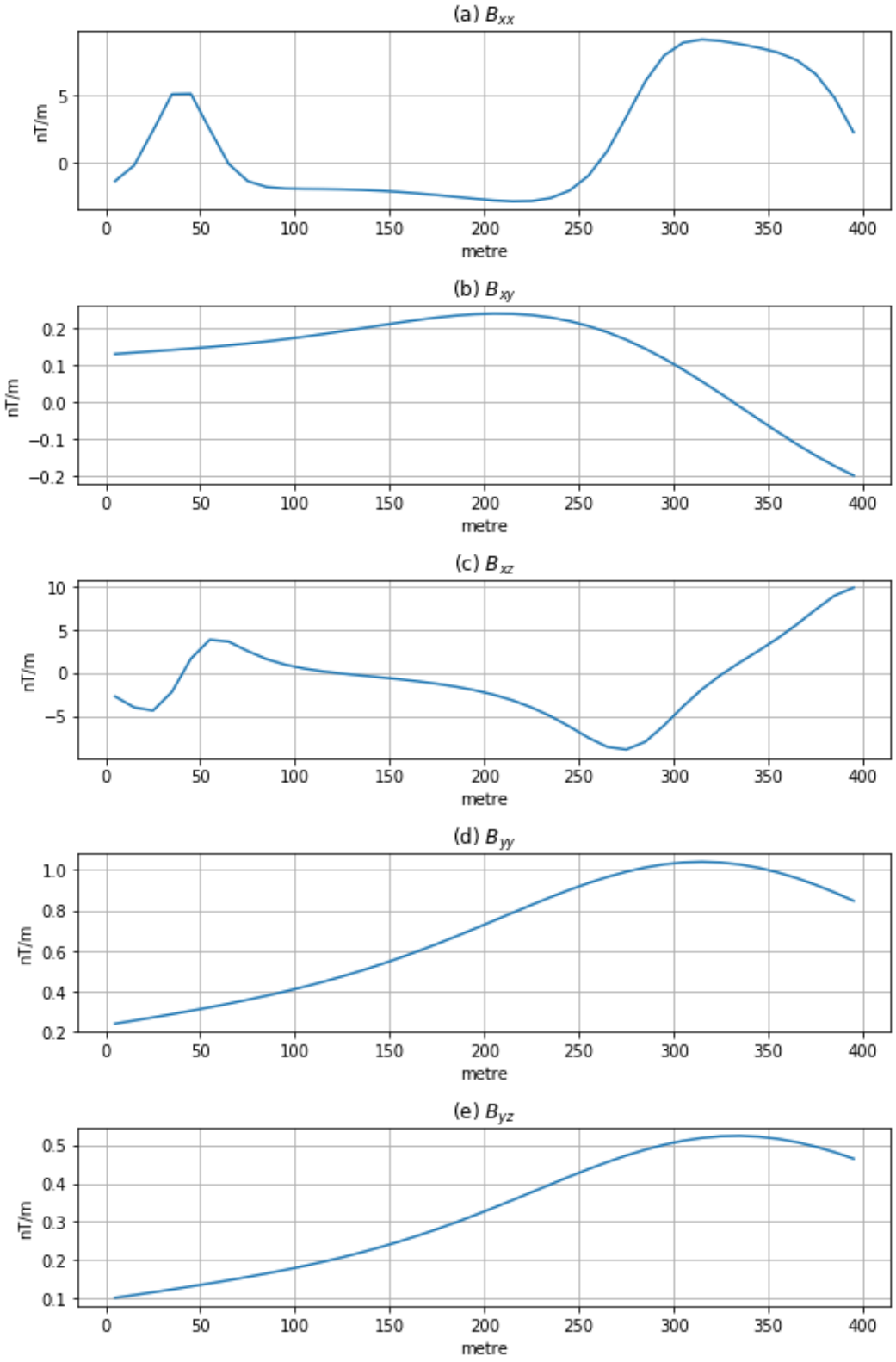

Figure 45 Tensor component results for model.

CHAPTER 4 SOURCE DISTANCE CALCULATIONS

4.1 Introduction

Calculating the distance to a buried source of a magnetic or gravity anomaly has been examined throughout the last century. Of particular interest is the work of Nabighian (1972), who realised that potential fields were analytic in nature and could be described in terms of a so called analytic signal amplitude, and Thompson (1982) who used Euler deconvolution to solve source distances for a variety of sources. Hsu, Coppens and Shyu (1998) calculated depths to magnetic sources using the analytic signal. More recently (Ma and Du, 2012; Cooper, 2014a, 2014b, 2015; Cooper and Whitehead, 2016) utilized various orders of the analytic signal to calculate source distance parameters. Although there has been work with tensor data (Zhang et al., 2000), the application of tensor data to improving source distance techniques has yet to be fully explored and is therefore an important focus of this thesis. The direct application of tensor data to source distance techniques is not the only objective, but also the development of analogous versions of source distance equations using tensor components only.

Although gravity and magnetic data is commonly measured as a single value, gradiometer (or tensor) data is becoming more common, especially in airborne gravity surveying. This implies more data is available for modelling.

Tensor datasets are a possible way of overcoming the modelling ambiguity challenges. Although not as prevalent as the measurement of scalar magnetic and gravity fields, the measurement of gradiometer data means that for each location nine gradient magnetic or gravity values are recorded, instead of simply one overall magnitude of the relevant field. This influx of data presents opportunities for optimizing modelling of data and forms part of the basis for the new work in this field described in this section. Beiki (2010) used analytic signals of the gravity tensor to estimate source location. Cevallos (2014) has used curvatures derived from airborne gravity gradient data to produce 3D models. Fitzgerald and Holstein (2016) used gravity gradiometry inversion to optimise the surface mapping of elongated geological features. In the case of magnetic tensors, the normalised source strength and its vector gradient has been used to determine source locations for compact sources, thin sheets, contacts and other models such as vertical pipes (Clark, 2012, 2013; Beiki et al., 2012). Schmidt et al., (2004) have used tensor Euler deconvolution applied to the SQUID based GETMAG system.

This section will examine source distance techniques relating to the analytic signal and the application of tensor data to this. The original versions of these techniques used TMI

(Total Magnetic Intensity) data and the analytic signal from TMI data only, but new work developed includes the derivation and testing of analogous versions of these source distance methods using tensor components and tensor versions of the analytic signal. This has been published in Cole and Cooper (2018).

At this point it is worth discussing homogeneity versus heterogeneity with respect to modelling. Homogeneity assumes that a model, or body, is uniform in composition with respect to a characteristic of that body, such as susceptibility. In reality, this is not an accurate assumption to make since the earth is seldom homogenous. Heterogeneity implies non-uniformity in this characteristic. Traditional forward modelling is an example of homogeneity since each body being modelled has constant physical properties. This is not always the case, and Holstein (2003) derived gravity and magnetic formulas for media where, for example, density and magnetisation are varying linearly. Inversion applied to voxels (as described in the next section) is also an example of heterogeneity since each voxel can have a slightly different value for the characteristic it describes.

Source distance techniques are interesting in this respect in that for a single value, homogeneity is assumed (i.e. if we are calculating the distance to a dyke, the dyke is assumed to have constant susceptibility). However, when applied over observations covering an area, the variety of depths and susceptibilities determined by the source distance technique (as will be seen in CHAPTER 6) shows a more heterogeneous set of solutions and hints at perhaps a hybrid between homogenous techniques and heterogeneous techniques.

4.2 Inverse Modelling

As mentioned in section 2.3.8, inverse modelling can be thought of as the inverse of the forward problem, in that instead of postulating model parameters to calculate data, data is used to calculate the model parameters directly. A solution to an inverse problem is obtained by estimating some model, and testing predicted data against observed data using misfit and mathematical acceptability criteria. If these two criteria are unacceptable, the model is adjusted until the two criteria are satisfied. Predicted data can be calculated using forward modelling applied to the model. Upon final generation of the model, it must still be examined to see whether it makes geological sense. If not, the method for generating the model may have to be changed. There are several approaches to inversion, including stochastic (for example, Monte Carlo, Genetic Algorithms) deterministic (for example steepest descent, conjugate gradients) and analytical (Bilayer in DC resistivity). Many of the different types of inversion applicable to magnetic data can be found in Nabighian et al. (2005)

In the case of potential fields, the solutions are non-unique. This means that a single measured anomaly can be described by a shallow, broad anomaly, as well as a deeper compact anomaly (Johnson and van Klinken, 1979). This implies that some strategy is required to improve the calculation of the model parameters.

The complexity of the inversion process means that there are two challenges to overcome:

1) The non-uniqueness of the solution
2) The complexity of the model means that calculations can be prohibitively time consuming.

There are many different approaches to inversion. Polygonal inversion (where the modelling body is defined by a series of polygons or polygonal facets) inverts the nodes of the polygons to adjust for the location of the body, as well the body parameters (such as susceptibility or density). The one constraint is that the body is generally regarded as homogenous.

Voxel based inversion does not seek to define a body via nodes, but rather in terms of the distribution of some physical property (such as density or susceptibility). The resulting solutions are therefore heterogeneous, since each voxel can have a different solution. The manner in which the inversion is optimised is defined by some inversion strategy. For example Barbosa (1994) describes a means to model gravity data using body compactness as the key model for the inversion.

The non-uniqueness of the solution is dealt with in a number of different ways. Firstly, it can be restricted based on prior information. This can take the form of geological information (knowing which areas may host magnetic rocks based on geology), geophysical information (restrictions to ranges of physical properties such as density and susceptibility) and can also include some logical hypothesis on the nature of the model (i.e. compactness as mentioned above). The University of British Columbia (UBC) codes (MAG3D, 2017) use compactness as well as depth weighting for their inversion of magnetic data. Depth weighting is a method used to counteract the natural decay in magnetic data (Li and Oldenburg, 1996). Without this, most solutions will be concentrated close to the observations. Accurate knowledge of the topography is also vital to obtaining meaningful solutions.

Adjusting model constraints can be an effective method to overcome point 1. However, for point 2, the time consuming nature of inversion means that in three dimensions, iteratively correcting mistakes is not always a realistic solution. To illustrate this, assume we have a grid at the surface of the earth of 100 by 100 observations. If our model has 100 layers, without some suitable optimisation strategy, the forward modelling alone would imply that the 1 000 000 voxels equates to 10 000 000 000 calculations. One way to deal with this is by increasing the layer thickness with increasing depth, thereby reducing the number of layers necessary.

Source detection techniques fall into a sub category of inverse techniques known as depth to source estimation techniques (Nabighian et al., 2005). This category includes Werner deconvolution, the Naudy method, Euler deconvolution and the analytic signal, to name a few. It seeks to overcome the non-uniqueness and time efficiency challenges by simplifying the problem and avoiding the need to iteratively change a model in order to generate an optimal solution. This is generally achieved by limiting the number of model geometries, or sources, and by targeting simpler parameters, such as depth to source instead of the full geometrical description of the model.

4.3 Total Magnetic Intensity Derivative Calculations

The total magnetic intensity and its derivatives are used in source distance and tensor calculations. Since there is more than one version of calculating the total magnetic intensity and its derivatives, both are covered here.

4.3.1 Approximate Total Magnetic Intensity

As mentioned in section 2.2.6, the approximate total magnetic intensity is defined as:

$$B_{tmi} = \alpha \cdot B_x + \beta \cdot B_y + \gamma \cdot B_z \tag{4.1}$$

where B_x, B_y and B_z are defined to be three components of a magnetic field $\boldsymbol{B}$, and α, β, γ are defined to be the direction cosines relating to the direction of the magnetic field.

The tensor for a magnetic field is given by:

$$\boldsymbol{B} = \begin{bmatrix} \frac{\partial B_x}{\partial x} & \frac{\partial B_x}{\partial y} & \frac{\partial B_x}{\partial z} \\ \frac{\partial B_y}{\partial x} & \frac{\partial B_y}{\partial y} & \frac{\partial B_y}{\partial z} \\ \frac{\partial B_z}{\partial x} & \frac{\partial B_z}{\partial y} & \frac{\partial B_z}{\partial z} \end{bmatrix} = \begin{bmatrix} B_{xx} & B_{xy} & B_{xz} \\ B_{xy} & B_{yy} & B_{yz} \\ B_{xz} & B_{yz} & B_{zz} \end{bmatrix} \tag{4.2}$$

In tensor notation therefore, the x, y and z derivatives of B_{tmi} are:

$$\frac{\partial B_{tmi}}{\partial x} = \alpha \cdot B_{xx} + \beta \cdot B_{yx} + \gamma \cdot B_{zx} \tag{4.3}$$

$$\frac{\partial B_{tmi}}{\partial y} = \alpha \cdot B_{xy} + \beta \cdot B_{yy} + \gamma \cdot B_{zy} \tag{4.4}$$

$$\frac{\partial B_{tmi}}{\partial z} = \alpha \cdot B_{xz} + \beta \cdot B_{yz} + \gamma \cdot B_{zz} \tag{4.5}$$

The advantage of these expressions is not only convenience, but also the fact that they do not require knowledge of either the total magnetic intensity, or the three components of the magnetic field.

Second derivatives can be easily defined as follows:

$$\frac{\partial^2 B_{tmi}}{\partial x^2} = \alpha \cdot \frac{\partial B_{xx}}{\partial x} + \beta \cdot \frac{\partial B_{yx}}{\partial x} + \gamma \cdot \frac{\partial B_{zx}}{\partial x} \tag{4.6}$$

$$\frac{\partial^2 B_{tmi}}{\partial y \partial x} = \alpha \cdot \frac{\partial B_{xy}}{\partial x} + \beta \cdot \frac{\partial B_{yy}}{\partial x} + \gamma \cdot \frac{\partial B_{zy}}{\partial x} \tag{4.7}$$

$$\frac{\partial^2 B_{tmi}}{\partial z \partial x} = \alpha \cdot \frac{\partial B_{xz}}{\partial x} + \beta \cdot \frac{\partial B_{yz}}{\partial x} + \gamma \cdot \frac{\partial B_{zz}}{\partial x} \tag{4.8}$$

$$\frac{\partial^2 B_{tmi}}{\partial y^2} = \alpha \cdot \frac{\partial B_{xy}}{\partial y} + \beta \cdot \frac{\partial B_{yy}}{\partial y} + \gamma \cdot \frac{\partial B_{zy}}{\partial y} \tag{4.9}$$

$$\frac{\partial^2 B_{tmi}}{\partial z^2} = \alpha \cdot \frac{\partial B_{xz}}{\partial z} + \beta \cdot \frac{\partial B_{yz}}{\partial z} + \gamma \cdot \frac{\partial B_{zz}}{\partial z} \tag{4.10}$$

$$\frac{\partial^2 B_{tmi}}{\partial z \partial y} = \alpha \cdot \frac{\partial B_{xz}}{\partial y} + \beta \cdot \frac{\partial B_{yz}}{\partial y} + \gamma \cdot \frac{\partial B_{zz}}{\partial y} \tag{4.11}$$

And remembering that $\partial^2 B_{tmi} / \partial z \partial y = \partial^2 B_{tmi} / \partial y \partial z$, $\partial^2 B_{tmi} / \partial y \partial x = \partial^2 B_{tmi} / \partial x \partial y$ and $\partial^2 B_{tmi} / \partial z \partial x = \partial^2 B_{tmi} / \partial x \partial z$.

4.3.2 Total Magnetic Intensity

As mentioned in section 2.2.6, when anomalies are too strong, then the approximation in equation (4.1) is no longer accurate and the full measured total field anomaly should be used. The full total magnetic intensity is defined as:

$$B_{tmi} = \sqrt{(B_x + \alpha B_a)^2 + \left(B_y + \beta B_a\right)^2 + (B_z + \gamma B_a)^2} - B_a \tag{4.12}$$

where B_a is the ambient magnetic field. The derivatives are:

$$\frac{\partial B_{tmi}}{\partial x} = \frac{B_{xx}(B_x + \alpha B_a) + B_{yx}\left(B_y + \beta B_a\right) + B_{zx}(B_z + \gamma B_a)}{B_{tmi} + B_a} \tag{4.13}$$

$$\frac{\partial B_{tmi}}{\partial y} = \frac{B_{xy}(B_x + \alpha B_a) + B_{yy}\left(B_y + \beta B_a\right) + B_{zy}(B_z + \gamma B_a)}{B_{tmi} + B_a} \tag{4.14}$$

$$\frac{\partial B_{tmi}}{\partial z} = \frac{B_{xz}(B_x + \alpha B_a) + B_{yz}\left(B_y + \beta B_a\right) + B_{zz}(B_z + \gamma B_a)}{B_{tmi} + B_a} \tag{4.15}$$

The second derivatives are expressed as:

$$\frac{\partial^2 B_{tmi}}{\partial x^2} = \frac{B_{xxx}D + B_{xyx}E + B_{zxx}F + B_{xx}^2 + B_{xy}^2 + B_{zx}^2}{B_{tmi} + B_a} - \frac{\left(B_{xx}D + B_{xy}E + B_{zx}F\right)^2}{(B_{tmi} + B_a)^3} \tag{4.16}$$

$$\frac{\partial^2 B_{tmi}}{\partial y \partial x} = \frac{B_{xxy}D + B_{xyy}E + B_{zxy}F + B_{xy}B_{xx} + B_{yy}B_{xy} + B_{zy}B_{zx}}{B_{tmi} + B_a} - \frac{\left(B_{xy}D + B_{yy}E + B_{zy}F\right)\left(B_{xx}D + B_{xy}E + B_{zx}F\right)}{(B_{tmi} + B_a)^3} \tag{4.17}$$

$$\frac{\partial^2 B_{tmi}}{\partial z \partial x} = \frac{B_{zxx}D + B_{xyz}E + B_{zzx}F + B_{zx}B_{xx} + B_{zy}B_{xy} + B_{zz}B_{zx}}{B_{tmi} + B_a} - \frac{\left(B_{zx}D + B_{zy}E + B_{zz}F\right)\left(B_{xx}D + B_{xy}E + B_{zx}F\right)}{(B_{tmi} + B_a)^3} \tag{4.18}$$

$$\frac{\partial^2 B_{tmi}}{\partial y^2} = \frac{B_{xyy}D + B_{yyy}E + B_{zyy}F + B_{xy}^2 + B_{yy}^2 + B_{zy}^2}{B_{tmi} + B_a} - \frac{\left(B_{xy}D + B_{yy}E + B_{zy}F\right)^2}{(B_{tmi} + B_a)^3} \tag{4.19}$$

$$\frac{\partial^2 B_{tmi}}{\partial z^2} = \frac{B_{zxz}D + B_{zzy}E + B_{zzz}F + B_{zx}^2 + B_{zy}^2 + B_{zz}^2}{B_{tmi} + B_a} - \frac{\left(B_{zx}D + B_{zy}E + B_{zz}F\right)^2}{(B_{tmi} + B_a)^3} \quad (4.20)$$

$$\frac{\partial^2 B_{tmi}}{\partial z \partial y} = \frac{B_{xyz}D + B_{zyy}E + B_{zzy}F + B_{zx}B_{xy} + B_{zy}B_{yy} + B_{zz}B_{zy}}{B_{tmi} + B_a} - \frac{\left(B_{zx}D + B_{zy}E + B_{zz}F\right)\left(B_{xy}D + B_{yy}E + B_{zy}F\right)}{(B_{tmi} + B_a)^3} \quad (4.21)$$

where $D = B_x + \alpha B_a$, $E = B_y + \beta B_a$ and $F = B_z + \gamma B_a$.

Since derivatives of the tensor components are calculated numerically, second x and y derivatives of B_{tmi} can be calculated by numerical differentiation of the results of equations (4.13) and (4.14), as there is no advantage to using derived equations. Therefore only equation (4.20) is necessary if avoiding calculating vertical derivatives (i.e. the z derivative).

4.3.3 Numerical calculation of rank 3 tensor components and higher order analytic signals

The numerical calculation of the rank 3 tensors used in equations (4.6) to (4.11) or the derivatives of equations (4.3) to (4.5) and (4.13) to (4.15) can be done using numerical differentiation. This can be achieved using, for example, central differences and the gradient function in a mathematical language such as MATLAB or Python. Central differences and finite difference schemes are well documented in literature (for example Numerical recipes by Press et al., 2007). A disadvantage of finite difference schemes is that they are an approximation to the exact analytical solution. They are affected by round-off error, which is the loss of precision due to computer rounding of decimal numbers. Since they are related to a truncated form of a Taylor series, a truncation error also exists (error due to approximating and infinite sum with a finite sum). As a simple example, we can define the forward Euler scheme as follows:

$$\frac{\partial u(x_j)}{\partial x} = \frac{u(x_j + \Delta x) - u(x_j)}{\Delta x} \quad (4.22)$$

where Δx is the grid spacing, x_i is the grid coordinate, and u is the grid data from which we want a derivative calculated. This scheme compares the current data point with the

next data point. The backward Euler scheme compares the current data point with the previous data point, and is defined as:

$$\frac{\partial u(x_j)}{\partial x} = \frac{u(x_j) - u(x_j - \Delta x)}{\Delta x} \tag{4.23}$$

And for the central differences, which take advantage of data points on both sides of the current data point, we have:

$$\frac{\partial u(x_j)}{\partial x} = \frac{u(x_j + \Delta x) - u(x_j - \Delta x)}{2\Delta x} \tag{4.24}$$

To understand the error, we need to look at the Taylor series. Using the above variables, we have:

$$u(x_j + \Delta x) = u(x_j) + \Delta x_j \left.\frac{\partial u}{\partial x}\right|_{x_j} + \frac{\Delta x_j^2}{2!}\left.\frac{\partial^2 u}{\partial x^2}\right|_{x_j} + \frac{\Delta x_j^3}{3!}\left.\frac{\partial^3 u}{\partial^3 x}\right|_{x_j} + \cdots \tag{4.25}$$

Rearranging this, we see the following:

$$\frac{u(x_j + \Delta x) - u(x_j)}{\Delta x_j} - \left.\frac{\partial u}{\partial x}\right|_{x_j} = \frac{\Delta x_j}{2!}\left.\frac{\partial^2 u}{\partial x^2}\right|_{x_j} + \frac{\Delta x_j^2}{3!}\left.\frac{\partial^3 u}{\partial^3 x}\right|_{x_j} + \cdots \tag{4.26}$$

The terms on the left side are clearly the forward Euler formula subtracting the desired answer, while the terms on the right are the truncation error terms. If u is sufficiently smooth (i.e. possesses higher order derivatives), the first truncation error term is used to define the order of magnitude of the error. Typically, forward and reverse Euler schemes have first order errors, while central difference schemes have second order errors. There do exist higher order variations of these finite difference schemes.

This error is quoted using Big O notation. A second order error is therefore quoted as $O(\Delta x^2)$. Note that it does not mean that the error is a big as Δx^2 since there are other terms in the Taylor expansion coupled with the Δx^2 term, but rather that the error (or reduction thereof) is proportional to Δx^2. The interpretation of this is that for a second order error, if, for example, we decrease the spacing by half, we can expect a reduction in error of 2^2. Higher order finite difference schemes are therefore desirable. As long as higher order derivatives for the data exist, higher order versions of finite difference

schemes can produce viable solutions. In the case of numerical differentiation used in this thesis, a fourth order scheme was applied and was found to be optimal.

An alternative to using finite differences is to use Fourier spectral methods, where differentiation is a simple multiplication. This can be significantly more accurate, but data must be periodic, which may not be the case. Care must be taken here, to avoid phenomena such as ringing, otherwise the accuracy will deteriorate severely. Equation (4.27) shows relationship to calculate the x-derivative in the Fourier domain.

$$\mathcal{F}\left[\frac{\partial u(x)}{\partial x}\right] = i \cdot k_x \mathcal{F}[u(x)] \tag{4.27}$$

where k_x is the spatial frequencies in x-direction direction and should not be confused with susceptibility. $\mathcal{F}[\,]$ expresses the Fourier transform.

To see how the forward difference, central difference and Fourier relationships relate to each other, take the Fourier transform of the Taylor series. We then have:

$$\begin{aligned}\mathcal{F}\big[u(x_j + \Delta x)\big] &= \mathcal{F}\big[u(x_j)\big] + \mathcal{F}\left[\Delta x_j \frac{\partial u}{\partial x}\Big|_{x_j}\right] + \mathcal{F}\left[\frac{\Delta x_j^2}{2!}\frac{\partial^2 u}{\partial x^2}\Big|_{x_j}\right] + \cdots \\ &= \mathcal{F}[u(x_i)] + (i \cdot k_x)\Delta x_j \mathcal{F}\big[u(x_j)\big] + \frac{(i \cdot k_x)^2 \Delta x_j^2}{2!}\mathcal{F}\big[u(x_j)\big] + \cdots\end{aligned} \tag{4.28}$$

This is the expansion for an exponential function, i.e. $e^x = 1 + x + x^2/2!$. Therefore we get:

$$\mathcal{F}\big[u(x_j + \Delta x)\big] = \mathcal{F}\big[u(x_j)\big] \cdot e^{ik_x \Delta x_j} \tag{4.29}$$

Equation (4.29) can now be used to compare the different schemes. If we now take the Fourier transform of the forward difference scheme, we get:

$$\begin{aligned}\mathcal{F}\left[\frac{\partial u(x_j)}{\partial x}\right] &= \mathcal{F}\left[\frac{u(x_j + \Delta x) - u(x_j)}{\Delta x}\right] \\ &= \mathcal{F}\big[u(x_j)\big] \cdot \frac{e^{ik_x \Delta x_j} - 1}{\Delta x} \\ &= \mathcal{F}\big[u(x_j)\big] \cdot (-1 + 1 + i \cdot k_x \Delta x + \cdots)/\Delta x \\ &= i \cdot k_x\, \mathcal{F}\big[u(x_j)\big] + \cdots\end{aligned} \tag{4.30}$$

The final simplification is achieved taking the Taylor expansion of the exponential term. Equation (4.30) implies that the frequency domain operator is an approximation to the forward difference. Taking note of the identity $\sin\theta = \frac{1}{2i}(e^{i\theta} - e^{-i\theta})$, we can do the same with the central difference:

$$\begin{aligned}\mathcal{F}\left[\frac{\partial u(x_j)}{\partial x}\right] &= \mathcal{F}\left[\frac{u(x_j + \Delta x) - u(x_j - \Delta x)}{2\Delta x}\right] \\ &= \mathcal{F}[u(x_j)] \cdot \frac{e^{ik_x \Delta x_j} - e^{-ik_x \Delta x_j}}{2\Delta x} \\ &= \frac{i \cdot \sin(k_x \Delta x)}{\Delta x}\, \mathcal{F}[u(x_j)] \\ &\approx i \cdot k_x\, \mathcal{F}[u(x_j)]\end{aligned} \tag{4.31}$$

The last line is the first term of the Taylor expansion for a sine function. This result also implies that the frequency domain operator is an approximation to the central difference, and that in the frequency domain the central difference corresponds to a sine function.

Both methods have been used in geophysics. For example, Daudt et al. (1989) compares both techniques when applied to seismograms. They came to the conclusion that for their application, all techniques tested had comparable accuracy.

Pajot et al. (2008) describe a third option, involving computing a low-degree polynomial fit of the data using a sample of neighbouring points. The analytical derivative of this polynomial function is then used to approximate the derivative of the component. This method is necessary when data are sampled more densely in one direction than in the other, but if the data are distributed evenly, according to Pajot, the method offers no advantage over the finite difference method.

Only horizontal gradients are calculated using one of these techniques. Vertical gradients use continuation as an integral part of the calculation. Horizontal gradients are always preferred to vertical gradients due to simplicity in calculations and reduction in possible noise. Fortunately, a rank 3 tensor can always be rewritten so that only horizontal gradients are calculated. For example:

$$B_{xyz} = B_{zxy} = B_{zyx} = B_{xzy} = B_{yzx} \tag{4.32}$$

$$B_{xxz} = B_{zxx} = B_{xzx} \tag{4.33}$$

$$B_{yyz} = B_{zyy} = B_{yzy} \tag{4.34}$$

$$B_{zzz} = -B_{xxz} - B_{yyz} \; = \; -B_{zxx} - B_{zyy} \tag{4.35}$$

Therefore, by using the relationships in equations (4.32) to (4.35), the calculation of vertical gradients directly can be avoided. The appropriate horizontal gradient calculation can be chosen to minimise on calculation errors.

4.4 Analytic Signal

Source detection techniques quite often utilise properties of the analytic signal, so this will be described as well.

An analytic signal is a complex signal which has no negative-frequency components. Real and imaginary parts of an analytic signal are related to each other by the Hilbert transform (Gabor, 1946). Thus we have:

$$\boldsymbol{A} = F + i\mathcal{H} \tag{4.36}$$

where $\boldsymbol{A}$ is the analytic signal, F is the original signal and $\mathcal{H}$ is the Hilbert transform of that signal.

The construction of the analytic signal can also be understood as suppressing the negative frequency components.

The analytic signal plays an important role in one-dimensional signal processing. One of the main reasons for this fact is that the instantaneous amplitude and the instantaneous phase of a real signal F at a certain position *x* can be defined as the magnitude and the angular argument of the complex-valued analytic signal $\boldsymbol{A}$ at the position *x*.

The analytic signal is a global concept, i.e. the analytic signal at a position x depends on the entire original signal and not only on values at positions near x.

Nabighian (1972) noticed that $\partial B_{tmi}/\partial x$ is the negative Hilbert transform of $\partial B_{tmi}/\partial z$, where B_{tmi} is the total magnetic intensity. Thus in 2D:

$$\begin{aligned} \boldsymbol{As_1} &= \frac{\partial B_{tmi}}{\partial x} + i\mathcal{H}\left(\frac{\partial B_{tmi}}{\partial x}\right) \\ &= \frac{\partial B_{tmi}}{\partial x} - \frac{i\partial B_{tmi}}{\partial z} \end{aligned} \tag{4.37}$$

$\boldsymbol{As_1}$ is the first order analytic signal. Essentially, vertical and horizontal derivatives of potential field data are the Hilbert transforms of each other. The amplitude of the 2D analytic signal is then:

$$As_1 = \sqrt{\left(\frac{\partial B_{tmi}}{\partial x}\right)^2 + \left(\frac{\partial B_{tmi}}{\partial z}\right)^2} \tag{4.38}$$

In 3D it follows that the analytic signal vector is defined as:

$$\boldsymbol{As_1} = \left(\frac{\partial B_{tmi}}{\partial x}, \frac{\partial B_{tmi}}{\partial y}, -\frac{i\partial B_{tmi}}{\partial z}\right) \tag{4.39}$$

The amplitude of this is:

$$As_1 = \sqrt{\left(\frac{\partial B_{tmi}}{\partial x}\right)^2 + \left(\frac{\partial B_{tmi}}{\partial y}\right)^2 + \left(\frac{\partial B_{tmi}}{\partial z}\right)^2} \tag{4.40}$$

Three versions of analytic signals exist for tensor data (Beiki, 2010) as well. They are:

$$\begin{aligned} As_{x1} &= \sqrt{\left(\frac{\partial B_x}{\partial x}\right)^2 + \left(\frac{\partial B_x}{\partial y}\right)^2 + \left(\frac{\partial B_x}{\partial z}\right)^2} \\ &= \sqrt{B_{xx}^2 + B_{xy}^2 + B_{xz}^2} \end{aligned} \tag{4.41}$$

$$\begin{aligned} As_{y1} &= \sqrt{\left(\frac{\partial B_y}{\partial x}\right)^2 + \left(\frac{\partial B_y}{\partial y}\right)^2 + \left(\frac{\partial B_y}{\partial z}\right)^2} \\ &= \sqrt{B_{xy}^2 + B_{yy}^2 + B_{yz}^2} \end{aligned} \tag{4.42}$$

$$\begin{aligned} As_{z1} &= \sqrt{\left(\frac{\partial B_z}{\partial x}\right)^2 + \left(\frac{\partial B_z}{\partial y}\right)^2 + \left(\frac{\partial B_z}{\partial z}\right)^2} \\ &= \sqrt{B_{xz}^2 + B_{yz}^2 + B_{zz}^2} \end{aligned} \tag{4.43}$$

where B_x, B_y, B_z are components of the field, and $B_{xx}, B_{xy}, B_{xz}, B_{yy}, B_{yz}, B_{zz}$ are tensor components.

Tensor analytic signals are useful for source distance calculations. However, in some cases the directional bias of each tensor analytic signal might not be desired. To deal with this, a tensor analytic signal magnitude is defined as follows:

$$As_{xyz} = \sqrt{As_x^2 + As_y^2 + As_z^2} \tag{4.44}$$

For first order analytic signals, this simplifies to:

$$As_{xyz1} = \sqrt{B_{xx}^2 + 2B_{xy}^2 + 2B_{xz}^2 + B_{yy}^2 + 2B_{yz}^2 + B_{zz}^2} \tag{4.45}$$

Other orders of analytic signal exist, although there is more than one definition of these orders. The first variant presented here is the variant in use by the source detection calculations (Cooper and Whitehead, 2016). In this case the second order analytic signal is defined as:

$$As_2 = \sqrt{\left(\frac{\partial As_1}{\partial x}\right)^2 + \left(\frac{\partial As_1}{\partial y}\right)^2 + \left(\frac{\partial As_1}{\partial z}\right)^2} \tag{4.46}$$

The zero order analytic signal is defined as

$$As_0 = \sqrt{{B_{tmi}}^2 + \mathcal{H}_x(B_{tmi})^2 + \mathcal{H}_y(B_{tmi})^2} \tag{4.47}$$

where $\mathcal{H}_x, \mathcal{H}_y$ are the Hilbert transforms in the x and y directions.

Building on this, higher orders of analytic signal for tensors are derived as follows:

$$As_{x2} = \sqrt{\left(\frac{\partial As_{x1}}{\partial x}\right)^2 + \left(\frac{\partial As_{x1}}{\partial y}\right)^2 + \left(\frac{\partial As_{x1}}{\partial z}\right)^2} \tag{4.48}$$

$$As_{y2} = \sqrt{\left(\frac{\partial As_{y1}}{\partial x}\right)^2 + \left(\frac{\partial As_{y1}}{\partial y}\right)^2 + \left(\frac{\partial As_{y1}}{\partial z}\right)^2} \tag{4.49}$$

$$As_{z2} = \sqrt{\left(\frac{\partial As_{z1}}{\partial x}\right)^2 + \left(\frac{\partial As_{z1}}{\partial y}\right)^2 + \left(\frac{\partial As_{z1}}{\partial z}\right)^2} \tag{4.50}$$

Similarly, the zero order tensor analytic signal is defined as:

$$As_{x0} = \sqrt{{B_x}^2 + \mathcal{H}_x(B_x)^2 + \mathcal{H}_y(B_x)^2} \tag{4.51}$$

$$As_{y0} = \sqrt{{B_y}^2 + \mathcal{H}_x\left(B_y\right)^2 + \mathcal{H}_y\left(B_y\right)^2} \tag{4.52}$$

$$As_{z0} = \sqrt{{B_z}^2 + \mathcal{H}_x(B_z)^2 + \mathcal{H}_y(B_z)^2} \tag{4.53}$$

The second variant of analytic signal orders by Hsu, Coppens and Shyu, (1998) is defined differently, namely as:

$$|A_n| = \sqrt{\left[\frac{\partial}{\partial x}(\nabla^n B_{tmi})\right]^2 + \left[\frac{\partial}{\partial y}(\nabla^n B_{tmi})\right]^2 + \left[\frac{\partial}{\partial z}(\nabla^n B_{tmi})\right]^2} \tag{4.54}$$

Where $\nabla^n = \frac{\partial^n}{\partial z^n}$ and $\nabla^0 = 1$. In this case, A_0 is the equivalent of the conventional analytic signal As_1.

4.4.1 Calculation of First and Second Order Analytic Signals

Numerical calculation of first and second order analytic signals is straight forward. This procedure is generic, so input is defined as dx, dy, dz. These can be any relevant x, y or z component of the field respectively. For example, dx can be $B_{xx}, B_{xy}, B_{yx}, B_{xz}$ or B_{zx}.

$$As_1 = \sqrt{dx^2 + dy^2 + dz^2} \tag{4.55}$$

Using central differences, the following horizontal gradients are calculated:

$$\begin{aligned} dyz &= gradient\ of\ dz\ w.r.t.\ y \\ dxz &= gradient\ of\ dz\ w.r.t.\ x \\ dyy &= gradient\ of\ dy\ w.r.t.\ y \\ dxy &= gradient\ of\ dx\ w.r.t.\ y \\ dxx &= gradient\ of\ dx\ w.r.t.\ x \\ dzz &= -dxx - dyy \end{aligned} \tag{4.56}$$

Using this, the following can then be calculated:

$$\frac{\partial As_1}{\partial x} = \frac{dx \cdot dxx + dy \cdot dxy + dz \cdot dxz}{As} \tag{4.57}$$

$$\frac{\partial As_1}{\partial y} = \frac{dx \cdot dxy + dy \cdot dyy + dz \cdot dyz}{As} \tag{4.58}$$

$$\frac{\partial As_1}{\partial z} = \frac{dx \cdot dxz + dy \cdot dzy + dz \cdot dzz}{As} \tag{4.59}$$

$$As_2 = \sqrt{\frac{\partial As_1^2}{\partial x} + \frac{\partial As_1^2}{\partial y} + \frac{\partial As_1^2}{\partial z}} \tag{4.60}$$

4.5 Source Distance

The following sections detail an application of tensors to source distance calculations, especially the work of Ma and Du (2012), Cooper (2014a, 2014b, 2015) and Cooper and Whitehead (2016). All tensor derivations are new and have been published (as an output of this thesis) in Cole and Cooper (2018).

4.5.1 Conventional method and results

Source detection techniques seek to calculate the distance from the surface to a source of some type, whether it is a dyke, a contact or other body. They are typically based on some form of manipulation of the analytic signal. Since the analytic signal finds the edges of bodies, or thin features such as dykes, these techniques are generally limited to this as well.

Cooper (2014c) showed that:

$$r = \frac{(N+1)As_1}{\mathrm{A}s_2} \tag{4.61}$$

where *r* is the distance to the source, *As* is the analytic signal amplitude of the magnetic field B_{tmi} and *As₂* is the 2nd order analytic signal. *N* is an index denoting the source type (Table 2).

As_1 and As_2 are defined as:

$$As_1 = \sqrt{\left(\frac{\partial B_{tmi}}{\partial x}\right)^2 + \left(\frac{\partial B_{tmi}}{\partial y}\right)^2 + \left(\frac{\partial B_{tmi}}{\partial z}\right)^2} \tag{4.62}$$

$$As_2 = \sqrt{\left(\frac{\partial As_1}{\partial x}\right)^2 + \left(\frac{\partial As_1}{\partial y}\right)^2 + \left(\frac{\partial As_1}{\partial z}\right)^2} \tag{4.63}$$

Table 2 Values of N versus magnetic source types (Ma and Li, 2013)

N	Magnetic Source Type
0	Contact or step
1	Vertical Dyke
2	Horizontal Cylinder
3	Dipole

The values in Table 2 represent indexes for the nature of a source and are not to be confused with the structural index as defined by Euler deconvolution. In Euler deconvolution, separate indices are defined for both magnetic and gravity cases (Reid and Thurston, 2014). However, the indices in Table 2 are related directly to different source types as modelled through the analytic signal. MacLeod, Jones and Dai (1993) give these equations for contacts, sheets (or dykes) and horizontal cylinders. This was generalized by Salem *et al.* (2004) and is expressed as:

$$As_1 = \frac{K}{(x^2 + z^2)^{\frac{(N+1)}{2}}} \tag{4.64}$$

Where *K* is a constant. We can substitute expression for B_{tmi} and its derivatives (section 4.3) into equation (4.62) to obtain the tensor version of the analytic signal amplitude As_1. Similarly (Cooper, 2014c):

$$\frac{\partial As_1}{\partial x} = (\frac{\partial B_{tmi}}{\partial x} \cdot \frac{\partial^2 B_{tmi}}{\partial x^2} + \frac{\partial B_{tmi}}{\partial y} \cdot \frac{\partial^2 B_{tmi}}{\partial y \partial x} + \frac{\partial B_{tmi}}{\partial z} \cdot \frac{\partial^2 B_{tmi}}{\partial z \partial x}) / As_1 \tag{4.65}$$

$$\frac{\partial As_1}{\partial y} = (\frac{\partial B_{tmi}}{\partial x} \cdot \frac{\partial^2 B_{tmi}}{\partial x \partial y} + \frac{\partial B_{tmi}}{\partial y} \cdot \frac{\partial^2 B_{tmi}}{\partial y^2} + \frac{\partial B_{tmi}}{\partial z} \cdot \frac{\partial^2 B_{tmi}}{\partial z \partial y}) / As_1 \tag{4.66}$$

$$\frac{\partial As_1}{\partial z} = (\frac{\partial B_{tmi}}{\partial x} \cdot \frac{\partial^2 B_{tmi}}{\partial x \partial z} + \frac{\partial B_{tmi}}{\partial y} \cdot \frac{\partial^2 B_{tmi}}{\partial y \partial z} + \frac{\partial B_{tmi}}{\partial z} \cdot \frac{\partial^2 B_{tmi}}{\partial z^2}) / As_1 \tag{4.67}$$

where derivative terms can be calculated using the equations and processes in section 4.3.

Equations (4.65), (4.66) and (4.67) are then substituted into (4.63) to obtain As_2. The tensor versions of As and As_2 can then be substituted into (4.61) to calculate *r*.

Synthetic tensor data, calculated for a cubic body, is shown in Figure 46. This data is used to test the tensor derivations for the source equations. The magnetic field intensity is 28,000 nT, susceptibility is 0.1 SI, inclination is -60 degrees, declination is -15 degrees. The rectangular prism has a width of 200 m and height of 280 meters, situated between 20 meters and 300 meters.

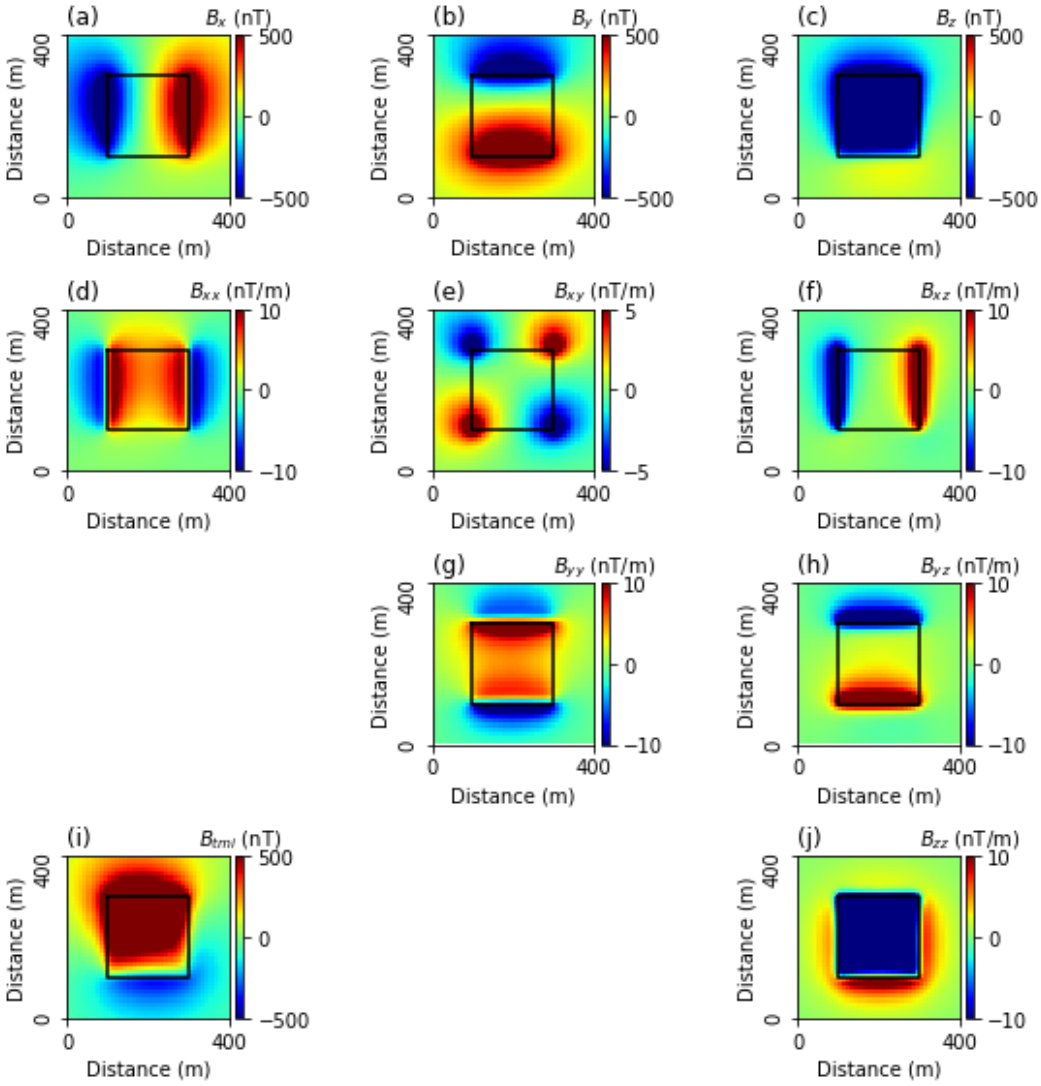

Figure 46 Synthetic magnetic tensor calculations for a rectangular prism. a) to c) show the three components of the magnetic field - B_x, B_y, B_z. d), e), f), g), h) and j) show the tensor components $B_{xx}, B_{xy}, B_{xz}, B_{yy}, B_{yz}, B_{zz}$ respectively. i) Total magnetic intensity of the study area. Note that $B_{xy} = B_{yx}, B_{xz} = B_{zx}, B_{yz} = B_{zy}$, so they are not shown. The magnetic field intensity used was 28000 nT, the susceptibility was 0.1 SI, the inclination was -60º, the declination was -15º. The depth of the rectangular prism was 20 m. The horizontal extent of the rectangular prism is 200 m by 200 m and it goes down to a depth of 3000 m

Figure 47 shows the datasets used to calculate *r*, namely As_1 and As_2, as well as r. Using the tensor data to calculate *r* located the edges of the body correctly (Figure 47d). Over the edges of the body the distance to the source becomes its depth. Notice that valid solutions for r are those closest to the source, and are therefore directly above the source. Insofar as a depth estimate is concerned, other solutions are invalid. The source itself is defined by the analytic signal representing either a contact or edge of a body (Figure 47 (b) is an example of the edge of body seen in the analytic signal), or a dyke.

Source distance calculations done in this way are not affected by field direction or susceptibility, making it a robust technique.

The equations for a dyke are valid for the lower extent of the dyke extending to infinity (or sufficiently deep enough). Any error in the calculation therefore decreases with increasing lower depth extent for the dyke. This is not necessarily a bad assumption in a model, given that dykes originate from deep below the surface.

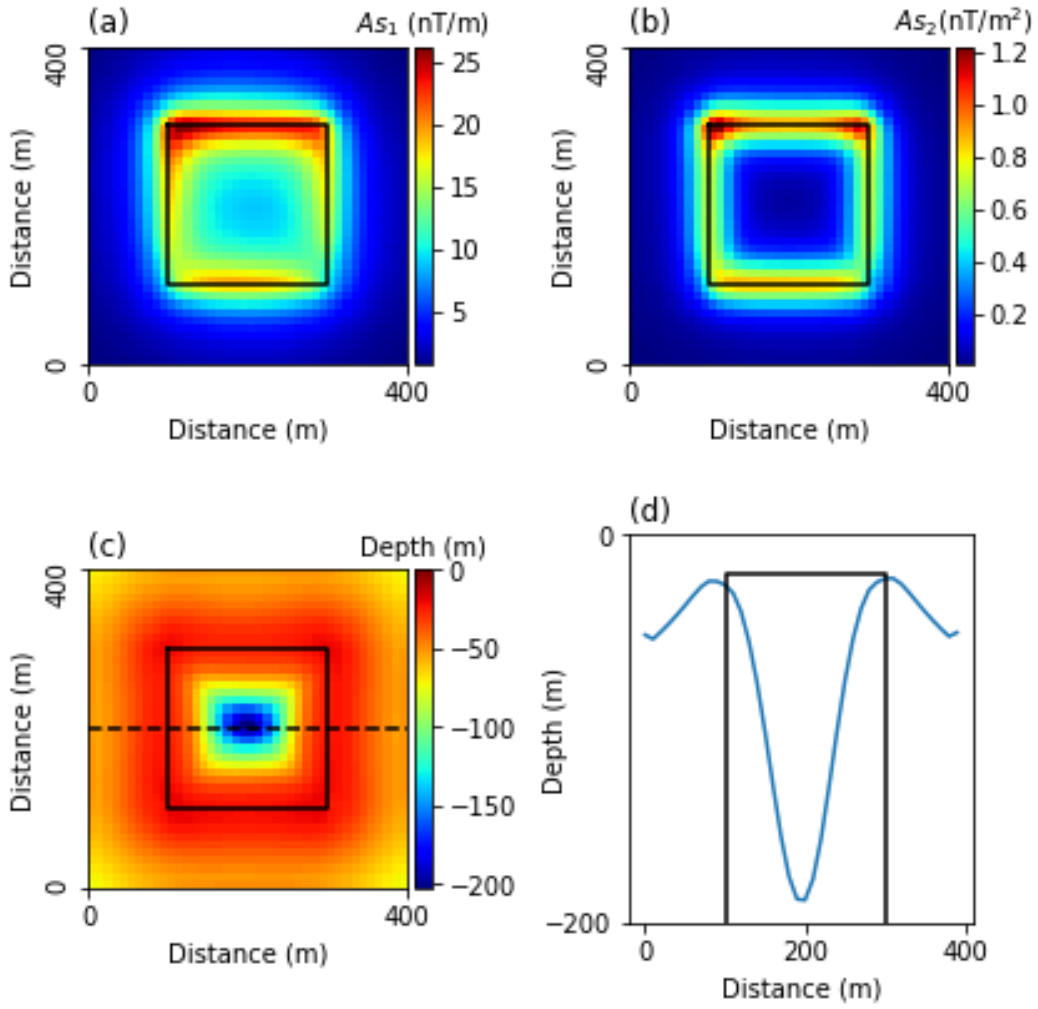

Figure 47 a) First order analytic signal of the data shown in Figure 46i). b) Second order analytic signal of the data shown in Figure 46i). c) Source-distance calculation results. d) Results of calculation of r from equation (4.61) (blue) over the synthetic modelled body for the profile shown as a dashed black line in c). A value of N = 0 was used in equation (4.61). Note that the negative of r is plotted so that the values closest to zero represent the source depth.

4.5.2 Alternative methods and results

Tensors provide alternative derivation possibilities for source distance equations. Two alternative methods are presented here.

Method 1

Cooper (2014c) derived equation (4.61) based on the work of MacLeod, Jones and Dai (1993) and then Salem *et al.* (2004), who showed that the analytic signal can be defined as:

$$As_1 = \frac{K}{(x^2 + z^2)^{\frac{(N+1)}{2}}} \tag{4.68}$$

where *K* is a constant and *N* is an index relating to source type. This equation is originally based on the work of Nabighian (1972) who defined the analytic signal as:

$$As_1 = \frac{\psi^2}{x^2 + z^2} \tag{4.69}$$

This is derived from the components of the magnetic anomaly from a thin dyke, which form a Hilbert transform pair, and is given by Nabighian (1972) :

$$\frac{\partial f}{\partial x} = \frac{\psi(x \sin\phi + z \cos\phi)}{x^2 + z^2} \tag{4.70}$$

$$\frac{\partial f}{\partial z} = \frac{\psi(x \cos\phi - z \sin\phi)}{x^2 + z^2} \tag{4.71}$$

where $\psi = 2kB_a c \sin d$, *k* is susceptibility, B_a is the earth's magnetic field, *d* is the dip of a thin infinite sheet, z is the depth of the dyke, x is the horizontal displacement of the dyke and *c* and ϕ are given in Table 3. The quantity f can be either the total, vertical or horizontal field. The derivative simply converts the anomaly from that of a step to that of a sheet. Therefore, when considering steps, one need deal with f only, but when dealing with dykes, the entire derivative must be considered. This is an important property, as will be seen later.

Table 3 Values of c and β for total, vertical and horizontal fields, where i is the inclination of the earth's magnetic field, A is the angle between magnetic north and the positive x axis , $\tan I = \tan i/\cos A$ and d is the dip of a thin infinite sheet or step. From (Nabighian, 1972)

	Total Field Anomaly	Vertical Field Anomaly	Horizontal Field Anomaly
c	$1 - \cos^2 i \sin^2 A$	$\sqrt{1 - \cos^2 i \sin^2 A}$	$\cos A \sqrt{1 - \cos^2 i \sin^2 A}$
ϕ	$2I - d - 90$	$I - d$	$I - d - 90$

To convert between total, vertical or horizontal fields, one need only adjust the equations for c and ϕ as given in Table 3. What is important to note from equations (4.68) and (4.69) is that they are equally valid for total, vertical and horizontal fields. This means that equation (4.61) is also valid for the primary field components for a thin dyke or a step. This implies that in equation (4.70) ${}^{\partial f}/_{\partial x}$ (for dykes) or f (for steps) can be substituted with B_{tmi}, B_x, B_y, B_z, while making the appropriate changes from Table 3

Noting that (4.41), (4.42) and (4.43) are the analytic signals of the primary field components, B_x, B_y, B_z can be combined with (4.61), resulting in three new relationships:

$$r = \frac{(N+1)As_{x1}}{\mathrm{A}s_{x2}} \tag{4.72}$$

$$r = \frac{(N+1)As_{y1}}{\mathrm{A}s_{y2}} \tag{4.73}$$

$$r = \frac{(N+1)As_{z1}}{\mathrm{A}s_{z2}} \tag{4.74}$$

The central differences technique shown in section 4.3.3 can easily be used to calculate the second order analytic signal.

Method 2

The equations in method 1 are sensitive to noise, because of the use of higher order derivatives. To overcome this by using lower order derivatives Cooper (2015) demonstrated that:

$$r = \frac{NAs_0}{As_1} \tag{4.75}$$

where

$$As_0 = \sqrt{B_{tmi}^2 + \mathcal{H}_x(B_{tmi})^2 + \mathcal{H}_y(B_{tmi})^2} \tag{4.76}$$

N is the index relating to source type (Table 2) and $\mathcal{H}_x, \mathcal{H}_y$ are the Hilbert transforms in the x and y directions. It can be shown that this also holds for tensor components. Let us start by examining the component B_z. For a dyke, we write (4.70) and (4.71) in terms of this component (e.g. $^{\partial f}/_{\partial z} = B_z$ and $^{\partial f}/_{\partial x} = \mathcal{H}_x(B_z)$) it follows that (from Nabighian, 1972):

$$\mathcal{H}_x(B_z) = -\frac{\psi_z(x \sin \phi_z + z \cos \phi_z)}{x^2 + z^2} \tag{4.77}$$

$$B_z = \frac{\psi_z(x \cos \phi_z - z \sin \phi_z)}{x^2 + z^2} \tag{4.78}$$

Where $\psi_z = 2kB_a\sqrt{1 - \cos^2 i \sin^2 A} \sin d$ and $\phi_z = I - d$, taken from Table 3, since in this case we are dealing with a vertical field anomaly. Values are also given for total field and horizontal field anomalies. Now, noting that (4.77) and (4.78) are 2D equations, we can assume the following:

$$\mathcal{H}_y(B_z) = 0 \tag{4.79}$$

Substituting (4.77), (4.78) and (4.79) into (4.76) and simplifying:

$$\begin{aligned} As_{z0} &= \sqrt{B_z^2 + \mathcal{H}_x(B_z)^2 + 0} \\ &= \sqrt{\frac{\psi_z^2(x \cos \phi_z - z \sin \phi_z)^2}{(x^2 + z^2)^2} + \frac{\psi_z^2(x \sin \phi_z + z \cos \phi_z)^2}{(x^2 + z^2)^2}} \\ &= \frac{\psi_z}{\sqrt{(x^2 + z^2)}} \end{aligned} \tag{4.80}$$

The horizontal and vertical derivatives of B_z are:

$$B_{zx} = \frac{-\psi_z\left((x^2 - z^2) \cos \phi_z - 2xz \sin \phi_z\right)}{(x^2 + z^2)^2} \tag{4.81}$$

$$B_{zz} = \frac{-\psi_z\left((x^2 - z^2)\sin\phi_z + 2xz\cos\phi_z\right)}{(x^2 + z^2)^2} \tag{4.82}$$

Given that:

$$As_{z1} = \sqrt{B_{zx}^2 + B_{zz}^2} \tag{4.83}$$

It follows that:

$$As_{z1} = \frac{\psi_z}{x^2 + z^2} \tag{4.84}$$

Dividing (4.80) by (4.84):

$$r = \frac{As_{z0}}{As_{z1}} \tag{4.85}$$

In a similar fashion it can be seen that:

$$r = \frac{As_{x0}}{As_{x1}} \tag{4.86}$$

If the dyke were oriented along the y axis instead of the x axis then

$$r = \frac{As_{y0}}{As_{y1}} \tag{4.87}$$

$As_{x0}, As_{y0}, As_{z0}$ are calculated using equation (4.76) with $f = B_x$, $f = B_y$ or $f = B_z$ respectively. If (4.85) is generalized for different $N > 0,\ r$ becomes:

$$r = \frac{NAs_{z0}}{As_{z1}} \tag{4.88}$$

As_z is also ideal since it is not biased towards any one direction and is symmetrical in the sense of reduced to the pole data. The other advantage of (4.88) is that there is no need to calculate any rank 3 tensor component numerically.

Note that the components of the analytic signal are interchangeable, as long as both components sample or detect the anomaly. For example, if the feature is a north-south dyke, it is detectable on both the x and the z components, but not the y component. Therefore, the y component cannot be used interchangeably while the x and z components can. Examples of interchanging these components are given below.

$$r = \frac{(N+1)As_{x1}}{As_{z2}} \tag{4.89}$$

$$r = \frac{(N+1)As_{y1}}{As_{z2}} \tag{4.90}$$

$$r = \frac{(N+1)As_{x1}}{As_{y2}} \tag{4.91}$$

And

$$r = \frac{NAs_{x0}}{As_{z1}} \tag{4.92}$$

$$r = \frac{NAs_{y0}}{As_{z1}} \tag{4.93}$$

$$r = \frac{NAs_{x0}}{As_{y1}} \tag{4.94}$$

We can build on this derivation for a dyke, by proposing a new derivation for a step. If we remember that (from Nabighian, 1972) f in (4.70) and (4.71) is actually the field of a step, we can substitute B_z directly into (4.70) and (4.71). We get (for a step)

$$\frac{\partial B_z}{\partial x} = B_{zx} = -\frac{\psi_z(x\sin\phi_z + z\cos\phi_z)}{x^2+z^2} \tag{4.95}$$

$$\frac{\partial B_z}{\partial z} = B_{zz} = \frac{\psi_z(x\cos\phi_z - z\sin\phi_z)}{x^2+z^2} \tag{4.96}$$

If we remember that:

$$\mathcal{H}_x(B_{zx}) = -B_{zz} \tag{4.97}$$

The implication of (4.95), (4.96) and (4.97) is that:

$$As_{zz0} = \sqrt{B_{zz}^2 + \mathcal{H}_x(B_{zz})^2 + \mathcal{H}_y(B_{zz})^2} \tag{4.98}$$

$$As_{zz1} = \sqrt{B_{zzx}^2 + B_{zzy}^2 + B_{zzz}^2} \tag{4.99}$$

Therefore following the same logic in deriving (4.85), for a step,

$$r = \frac{As_{zz0}}{As_{zz1}} \tag{4.100}$$

A final version of the depth equations is presented, using As_{xyz} demonstrated in section 4.4. These analytic signals are magnitude combinations of the As_x, As_y and As_z analytic signals. Using these, the new depth equations are:

$$r = \frac{NAs_{xyz0}}{As_{xyz1}} \tag{4.101}$$

$$r = \frac{(N+1)As_{xyz1}}{As_{xyz2}} \tag{4.102}$$

4.5.3 Edge Detection

In 2D, source distance techniques based on the analytic signal give optimal solutions on the crests of analytic signal peaks. This is because these points coincide with the location of the source in question. These crests are either centred over the source, in the case dykes, or are situated on the edges, in the case of steps or contacts. This fact can therefore be used to predetermine the optimal locations for solutions of r, should this be desired. It should be noted that peak values of r can be used, since they are by definition

closest to the source. Under perfect conditions this may be the best choice for determining source locations (since these peaks relate directly to the calculation of r itself). However, complex geological sources or noise and uncertainty in the data may necessitate using potentially cleaner signals for edge and source detection.

The difference between steps and dykes, in terms of analytic signals is important. Hsu, Coppens and Shyu (1998) point out that when a dyke-like feature becomes too shallow, the single peak separate into two separate peaks. For the analytic signal this happens when the width of the dyke is greater than the depth to the top of the dyke. For higher orders of analytic signal, this can happen sooner. An example of this is shown in Figure 48. The source was modelled in an ambient field of 28 000 nT, with inclination of -60 degrees, and declination of -15 degrees. The susceptibility was 0.1 SI. The analytic signals calculated were normalized for comparison purposes.

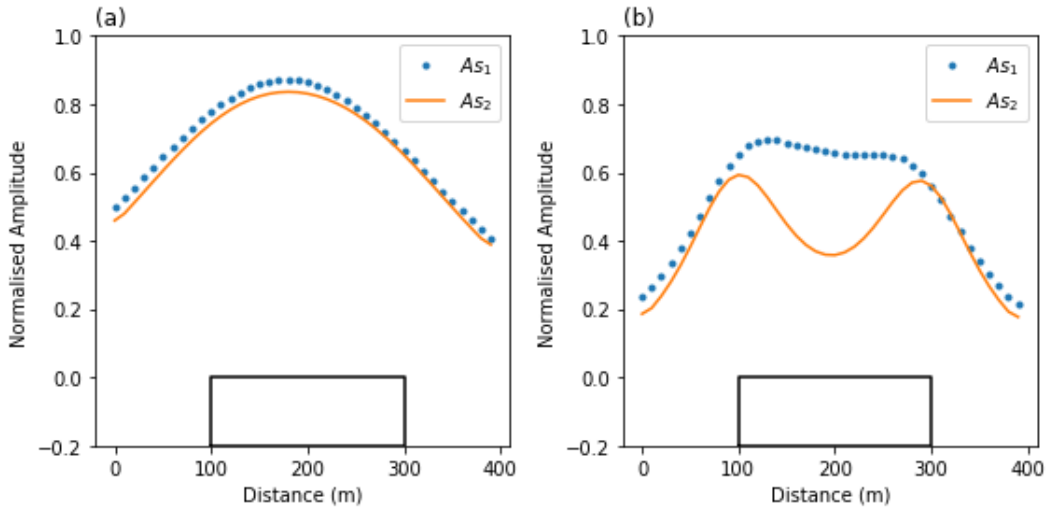

Figure 48 (a) Comparison between As and As_2, with a source of width 200 m and depth 200 m and (b) width 200 m and depth 70 m. The separation of peaks is apparent in (b).

Figure 48(b) in particular shows the transitionary situation where the lower order analytic signal has only one clear peak, while the higher order analytic signal shows two peaks. The implication of this is that the higher order analytic signal should be used for the detection of optimal solution locations. In addition, the transitionary nature of the analytic signal may be diagnostic of N being a fractional value, between 0 (for a step) and 1 (for a dyke). In this case the value of N=0.3 gave a solution of 75 m for depth to source edges, while in the case of Figure 48(a) the standard value of N=1 for a dyke gave a minimum solution of 220 m at the centre of the anomaly.

4.5.4 Susceptibility Calculation

Cooper (2015) showed that the susceptibility-width product of a thin dyke can be calculated, using the following equation (in SI units):

$$k.w = \frac{4\pi \cdot As_0 \cdot z}{2B_a.c} = \frac{4\pi \cdot As_1 \cdot z^2}{2B_a.c} \tag{4.103}$$

where k is susceptibility, w is the width of the dyke, B_a is the earth's magnetic field, z is the depth to the dyke, and $c = 1 - \cos^2 i \sin^2 A$ from Table 3. This was derived using the following two equations and setting $x = 0$:

$$As_0 = \frac{2k.B_a.c.w}{4\pi\sqrt{x^2 + z^2}} \tag{4.104}$$

$$As_1 = \frac{2k.B_a.c.w}{4\pi(x^2 + z^2)} \tag{4.105}$$

However, a simpler, new, version of this equation can be derived by squaring (4.104) and dividing this by (4.105). From this it follows that:

$$k.w = \frac{4\pi \cdot As_0^2}{As_1 \cdot 2 \cdot B_a \cdot c} \tag{4.106}$$

Similarly, by using the value of $cnew = \sqrt{1 - \cos^2 i \sin^2 A} = \sqrt{c}$ (Table 2), expressions from A_z can be derived. Therefore:

$$As_{z0} = \frac{2k.B_a.\sqrt{c}.w}{4\pi\sqrt{x^2 + z^2}} \tag{4.107}$$

$$As_{z1} = \frac{2k.B_a.\sqrt{c}.w}{4\pi(x^2 + z^2)} \tag{4.108}$$

From this, it follows that:

$$k.w = \frac{4\pi \cdot As_{z0}^2}{As_{z1} \cdot 2 \cdot B_a \cdot \sqrt{c}} \tag{4.109}$$

These equations have the advantage that they are not dependant on the depth of the source. However, if one of the analytic signal calculations is affected by noise, they may not be ideal. For versions incorporating depth and focussing on only one analytic signal, we have:

$$k.w = \frac{4\pi \cdot As_{z0} \cdot z}{2B_a.\sqrt{c}} \tag{4.110}$$

$$k.w = \frac{4\pi \cdot As_{z1} \cdot z^2}{2B_a.\sqrt{c}} \tag{4.111}$$

These last two equations are similar to (4.103), except that they now relate to A_{z0} and A_{z1}. The incorporation of depth means that more accurate estimates are possible if an accurate depth is available, through boreholes for example.

It is important to remember that the susceptibility-width product is being calculated, and not susceptibility alone. This means that an estimate for dyke width must be made in order to calculate the susceptibility of the dyke.

If the width can be estimated, then the calculation of the susceptibility is possible. For dyke like anomalies, the width of the dyke has to be less than the depth of the dyke. If this is not the case, the analytic anomaly separates into two anomalies (one for each edge) and the behaviour of the anomaly is more step–like.

Width estimation using analytic signals has been looked at by other authors. (Hsu, Coppens and Shyu, 1998; Bastani and Pedersen, 2001). The method by Hsu, Coppens and Shyu (1998) is examined here. The method calls for an alternative definition of the analytic signal.

$$|A_n| = \sqrt{\left[\frac{\partial}{\partial x}(\nabla^n B_{tmi})\right]^2 + \left[\frac{\partial}{\partial y}(\nabla^n B_{tmi})\right]^2 + \left[\frac{\partial}{\partial z}(\nabla^n B_{tmi})\right]^2} \tag{4.112}$$

where $\nabla^n = \frac{\partial^n}{\partial z^n}$ and $\nabla^0 = 1$. Using this definition, the half-width of a dyke was defined to be:

$$w = \sqrt{\frac{2d}{c_1} - d^2} \tag{4.113}$$

where $c_1 = \left|\frac{A_1}{A_0}\right|$. A_0 is the conventional analytic signal As_1 and d is the depth to the source. To calculate A_1, using the standard tensor components, we make use of equation (4.1), to calculate the following:

$$\frac{\partial}{\partial x}(\nabla^1 B_{tmi}) = \frac{\partial^2 B_{tmi}}{\partial xz} \tag{4.114}$$

$$\frac{\partial}{\partial y}(\nabla^1 B_{tmi}) = \frac{\partial^2 B_{tmi}}{\partial yz} \tag{4.115}$$

$$\frac{\partial}{\partial z}(\nabla^1 B_{tmi}) = \frac{\partial^2 B_{tmi}}{\partial z^2} \tag{4.116}$$

Expressions for this using tensor components can be found in section 4.3. These equations can easily be used to calculate width. Table 4 shows some examples of width calculations. The widths tend to be overestimated, but are nonetheless a good starting point. The accuracy of this width is dependent on an accurate estimate of depth.

Table 4 Comparison between true widths and calculated widths for various depths.

Depth (m)	True Width (m)	Calculated Width (m)
30	30	36
50	30	40.28
100	30	44
100	40	50
50	40	47.32

Alternatively, if the susceptibility is known (through petrophysical analysis or given an estimate based on geological knowledge), then the width can be estimated through the above equations.

4.5.5 Test – Step Model

The term 'step' is used to describe the steeply-dipping boundary between two bodies with significantly different density or magnetisation, where the far extents of the bodies are far enough away, that they do not contribute significantly to the anomaly at the boundary. Examples of where steps can occur are terraces and contacts.

Steps can be modelled by large rectangular prisms. A rectangular prism with a width of 200 meters, a height of 280 meters, at a depth of 20 meters was modelled. Since this is a step, a value of N=0 is chosen. The field strength was 28 000 nT. The inclination is -60 degrees and the declination is -15 degrees. The susceptibility is 0.1 SI. There is no remanence. The results are shown in Figure 49.

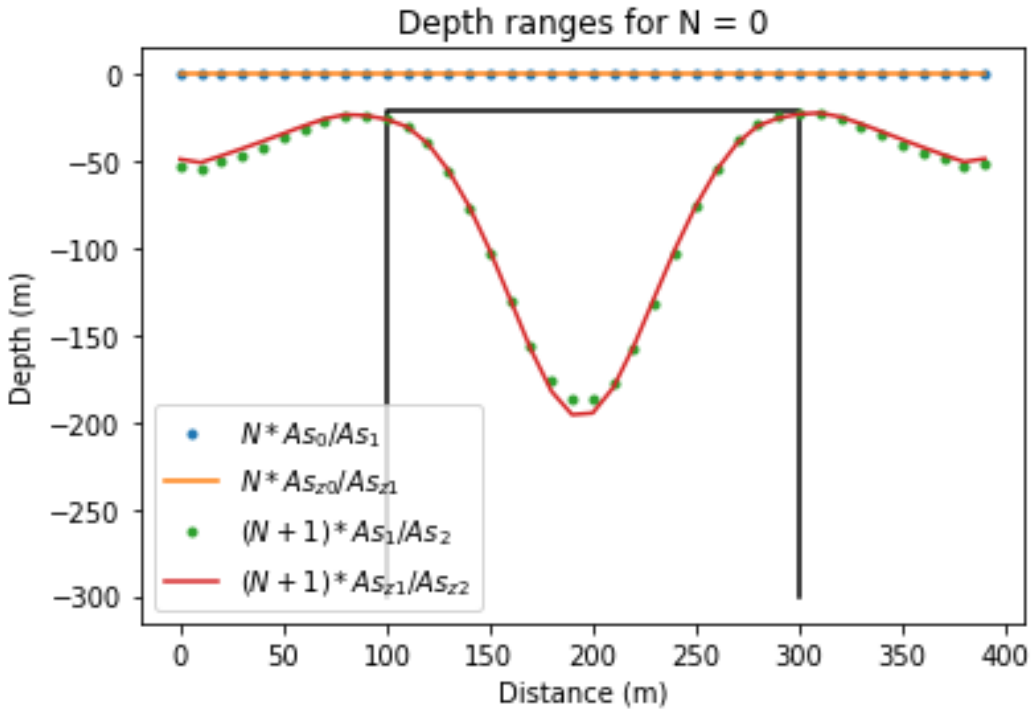

Figure 49 Results of negative r calculated for a step. Dots are non-tensor calculations. Lines are from tensor component calculations. Both were calculated at the same position. Values closest to zero give source depth and location. The synthetic model is also shown. Notice the straight line solution due to the inability of zero order analytic signals to calculate depth in this case (since N = 0).

Lines denote calculations using tensor analytic signals, and dots denote calculations using regular analytic signals. Notice that for the case N=0, no solution is possible for steps with zero order analytic signals. However, all equations with first and second order analytic signals produce accurate depth results.

4.5.6 Test – Dyke Model

A dyke model differs from a step in that the width is negligible compared to the depth. Both sides of the dyke contribute to the anomaly.

Dykes can be modelled by thin rectangular prisms. A dyke with a width of 2 meters, length of 4000 meters, a height of 2800 meters, at a depth of 200 meters was modelled. Since this is a dyke, a value of N=1 is chosen. The field strength was 28 000 nT. The inclination is -60 degrees and the declination is -15 degrees. The susceptibility is 0.1 SI. There is no remanence. The results are shown in Figure 50.

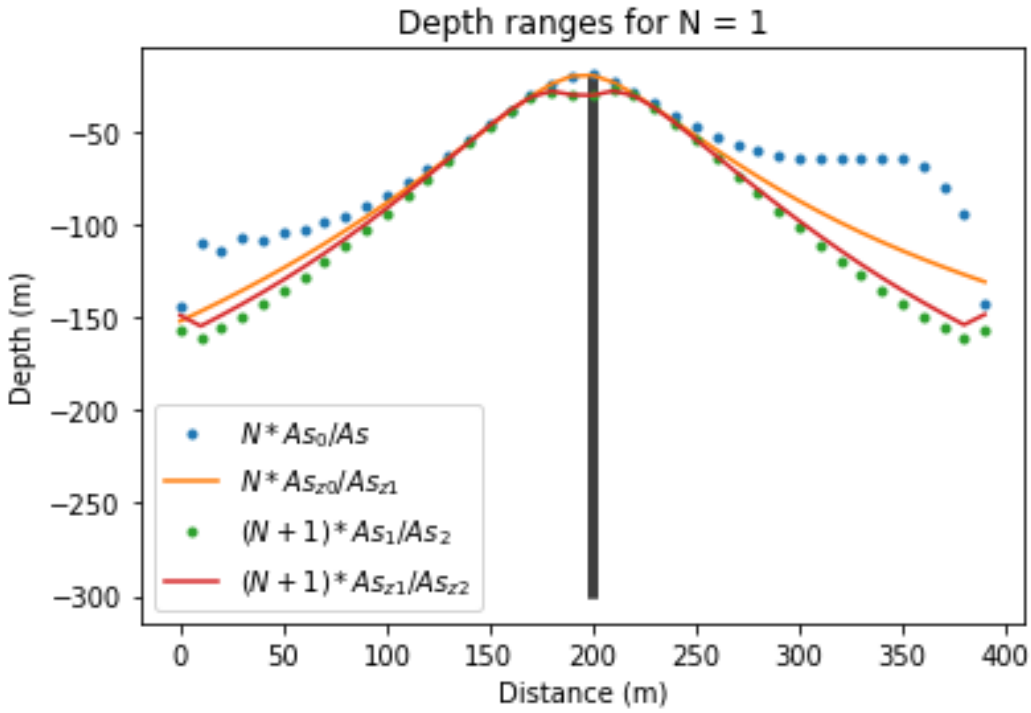

Figure 50 Results of r calculated for a dyke. Dots are conventional calculations. Lines are from tensor component calculations. The synthetic model is also shown.

Lines denote calculations using tensor analytic signals, and dots denote calculations using regular analytic signals. The tensor analytic signals produce similar results to the regular analytic signals. Of interest is that the results for the zero order analytic signals seem superior. All equations produce similar results and the depth calculation seems to be effective.

In order to examine the effect of dip on the calculation of depth, two models were calculated. One model was a vertical dyke (Figure 51) and the other model was a dipping dyke (Figure 52). The field strength was 28 000 nT. The inclination is -60 degrees and the

declination is -15 degrees. The susceptibility is 0.1 SI. There is no remanence. Both dykes had a width of 20 m and a depth of 30 m. Equation (4.75) was used to calculate depth estimates. In both cases, as can be seen from the figures, the depth estimate is calculated reliably. The reason for this is due to the high decay rate of magnetic anomalies, which means that the majority of the signal being used for the depth comes from the closest point to the surface. Therefore, dip does not affect the depth calculation. With this in mind, future models will use a vertical dyke, for simplicity's sake. The simple visual identification of a dipping dyke can be seen by the separation between the analytic signal anomaly and the corresponding magnetic anomaly, as can be seen by a simple comparison between Figure 51 and Figure 52.

A more detailed examination of the effects of noise, is shown in Figure 53.

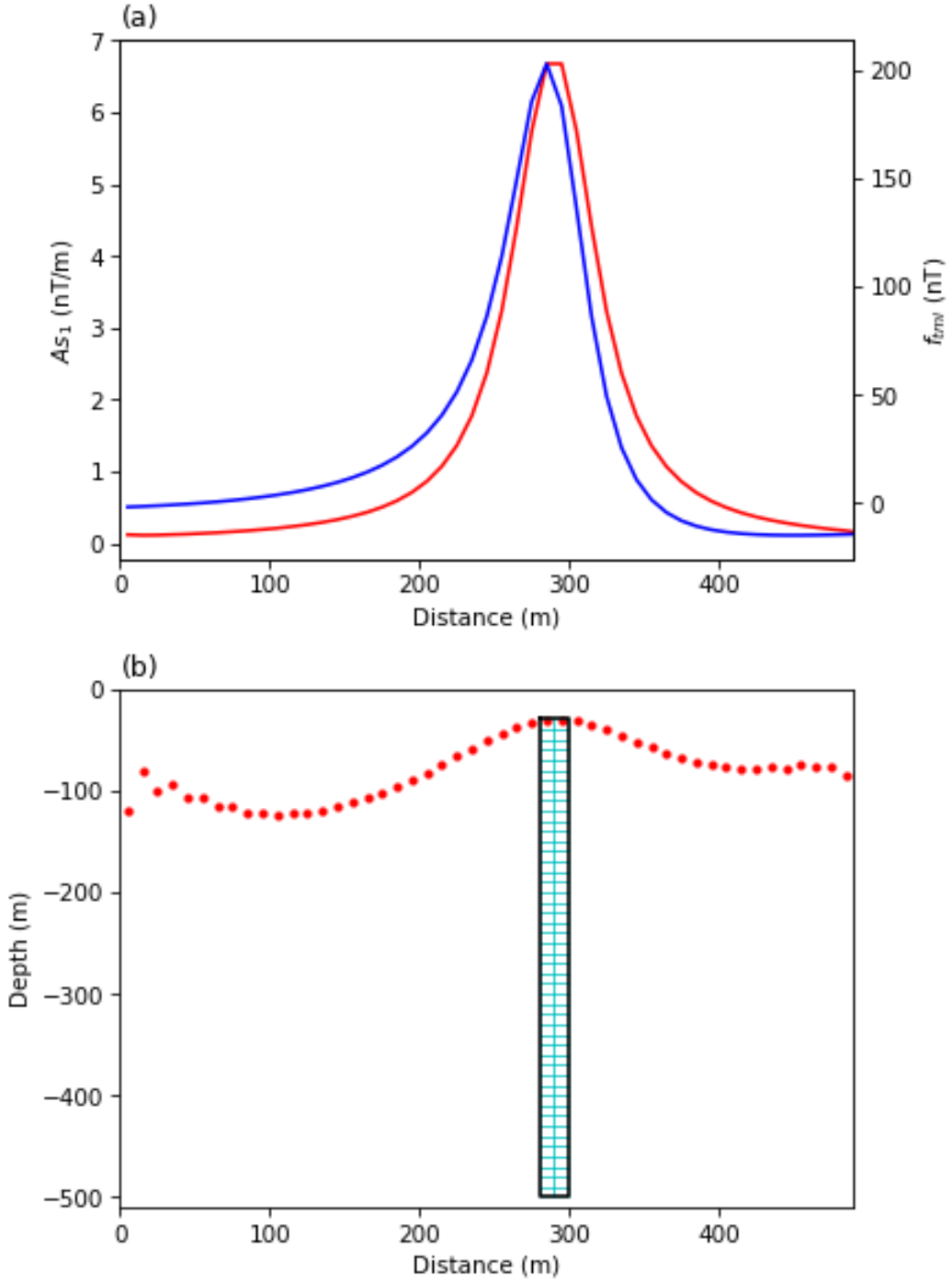

Figure 51 a) shows the magnetic field (blue) and analytic signal (red) over a vertical dyke b) shows the dyke and the calculated solutions for depth.

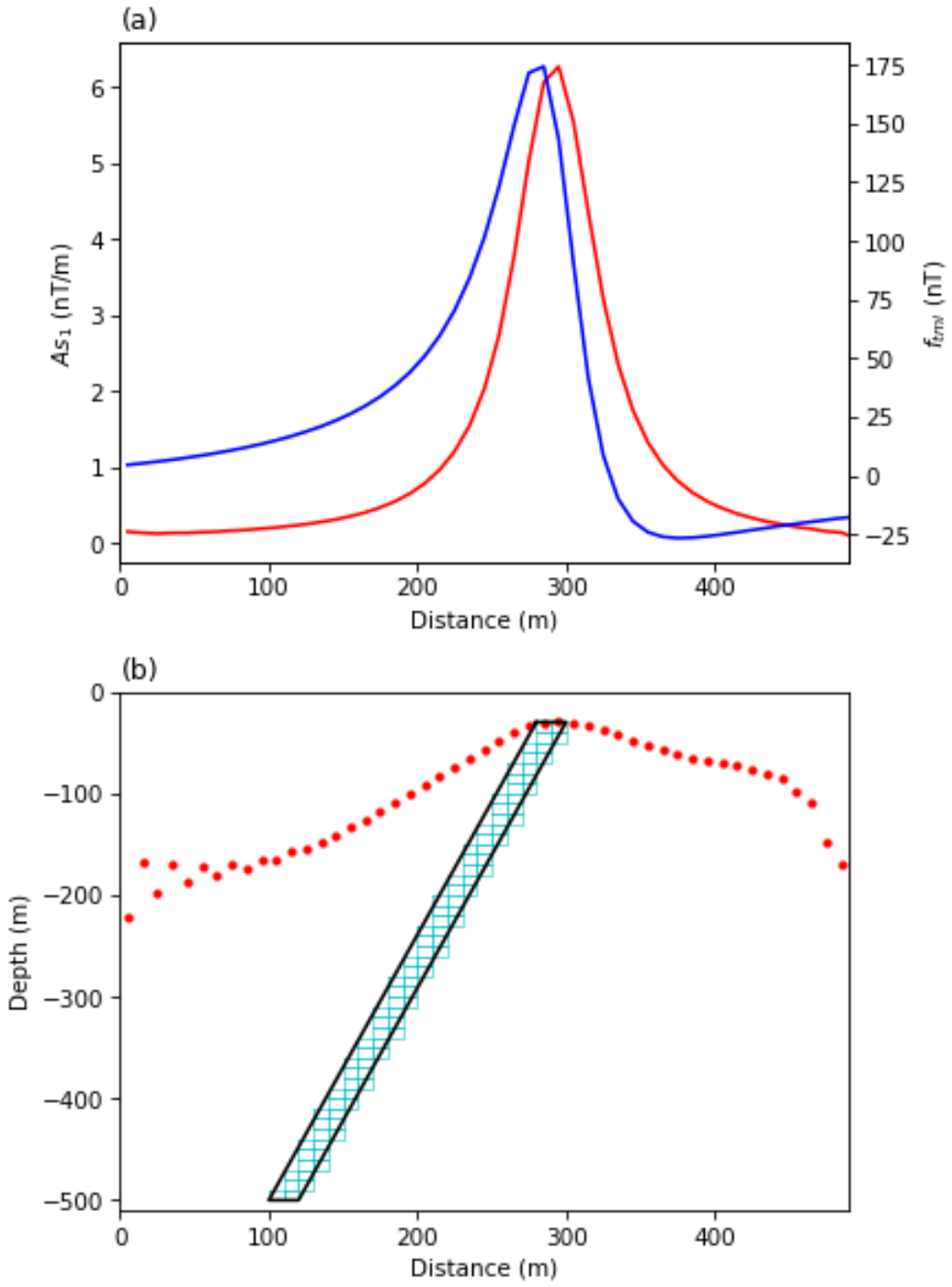

Figure 52 a) shows the magnetic field (blue) and analytic signal (red) over a dipping dyke b) shows the dyke and the calculated solutions for depth.

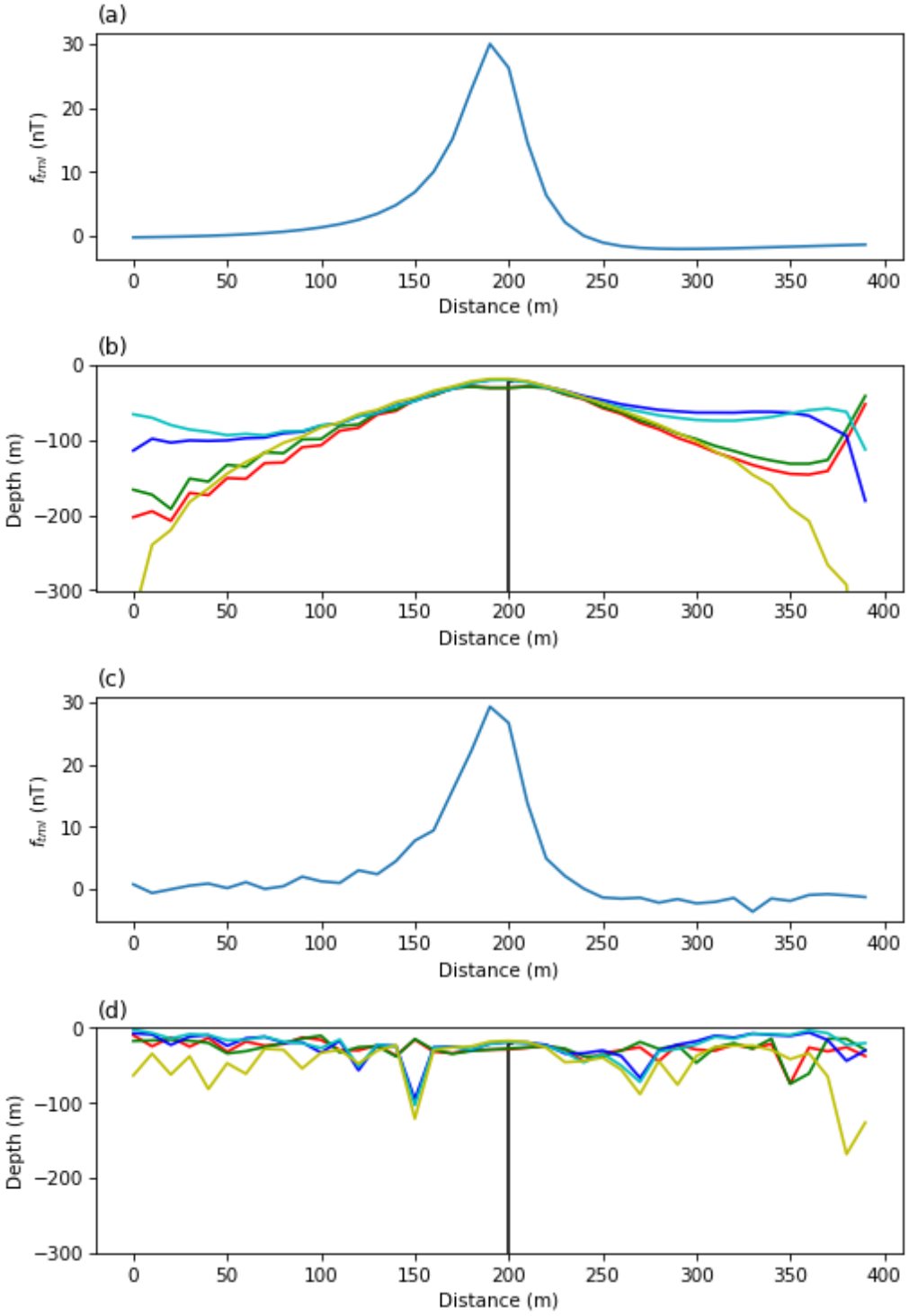

Figure 53 a) Noise free total magnetic intensity over a thin dyke of width 2 metres, with magnetic field intensity of 28,000 nT, susceptibility of 0.1 SI, inclination of -60º, declination of -15º and depth of the dyke equal to 20 m, b) Source-distance calculations over a thin dyke with noise-free data. Results from equations (4.72), (4.74), (4.75), (4.85) and (4.86) are shown in red, green, blue, cyan and yellow respectively. c) Total Magnetic Intensity for the same dyke with Gaussian noise with a standard deviation equal to 1.58% of the maximum data amplitude added d) Source-distance calculations over the thin dyke with the noisy TMI data, using same colour scheme as in (b).

Figure 53 shows source-distance calculations for the magnetic anomaly from a dyke using equations (4.72), (4.74), (4.85) and (4.86) with N = 1. The depth to the dyke is determined reliably both when the data is noise free (Figure 53a and b) or had Gaussian noise with a standard deviation equal to 1.58% of the maximum data amplitude added to the TMI (Figure 53c and d).

Figure 54 investigates the effect of varying degrees of noise added to the individual tensor components. This test uses equations (4.85) and (4.86) with N = 1, since these equations produced better results than equation (4.72) and (4.74). Figure 54 a), b), c) and d) show Gaussian noise envelopes with a standard deviation of 0 nT, 0.01 nT, 0.1 nT and 1 nT added to each tensor component respectively. In all cases the estimate of depth over the dyke is reasonable, although in the last case the dyke location is harder to detect on the data itself. Moving away from the location of the dyke, the stability of the solution becomes significantly compromised by the noise.

Figure 55 investigates the same noise effect using equations (4.72) and (4.74) with N = 1, In all cases the estimate of depth over the dyke is over estimated. In the case of (d), the depth is compromised by the noise.

In all these cases, the need to identify (or at least verify) the dyke location using a method other than source distance is obvious. Noisy data can obscure the location of the dyke, but if that location is known, the solution remains viable.

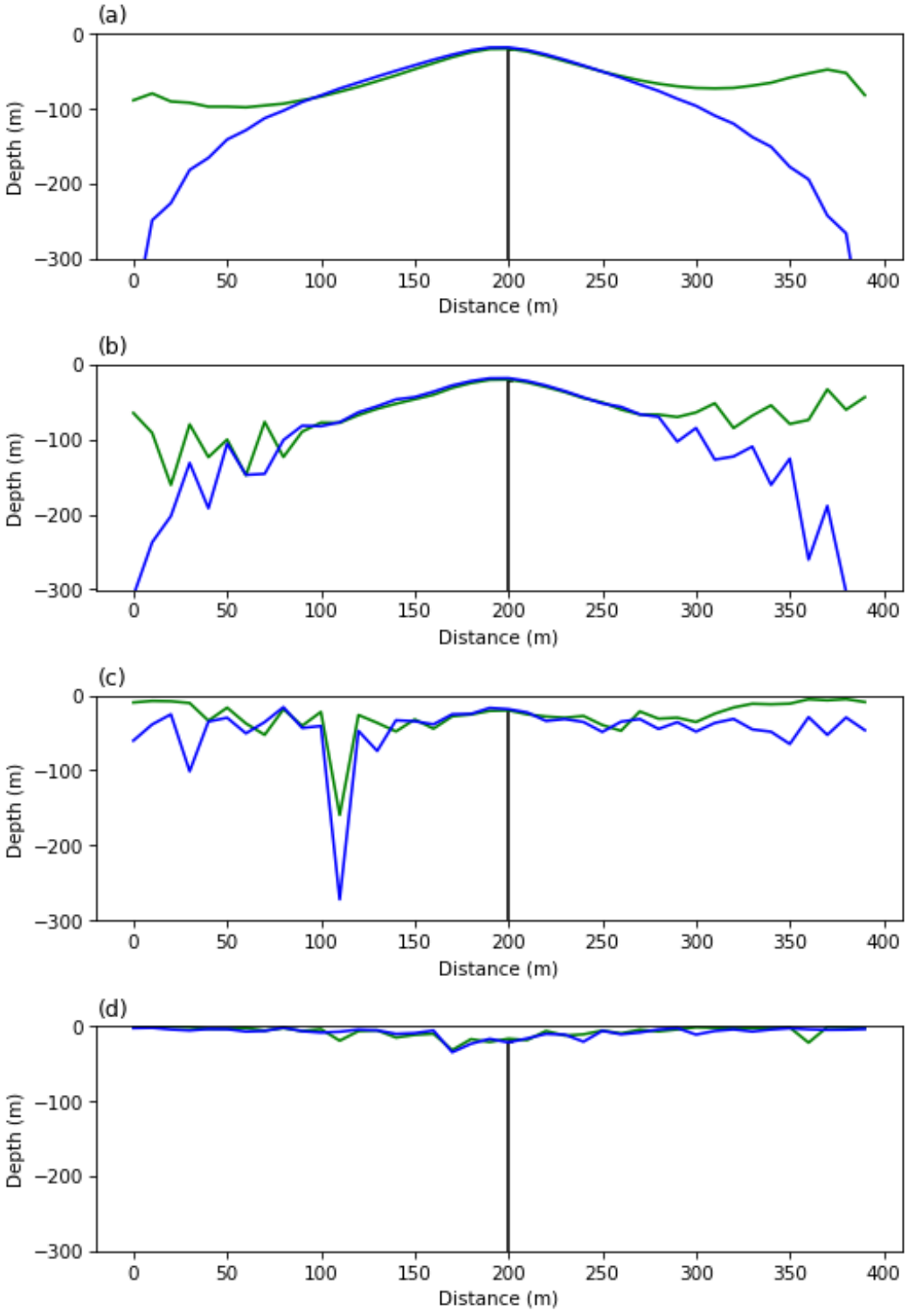

Figure 54 Varying levels of noise applied to the dyke from Figure 53, using equations (4.85) in green and (4.86) in blue. a) 0 nT Gaussian noise added, b) 0.01 nT Gaussian noise added, c) 0.1 nT Gaussian noise added d) 1 nT Gaussian noise added.

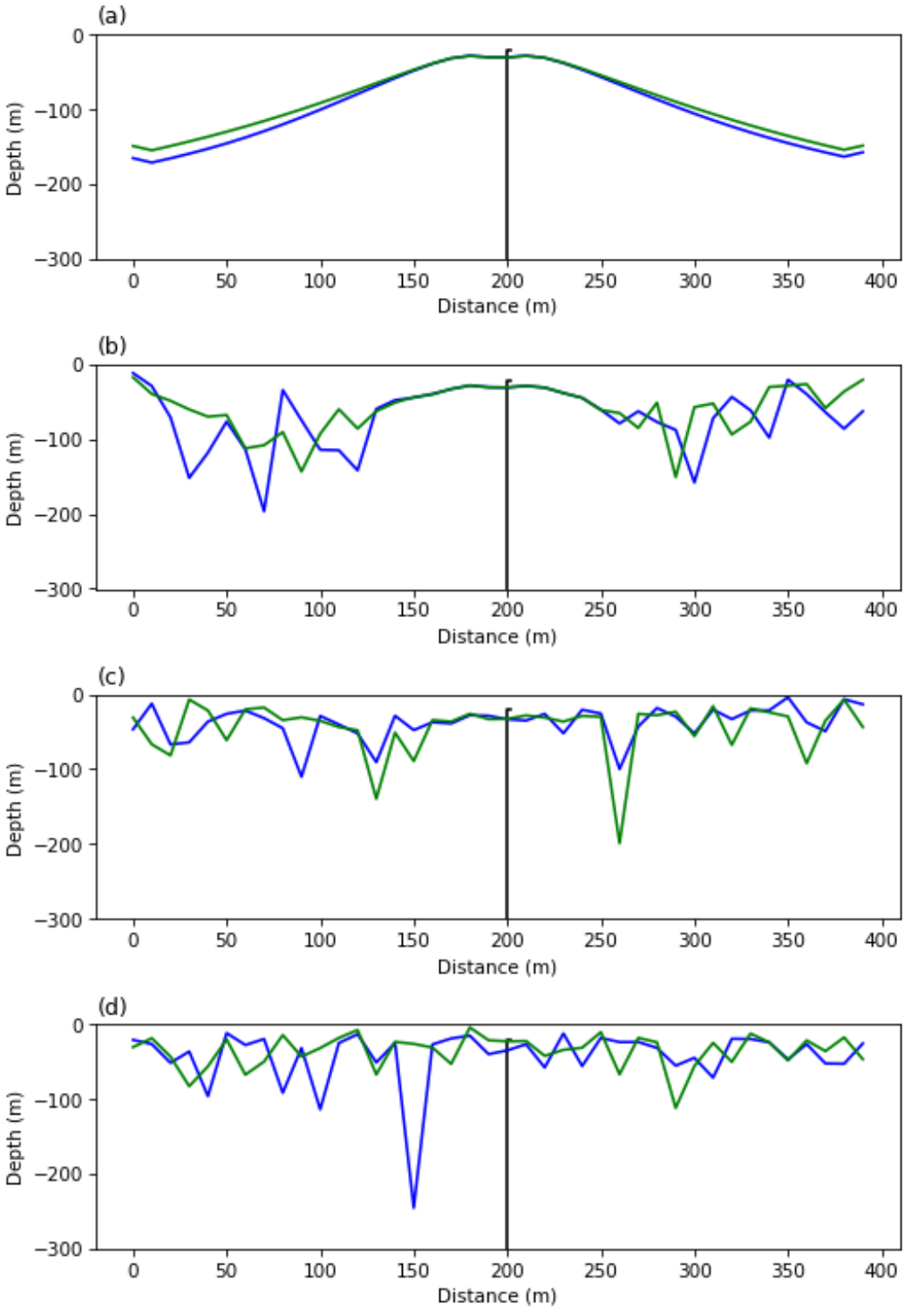

Figure 55 Varying levels of noise applied to the dyke from Figure 53 using (4.72) in blue and (4.74) in green. a) 0 nT Gaussian noise added, b) 0.01 nT Gaussian noise added, c) 0.1 nT Gaussian noise added d) 1 nT Gaussian noise added.

Figure 56 shows the effects of using equations (4.74), (4.85), (4.89) to (4.94) over the same north–south dyke. Results from equations (4.74) and (4.85) are shown in solid yellow and solid red. Equations (4.89) and (4.92) are shown in yellow and red x's. Equations (4.90) and (4.93) are shown in yellow and red dots. Red was chosen to denote ratios which use As_0/As_1 and yellow was chosen to denote ratios which use As_1/As_2.

Equations (4.89), (4.92), (4.74) and (4.85) all produce good results. Equations (4.90) and (4.93) however, use As_{y0} and As_{y1} which cannot detect the dyke in its north-south orientation easily, so the calculations break down. Equations (4.91) and (4.94) both use only As_x and As_y. Since there is extremely bad coupling of As_y to the dyke, the results are unstable and produce depths in excess of the 200 m limit on Figure 56.

The equations have different strengths and potentially different uses. Equations which use only As_{z0} and As_{z1} will detect all anomalies, since these analytic signals have no horizontal bias. For pure source depth calculation, it is advisable to therefore use equations which rely on As_z only, such as (4.74) and (4.85).

If selective depth calculations are needed with some directional bias, then equations (4.89) to (4.94) may be helpful. Detecting which solutions are valid in such cases is straightforward. One needs only compare the solution of one of these equations with depths produced by (4.74) and (4.85). If they are within an acceptable threshold, then the directionally biased version of depth is valid. Other depths can then be discarded.

The solutions for equations (4.74) and (4.89) (shown in a yellow line and yellow dots) show a low where the dyke occurs. As discussed in section 4.5.3, this is due to the depth of the dyke being too shallow for the use of those equations. The depth can be halved in this case, converting the equations to that of a step, and potentially improving the depth estimate. Alternatively, solutions from (4.74) and (4.85) can be compared as a diagnostic tool to confirm that a shallow dyke is present or verify the reliability of assuming the anomaly belongs to a dyke.

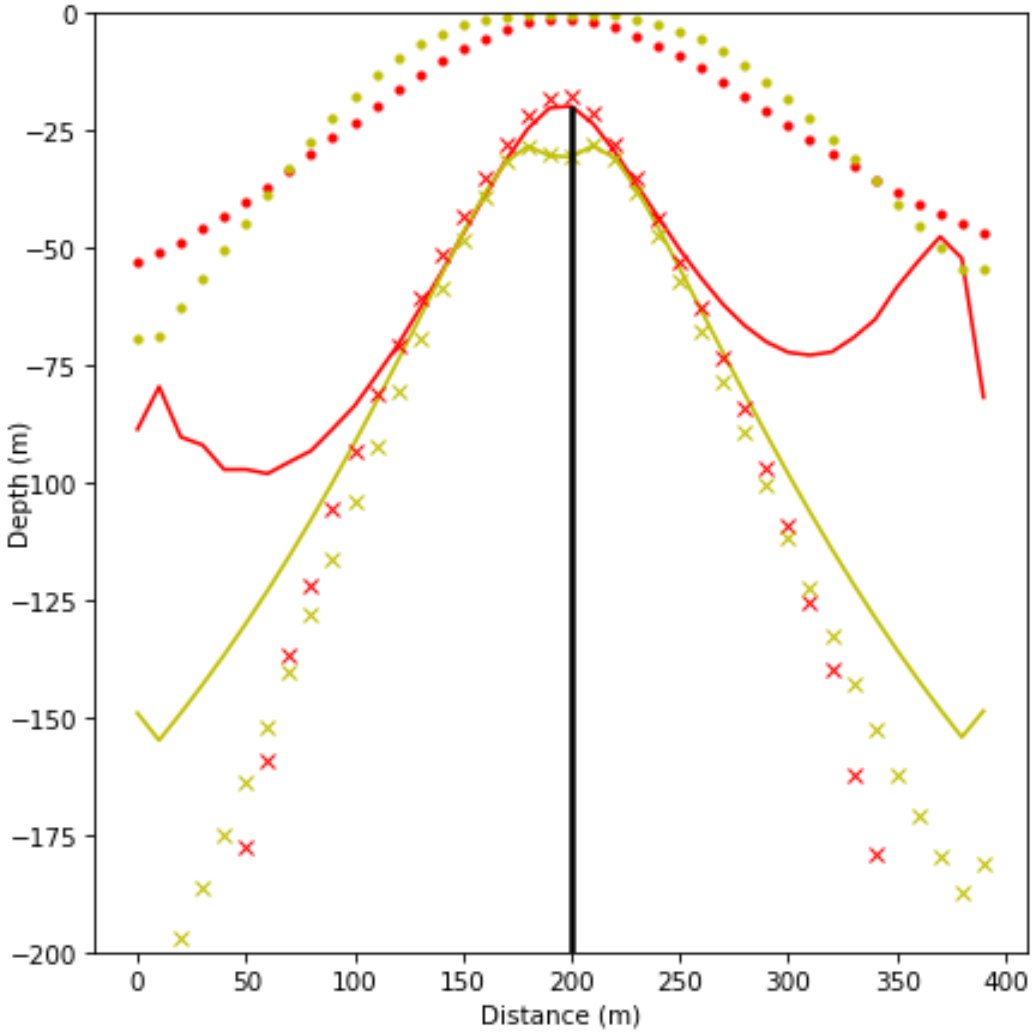

Figure 56 Results from equations (4.74) and (4.85) are shown in solid yellow and solid red. Equations (4.89) and (4.92) are shown in yellow and red x's. Equations (4.90) and (4.93) are shown in yellow and red dots. The model is shown in black.

4.6 Discussion

The source distance technique is a reliable method for depth determination, as long as the source type is known. Still, this limitation is not too restrictive, since it enables interpreters to have good initial starting models which can be refined later using forward modelling.

As a general rule, step or contact like bodies are defined to be those which have widths greater than their depths. Dykes are defined to be those where the width of the dyke is less than the depth (Hsu, Coppens and Shyu, 1998). Source type determination is important, especially from the perspective of what the magnetic field can resolve. If the dyke is very deep, however, it may masquerade as another source type (such as a line of dipoles from the perspective of the analytic signal) and higher orders of N may be necessary. Nevertheless, magnetic data is recognised as a shallow interpretation technique, so this limitation is not too problematic within the context of the dataset.

It should be noted that the use of magnetics might not limited to shallow interpretations. For example, it is routinely used for deeper volcanic body interpretation in the oil industry. In this case, this limitation must be taken into account. Alternative techniques can, of course, be considered, such as the depth estimation through power spectrum analysis proposed by Spector and Grant, (1970).

CHAPTER 5 REMANENCE CALCULATION

5.1 Introduction

Magnetic sources can be magnetised in a direction different to the direction of the geomagnetic field. If such magnetisation occurs, we say that the source exhibits remanence. Effective interpretation of magnetic data means that this magnetisation must be quantified and accommodated. Many methods have been developed to determine remanence and Clark (2014) reviewed these. This introduction will briefly present the main methods used historically.

The magnetization can be measured directly on oriented samples taken directly from the field (Clark and Emerson, 1991). The advantage of this method is the accuracy of measurement. Challenges here are in obtaining samples that are representative of the anomaly being studied, or all facets of it (especially if the anomaly is not homogenous).

Borehole measurements can provide another source of information. This can provide reliable information of a source from within a borehole which intersects the source. The measurements are taken either from samples extracted from the hole, or i*n situ* by a borehole logging tool (Bosum, Eberle and Rehi, 1988).

Petrologic and palaeomagnetic information can be used to infer probable susceptibility and remanence directions based on geological history, magnetic mineral grain size ranges, compositions and modal percentages. This requires some form of model to be developed based on the type of ore being studied. Ranges of susceptibility, remanence intensity and Q-ratio (Koenigsberger ratio, see section 2.2.4) for a variety of magnetic mineral compositions, domain states and types of NRM are available in the literature for use (Clark, 1997) These models are normally expressed in terms of simple formulae. By applying some form of constraint (such as model body compactness) which is thought to mimic the geology being studied, magnetisation can be determined, or at the very least, accounted for. Examples of this are the determination of apparent susceptibility and apparent magnetisation based on model constraints provided by terrain, geology etc. These assumptions must be at least approximately correct for good solutions.

Vector and gradient data have opened up possibilities for direct inversion leading to the estimation of remanent parameters. Simple sources are a convenient method for obtaining these parameters from such data.

If both vector magnetic and vector gravity data are available, the use of Poisson's theorem can allow for the estimation of magnetisation parameters. The reasoning is that the same source, giving rise to both magnetic and gravity anomalies, can make use of a magnetic anomaly calculated directly from the gravity data in order to compare this with measured magnetic field. To clarify this, Clark (2014) showed that the reduced to the pole magnetic anomaly is proportional to the vertical gradient of the gravity anomaly. Comparison of this calculated magnetic field against the measured magnetic field offers the possibility for parameter estimation.

Another method uses a controlled magnetic source to investigate the susceptibility distribution of a (typically shallow) subsurface. This takes the form of a long current carrying cable or a dipole source and frequency domain EM systems are one choice for this.

Helbig analysis (Helbig, 1963) provides another means to calculate magnetisation parameters and will be described in the next section.

Reduction to the pole has also been used. Correctly estimating magnetisation direction results in an anomaly that is predominantly positive, and more symmetric than TMI anomalies. Many different methods to achieve this have been described by Clark (2014). These techniques tend to work well for simple, shallow sources with steep dips, or when contacts are shallow. They are suited to scanning large areas to detect anomalous magnetisation direction, which can then be analysed further with other methods (Clark, 2014).

Base stations can be used in the vicinity of sources to measure the vector and gradient fields. The principle is that time-varying electric currents flowing in the ionosphere and magnetosphere results in changes in induced magnetisation. However, remanent magnetisation remains unaffected. These techniques take advantage of this difference in order to determine magnetisation parameters.

5.1.1 Helbig Method

A review of this technique is warranted because of its application to tensor data. Helbig (1963) noticed that the integrals of anomaly components, all having the same magnetisation direction over an infinite plane, are zero. The Helbig method makes use of this fact to estimate the vector components of the total magnetisation (M_x, M_y, M_z) for a compact source, from the first integral moments of the vector components of the anomalous magnetic field (B_x, B_y, B_z). These moments are calculated as follows:

$$M_x = -\frac{1}{2\pi}\int_{-\infty}^{\infty}\int_{-\infty}^{\infty} xB_z dxdy \tag{5.1}$$

$$M_y = -\frac{1}{2\pi}\int_{-\infty}^{\infty}\int_{-\infty}^{\infty} yB_z dxdy \tag{5.2}$$

$$M_z = -\frac{1}{2\pi}\int_{-\infty}^{\infty}\int_{-\infty}^{\infty} xB_x dxdy = -\frac{1}{2\pi}\int_{-\infty}^{\infty}\int_{-\infty}^{\infty} yB_y dxdy \tag{5.3}$$

Schmidt and Clark (1998) applied this theory to vector components calculated by Fourier processing of a conventional total magnetic intensity survey. Others have continued to study this approach (Phillips, 2005). Phillips et al. (2007) applied the methodology to magnetic tensor data.

As long as a planar surface is subtracted from each magnetic vector component, one can integrate over a limited area, since the mean component value in the window is zero, and the requirement that the first order integral moments should vanish is approximately met (Phillips, 2005). From these calculations, total magnetisation directions can be estimated. Phillips et al., (2007) proposed using this form of calculation in a moving window over a grid in order to estimate the total magnetisation continuously. They did point out, though, that the estimate would only be accurate over sources. According to them, the location of the source would be determined through some other technique. Phillips et al., (2007) extended the equations to the magnetic tensor, by deriving the following expressions:

$$M_x = \frac{1}{4\pi}\int_{-\infty}^{\infty}\int_{-\infty}^{\infty} x^2 B_{zx} dxdy = \frac{1}{2\pi}\int_{-\infty}^{\infty}\int_{-\infty}^{\infty} xyB_{zy} dxdy \tag{5.4}$$

$$M_y = \frac{1}{4\pi}\int_{-\infty}^{\infty}\int_{-\infty}^{\infty} y^2 B_{zy} dxdy = \frac{1}{2\pi}\int_{-\infty}^{\infty}\int_{-\infty}^{\infty} xyB_{zx} dxdy \tag{5.5}$$

$$
\begin{aligned}
M_z &= \frac{1}{4\pi} \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} x^2 B_{xx} dxdy \\
&= \frac{1}{4\pi} \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} y^2 B_{yy} dxdy \\
&= \frac{1}{2\pi} \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} xy B_{xy} dxdy \\
&= \frac{1}{2\pi} \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} xy B_{yx} dxdy
\end{aligned}
\tag{5.6}
$$

Clark (2014) expands this further to adapt the Helbig method to the concept of Normalised Source Strength (NSS). If we calculate the eigenvalues and eigenvectors of a magnetic gradient tensor, we obtain three eigenvalues $(\lambda_1, \lambda_2, \lambda_3)$. From this, the NSS is defined as:

$$
\mu = \sqrt{-\lambda_2^2 - \lambda_1^2 \lambda_3^2} \tag{5.7}
$$

where $\lambda_1 \geq \lambda_2 \geq \lambda_3$. Based on this, the horizontal location, depth and magnitude of magnetisation can be calculated using the following equations:

$$
x_0 = \frac{\int_{-\infty}^{\infty} \int_{-\infty}^{\infty} x\mu dxdy}{\int_{-\infty}^{\infty} \int_{-\infty}^{\infty} \mu dxdy} = \frac{\int_{-\infty}^{\infty} \int_{-\infty}^{\infty} x\mu^2 dxdy}{\int_{-\infty}^{\infty} \int_{-\infty}^{\infty} \mu^2 dxdy} \tag{5.8}
$$

$$
y_0 = \frac{\int_{-\infty}^{\infty} \int_{-\infty}^{\infty} y\mu dxdy}{\int_{-\infty}^{\infty} \int_{-\infty}^{\infty} \mu dxdy} = \frac{\int_{-\infty}^{\infty} \int_{-\infty}^{\infty} y\mu^2 dxdy}{\int_{-\infty}^{\infty} \int_{-\infty}^{\infty} \mu^2 dxdy} \tag{5.9}
$$

$$
h' = \frac{\int_{-\infty}^{\infty} \int_{-\infty}^{\infty} \mu dxdy}{\sqrt{3\pi \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} \mu^2 dxdy}} \tag{5.10}
$$

$$
m' = \frac{\int_{-\infty}^{\infty} \int_{-\infty}^{\infty} \mu dxdy}{9\pi^2 C \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} \mu^2 dxdy} \tag{5.11}
$$

where x_0, y_0 is the source location, h' is the estimated depth to the source, m' is the estimated total magnetisation, $C = \mu_0/4\pi$ and $\mu_0$is the permeability of free space. Clark (2014) reports that equations (5.8) to (5.11) are exact for a point dipole source and are

very accurate for equidimensional sources buried at moderate depths and compact sources of arbitrary shape, where the dipole contribution dominates the anomaly.

Because finite integrals have issues with the tails of anomalies (such as noise and interfering anomalies), Clark (2014) also developed a series of corrections centred on (x_0, y_0) and using polar coordinates to accommodate these issues. These are

$$h = h' \sqrt{\frac{2}{1 - 3(R/h')^2 + \sqrt{1 - 2(R/h')^2 - 3(R/h')^4}}} \tag{5.12}$$

$$m = m'(1 + 3(R/h)^2 + 3(R/h)^4) \tag{5.13}$$

where R is the radius of a disc around the source location. Clark (2014) points out that a disc can be used in the place of a square of equivalent area since the difference between the two is at most 1% over a wide range of R/h. Clark (2014) then presents the derived version of (M_x, M_y, M_z) as follows:

$$\begin{aligned} M_x &= \frac{h}{2\pi C} \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} (x - x_0)\, \lambda_2 dx dy \\ &= \frac{\frac{h}{2\pi C} \int_0^{2\pi} \int_0^R (x - x_0)\, \lambda_2 \rho d\rho d\theta}{1 - \frac{1 + 3R^2/2h^2}{(1 + R^2/h^2)^{3/2}}} \end{aligned} \tag{5.14}$$

$$\begin{aligned} M_y &= \frac{h}{2\pi C} \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} (y - y_0)\, \lambda_2 dx dy \\ &= \frac{\frac{h}{2\pi C} \int_0^{2\pi} \int_0^R (y - y_0)\, \lambda_2 \rho d\rho d\theta}{1 - \frac{1 + 3R^2/2h^2}{(1 + R^2/h^2)^{3/2}}} \end{aligned} \tag{5.15}$$

$$\begin{aligned} M_z &= \frac{h^2}{2\pi C} \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} \lambda_2 dx dy \\ &= \frac{\frac{h^2}{2\pi C} \int_0^{2\pi} \int_0^R \lambda_2 \rho d\rho d\theta}{1 - \frac{1}{(1 + R^2/h^2)^{3/2}}} \end{aligned} \tag{5.16}$$

where ρ, θ are polar coordinates centred on (x_0, y_0).

5.2 Theory

The ability to detect and calculate remanent parameters directly from magnetic data is extremely appealing. Tensors provide the necessary extra data to not only detect remanence, but also to quantify it to an extent. This chapter proposes applications of tensors to remanence and shows one of the true strengths of magnetic tensors, namely the calculation of remanent parameters from the magnetic tensor field.

While some parameters can be derived in general (such as direction cosines), other parameters (such as magnetisation) are model dependant. In particular, remanence can be detected and calculated in dykes using source distance equations. To do so, the total magnetisation of the field over the source must first be calculated. To solve this, the relationships between direction cosines can be taken advantage of. Remembering that (Blakely, 1995, p.89):

$$\begin{aligned} \boldsymbol{M_t} &= \boldsymbol{M_i} + \boldsymbol{M_r} \\ &= k.\frac{\boldsymbol{B_a}}{400\pi} + \boldsymbol{M_r} \end{aligned} \tag{5.17}$$

$\boldsymbol{B_a}$ is the ambient inducing field, $\boldsymbol{M_i}$ is the induced magnetisation, $\boldsymbol{M_r}$ is the remanent magnetisation and $\boldsymbol{M_t}$ is the total magnetisation. This can be written as:

$$|\boldsymbol{M_t}|.[\alpha_t, \beta_t, \gamma_t] = \frac{k.|\boldsymbol{B_a}|}{400\pi}.[\alpha_i, \beta_i, \gamma_i] + |\boldsymbol{M_r}|.[\alpha_r, \beta_r, \gamma_r] \tag{5.18}$$

Where $\alpha_r, \beta_r, \gamma_r$ are the direction cosines due to the remanent field, $\alpha_i, \beta_i, \gamma_i$ are the direction cosines due to the inducing field and $\alpha_t, \beta_t, \gamma_t$ are the resultant total direction cosines. This can be rearranged to:

$$|\boldsymbol{M_r}|.[\alpha_r, \beta_r, \gamma_r] = |\boldsymbol{M_t}|.[\alpha_t, \beta_t, \gamma_t] - \frac{k.|\boldsymbol{B_a}|}{400\pi}.[\alpha_i, \beta_i, \gamma_i] \tag{5.19}$$

If $\boldsymbol{M_r}$ has inclination and declination M_{inc} and M_{dec}, then (for NED convention):

$$\alpha_r = \cos(M_{inc}) \cdot \cos(M_{dec}) \tag{5.20}$$

$$\beta_r = \cos(M_{inc}) \cdot \sin(M_{dec}) \tag{5.21}$$

$$\gamma_r = \sin(M_{inc}) \tag{5.22}$$

For ENU convention, 90 degrees should be subtracted from the declination.

Rewriting (5.19), it follows that:

$$|\boldsymbol{M_r}|.\cos(M_{inc}) \cdot \cos(M_{dec}) = |\boldsymbol{M_t}|.\alpha_t - \frac{k.|\boldsymbol{B_a}|}{400\pi}.\alpha_i \tag{5.23}$$

$$|\boldsymbol{M_r}|.\cos(M_{inc}) \cdot \sin(M_{dec}) = |\boldsymbol{M_t}|.\beta_t - \frac{k.|\boldsymbol{B_a}|}{400\pi}.\beta_i \tag{5.24}$$

$$|\boldsymbol{M_r}|.\sin(M_{inc}) = |\boldsymbol{M_t}|.\gamma_t - \frac{k.|\boldsymbol{B_a}|}{400\pi}.\gamma_i \tag{5.25}$$

Using (5.23), (5.24), (5.25), M_{inc} and M_{dec} can be solved for:

$$M_{dec} = -2\tan^{-1}\left(\frac{A + \sqrt{A^2 + B^2}}{B}\right) - \pi + azimuth \tag{5.26}$$

$$M_{inc} = -2\tan^{-1}\left(\frac{\sqrt{A^2 + B^2} - \sqrt{\mathrm{A}^2 + \mathrm{B}^2 + \mathrm{C}^2}}{C}\right) \tag{5.27}$$

Where $azimuth$ refers to the angle between the x-direction and north, π is simply to correct for misorientation of the solution, and:

$$A = |\boldsymbol{M_t}|.\alpha_t - \frac{k.|\boldsymbol{B_a}|}{400\pi}.\alpha_i \tag{5.28}$$

$$B = |\boldsymbol{M_t}|.\beta_t - \frac{k.|\boldsymbol{B_a}|}{400\pi}.\beta_i \tag{5.29}$$

$$C = |\boldsymbol{M_t}|.\gamma_t - \frac{k.|\boldsymbol{B_a}|}{400\pi}.\gamma_i \tag{5.30}$$

The unknowns are $|\boldsymbol{M_t}|, \alpha_t, \beta_t, \gamma_t$ and k.

5.2.1 Direction Cosine Calculation

The total field direction cosines ($\alpha_t, \beta_t, \gamma_t$) are the combined remanence and ambient field direction cosines. While they do not give the remanent vector direction in a source independent manner, they can be compared with the ambient field direction cosines. This comparison can show whether remanence exists in a particular area, and give an indication of the general direction of the remanent field vector (although this is imprecise.) These equations can also be used with $|\boldsymbol{M_t}|$ and k to determine the remanent vector direction more accurately, should these last two parameters be known.

To solve for $\alpha_t, \beta_t, \gamma_t$ the relationship between tensors, magnetic field components and the approximate magnetic field is used, namely:

$$B_{tmi} = \alpha B_x + \beta B_y + \gamma B_z \tag{5.31}$$

This relationship is well understood and is independent of source. This implies that:

$$\frac{dB_{tmi}}{dx} = \alpha_t B_{xx} + \beta_t B_{yx} + \gamma_t B_{zx} = B_{tx} \tag{5.32}$$

$$\frac{dB_{tmi}}{dy} = \alpha_t B_{xy} + \beta_t B_{yy} + \gamma_t B_{zy} = B_{ty} \tag{5.33}$$

$$\frac{dB_{tmi}}{dz} = \alpha_t B_{xz} + \beta_t B_{yz} + \gamma_t B_{zz} = B_{tz} \tag{5.34}$$

From equations (5.32), (5.33), (5.34) through simple substitution, the following general equations are derived:

$$\alpha_t = \frac{\mathrm{B_{tx}}\left(\mathrm{B_{yz}^2} - \mathrm{B_{yy}B_{zz}}\right) + \mathrm{B_{ty}}\left(\mathrm{B_{xy}B_{zz}} - \mathrm{B_{xz}B_{yz}}\right) + \mathrm{B_{tz}}\left(\mathrm{B_{xz}B_{yy}} - \mathrm{B_{xy}B_{yz}}\right)}{-\mathrm{B_{xx}B_{yy}B_{zz}} + \mathrm{B_{xx}B_{yz}^2} + \mathrm{B_{xy}^2B_{zz}} - 2\mathrm{B_{xy}B_{xz}B_{yz}} + \mathrm{B_{xz}^2B_{yy}}} \tag{5.35}$$

$$\beta_t = \frac{\mathrm{B_{tx}}\left(\mathrm{B_{xz}B_{yz}} - \mathrm{B_{xy}B_{zz}}\right) + \mathrm{B_{ty}}\left(\mathrm{B_{xx}B_{zz}} - \mathrm{B_{xz}^2}\right) + \mathrm{B_{tz}}\left(\mathrm{B_{xy}B_{xz}} - \mathrm{B_{xx}B_{yz}}\right)}{\mathrm{B_{xx}B_{yy}B_{zz}} - \mathrm{B_{xx}B_{yz}^2} - \mathrm{B_{xy}^2B_{zz}} + 2\mathrm{B_{xy}B_{xz}B_{yz}} - \mathrm{B_{xz}^2B_{yy}}} \tag{5.36}$$

$$\gamma_t = \frac{B_{tx}\left(B_{xy}B_{yz} - B_{xz}B_{yy}\right) + B_{ty}\left(B_{xy}B_{xz} - B_{xx}B_{yz}\right) + B_{tz}\left(B_{xx}B_{yy} - B_{xy}^2\right)}{B_{xx}B_{yy}B_{zz} - B_{xx}B_{yz}^2 - B_{xy}^2B_{zz} + 2B_{xy}B_{xz}B_{yz} - B_{xz}^2B_{yy}} \quad (5.37)$$

An alternate derivation is provided in the event that calculating the vertical derivative of the TMI is not desired, and using equations (5.31), (5.32) and (5.33).

$$\alpha_t = \frac{B_{xy}B_{yz}B_{tmi} - B_{xy}B_zB_{ty} + B_{xz}B_yB_{ty} - B_{xz}B_{yy}B_{tmi} - B_yB_{yz}B_{tx} + B_{yy}B_zB_{tx}}{B_xB_{xy}B_{yz} - B_xB_{xz}B_{yy} - B_{xx}B_yB_{yz} + B_{xx}B_{yy}B_z - B_{xy}^2B_z + B_{xy}B_{xz}B_y} \quad (5.38)$$

$$\beta_t = \frac{-B_xB_{xz}B_{ty} + B_xB_{yz}B_{tx} - B_{xx}B_{yz}B_{tmi} + B_{xx}B_zB_{ty} + B_{xy}B_{xz}B_{tmi} - B_{xy}B_zB_{tx}}{B_xB_{xy}B_{yz} - B_xB_{xz}B_{yy} - B_{xx}B_yB_{yz} + B_{xx}B_{yy}B_z - B_{xy}^2B_z + B_{xy}B_{xz}B_y} \quad (5.39)$$

$$\gamma_t = \frac{B_xB_{xy}B_{ty} - B_xB_{yy}B_{tx} - B_{xx}B_yB_{ty} + B_{xx}B_{yy}B_{tmi} - B_{xy}^2B_{tmi} + B_{xy}B_yB_{tx}}{B_xB_{xy}B_{yz} - B_xB_{xz}B_{yy} - B_{xx}B_yB_{yz} + B_{xx}B_{yy}B_z - B_{xy}^2B_z + B_{xy}B_{xz}B_y} \quad (5.40)$$

A more accurate version of these equations can be derived using the same process from full total magnetic intensity, which is defined as:

$$B_{tmi} = \sqrt{(B_x + \alpha B_a)^2 + \left(B_y + \beta B_a\right)^2 + (B_z + \gamma B_a)^2} - B_a \quad (5.41)$$

where B_a is the ambient magnetic field. Calculating derivatives of this, we get:

$$\frac{dB_{tmi}}{dx} = \frac{B_{xx}(B_x + \alpha B_a) + B_{yx}\left(B_y + \beta B_a\right) + B_{zx}(B_z + \gamma B_a)}{B_{tmi} + B_a} \quad (5.42)$$

$$\frac{dB_{tmi}}{dy} = \frac{B_{xy}(B_x + \alpha B_a) + B_{yy}\left(B_y + \beta B_a\right) + B_{zy}(B_z + \gamma B_a)}{B_{tmi} + B_a} \quad (5.43)$$

$$\frac{dB_{tmi}}{dz} = \frac{B_{xz}(B_x + \alpha B_a) + B_{yz}\left(B_y + \beta B_a\right) + B_{zz}(B_z + \gamma B_a)}{B_{tmi} + B_a} \quad (5.44)$$

Once again, we can solve for α, β, γ from equations (5.42), (5.43) and (5.44). We get:

$$\begin{aligned}\alpha = \big(&-B_a B_{xx} B_{xy} B_{ty} + B_a B_{xx} B_{yy} B_{tx} - B_a B_{xy} B_{yy} B_{ty} - B_a B_{xy} B_{yz} B_{tz} \\ &+ B_a B_{xz} B_{yy} B_{tz} - B_a B_{xz} B_{yz} B_{ty} + B_a B_{yy}^2 B_{tx} + B_a B_{yz}^2 B_{tx} \\ &- B_x B_{xx}^2 B_{yy} + B_x B_{xx} B_{xy}^2 - B_x B_{xx} B_{yy}^2 - B_x B_{xx} B_{yz}^2 + B_x B_{xy}^2 B_{yy} \\ &+ 2\, B_x B_{xy} B_{xz} B_{yz} - B_x B_{xz}^2 B_{yy} - B_{xx} B_{xy} B_{tmi}\, B_{ty} \\ &+ B_{xx} B_{yy} B_{tmi}\, B_{tx} - B_{xy} B_{yy} B_{tmi}\, B_{ty} - B_{xy} B_{yz} B_{tmi}\, B_{tz} \\ &+ B_{xz} B_{yy} B_{tmi}\, B_{tz} - B_{xz} B_{yz} B_{tmi}\, B_{ty} + B_{yy}^2 B_{tmi}\, B_{tx} \\ &+ B_{yz}^2 B_{tmi}\, B_{tx}\big) \\ &\div \Big(B_a\big(B_{xx}^2 B_{yy} - B_{xx} B_{xy}^2 + B_{xx} B_{yy}^2 + B_{xx} B_{yz}^2 - B_{xy}^2 B_{yy} \\ &- 2\, B_{xy} B_{xz} B_{yz} + B_{xz}^2 B_{yy}\big)\Big)\end{aligned} \tag{5.45}$$

$$\begin{aligned}\beta = \big(&B_a B_{xx}^2 B_{ty} - B_a B_{xx} B_{xy} B_{tx} + B_a B_{xx} B_{yy} B_{ty} + B_a B_{xx} B_{yz} B_{tz} - B_a B_{xy} B_{xz} B_{tz} \\ &- B_a B_{xy} B_{yy} B_{tx} + B_a B_{xz}^2 B_{ty} - B_a B_{xz} B_{yz} B_{tx} - B_{xx}^2 B_y B_{yy} \\ &+ B_{xx}^2 B_{tmi}\, B_{ty} + B_{xx} B_{xy}^2 B_y - B_{xx} B_{xy} B_{tmi}\, B_{tx} - B_{xx} B_y B_{yy}^2 \\ &- B_{xx} B_y B_{yz}^2 + B_{xx} B_{yy} B_{tmi}\, B_{ty} + B_{xx} B_{yz} B_{tmi}\, B_{tz} + B_{xy}^2 B_y B_{yy} \\ &+ 2\, B_{xy} B_{xz} B_y B_{yz} - B_{xy} B_{xz} B_{tmi}\, B_{tz} - B_{xy} B_{yy} B_{tmi}\, B_{tx} \\ &- B_{xz}^2 B_y B_{yy} + B_{xz}^2 B_{tmi}\, B_{ty} - B_{xz} B_{yz} B_{tmi}\, B_{tx}\big) \\ &\div \Big(B_a\big(B_{xx}^2 B_{yy} - B_{xx} B_{xy}^2 + B_{xx} B_{yy}^2 + B_{xx} B_{yz}^2 - B_{xy}^2 B_{yy} \\ &- 2\, B_{xy} B_{xz} B_{yz} + B_{xz}^2 B_{yy}\big)\Big)\end{aligned} \tag{5.46}$$

$$\begin{aligned}\gamma = \big(&-B_a B_{xx} B_{yy} B_{tz} + B_a B_{xx} B_{yz} B_{ty} + B_a B_{xy}^2 B_{tz} - B_a B_{xy} B_{xz} B_{ty} - B_a B_{xy} B_{yz} B_{tx} \\ &+ B_a B_{xz} B_{yy} B_{tx} - B_{xx}^2 B_{yy} B_z + B_{xx} B_{xy}^2 B_z - B_{xx} B_{yy}^2 B_z \\ &- B_{xx} B_{yy} B_{tmi}\, B_{tz} - B_{xx} B_{yz}^2 B_z + B_{xx} B_{yz} B_{tmi}\, B_{ty} + B_{xy}^2 B_{yy} B_z \\ &+ B_{xy}^2 B_{tmi}\, B_{tz} + 2\, B_{xy} B_{xz} B_{yz} B_z - B_{xy} B_{xz} B_{tmi}\, B_{ty} \\ &- B_{xy} B_{yz} B_{tmi}\, B_{tx} - B_{xz}^2 B_{yy} B_z + B_{xz} B_{yy} B_{tmi}\, B_{tx}\big) \\ &\div \Big(B_a\big(B_{xx}^2 B_{yy} - B_{xx} B_{xy}^2 + B_{xx} B_{yy}^2 + B_{xx} B_{yz}^2 - B_{xy}^2 B_{yy} \\ &- 2\, B_{xy} B_{xz} B_{yz} + B_{xz}^2 B_{yy}\big)\Big)\end{aligned} \tag{5.47}$$

Notice that to keep these equations manageable, a value for B_{tmi} must be measured and used. The choice of which equations to use really depends on which datasets have been measured. The last three equations are preferred, since they give precise results not affected by field approximations.

Results from these direction cosine equations can be compared with the ambient field in the form of quiver plots. An example of this is shown in Figure 57. For visual representation, each arrow is located at its magnetic value in TMI. The arrows themselves represent the x and y component of the field in the inclination plane or the declination plane. So for declinations, these components are (in NED):

$$Declination\ Arrow\ X\ Component = \frac{\beta}{\sqrt{\alpha^2 + \beta^2}} \tag{5.48}$$

$$Declination\ Arrow\ Y\ Component = \frac{\alpha}{\sqrt{\alpha^2 + \beta^2}} \tag{5.49}$$

And for inclinations, these are:

$$Inclination\ Arrow\ X\ Component = \frac{\sqrt{\alpha^2 + \beta^2}}{\sqrt{\alpha^2 + \beta^2 + \gamma^2}} \tag{5.50}$$

$$Inclination\ Arrow\ Y\ Component = \frac{\gamma}{\sqrt{\alpha^2 + \beta^2 + \gamma^2}} \tag{5.51}$$

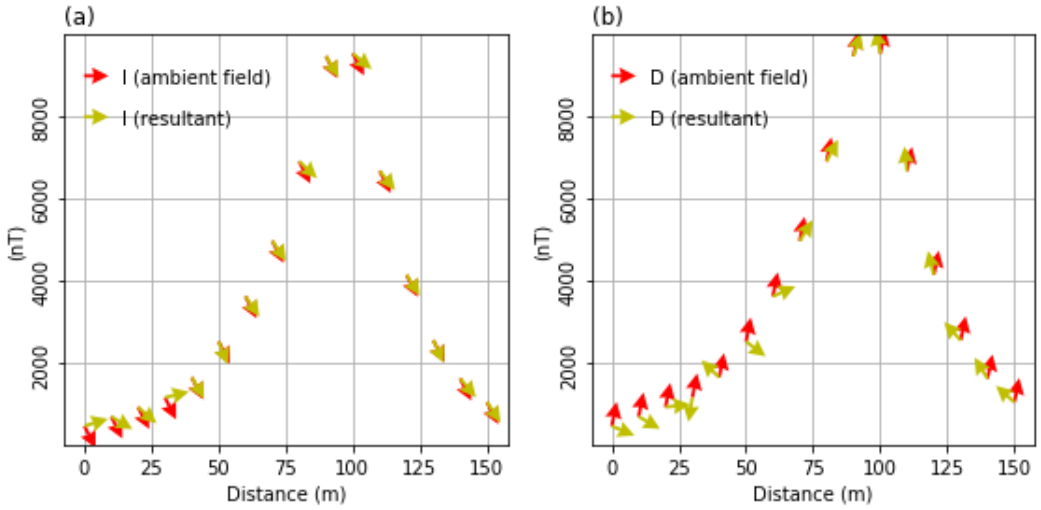

Figure 57 (a) Quiver plot showing a comparison between inclinations from the ambient field and the measured field including remanence. (b) Quiver plot showing a comparison between declination from the ambient field and the measured field including remanence.

These equations are model independent and therein lies their usefulness.

5.2.2 Magnetisation and Susceptibility Estimation

For magnetisation and susceptibility, a model of some sort must be assumed. Source distance equations provide simple models from which to calculate magnetisation in certain instances. With this in mind, k can be estimated for dykes using equations (4.106) or (4.109), if an assumption about the width of the dyke can be made. To calculate for magnetisation due to a thin dyke, $k.F$ in equations (4.104), (4.105), (4.107), (4.108), is replaced with $M_t \cdot \mu_0 \cdot 10^{-9}/4\pi = 100\, M_t$ (to include remanent magnetisation).

Therefore:

$$As_0 = \frac{200.M_t.c.w}{\sqrt{x^2+z^2}} \tag{5.52}$$

$$As_1 = \frac{200.M_t.c.w}{x^2+z^2} \tag{5.53}$$

$$As_{z0} = \frac{200.M_t.\sqrt{c}.w}{\sqrt{x^2+z^2}} \tag{5.54}$$

$$As_{z1} = \frac{200.M_t.\sqrt{c}.w}{x^2+z^2} \tag{5.55}$$

where w is the width of the dyke, x, z is the distance and depth to the dyke respectively, and c is from Table 3. As_0 and $As_1$are the zero and first order analytic signals. As_{z0} and $As_{z1}$are the zero and first order tensor analytic signals in the z direction.

We can just rewrite (5.52), (5.53), (5.54) and (5.55) in terms of M_t.

$$M_t = \frac{As_0\sqrt{x^2+z^2}}{200 \cdot w \cdot c} \tag{5.56}$$

$$M_t = \frac{As_1(x^2+z^2)}{200 \cdot w \cdot c} \tag{5.57}$$

$$M_t = \frac{As_{z0}\sqrt{x^2+z^2}}{200 \cdot w \cdot \sqrt{c}} \tag{5.58}$$

$$M_t = \frac{As_{z1}(x^2+z^2)}{200 \cdot w \cdot \sqrt{c}} \tag{5.59}$$

Alternatively, from (5.52) and (5.53) M_t can be solved for:

$$M_t = \frac{As_0^2}{200 \cdot As_1 \cdot w \cdot c} \tag{5.60}$$

From (5.54) and (5.55) M_t can be solved for:

$$M_t = \frac{As_{z0}^2}{200 \cdot As_{z1} \cdot w \cdot \sqrt{c}} \tag{5.61}$$

Note that $c = 1 - \beta_t^2$ for NED convention, and $c = 1 - \alpha_t^2$ for ENU convention. Alternatively, by squaring (5.54) and dividing by (5.53) we get (for NED):

$$M_t = \frac{As_{z0}^2}{200 \cdot As_1 \cdot w} \tag{5.62}$$

This is the total magnetisation of the dyke. This can then be substituted into equation (5.23), and it follows that:

$$M_r = \frac{M_t \cdot \alpha_t - \frac{k|\boldsymbol{B_a}|}{400\pi} \cdot \alpha_i}{\cos(M_{inc}) \cdot \cos(M_{dec})} = \sqrt{A^2 + B^2 + C^2} \tag{5.63}$$

Where A, B, C are defined in equations (5.28), (5.29) and (5.30). Estimates for k must either be known through paleomagnetic measurement, or by using estimates obtained using equation (4.106) and equation (4.113). These estimates are not ideal since susceptibility estimates using analytic signals include the effect of remanence in them, and would not be accurate indications of true susceptibility. In fact, these susceptibility estimates are just $k = 400\pi M_t / B_a$. This is perfectly fine if there is no remanence but not ideal for remanent calculation.

As a side note, these equations are only useful for actual, measured or modelled tensor datasets with TMI. They will not work with pseudo-tensor derivations (Pedersen, Rasmussen and Dyrelius (1990) and Yin et al.(2016)). This is because pseudo-tensor derivations makes the assumption of constant direction cosines (no remanence) and so no remanence will be detected. This is discussed further in CHAPTER 6.

5.3 Test – Remanence

5.3.1 Tests on a single dyke

A single dyke was modelled to test different scenarios. The dyke has a width of 10 m, and a depth of 20 m. The bottom of the modelled dyke is 3000 m. The grid spacing was 10 m, with a measurement centred over each dyke. Table 5 to Table 7 below show the results from using the derived equations, versus the original values. The values for M_t are calculated using equation (5.61). For the purpose of this test, knowledge of the susceptibility is assumed (0.01 SI)

Table 5 shows the results for the single dyke with no remanence. Table 6 and Table 7 show the same situation with remanence. In all three cases, the direction cosines are calculated perfectly, while M_t tends to be underestimated. Values for M_r, M_{inc}, M_{dec} are dependent on accurate values of M_t and deteriorate accordingly. However, even in the worst case (Table 7) where remanent direction is vastly different to the ambient field direction, the calculated remanent inclination and declination values are close to the real values. If however, the magnetisation and susceptibility estimates are worse, then results will deteriorate.

Table 5 Summary of results for a dyke of width 10 meters, depth of 20 meters, depth extent of 3000 m, with $B = 28\,000\ nT, H_{inc} = 60°, H_{dec} = -30°, k = 0.01\ SI$ and no remanence

Parameter	Original	Calculated
α_t	-0.25	-0.2504
β_t	-0.433	-0.4331
γ_t	0.866	0.8658
M_{inc}	N/A	N/A
M_{dec}	N/A	N/A
M_t	0.2228	0.2013
M_r	N/A	N/A

Table 6 Summary of results for a dyke of width 10 meters, depth of 20 meters, depth extent of 3000 m, with $B = 28\,000\ nT, H_{inc} = 60°, H_{dec} = -30°, k = 0.01\ SI$, $M_r = 0.323\ A/m$, $M_{inc} = 50°$ and $M_{dec} = -20°$

Parameter	Original	Calculated
α_t	-0.2332	-0.2344
β_t	-0.5368	-0.5372
γ_t	0.8108	0.8102
M_{inc}	50	49.04
M_{dec}	-20	-19.52
M_t	0.5432	0.4887
M_r	0.323	0.2688

Table 7 Summary of results for a dyke of width 10 meters, depth of 20 meters, depth extent of 3000 m, with, $\boldsymbol{B = 28\,000\ \text{nT}, H_{inc} = 60°, H_{dec} = -30°, k = 0.01\ SI, M_r = 0.323\ A/m, M_{inc} = -20°}$ and $\boldsymbol{M_{dec} = 40°}$

Parameter	Original	Calculated
α_t	0.3801	0.3796
β_t	-0.8972	-0.8976
γ_t	0.2249	0.2243
M_{inc}	-20	-22.01
M_{dec}	40	41.10
M_t	0.3667	0.3467
M_r	0.323	0.3074

A Gaussian noise was added to each of the tensor datasets used in the calculation of Table 8. The standard deviation of the noise added to each dataset was equal to 1.0% of the maximum respective dataset amplitude. The results are shown in Table 8. Noise affects the results, and the direction cosine calculation is sensitive to it. The calculation of β_t in this case has a value greater than 1, which would not be possible if it were not for noise. Outliers like this can be an indicator to the quality of the data being collected, since this sort of value would not happen with noise free data. The total magnetisation calculation, by contrast, is less sensitive to the noise.

Table 8 Summary of results for a dyke of width 10 meters, depth of 20 meters, depth extent of 3000 m, with, $\boldsymbol{B = 28\,000\ \text{nT}, H_{inc} = 60°, H_{dec} = -30°, k = 0.01\ SI, M_r = 0.323\ A/m, M_{inc} = -20°}$ and $\boldsymbol{M_{dec} = 40°}$. A Gaussian noise was added to each input dataset, with standard deviation equal to 1.0% of the respective dataset amplitude.

Parameter	Original	Calculated
α_t	0.3801	0.3611
β_t	-0.8972	-2.525
γ_t	0.2249	0.2086
M_{inc}	-20	-8.98
M_{dec}	40	13.22
M_t	0.3667	0.3373
M_r	0.323	0.7853

Table 9 shows the effect of decreasing the remanent magnetisation in the dyke, and hence decreasing the Q-ratio. As the amount of remanence decreases, the effect of the error in the total magnetisation estimate increases, and the result is a decrease in accuracy for inclination and declination estimation. High Q-ratios implying high remanence can offset some of the inaccuracy in the magnetisation calculation.

Table 9 Indication of errors in technique through a range of Q-ratios. The results represent a dyke of width 10 meters, depth of 20 meters, depth extent of 3000 m, with, $\boldsymbol{B} = \mathbf{28\,000\ nT}, \boldsymbol{H_{inc}} = \mathbf{60°}, \boldsymbol{H_{dec}} = \mathbf{-30°}, \boldsymbol{k} = \mathbf{0.01}\ \boldsymbol{SI}, \boldsymbol{M_{inc}} = \mathbf{-20°}$ and $\boldsymbol{M_{dec}} = \mathbf{40°}$. Values for $\boldsymbol{M_r}$ are changed in the test.

M_r (A/m)	Q-Ratio	Inclination	Declination
0.323	2.1	-21.97	41.07
0.269	1.46	-23.25	41.84
0.215	0.934	-25.25	43.07
0.1615	0.525	-28.40	45.11
0.108	0.233	-33.48	48.69
0.081	0.131	-37.35	51.77

This is an important limitation, since inaccurate magnetisation results will also result in inaccurate remanent field calculations.

5.3.2 Tests on two dykes with no remanence

A test is necessary to see if the calculation holds up when more than one dyke in present. Two north-south dykes were modelled with the following parameters. B_a = 28,000 nT, $H_{inc} = 60°$, $H_{dec} = -30°$, $k = 0.01\ SI$. Each dyke has a width of 10 m, and a depth of 20 m and length of 400 m. The bottom of the modelled dykes is 3000 m. The grid spacing was 10 m, with a measurement centred over each dyke. The area is 400 m by 400 m, implying a 40x40 calculated grid. Figure 58 shows the forward model, and Figure 59 shows the tensor data calculated from that model. Figure 60 to Figure 64 are all calculated using equations (5.35), (5.36), (5.37), (5.62) and (5.63) respectively. Figure 60, Figure 61 and Figure 62 all show constant values, which is what would be expected since neither dyke has remanence. As a side note, only dykes were modelled because of the limitation of the magnetisation calculation to dykes. Table 10 shows the results of the calculations for both dykes in this case. Discrepancies in the magnetisation values are due to the additive effect of the fields from the two dykes on each other.

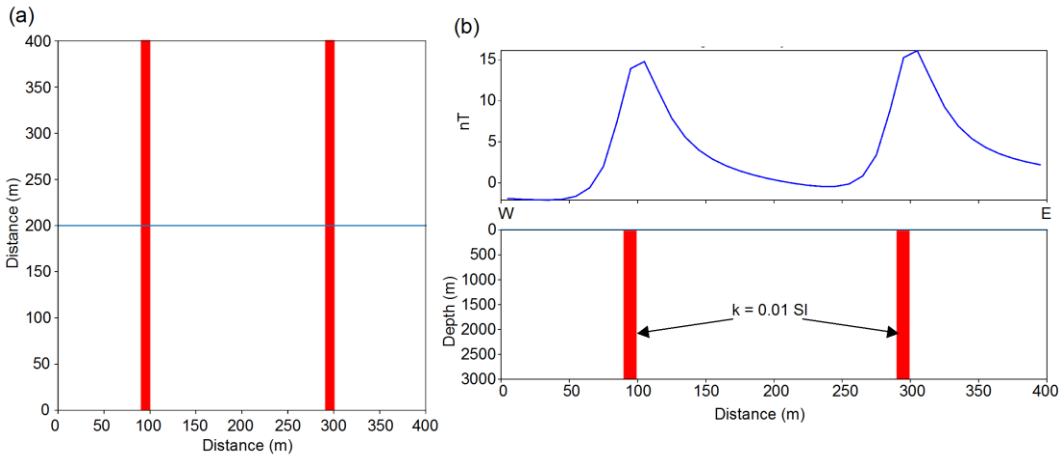

Figure 58 Two dykes modelled with the following parameters. B_a = 28,000 nT, $H_{inc} = 60°$, $H_{dec} = -30°$, $k = 0.01\ SI$. Each dyke has a width of 10 m, a depth of 20 m and length of 400 m. The bottom of the modelled dykes is 3000 m. The grid spacing was 10 m. (a) Top view of the model. (b) Side view of the model with calculated TMI, at the location of the blue line in (a)

Table 10 Summary of results for two dykes of width 10 meters, depth of 20 meters, depth extent of 3000 m, with $B = 28\,000\ nT, H_{inc} = 60°, H_{dec} = -30°, k = 0.01\ SI$ and no remanence.

Parameter	Left Dyke		Right Dyke	
	Original	Calculated	Original	Calculated
α_t	-0.25	-0.2500	-0.25	-0.2499
β_t	-0.433	-0.4330	-0.433	-0.4331
γ_t	0.866	0.8660	0.866	0.8661
M_{inc}	N/A	N/A	N/A	N/A
M_{dec}	N/A	N/A	N/A	N/A
M_t	0.2228	0.2138	0.2228	0.2295
M_r	N/A	N/A	N/A	N/A

Figure 63 shows the total magnetisation calculated for two dykes. The results given on this image can also be compared with the results in Table 5 which were for a single dyke. The calculated value for total magnetisation lies values between 0.21 A/m and 0.23 A/m, which is close to the true value of $M_t = 0.2228$ A/m. Note that for magnetisation and remanent inclination and declination, only the results above the dykes are valid. This is due to the depth portion of the calculation which is only correct over the dyke. The off dyke asymmetry in the magnetisation calculations is an artefact caused by the depth portion of the calculation in equation (5.61), given by the ratio of analytic signals, changing as the observation point moves away from the dyke. In contrast, Figure 65 shows the same calculation using equation (5.59), which has a constant depth for the

dyke. Consequently the result is far more symmetrical. However, the depth to the dyke must be known in this case. The reality is that the symmetry of the anomalies is of no consequence (other than for visual effect), since the positions of the dyke should already be known either from the analytic signal itself or standard source depth calculations.

Figure 64 shows the calculated remanent magnetisation. Since this calculation uses the total magnetisation values for a dyke, and since that calculation is only valid above a dyke, only the areas above the dykes are valid. As can be seen, the values are low (blue) over the dykes, indicating no remanent magnetisation is present.

The interference pattern on the edge of Figure 63 and Figure 64 is a side effect of the calculation of As_{z0} or As_0 , which have edge effects as a result of FFT generated Hilbert transforms. This effect is amplified in (5.61) and (5.62), which takes the square of the analytic signal. The pattern is not present when using equation (5.59), which does not square the analytic signal.

Figure 65 and Figure 66 show the total magnetisation and remanence calculated using equation (5.59). The results are better than in the case of (5.61). The magnetisation estimates are close to the true value of 0.2228 A.m. Anomalies are far more symmetric, leaving little doubt as to the location of the dyke. The symmetry of the two anomalies is a result of the fact that this equation does not divide one analytic signal by another, but rather uses a constant depth estimate, implying that the depth to the dyke must be known in order to use this equation. This is not necessarily a disadvantage, since the depth estimate can come from an outside source, allowing for better depth estimates. In the case of (5.61); one analytic signal is divided by another to automatically calculate this depth, but with the consequence that the depth estimates will vary away from the dyke, leading to a non-symmetric response. The advantage of using equations like (5.61) is that should the dykes have different depths, it will automatically account for this.

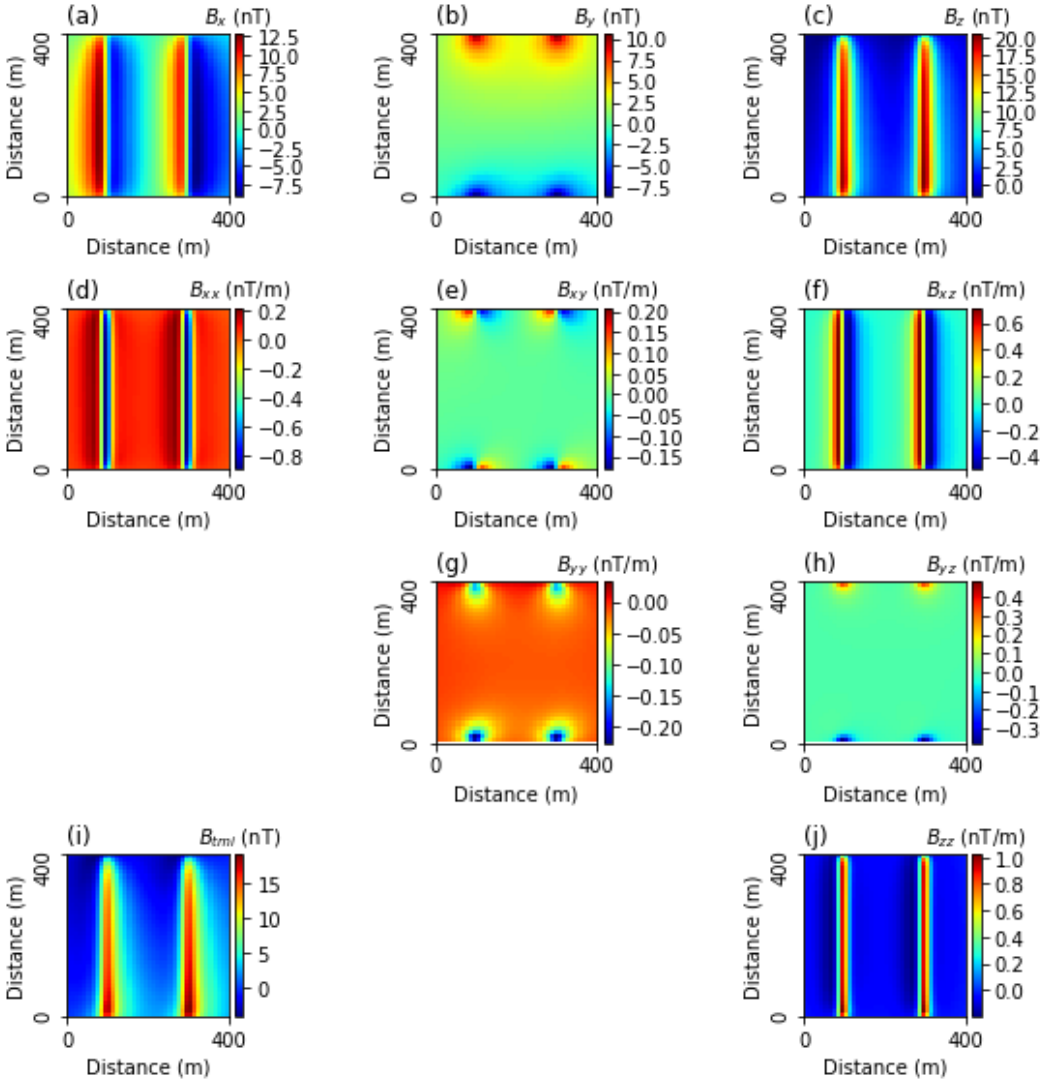

Figure 59 Magnetic field and components. a) to c) show the three components of the magnetic field - B_x, B_y, B_z. d), e), f), g), h) and j) show the tensor components $B_{xx}, B_{xy}, B_{xz}, B_{yy}, B_{yz}, B_{zz}$ respectively. i) Total magnetic intensity of the study area. The magnetic field intensity used was 28000 nT, the susceptibility was 0.01 SI, the inclination was 60º, the declination was $-30°$. Each dyke has a width of 10 m, and a depth of 20 m. There is no remanence.

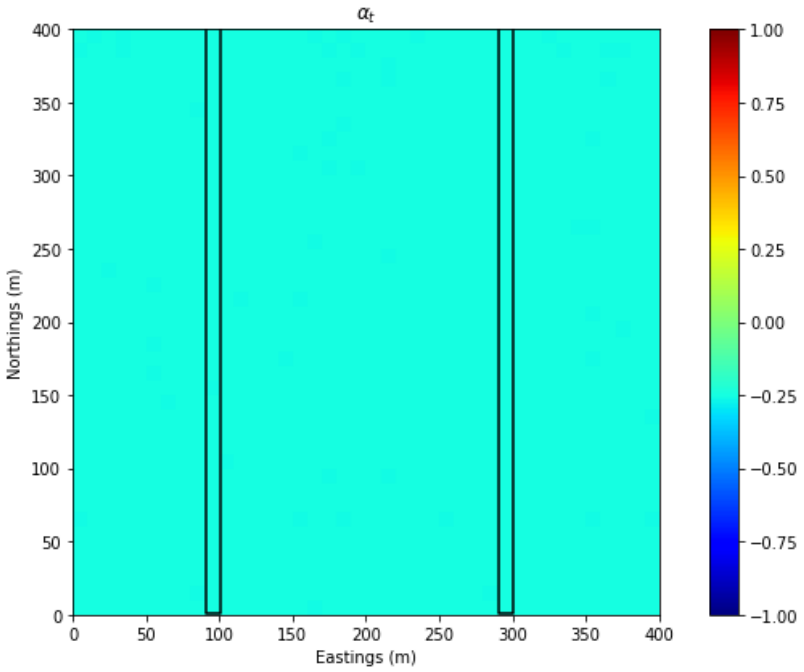

Figure 60 Results of calculating the α_t direction cosine. The value is constant everywhere since the dykes are not remanently magnetised.

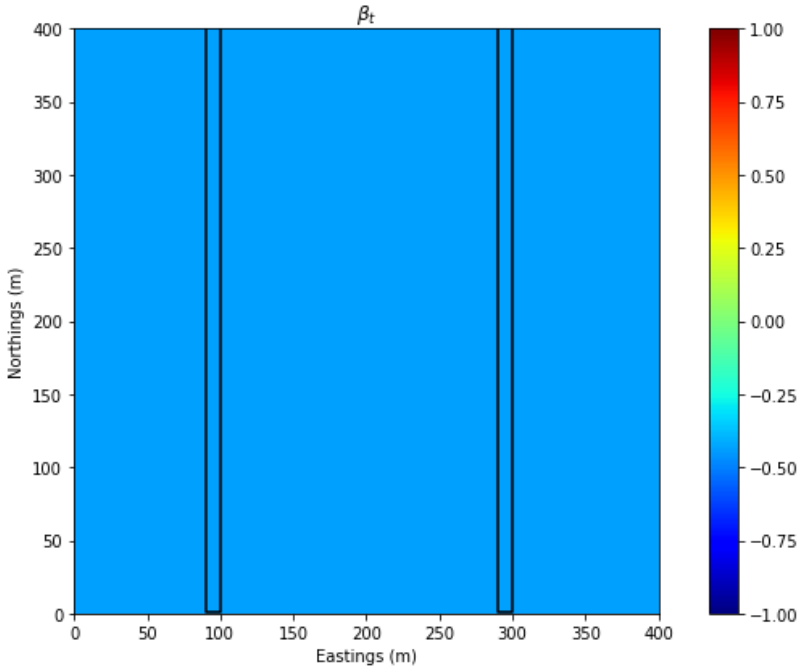

Figure 61 Results of calculating the β_t direction cosine. The value is constant everywhere since the dykes are not remanently magnetised.

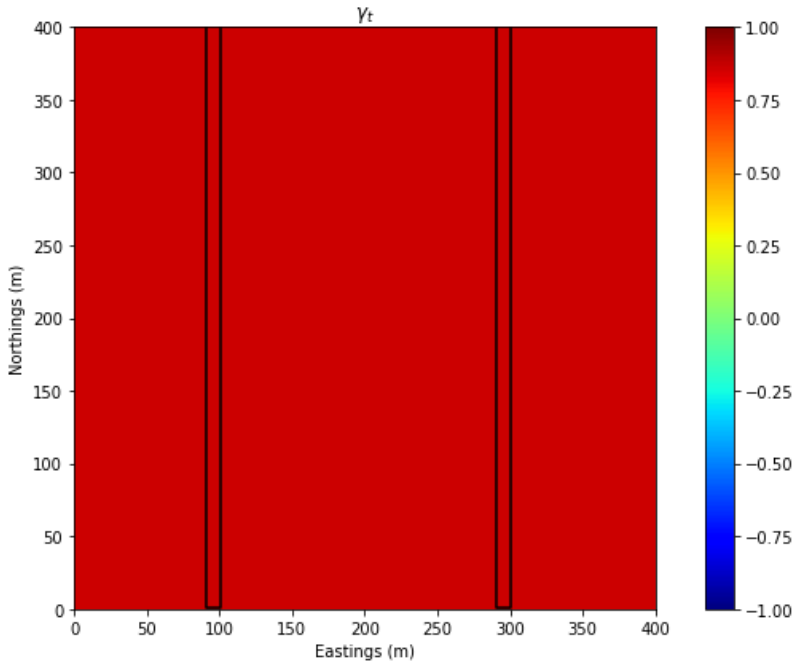

Figure 62 Results of calculating the γ_t direction cosine. The value is constant everywhere since the dykes are not remanently magnetised.

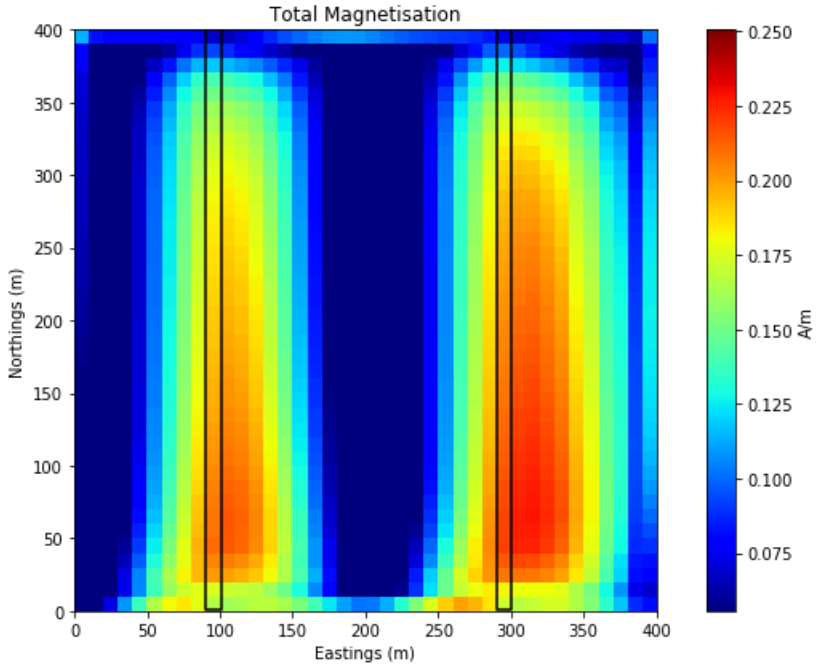

Figure 63 Total magnetisation over two dykes with no remanent magnetisation and using equation (5.61) (outline shown in black). The off dyke asymmetry is caused by the variation in the depth solution component of this calculation resulting from the ratio of the two analytic signals in the calculation of equation (5.61)

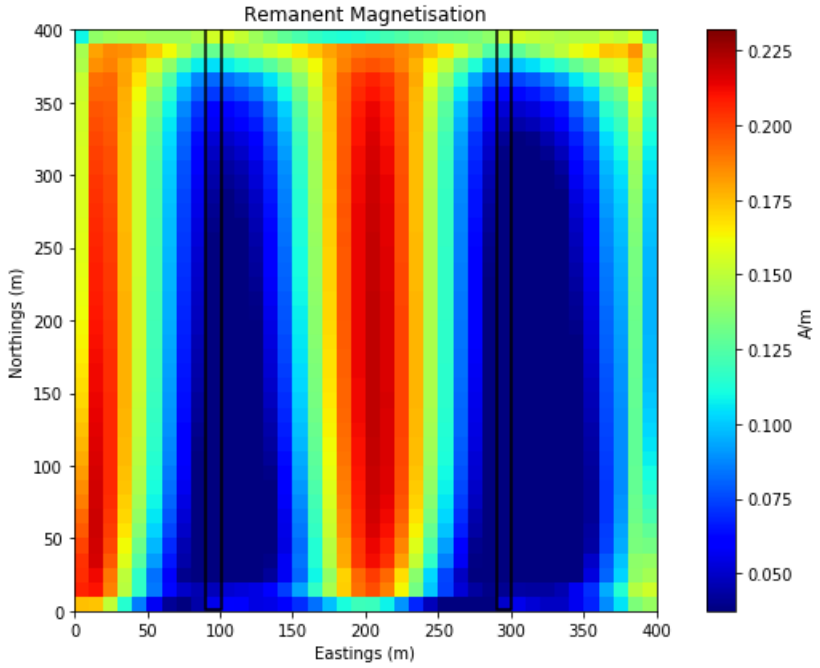

Figure 64 Output from remanent magnetisation calculation over two dykes with no remanent magnetisation and using equations (5.63) and (5.61) (outline shown in black).

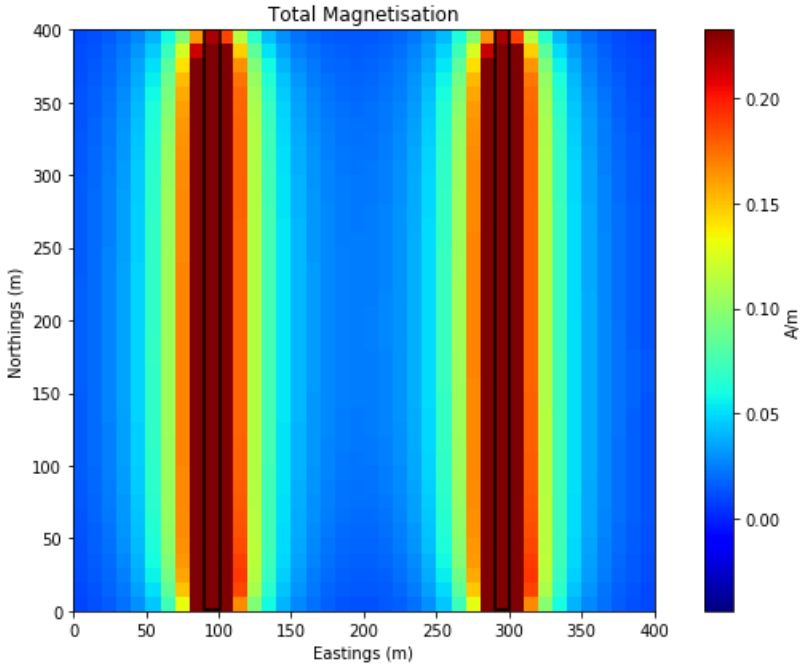

Figure 65 Total magnetisation over two dykes with no remanent magnetisation and using equation (5.59) (outline shown in black)

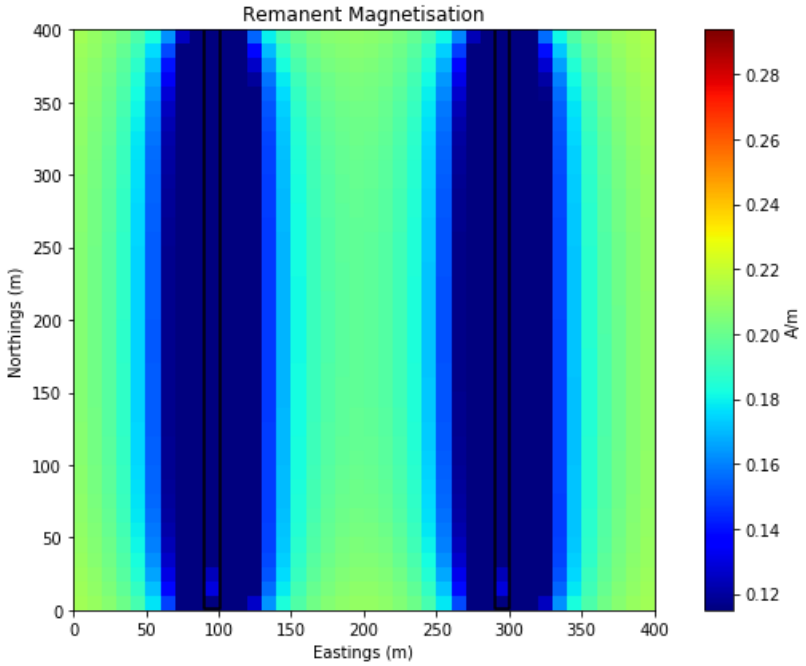

Figure 66 Output from remanent magnetisation calculation over two dykes with no remanent magnetisation using equation (5.63) and (5.59) (outline shown in black)

5.3.3 Tests on two dykes with remanence introduced

To test the effect of remanence in the calculation across multiple dykes, two north-south dykes were modelled with the following parameters. *B* = 28,000 nT, $H_{inc} = 60°$, $H_{dec} = -30°$, $k = 0.01\ SI$. The right dyke has the following remanent values: $M_r = 0.323\ A/m$, $M_{inc} = 50°$ and $M_{dec} = -20°$. Each dyke has a width of 10 m, length of 400 m and a depth of 20 m. The bottom of the modelled dykes is 3000 m. The grid spacing was 10 m, with a measurement centred over each dyke. Note that the eastern dyke has remanence, while the western dyke has no remanence. The area is 400 m by 400 m, implying a 40x40 calculated grid. Figure 67 shows the forward model, and Figure 68 shows the tensor data calculated from that model. Figure 69 to Figure 74 are all calculated using equations (5.35), (5.36), (5.37), (5.62) and (5.63) respectively. Figure 69 to Figure 71 show the direction cosine results. As can be expected, the direction cosines are different between the two dykes. Table 11 shows a summary of the results.

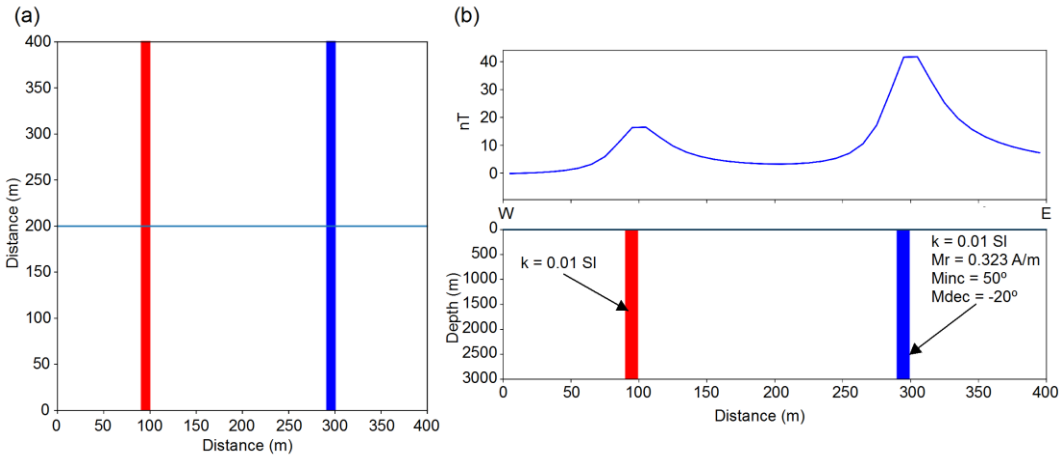

Figure 67 with the following parameters. *B* = 28,000 nT, $H_{inc} = 60°$, $H_{dec} = -30°$, $k = 0.01\ SI$. The right dyke (blue) has the following remanent values: $M_r = 0.323\ A/m$, $M_{inc} = 50°$ and $M_{dec} = -20°$. Each dyke has a width of 10 m, length of 400 m and a depth of 20 m. The depth extent of the modelled dykes is 3000 m. The grid spacing was 10 m. (a) Top view of the model. (b) Side view of the model with calculated TMI, at the location of the blue line in (a)

Two things must be noted. Firstly, the calculation of susceptibility does not give good results over the remanent dyke. This is because the equations from section 4.5.4 do not account for the effect of remanence. Because of this, knowledge of the susceptibility is assumed (0.01 SI) as before and as a result calculated magnetisations and remanent field directions are close to the original values. Without knowing the susceptibility, the remanent parameters will be incorrect. A simple test for the presence of remanence is by comparison of the calculated direction cosines to that of the ambient field. If they are approximately the same, there is no remanence and the susceptibility value can be used in modelling.

The second is that the deviations of the direction cosines from the modelled direction cosines is not strictly speaking wrong. The difference in the values comes from perspective. The original values are from the perspective of the sources and reflect the field direction from the respective source only. The calculated values are from the combination of all sources in the region and reflect what an observer will see. As such they will be different. Nevertheless, as can be seen, directly over the sources they are a reasonable approximation for the source cosines.

Table 11 Summary of results for two dykes of width 10 meters, depth of 20 meters, depth extent of 3000 m, with $B = 28\,000\ nT, H_{inc} = 60°, H_{dec} = -30°, k = 0.01\ SI$ and no remanence. The dyke on the right has remanence with the following parameters: $M_r = 0.323\ A/m, M_{inc} = 50°$ and $M_{dec} = -20°$.

Parameter	Left Dyke (No Remanence)		Right Dyke (Remanence)	
	Original	Calculated	Original	Calculated
α_t	-0.25	-0.2500	-0.2332	-0.2330
β_t	-0.433	-0.459	-0.5368	-0.5250
γ_t	0.866	0.867	0.8108	0.8106
M_{inc}	N/A	N/A	50	51.245
M_{dec}	N/A	N/A	-20	-20.915
M_t	0.2228	0.2054	0.5432	0.5820
M_r	N/A	N/A	0.323	0.3576
k	0.01	0.009	0.01	0.026

The prominent anomaly at the top of Figure 69, Figure 70 and Figure 71 is an important discussion point. The same anomaly results through the use of any of the options for direction cosine calculations and is therefore not an error in the equations. The direction cosine images contain no information about magnetization, and only give information about the field direction at each point. In these figures, the two dykes stop at the top and the bottom of the figure. Along the length of the dykes, their respective equipotential field lines will run parallel to each other, resulting in a smooth transition of direction cosines. At the ends of the dykes (such as in the north of the figures), these lines diverge resulting in a more varied field direction. This, coupled with the field direction from each source, can cause anomalies such as seen to the north. The anomaly is real, and although it is not an indicator of a source beneath it, it is an indicator of a remanent source nearby. The implication of this is that for modelling purposes, the source location must be known beforehand. Fortunately, source location is easily determinable using other methods, such as the analytic signal.

Figure 72 shows the total magnetisation calculated for a dyke. Figure 74 shows remanent magnetisation. Since this calculation uses the total magnetisation values for a dyke, and since that calculation is only valid above a dyke, only the results above the dykes are valid. Edge effects have been masked for presentation purposes.

Figure 73 shows the same calculation using equation (5.59). As before, the result is far more symmetrical. However, the depth to the dyke must be known in order to use this equation.

Figure 74 shows remanent magnetisation using equations (5.63) and (5.61). As can be seen, the values are low (blue) over the non-remanent dyke and high (red) over the remanent dyke. Edge effects have been masked for presentation purposes. In contrast,

Figure 75 shows remanent magnetisation using equations (5.63) and (5.59). The anomalies are far more symmetric, since a constant depth to the dyke is used in equation 5.59.

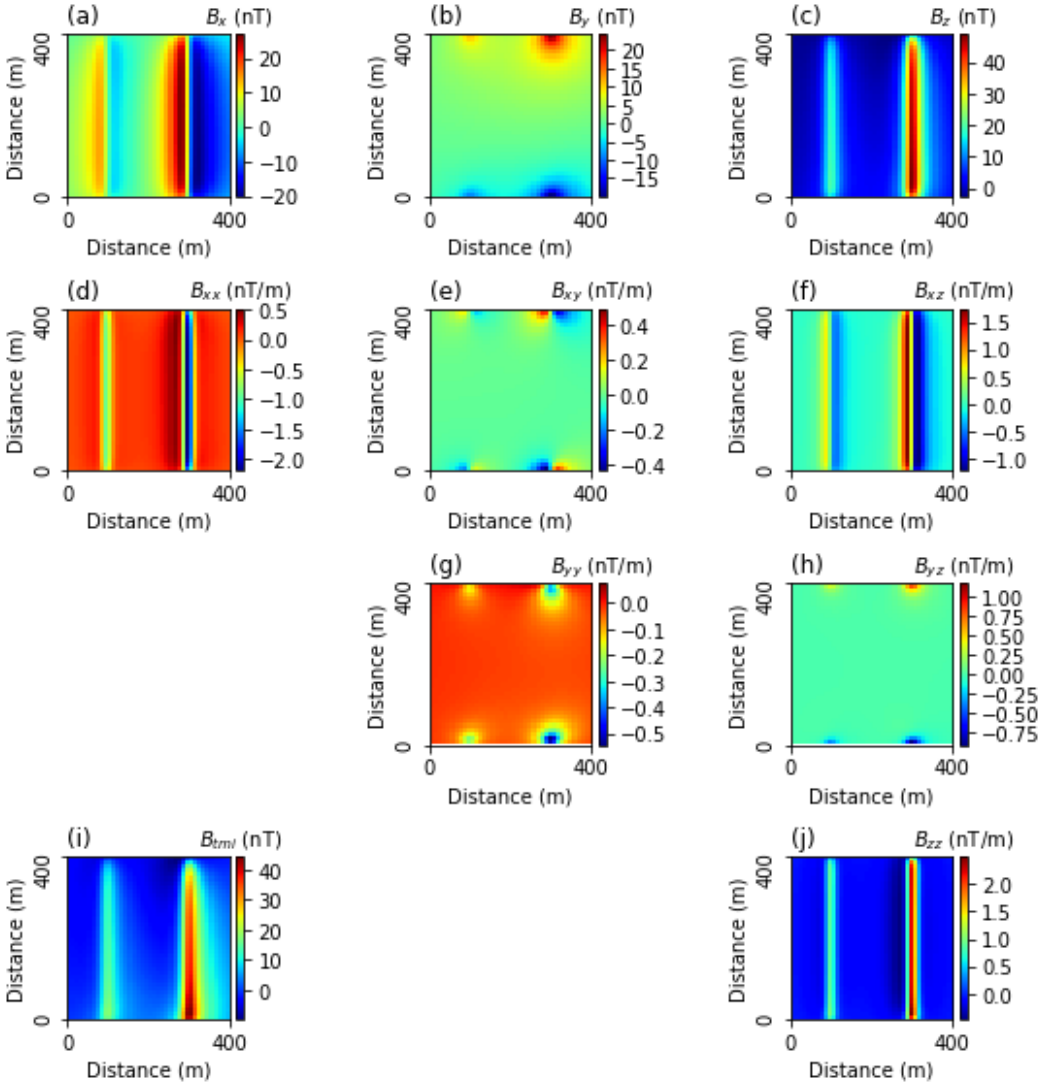

Figure 68 Magnetic field and components. a) to c) show the three components of the magnetic field - B_x, B_y, B_z. d), e), f), g), h) and j) show the tensor components $B_{xx}, B_{xy}, B_{xz}, B_{yy}, B_{yz}, B_{zz}$ respectively. i) Total magnetic intensity of the study area. The magnetic field intensity used was 28000 nT, the susceptibility was 0.01 SI, the inclination was 60º, the declination was $-30°$. The eastern dyke has $M_r = 0.323$ A/m, $M_{inc} = 50°$ and $M_{dec} = -20°$. Each dyke has a width of 10 m, and a depth of 20 m.

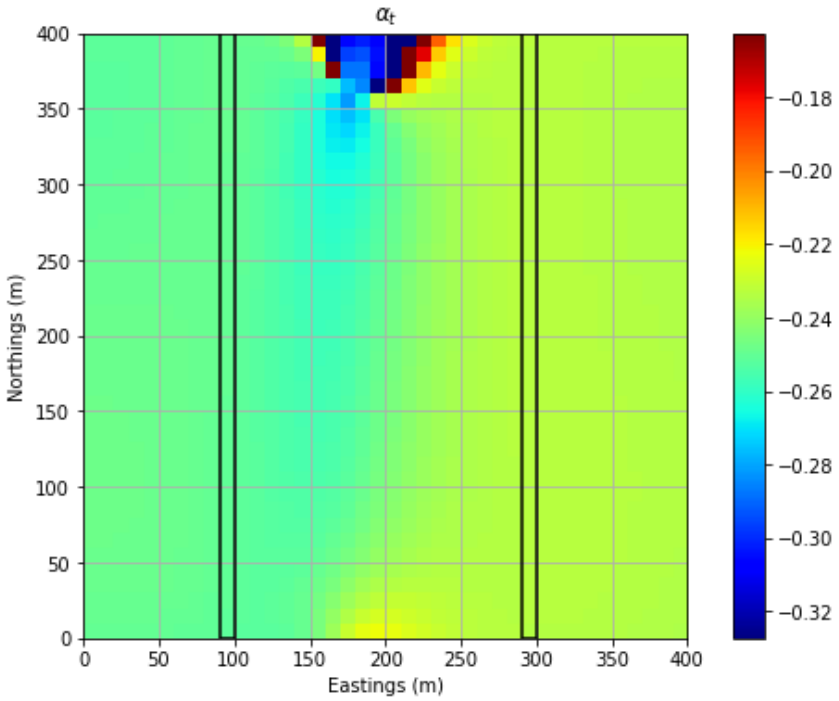

Figure 69 Results of calculating the α_t direction cosine. The dyke on the right has remanent magnetisation.

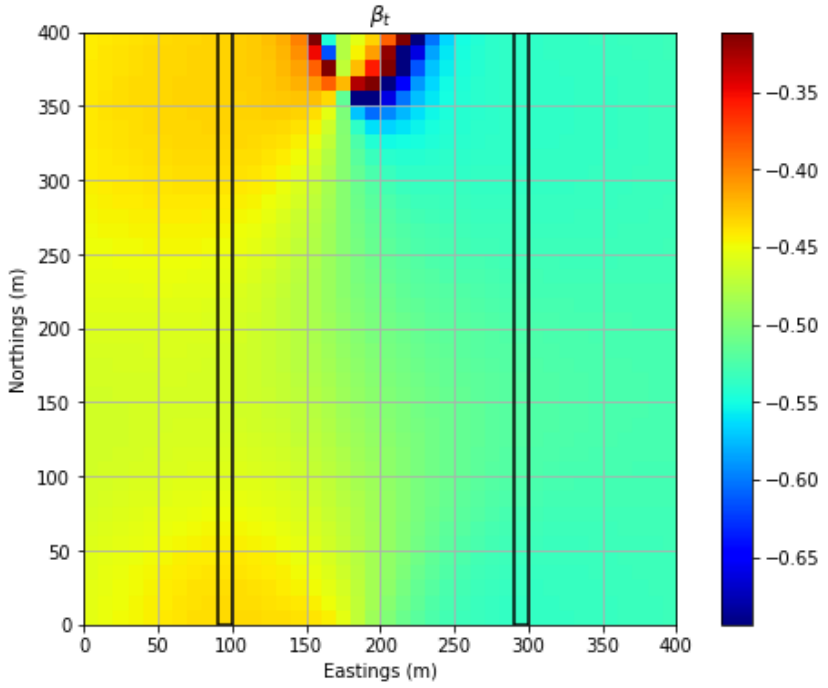

Figure 70 Results of calculating the β_t direction cosine. The dyke on the right has remanent magnetisation.

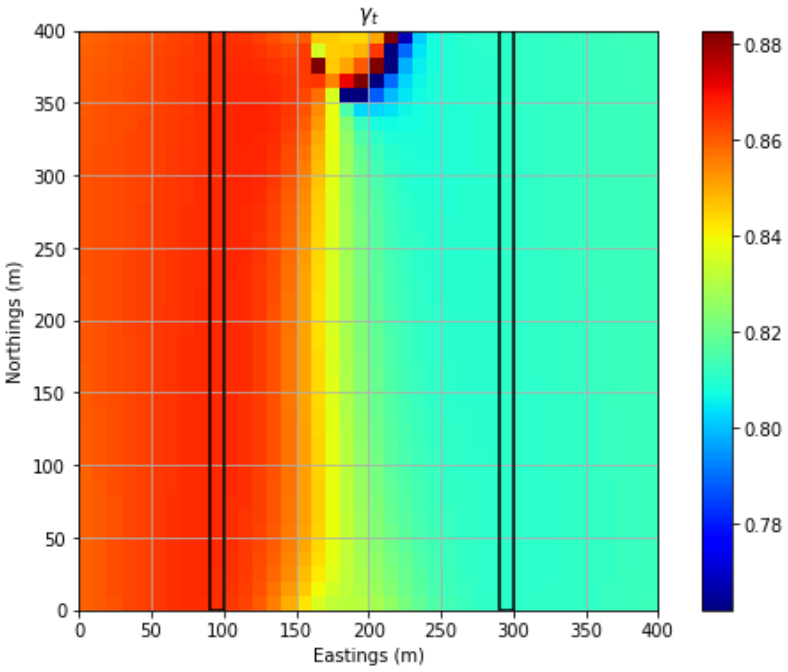

Figure 71 Results of calculating the γ_t direction cosine. The dyke on the right has remanent magnetisation.

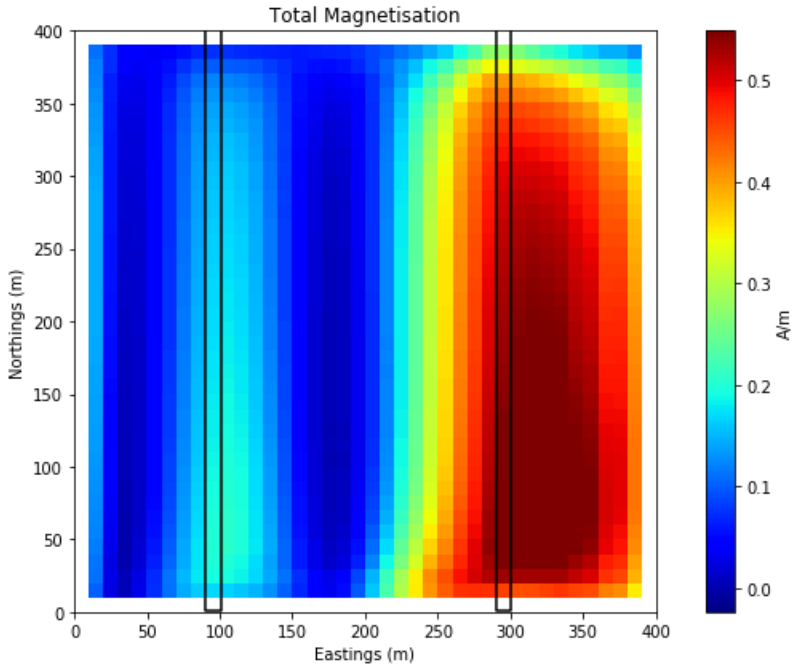

Figure 72 Total magnetisation over two dykes and using equation (5.61) (outline shown in black). The dyke on the right has remanent magnetisation. Edge effects due to the Hilbert transform have been masked.

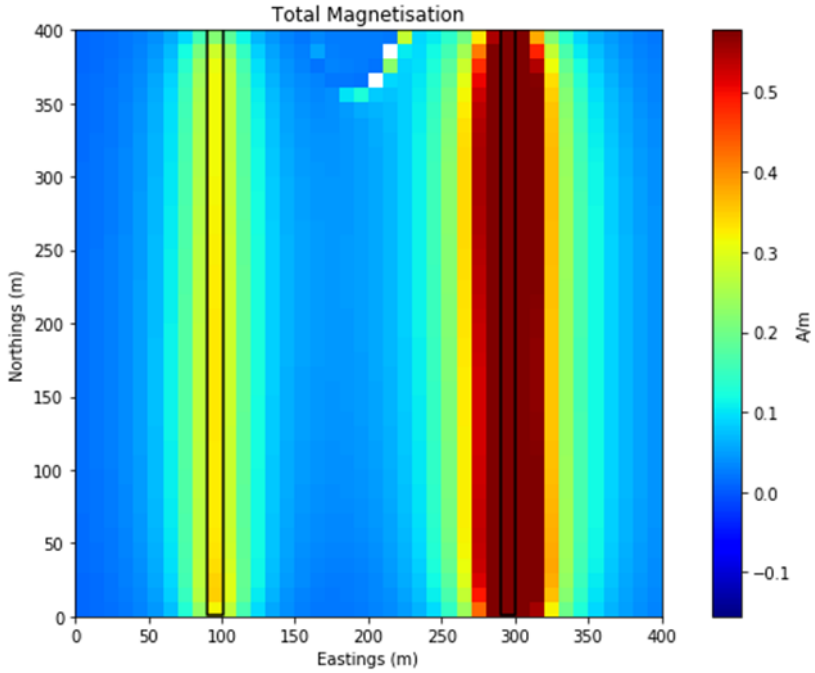

Figure 73 Total magnetisation over two dykes and using equation (5.59) (outline shown in black). The dyke on the right has remanent magnetisation

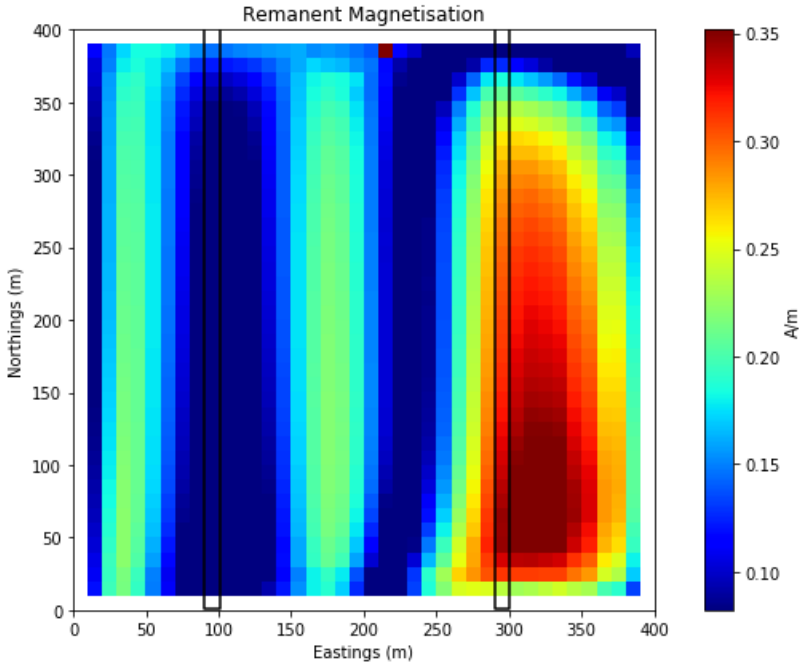

Figure 74 Output from the remanent magnetisation calculation over two dykes and using equations (5.63) and (5.61) (outline shown in black). The dyke on the right has remanent magnetisation. Edge effects due to the Hilbert transform have been masked.

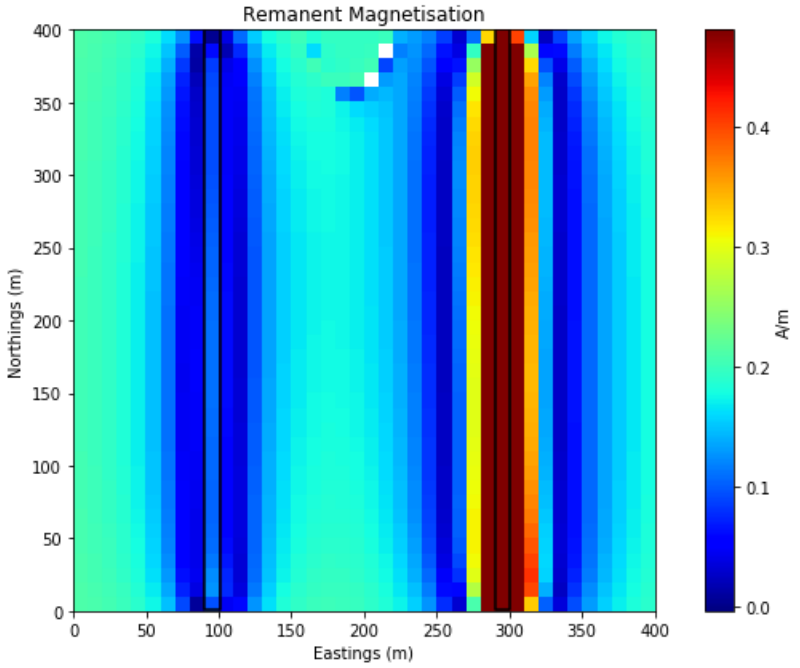

Figure 75 Output from the remanent magnetisation calculation over two dykes using equation (5.63) and (5.59) (outline shown in black). The dyke on the right has remanent magnetisation.

CHAPTER 6 SYNTHESIS OF MODELLING TECHNIQUES APPLIED TO REAL DATA

The synthesis of the techniques discussed in the preceding chapters is straightforward:

1) Use the source detection routine to calculate the depths to sources for the model (section 4.5.1 and 4.5.2) across profiles or grids.
2) Obtain estimates or calculate the susceptibilities of the sources (section 4.5.4) across profiles or grids.
3) If the data is real tensor data, assess the remanence of the bodies (section 5.2) across profiles or grids.
4) Determine connectivity, if any, between identified sources (section 4.5.3) and extract relevant dyke or step solutions from these features (section 6.2)
5) Use the voxel based forward modelling to complete the model, based on tensor data (section 3.6). The number of input lithologies can be determined by using classification on susceptibility solutions, and remanent parameters if available.

Points 1 to 3 calculate values across entire datasets. Obviously, not all cells in a grid refer to dykes or steps, so point 4 uses peak following to extract only the relevant solutions. Point 5 then applies this calculated information to forward modelling. Figure 76 shows the process flow for the case where tensor data is calculated from TMI data (pseudo tensor data). Figure 77 shows the case where measured tensor data is available.

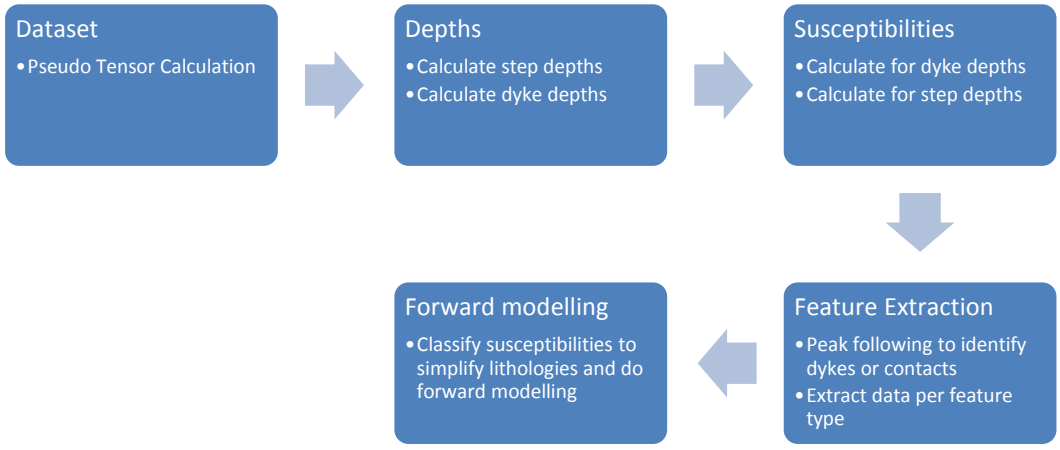

Figure 76 The process flow for calculations involving pseudo tensor calculations

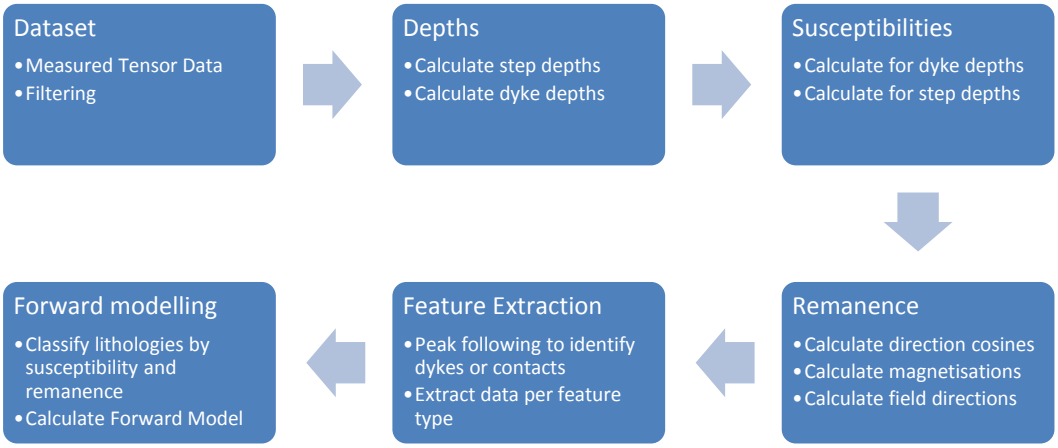

Figure 77 The process flow for calculations involving measured tensor data

These steps will be demonstrated and elaborated on using real data.

In addition to measured tensor data available over the Tallawang area, tensor data will be calculated from conventional total magnetic intensity data to illustrate the technique over a wider area.

6.1 Calculation of tensor components from TMI

Clark (2013) discusses the use of a calculated tensor dataset versus a measured tensor dataset. He confirms that the tensor components can be calculated from TMI data, but may be affected by deficiencies in the TMI survey itself. Clark (2013) further stressed that while direct measurement of the gradient tensor would produce superior results, useful gradient tensor data can be produced via Fourier processing, provided that a number of issues are addressed. These include:

- Effective removal of regional trend.
- Careful windowing, to minimise Gibbs phenomenon “ringing” and spectral leakage.
- Sample density (including between lines) appropriate to eliminate aliasing of high frequencies in the measured fields.
- Appropriate grid interpolation method for TMI data, with sampling no greater than $h/2$, where h is the depth to the source (Reid, 1980).
- Accurate reduction of TMI data to a common level.

Clark (2013) states that the TMI surveys require a line spacing comparable to the survey altitude. If there is non-magnetic cover, the depth to magnetic basement can be added to the survey altitude. The grid spacing should be no more than half the depth to sources. Clark (2013) does confirm that grid spacing restriction can be relaxed slightly in situations where the source crosses multiple lines.

The creation of a tensor dataset is straightforward. The process is described by Clark (2013), Pedersen, Rasmussen and Dyrelius (1990) and Yin et al.(2016). Using fast Fourier transforms, the components of the magnetic field are related to the total magnetic intensity f as follows:

$$\mathcal{F}[B_x] = \frac{i \cdot k_x}{i \cdot (\alpha k_x + \beta k_y) + \gamma k} \mathcal{F}[f] \tag{6.1}$$

$$\mathcal{F}[B_y] = \frac{i \cdot k_y}{i \cdot (\alpha k_x + \beta k_y) + \gamma k} \mathcal{F}[f] \tag{6.2}$$

$$\mathcal{F}[B_z] = \frac{k}{i \cdot (\alpha k_x + \beta k_y) + \gamma k} \mathcal{F}[f] \tag{6.3}$$

where $k = \sqrt{k_x^2 + k_y^2}$, k_x and k_y are spatial frequencies in horizontal direction, and should not be confused with susceptibility. $\mathcal{F}[\,]$ expresses the Fourier transform. Also, α, β and γ are direction cosines as defined in equations (2.14), (2.15) and (2.16). Similarly, the tensor operators are defined as:

$$\mathcal{F}[B_{xx}] = \frac{-k_x^2}{i \cdot (\alpha k_x + \beta k_y) + \gamma k} \mathcal{F}[f] \tag{6.4}$$

$$\mathcal{F}[B_{xy}] = \frac{-k_x k_y}{i \cdot (\alpha k_x + \beta k_y) + \gamma k} \mathcal{F}[f] \tag{6.5}$$

$$\mathcal{F}[B_{xz}] = \frac{i \cdot k_x k}{i \cdot (\alpha k_x + \beta k_y) + \gamma k} \mathcal{F}[f] \tag{6.6}$$

$$\mathcal{F}[B_{yz}] = \frac{i \cdot k_y k}{i \cdot (\alpha k_x + \beta k_y) + \gamma k} \mathcal{F}[f] \tag{6.7}$$

$$\mathcal{F}\left[B_{yy}\right] = \frac{-k_y^2}{i \cdot \left(\alpha k_x + \beta k_y\right) + \gamma k} \mathcal{F}[f] \tag{6.8}$$

$$\mathcal{F}[B_{zz}] = \frac{k^2}{i \cdot \left(\alpha k_x + \beta k_y\right) + \gamma k} \mathcal{F}[f] \tag{6.9}$$

These equation make use of the approximation to the TMI, i.e.

$$f = \alpha \cdot B_x + \beta \cdot B_y + \gamma \cdot B_z \tag{6.10}$$

The true expression is actually (Schmidt and Clark, 2006):

$$\begin{aligned} B_{tmi} &= \sqrt{(B_x + \alpha B)^2 + \left(B_y + \beta B\right)^2 + (B_z + \gamma B)^2} - B_a \\ &= f + \frac{B_x^2 + {B_y}^2 + {B_z}^2 - B_{tmi}^2}{2B_a} \end{aligned} \tag{6.11}$$

where B_{tmi} also refers to the measured total magnetic intensity and B is the ambient field. This implies that if the anomalies are big enough, the calculations for the tensors will be inaccurate. The process to correct this is as follows (Clark, 2013):

1) Calculate the values for the magnetic vector data using equations (6.1), (6.2) and (6.3)
2) Calculate a corrected estimate for the magnetic field f using the following equation:

$$f = B_{tmi} - \frac{B_x^2 + {B_y}^2 + {B_z}^2 - B_{tmi}^2}{2B_a} \tag{6.12}$$

3) Repeat steps 1 and 2 with the updated value for f, until the difference between f and B_{tmi} is less than an acceptable threshold defined by the expected noise level.
4) Apply this updated value to equations (6.4) to (6.9).

This technique is not without its limitations though. The process relies on direction cosines which must be known beforehand. These direction cosines are not due to only the ambient field, but also include the effect of remanence in the source. If remanence is present, especially if its magnetisation is proportional to that of the inducing field,

incorrect solutions will result. Figure 78 illustrates this. The source was modelled in an ambient field of 28 000 nT, with inclination of 60 degrees, and declination of -30 degrees. The susceptibility was 0.01 SI. The source was 20 m below the surface. In Figure 78(b) remanent magnetisation of 0.323 A/m with an inclination of -40 degrees and declination of 20 degrees is applied.

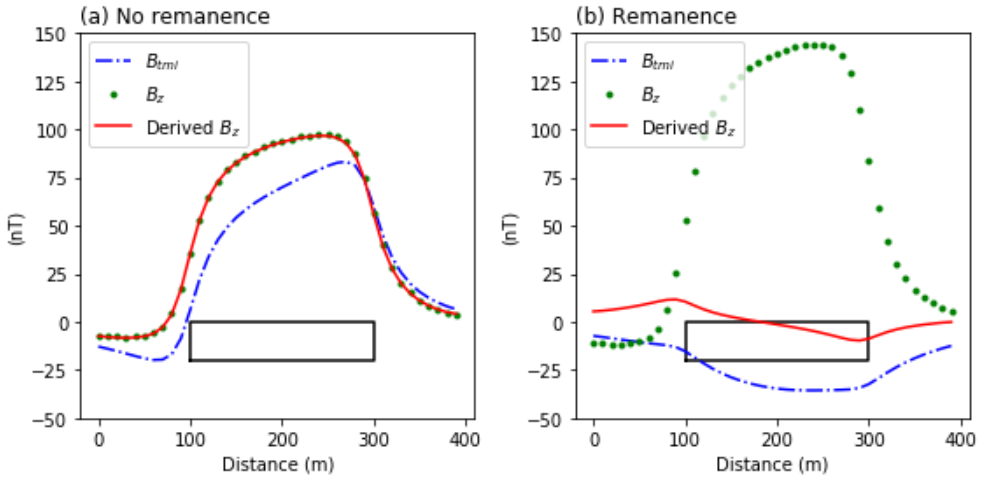

Figure 78 Demonstration of the effect of remanence. The black square shows the horizontal location of the source (a) No remanence and the derived z component matches its modelled counterpart. (b) Remanence is now modelled but the derived field is now different.

Figure 79 shows the effect of using direction cosines which incorporate the remanent magnetisation of the source. The resultant derived field component is now a more accurate approximation of the true field component.

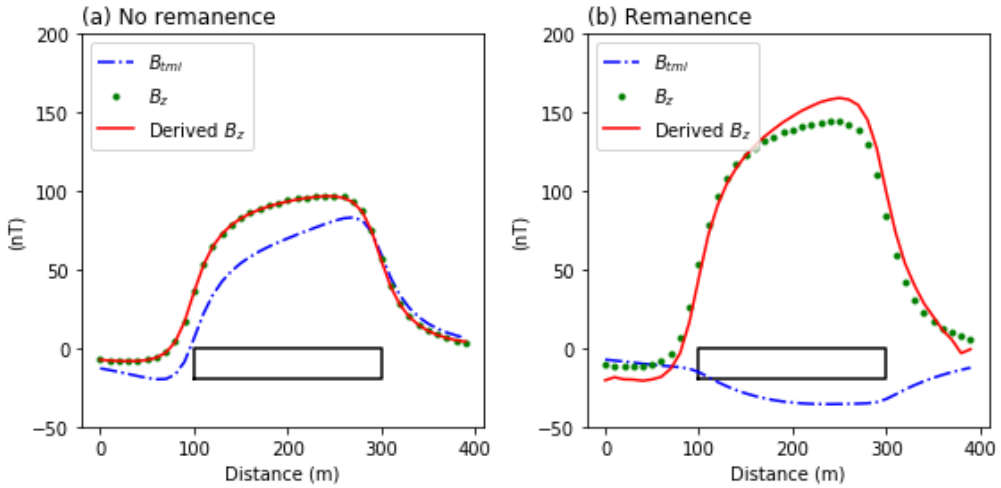

Figure 79 Demonstration of the effect using correct direction cosines. The black square shows the horizontal location of the source (a) No remanence and the derived z component matches its modelled counterpart. (b) Remanence is now modelled, and the derived field is a better approximation of the true field.

The paper by Schmidt et al., (2004) reports that the calculation of tensor data from TMI data does retain the remanent information, as is evident from their results when comparing calculated versus measured tensor data. In their case there is good agreement between these two datasets. A body was modelled using remanent field parameters and ambient field parameters similar to reported by Schmidt et al., (2004) for the Tallawang survey. The source was modelled in an ambient field of 57 000 nT, with inclination of -63 degrees, and declination of 12 degrees. The susceptibility was 2.5 SI. The source was 20 m below the surface. The remanent magnetisation was 40 A/m with an inclination of -70 degrees and declination of -60. The results are shown in Figure 80. As can been seen there is negligible difference in anomaly shape between the calculations where the body has remanence, to when the body does not have remanence. In addition, in the remanent case, there is now good agreement between the derived and the true values for the B_z field.The logical conclusion is that for this case, the combination of similar inclinations and low Q-ratio (.124 in this case) make the impact of remanence on the peak shape negligible, giving the appearance that the technique can account for remanence.

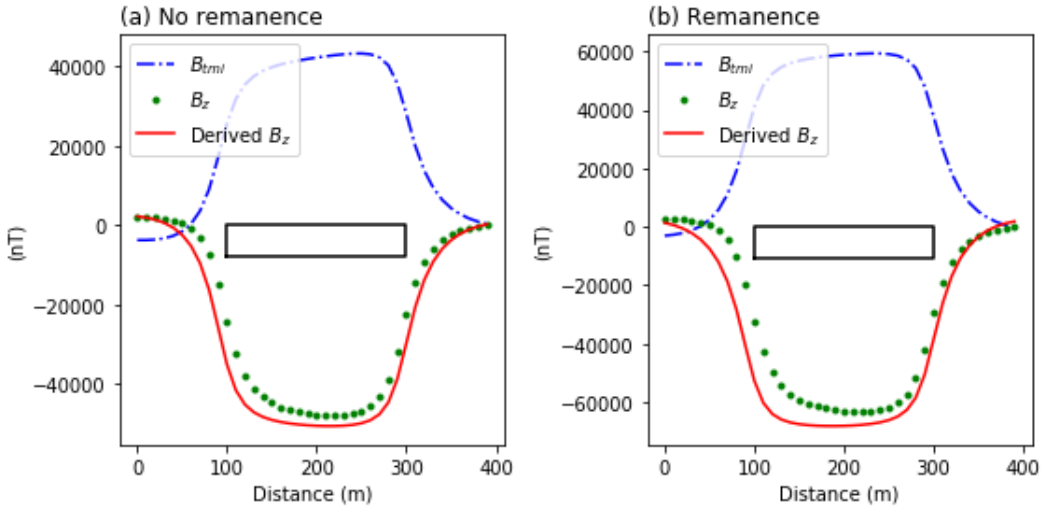

Figure 80 Demonstration of the effect of remanence using field parameters similar to that of the Tallawang area. The black square shows the horizontal location of the source (a) No remanence and the derived z component matches its modelled counterpart. (b) Remanence is now modelled showing similar results.

A final case is shown in examining the effect of noise of this technique. The model used is the same as that used for Figure 78. Gaussian noise with a standard deviation equal to 1.0% of the maximum data amplitude was added to the modelled TMI data. The same noise was added for the case with remanence. In both cases the effect of noise has a can be seen on the calculation of the component, but it is a minor effect and the integrity of the component calculation is intact. In the remanent case, since this anomaly is flatter,

the effect of noise is larger, which is to be expected. These results show the robustness of this technique.

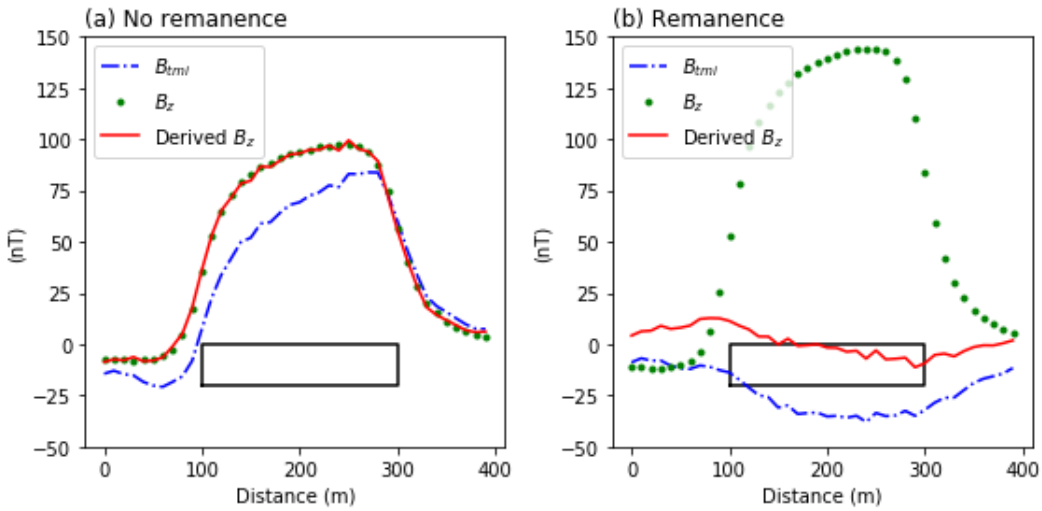

Figure 81 Demonstration of the effect of Gaussian noise with a standard deviation equal to 1.0% of the maximum data amplitude added to the TMI. The black square shows the horizontal location of the source (a) No remanence and the derived z component matches its modelled counterpart, even with noise. (b) Remanence is now modelled but the derived field is now different, and noise can be seen clearly here.

Figure 82 shows the effect of increased noise with Gaussian noise with a standard deviation equal to 5.0% of the maximum data amplitude was added to the modelled TMI data. Once again, the same noise was added for the case with remanence. The effect of the noise has now increased dramatically, impacting on the integrity of the calculated components. Therefore, the quality of the TMI data is important when using this technique.

Figure 83 shows the effect of calculating the B_z component from the derived B_{xz} tensor component. This is achieved through numerical integration, and as expected there is a shift in the results from the true value. This is because any derivative (such as B_{xz} which is the x-derivative of B_z) eliminates any constant from the original quantity. Upon integration, that constant is no longer present and a shift is the result. Since this calculation was performed on the derived B_{xz} tensor data, the integrated value mimics the derived B_z data.

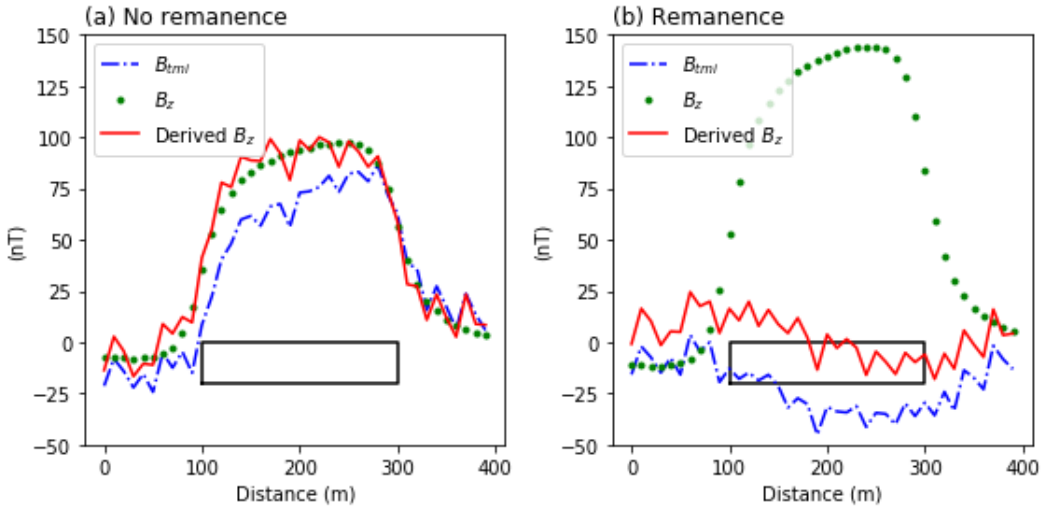

Figure 82 Demonstration of the effect of Gaussian noise with a standard deviation equal to 5.0% of the maximum data amplitude added to the TMI. The black square shows the horizontal location of the source (a) No remanence and the derived z component has deteriorated when compare to its modelled counterpart. (b) Remanence is now modelled but the derived field is now different, and noise can be seen more clearly here

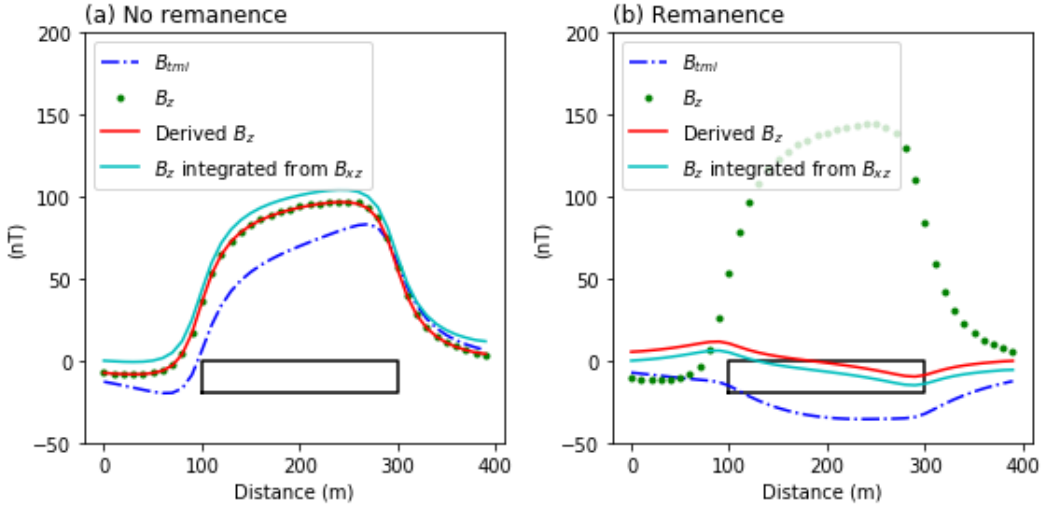

Figure 83 Demonstration of the effect of integrating the derived tensor values. (a) No remanence and the integrated z component shows a shift from the true z component value. (b) Remanence is once again modelled. The integrated value mirrors the derived field, with a shift, and does not correspond to the true z component value.

In summary, tensor components can be calculated from TMI data, but care must be taken to understand the limitations. Best results are achieved with data that is relatively noise free, and for areas where there is no remanence. If remanence does occur in the study area, good results can be achieved if the remanent directions are similar to the ambient field direction, and the Q-ratio is low. Areas with remanence may need laboratory samples measured to confirm this.

6.2 Peak Following Routine

The determination of the location of optimal solutions can be achieved through the use of peak following on an appropriate edge detection dataset. In this case, the analytic signal As can be used to find the location for relevant depth and susceptibility solutions. A number of peak following routines are available. Some utilize derivatives to find peaks and troughs where derivatives are zero. Others simply compare arrays with a moving window and identify maximum points. False positives due to noise may be eliminated through smoothing. This latter technique is what was used, and can be found in the SciPy library (Jones, Oliphant, Peterson, et al, 2001) as the command *argrelmax*. The output from this should be a set of grid row and column indices relating to peaks found. The indices are then applied to the depth and susceptibility datasets to extract the correct depth and susceptibility solutions. Figure 84 shows the input dataset from the Lichtenberg/Zeerust region and Figure 85 shows the results.

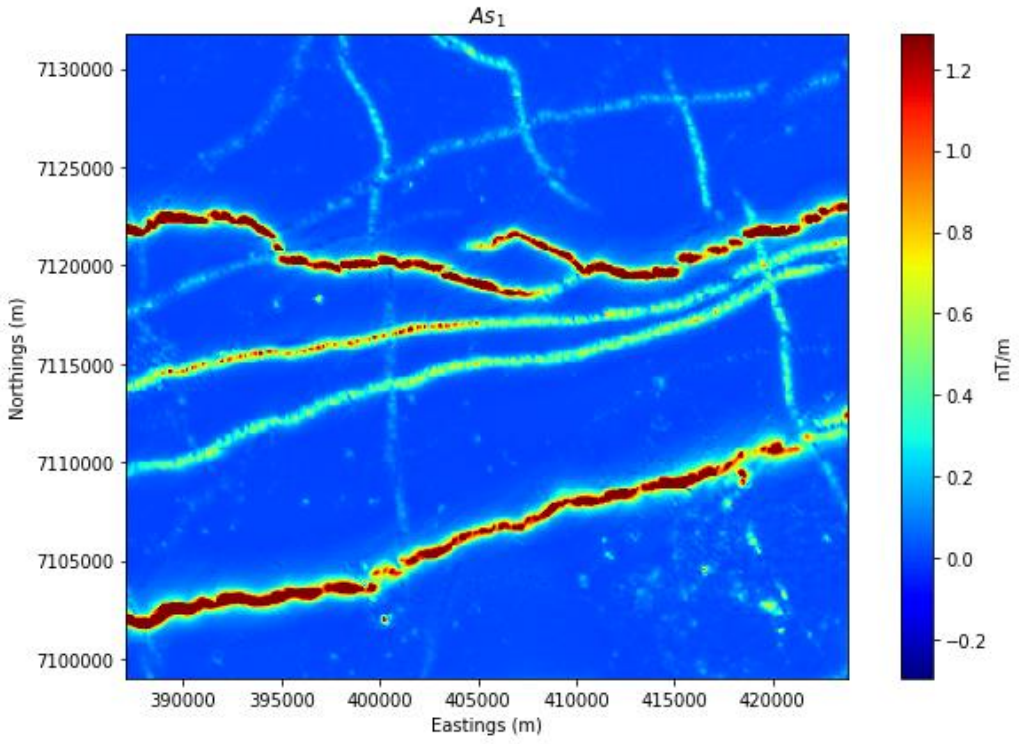

Figure 84 Analytic signal dataset used as input for peak following routine

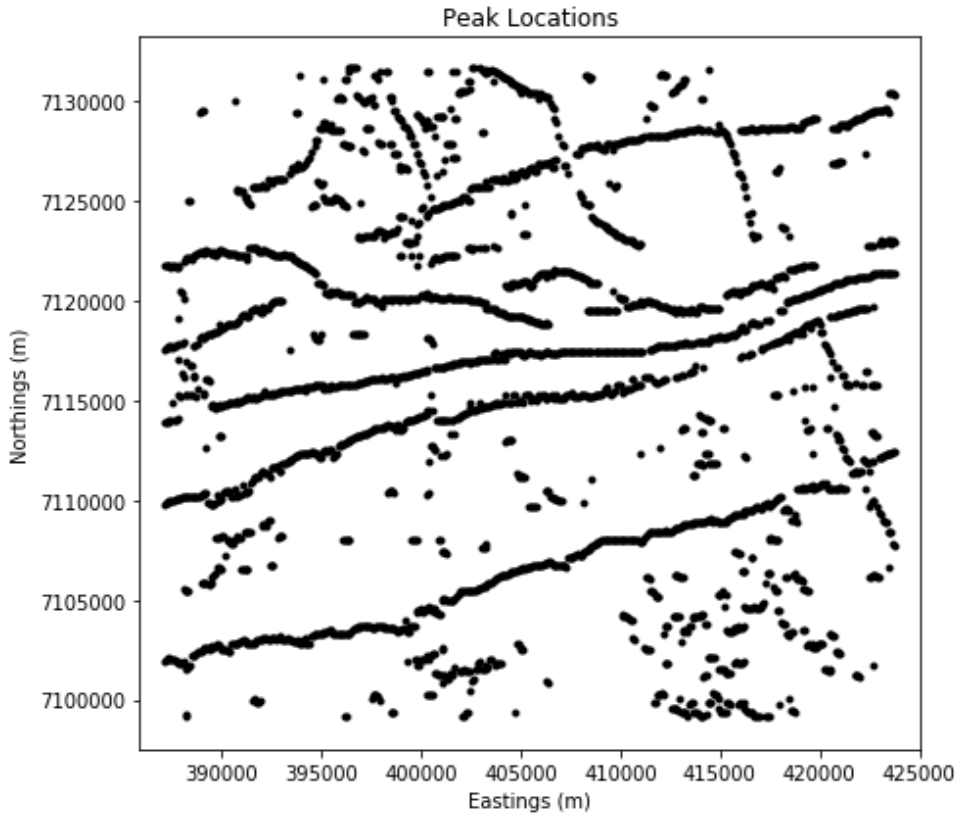

Figure 85 Peak locations plotted in terms of eastings and northings

Depths which are greater than the predefined depth of the model can then be filtered out from these initial solution. Indices are reduced to reflect this filtering.

The next step is to differentiate between solutions of different sources. Point sources can be differentiated from linear sources (such as edges or dykes) by using a relevant form of cluster analysis with the index data (solution coordinates) as input. The Density-Based Spatial Clustering of Applications with Noise (DBSCAN) method (Ester et al., 1996) was used in this case. DBSCAN has the advantage that it can find arbitrary shaped clusters. Since valid solutions must be grouped into dykes and other sources, this is critical.

The DBSCAN method is a density-based clustering algorithm. It groups together points in space that are packed closely together (points with many nearby neighbours). It classifies points which are in low density regions as outliers. It requires two parameters, namely the minimum number of points to form a dense region, and the minimum distance between any two samples to be considered in the same neighbourhood. Because of the density criterion, DBSCAN can find arbitrarily shaped clusters. It also does not require the number of clusters to be set beforehand.

Filtering is applied to the DBSCAN results. In this case, since dykes are the target, classes with too few members are excluded. Figure 86 shows the results.

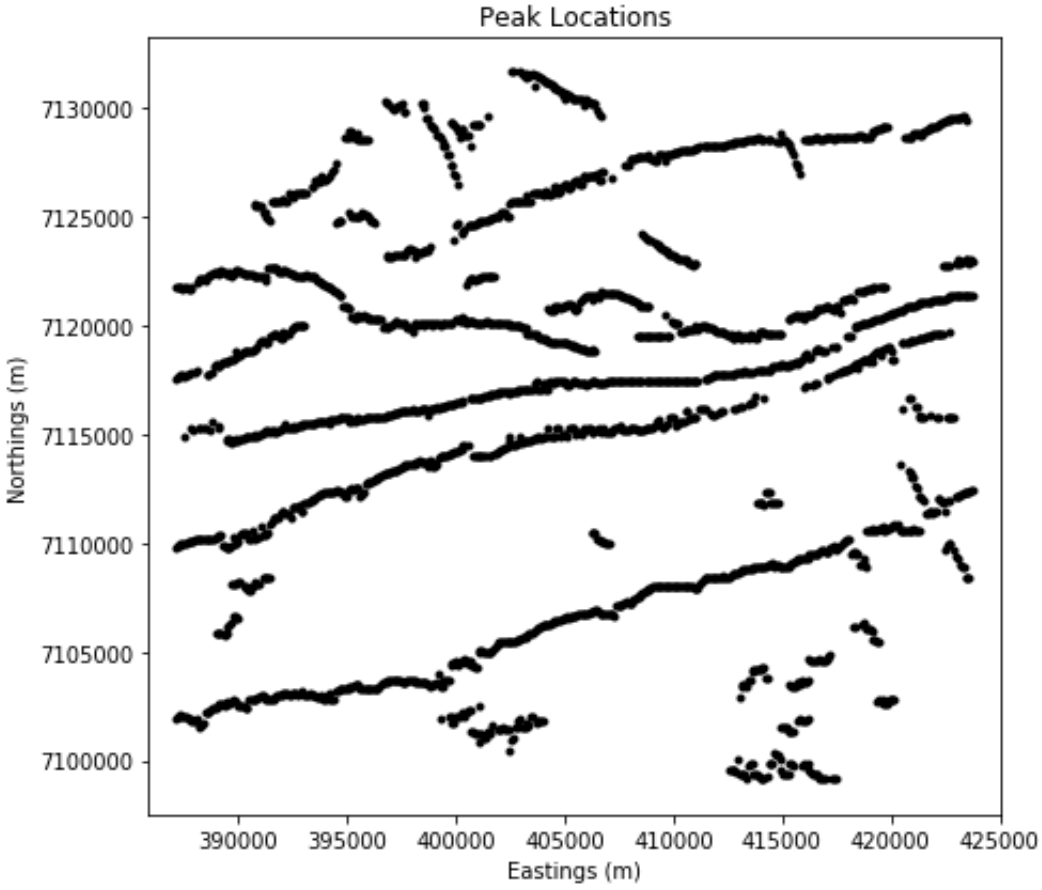

Figure 86 Filtered results for the peak locations (Figure 85) after DBSCAN

A final point is that results may not be produced for only the feature being investigated. For example, we may be interested in dykes only, but some steps may be visible. The peak following routines may follow some of these sources as well (for example, since the edge of a step is identified on the analytic signal by a peak). A final differentiation between dykes, steps and other features can then be done through a process of manual examination and elimination of relevant filtered results. This can be done in a GIS, for example. While full automation is a desirable goal, manual interaction such as this can improve the relevance and quality of results vastly.

6.3 Tallawang Field Trial

6.3.1 Geological Setting and Data

Tensor data over the Tallawang magnetite deposit was generously provided by Dr David Clark of the CSIRO (Commonwealth Scientific and Industrial Research Organisation).

The Tallawang magnetite deposit ($32°12'S, 149°27'E$) is a tabular skarn body (Figure 87). It is located 18 km north of Gulgong, New South Wales, Australia, along the western margin of the Carboniferous Gulgong Granite (Schmidt et al., 2004; Clark et al., 1998). The skarn was intruded during the late stages of the Kanimblan Orogeny in the Late Carboniferous. The deposit strikes NNW and dips steeply to the west. The magnetite occurs in lenses with magnetite zones being displaced in an east-west direction, by transverse faulting (Schmidt et al., 2004).

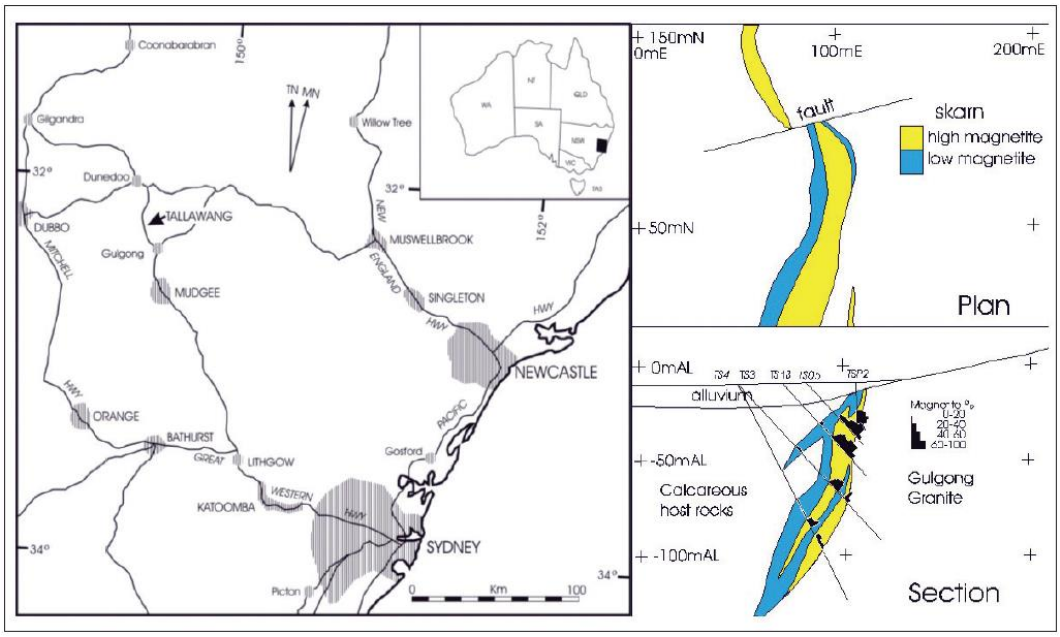

Figure 87 Geology of the Tallawang skarn (Schmidt et al., 2004).

The magnetite body has been drilled, and the properties of oriented block samples of the magnetite characterised. Schmidt et al., (2004) reported that the strongest samples had a susceptibility of 3.8 SI, with a remanence of 40 A/m and Q-ratios of 0.2 to 0.5. The remanence had a mean direction of WNW and steeply up. They further stated that this direction was a combination of two directions, namely, a viscous remanent magnetization (VRM) component in the direction of the recent geomagnetic field, and a reversed mid-Carboniferous field component. Clark et al., (1998) calculated the inclination to be $-72° \pm 30°$.

Schmidt et al., (2004) performed a tensor survey using the GETMAG system, detailed in section 3.3.1. They also collected high resolution TMI data using two Caesium vapour magnetometers separated vertically by 1 m. Readings were taken at 10 m intervals along east-west lines. The TMI data was collected for comparison purposes with the tensor data.

Three profiles were selected for the survey, denoted by 50mN, 60mN and 120mN by Schmidt et al., (2004) and are shown in Figure 88.

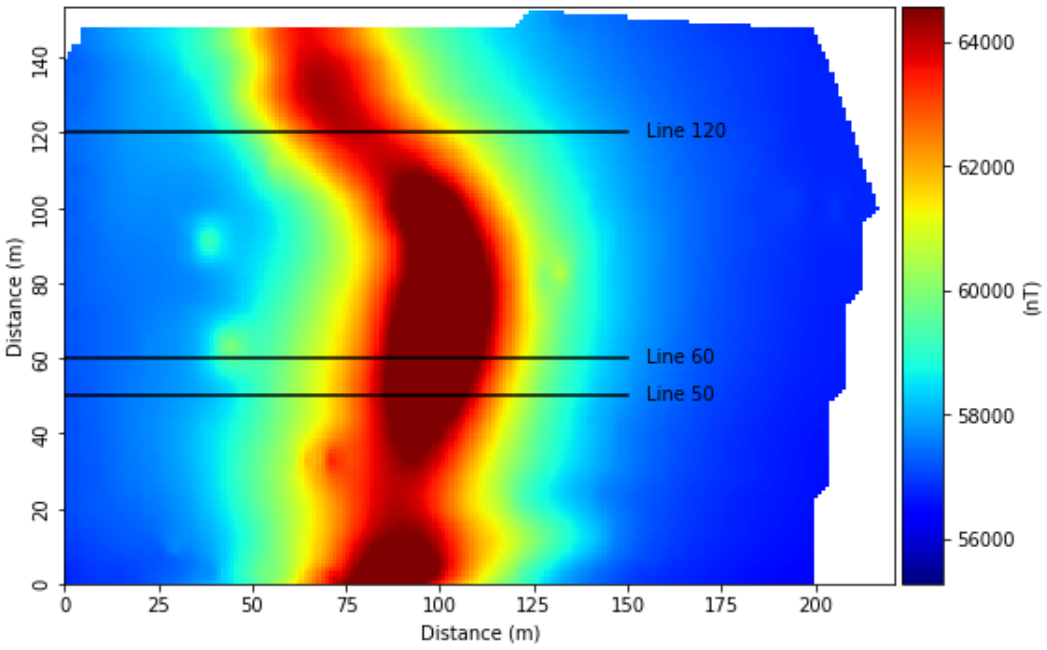

Figure 88 TMI data collected over Tallawang, showing the location of the three tensor profiles. Grid north is 340º True. Tensor survey lines are shown in black and are labelled.

6.3.2 Baseline Model of Tallawang Body

A simple baseline forward model of the Tallawang body was created in order to test methodology and understand the results from the measured data. The Tallawang body comprises a 10 m wide section of high magnetite content, with a further low magnetite content section to the west of the body (Figure 87). The high magnetite section was modelled since the majority of the anomaly comes from this ore and this makes for a good approximation of a dyke. The body was modelled in an ambient field of 56 701.6 nT, with inclination of -63 degrees, and declination of 11.47 degrees. The susceptibility was 2.5 SI. The source was 20 m below the surface. The remanent magnetisation was 30 A/m with an inclination of -70 degrees and declination of -60 degrees. The model results are

shown in Figure 89. The axis orientation of the model is NED to match that of the measured survey data.

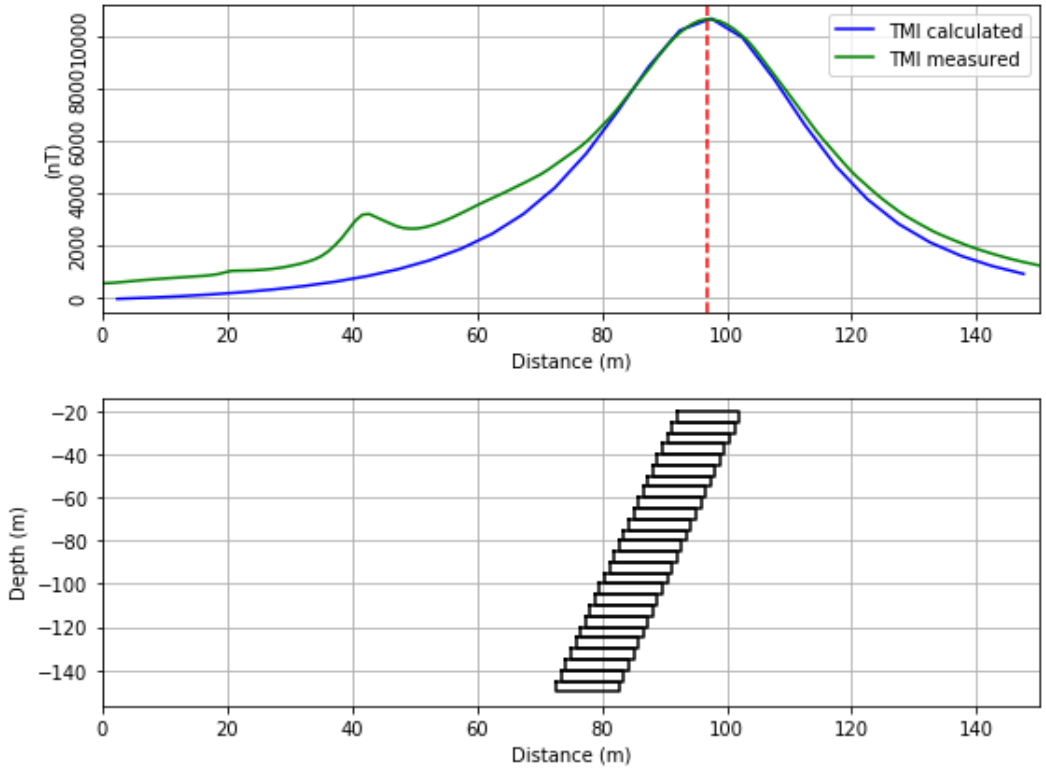

Figure 89 Model of the dipping Tallawang body showing the calculated and measured data over the deposit. The dashed black line indicates the centre of the body closest to the surface. The body comprises of layers of rectangular prisms, illustrating the calculation of such anomalies using rectangular prisms.

Figure 90 shows the tensor components calculated for the body. Tensor information is contained largely in the y and z directions, which is to be expected since the body runs north-south.

Figure 91 shows depth calculations using zero and first order analytic signal calculations. Equations used for the depth are shown on the figure. Since the modelled body is dyke-like, N=1. Depths calculated using these equations range from 14.5 to 19 metres. The shallowest depth uses the y-component only, with the other three depths clustering between 17 and 19 metres.

Figure 92 shows depth calculations using first and second order analytic signals. Equations used for the depth are shown on the figure. These depths tend to be deeper, ranging from 21 to 24 metres. Most of the depths cluster between approximately 21 and 22.5 metres, with the depth from the conventional analytic signal at 24 metres.

The hybrid depth using all three analytic signal components (denoted As_{xyz} in equations on the figures) gives conservative and stable solutions, since it appears with the clusters of solutions on both cases.

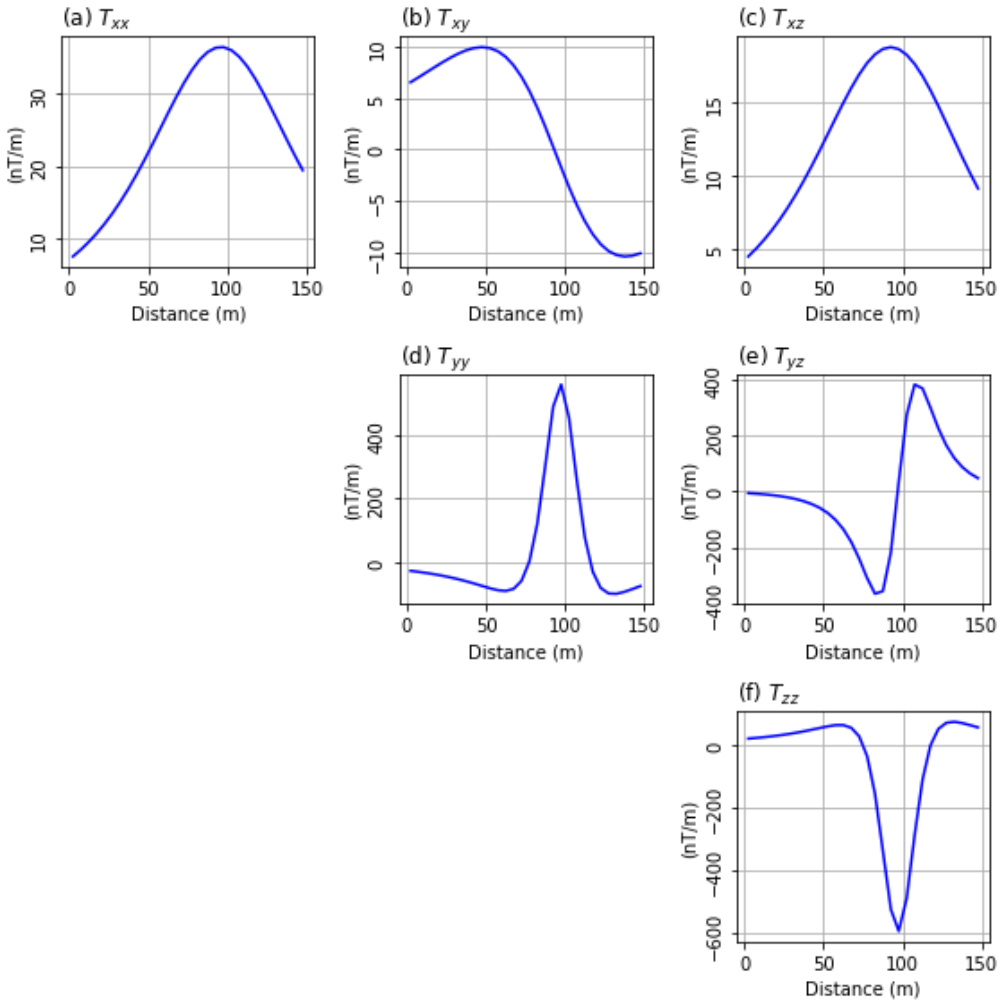

Figure 90 Tensor components calculated for the Tallawang body.

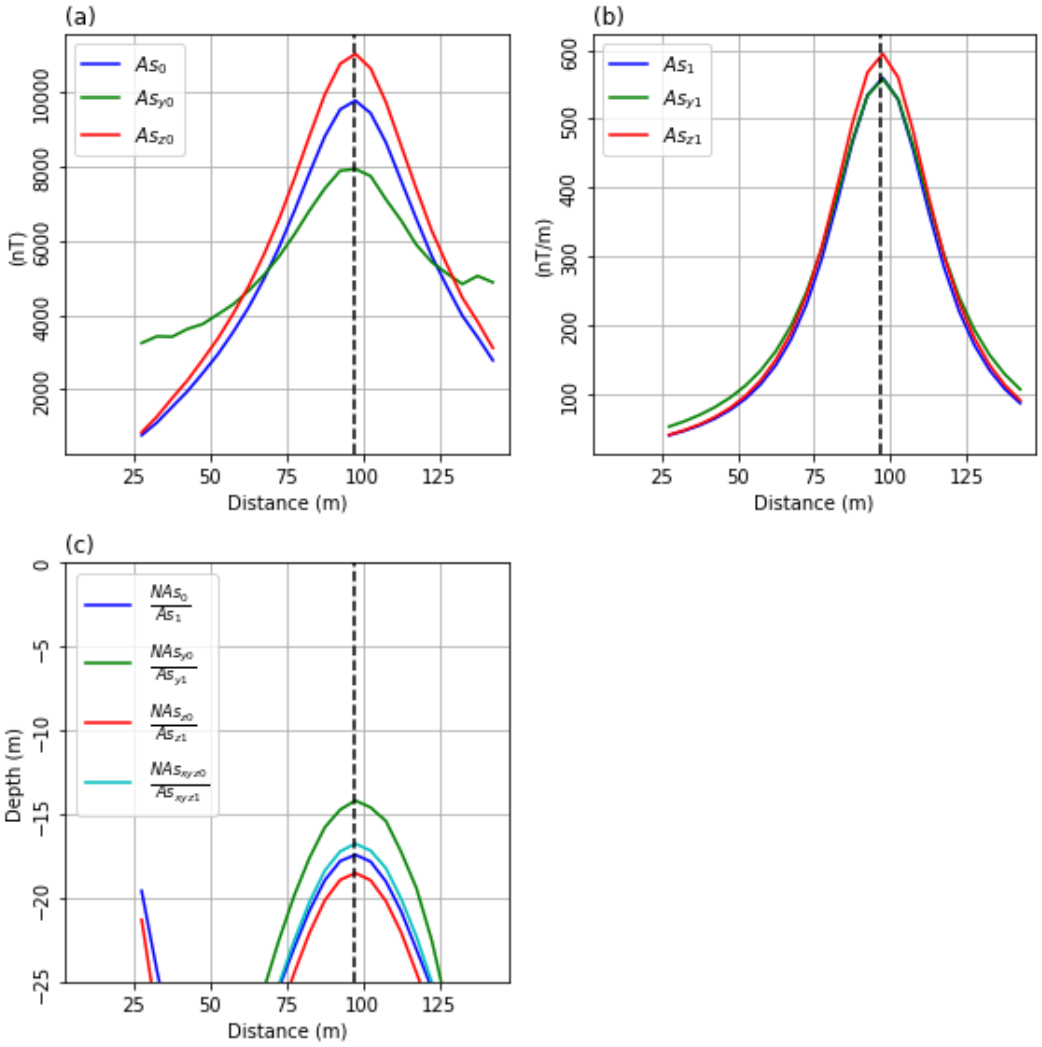

Figure 91 Depth calculation using zero and first order analytic signals. (a) shows the zero order calculations (b) shows the first order calculations (c) shows the depth results over the centre of the anomaly. The dashed line shows the location of the body centre.

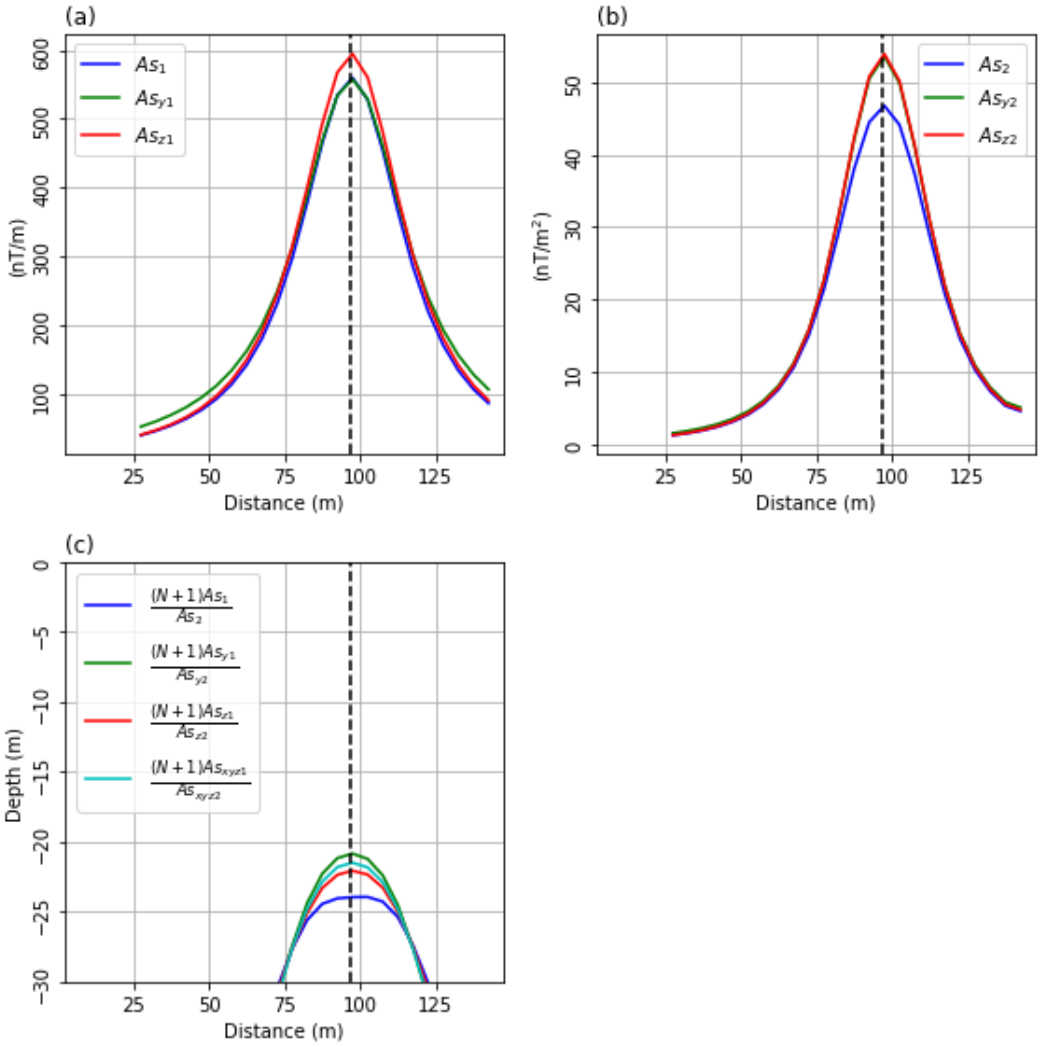

Figure 92 Depth calculation using first and second order analytic signals. (a) shows the first order calculations (b) shows the second order calculations (c) shows the depth results over the centre of the anomaly. The dashed line shows the location of the body centre.

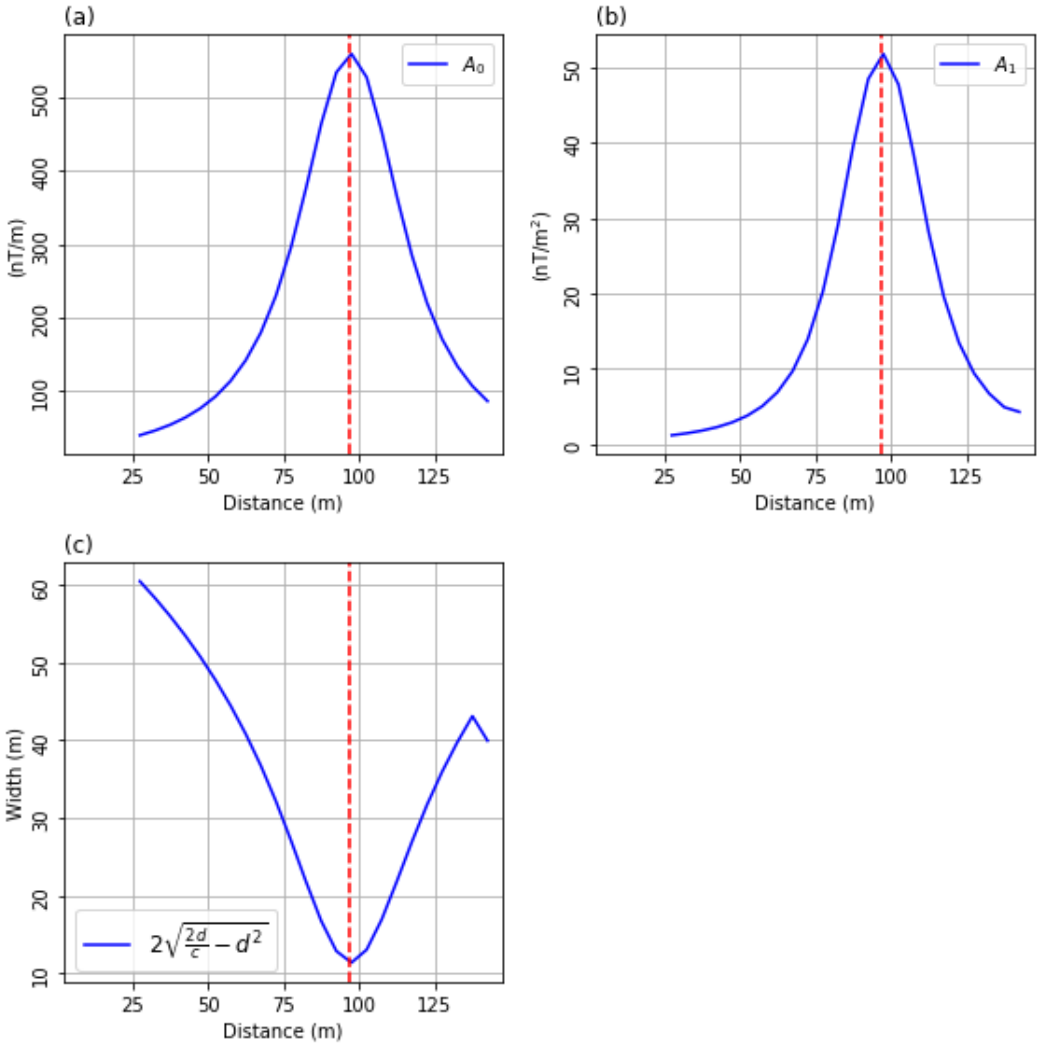

Figure 93 Results for the dyke width calculation. (a) shows the conventional analytic signal, denoted A_0 (b) shows the higher order analytic signal, denoted A_1. (c) shows results for the width calculation. The dashed line shows the location of the body centre.

Figure 93 shows results for the width calculations. The width is estimated at 11.3 metres, which is close to the modelled solution. A depth of 20 metres was used for this width estimate, to show the relative accuracy of the width estimate under ideal conditions. This depth is a reasonable mean value given the range of depths calculated using analytic signals.

These results are presented first, since they lead into the susceptibility and magnetisation calculations shown in Figure 94.

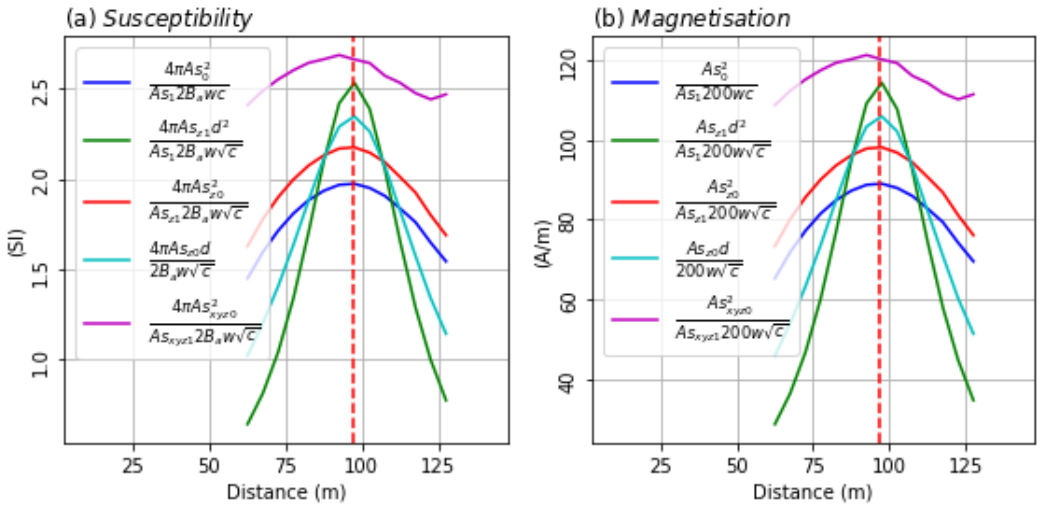

Figure 94 (a) Susceptibility calculations and (b) magnetisation calculations for the body. The dashed line shows the location of the body centre.

Steep anomalies relate to equations where a fixed depth (20 m) was used. In all cases, a width of 11.3 metres was used. The results give susceptibility ranges from 1.97 to 2.66. Steep anomalies with accurate fixed depths give the most reliable solutions, between 2.35 to 2.53 SI. The true total magnetisation for the model is 140 A/m. Relative to this, the solutions are underestimated, ranging from approximately 90 to 120 A/m.

Direction cosines for the modelled field were also calculated and are shown in Figure 95.

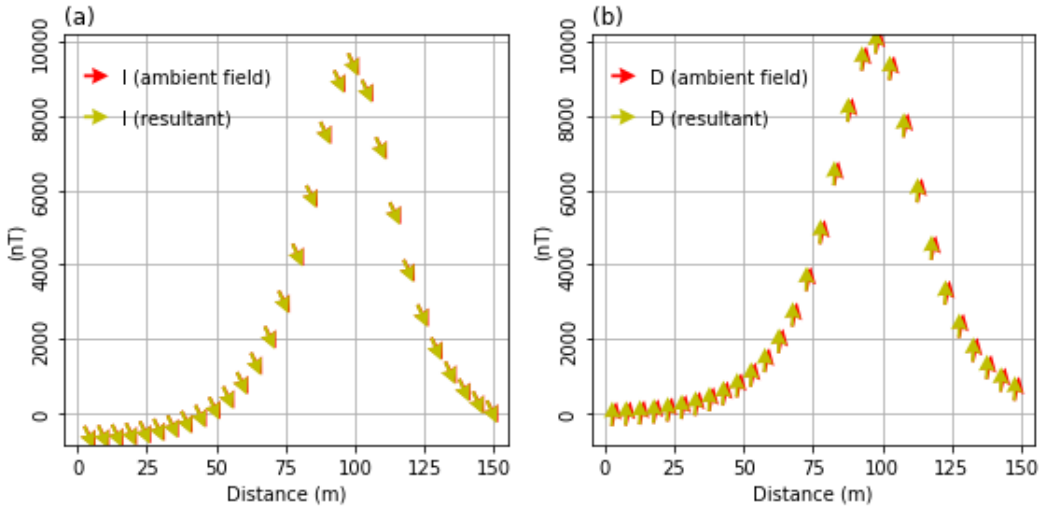

Figure 95 (a) Quiver plot showing a comparison between inclinations from the ambient field and the modelled field including remanence. (b) Quiver plot showing a comparison between declination from the ambient field and the total modelled field including remanence. There is not much difference between the two fields.

As can be seen, the inclinations are almost identical, where the ambient inclination was -63 degrees, and the total inclination including remanence, is now -66 degrees. Accordingly the quiver plot shows not much change. In the case of the declinations, there is a small change visible, where the ambient declination was 11.47 degrees and the new resultant declination is now 1.32 degrees. The quiver plot is useful since it gives an indication of where the remanent field direction lies, as well as the overall impact of remanence on the ambient field. In this case, the new total declination is 1.32 degrees, which is less than 11.47 degrees, and is diagnostic of a remanent field which is less than 1.32 degrees (since in this case only a value lower than 1.32 degrees can reduce the ambient field from 11.47 degrees to 1.32 degrees).

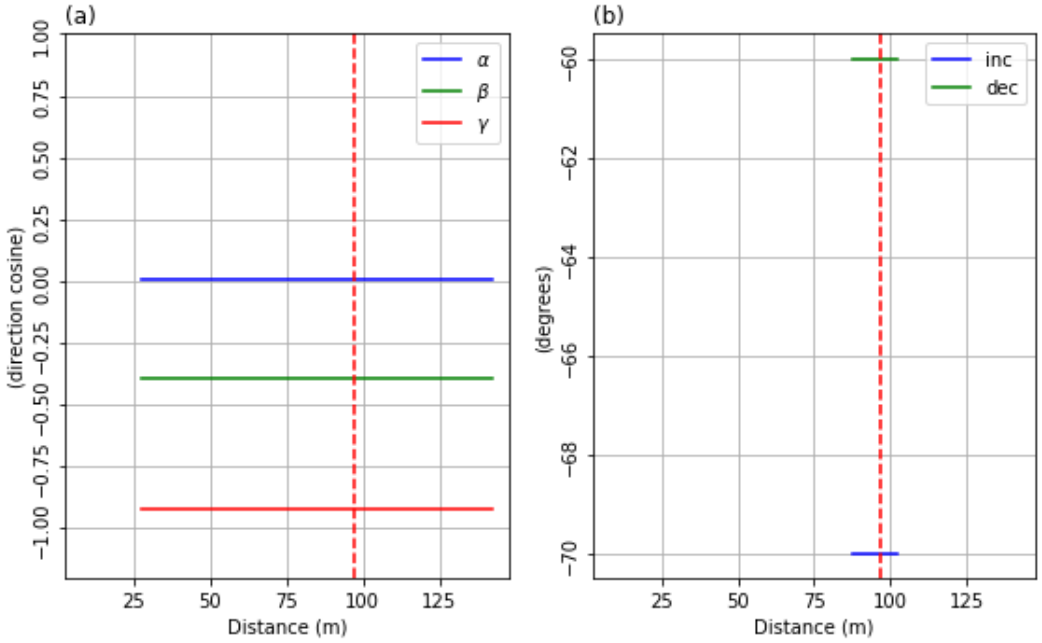

Figure 96 (a) Direction cosines resulting from the model. (b) calculated inclinations and declinations using ideal susceptibilities and magnetisations. (2.5 SI and 140 A/m). The dashed line shows the location of the body centre.

Figure 96(a) shows direction cosines using ordinary graphs. They are constant since there is only one body in the model, and the field which results is constant. Figure 96(b) shows values for inclination and declination when ideal values of 2.5 SI and 140 A/m are used for susceptibility and magnetisation. If the magnetisation is set to 114 A/m, in line with one of the magnetisation estimates, the results are not as accurate, with a resulting inclination of -21 degrees and declination of -119 degrees, This then shows that, as discussed in section 5.2, because of the low Q-ratio (0.07 in the case of this model) remanent magnetisation inclination and declination estimates of real data are unlikely to be accurate for this area.

6.3.3 Modelling of measured tensor data.

Line 50 and line 60 were examined since, being 10 metres apart, they offer the unique possibility of being able to measure tensor gradients between the lines. This is necessary for some of the calculations (for example the 3D analytic signal), which require cross line gradients for best results. The sample spacing of the data along each line was also 10 metres. Figure 97 shows the TMI data of the two lines. The location of the lines can be seen in Figure 88.

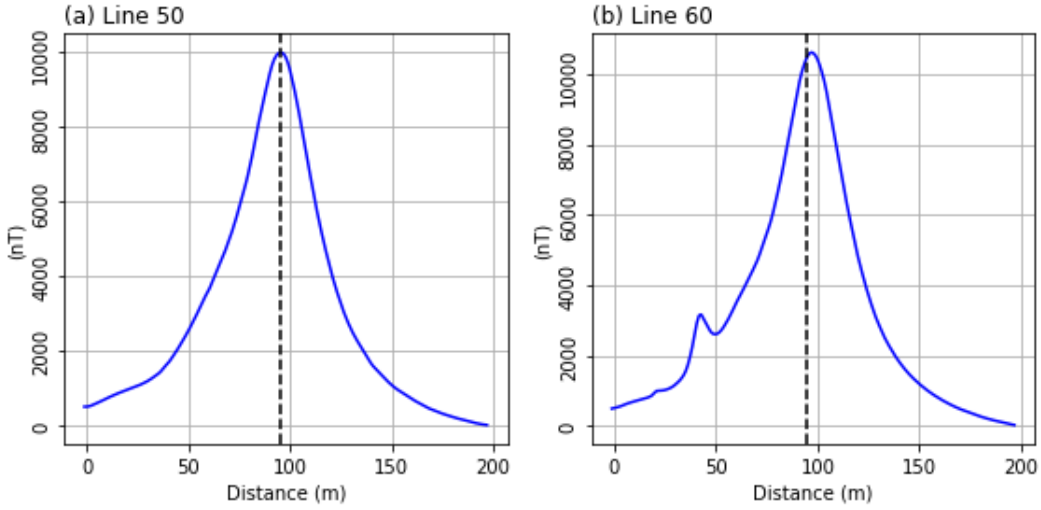

Figure 97 TMI data of (a) line 50 and (b) line 60. The dashed line shows the location of the body centre.

Figure 98 shows the first and second order analytic signals from line 50. Both line 50 and line 60 are used in the x (northern) components, which accounts for the small peak in the west. What is important to note is that while there is a single peak for the first order analytic signal, the second order analytic signal has multiple peaks. In analytic signals, this suggests that the body is shallow, and that for this dataset, a step solution (N=0) may need to be used as opposed to a dyke solution (N=1).

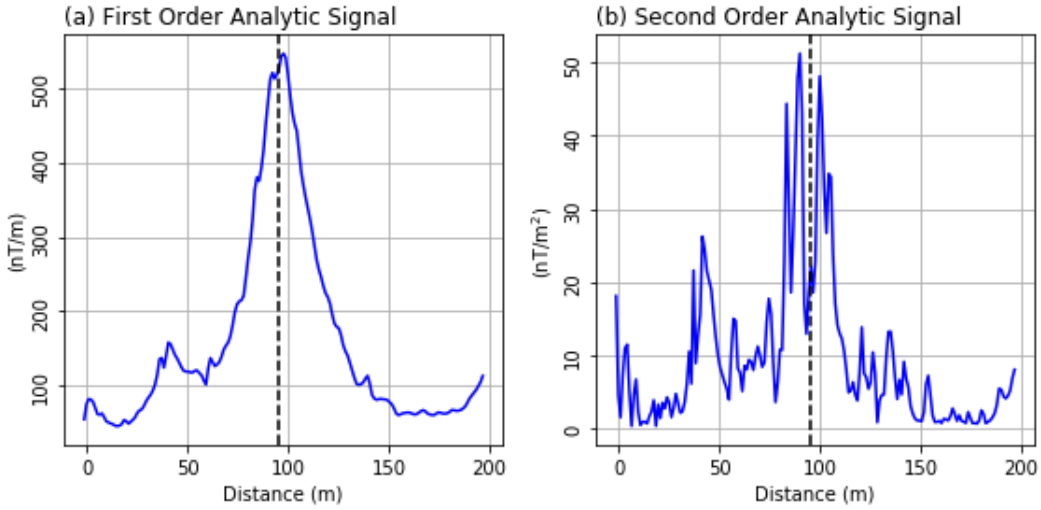

Figure 98 (a) First order analytic signal of line 50 showing distinct peak (b) Second order analytic signal of line 50 showing more complex peaks. The dashed line shows the location of the body centre.

Line 60

Figure 99 shows the tensor data recorded for line 60. The samples are spaced 10 metres apart. The sharp anomaly in the west is caused by a steel drill collar (Clark, 2012). As such, this region is of no interest and is trimmed off in subsequent plots. A comparison between Figure 90 and Figure 99 shows that the tensor components for the measured data has a greater range over the skarn than in the model. This clearly shows that the skarn is more complex than a simple dyke and already may test some of the assumptions with respect to a dyke formula. Figure 100 shows the depth calculations using zero and first order analytic signals. The solutions range between 16 metres and 24 metres which the worst solution being that using As_y. Figure 101 shows the depth calculation using first and second order analytic solutions for a dyke. These range between 40 and approximately 100 metres. However, if the step model is used, since As_2 better resembles step data (it has a low instead of a peak over the source), then the depth solutions would range between 20 and 50 metres, with the majority of the solutions between 20 and 32 metres. Figure 102 shows the calculation of body width, which ranges between 10 to 25 metres. All these solutions are consistent with the published depth of approximately 19.9 metres (Clark, 2012) and the width on the geological map (Figure 87) (Schmidt et al., 2004).

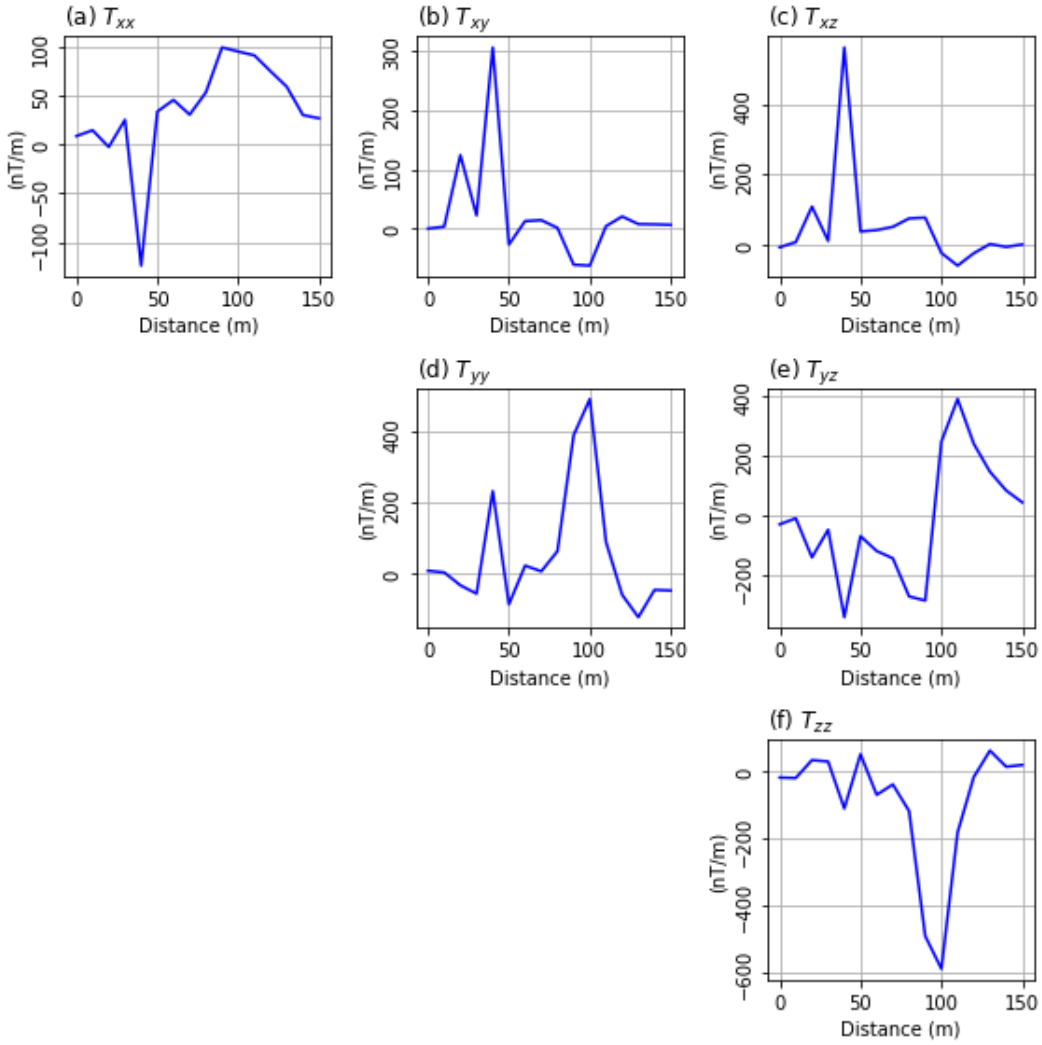

Figure 99 Tensor components measured for line 60 of the Tallawang body.

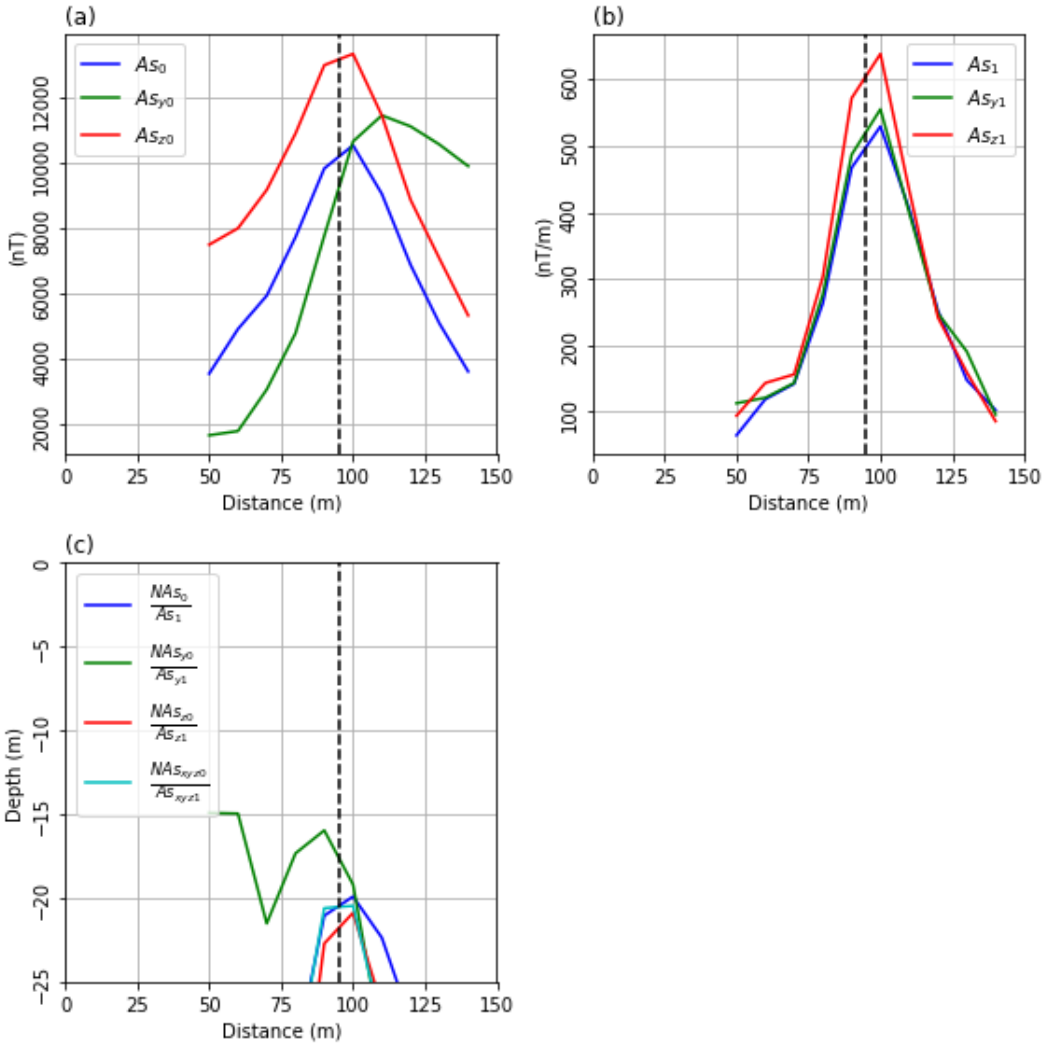

Figure 100 Depth calculation using zero and first order analytic signals. (a) shows the zero order calculations (b) shows the first order calculations (c) shows the depth results over the centre of the anomaly. N = 1. The dashed line shows the location of the body centre.

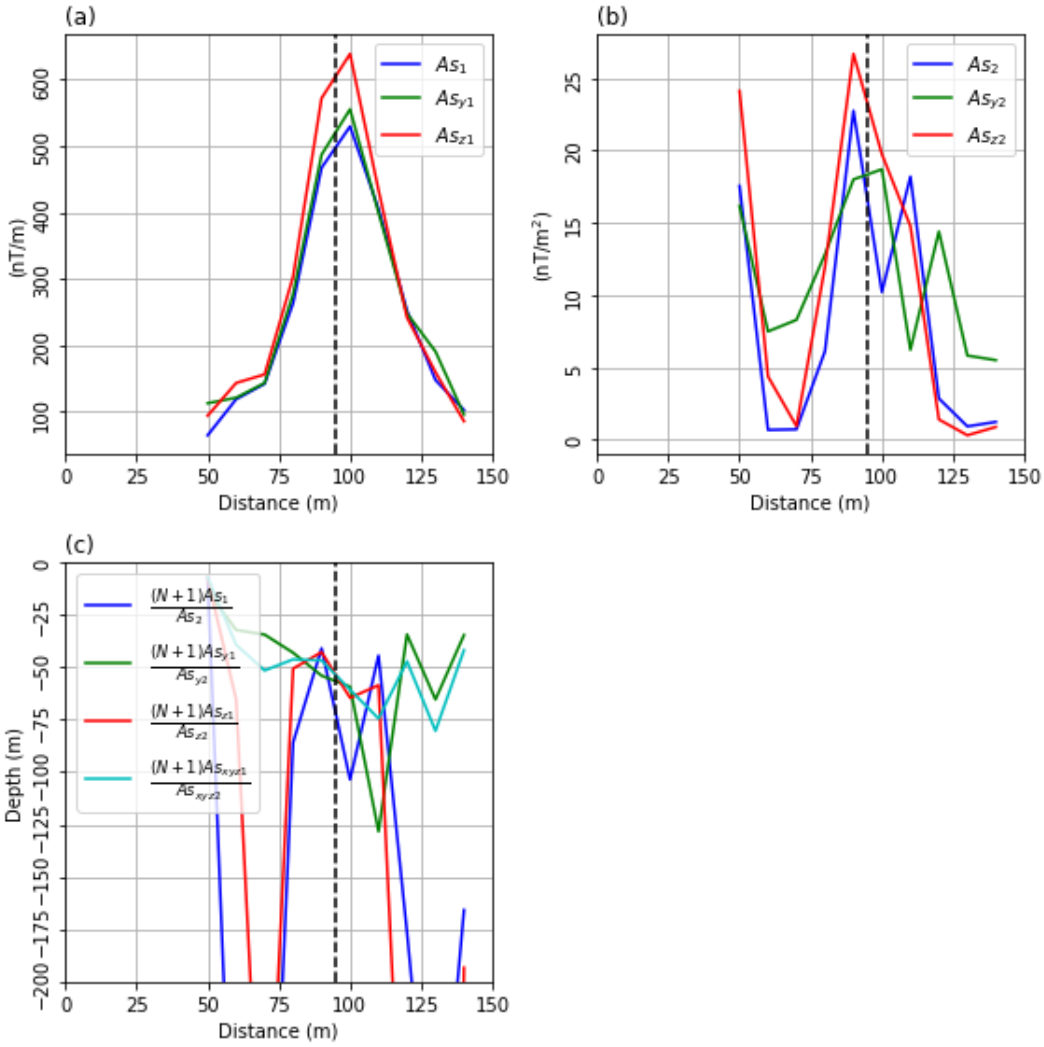

Figure 101 Depth calculation using first and second order analytic signals. (a) shows the first order calculations (b) shows the second order calculations (c) shows the depth results over the centre of the anomaly. N = 1. The dashed line shows the location of the body centre.

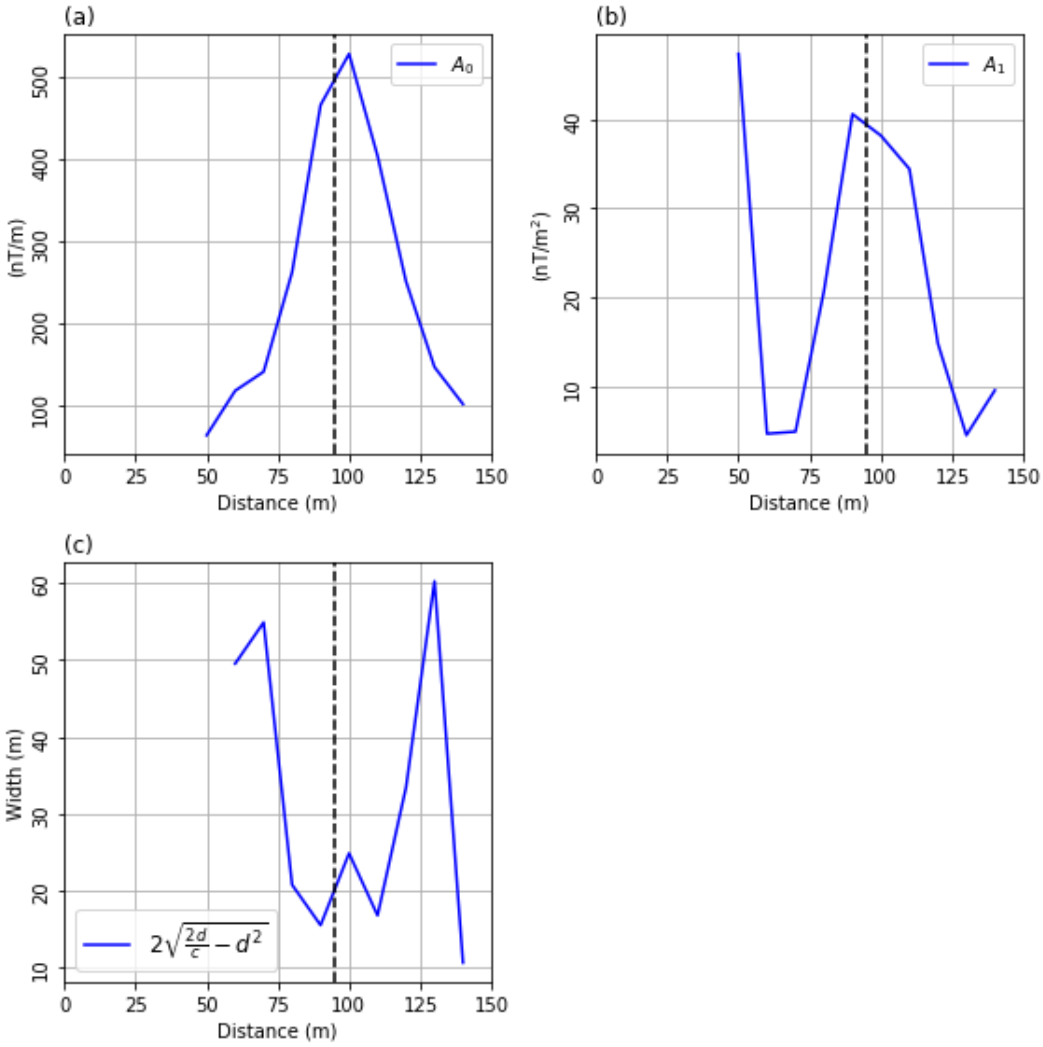

Figure 102 Results for the dyke width calculation. (a) shows the conventional analytic signal, denoted A_0 (b) shows the higher order analytic signal, denoted A_1. (c) shows results for the width calculation. The dashed line shows the location of the body centre

Figure 103 shows results of the calculated susceptibility and magnetism. A depth of 20 metres and a width of 15 metres was used in the calculation. Susceptibilities range between 2 SI and 14 SI over the body, and magnetisations range between 100 A/m and 600 A/m. Clearly many of these values are too large, with the components using As_z closest to the real values (4.3 SI to 4.7 SI and 193 A/m to 211 A/m). The inaccuracy may be indicative of the incorrect model used for calculating magnetisation and susceptibility and therefore not fitting this case perfectly.

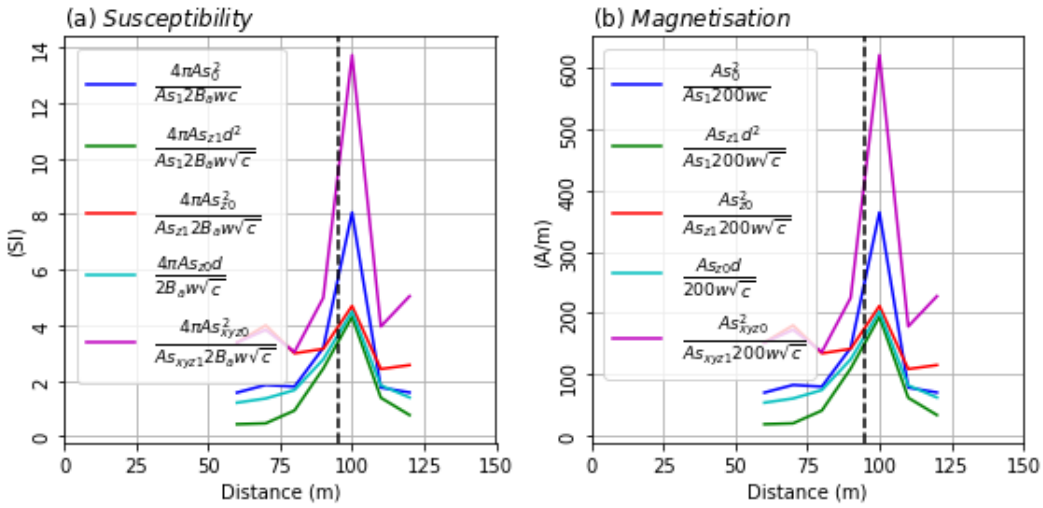

Figure 103 (a) susceptibility calculated from analytic signal formulae (b) magnetisation calculated from analytic signal formulae. The dashed line shows the location of the body centre.

Figure 104 shows quiver plots of the inclinations and declinations of the measured tensor values, compared with those of the ambient field. Measured inclinations, for the most part, are largely coincident with that of the ambient field. This makes sense since the remanence is steeply up, which is largely coincident with the ambient field. Deviations in the values to the west are caused either by another source or remanence, or, errors in the data (since the magnitude of the cosines in this section are occasionally greater than 1, which can only occur if there are errors in the measured data). Measured declinations show a more complex picture. They are largely positive west of the skarn and negative east of the skarn. This may relate to the complexity of the remanence which resides in the magnetite lenses.

The direction cosines shown on Figure 105(a) show a complexity in the field over the body, meaning that assumptions of homogeneity in modelling may not be valid. Some of the variability may be due to inherent data accuracy.

Figure 105(b) shows results for the inclination and declination calculations. The values of susceptibility and magnetisation used were calculated values and were 4.69 SI and 211.67 A/m respectively. These values are too large and the inaccuracies are reflected in the calculated inclinations and declinations, which are incorrect. The remanent magnetisation calculated from this is 103.97 A/m, which is larger than the maximum value of 40 A/m by a factor of 2.5.

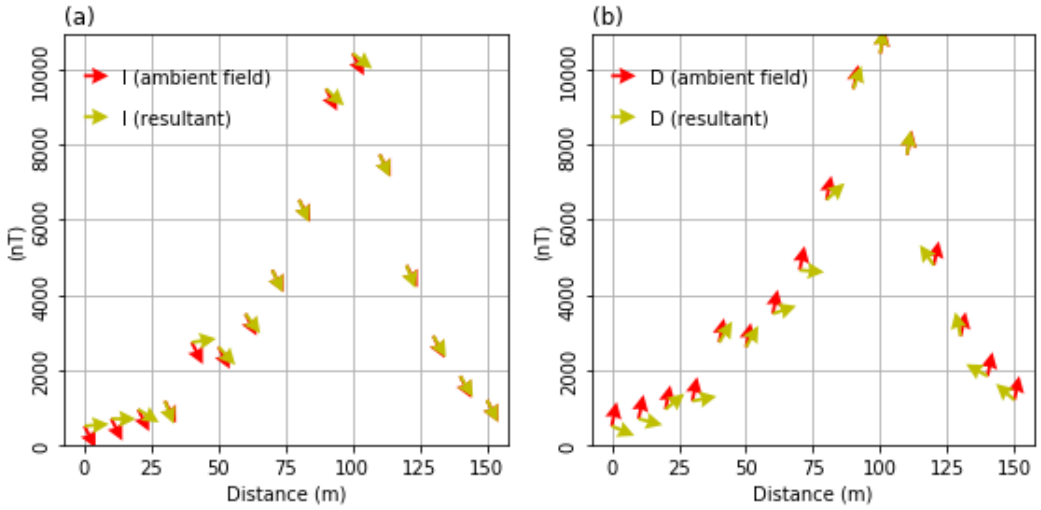

Figure 104 (a) Quiver plot showing a comparison between inclinations from the ambient field and the measured field including remanence. (b) Quiver plot showing a comparison between declination from the ambient field and the measured field including remanence

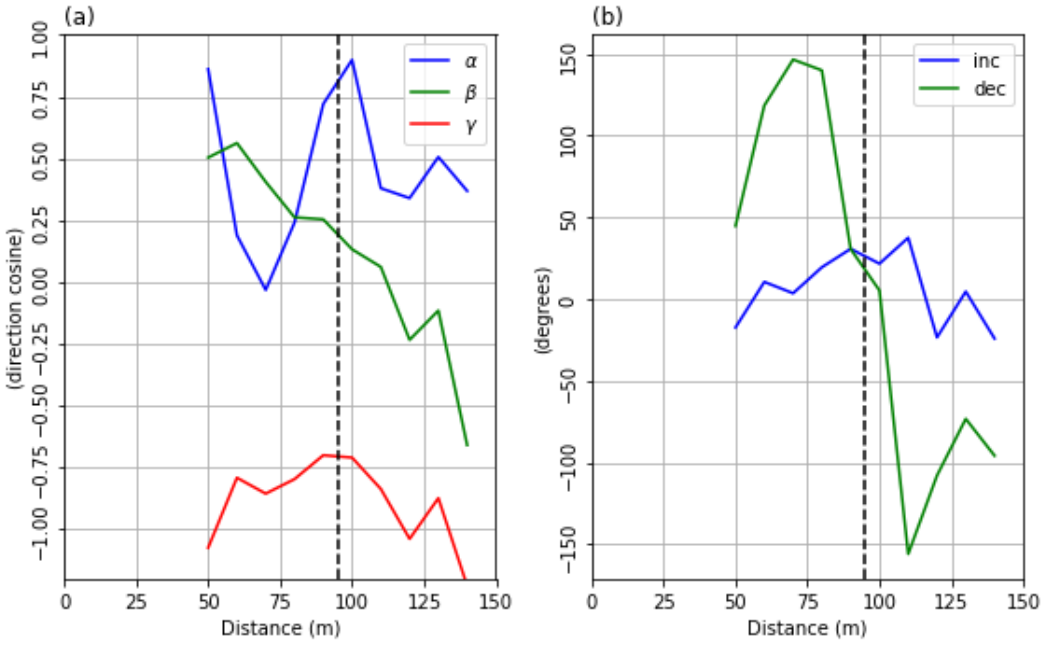

Figure 105 (a) Direction cosines resulting from the model. (b) calculated inclinations and declinations. The dashed line shows the location of the body centre.

Line 50

Line 50 shows a similar picture. Figure 106 shows tensor data measured over the skarn. The anomaly to the west may be related to the magnetic collar and is excluded from future figures.

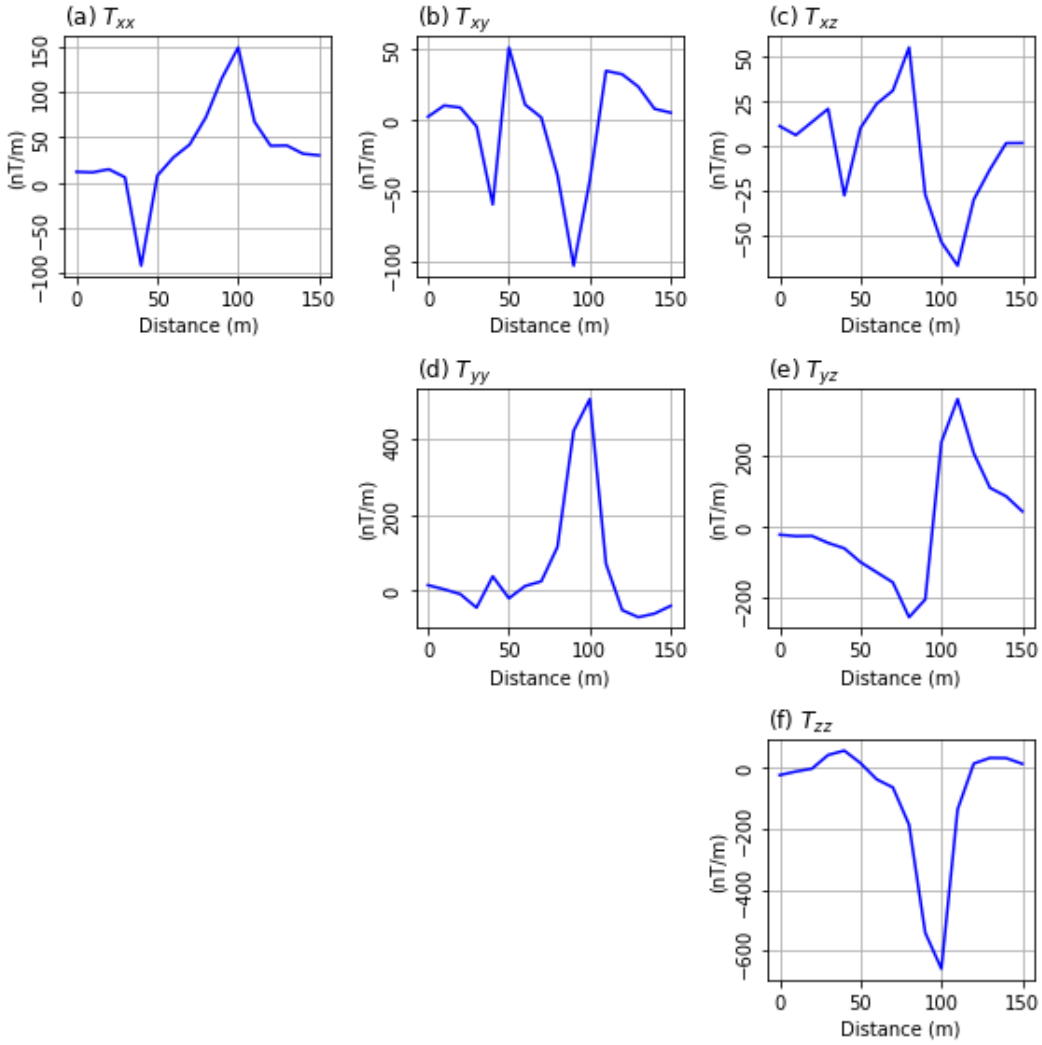

Figure 106 Tensor components measured for line 50 of the Tallawang body.

Figure 107 shows the depth calculations using zero and first order analytic signals. The solutions range between 13 metres and 20 metres with the worst solution being that using As_z in this case. Figure 108 shows the depth calculation using first and second order analytic solutions for a dyke. These range between 35 and approximately 65 metres. However, if the step model is used, since As_2 resembles step data more, then the depth solutions would range between 17 and 32 metres, with the majority of the solutions between 17 and 27 metres. Figure 109 shows the calculation of body width, which ranges

between 20 to 24 metres. All these solutions are consistent with the published depth of approximately 19.9 metres (Clark, 2012) and the width on the geological map in Figure 87 (Schmidt et al., 2004).

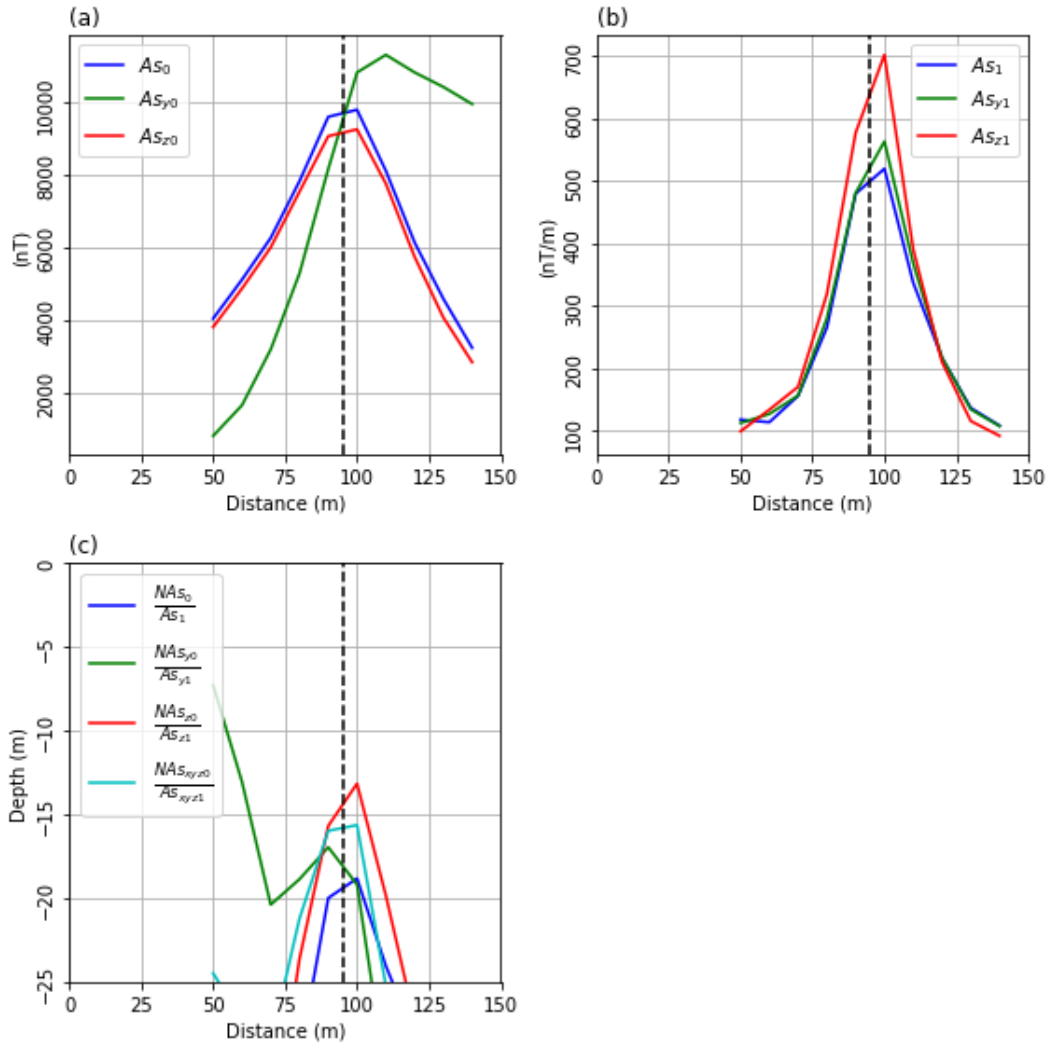

Figure 107 Depth calculation using zero and first order analytic signals. (a) shows the zero order calculations (b) shows the first order calculations (c) shows the depth results over the centre of the anomaly. The dashed line shows the location of the body centre.

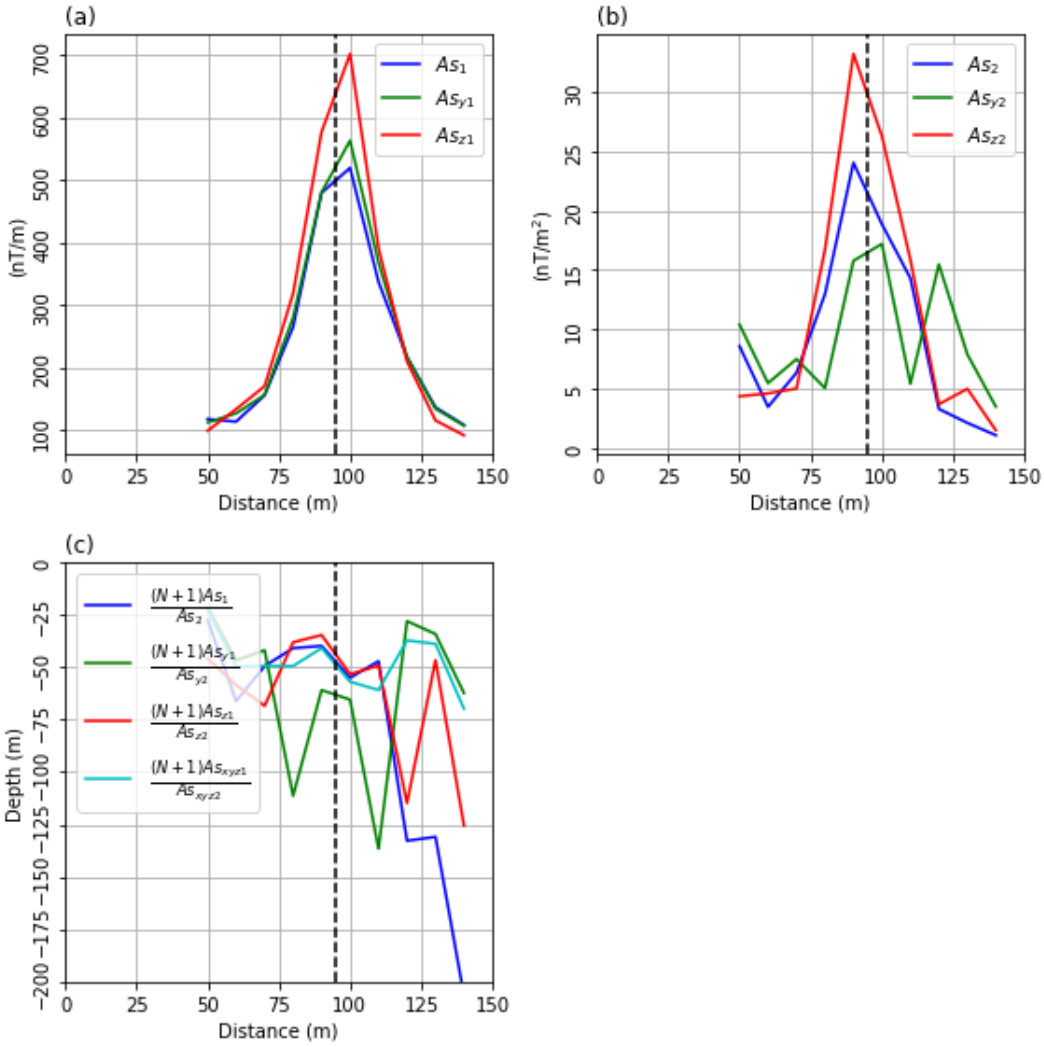

Figure 108 Depth calculation using first and second order analytic signals. (a) shows the first order calculations (b) shows the second order calculations (c) shows the depth results over the centre of the anomaly. The dashed line shows the location of the body centre.

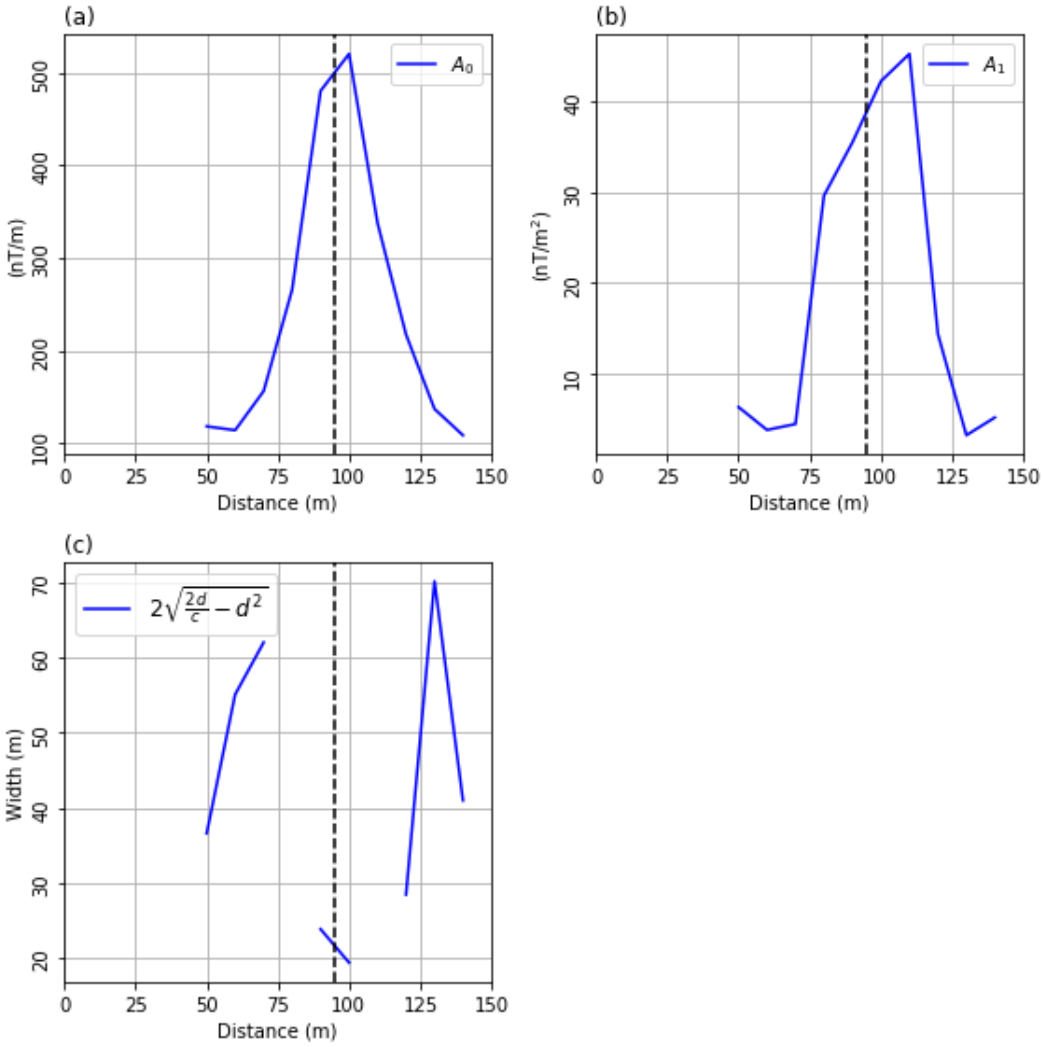

Figure 109 Results for the dyke width calculation. (a) shows the conventional analytic signal, denoted A_0 (b) shows the higher order analytic signal, denoted A_1. (c) shows results for the width calculation. The dashed line shows the location of the body centre. Gaps in results are where no solution was possible.

Figure 110 shows results of the calculated susceptibility and magnetism. A depth of 20 metres and a width of 15 metres was used in the calculation. Susceptibilities range between 0.5 SI and 7 SI over the body, and magnetisations range between 25 A/m and 300 A/m. Once again, many of these values are too large. In this case As_z formulae provide the best solutions, ranging from 1.7 SI to 3.95 SI and 77 A/m to 178 A/m. Given the limited data available for this study, more survey data is needed for more conclusive formulae comparisons.

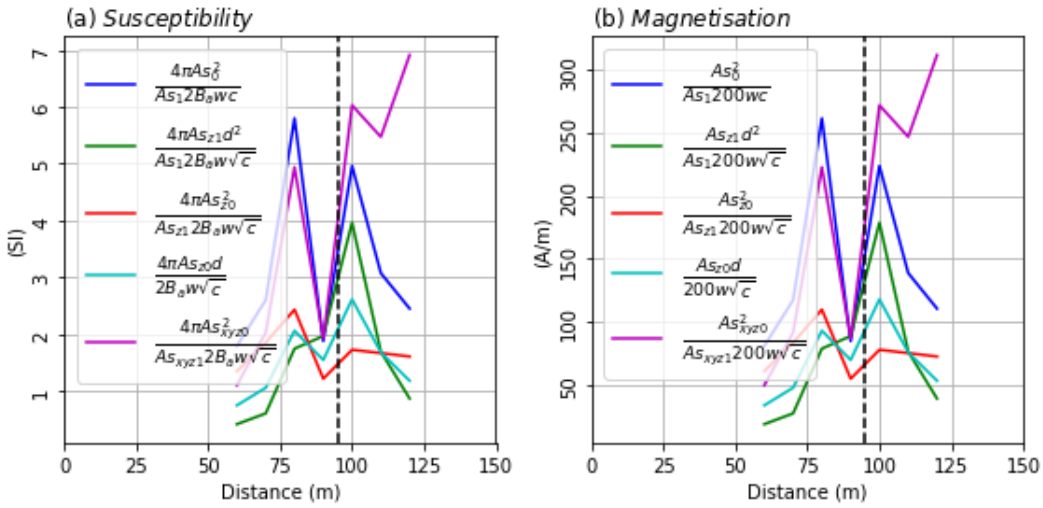

Figure 110 (a) susceptibility calculated from analytic signal formulae (b) magnetisation calculated from analytic signal formulae. The dashed line shows the location of the body centre.

Figure 111 shows quiver plots of the inclinations and declinations of the measured tensor values derived from direction cosines, compared with those of the ambient field. Measured inclinations, for the most part, are once again largely coincident with that of the ambient field. Measured declinations are largely coincident over the peak, and are positive west of the skarn while being negative east of the skarn. This is consistent with line 60's results

The direction cosines shown on Figure 112(a) once again show a complexity in the field over the body. They are slightly different to those in Figure 105(a), emphasizing inhomogeneity, while close enough to show that similar results are obtained for both lines.

Figure 112 (b) shows results for the inclination and declination calculations. The values of susceptibility and magnetisation used were calculated values and were 1.71 SI and 77.56 A/m respectively. The values are chosen from the same magnetisation and susceptibility calculations as for profile 60. These values are smaller than expected, and once gain the inaccuracies are reflected in the calculated inclinations and declinations, which are incorrect. The remanent magnetisation calculated from this is 39.24 A/m, which is close to the true value.

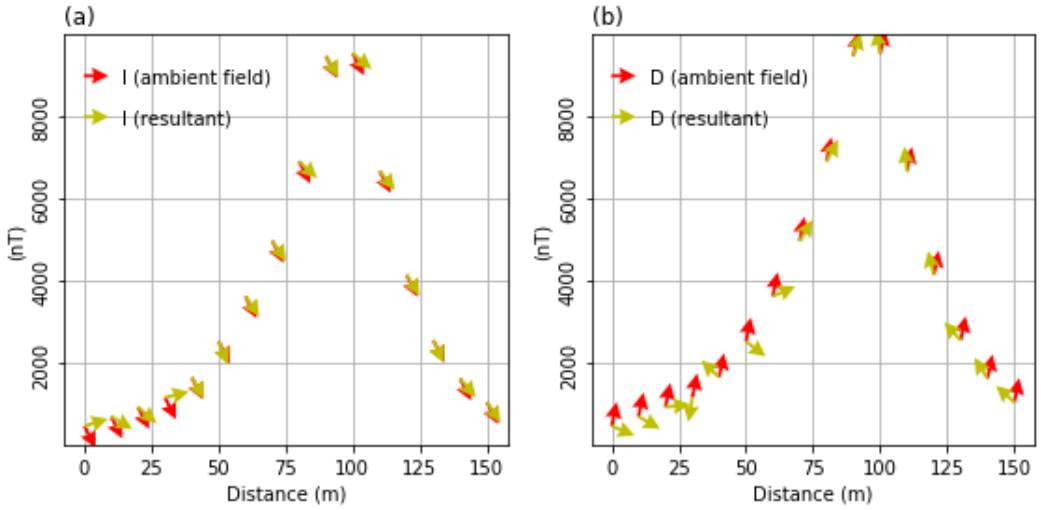

Figure 111 (a) Quiver plot showing a comparison between inclinations from the ambient field and the measured field including remanence. (b) Quiver plot showing a comparison between declination from the ambient field and the measured field including remanence.

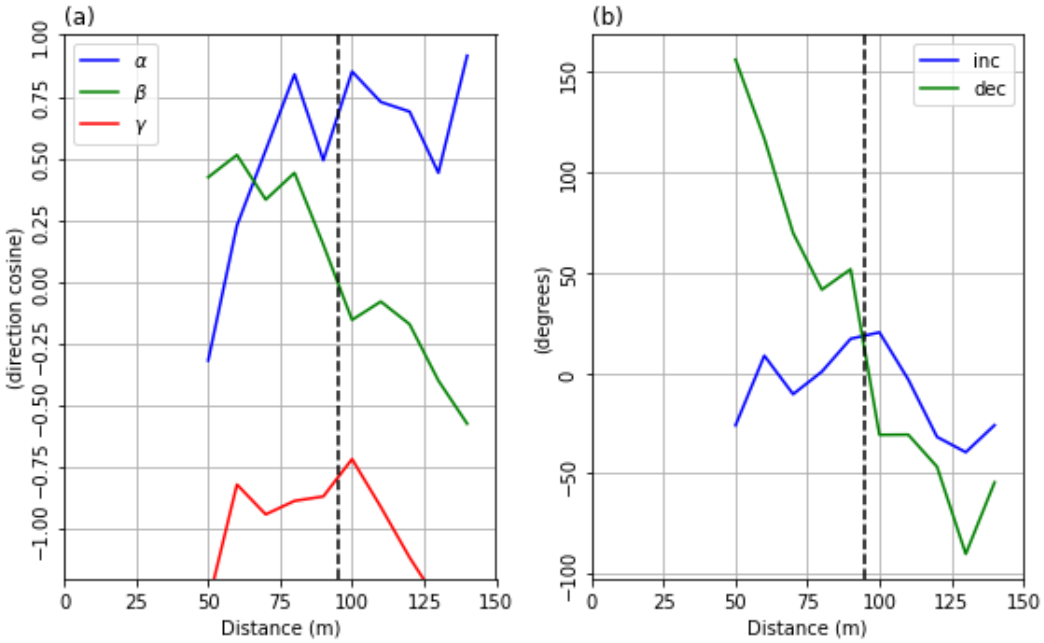

Figure 112 (a) Direction cosines resulting from the model. (b) calculated inclinations and declinations. The dashed line shows the location of the body centre.

6.3.4 Discussion of Results

The testing of source distance methods over Tallawang followed two phases. First, a test model was created in order to understand and predict typical values which should be expected as well as identify any possible limitations in this specific magnetic field and remanence environment. From this, successful width and depth predictions resulted. Predicted magnetisations were lower than in the model, and hinted at possible problems in the remanence calculation to follow. If the real magnetisation and susceptibility values were used, then inclination and declination could be predicted reliably. However, the use of predicted magnetisations gave incorrect inclinations and declinations for remanence. This simple test can be used in any environment to ascertain whether the prevailing geomagnetic and geological conditions will allow robust determination of remanent magnetisation parameters.

In the second phase, two of the Tallawang lines were investigated. Depth and width results were in line with real depth values. Susceptibility and magnetisation solutions were extremely varied and as such remanent parameter estimations were incorrect in both cases. It should come as no surprise that in an area of remanence, the susceptibility solutions would vary since the analytic formula for susceptibility does not account properly for remanence. However, the direction cosine calculations are robust and make no assumptions other than accuracy for measured data. These results showed a complex body which is probably not entirely homogenous with respect to it physical properties. Although the direction cosines are for the total magnetic field, and not the remanence field alone, when compared with the ambient field direction cosines, a sense of remanent field direction, at least in a gross sense, can be gleaned.

A comment should be made regarding the sample separation and the depth calculation. The minimum depth obtainable from a sample spacing of Δx is $r = 2\Delta x - flying\ height$ (Reid, 1980). This equation is applied to any calculation which relies on the relationship between adjacent samples – for example derivatives and power spectra calculations. In this case, since there is no flying height, this implies that with a sample spacing of 10 metres, the minimum depth calculated is 20 metres. This indicates that the depth calculation is operating at the limits of its accuracy for the Tallawang survey. However, should all the tensor field data be entirely measured (i.e. no gradient calculations are performed), then this limitation is not present, since the depth calculations will then not rely on any information (derivative calculations in this case) derived from adjacent samples.

As a final note, even though low Q-ratios are indicative of inaccurate remanence calculations, low Q-ratios are also indicative of a situation where remanence does not play a serious role in the field directions, and so forward modelling is still effective using just ambient field parameters. Therefore the restriction of this technique to higher Q-ratios basically limits it to situations where it is really needed.

6.4 Lichtenberg/Zeerust

6.4.1 Geological Setting and Data

The study area chosen is a small area in the Lichtenberg/Zeerust region in the North West Province of South Africa. Dolomite of the Malmani Subgroup covers most of the area with Ventersdorp Supergroup present along the southern margin. Karoo Supergroup rocks partly cover the older material in the southeast (Figure 113). The Ventersdorp Supergroup consists of basaltic and andesitic rocks and sedimentary sequences and is considered the largest “Large Igneous Province” of the Kaapvaal Craton (Altermann and Lenhardt, 2012).

The Malmani Subgroup forms part of the Transvaal Supergroup that rests unconformably on the Ventersdorp Supergroup. It consists of dolomites that formed in a carbonate platform system between ~2.58 and 2.5 Ga (Eroglu et al., 2015). None of the geological formations in the study area contains substantial magnetic minerals and will therefore not cause anomalies on the magnetic data.

A few diabase dykes and quartz veins have been mapped in the northeast, but most of the large predominantly ENE-striking lineaments were inferred from regional airborne magnetic data.

Borehole data from the Council for Geoscience log database were examined. Locations are shown in Figure 113, with associated borehole logs in Figure 114. The towns of Bakerville and Lichtenburg are shown on the map. In particular two boreholes on the farm Dudfield, drilled in 1974 by Anglo Alpha Cement, are on or very close to the southern magnetic lineament (Dudfield 2 and Dudfield 3), but failed to intersect a dyke. This is an indication that the dykes may be deeper than these two boreholes, the maximum depth of which are 55 metres and 71 metres respectively. The top 44 to 50 meters consist of surficial deposits, followed by the lava of the Ventersdorp Supergroup. There is no mention of dolerite or diabase encountered in the borehole.

Other boreholes were drilled by the same company north of the lineament. These indicate surficial deposits (overburden, calcrete, clay) over dolomite. In the late 1930s, two deep boreholes (L1 and Welverdiend) were drilled in the north. These only indicate dolomite over Ventersdorp lava.

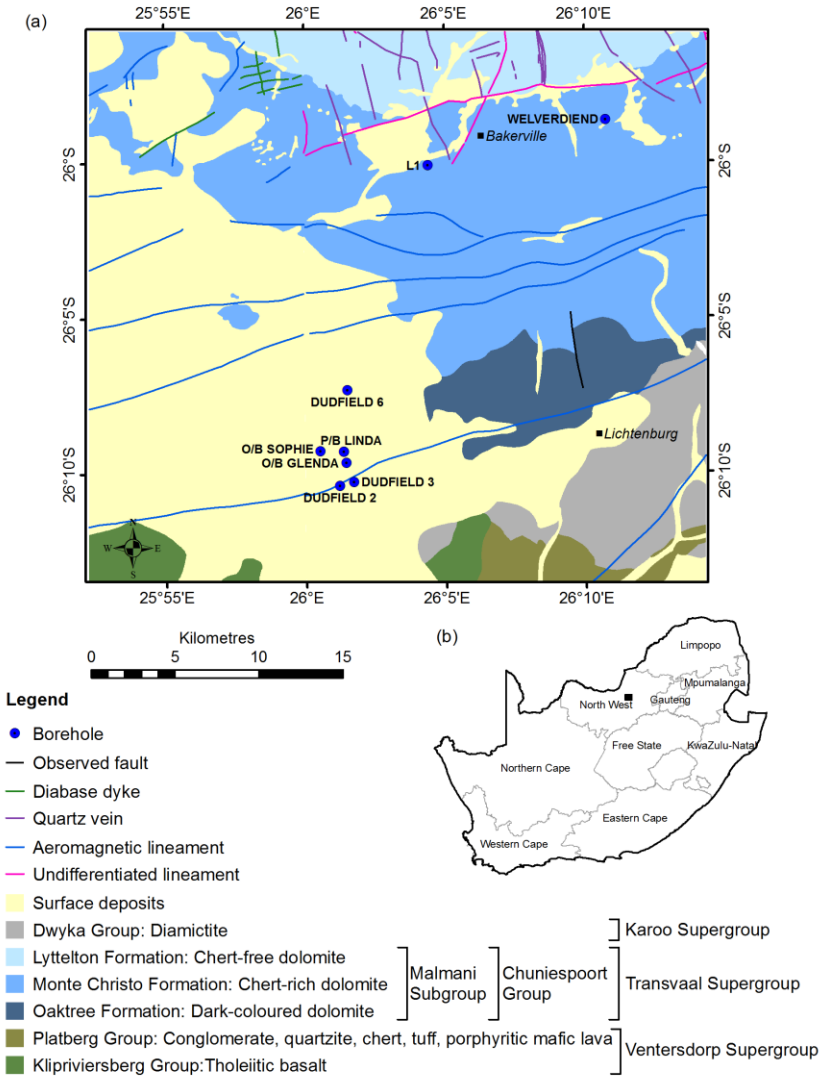

Figure 113 a) Geology of the area (Geological Survey of South Africa, 1993). b) The location of the survey is shown on the map of South Africa.

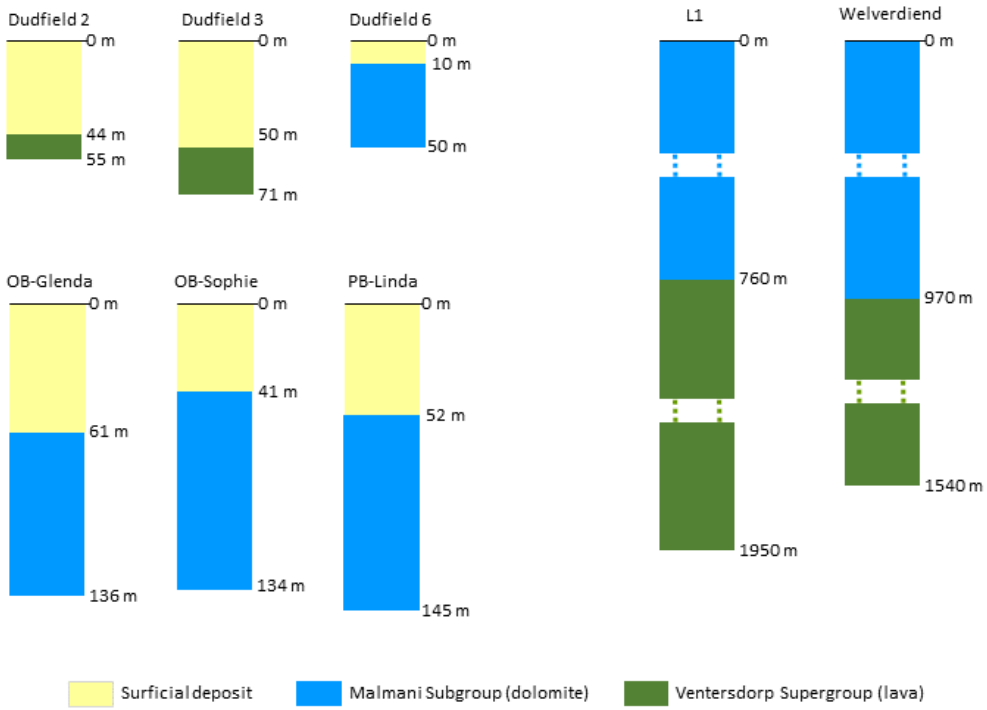

Figure 114 Borehole information over the Lichtenberg/Zeerust Area. No intersections with dykes is visible.

Expanding on the assessment of dyke depths, an area immediately to the north was chosen with clusters of boreholes over similar-striking dykes. Clusters of boreholes were drilled on or very close to the ENE striking lineament. Kareebosch 70 and 71 were drilled by United States Steel International (New York) Inc. in 1971 to depths of 120 m and 93 m respectively. These boreholes only encountered dolomite. The same company drilled N10 to the east. It was only 31 meters deep and only encountered dolomite.

A series of holes was drilled across the lineament on Strydfontein by Armco-Bronne (Pty) Ltd in 1979. The drilling configuration suggests that the lineament may have been targeted, with the deepest hole reaching a depth of 130 m. None of holes found diabase or dolerite.

Since none of the clusters intersected with dykes, it suggests that for the most part the ENE dyke depths may be deeper than the maximum borehole depth of 130 metres.

Only in the east, an isolated borehole (Witrand 1) contained diabase, indicating the intersection with dyke-like material. The hole was drilled by the CGS, and is located on a NW-striking dyke (indicating that it is not the same dyke swarm as the ENE dykes). The

borehole starts in the shale of the Pretoria Group and ends in dolomite. It encountered a 20 m thick diabase intrusion at 30 meters and a 13 m thick intrusion at 56 m depth.

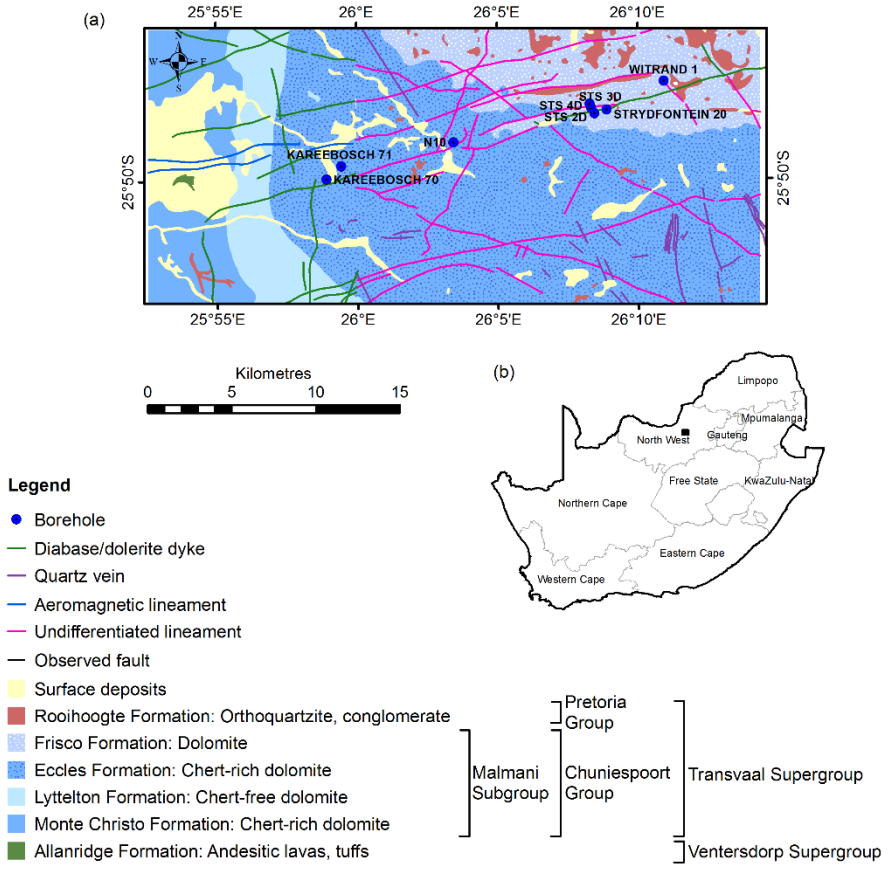

Figure 115 a) Geology of the northern area (Geological Survey of South Africa, 1993). b) The location of the survey is shown on the map of South Africa.

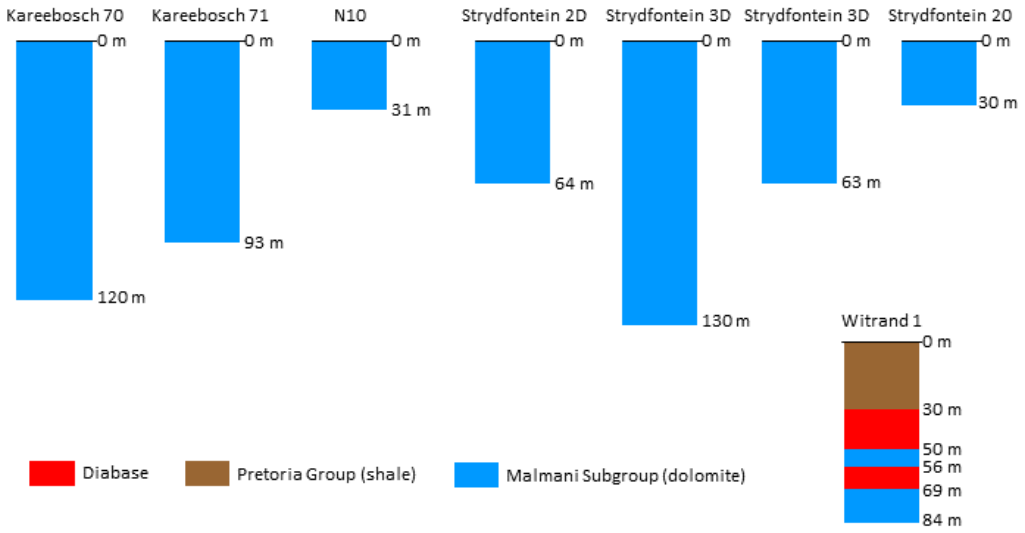

Figure 116 Borehole information over the northern area. Only one dyke was intersected (Witrand 1) as is evident by the presence of diabase.

The data to be used are a section of a single sensor (scalar) magnetic survey flown by the Council for Geoscience in 2003. The line spacing was 120 m and was flown NE to SW using a Cessna 206 at 50 m flying height. The along line sampling interval is generally between 3 to 4 metres, depending on the aircraft speed. The data was gridded at 40 m spacing. It includes total magnetic intensity data (Figure 117) as well as a digital elevation model (Figure 118) which is used for the 3D model creation. The digital elevation model was calculated using GPS and laser altimeter data.

The area chosen was relatively flat in order to minimise survey flying height errors in the acquisition of the data. The magnetic dataset consists of a series of mostly east-northeast striking dykes. These come in three flavours – highly magnetic, moderately magnetic and remanently magnetised (negative anomaly).

A cursory visual inspection of the TMI data (Figure 117) suggests that there are at least two groups of dykes without obvious remanence – two dykes with strong anomalies striking in an east west direction, and other dykes of more varied orientation with lower amplitude anomalies. A third group contains remanence (a negative anomaly in the South African context suggests this) and can be seen on an east west striking dyke running across the centre of the area. This is easier to see on images later in the chapter, for example Figure 120.

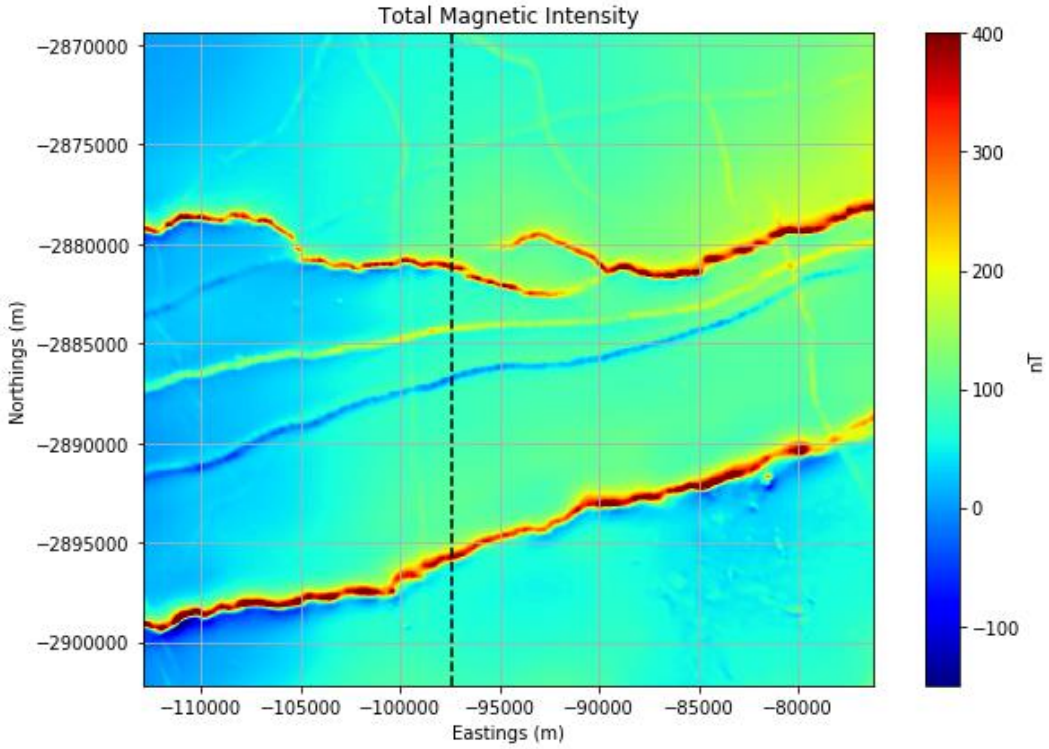

Figure 117 Total Magnetic Intensity of the study area. The dashed line shows the location of a profile shown in Figure 132

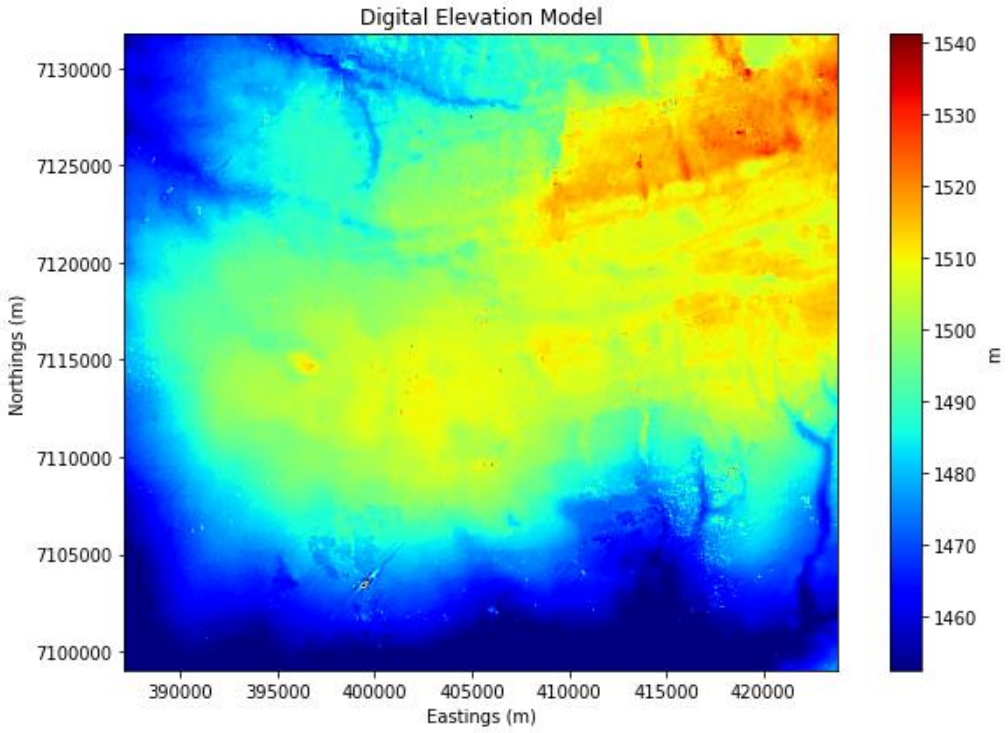

Figure 118 Digital Elevation Model of the study area, obtained from the magnetic survey.

6.4.2 Data Preparation

It is standard in FFT processing to subtract a polynomial surface from the data being processed in order to ensure more stable solutions. It will also remove or minimise the effect of deeper large bodies from the estimation of shallow sources. This, and the resultant magnetic field, is shown in Figure 119 and Figure 120. The surface itself can be estimated directly from the data by constructing a low degree Vandermonde matrix, and solving using least squares. In this case a second degree Vandermonde matrix was constructed.

Figure 121 shows the derived datasets for the components and tensor components of the magnetic field. The x direction is east-west, and the y direction is north-south (ENU convention). The geomagnetic field strength during the survey was 28 280 nT, with an inclination and declination of -64º and -17º respectively.

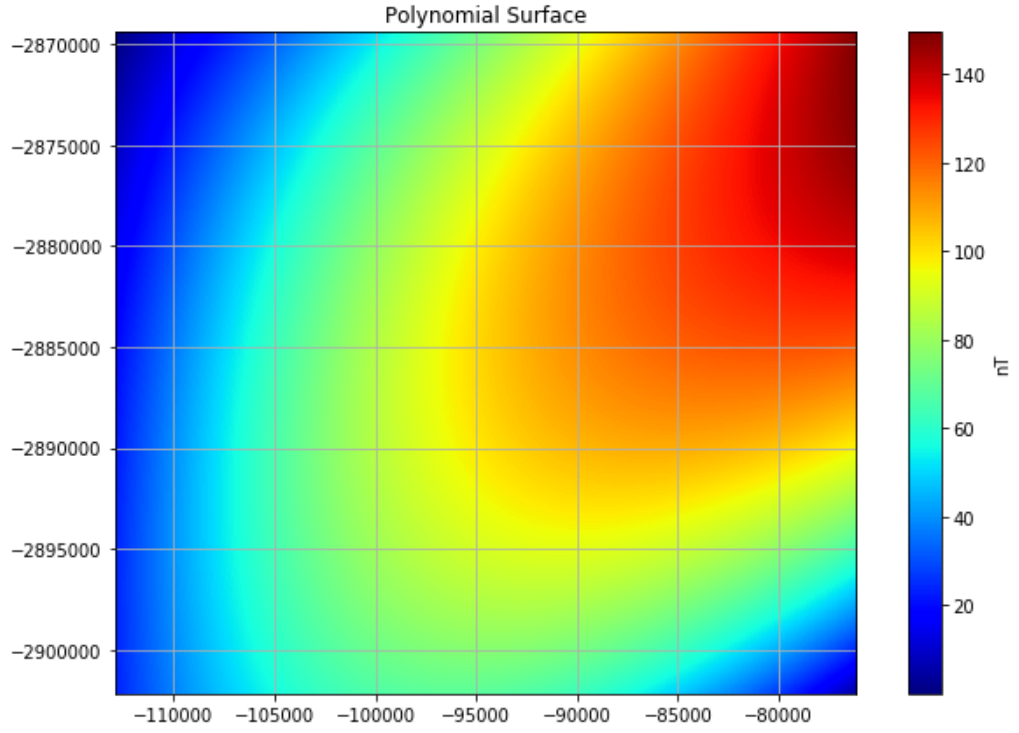

Figure 119 Polynomial surface used to prepare magnetic data for FFT

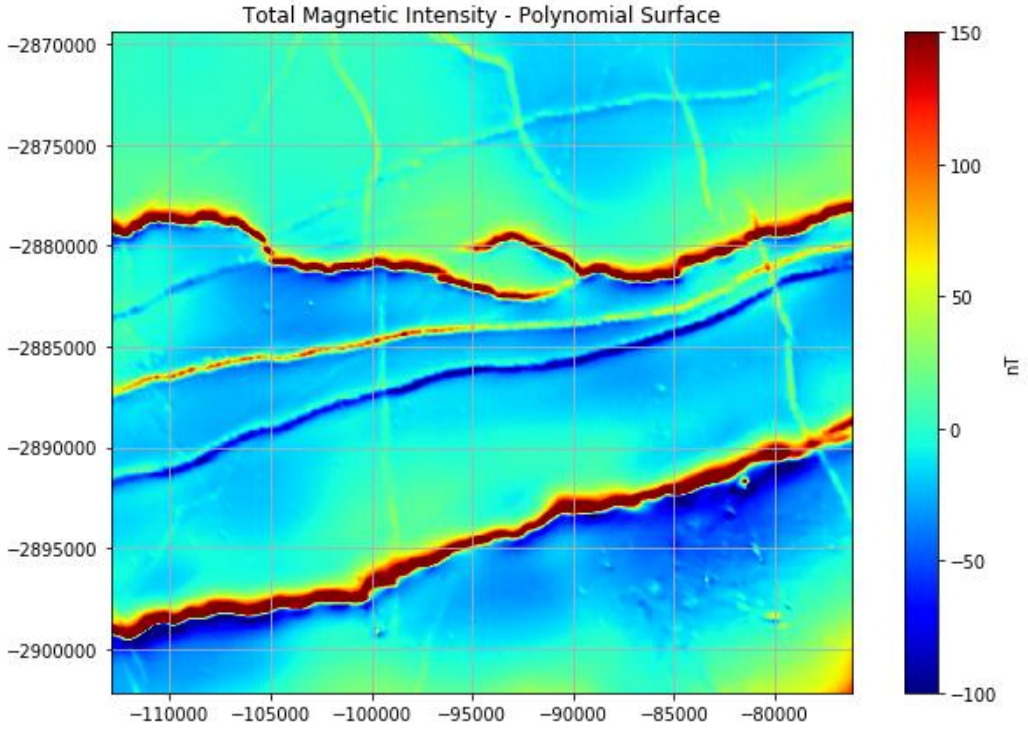

Figure 120 Resultant magnetic field once polynomial surface is subtracted.

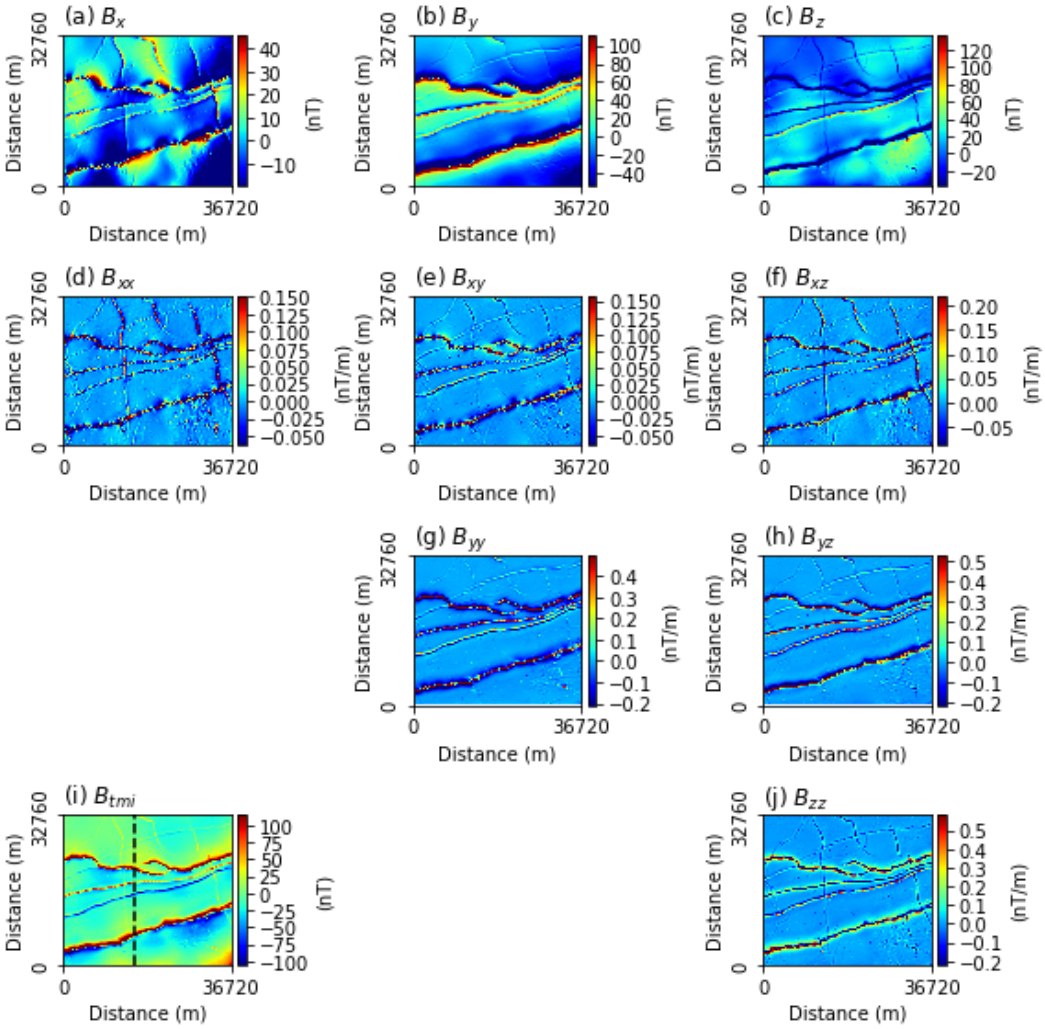

Figure 121 Derived tensor components from the TMI. The dashed line in (i) shows the location of a profile shown in Figure 132

6.4.3 Methodology and Results

The methodology to create 3D models from the source distance calculations is straightforward. First, the extents and resolution of the model are set. The extents are in the x, y and z dimensions and the resolution should be relevant to the detection limit of features to be delineated.

Secondly, a number of datasets can be calculated. The depths to sources (r) are calculated using formulae described in section 4.5. Since these depths are actually the distance from the observation to the top of the source, they are corrected for flying height. These are all calculated at the resolution of the input data, which in this case is 40 m. This ensures that aliasing problems do not affect the integrity of the depths. The minimum depth obtainable from a sample spacing of Δx is $r = 2\Delta x - flying\ height$ (Reid, 1980). Therefore, for a survey at a flying height of 50 m, this implies a minimum depth of 30 m.

Dyke widths are estimated using equation (4.113). These widths are dependent on the depth solutions obtained. The mean widths using either conventional analytic signal depths (equation (4.75)) or component analytic signal depths (such as equations (4.74) and (4.85)) was approximately 133 metres with a standard deviation of 129 metres, showing a wide variability of possible widths. To simplify the calculations for rectangular prism modelling, a representative width was chosen. Given the tendency of the width estimation to overestimate widths of deeper sources, this was set to 100 meters as a starting point. It can be refined later through a forward modelling phase, if necessary.

The susceptibilities (k) of the sources are calculated using formulae described in 4.5.4, and assuming a dyke width of 100 m (which is the resolution of the model). Even though the widths have been fixed and may result in inaccurate susceptibilities, it should be noted that the overall width-susceptibility relationship means that from a forward modelling point of view, this inaccuracy in susceptibility is compensated by the width and the fitting of the anomaly should still be achieved.

Figure 122 and Figure 123 show the analytic signal datasets used in the depth to source calculations. Figure 124 shows the results of the depth to source calculations. A value of N=1 was used for the depth calculation, which used equation (4.75) defined by Cooper (2015). Figure 125 shows the depth calculations using (4.74). Note that only the values along the dykes (typically closest to zero) are valid depths. This is because any source distance equation represents a distance to the closest source, which only becomes a depth over the source. Similarly, in Figure 126 only the susceptibility values over the peaks are valid. For the susceptibility calculations, $w = 100\ m, F = 28\,280\ nT,\ I = 63.88°, D = -17.18°$.

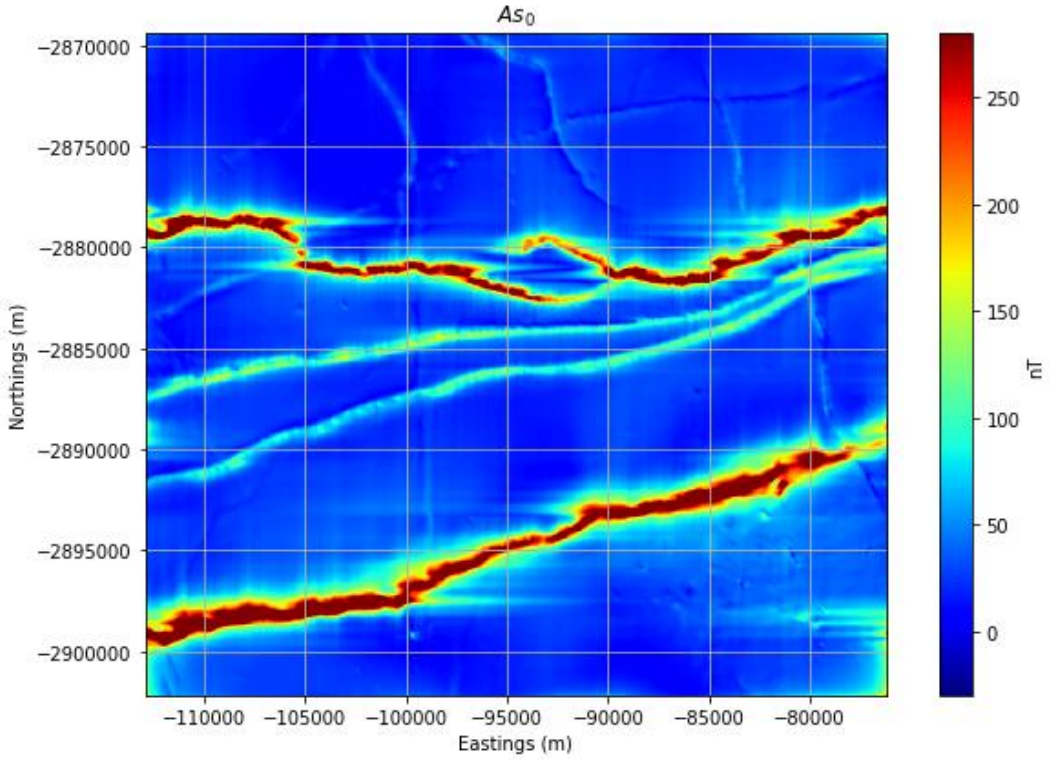

Figure 122 As_0 dataset used in calculations for r and k.

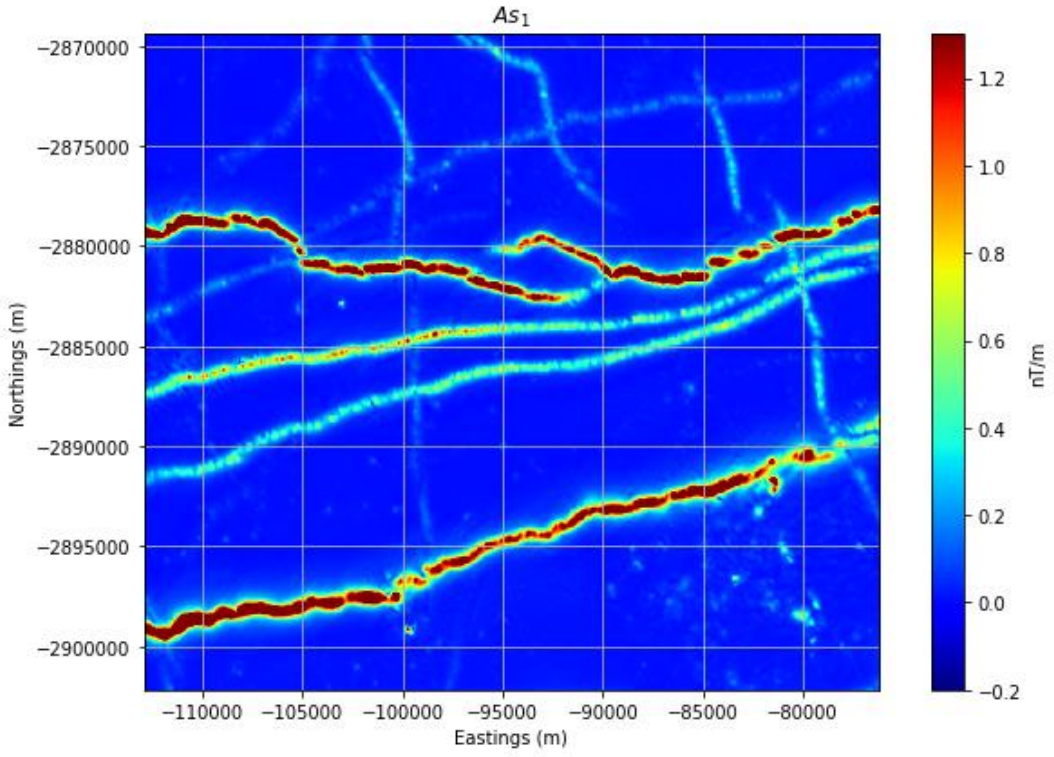

Figure 123 As_1 dataset used in calculations for r and k.

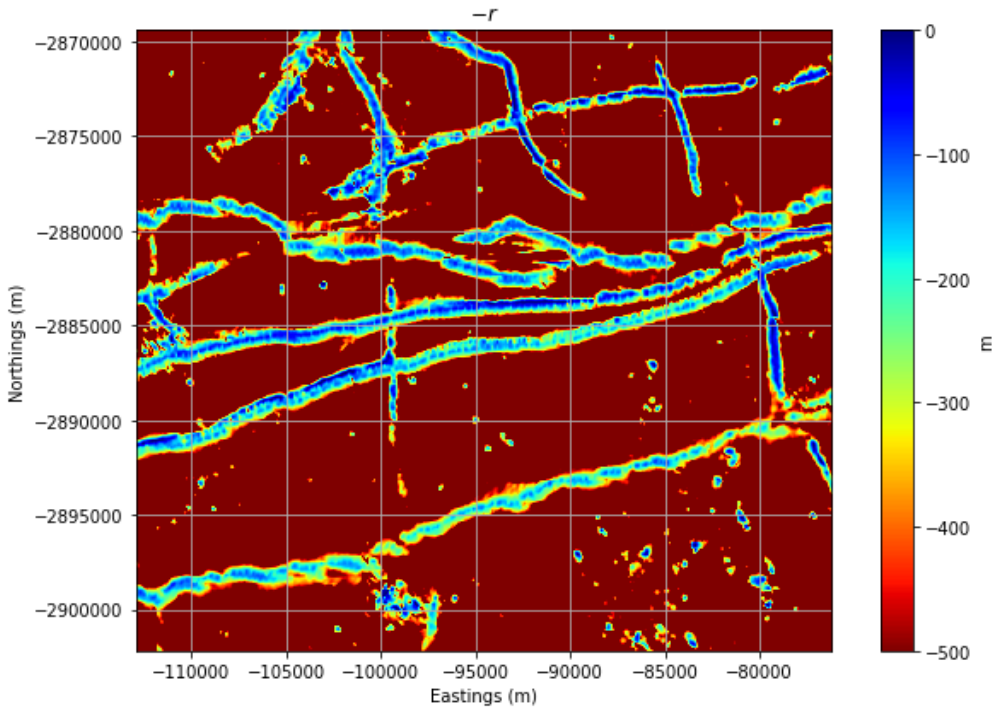

Figure 124 Depth to source results using (4.75) (Cooper 2015). Only values over peaks are valid. Source depths have been corrected for flying height.

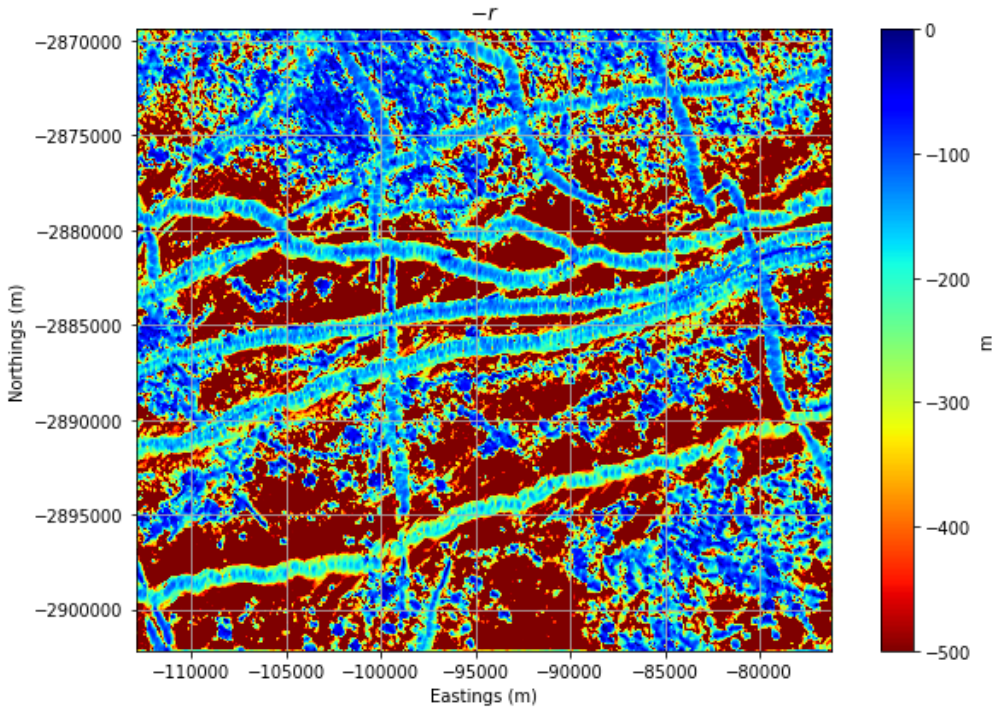

Figure 125 Depth to source results using (4.74). Source depths have been corrected for flying height.

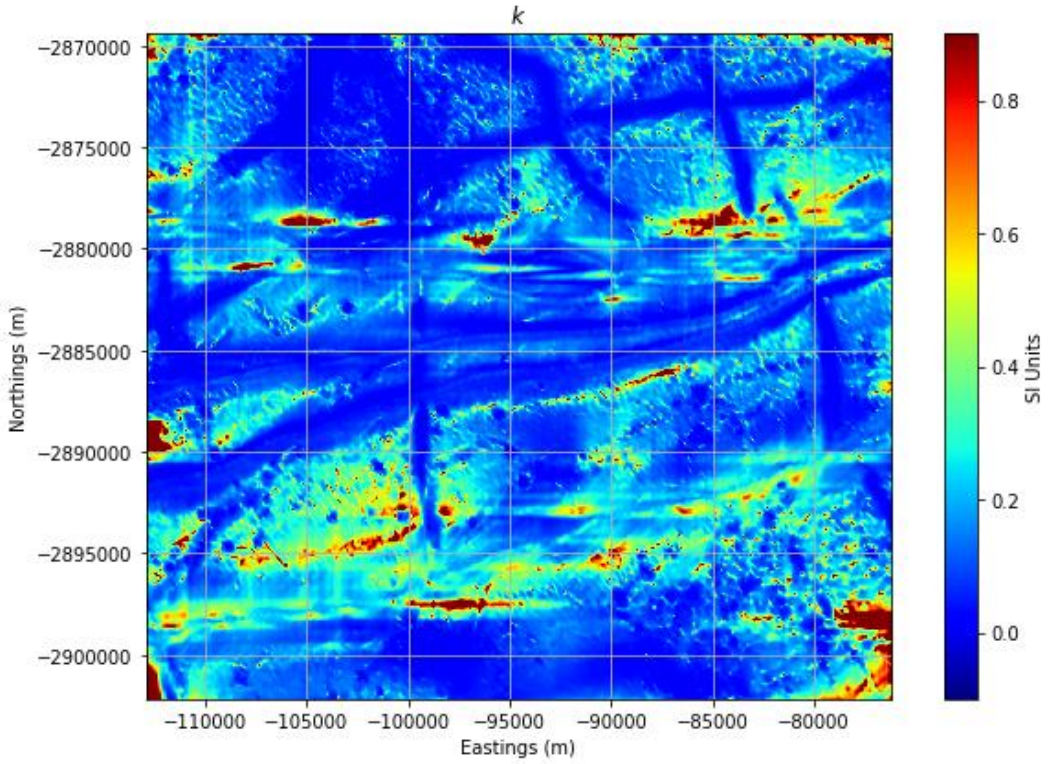

Figure 126 Calculated values for susceptibilities. Only values over dykes are valid.

For comparison, Figure 125 shows results using equation (4.74) as opposed to equation (4.75) in Figure 124. The critical difference is that in Figure 124 a zero order analytic signal is used and produces cleaner results. This is due to higher orders of analytic signal being more susceptible to noise.

A more extensive comparison is shown in Figure 127. Figure 127 a) and d) show depths calculated using As_x (equations (4.72) and (4.86) respectively) and therefore highlight structures in the *y*-direction, whereas Figure 127 b) and e) show depths calculated using As_y (equations (4.73) and (4.87) respectively) and highlight structures in the *x*-direction. Equations (4.74), (4.75) and (4.85) (As_z and As) do not have any azimuthal bias, as shown in Figure 127 c), f), and g). Features that are common to results from the use of equations (4.74), (4.75) and (4.85) are probably due to valid sources. The different solutions that are obtained from the tensor data can therefore can be used for quality control. Since the numerical calculation of the source distances by the different methods is computationally trivial, they should therefore be used together in interpretation projects (Cooper and Whitehead, 2016). Figure 127 h) shows the susceptibility results (calculated using equation (4.109)).

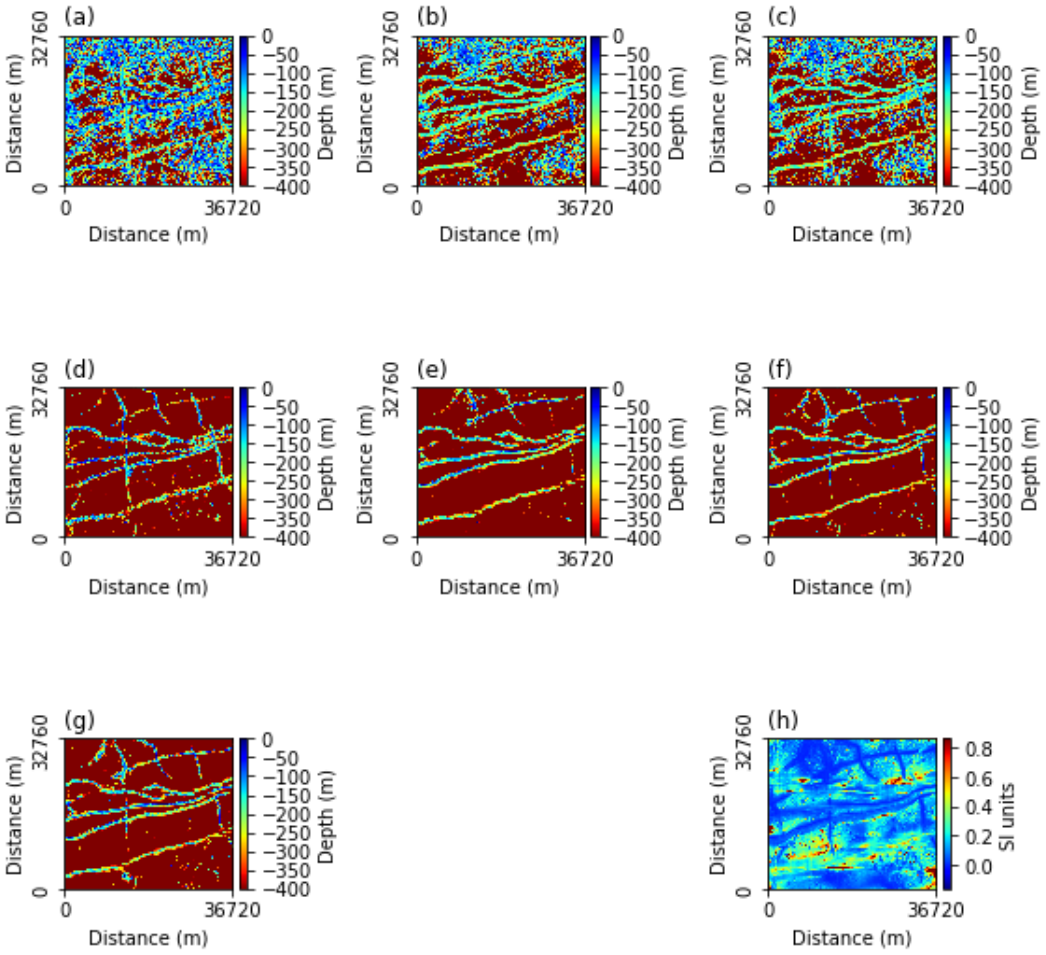

Figure 127 a), b) and c) show the source-distance results calculated from equations (4.72), (4.73) and (4.74). d), e) and f) show the depth results from equations (4.85), (4.86), (4.87). g) Source-distance from equation (4.75). h) Susceptibility from equation (4.109). A dyke width of 100 meters was used. The flight height of 50 m was removed from the distances.

All algorithms show similar results for the source depths. The depths have a mean value of -165 metres with a standard deviation of 83 metres, which is geologically acceptable since this area is covered by surface deposits (a flight height of 50 metres was removed from the distances) However, equations (4.72), (4.73) and (4.74) (Figure 127a), b) and c)) use higher order derivatives in their calculation, and are therefore more sensitive to noise, but conversely they should be less sensitive to interference (Cooper, 2016; Cooper, 2015). The lower orders of As used in equations (4.85), (4.86), (4.87) and (4.75) (Figure 127d), e), f) and g)) give cleaner results. In either case, solutions can be located using As only and extracting the depth solution from the relevant location.

Following this, a peak finding routine was applied to As to obtain the optimal locations for sources. Results are shown in Figure 128. Notice that depending on the threshold, locations which are not valid sources (i.e. plugs as opposed to dykes) may be detected.

An example of this is highlighted by a red circle in Figure 128. Such outliers can either be manually edited out, using a GIS for example, or can be left in, with solutions corrected in the forward modelling phase.

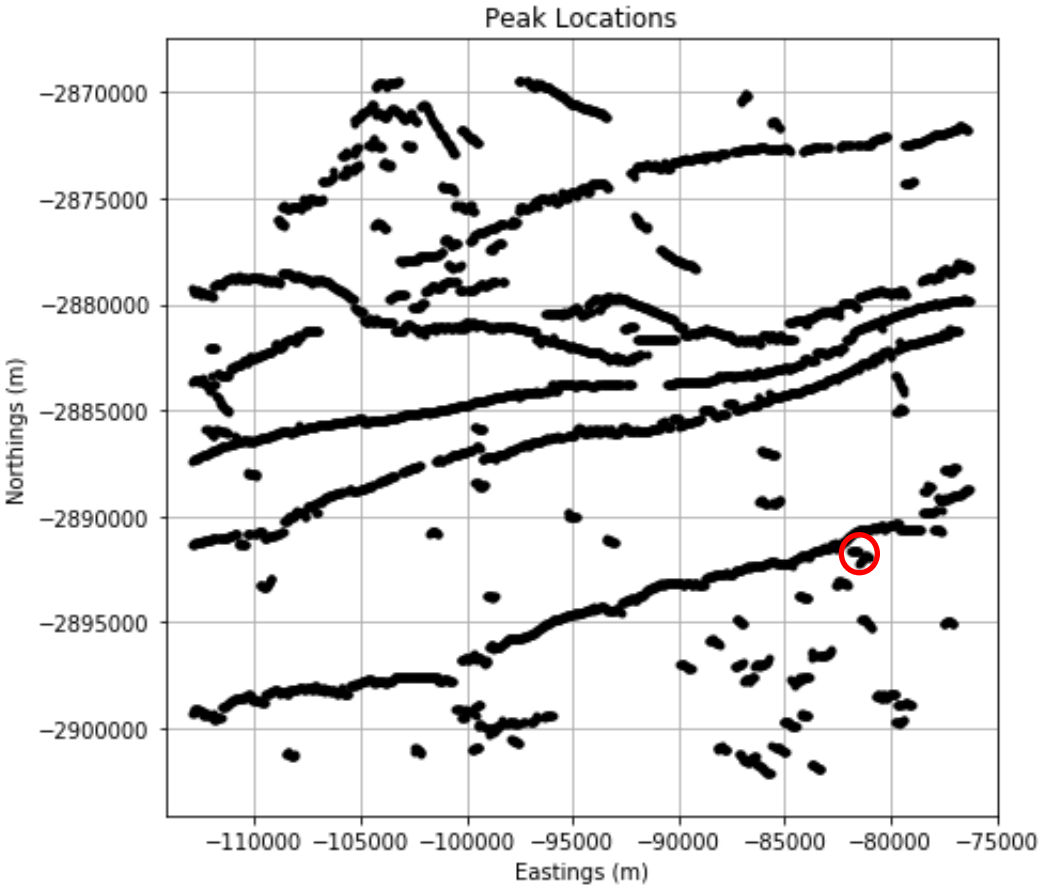

Figure 128 Filtered results after DBSCAN, The red outline indicates an example of a possible plug-like body which has been misclassified.

Now, each point in Figure 128 will have a different susceptibility solution. This is shown in Figure 129. This may not be desirable for a number of reasons:

1) Some of the solutions may be outliers which are not realistic and have been generated due to noise and/or aliasing in the datasets
2) The number of solutions is akin to having a separate lithology for each solution – complicating forward modelling enormously. Depending on hardware available, this may not be a problem.

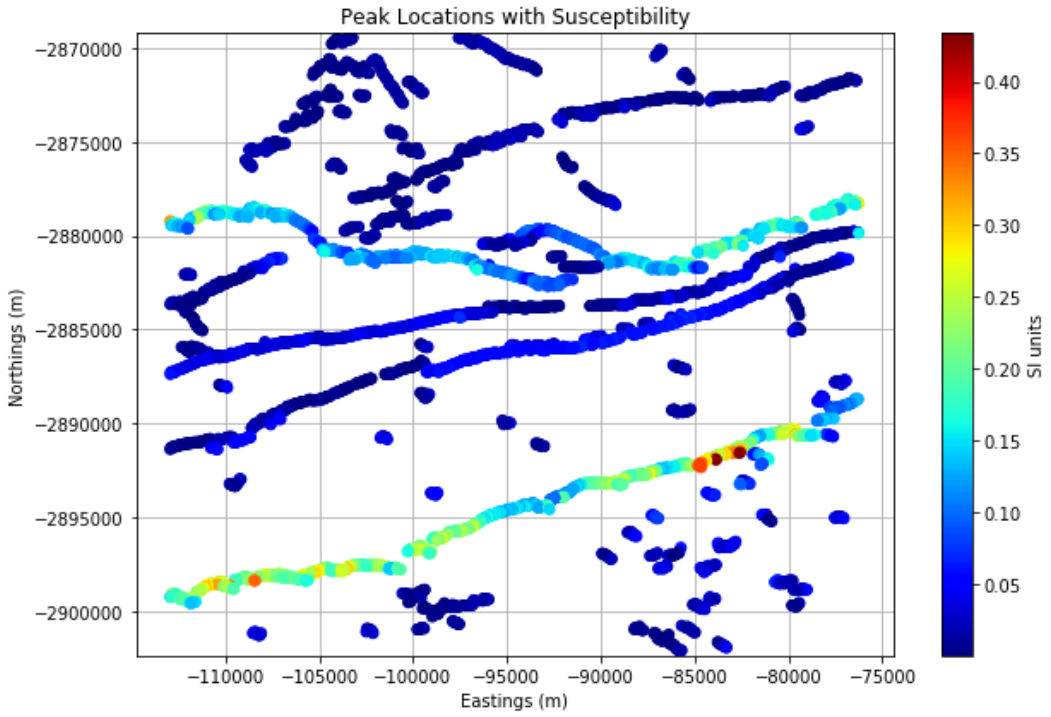

Figure 129 Peak locations with susceptibilities. The susceptibilities are displayed to illustrate the general susceptibility regimes within the dykes, for input into determining how many general susceptibility classes are in the data.

Therefore, to simplify the number of susceptibility solutions, conventional k-means cluster analysis (MacQueen, 1967) can be used to establish susceptibility groupings in the data. The usefulness of this is that it can be done to 1-D data as well. The class centre points become the susceptibilities used in the model and are assigned to the reduced depth solutions.

Examination of the magnetic data shows at least two different types of non remanent dykes – one with high susceptibility, and others with moderate susceptibility and one remanent dyke. However, examination of the susceptibility distribution shows two prominent groups, most likely because the susceptibility calculations cannot account for remanence and the remanent dyke had similar calculated susceptibilities to one of the non-remanent groups. Because of this, the classification was performed with two classes. Class 1 had a centre point of k = 0.023 and class 2 had a centre point of 0.175. The distribution of susceptibilities with classes overlain can be seen in Figure 130. Note that a log normal distribution can be used when displaying magnetic properties.

These depth solutions are then converted to the 3D voxel based model (Figure 131), where they can be used in forward modelling (Figure 132) to further improve the results. The forward modelling can be in conventional or tensor form. Figure 131 in particular

shows that the susceptibility groups shown in red and blue occur coincide exactly with corresponding dyke groups visually seen in Figure 120.

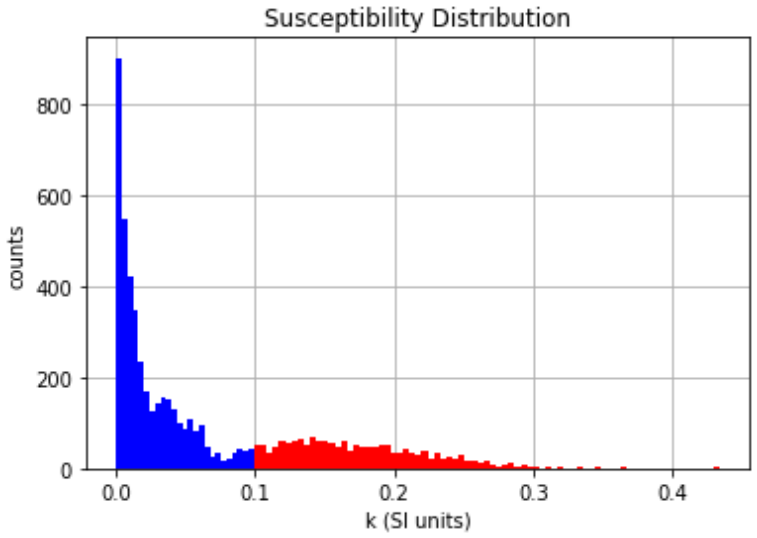

Figure 130 Susceptibility distribution with classes in colour. Class1 is in blue and class 2 is in red.

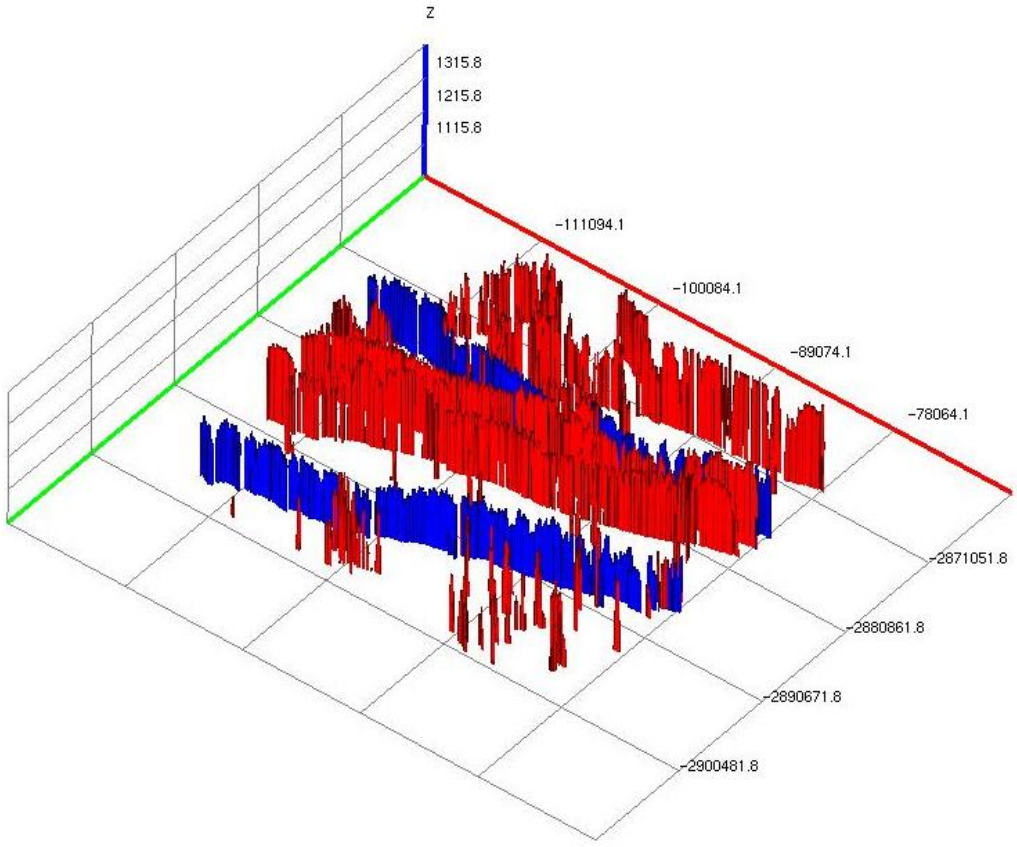

Figure 131 3D model of dykes. All coordinates are in metres.

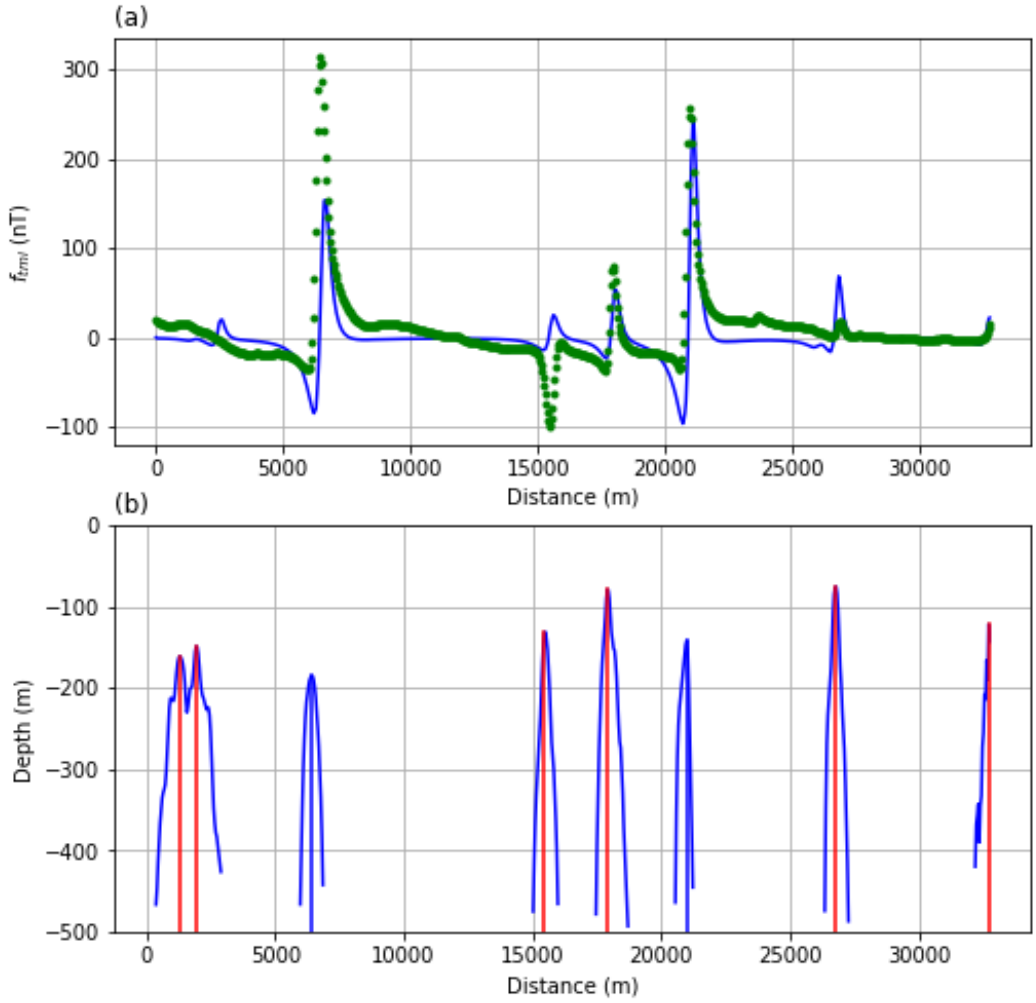

Figure 132 a) A north-south profile extracted from the centre of the study area (orange) as shown in Figure 117, and forward model response of the model shown as a solid line, (blue) using the results of the source-distance and susceptibility calculations. b) Model used to generate the synthetic magnetic data shown in (a). The red dykes have a susceptibility of 0.023 SI and the blue dykes have a susceptibility of 0.175 SI. The results of the source-distance calculations are overlain.

The results of the modelling (Figure 132)) confirm the validity of both the depths and susceptibility estimates. These susceptibility values are reasonable for dolerite dykes. Discrepancies between the source distance results and the forward modelled results may be due to the susceptibility values being over simplified, incorrect dyke widths, or the presence of remanent magnetisation. This does not affect the source distance calculations, only the susceptibility estimates.

Therefore, the process of taking depth solutions dataset, extracting the correct solutions, identifying key features (such as dykes as opposed to point/plug like features) and converting that information into a model is straightforward. Initial dykes/edges will be vertical, but this just forms a reasonable starting point for further forward modelling.

A more heterogeneous susceptibility solution can be obtained by simply adding more classes in the classification phase of the process. As an example, k-means cluster analysis of the susceptibilities was performed on 10 classes. The results are of the susceptibility distribution are shown in Figure 133.

Figure 134 shows the results. In general, all solutions are better. With small magnetic peaks being better represented. The second large peak at 21 000 metres has a slightly worse solution. The remanent peak just after 15 000 meters is interesting, since it is larger than before but mirrors the negative remanent anomaly better. Since the analytic signal of the negative remanent anomaly would be positive, this shows how the routine is trying to fit a non-remanent version of this peak.

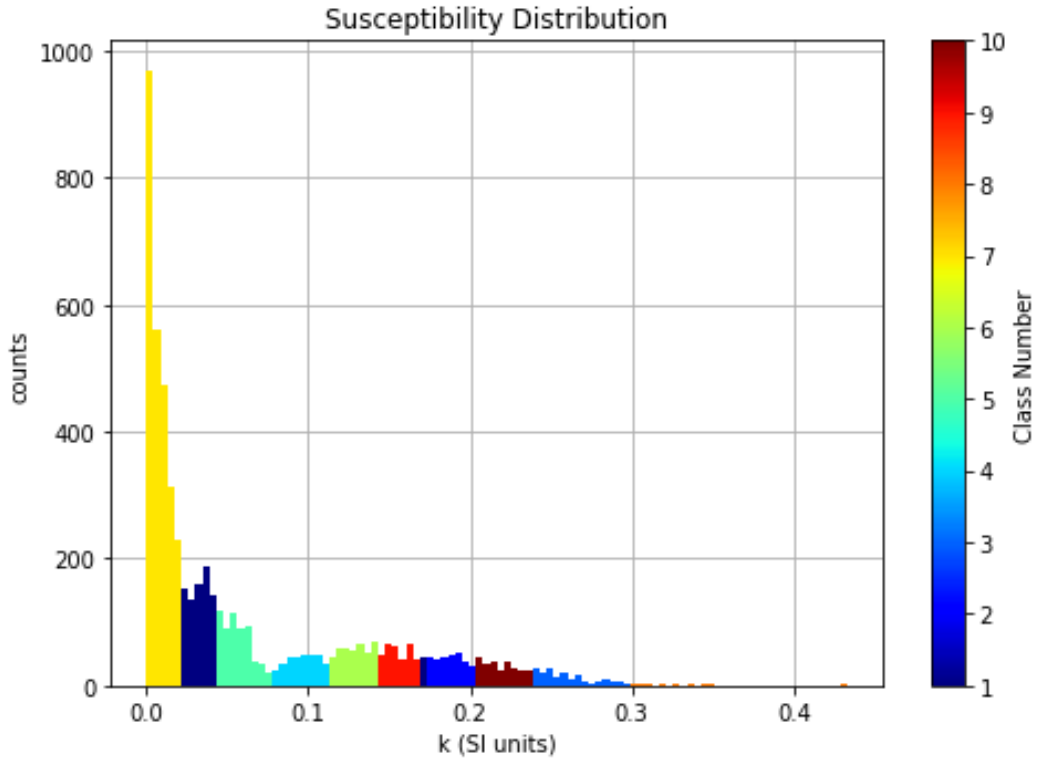

Figure 133 Susceptibility distribution of cluster analysis performed on 10 classes.

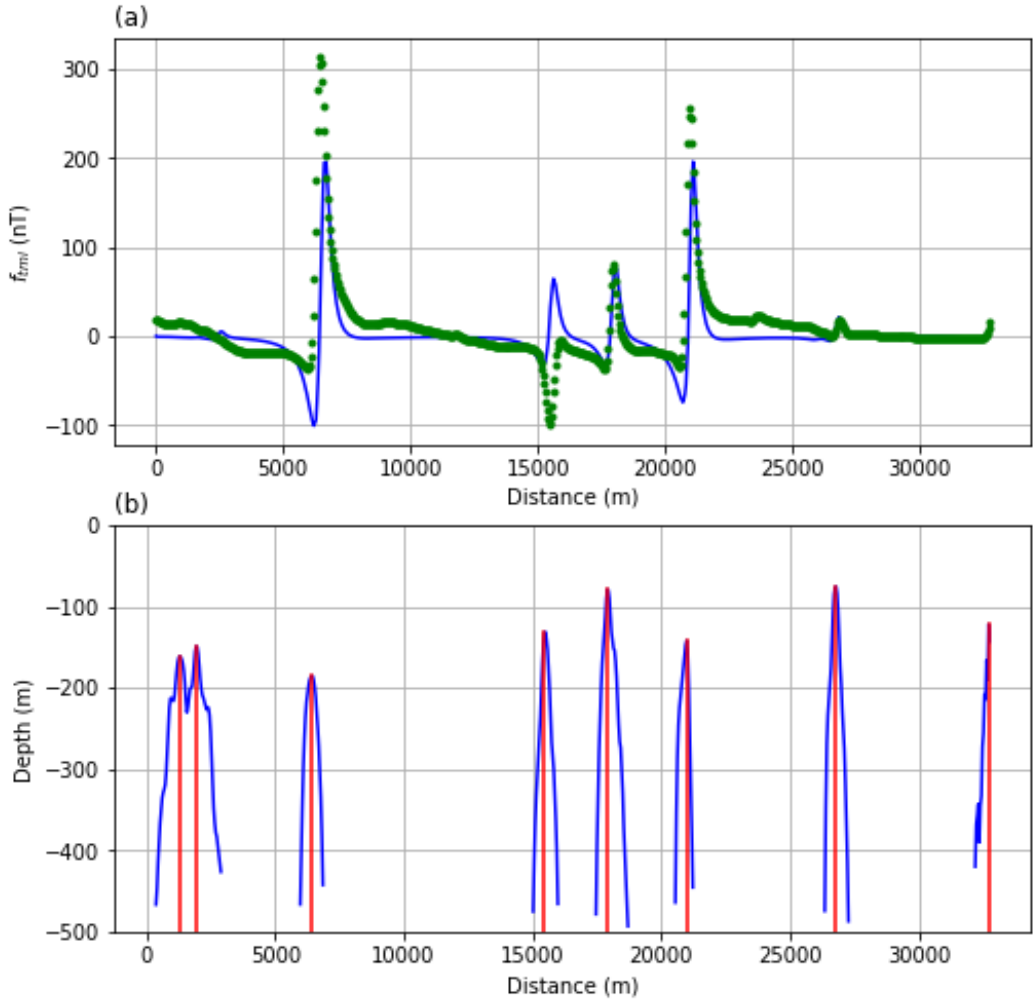

Figure 134 a) The same A north-south profile extracted as in Figure 132, from the centre of the study area (orange) as shown in Figure 107, and forward model response of the model shown as a solid line, (blue) using the results of the source-distance and susceptibility calculations. b) Locations of dykes are shown with depth solutions.

CHAPTER 7 CONCLUSIONS

The forward modelling of voxel data, both in conventional and tensor form, has been shown to not only be viable but also has efficiencies which are as good, if not better (when taking editing into account) than non-voxel techniques. Rectangular prisms in particular allow for a number of optimisations in the calculation of the forward model.

However, the reduction of modelling time is best made with a reasonable starting point to the modelling process. Source distance calculations provide an efficient form of inversion with which to provide this starting model. The application of tensor data to source detection techniques has been shown to be viable and agrees well with previous work done by Cooper, (Cooper, 2014c; 2014b, 2015; Cooper and Whitehead, 2016). The advantage is potentially lower noise in the calculation of the analytic signal (As_1), due to using direct measurements rather than derived derivative products.

Remanence was examined and equations for direction cosines relating the total magnetisation were developed. These cosines are model independent and are extremely useful in assessing whether remanence exists in an area (even if only subtly) as well as giving an indication of the possible direction of remanence in order to achieve the current field direction. When applied to real tensor data, the solutions also give an indication of the variability in the remanent field over the body, indicating the possibly homogeneity or heterogeneity of remanent material.

From the direction cosines, equations for remanent magnetisation, inclination and declination were developed. The limitation on these equations are that they are meant for dykes, and are more effective at higher Q-ratios. In addition, the susceptibility derived from the analytic signal is not well suited for this, since it is based on a field which includes remanence. Therefore, it may give an indication of susceptibility, but in a remanent environment an external measurement of susceptibility will be needed.

The derivation of tensor datasets from total magnetic intensity data showed that the process not only is viable, but also achieves good results. However, one restriction is that remanence is not captured in such a derivation. The extraction of valid source distance solutions from raster data is straightforward and allows fast creation of the 3D starter model for the area, from which improvements can be made through further forward modelling. Calculations of susceptibility are important for accurate forward modelling of these datasets, and are most effective in non-remanent areas. The use of direction cosine calculations to detect remanence is not possible in the case of synthetic data,

since the basic premise of such data is that one constant set of direction cosines was used to create the dataset in the first place.

The testing of source distance techniques on tensor data showed both strengths and limitations. In spite of the fact that the tensor dataset tested (Tallawang) was not over a perfect dyke, the calculations for depth and width proved robust with solutions in the expected range. The low Q-ratio and uncertainty in susceptibility contributed to non-optimal solution for total magnetisation and from this, remanent magnetisation, inclination and declination. A synthetic model describing the Tallawang skarn proved though, that should the total magnetisation and susceptibility be accurately known, it is possible to accurately derive remanent magnetisation, inclination and declination. Direction cosine solutions over the body showed a degree of complexity in the remanence, possibly due to the presence of magnetite in lenses, thereby suggesting a complex composition.

REFERENCES

Altermann, W. and Lenhardt, N., 2012. The volcano-sedimentary succession of the Archean Sodium Group, Ventersdorp Supergroup, South Africa: Volcanology, sedimentology and geochemistry. *Precambrian Research*, 214–215, pp.60–81.

Andreasen, G.E. and Zietz, I., 1969. *Magnetic fields for a 4x6 prismatic model*. Geological Survey Professional Paper 666. Washington, USA: United States Government Printing Office, 228 pages.

Argast, D., Fitzgerald, D.J., Holstein, H., Stolz, R. and Chwala, A., 2010. Compensation of the full magnetic tensor gradient signal. In: *ASEG 2010*. Sydney, Austrailia, pp.1–4.

Barbosa, V.C.F., 1994. Generalized compact gravity inversion. *Geophysics*, 59(1), pp.57–68.

Barnett, C.T., 1976. Theoretical modelling of the magnetic and gravitational fields of an arbitrarily shaped three-dimensional body. *Geophysics*, 41(6), pp.1353–1364.

Bastani, M. and Pedersen, L.B., 2001. Automatic interpretation of magnetic dike parameters using the analytical signal technique. *Geophysics*, 66(2), pp.551–561.

Baykiev, E., Ebbing, J., Brönner, M. and Fabian, K., 2016. Forward modeling magnetic fields of induced and remanent magnetization in the lithosphere using tesseroids. *Computers and Geosciences*, 96, pp.124–135.

Beiki, M., 2010. Analytic signals of gravity gradient tensor and their application to estimate source location. *Geophysics*, 75(6), pp.I59–I74.

Beiki, M., Clark, D.A., Austin, J.R. and Foss, C.A., 2012. Estimating source location using normalized magnetic source strength calculated from magnetic gradient tensor data. *Geophysics*, 77(6), pp.J23–J37.

Bhattacharyya, B.K., 1964. Magnetic anomalies due to prism-shaped bodies with arbitrary polarization. *Geophysics*, 29(4), pp.517–531.

Billings, S., 2012. *Superconducting Magnetic Tensor Gradiometer System for Detection of Underwater Military Munitions (MR-1661)*. Sky Research, Inc., 108 pages.

Blakely, R.J., 1995. *Potential Theory in Gravity and Magnetic Applications*. Cambridge University Press, Cambridge (UK), 441 pages.

Bosum, W., Eberle, D. and Rehi, H.-J., 1988. A gyro-oriented 3-component borehole magnetometer for mineral prospecting, with examples of its application. *Geophysical Prospecting*, 36(8), pp.933–961.

Briggs, I.C., 1974. Machine contouring using minimum curvature. *Geophysics*, 39(1), pp.39–48.

Buchmann, J.P., 1960. Exploration of a geophysical anomaly at Trompsburg, Orange Free State, South Africa. *Transactions of the Geological Society of South Africa*, 63, pp.1–10.

Butler, R.F., 2004. *Paleomagnetism: Magnetic Domains to Geologic Terranes*. *Electronic edition*, University of Portland, 248 pages.

Cai, G., Chen, B.M. and Lee, T.H., 2011. Coordinate Systems and Transformations. In: Advances in Industrial Control. Springer London, pp.23–34.

Calcagno, P., Chilès, J.P., Courrioux, G. and Guillen, A., 2008. Geological modelling from field data and geological knowledge. Part I. Modelling method coupling 3D potential-field interpolation and geological rules. *Physics of the Earth and Planetary Interiors*, 171(1–4), pp.147–157.

Caratori Tontini, F., Cocchi, L. and Carmisciano, C., 2009. Rapid 3-D forward model of potential fields with application to the Palinuro Seamount magnetic anomaly (southern Tyrrhenian Sea, Italy). *Journal of Geophysical Research*, 114(B2), p.B02103.

Cevallos, C., 2014. Automatic generation of 3D geophysical models using curvatures derived from airborne gravity gradient data. *Geophysics*, 79(5), pp.G49–G58.

Cevallos, C., 2016. Interpreting the direction of the gravity gradient tensor eigenvectors: Their relation to curvature parameters of the gravity field. *Geophysics*, 81(3), pp.G49–G57.

Christensen, A.N., Dransfield, M.H., Van Galder, C. and Methods, P., 2015. Noise and repeatability of airborne gravity gradiometry. *First Break*, 33(April), pp.55–63.

Clark, D.A., 1997. Magnetic petrophysics and magnetic petrology: aids to geological interpretation of magnetic surveys. *AGSO Journal of Australian Geology & Geophysics*, 17(2), pp.83–103.

Clark, D.A., 2012. New methods for interpretation of magnetic vector and gradient tensor data I: Eigenvector analysis and the normalised source strength. *Exploration Geophysics*, 43(4), pp.267–282.

Clark, D.A., 2013. New methods for interpretation of magnetic vector and gradient tensor data II: Application to the Mount Leyshon anomaly, Queensland, Australia. *Exploration Geophysics*, 44(2), pp.114–127.

Clark, D.A., 2014. *Integrated Magnetics: Contributions to improved processing and interpretation of magnetic gradient tensor data, new methods for source location and estimation of magnetisation, and predictive magnetic exploration models*. Macquarie University, PhD Thesis, 345 pages.

Clark, D.A. and Emerson, D.W., 1991. Notes on rock magnetization characteristics in applied geophysical studies. *Exploration Geophysics*, 22(3), pp.547–555.

Clark, D.A., Schmidt, P.W., Coward, D.A. and Huddleston, M.P., 1998. Remote determination of magnetic properties and improved drill targeting of magnetic anomaly sources by Differential Vector Magnetometry (DVM). *Exploration Geophysics*, 29(4), pp.312–319.

Clarke, J. and Braginski, A.I., 2004. *The SQUID Handbook*. *Vol. I Fundamentals and Technology of SQUIDS and SQUID Systems*, Weinheim, FRG: Wiley-VCH Verlag GmbH & Co. KGaA, 414 pages.

Coggon, J.H., 1976. Magnetic and gravity anomalies of polyhedra. *Geoexploration*, 14(2),

pp.93–105.

Cole, P. and Cooper, G.R.J., 2018. Determination of the distance to magnetic sources using tensor data. *Pure and Applied Geophysics*, 175(6), pp.2237–2250.

Cooper, G.R.J., 1997. GravMap and PFproc: Software for filtering geophysical map data. *Computers & Geosciences*, 23(1), pp.91–101.

Cooper, G.R.J., 2014a. Reducing the dependence of the analytic signal amplitude of aeromagnetic data on the source vector direction. *Geophysics*, 79(4), pp.J55–J60.

Cooper, G.R.J., 2014b. The automatic determination of the location, depth, and dip of contacts from aeromagnetic data. *Geophysics*, 79(3), pp.35–41.

Cooper, G.R.J., 2014c. The automatic determination of the location and depth of contacts and dykes from aeromagnetic data. *Pure and Applied Geophysics*, 171(9), pp.2417–2423.

Cooper, G.R.J., 2015. Using the analytic signal amplitude to determine the location and depth of thin dikes from magnetic data. *Geophysics*, 80(1), pp.J1–J6.

Cooper, G.R.J., 2016. An improved method for determining the distance to magnetic sources. *Pure and Applied Geophysics*, 173(4), pp.1279–1288.

Cooper, G.R.J. and Whitehead, R.C., 2016. Determining the distance to magnetic sources. *Geophysics*, 81(2), pp.J39–J48.

Daudt, C.R., Braile, L.W., Nowack, R.. and Chiang, C.S., 1989. A comparison of finite-difference and Fourier method calculations of synthetic seismograms. *Bulletin of the Seismological Society of America*, 79(4), pp.1210–1230.

Diebel, J., 2006. Representing attitude: Euler angles, unit quaternions, and rotation vectors. *Matrix*, 58, pp.1–35.

Eroglu, S., Schoenberg, R., Wille, M., Beukes, N. and Taubald, H., 2015. Geochemical stratigraphy, sedimentology, and Mo isotope systematics of the ca. 2.58-2.50Ga-old Transvaal Supergroup carbonate platform, South Africa. *Precambrian Research*, 266, pp.27–46.

Eschner, W. and Ludwig, W., 1995. *Planar gradiomenetrs arranged on non-parallel surfaces for determination of a gradient tensor of a magnetic field*. US005469056A.

Ester, M., Kriegel, H.-P., Sander, J. and Xu, X., 1996. A density-based algorithm for discovering clusters in large spatial databases with noise. In: *Proceedings of the Second International Conference on Knowledge Discovery and Data Mining (KDD-96)*. Elsevier, pp.226–231.

Farr, T., Rosen, P.A., Caro, E., Crippen, R., Duren, R., Hensley, S., Kobrick, M., Paller, M., Rodriguez, E., Roth, L., Seal, D., Shaffer, S., Shimada, J., Umland, J., Werner, M., Oskin, M., Burbank, D. and Alsdorf, D., 2007. The shuttle radar topography mission. *Reviews of Geophysics*, 45(2005), pp.1–33.

Fisher, R., 1953. Dispersion on a sphere. *Proceedings of the Royal Society A: Mathematical, Physical and Engineering Sciences*, 217(1130), pp.295–305.

Fitzgerald, D.J., Argast, D. and Holstein, H., 2009. Further developments with full tensor gradiometry datasets. In: *20th International Geophysical Conference and Exhibition (22-25 November 2009)*. Adelaide, South Australia, pp.1–7.

FitzGerald, D.J., Argast, D., Paterson, R. and Holstein, H., 2009. Full tensor magnetic gradiometry processing and interpretation developments. In: *11th SAGA Biennial Technical Meeting and Exhibition*. pp.265–272.

Fitzgerald, D.J. and Holstein, H., 2006. Innovative data processing methods for gradient airborne geophysical data sets. *The Leading Edge*, 25(1), pp.87–94.

Fitzgerald, D.J. and Holstein, H., 2016. Optimising surface mapping of elongated geological features from full tensor gravity gradiometry. In: *Vienna 2016 - 78th EAGE Conference & Exhibition 2016*. Vienna, Austria, 30 May - 2 June 2016.

Fitzgerald, D.J. and Milligan, P., 2013. Defining a deep fault network for Australia, using 3D 'worming' (SEG annual meeting, Houston 2013). In: *SEG Annual Meeting, Houston 2013*. Houston, USA, pp.1126–1130.

Fitzgerald, D.J. and Paterson, R., 2013. Getting the best value from gravity gradiometry. In: *13th SAGA Biennial Conference & Exhibition*. Kruger National Park, South Africa.

Fitzgerald, D.J., Reid, A.B., Holstein, H. and Biegert, E., 2007. The amplitude / phase treatment of full tensor gradiometry. In: *SEG / San Antonio 2007 Annual Meeting*. San Antonio, pp.765–769.

Gabor, D., 1946. Theory of communication. *Journal of the Institution of Electrical Engineers - Part III: Radio and Communication Engineering*, 93(26), pp.429–457.

Geological Survey of South Africa, 1993. *1:250 000 Geological Series, Vryburg (2624)*.

Goldfarb, R.B. and Fickett, F.R., 1985. *Units for Magnetic Properties, NBS Special Publication 696*. Boulder, Colorado, 1 page.

Grant, F.S., 1985. Aeromagnetics, geology and ore environments, I. Magnetite in igneous, sedimentary and metamorphic rocks: An overview. *Geoexploration*, 23(3), pp.303–333.

Green, A.A., 1983. A comparison of adjustment procedures for leveling aeromagnetic survey data. *Geophysics*, 48(6), pp.745–753.

Guillen, A., Calcagno, P., Courrioux, G., Joly, A. and Ledru, P., 2008. Geological modelling from field data and geological knowledge. Part II. Modelling validation using gravity and magnetic data inversion. *Physics of the Earth and Planetary Interiors*, 171(1–4), pp.158–169.

Guptasarma, D. and Singh, B., 1999. New scheme for computing the magnetic field resulting from a uniformly magnetized arbitrary polyhedron. *Geophysics*, 64(1), pp.70–74.

Hall, D.H., 1959. Direction of polarization determined from magnetic anomalies. *Journal of Geophysical Research*, 64(11), pp.1945–1959.

Hamilton, W.R., 1853. Lectures on quaternions. *Royal Irish Academy*, pp.1–736.

Heath, P., 2003. Evolving the regolith from gravity and magnetics tensor data : theory and preliminary results. In: *Advances in Regolith*. pp.165–169.

Heath, P., 2007. *Analysis of potential field gradient tensor data: forward modelling, inversion and near-surface exploration*. University of Adelaide, PhD Thesis, 206 pages.

Heath, P., Heinson, G. and Greenhalgh, S., 2003. Some comments on potential field tensor data. *Exploration Geophysics*, 34(1), pp.57–62.

Helbig, K., 1963. Some integrals of magnetic anomalies and their relation to the parameters of the disturbing body. *Zeitschrift für Geophysik*, (29), pp.83–96.

Henderson, R.G. and Zietz, I., 1948. Analysis of total magnetic intensity anomalies produced by point and line sources. *Geophysics*, 13(3), pp.428–436.

Hjelt, S.-E., 1972. Magnetostatic anomalies of dipping prisms. *Geoexploration*, 10(4), pp.239–254.

Hjelt, S.-E., 1974. The gravity anomaly of a dipping prism. *Geoexploration*, 12(1), pp.29–39.

Holstein, H., 2002. Gravimagnetic similarity in anomaly formulas for uniform polyhedra. *Geophysics*, 67(4), pp.1126–1133.

Holstein, H., 2003. Gravimagnetic anomaly formulas for polyhedra of spatially linear media. *Geophysics*, 68(1), pp.157–167.

Holstein, H., FitzGerald, D.J. and Stefanov, H., 2013. Gravimagnetic similarity for homogeneous rectangular prisms. In: *75th EAGE Conference & Exhibition*. London, UK, pp.10–13.

Holstein, H., Fitzgerald, D.J., Zengerer, M. and Starr, A., 2015. Left or right handed potential data? *First Break*, 33(April), pp.87–92.

Hsu, S.-K., Coppens, D. and Shyu, C.-T., 1998. Depth to magnetic source using the generalized analytic signal. *Geophysics*, 63(6), pp.1947–1957.

Humphrey, K.P., Horton, T.J. and Keene, M.N., 2005. Detection of mobile targets from a moving platform using an actively shielded, adaptively balanced SQUID gradiometer. *IEEE Transactions on Applied Superconductivity*, 15(2 PART I), pp.753–756.

Husson, E., Guillen, A., Séranne, M., Courrioux, G. and Couëffé, R., 2018. 3D Geological modelling and gravity inversion of a structurally complex carbonate area: application for karstified massif localization. *Basin Research*, 30(4), pp.766–782.

Johnson, B.D. and van Klinken, G., 1979. Some equivalent bodies and ambiguity in magnetic and Gravity interpretation. *Exploration Geophysics*, 10(1), pp.109–110.

Jones, E., Oliphant, T., Peterson, P. and Others, 2001. *SciPy: Open Source Scientific Tools for Python*. [online] Available at: <http://www.scipy.org/> [Accessed 23 Aug. 2017].

Keene, M.N., Humphrey, K.P. and Horton, T.J., 2005. Actively shielded, adaptively balanced SQUID gradiometer system for operation aboard moving platforms. *IEEE Transactions on Applied Superconductivity*, 15(2 PART I), pp.761–764.

Kogbetliantz, E.G., 1944. Quantitative interpretation of magnetic and gravitational anomalies. *Geophysics*, 9, pp.463–493.

Kolecki, J.C., 2002. *An Introduction to Tensors for Students of Physics and Engineering*. NASA/TM—2002-211716. Glenn Research Center, Cleveland, Ohio, 29 pages.

Lajaunie, C., Courrioux, G. and Manuel, L., 1997. Foliation fields and 3D cartography in geology: Principles of a method based on potential interpolation. *Mathematical Geology*, 29(4), pp.571–584.

Li, X. and Chouteau, M., 1998. Three dimensional gravity modeling in all space. *Surveys in Geophysics*, 19(4), pp.339–368.

Li, Y., 2001. 3-D inversion of gravity gradiometer data. In: *SEG Technical Program Expanded Abstracts 2001*. Society of Exploration Geophysicists, pp.1470–1473.

Li, Y. and Oldenburg, D.W., 1996. 3-D inversion of magnetic data. *Geophysics*, 61(2), pp.394–408.

Li, Y. and Oldenburg, D.W., 2001. Stable reduction to the pole at the magnetic equator. *Geophysics*, 66(2), p.571.

Ma, G. and Du, X., 2012. An improved analytic signal technique for the depth and structural index from 2D magnetic anomaly data. *Pure and Applied Geophysics*, 169(12), pp.2193–2200.

Ma, G. and Li, L., 2013. Direct analytic signal (DAS) method in the interpretation of magnetic data. *Journal of Applied Geophysics*, 88, pp.101–104.

MacLeod, I.N., Jones, K. and Dai, T.F., 1993. 3-D analytic signal in the interpretation of total magnetic field data at low magnetic latitudes. *Exploration Geophysics*, 24(4), pp.679–690.

MacQueen, J., 1967. Some methods for classification and analysis of multivariate observations. *Proceedings of the Fifth Berkeley Symposium on Mathematical Statistics and Probability*, 1(233), pp.281–297.

MAG3D, 2017. *A Program Library for Forward Modelling and Inversion of Magnetic Data over 3D Structures, version 6.* Developed under the consortium research project Joint/Cooperative Inversion of Geophysical and Geological Data, UBC-Geophysical Inversion Facility, Department of Earth and Ocean Sciences, University of British Columbia, Vancouver, British Columbia.

Maier, W.D., Peltonen, P., Grantham, G. and Mänttári, I., 2003. A new 1.9 Ga age for the Trompsburg intrusion, South Africa. *Earth and Planetary Science Letters*, 212, pp.251–360.

Maré, L.P. and Cole, J., 2005. The Trompsburg Complex, South Africa: A preliminary three dimensional model. *Journal of African Earth Sciences*, 44, pp.314–330.

McInerney, P., Goldberg, A., Calcagno, P., Courrioux, G., Guillen, R. and Seikel, R., 2007. Improved 3D geology modelling using an implicit function interpolator and forward modelling of potential field data. *Exploration 07: Fifth Decennial International Conference on Mineral Exploration*, pp.919–922.

Munschy, M. and Fleury, S., 2011. Scalar, vector, tensor magnetic anomalies: Measurement or computation? *Geophysical Prospecting*, 59(6), pp.1035–1045.

Nabighian, M.N., 1972. The analytic signal of two dimensional magnetic bodies with polygonal cross section: its properties and use for automated anomaly interpretation. *Geophysics*, 37(3), pp.507–517.

Nabighian, M.N., Grauch, V.J.S., Hansen, R.O., LaFehr, T.R., Li, Y., Peirce, J.W., Phillips, J.D. and Ruder, M.E., 2005. The historical development of the magnetic method in exploration. *Geophysics*, 70(6), p.33ND–61ND.

Nelson, J.B., 1988. Calculation of the magnetic gradient tensor from total field gradient measurements and its application to geophysical interpretation. *Geophysics*, 53(7), pp.957–966.

Nettleton, L.L., 1942. Gravity and magnetic calculations. *Geophysics*, 7(3), pp.293–310.

Ortlepp, R.J., 1959. A pre-Karoo igneous complex at Trompsburg, Orange Free State, revealed by drilling exploration. *Transactions of the Geological Society of South Africa*, 62, pp.33–57.

Pajot, G., de Viron, O., Diament, M., Lequentrec-Lalancette, M.-F. and Mikhailov, V., 2008. Noise reduction through joint processing of gravity and gravity gradient data. *Geophysics*, 73(3), pp.I23–I34.

Parker, R., 1973. The rapid calcuation of potential anomalies. *Geophysical Journal of the Royal Astronomical Society*, 31, pp.447–455.

Pedersen, L.B. and Rasmussen, T.M., 1990. The gradient tensor of potential field anomalies: Some implications on data collection and data processing of maps. *Geophysics*, 55(12), p.1558.

Pedersen, L.B., Rasmussen, T.M. and Dyrelius, D., 1990. Construction of component maps from aeromagnetic total field anomaly maps. *Geophysical Prospecting*, 38(7), pp.795–804.

Phillips, J.D., 2005. Can we estimate total magnetization directions from aeromagnetic data using Helbig's integrals? *Earth Planets Space*, 57(1), pp.681–689.

Phillips, J.D., Nabighian, M.N., Smith, D. V. and Li, Y., 2007. Estimating locations and total magnetization vectors of compact magnetic sources from scalar, vector, or tensor magnetic measurements through combined Helbig and Euler analysis. In: *SEG Technical Program Expanded Abstracts*. San Antonio, pp.770–774.

Plouff, D., 1976. Gravity and magnetic fields of polygonal prisms and application to magnetic terrain corrections. *Geophysics*, 41(4), pp.727–741.

Press, W.H., Teukolsky, S.A., Vettering, W.T. and Flannery, B.P., 2007. Numerical Recipes in C: The Art of Scientific Computing. In: 3rd ed. New York: Cambridge University Press., pp.186–190.

Rasmussen, R. and Pedersen, L.B., 1979. End corrections in potential field modelling. *Geophysical Prospecting*, 27(4), pp.749–760.

Reford, M.S., 1964. Magnetic anomalies over thin sheets. *Geophysics*, 29(4), pp.532–

536.

Reid, A.B., 1980. Aeromagnetic survey design. *Geophysics*, 45(5), pp.973–976.

Reid, A.B. and Thurston, J.B., 2014. The structural index in gravity and magnetic interpretation: Errors, uses, and abuses. *Geophysics*, 79(4), pp.J61–J66.

Rossberg, K., 1983. A First Course in Analytical Mechanics. John Wiley and Sons, Inc., pp.228–231.

Salem, A., Ravat, D., Mushayandebvu, M.F. and Ushijima, K., 2004. Linearized least-squares method for interpretation of potential-field data from sources of simple geometry. *Geophysics*, 69(3), pp.783–788.

Schiffler, M., Queitsch, M., Stolz, R., Chwala, A., Krech, W., Meyer, H.G. and Kukowski, N., 2014. Calibration of SQUID vector magnetometers in full tensor gradiometry systems. *Geophysical Journal International*, 198(2), pp.954–964.

Schiffler, M., Queitsch, M., Stolz, R., Meyer, H.G. and Kukowski, N., 2017. Application of Hilbert-like transforms for enhanced processing of full tensor magnetic gradient data. *Geophysical Prospecting*, 65, pp.68–81.

Schmidt, P., Clark, D.A., Leslie, K., Bick, M., Tilbrook, D. and Foley, C., 2004. GETMAG - a SQUID magnetic tensor gradiometer for mineral and oil exploration. *Exploration Geophysics*, 35(4), pp.297–305.

Schmidt, P.W. and Clark, D.A., 1998. The calculation of magnetic components and moments from TMI: a case study from the Tuckers igneous complex, Queensland. *Exploration Geophysics*, 29(4), p.609.

Schmidt, P.W. and Clark, D.A., 2006. The magnetic gradient tensor: Its properties and uses in source characterization. *The Leading Edge*, 25(1), pp.75–78.

Sheriff, R.E., 1991. *Encyclopedic Dictionary of Applied Geophysics*. 3rd ed. Society of Exploration Geophysicists.

Singh, B. and Guptasarma, D., 2001a. Joint modelling of gravity and magnetic fields - a new computational approach. *Current Science*, 81(12), pp.1626–1628.

Singh, B. and Guptasarma, D., 2001b. New method for fast computation of gravity and magnetic anomalies from arbitrary polyhedra. *Geophysics*, 66(2), p.521.

Spector, A. and Grant, F.S., 1970. Statistical models for interpreting aeromagnetic data. *Geophysics*, 35(2), pp.293–302.

Stolz, R., Schiffler, M., Queitsch, M., Schmelz, M., Goepel, A., Kukowski, N., Meyer, M. and Meyer, H., 2015. Why bother about gradients ? - SQUID based full tensor magnetic gradiometer for mineral exploration. In: *14th SAGA Biennial Technical Meeting and Exhibition*. Drakensberg, South Africa.

Stolz, R., Zakosarenko, V., Schulz, M., Chwala, A., Fritzsch, L., Meyer, H. and Köstlin, E., 2006. Magnetic full-tensor SQUID gradiometer system for geophysical applications. *Leading Edge*, (February), pp.178–180.

Talwani, M., 1965. Computation with the help of a digital computer of magnetic anomalies caused by bodies of arbitrary shape. *Geophysics*, 30(5), pp.797–817.

Tarlowski, C., 1989. *Magnetic modelling of two and three-dimensional bodies*. Australia: Australian Government Publishing Service.

Thompson, D.T., 1982. EULDPH: A new technique for making computer-assisted depth estimates from magnetic data. *Geophysics*, 47(1), pp.31–37.

Yin, G., Zhang, Y., Mi, S., Fan, H. and Li, Z., 2016. Calculation of the magnetic gradient tensor from total magnetic anomaly field based on regularized method in frequency domain. *Journal of Applied Geophysics*, 134, pp.44–54.

Zhang, C., Mushayandebvu, M.F., Reid, A.B., Fairhead, J.D. and Odegard, M.E., 2000. Euler deconvolution of gravity tensor gradient data. *Geophysics*, 65(2), pp.512–520.

Zhu, L., 2007. *Gravity gradient modeling using with Gravity and DEM*. The Ohio State University Columbus, Ohio 43210.

APPENDIX

Source code examples for performing calculations used in this thesis are provided below. They are all written in python and will require a python installation. The code contains routines for forward modelling, both for conventional and tensor data, as well as tests for source depth calculations

The program requires at a minimum the following:

- python 3.5.4
- GDAL 2.1.4
- matplotlib 2.0.2
- numba 0.34.0
- numpy 1.13.1
- scipy 0.19.1
- scikit_learn 0.18.2
- cycler 0.10.0
- pygmi 2.4.1

```
# ------------------------------------------------------------------------------
# Name:        tensorfin.py
#
# Author:      Patrick Cole
# E-Mail:      pcole@geoscience.org.za
#
# Copyright:   (c) 2018 Council for Geoscience
# Licence:     GPL-3.0
#
# This code is free software: you can redistribute it and/or modify
# it under the terms of the GNU General Public License as published by
# the Free Software Foundation, either version 3 of the License, or
# (at your option) any later version.
#
# This code is distributed in the hope that it will be useful,
# but WITHOUT ANY WARRANTY; without even the implied warranty of
# MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
# GNU General Public License for more details.
#
# You should have received a copy of the GNU General Public License
# along with this program.  If not, see <http://www.gnu.org/licenses/>.
# ------------------------------------------------------------------------------
"""
tensorfin.py forms part of the PhD submission for Patrick Cole at the
University of the Witwatersrand.

The code contains routines for forward modelling, both for conventional and
tensor data.

The program requires at a minimum the following:
* python 3.5.4
* GDAL 2.1.4
* matplotlib 2.0.2
* numba 0.34.0
* numpy 1.13.1
* scipy 0.19.1
* scikit learn 0.18.2
* cycler 0.10.0
* PyGMI

Since PyGMI is likely to be a separate install, the path to it can be specified
below.

Forward modelling is based on the work by Blakely (1995) and Heath (2007)

Blakely, R.J., 1995. Potential Theory in Gravity and Magnetic Applications.
Heath, P., 2007. Analysis of Potential Field Gradient Tensor Data: Forward
Modelling, Inversion and Near-Surface Exploration. The University of Adelaide.

"""

# pylint: disable=C0103,R0201,R0904,R0914,R0915, W0612, E1101

import sys
import copy
import glob
import tempfile
import winsound
import warnings
from osgeo import gdal, osr
import numpy as np
from numpy.polynomial import polynomial
import scipy.interpolate as si
import scipy.signal as ss
from numba import jit
import matplotlib.pyplot as plt
from matplotlib import colors
from mpl_toolkits.axes_grid1 import make_axes_locatable
from sklearn.cluster import DBSCAN, KMeans
from cycler import cycler
```

```
from pygmi.pfmod.iodefs import ExportMod3D


PyGMIPATH = r'C:\Work\Programming\pygmi'
plt.rcParams['axes.prop_cycle'] = cycler(color='bgrcmyk')
plt.rcParams['axes.grid'] = True
plt.rcParams['axes.axisbelow'] = True
plt.rcParams['image.cmap'] = 'jet'


class GeoData(object):
    """
    Data layer class:
    This class defines each geological type and calculates the field
    for one cube from the standard definitions.

    The is a class which contains the geophysical information for a single
    lithology. This includes the final calculated field for that lithology
    only.

    Attributes
    ----------

    qratio : float
        q ratio for remanence
    minc : float
        remanence inclination
    mdec : float
        remanence declination
    mstrength : float
        remanence magnetization
    finc : float
        field inclination
    fdec : float
        field declination
    hintn : float
        field strength
    theta : float
        azimuth
    dxy : float
        cube dimension
    susc : float
        susceptibility
    dens : float
        density
    bdens : float
        background density
    height : float
        observation height
    Gc : float
        Gravitation con
    """
    def __init__(self, parent=None, ncols=10, nrows=10, numz=10, dxy=10.,
                 d_z=10., mht=80., ght=0.):
        self.hintn = 30000.
        self.susc = 0.01
        self.mstrength = 0.
        self.finc = -63.
        self.fdec = -17.
        self.minc = -63.
        self.mdec = -17.
        self.theta = 90.
        self.bdensity = 2.67
        self.density = 2.85
        self.qratio = 0.0
        self.lith_index = 0
        self.parent = parent
        if hasattr(parent, 'pbars'):
            self.pbars = parent.pbars
        else:
            self.pbars = None

        if hasattr(parent, 'showtext'):
```

```
            self.showtext = parent.showtext
        else:
            self.showtext = print

    # ncols and nrows are the smaller dimension of the original grid.
    # numx, numy, numz are the dimensions of the larger grid to be used as a
    # template.

        self.modified = True
        self.g_cols = None
        self.g_rows = None
        self.g_dxy = None
        self.numz = None
        self.dxy = None
        self.d_z = None
        self.zobsm = None
        self.zobsg = None

        self.mlayers = None
        self.mtmp = None
        self.glayers = None

        self.x12 = None
        self.y12 = None
        self.z12 = None

        self.set_xyz(ncols, nrows, numz, dxy, mht, ght, d_z)

    def calc_origin_grav(self, hcor=None):
        """ Calculate the field values for the lithologies"""

        if self.modified is True:
            numx = self.g_cols*self.g_dxy
            numy = self.g_rows*self.g_dxy

# The 2 lines below ensure that the profile goes over the center of the grid
# cell
            xdist = np.arange(self.g_dxy/2, numx+self.g_dxy/2, self.g_dxy,
                              dtype=float)
            ydist = np.arange(numy-self.g_dxy/2, -1*self.g_dxy/2,
                              -1*self.g_dxy, dtype=float)

            if hcor is None:
                hcor2 = 0
            else:
                hcor2 = int(self.numz-hcor.max())

            self.showtext('   Calculate gravity origin field')
            self.gboxmain(xdist, ydist, self.zobsg, hcor2)

            self.modified = False

    def calc_origin_mag(self, hcor=None):
        """ Calculate the field values for the lithologies"""

        if self.modified is True:
            numx = self.g_cols*self.g_dxy
            numy = self.g_rows*self.g_dxy

# The 2 lines below ensure that the profile goes over the center of the grid
# cell
            xdist = np.arange(self.g_dxy/2, numx+self.g_dxy/2, self.g_dxy,
                              dtype=float)
            ydist = np.arange(numy-self.g_dxy/2, -1*self.g_dxy/2,
                              -1*self.g_dxy, dtype=float)

            self.showtext('   Calculate magnetic origin field')

            if hcor is None:
                hcor2 = 0
            else:
                hcor2 = int(self.numz-hcor.max())
```

```
            self.mboxmain(xdist, ydist, self.zobsm, hcor2)
#            self.mtmp = self.mlayers.copy()
#            self.gmmain(xdist, ydist)

            self.modified = False

    def rho(self):
        """ Returns the density contrast """
        return self.density - self.bdensity

    def set_xyz(self, ncols, nrows, numz, g_dxy, mht, ght, d_z, dxy=None,
                modified=True):
        """ Sets/updates xyz parameters again """
        self.modified = modified
        self.g_cols = ncols*2+1
        self.g_rows = nrows*2+1
        self.numz = numz
        self.g_dxy = g_dxy
        self.d_z = d_z
        self.zobsm = -mht
        self.zobsg = -ght

        if dxy is None:
            self.dxy = g_dxy  # This must be a multiple of g_dxy or equal to it
        else:
            self.dxy = dxy  # This must be a multiple of g_dxy or equal to it.

        self.set_xyz12()

    def set_xyz12(self):
        """ Set x12, y12, z12. This is the limits of the cubes for the model"""

        numx = self.g_cols*self.g_dxy
        numy = self.g_rows*self.g_dxy
        numz = self.numz*self.d_z
        dxy = self.dxy
        d_z = self.d_z

        self.x12 = np.array([numx/2-dxy/2, numx/2+dxy/2])
        self.y12 = np.array([numy/2-dxy/2, numy/2+dxy/2])
        self.z12 = np.arange(-numz, numz+d_z, d_z)

    def gboxmain(self, xobs, yobs, zobs, hcor):
        """ Gbox routine by Blakely
            Note: xobs, yobs and zobs must be floats or there will be problems
            later.

        Subroutine GBOX computes the vertical attraction of a
        rectangular prism.  Sides of prism are parallel to x,y,z axes,
        and z axis is vertical down.

        Input parameters:
            Observation point is (x0,y0,z0).  The prism extends from x1
            to x2, from y1 to y2, and from z1 to z2 in the x, y, and z
            directions, respectively.  Density of prism is rho.  All
            distance parameters in units of m;

        Output parameters:
            Vertical attraction of gravity, g, in mGal/rho.
            Must still be multiplied by rho outside routine.
            Done this way for speed. """

        glayers = []
        piter = iter

        z1122 = self.z12.copy()
        x_1 = float(self.x12[0])
        y_1 = float(self.y12[0])
        x_2 = float(self.x12[1])
        y_2 = float(self.y12[1])
        z_0 = float(zobs)
        numx = int(self.g_cols)
        numy = int(self.g_rows)
```

```
        if zobs == 0:
            zobs = -0.01

        for z1 in piter(z1122[:-1]):
            if z1 < z1122[hcor]:
                glayers.append(np.zeros((self.g_cols, self.g_rows)))
                continue

            z2 = z1 + self.d_z

            gval = np.zeros([self.g_cols, self.g_rows])

            gval = gbox(gval, xobs, yobs, numx, numy, z_0, x_1, y_1, z1,
                        x_2, y_2, z2, np.ones(2), np.ones(2), np.ones(2),
                        np.array([-1, 1]))

            gval *= 6.6732e-3
            glayers.append(gval)

        self.glayers = np.array(glayers)

    def mboxmain(self, xobs, yobs, zobs, hcor):
        """ Mbox routine by Blakely
            Note: xobs, yobs and zobs must be floats or there will be problems
            later.

        Subroutine MBOX computes the total field anomaly of an infinitely
        extended rectangular prism.  Sides of prism are parallel to x,y,z
        axes, and z is vertical down.  Bottom of prism extends to infinity.
        Two calls to mbox can provide the anomaly of a prism with finite
        thickness; e.g.,

            call mbox(x0,y0,z0,x1,y1,z1,x2,y2,mi,md,fi,fd,m,theta,t1)
            call mbox(x0,y0,z0,x1,y1,z2,x2,y2,mi,md,fi,fd,m,theta,t2)
            t=t1-t2

        Requires subroutine DIRCOS.  Method from Bhattacharyya (1964).

        Input parameters:
            Observation point is (x0,y0,z0).  Prism extends from x1 to
            x2, y1 to y2, and z1 to infinity in x, y, and z directions,
            respectively.  Magnetization defined by inclination mi,
            declination md, intensity m.  Ambient field defined by
            inclination fi and declination fd.  X axis has declination
            theta. Distance units are irrelevant but must be consistent.
            Angles are in degrees, with inclinations positive below
            horizontal and declinations positive east of true north.
            Magnetization in A/m.

        Output paramters:
            Total field anomaly t, in nT."""

        mlayers = []
        piter = iter

        z1122 = self.z12.copy()
        z1122 = z1122.astype(float)
        x1 = float(self.x12[0])
        y1 = float(self.y12[0])
        x2 = float(self.x12[1])
        y2 = float(self.y12[1])
        z0 = float(zobs)
        numx = int(self.g_cols)
        numy = int(self.g_rows)

        ma, mb, mc = dircos(self.minc, self.mdec, self.theta)
        fa, fb, fc = dircos(self.finc, self.fdec, self.theta)

        mr = self.mstrength * np.array([ma, mb, mc]) * 100
        mi = self.susc*self.hintn*np.array([fa, fb, fc]) / (4*np.pi)
        m3 = mr+mi
```

```
        mt = np.sqrt(m3 @ m3)
        if mt > 0:
            m3 /= mt

        ma, mb, mc = m3

        fm1 = ma*fb + mb*fa
        fm2 = ma*fc + mc*fa
        fm3 = mb*fc + mc*fb
        fm4 = ma*fa
        fm5 = mb*fb
        fm6 = mc*fc

        if zobs == 0:
            zobs = -0.01

        z1122 = np.append(z1122, [2*z1122[-1]-z1122[-2]])

        for z1 in piter(z1122):
            if z1 < z1122[hcor]:
                mlayers.append(np.zeros((self.g_cols, self.g_rows)))
                continue

            mval = np.zeros([self.g_cols, self.g_rows])

            mval = mbox(mval, xobs, yobs, numx, numy, z0, x1, y1, z1, x2, y2,
                        fm1, fm2, fm3, fm4, fm5, fm6, np.ones(2), np.ones(2))

            mlayers.append(mval)

        self.mlayers = np.array(mlayers) * mt
        self.mlayers = self.mlayers[:-1]-self.mlayers[1:]


class Data(object):
    """
    PyGMI Data Object

    Attributes
    ----------
    data : numpy masked array
        array to contain raster data
    tlx : float
        Top Left X coordinate of raster grid
    tly : float
        Top Left Y coordinate of raster grid
    xdim : float
        x-dimension of grid cell
    ydim : float
        y-dimension of grid cell
    nrofbands : int
        number of raster bands
    dataid : str
        band name or id
    rows : int
        number of rows for each raster grid/band
    cols : int
        number of columns for each raster grid/band
    nullvalue : float
        grid null or nodata value
    norm : dictionary
        normalized data
    gtr : tuple
        projection information
    wkt : str
        projection information
    units : str
        description of units to be used with color bars
    """
    def __init__(self):
        self.data = np.ma.array([])
        self.tlx = 0.0  # Top Left X coordinate
        self.tly = 0.0  # Top Left Y coordinate
```

```
        self.xdim = 1.0
        self.ydim = 1.0
        self.nrofbands = 1
        self.dataid = ''
        self.rows = -1
        self.cols = -1
        self.nullvalue = 1e+20
        self.norm = {}
        self.gtr = (0.0, 1.0, 0.0, 0.0, -1.0)
        self.wkt = ''
        self.units = ''


class LithModel(object):
    """ Lithological Model Data.

    This is the main data structure for the modelling program

    Attributes
    ----------
    mlut : dictionary
        color table for lithologies
    numx : int
        number of columns per layer in model
    numy : int
        number of rows per layer in model
    numz : int
        number of layers in model
    dxy : float
        dimension of cubes in the x and y directions
    d z : float
        dimension of cubes in the z direction
    lith index : numpy array
        3D array of lithological indices.
    curlayer : int
        Current layer
    xrange : list
        minimum and maximum x coordinates
    yrange : list
        minimum and maximum y coordinates
    zrange : list
        minimum and maximum z coordinates
    curprof : int
        current profile : in x or y direction)
    griddata : dictionary
        dictionary of Data classes with raster data
    custprofx : dictionary
        custom profile x coordinates
    custprofy : dictionary
        custom profile y coordinates
    profpics : dictionary
        profile pictures
    lith list : dictionary
        list of lithologies
    lith list reverse : dictionary
        reverse lookup for lith_list
    mht : float
        height of magnetic sensor
    ght : float
        height of gravity sensor
    gregional : float
        gravity regional correction
    name : str
        name of the model
    """

    def __init__(self):
        self.mlut = {0: [170, 125, 90], 1: [255, 255, 0]}
        self.numx = None
        self.numy = None
        self.numz = None
        self.dxy = None
        self.d_z = None
```

```
        self.lith_index = None
        self.lith_index_old = None
        self.curlayer = None
        self.xrange = [None, None]
        self.yrange = [None, None]
        self.zrange = [None, None]
        self.curprof = None
        self.griddata = {}
        self.custprofx = {}
        self.custprofy = {}
        self.profpics = {}
        self.lith_list = {}
        self.lith_list_reverse = {}
        self.mht = None
        self.ght = None
        self.gregional = 100
        self.name = '3D Model'
        self.dataid = '3D Model'
        self.tmpfiles = None

        # Next line calls a function to update the variables above.
        self.update(50, 40, 5, 0, 0, 0, 100, 100, 100, 0)

        self.olith_index = None
        self.odxy = None
        self.od_z = None
        self.oxrng = None
        self.oyrng = None
        self.ozrng = None
        self.onumx = None
        self.onumy = None
        self.onumz = None

        self.is_ew = True

    def lithold_to_lith(self, nodtm=False):
        """ Transfers an old lithology to the new one, using updates parameters
        """
        if self.olith_index is None:
            return

        xvals = np.arange(self.xrange[0], self.xrange[1], self.dxy)
        yvals = np.arange(self.yrange[0], self.yrange[1], self.dxy)
        zvals = np.arange(self.zrange[0], self.zrange[1], self.d_z)

        if xvals[-1] == self.xrange[1]:
            xvals = xvals[:-1]
        if yvals[-1] == self.yrange[1]:
            yvals = yvals[:-1]
        if zvals[-1] == self.zrange[1]:
            yvals = yvals[:-1]

        xvals += 0.5 * self.dxy
        yvals += 0.5 * self.dxy
        zvals += 0.5 * self.d_z

        xvals = xvals[self.oxrng[0] < xvals]
        xvals = xvals[xvals < self.oxrng[1]]
        yvals = yvals[self.oyrng[0] < yvals]
        yvals = yvals[yvals < self.oyrng[1]]
        zvals = zvals[self.ozrng[0] < zvals]
        zvals = zvals[zvals < self.ozrng[1]]

        for x_i in xvals:
            o_i = int((x_i - self.oxrng[0]) / self.odxy)
            i = int((x_i - self.xrange[0]) / self.dxy)
            for x_j in yvals:
                o_j = int((x_j - self.oyrng[0]) / self.odxy)
                j = int((x_j - self.yrange[0]) / self.dxy)
                for x_k in zvals:
                    o_k = int((self.ozrng[1] - x_k) / self.od_z)
                    k = int((self.zrange[1] - x_k) / self.d_z)
```

```
                if (self.lith_index[i, j, k] != -1 and
                        self.olith_index[o_i, o_j, o_k] != -1) or nodtm:
                    self.lith_index[i, j, k] = \
                        self.olith_index[o_i, o_j, o_k]

    def dtm_to_lith(self):
        """ Assign the DTM to the model. This means creating nodata values in
        areas above the DTM. These values are assigned a lithology of -1."""

        if 'DTM Dataset' not in self.griddata:
            return

        self.lith_index = np.zeros([self.numx, self.numy, self.numz],
                                   dtype=int)

        curgrid = self.griddata['DTM Dataset']

        d_x = curgrid.xdim
        d_y = curgrid.ydim
        utlx = curgrid.tlx
        utly = curgrid.tly
        gcols = curgrid.cols
        grows = curgrid.rows

        gxmin = utlx
        gymax = utly
        utlz = curgrid.data.max()

        self.lith_index[:, :, :] = 0

        for i in range(self.numx):
            xcrd = self.xrange[0] + (i + .5) * self.dxy
            xcrd2 = int((xcrd - gxmin) / d_x)
            for j in range(self.numy):
                ycrd = self.yrange[1] - (j + .5) * self.dxy
                ycrd2 = grows - int((gymax - ycrd) / d_y)
                if ycrd2 == grows:
                    ycrd2 = grows-1

                if (ycrd2 >= 0 and xcrd2 >= 0 and ycrd2 < grows and
                        xcrd2 < gcols):
                    alt = curgrid.data.data[ycrd2, xcrd2]
                    if (curgrid.data.mask[ycrd2, xcrd2] or
                            np.isnan(alt) or alt == curgrid.nullvalue):
                        alt = curgrid.data.mean()
                    k_2 = int((utlz - alt) / self.d_z)
                    self.lith_index[i, j, :k_2] = -1

    def init_grid(self, data):
        """ Initializes raster variables in the Data class

        Args:
            data (numpy masked array): masked array containing raster data."""

        grid = Data()
        grid.data = data
        grid.cols = self.numx
        grid.rows = self.numy
        grid.xdim = self.dxy
        grid.ydim = self.dxy
        grid.tlx = self.xrange[0]
        grid.tly = self.yrange[1]
        return grid

    def init_calc_grids(self):
        """ Initializes mag and gravity from the model """
        tmp = np.ma.zeros([self.numy, self.numx])
        self.griddata['Calculated Magnetics'] = self.init_grid(tmp.copy())
        self.griddata['Calculated Magnetics'].dataid = 'Calculated Magnetics'
        self.griddata['Calculated Magnetics'].units = 'nT'
        self.griddata['Calculated Gravity'] = self.init_grid(tmp.copy())
        self.griddata['Calculated Gravity'].dataid = 'Calculated Gravity'
        self.griddata['Calculated Gravity'].units = 'mgal'
```

```
def is_modified(self, modified=True):
    """ Updates modified flag

    Args:
        modified (bool): flag for whether the lithology has been modified
    """
    for i in self.lith_list:
        self.lith_list[i].modified = modified

def update(self, cols, rows, layers, utlx, utly, utlz, dxy, d_z, mht=-1,
           ght=-1, usedtm=True):
    """ Updates the local variables for the LithModel class

    Args:
        cols (int): number of columns per layer in model
        rows (int): number of rows per layer in model
        layers (int): number of layers in model
        utlx (float): upper top left (NW) x coordinate
        utly (float): upper top left (NW) y coordinate
        utlz (float): upper top left (NW) z coordinate
        dxy (float): dimension of cubes in the x and y directions
        d_z (float): dimension of cubes in the z direction
        mht (float): height of magnetic sensor
        ght (float): height of gravity sensor
    """
    if mht != -1:
        self.mht = mht
    if ght != -1:
        self.ght = ght

    self.olith_index = self.lith_index
    self.odxy = self.dxy
    self.od_z = self.d_z
    self.oxrng = np.copy(self.xrange)
    self.oyrng = np.copy(self.yrange)
    self.ozrng = np.copy(self.zrange)
    self.onumx = self.numx
    self.onumy = self.numy
    self.onumz = self.numz

    xextent = cols * dxy
    yextent = rows * dxy
    zextent = layers * d_z

    self.numx = cols
    self.numy = rows
    self.numz = layers
    self.xrange = [utlx, utlx + xextent]
    self.yrange = [utly - yextent, utly]
    self.zrange = [utlz - zextent, utlz]

    self.custprofx[0] = self.xrange
    self.custprofy[0] = (self.yrange[0], self.yrange[0])

    self.dxy = dxy
    self.d_z = d_z
    self.curlayer = 0
    self.curprof = 0
    self.lith_index = np.zeros([self.numx, self.numy, self.numz],
                               dtype=int)
    self.lith_index_old = np.zeros([self.numx, self.numy, self.numz],
                                   dtype=int)
    self.lith_index_old[:] = -1

    self.init_calc_grids()
    if usedtm:
        self.dtm_to_lith()
    self.lithold_to_lith(not usedtm)
    self.update_lithlist()
    self.is_modified()

def update_lithlist(self):
```

```
        """ Updates lith_list from local variables"""
        for i in self.lith_list:
            self.lith_list[i].set_xyz(self.numx, self.numy, self.numz,
                                      self.dxy, self.mht, self.ght, self.d_z,
                                      modified=False)

    def update_lith_list_reverse(self):
        """ Update the lith list reverse lookup. It must be run at least once
        before using lith list reverse"""
        keys = list(self.lith_list.keys())
        values = list(self.lith_list.values())

        if not keys:
            return

        self.lith_list_reverse = {}
        for i in range(len(keys)):
            self.lith_list_reverse[list(values)[i].lith_index] = list(keys)[i]


class TensorCube(object):
    """
    This class computes the forward modelled tensor responses for a cube.

    Attributes
    ----------
    minc : float
        remanence inclination
    mdec : float
        remanence declination
    mstrength : float
        remanence magnetization
    inc : float
        field inclination
    dec : float
        field declination
    hintn : float
        field strength
    azim : float
        azimuth
    dxy : float
        cube dimension
    susc : float
        susceptibility
    dens : float
        density
    bdens : float
        background density
    height : float
        observation height
    Gc : float
        Gravitation constant = 6.6732e-3  # includes 100000 factor to convert
        to mGal
    u : list
        x cube coordinates
    v : list
        y cube coordinates
    w : list
        z cube coordinates
    rc : list
        length of model
    """
    def __init__(self):

        self.minc = -60.0
        self.mdec = -15.0
        self.mstrength = 0
        self.inc = -60.0
        self.dec = -15.0
        self.hintn = 28000
        self.azim = 90
        self.dxy = 10
        self.susc = 0.1
```

```
        self.dens = 2.85
        self.bdens = 2.67
        self.height = 0.0
        self.Gc = 6.6732e-3  # includes 100000 factor to convert to mGal
        self.mt = None

        self.u = [100, 300]
        self.v = [100, 300]
        self.w = [-20, -3000]
        self.rc = 400

        self.cx = None
        self.cy = None
        self.cz = None
        self.pmag = None
        self.pbx = None
        self.pby = None
        self.pbz = None

        self.pgrv = None
        self.pgx = None
        self.pgy = None
        self.pgz = None

        self.bx = None
        self.by = None
        self.bz = None
        self.bxx = None
        self.byy = None
        self.bzz = None
        self.bxy = None
        self.byz = None
        self.bxz = None
        self.magval = None

        self.gx = None
        self.gy = None
        self.gz = None
        self.gxx = None
        self.gyy = None
        self.gzz = None
        self.gxy = None
        self.gyz = None
        self.gxz = None
        self.grvval = None
        self.xyall = None
        self.coords = None

    def init_grids(self):
        """ init grids """
        self.xyall = np.arange(0, int(self.rc), int(self.dxy),
                               dtype=np.float64)
        self.xyall += self.dxy/2.

        tmp = (len(self.xyall), len(self.xyall))

        self.bx = np.zeros(tmp)
        self.by = np.zeros(tmp)
        self.bz = np.zeros(tmp)
        self.bxx = np.zeros(tmp)
        self.byy = np.zeros(tmp)
        self.bzz = np.zeros(tmp)
        self.bxy = np.zeros(tmp)
        self.byz = np.zeros(tmp)
        self.bxz = np.zeros(tmp)

        self.gx = np.zeros(tmp)
        self.gy = np.zeros(tmp)
        self.gz = np.zeros(tmp)
        self.gxx = np.zeros(tmp)
        self.gyy = np.zeros(tmp)
        self.gzz = np.zeros(tmp)
        self.gxy = np.zeros(tmp)
```

```
        self.gyz = np.zeros(tmp)
        self.gxz = np.zeros(tmp)

        self.coords = np.zeros((len(self.xyall), len(self.xyall), 2))

    def calc_all(self):
        """ calc all """
        self.init_grids()

        ma, mb, mc = dircos(self.minc, self.mdec, self.azim)
        fa, fb, fc = dircos(self.inc, self.dec, self.azim)

        mr = self.mstrength*np.array([ma, mb, mc])*100
        mi = self.susc*self.hintn/(4*np.pi)*np.array([fa, fb, fc])
        m3 = mr+mi
        m = np.sqrt(m3 @ m3)
        m3 /= m
        self.cx, self.cy, self.cz = m3

        self.mt = m


        print('Q-ratio', (mr@mr)/(mi@mi), 'mt', self.mt)


        const = m
        for i, y in enumerate(self.xyall):
            for j, x in enumerate(self.xyall):
                self.bx[-i-1, j] = self.fsum(self.Bx, x, y, self.height)
                self.by[-i-1, j] = self.fsum(self.By, x, y, self.height)
                self.bz[-i-1, j] = self.fsum(self.Bz, x, y, self.height)
                self.bxx[-i-1, j] = self.fsum(self.Bxx, x, y, self.height)
                self.byy[-i-1, j] = self.fsum(self.Byy, x, y, self.height)
                self.bzz[-i-1, j] = self.fsum(self.Bzz, x, y, self.height)
                self.bxy[-i-1, j] = self.fsum(self.Bxy, x, y, self.height)
                self.byz[-i-1, j] = self.fsum(self.Byz, x, y, self.height)
                self.bxz[-i-1, j] = self.fsum(self.Bxz, x, y, self.height)
                self.coords[-i-1, j, 0] = x
                self.coords[-i-1, j, 1] = y

        self.bx *= const
        self.by *= const
        self.bz *= const
        self.bxx *= const
        self.byy *= const
        self.bzz *= const
        self.byz *= const
        self.bxz *= const
        self.bxy *= const

        for j, x in enumerate(self.xyall):
            for i, y in enumerate(self.xyall):
                self.gx[-i-1, j] = self.fsum(self.Gx, x, y, 0.0)
                self.gy[-i-1, j] = self.fsum(self.Gy, x, y, 0.0)
                self.gz[-i-1, j] = self.fsum(self.Gz, x, y, 0.0)
                self.gxx[-i-1, j] = self.fsum(self.Gxx, x, y, 0.0)
                self.gyy[-i-1, j] = self.fsum(self.Gyy, x, y, 0.0)
                self.gzz[-i-1, j] = self.fsum(self.Gzz, x, y, 0.0)
                self.gxy[-i-1, j] = self.fsum(self.Gxy, x, y, 0.0)
                self.gyz[-i-1, j] = self.fsum(self.Gyz, x, y, 0.0)
                self.gxz[-i-1, j] = self.fsum(self.Gxz, x, y, 0.0)

        constg = (self.dens-self.bdens)*self.Gc
        self.gx *= constg
        self.gy *= constg
        self.gz *= constg
        self.gxx *= constg
        self.gyy *= constg
        self.gzz *= constg
        self.gyz *= constg
        self.gxz *= constg
        self.gxy *= constg
```

```
#         self.magval = self.cx*self.bx+self.cy*self.by+self.cz*self.bz
        self.grvval = self.gz
        self.magval = np.sqrt((self.bx+self.hintn*self.cx)**2 +
                              (self.by+self.hintn*self.cy)**2 +
                              (self.bz+self.hintn*self.cz)**2)-self.hintn

    def fsum(self, func, x, y, z):
        """ function """
        x1 = (x-self.u[0])
        x2 = (x-self.u[1])
        y2 = -(y-self.v[0])
        y1 = -(y-self.v[1])
        z2 = -(z-self.w[0])
        z1 = -(z-self.w[1])

#         y1 = (y-self.v[0])
#         y2 = (y-self.v[1])
#         z2 = -(z-self.w[0])
#         z1 = -(z-self.w[1])

        tmp = (func(x2, y2, z2) - func(x1, y2, z2) - func(x2, y1, z2) +
               func(x1, y1, z2) - func(x2, y2, z1) + func(x1, y2, z1) +
               func(x2, y1, z1) - func(x1, y1, z1))

        return tmp

    def getabg(self):
        """ gets alpha, beta and gamma """

        a, b, g = dircos(self.inc, self.dec, self.azim)

        return a, b, g

    def plot_mag(self):
        """ plots mag components """

        extent = (0, self.rc, 0, self.rc)
        x1, x2 = self.u
        y1, y2 = self.v

        plt.figure(figsize=(8, 8))

        ax = plt.subplot(4, 3, 1)
        plt.ylabel('Distance (m)')
        plt.xlabel('Distance (m)')
        im = plt.imshow(self.bx, vmin=-500, vmax=500, extent=extent)
        ply = plt.Line2D([x1, x1, x2, x2, x1], [y2, y1, y1, y2, y2], color='k')
        plt.gca().add_line(ply)
        plt.xticks([0, self.rc])
        plt.yticks([0, self.rc], rotation='vertical')
        plt.title('(a)', loc='left')
        divider = make_axes_locatable(ax)
        cax = divider.append_axes("right", size="5%", pad=0.05)
        plt.title('$B_{x}$ (nT)', size='medium')
        plt.colorbar(im, cax=cax, ticks=[-500, 0, 500])

        ax = plt.subplot(4, 3, 2)
        plt.ylabel('Distance (m)')
        plt.xlabel('Distance (m)')
        im = plt.imshow(self.by, vmin=-500, vmax=500, extent=extent)
        ply = plt.Line2D([x1, x1, x2, x2, x1], [y2, y1, y1, y2, y2], color='k')
        plt.gca().add_line(ply)
        plt.xticks([0, self.rc])
        plt.yticks([0, self.rc], rotation='vertical')
        plt.title('(b)', loc='left')
        divider = make_axes_locatable(ax)
        cax = divider.append_axes("right", size="5%", pad=0.05)
        plt.title('$B_{y}$ (nT)', size='medium')
        plt.colorbar(im, cax=cax, ticks=[-500, 0, 500])

        ax = plt.subplot(4, 3, 3)
        plt.ylabel('Distance (m)')
        plt.xlabel('Distance (m)')
```

```
im = plt.imshow(self.bz, vmin=-500, vmax=500, extent=extent)
ply = plt.Line2D([x1, x1, x2, x2, x1], [y2, y1, y1, y2, y2], color='k')
plt.gca().add_line(ply)
plt.xticks([0, self.rc])
plt.yticks([0, self.rc], rotation='vertical')
plt.title('(c)', loc='left')
divider = make_axes_locatable(ax)
cax = divider.append_axes("right", size="5%", pad=0.05)
plt.title('$B_{z}$ (nT)', size='medium')
plt.colorbar(im, cax=cax, ticks=[-500, 0, 500])

ax = plt.subplot(4, 3, 4)
plt.ylabel('Distance (m)')
plt.xlabel('Distance (m)')
im = plt.imshow(self.bxx, vmin=-10, vmax=10, extent=extent)
ply = plt.Line2D([x1, x1, x2, x2, x1], [y2, y1, y1, y2, y2], color='k')
plt.gca().add_line(ply)
plt.xticks([0, self.rc])
plt.yticks([0, self.rc], rotation='vertical')
plt.title('(d)', loc='left')
divider = make_axes_locatable(ax)
cax = divider.append_axes("right", size="5%", pad=0.05)
plt.title('$B_{xx}$ (nT/m)', size='medium')
plt.colorbar(im, cax=cax, ticks=[-10, 0, 10])

ax = plt.subplot(4, 3, 8)
plt.ylabel('Distance (m)')
plt.xlabel('Distance (m)')
im = plt.imshow(self.byy, vmin=-10, vmax=10, extent=extent)
ply = plt.Line2D([x1, x1, x2, x2, x1], [y2, y1, y1, y2, y2], color='k')
plt.gca().add_line(ply)
plt.xticks([0, self.rc])
plt.yticks([0, self.rc], rotation='vertical')
plt.title('(g)', loc='left')
divider = make_axes_locatable(ax)
cax = divider.append_axes("right", size="5%", pad=0.05)
plt.title('$B_{yy}$ (nT/m)', size='medium')
plt.colorbar(im, cax=cax, ticks=[-10, 0, 10])

ax = plt.subplot(4, 3, 12)
plt.ylabel('Distance (m)')
plt.xlabel('Distance (m)')
im = plt.imshow(self.bzz, vmin=-10, vmax=10, extent=extent)
ply = plt.Line2D([x1, x1, x2, x2, x1], [y2, y1, y1, y2, y2], color='k')
plt.gca().add_line(ply)
plt.xticks([0, self.rc])
plt.yticks([0, self.rc], rotation='vertical')
plt.title('(j)', loc='left')
divider = make_axes_locatable(ax)
cax = divider.append axes("right", size="5%", pad=0.05)
plt.title('$B_{zz}$ (nT/m)', size='medium')
plt.colorbar(im, cax=cax, ticks=[-10, 0, 10])

ax = plt.subplot(4, 3, 5)
plt.ylabel('Distance (m)')
plt.xlabel('Distance (m)')
im = plt.imshow(self.bxy, vmin=-5, vmax=5, extent=extent)
ply = plt.Line2D([x1, x1, x2, x2, x1], [y2, y1, y1, y2, y2], color='k')
plt.gca().add_line(ply)
plt.xticks([0, self.rc])
plt.yticks([0, self.rc], rotation='vertical')
plt.title('(e)', loc='left')
divider = make_axes_locatable(ax)
cax = divider.append_axes("right", size="5%", pad=0.05)
plt.title('$B_{xy}$ (nT/m)', size='medium')
plt.colorbar(im, cax=cax, ticks=[-5, 0, 5])

ax = plt.subplot(4, 3, 9)
plt.ylabel('Distance (m)')
plt.xlabel('Distance (m)')
im = plt.imshow(self.byz, vmin=-10, vmax=10, extent=extent)
ply = plt.Line2D([x1, x1, x2, x2, x1], [y2, y1, y1, y2, y2], color='k')
plt.gca().add_line(ply)
```

```
        plt.xticks([0, self.rc])
        plt.yticks([0, self.rc], rotation='vertical')
        plt.title('(h)', loc='left')
        divider = make_axes_locatable(ax)
        cax = divider.append_axes("right", size="5%", pad=0.05)
        plt.title('$B_{yz}$ (nT/m)', size='medium')
        plt.colorbar(im, cax=cax, ticks=[-10, 0, 10])

        ax = plt.subplot(4, 3, 6)
        plt.ylabel('Distance (m)')
        plt.xlabel('Distance (m)')
        im = plt.imshow(self.bxz, vmin=-10, vmax=10, extent=extent)
        ply = plt.Line2D([x1, x1, x2, x2, x1], [y2, y1, y1, y2, y2], color='k')
        plt.gca().add_line(ply)
        plt.xticks([0, self.rc])
        plt.yticks([0, self.rc], rotation='vertical')
        plt.title('(f)', loc='left')
        divider = make_axes_locatable(ax)
        cax = divider.append_axes("right", size="5%", pad=0.05)
        plt.title('$B_{xz}$ (nT/m)', size='medium')
        plt.colorbar(im, cax=cax, ticks=[-10, 0, 10])

        ax = plt.subplot(4, 3, 10)
        plt.ylabel('Distance (m)')
        plt.xlabel('Distance (m)')
        dtmp = self.magval
        im = plt.imshow(dtmp, vmin=-500, vmax=500, extent=extent)
        ply = plt.Line2D([x1, x1, x2, x2, x1], [y2, y1, y1, y2, y2], color='k')
        plt.gca().add_line(ply)
        plt.xticks([0, self.rc])
        plt.yticks([0, self.rc], rotation='vertical')
        plt.title('(i)', loc='left')
        divider = make_axes_locatable(ax)
        cax = divider.append_axes("right", size="5%", pad=0.05)
        plt.title('$B_{tmi}$ (nT)', size='medium')
        plt.colorbar(im, cax=cax, ticks=[-500, 0, 500])

        plt.tight_layout()
        plt.show()

#         print('Manual calculations for gradient')
#         plt.figure(figsize=(8, 8))
#
#         bxy1, bxx = np.gradient(self.bx[::-1], self.dxy)
#         byy, bxy = np.gradient(self.by[::-1], self.dxy)
#         byz, bxz = np.gradient(self.bz[::-1], self.dxy)
#         bxy = bxy[::-1]
#         bxy1 = bxy1[::-1]
#         byy = byy[::-1]
#         bxx = bxx[::-1]
#         byz = byz[::-1]
#         bxz = bxz[::-1]
#         bzz = -bxx-byy
#
#         ax = plt.subplot(4, 3, 4)
#         plt.ylabel('Distance (m)')
#         plt.xlabel('Distance (m)')
#         im = plt.imshow(bxx, cmap=cm, vmin=-10, vmax=10, extent=extent)
#         ply = plt.Line2D([x1, x1, x2, x2, x1], [y2, y1, y1, y2, y2], c='k')
#         plt.gca().add line(ply)
#         plt.xticks([0, self.rc])
#         plt.yticks([0, self.rc], rotation='vertical')
#         plt.title('(d)', loc='left')
#         divider = make axes locatable(ax)
#         cax = divider.append axes("right", size="5%", pad=0.05)
#         plt.title('$B {xx}$ (nT/m)', size='medium')
#         plt.colorbar(im, cax=cax, ticks=[-10, 0, 10])
#
#         ax = plt.subplot(4, 3, 8)
#         plt.ylabel('Distance (m)')
#         plt.xlabel('Distance (m)')
#         im = plt.imshow(byy, cmap=cm, vmin=-10, vmax=10, extent=extent)
#         ply = plt.Line2D([x1, x1, x2, x2, x1], [y2, y1, y1, y2, y2], c='k')
```

```
#         plt.gca().add_line(ply)
#         plt.xticks([0, self.rc])
#         plt.yticks([0, self.rc], rotation='vertical')
#         plt.title('(g)', loc='left')
#         divider = make axes locatable(ax)
#         cax = divider.append_axes("right", size="5%", pad=0.05)
#         plt.title('$B_{yy}$ (nT/m)', size='medium')
#         plt.colorbar(im, cax=cax, ticks=[-10, 0, 10])
#
#         ax = plt.subplot(4, 3, 12)
#         plt.ylabel('Distance (m)')
#         plt.xlabel('Distance (m)')
#         im = plt.imshow(bzz, cmap=cm, vmin=-10, vmax=10, extent=extent)
#         ply = plt.Line2D([x1, x1, x2, x2, x1], [y2, y1, y1, y2, y2], c='k')
#         plt.gca().add line(ply)
#         plt.xticks([0, self.rc])
#         plt.yticks([0, self.rc], rotation='vertical')
#         plt.title('(j)', loc='left')
#         divider = make axes locatable(ax)
#         cax = divider.append axes("right", size="5%", pad=0.05)
#         plt.title('$B {zz}$ (nT/m)', size='medium')
#         plt.colorbar(im, cax=cax, ticks=[-10, 0, 10])
#
#         ax = plt.subplot(4, 3, 5)
#         plt.ylabel('Distance (m)')
#         plt.xlabel('Distance (m)')
#         im = plt.imshow(bxy, cmap=cm, vmin=-5, vmax=5, extent=extent)
#         ply = plt.Line2D([x1, x1, x2, x2, x1], [y2, y1, y1, y2, y2], c='k')
#         plt.gca().add_line(ply)
#         plt.xticks([0, self.rc])
#         plt.yticks([0, self.rc], rotation='vertical')
#         plt.title('(e)', loc='left')
#         divider = make axes locatable(ax)
#         cax = divider.append_axes("right", size="5%", pad=0.05)
#         plt.title('$B_{xy}$ (nT/m)', size='medium')
#         plt.colorbar(im, cax=cax, ticks=[-5, 0, 5])
#
#         ax = plt.subplot(4, 3, 7)
#         plt.ylabel('Distance (m)')
#         plt.xlabel('Distance (m)')
#         im = plt.imshow(bxy1, cmap=cm, vmin=-5, vmax=5, extent=extent)
#         ply = plt.Line2D([x1, x1, x2, x2, x1], [y2, y1, y1, y2, y2], c='k')
#         plt.gca().add line(ply)
#         plt.xticks([0, self.rc])
#         plt.yticks([0, self.rc], rotation='vertical')
#         plt.title('(e)', loc='left')
#         divider = make axes locatable(ax)
#         cax = divider.append axes("right", size="5%", pad=0.05)
#         plt.title('$B {xy1}$ (nT/m)', size='medium')
#         plt.colorbar(im, cax=cax, ticks=[-5, 0, 5])
#
#
#         ax = plt.subplot(4, 3, 9)
#         plt.ylabel('Distance (m)')
#         plt.xlabel('Distance (m)')
#         im = plt.imshow(byz, cmap=cm, vmin=-10, vmax=10, extent=extent)
#         ply = plt.Line2D([x1, x1, x2, x2, x1], [y2, y1, y1, y2, y2], c='k')
#         plt.gca().add line(ply)
#         plt.xticks([0, self.rc])
#         plt.yticks([0, self.rc], rotation='vertical')
#         plt.title('(h)', loc='left')
#         divider = make_axes_locatable(ax)
#         cax = divider.append axes("right", size="5%", pad=0.05)
#         plt.title('$B {yz}$ (nT/m)', size='medium')
#         plt.colorbar(im, cax=cax, ticks=[-10, 0, 10])
#
#         ax = plt.subplot(4, 3, 6)
#         plt.ylabel('Distance (m)')
#         plt.xlabel('Distance (m)')
#         im = plt.imshow(bxz, cmap=cm, vmin=-10, vmax=10, extent=extent)
#         ply = plt.Line2D([x1, x1, x2, x2, x1], [y2, y1, y1, y2, y2], c='k')
#         plt.gca().add line(ply)
#         plt.xticks([0, self.rc])
```

```
#         plt.yticks([0, self.rc], rotation='vertical')
#         plt.title('(f)', loc='left')
#         divider = make_axes_locatable(ax)
#         cax = divider.append_axes("right", size="5%", pad=0.05)
#         plt.title('$B_{xz}$ (nT/m)', size='medium')
#         plt.colorbar(im, cax=cax, ticks=[-10, 0, 10])
#
#         plt.tight_layout()
#         plt.show()

    def plot_grav(self):
        """ Plots gravity """

        plt.figure(figsize=(8, 8))

        plt.subplot(4, 3, 1)
        plt.imshow(self.gx)
        plt.title('gx')
        plt.subplot(4, 3, 2)
        plt.imshow(self.gy)
        plt.title('gy')
        plt.subplot(4, 3, 3)
        plt.imshow(self.gz)
        plt.title('gz')
        plt.subplot(4, 3, 4)
        plt.imshow(self.gxx)
        plt.title('gxx')
        plt.subplot(4, 3, 8)
        plt.imshow(self.gyy)
        plt.title('gyy')
        plt.subplot(4, 3, 12)
        plt.imshow(self.gzz)
        plt.title('gzz')
        plt.subplot(4, 3, 5)
        plt.imshow(self.gxy)
        plt.title('gxy')
        plt.subplot(4, 3, 9)
        plt.imshow(self.gyz)
        plt.title('gyz')
        plt.subplot(4, 3, 6)
        plt.imshow(self.gxz)
        plt.title('gxz')

        plt.tight_layout()
        plt.show()

#         plt.figure(figsize=(8, 8))
#
#         gxy, gxx = np.gradient(self.gx[::-1], self.dxy)
#         gxy = gxy[::-1]
#         gxx = gxx[::-1]
#         gyy, gyx = np.gradient(self.gy[::-1], self.dxy)
#         gyy = gyy[::-1]
#         gyx = gyx[::-1]
#         gyz, gxz = np.gradient(self.gz[::-1], self.dxy)
#         gyz = gyz[::-1]
#         gxz = gxz[::-1]
#         gzz = -gxx-gyy
#
#         plt.subplot(4, 3, 4)
#         plt.imshow(gxx)
#         plt.title('gxx')
#         plt.subplot(4, 3, 8)
#         plt.imshow(gyy)
#         plt.title('gyy')
#         plt.subplot(4, 3, 12)
#         plt.imshow(gzz)
#         plt.title('gzz')
#         plt.subplot(4, 3, 5)
#         plt.imshow(gxy)
#         plt.title('gxy')
#         plt.subplot(4, 3, 9)
#         plt.imshow(gyz)
```

```
#         plt.title('gyz')
#         plt.subplot(4, 3, 6)
#         plt.imshow(gxz)
#         plt.title('gxz')
#         plt.subplot(4, 3, 7)
#         plt.imshow(gyx)
#         plt.title('gyx')
#
#         plt.tight layout()
#         plt.show()

    def regrid(self, data):
        """ fills holes """
        mask = np.logical_not(np.isnan(data))

        xx, yy = np.meshgrid(np.arange(data.shape[1]),
                             np.arange(data.shape[0]))
        xym = np.vstack((np.ravel(xx[mask]), np.ravel(yy[mask]))).T
        data0 = np.ravel(data[:, :][mask])
        interp0 = si.NearestNDInterpolator(xym, data0)
        result0 = interp0(np.ravel(xx), np.ravel(yy)).reshape(xx.shape)

        result0 = np.ma.masked_invalid(result0)

        return result0

    def Gx(self, x, y, z):
        """ function """
        r = np.sqrt(x**2+y**2+z**2)

        x = np.array(x)
        y = np.array(y)
        z = np.array(z)
        tmp = x*np.arctan2(y*z, x*r)-y*np.log(r+z)-z*np.log(r+y)

        return -tmp

    def Gy(self, x, y, z):
        """ function """
        r = np.sqrt(x**2+y**2+z**2)
        x = np.array(x)
        y = np.array(y)
        z = np.array(z)

        tmp = y*np.arctan2(x*z, y*r)-x*np.log(r+z)-z*np.log(x+r)

        return -tmp

    def Gz(self, x, y, z):
        """ function """
        r = np.sqrt(x**2+y**2+z**2)

        x = np.array(x)
        y = np.array(y)
        z = np.array(z)

#         tmp = z*np.arctan2(x*y, z*r)-x*np.log(r+y)-y*np.log(x+r)
        tmp = z*np.arctan((x*y)/(z*r))-x*np.log(r+y)-y*np.log(x+r)
        return -tmp

    def Gxx(self, x, y, z):
        """ function """
        r = np.sqrt(x**2+y**2+z**2)

        x = np.array(x)
        y = np.array(y)
        z = np.array(z)

        tmp = np.arctan2(y*z, x*r)
#         return -tmp
        return -tmp

    def Gyy(self, x, y, z):
```

```
        """ function """
        r = np.sqrt(x**2+y**2+z**2)

        x = np.array(x)
        y = np.array(y)
        z = np.array(z)

        tmp = np.arctan2(x*z, y*r)
#        return -tmp
        return -tmp

    def Gzz(self, x, y, z):
        """ function """
        r = np.sqrt(x**2+y**2+z**2)

        x = np.array(x)
        y = np.array(y)
        z = np.array(z)

        tmp = np.arctan2(x*y, z*r)
        return -tmp

    def Gxy(self, x, y, z):
        """ function """
        r = np.sqrt(x**2+y**2+z**2)

        if z+r == 0:
            tmp = np.nan
        else:
            tmp = -np.log(r+z)
        return -tmp

    def Gyz(self, x, y, z):
        """ function """
        r = np.sqrt(x**2+y**2+z**2)
        tmp = -np.log(x+r)
        return -tmp

    def Gxz(self, x, y, z):
        """ function """
        r = np.sqrt(x**2+y**2+z**2)
        tmp = -np.log(r+y)
        return -tmp

    def Bx(self, x, y, z):
        """ function """
        a, b, g = self.getabg()
        tmp = a*self.Gxx(x, y, z) + b*self.Gxy(x, y, z) + g*self.Gxz(x, y, z)

        return tmp

    def By(self, x, y, z):
        """ function """
        a, b, g = self.getabg()
        tmp = a*self.Gxy(x, y, z) + b*self.Gyy(x, y, z) + g*self.Gyz(x, y, z)
        return tmp

    def Bz(self, x, y, z):
        """ function """
        a, b, g = self.getabg()
        tmp = a*self.Gxz(x, y, z) + b*self.Gyz(x, y, z) + g*self.Gzz(x, y, z)

        return tmp

    def Bxx(self, x, y, z):
        """ function """
        a, b, g = self.getabg()
        r = np.sqrt(x**2+y**2+z**2)
        if x == 0 and y == 0:
            tmp = 0
        else:
            tmp = (a*y*z*(r**2 + x**2)/(r*(r**2*x**2 + y**2*z**2)) +
                   b*x/(r**2 + r*z) + g*x/(r**2 + r*y))
```

```
#             tmp = (a*y*z*(r**2 + x**2)/(r*(r**2*x**2 + y**2*z**2)) +
#                    b*x/(r**2 - r*z) + g*x/(r**2 - r*y))

        return tmp

    def Byy(self, x, y, z):
        """ function """
        a, b, g = self.getabg()
        r = np.sqrt(x**2+y**2+z**2)

        if x == 0 and y == 0:
            tmp = 0
        else:
            tmp = (a*y/(r**2 + r*z) +
                   b*x*z*(r**2 + y**2)/(r*(r**2*y**2 + x**2*z**2)) +
                   g*y/(r**2 + r*x))
#             tmp = (-a*y/(r**2 - r*z) - g*y/(r**2 + r*x) -
#                    b*x*z*(r**2 + y**2)/(r*(r**2*y**2 + x**2*z**2)))

        return tmp

    def Bzz(self, x, y, z):
        """ function """
        a, b, g = self.getabg()
        r = np.sqrt(x**2+y**2+z**2)

        tmp = (a*z/(r**2 + r*y) + b*z/(r**2 + r*x) +
               g*x*y*(r**2 + z**2)/(r*(r**2*z**2 + x**2*y**2)))
#         tmp = (-a*z/(r**2 - r*y) - b*z/(r**2 + r*x) -
#                g*x*y*(r**2 + z**2)/(r*(r**2*z**2 + x**2*y**2)))

        return tmp

    def Bxy(self, x, y, z):
        """ function """
        a, b, g = self.getabg()
        r = np.sqrt(x**2+y**2+z**2)

        if x == 0 and y == 0:
            tmp = g/r
        else:
            tmp = -a*x*z/(r*(x**2 + y**2)) + b*y/(r**2 + r*z) + g/r
#             tmp = a*x*z/(r*(x**2 + y**2)) - b*y/(r**2 - r*z) + g/r

        return tmp

    def Byz(self, x, y, z):
        """ function """
        a, b, g = self.getabg()
        r = np.sqrt(x**2+y**2+z**2)

#         tmp = a/r + b*x*y/(r*(y**2 + z**2)) - g*z/(r**2 + r*x)
        tmp = a/r - b*x*y/(r*(y**2 + z**2)) + g*z/(r**2 + r*x)

        return tmp

    def Bxz(self, x, y, z):
        """ function """
        a, b, g = self.getabg()
        r = np.sqrt(x**2+y**2+z**2)

#         tmp = a*x*y/(r*(x**2 + z**2)) + b/r - g*z/(r**2 - r*y)
        tmp = -a*x*y/(r*(x**2 + z**2)) + b/r + g*z/(r**2 + r*y)

        return tmp


def importmod3d(filename):
    """
    routine to convert a dictionary to an lmod

    Parameters
    ----------
```

```
    filename : str
        input filename of model file

    Returns
    -------
    lmod : LithModel
        model class
    """
    pre = ''
    lmod = LithModel()
    lmod.griddata.clear()
    lmod.lith_list.clear()
    indict = np.load(filename)
    lithkeys = indict[pre+'lithkeys']

    lmod.gregional = indict[pre+'gregional']
    lmod.ght = indict[pre+'ght']
    lmod.mht = indict[pre+'mht']
    lmod.numx = indict[pre+'numx']
    lmod.numy = indict[pre+'numy']
    lmod.numz = indict[pre+'numz']
    lmod.dxy = indict[pre+'dxy']
    lmod.d_z = indict[pre+'d_z']
    lmod.lith_index = indict[pre+'lith_index']
    lmod.curprof = 0
    lmod.curlayer = 0
    lmod.xrange = np.array(indict[pre+'xrange']).tolist()
    lmod.yrange = np.array(indict[pre+'yrange']).tolist()
    lmod.zrange = np.array(indict[pre+'zrange']).tolist()
    if pre+'custprofx' in indict:
        lmod.custprofx = np.asscalar(indict[pre+'custprofx'])
    else:
        lmod.custprofx = {0: (lmod.xrange[0], lmod.xrange[1])}
    if pre+'custprofy' in indict:
        lmod.custprofy = np.asscalar(indict[pre+'custprofy'])
    else:
        lmod.custprofy = {0: (lmod.yrange[0], lmod.yrange[0])}

    lmod.mlut = np.asscalar(indict[pre+'mlut'])
    lmod.init_calc_grids()

    lmod.griddata = np.asscalar(indict[pre+'griddata'])

    for i in lmod.griddata:
        lmod.griddata[i].data = np.ma.array(lmod.griddata[i].data)

    # This gets rid of a legacy variable name
    for i in lmod.griddata:
        if not hasattr(lmod.griddata[i], 'dataid'):
            lmod.griddata[i].dataid = ''
        if hasattr(lmod.griddata[i], 'bandid'):
            if lmod.griddata[i].dataid == '':
                lmod.griddata[i].dataid = lmod.griddata[i].bandid
            del lmod.griddata[i].bandid

    wktfin = None
    for i in lmod.griddata:
        wkt = lmod.griddata[i].wkt
        if wkt != '' and wkt is not None:
            wktfin = wkt

    if wktfin is not None:
        for i in lmod.griddata:
            wkt = lmod.griddata[i].wkt
            if wkt == '' or wkt is None:
                lmod.griddata[i].wkt = wktfin

# Section to load lithologies.
    lmod.lith_list['Background'] = GeoData()

    for itxt in lithkeys:
        if itxt != 'Background':
            lmod.lith_list[itxt] = GeoData()
```

```
            lmod.lith_list[itxt].hintn = np.asscalar(indict[pre+itxt+'_hintn'])
            lmod.lith_list[itxt].finc = np.asscalar(indict[pre+itxt+'_finc'])
            lmod.lith_list[itxt].fdec = np.asscalar(indict[pre+itxt+'_fdec'])
            lmod.lith_list[itxt].zobsm = np.asscalar(indict[pre+itxt+'_zobsm'])
            lmod.lith_list[itxt].susc = np.asscalar(indict[pre+itxt+'_susc'])
            lmod.lith_list[itxt].mstrength = np.asscalar(
                indict[pre+itxt+'_mstrength'])
            lmod.lith_list[itxt].qratio = np.asscalar(
                indict[pre+itxt+'_qratio'])
            lmod.lith_list[itxt].minc = np.asscalar(indict[pre+itxt+'_minc'])
            lmod.lith_list[itxt].mdec = np.asscalar(indict[pre+itxt+'_mdec'])
            lmod.lith_list[itxt].density = np.asscalar(
                indict[pre+itxt+'_density'])
            lmod.lith_list[itxt].bdensity = np.asscalar(
                indict[pre+itxt+'_bdensity'])
            lmod.lith_list[itxt].lith_index = np.asscalar(
                indict[pre+itxt+'_lith_index'])
            lmod.lith_list[itxt].g_cols = np.asscalar(indict[pre+itxt+'_numx'])
            lmod.lith_list[itxt].g_rows = np.asscalar(indict[pre+itxt+'_numy'])
            lmod.lith_list[itxt].numz = np.asscalar(indict[pre+itxt+'_numz'])
            lmod.lith_list[itxt].g_dxy = np.asscalar(indict[pre+itxt+'_dxy'])
            lmod.lith_list[itxt].dxy = np.asscalar(indict[pre+itxt+'_dxy'])
            lmod.lith_list[itxt].d_z = np.asscalar(indict[pre+itxt+'_d_z'])
            lmod.lith_list[itxt].zobsm = np.asscalar(indict[pre+itxt+'_zobsm'])
            lmod.lith_list[itxt].zobsg = np.asscalar(indict[pre+itxt+'_zobsg'])
            lmod.lith_list[itxt].modified = True
            lmod.lith_list[itxt].set_xyz12()

    return lmod


def save_layer(mlist):
    """
    Routine saves the mlayer and glayer to a file

    Parameters
    ----------
    mlist : list
        list of gridded layers

    Returns
    -------
    outfile : str
        temporary output filename
    """
    outfile = tempfile.TemporaryFile()

    outdict = {}

    outdict['mlayers'] = mlist[1].mlayers
    outdict['glayers'] = mlist[1].glayers

    np.savez(outfile, **outdict)
    outfile.seek(0)

    mlist[1].mlayers = None
    mlist[1].glayers = None

    return outfile


def data_to_gdal_mem(data, gtr, wkt, cols, rows, nodata=False):
    """
    Data to GDAL mem format

    Parameters
    ----------
    data : PyGMI Data
        PyGMI Dataset
    gtr : tuple
        Geotransform
    wkt : str
```

```
        Projection in wkt (well known text) format
    cols : int
        columns
    rows : int
        rows
    nodata : bool, optional
        no data

    Returns
    -------
    src : GDAL mem format
    """
    data.data = np.ma.array(data.data)
    dtype = data.data.dtype
# Get rid of array() which can break driver.create later
    cols = int(cols)
    rows = int(rows)

    if dtype == np.uint8:
        fmt = gdal.GDT_Byte
    elif dtype == np.int32:
        fmt = gdal.GDT_Int32
    elif dtype == np.float64:
        fmt = gdal.GDT_Float64
    else:
        fmt = gdal.GDT_Float32

    driver = gdal.GetDriverByName('MEM')
    src = driver.Create('', cols, rows, 1, fmt)

    src.SetGeoTransform(gtr)
    src.SetProjection(wkt)

    if nodata is False:
        if data.nullvalue is not None:
            src.GetRasterBand(1).SetNoDataValue(data.nullvalue)
        src.GetRasterBand(1).WriteArray(data.data)
    else:
        tmp = np.zeros((rows, cols))
        tmp = np.ma.masked_equal(tmp, 0)
        src.GetRasterBand(1).SetNoDataValue(0)  # Set to this because of Reproj
        src.GetRasterBand(1).WriteArray(tmp)

    return src


def gdal_to_dat(dest, bandid='Data'):
    """
    GDAL to Data format

    Parameters
    ----------
    dest - GDAL format
        GDAL format
    bandid - str
        band identity

    Returns
    -------
    dat : Data
        data
    """
    dat = Data()
    gtr = dest.GetGeoTransform()

    rtmp = dest.GetRasterBand(1)
    dat.data = rtmp.ReadAsArray()
    nval = rtmp.GetNoDataValue()

    dat.data = np.ma.masked_equal(dat.data, nval)
    dat.data.set_fill_value(nval)
    dat.data = np.ma.fix_invalid(dat.data)
```

```
    dat.nrofbands = dest.RasterCount
    dat.tlx = gtr[0]
    dat.tly = gtr[3]
    dat.dataid = bandid
    dat.nullvalue = nval
    dat.rows = dest.RasterYSize
    dat.cols = dest.RasterXSize
    dat.xdim = abs(gtr[1])
    dat.ydim = abs(gtr[5])
    dat.wkt = dest.GetProjection()
    dat.gtr = gtr

    return dat


def get_raster(ifile):
    """
    This function loads a raster dataset off the disk using the GDAL
    libraries. It returns the data in a PyGMI data object.

    Parameters
    ----------
    ifile : str
        filename to import

    Returns
    -------
    dat : PyGMI raster Data
        dataset imported
    """
    dat = []
    bname = ifile.split('/')[-1].rpartition('.')[0]+': '
    ifile = ifile[:]
    ext = ifile[-3:]
    custom_wkt = None

    # Envi Case
    if ext == 'hdr':
        ifile = ifile[:-4]
        tmp = glob.glob(ifile+'.dat')
        if tmp:
            ifile = tmp[0]

    if ext == 'ers':
        with open(ifile) as f:
            metadata = f.read()
            if 'STMLO' in metadata:
                clong = metadata.split('STMLO')[1][:2]

                orig = osr.SpatialReference()
                if 'CAPE' in metadata:
                    orig.ImportFromEPSG(4222)
                    orig.SetTM(0., float(clong), 1., 0., 0.)
                    orig.SetProjCS(r'Cape / TM'+clong)
                    custom_wkt = orig.ExportToWkt()
                elif 'WGS84' in metadata:
                    orig.ImportFromEPSG(4148)
                    orig.SetTM(0., float(clong), 1., 0., 0.)
                    orig.SetProjCS(r'Hartebeesthoek94 / TM'+clong)
                    custom_wkt = orig.ExportToWkt()

    dataset = gdal.Open(ifile, gdal.GA_ReadOnly)

    if dataset is None:
        return None

    gtr = dataset.GetGeoTransform()

    for i in range(dataset.RasterCount):
        rtmp = dataset.GetRasterBand(i+1)
        bandid = rtmp.GetDescription()
        nval = rtmp.GetNoDataValue()
```

```
            dat.append(Data())
            dat[i].data = rtmp.ReadAsArray()
            if dat[i].data.dtype.kind == 'i':
                if nval is None:
                    nval = 999999
                nval = int(nval)
            elif dat[i].data.dtype.kind == 'u':
                if nval is None:
                    nval = 0
                nval = int(nval)
            else:
                if nval is None:
                    nval = 1e+20
                nval = float(nval)
            if ext == 'ers' and nval == -1.0e+32:
                dat[i].data[np.ma.less_equal(dat[i].data, nval)] = -1.0e+32

        # Note that because the data is stored in a masked array, the array ends up
        # being double the size that it was on the disk.
            dat[i].data = np.ma.masked_invalid(dat[i].data)
            dat[i].data.mask = (np.ma.getmaskarray(dat[i].data) |
                                (dat[i].data == nval))
            if dat[i].data.mask.size == 1:
                dat[i].data.mask = (np.ma.make_mask_none(dat[i].data.shape) +
                                    np.ma.getmaskarray(dat[i].data))

            dat[i].nrofbands = dataset.RasterCount
            dat[i].tlx = gtr[0]
            dat[i].tly = gtr[3]
            if bandid == '':
                bandid = bname+str(i+1)
            dat[i].dataid = bandid
            if bandid[-1] == ')':
                dat[i].units = bandid[bandid.rfind('(')+1:-1]

            dat[i].nullvalue = nval
            dat[i].rows = dataset.RasterYSize
            dat[i].cols = dataset.RasterXSize
            dat[i].xdim = abs(gtr[1])
            dat[i].ydim = abs(gtr[5])
            dat[i].gtr = gtr

            if custom_wkt is None:
                srs = osr.SpatialReference()
                srs.ImportFromWkt(dataset.GetProjection())
                srs.AutoIdentifyEPSG()
                dat[i].wkt = srs.ExportToWkt()
            else:
                dat[i].wkt = custom_wkt

        return dat


    def gridmatch(lmod, ctxt, rtxt):
        """
        Matches the rows and columns of the second grid to the first grid


        Parameters
        ----------
        lmod : LithModel
            lithology model
        ctxt : str
            input grid discription 1
        rtxt : str
            input grid discription 2

        Returns
        -------
        dat.data : numpy array
            gridded data
        """
        rgrv = lmod.griddata[rtxt]
```

```
    cgrv = lmod.griddata[ctxt]

    data = rgrv
    data2 = cgrv
    orig_wkt = data.wkt
    orig_wkt2 = data2.wkt

    doffset = 0.0
    if data.data.min() <= 0:
        doffset = data.data.min()-1.
        data.data = data.data - doffset

    gtr0 = (data.tlx, data.xdim, 0.0, data.tly, 0.0, -data.ydim)
    gtr = (data2.tlx, data2.xdim, 0.0, data2.tly, 0.0, -data2.ydim)
    src = data_to_gdal_mem(data, gtr0, orig_wkt, data.cols, data.rows)
    dest = data_to_gdal_mem(data, gtr, orig_wkt2, data2.cols, data2.rows, True)

    gdal.ReprojectImage(src, dest, orig_wkt, orig_wkt2, gdal.GRA_Bilinear)

    dat = gdal_to_dat(dest, data.dataid)

    if doffset != 0.0:
        dat.data = dat.data + doffset
        data.data = data.data + doffset

    return dat.data


def calc_field(lmod, pbars=None, showtext=None, parent=None,
               showreports=False, magcalc=False):
    """ Calculate magnetic and gravity field

    This function calculates the magnetic and gravity field. It has two
    different modes of operation, by using the magcalc switch. If magcalc=True
    then magnetic fields are calculated, otherwize only gravity is calculated.

    Parameters
    ----------
    lmod : LithModel
        PyGMI lithological model
    pbars : module
        progress bar routine if available. (internal use)
    showtext : module
        showtext routine if available. (internal use)
    showreports : bool
        show extra reports
    magcalc : bool
        if true, calculates magnetic data, otherwize only gravity.

    Returns
    -------
    lmod.griddata : dictionary
        dictionary of items of type Data.
    """

    if showtext is None:
        showtext = print
    if pbars is not None:
        pbars.resetall(mmax=2*(len(lmod.lith_list)-1)+1)
        piter = pbars.iter
    else:
        piter = iter
    if np.max(lmod.lith_index) == -1:
        showtext('Error: Create a model first')
        return lmod.griddata

    # Init some variables for convenience
    lmod.update_lithlist()

    numx = int(lmod.numx)
    numy = int(lmod.numy)
    numz = int(lmod.numz)
```

```
    tmpfiles = {}

# model index
    modind = lmod.lith_index.copy()
    modindcheck = lmod.lith_index_old.copy()

    if modind.shape != modindcheck.shape:
        tmp = False
    else:
        tmp = (modind == modindcheck)

# If modind and modindcheck have different shapes, then tmp == False. The next
# line checks for that.

    if not isinstance(tmp, bool):
        modind[tmp] = -1
        modindcheck[tmp] = -1

    if np.unique(modind).size == 1:
        showtext('No changes to model!')
        return lmod.griddata

# get height corrections
    tmp = np.copy(lmod.lith_index)
    tmp[tmp > -1] = 0
    hcor = np.abs(tmp.sum(2))

#    if np.unique(modindcheck).size == 1 and np.unique(modindcheck)[0] == -1:
    for mlist in lmod.lith_list.items():
        mijk = mlist[1].lith_index
        if mijk not in modind and mijk not in modindcheck:
            continue
        if mlist[0] != 'Background':
            mlist[1].modified = True
            showtext(mlist[0]+':')
            if parent is not None:
                mlist[1].parent = parent
                mlist[1].pbars = parent.pbars
                mlist[1].showtext = parent.showtext
            if magcalc:
                mlist[1].calc_origin_mag(hcor)
            else:
                mlist[1].calc_origin_grav()
            tmpfiles[mlist[0]] = save_layer(mlist)
        lmod.tmpfiles = tmpfiles

    if showreports is True:
        showtext('Summing data')

# Get mlayers and glayers with correct rho and netmagn

    if pbars is not None:
        pbars.resetsub(maximum=(len(lmod.lith_list)-1))
        piter = pbars.iter

    mgvalin = np.zeros(numx*numy)
    mgval = np.zeros(numx*numy)

    hcorflat = numz-hcor.flatten()
    aaa = np.reshape(np.mgrid[0:numx, 0:numy], [2, numx*numy])

    for mlist in piter(lmod.lith_list.items()):
        if mlist[0] == 'Background':
            continue
        mijk = mlist[1].lith_index
        if mijk not in modind and mijk not in modindcheck:
            continue
        lmod.tmpfiles[mlist[0]].seek(0)

        mfile = np.load(lmod.tmpfiles[mlist[0]])

        if magcalc:
            mglayers = mfile['mlayers']
```

```
    else:
        mglayers = mfile['glayers']*mlist[1].rho()

    showtext('Summing '+mlist[0]+' (May become non-responsive' +
             ' during this calculation)')

    if np.unique(modind).size > 1 and mijk in modind:
        i, j, k = np.nonzero(modind == mijk)
        iuni = np.array(np.unique(i), dtype=np.int32)
        juni = np.array(np.unique(j), dtype=np.int32)
        kuni = np.array(np.unique(k), dtype=np.int32)

        for k in kuni:
            baba = sum_fields(k, mgval, numx, numy, modind, aaa[0],
                              aaa[1], mglayers, hcorflat, mijk, juni,
                              iuni)
            mgvalin += baba

    if np.unique(modindcheck).size > 1 and mijk in modindcheck:
        i, j, k = np.nonzero(modindcheck == mijk)
        iuni = np.array(np.unique(i), dtype=np.int32)
        juni = np.array(np.unique(j), dtype=np.int32)
        kuni = np.array(np.unique(k), dtype=np.int32)

        for k in kuni:
            baba = sum_fields(k, mgval, numx, numy, modindcheck,
                              aaa[0],
                              aaa[1], mglayers, hcorflat, mijk, juni,
                              iuni)
            mgvalin -= baba

    showtext('Done')

    if pbars is not None:
        pbars.incrmain()

mgvalin.resize([numx, numy])
mgvalin = mgvalin.T
mgvalin = mgvalin[::-1]
mgvalin = np.ma.array(mgvalin)

if np.unique(modindcheck).size > 1:
    if magcalc:
        mgvalin += lmod.griddata['Calculated Magnetics'].data
    else:
        mgvalin += lmod.griddata['Calculated Gravity'].data

if magcalc:
    lmod.griddata['Calculated Magnetics'].data = mgvalin
else:
    lmod.griddata['Calculated Gravity'].data = mgvalin

if ('Gravity Regional' in lmod.griddata and not magcalc and
        np.unique(modindcheck).size == 1):
    zfin = gridmatch(lmod, 'Calculated Gravity', 'Gravity Regional')
    lmod.griddata['Calculated Gravity'].data += zfin

if lmod.lith_index.max() <= 0:
    lmod.griddata['Calculated Magnetics'].data *= 0.
    lmod.griddata['Calculated Gravity'].data *= 0.

if 'Magnetic Dataset' in lmod.griddata:
    ztmp = gridmatch(lmod, 'Magnetic Dataset', 'Calculated Magnetics')
    lmod.griddata['Magnetic Residual'] = copy.deepcopy(
        lmod.griddata['Magnetic Dataset'])
    lmod.griddata['Magnetic Residual'].data = (
        lmod.griddata['Magnetic Dataset'].data - ztmp)
    lmod.griddata['Magnetic Residual'].dataid = 'Magnetic Residual'

if 'Gravity Dataset' in lmod.griddata:
    ztmp = gridmatch(lmod, 'Gravity Dataset', 'Calculated Gravity')
    lmod.griddata['Gravity Residual'] = copy.deepcopy(
        lmod.griddata['Gravity Dataset'])
```

```
            lmod.griddata['Gravity Residual'].data = (
                lmod.griddata['Gravity Dataset'].data - ztmp - lmod.gregional)
            lmod.griddata['Gravity Residual'].dataid = 'Gravity Residual'

        if parent is not None:
            tmp = [i for i in set(lmod.griddata.values())]
            parent.outdata['Raster'] = tmp
        showtext('Calculation Finished')
        if pbars is not None:
            pbars.maxall()

        lmod.lith_index_old = np.copy(lmod.lith_index)

        return lmod.griddata


@jit(nopython=True)
def sum_fields(k, mgval, numx, numy, modind, aaa0, aaa1, mlayers, hcorflat,
               mijk, jj, ii):
    """
    Sum Calculated magnetic or gravity data

    Parameters
    ----------
    ii : list
        list of x indices
    jj : list
        list of y indices
    k : int
        z index
    mgval : numpy array
        magnetic grid
    numx : int
        number of columns
    numy : int
        number of rows
    modind : numpy array
        modex indices
    aaa0 : numpy array
        relative x offset
    aaa1 : numpy array
        relative y offset
    hcorflat : numpy array
        relatrive z offset - height correction
    mlayers : numpy array
        multiple grids for each calculated layer
    mijk : int
        current lithology index

    Returns
    -------
    mgval : numpy array

    """

    b = numx*numy
    for j in range(b):
        mgval[j] = 0.

    for i in ii:
        xoff = numx-i
        for j in jj:
            yoff = numy-j
            if (modind[i, j, k] != mijk):
                continue
            for ijk in range(b):
                xoff2 = xoff + aaa0[ijk]
                yoff2 = aaa1[ijk]+yoff
                hcor2 = hcorflat[ijk]+k
                mgval[ijk] += mlayers[hcor2, xoff2, yoff2]

    return mgval
```

```
def gradientO4(f, *varargs):
    """
    Calculate the fourth-order-accurate gradient of an N-dimensional scalar
    function. Uses central differences on the interior and first differences
    on boundaries to give the same shape.

    Inputs:
      f -- An N-dimensional array giving samples of a scalar function
      varargs -- 0, 1, or N scalars giving the sample distances in each
      direction
    Outputs:
      N arrays of the same shape as f giving the derivative of f with respect
      to each dimension.

    from https://gist.github.com/deeplycloudy/1b9fa46d5290314d9be02a5156b48741

    """
    N = len(f.shape)  # number of dimensions
    n = len(varargs)
    if n == 0:
        dx = [1.0]*N
    elif n == 1:
        dx = [varargs[0]]*N
    elif n == N:
        dx = list(varargs)
    else:
        raise SyntaxError("invalid number of arguments")

    # use central differences on interior and first differences on endpoints
    outvals = []

    # create slice objects --- initially all are [:, :, ..., :]
    slice0 = [slice(None)]*N
    slice1 = [slice(None)]*N
    slice2 = [slice(None)]*N
    slice3 = [slice(None)]*N
    slice4 = [slice(None)]*N

    otype = f.dtype.char
    if otype not in ['f', 'd', 'F', 'D']:
        otype = 'd'

    for axis in range(N):
        # select out appropriate parts for this dimension
        out = np.zeros(f.shape, f.dtype.char)

        slice0[axis] = slice(2, -2)
        slice1[axis] = slice(None, -4)
        slice2[axis] = slice(1, -3)
        slice3[axis] = slice(3, -1)
        slice4[axis] = slice(4, None)
        # 1D equivalent -- out[2:-2] = (f[:4]-8*f[1:-3]+8*f[3:-1]-f[4:])/12.0
        out[slice0] = (f[slice1]-8.0*f[slice2]+8.0*f[slice3]-f[slice4])/12.0

        slice0[axis] = slice(None, 2)
        slice1[axis] = slice(1, 3)
        slice2[axis] = slice(None, 2)
        # 1D equivalent -- out[0:2] = (f[1:3] - f[0:2])
        out[slice0] = (f[slice1] - f[slice2])

        slice0[axis] = slice(-2, None)
        slice1[axis] = slice(-2, None)
        slice2[axis] = slice(-3, -1)
        # 1D equivalent -- out[-2:] = (f[-2:] - f[-3:-1])
        out[slice0] = (f[slice1] - f[slice2])

        # divide by step size
        outvals.append(out / dx[axis])

        # reset the slice object in this dimension to ":"
        slice0[axis] = slice(None)
        slice1[axis] = slice(None)
```

```
            slice2[axis] = slice(None)
            slice3[axis] = slice(None)
            slice4[axis] = slice(None)

        if N == 1:
            return outvals[0]

        return outvals


@jit(nopython=True)
def gbox(gval, xobs, yobs, numx, numy, z_0, x_1, y_1, z_1, x_2, y_2, z_2,
         x, y, z, isign):
    """
    GBOX routine by Blakely

    Subroutine GBOX computes the vertical attraction of a
    rectangular prism.  Sides of prism are parallel to x,y,z axes,
    and z axis is vertical down.

    Input parameters:
        Observation point is (x0,y0,z0).  The prism extends from x1
        to x2, from y1 to y2, and from z1 to z2 in the x, y, and z
        directions, respectively.  Density of prism is rho.  All
        distance parameters in units of m;

    Output parameters:
        Vertical attraction of gravity, g, in mGal/rho.
        Must still be multiplied by rho outside routine.
        Done this way for speed.

    Parameters
    ----------
    gval : numpy array
        gravity grid
    xobs : numpy array
        x observations
    yobs : numpy array
        y observations
    numx : int
        number of columns
    numy : int
        number of rows
    z 0 : float
        z observation point
    x 1 : float
        west side of cube
    y 1 : float
        south side of cube
    z 1 : float
        top of cube
    x 2 : float
        east side of cube
    y 2 : float
        north side of cube
    z 2 : float
        bottom of cube
    x : numpy array
        distance to x sides
    y : numpy array
        distance to y sides
    z : numpy array
        distance to z sides
    isign : numpy array
        calculation constants

    Returns
    -------
    gval : numpy array
        output gravity grid

    """
```

```
    z[0] = z_0-z_1
    z[1] = z_0-z_2

    for ii in range(numx):
        x[0] = xobs[ii]-x_1
        x[1] = xobs[ii]-x_2
        for jj in range(numy):
            y[0] = yobs[jj]-y_1
            y[1] = yobs[jj]-y_2
            sumi = 0.
            for i in range(2):
                for j in range(2):
                    for k in range(2):
                        rijk = np.sqrt(x[i]*x[i]+y[j]*y[j]+z[k]*z[k])
                        ijk = isign[i]*isign[j]*isign[k]
                        arg1 = np.arctan2(x[i]*y[j], z[k]*rijk)

                        if arg1 < 0.:
                            arg1 = arg1 + 2 * np.pi
                        arg2 = rijk+y[j]
                        arg3 = rijk+x[i]
                        arg2 = np.log(arg2)
                        arg3 = np.log(arg3)
                        sumi += ijk*(z[k]*arg1-x[i]*arg2-y[j]*arg3)
            gval[ii, jj] = sumi

    return gval


@jit(nopython=True)
def mbox(mval, xobs, yobs, numx, numy, z0, x1, y1, z1, x2, y2, fm1, fm2, fm3,
         fm4, fm5, fm6, alpha, beta):
    """
    MBOX routine by Blakely

    Subroutine MBOX computes the total field anomaly of an infinitely
    extended rectangular prism.  Sides of prism are parallel to x,y,z
    axes, and z is vertical down.  Bottom of prism extends to infinity.
    Two calls to mbox can provide the anomaly of a prism with finite
    thickness; e.g.,

        call mbox(x0,y0,z0,x1,y1,z1,x2,y2,mi,md,fi,fd,m,theta,t1)
        call mbox(x0,y0,z0,x1,y1,z2,x2,y2,mi,md,fi,fd,m,theta,t2)
        t=t1-t2

    Requires subroutine DIRCOS.  Method from Bhattacharyya (1964).

    Input parameters:
        Observation point is (x0,y0,z0).  Prism extends from x1 to
        x2, y1 to y2, and z1 to infinity in x, y, and z directions,
        respectively.  Magnetization defined by inclination mi,
        declination md, intensity m.  Ambient field defined by
        inclination fi and declination fd.  X axis has declination
        theta. Distance units are irrelevant but must be consistent.
        Angles are in degrees, with inclinations positive below
        horizontal and declinations positive east of true north.
        Magnetization in A/m.

    Output paramters:
        Total field anomaly t, in nT.

    Parameters
    ----------
    mval : numpy array
        gravity grid
    xobs : numpy array
        x observations
    yobs : numpy array
        y observations
    numx : int
        number of columns
    numy : int
        number of rows
```

```
    z_0 : float
        z observation point
    x 1 : float
        west side of cube
    y 1 : float
        south side of cube
    z_1 : float
        top of cube
    x 2 : float
        east side of cube
    y 2 : float
        north side of cube
    fm1 : float
        calculation constants
    fm2 : float
        calculation constants
    fm3 : float
        calculation constants
    fm4 : float
        calculation constants
    fm5 : float
        calculation constants
    fm6 : float
        calculation constants
    alpha : numpy array
        calculation constants
    beta : numpy array
        calculation constants

    Returns
    -------
    mval : numpy array
        output magnetic grid
    """

    h = z1-z0
    hsq = h**2

    for ii in range(numx):
        alpha[0] = x1-xobs[ii]
        alpha[1] = x2-xobs[ii]
        for jj in range(numy):
            beta[0] = y1-yobs[jj]
            beta[1] = y2-yobs[jj]
            t = 0.

            for i in range(2):
                alphasq = alpha[i]**2
                for j in range(2):
                    sign = 1.
                    if i != j:
                        sign = -1.
                    r0sq = alphasq+beta[j]**2+hsq
                    r0 = np.sqrt(r0sq)
                    r0h = r0*h
                    alphabeta = alpha[i]*beta[j]
                    arg1 = (r0-alpha[i])/(r0+alpha[i])
                    arg2 = (r0-beta[j])/(r0+beta[j])
                    arg3 = alphasq+r0h+hsq
                    arg4 = r0sq+r0h-alphasq
                    tlog = (fm3*np.log(arg1)/2.+fm2*np.log(arg2)/2. -
                            fm1*np.log(r0+h))
                    tatan = (-fm4*np.arctan2(alphabeta, arg3) -
                             fm5*np.arctan2(alphabeta, arg4) +
                             fm6*np.arctan2(alphabeta, r0h))

                    t = t+sign*(tlog+tatan)
            mval[ii, jj] = t

    return mval


def dircos(inc, dec, azim=90):
```

```
    """
    Calculate direction cosines from inclination and declination

    Parameters
    ----------
    inc : float
        inclination in degrees positive below horizontal.
    dec : float
        declination in degrees positive east of true north.
    azim : float
        azimuth of x axis in degrees positive east of north.

    Returns
    -------
    a, b, c : direction cosines
    """

    Inc = np.deg2rad(inc)
    Dec = np.deg2rad(dec-azim)

    a = np.cos(Inc)*np.cos(Dec)
    b = np.cos(Inc)*np.sin(Dec)
    g = np.sin(Inc)

    return a, b, g


def polyfit2d(x, y, f, deg):
    """
    This fits a polynomial surface through a set of data.

    Parameters
    ----------
    x : numpy array
        x coordinates
    y : numpy array
        y coordinates
    f : numpy array
        z values to fit surface through
    deg : list
        list of maximum degrees

    Returns
    -------
    c : numpy array
        output surface

    """
    x = np.asarray(x)
    y = np.asarray(y)
    f = np.asarray(f)
    deg = np.asarray(deg)
    vander = polynomial.polyvander2d(x, y, deg)
    vander = vander.reshape((-1, vander.shape[-1]))
    f = f.reshape((vander.shape[0],))
    c = np.linalg.lstsq(vander, f, rcond=None)[0]

    return c.reshape(deg+1)


def plot2(pdat, title='', clabel='', minstd=2, maxstd=2, extent=None,
          cmap=None, notstd=False, show=True):
    """
    Plotting routine for convenience. Plots images

    Parameters
    ----------
    pdat : numpy array
        grid to plot
    title : str
        title
    clabel: str
        colourbar label
```

```
    minstd : float
        minimum standard deviation or minimum value
    maxstd : float
        maximum standard deviation or maximum value
    extent : list
        coordinates of image extents
    cmap : matplotlib colour map
        colours to use in image
    notstd : bool
        flag for regular limits
    show : bool
        flag to show image immediately

    Returns
    -------
    None
    """
    if cmap is None:
        cmap = plt.cm.jet

    if notstd:
        vmin = minstd
        vmax = maxstd
    else:
        vmin = np.median(pdat)-np.std(pdat)*minstd
        vmax = np.median(pdat)+np.std(pdat)*maxstd

    if show is True:
        plt.figure(figsize=(9, 6))

    ax = plt.gca()
    plt.ylabel('Distance (m)')
    plt.xlabel('Distance (m)')

    extent = [0, extent[1]-extent[0], 0, extent[3]-extent[2]]

    plt.xticks(extent[0:2], ha='center')
    plt.yticks(extent[2:4], rotation='vertical', va='center')

#    plt.xticks(ha='center')
#    plt.yticks(rotation='vertical', va='center')

    plt.title(title, loc='left')
    im = plt.imshow(pdat, extent=extent, cmap=cmap, vmin=vmin, vmax=vmax)

    divider = make_axes_locatable(ax)
    cax = divider.append_axes("right", size="5%", pad=0.05)

#    plt.title(clabel, size='medium')
    cbar = plt.colorbar(im, cax=cax)
    cbar.set_label(clabel)
    if show is True:
        plt.tight_layout()
        plt.show()
    return ax


def As_calcs(dx, dy, dz, dxy=1):
    """
    Calculates Analytic signals

    Parameters
    ----------
    dx : numpy array
        dx grid
    dy : numpy array
        dy grid
    dz : numpy array
        dz grid
    dxy : float
        cube dimension

    Returns
```

```
    -------
    As : numpy array
        first order analytic signal
    As2 : numpy array
        second order analytic signal
    """

    deltax = dxy
    deltay = dxy

    As = np.sqrt(dx**2+dy**2+dz**2)

# Order 2 gradients
#    dyz, dxz = np.gradient(dz, deltax)
#    dyy,   = np.gradient(dy, deltay)
#    dxy, dxx = np.gradient(dx, deltax)


# Order 4 gradients
    dyz, dxz = gradient04(dz, deltax)
    dyy, _ = gradient04(dy, deltay)
    dxy, dxx = gradient04(dx, deltax)

    dzz = -(dxx+dyy)
    asxt = (dx*dxx + dy*dxy + dz*dxz)
    asyt = (dx*dxy + dy*dyy + dz*dyz)
    aszt = (dx*dxz + dy*dyz + dz*dzz)
    As2 = np.sqrt(asxt**2+asyt**2+aszt**2)/As

    return As, As2


def pseudo_tensor(magval, inc=-62, dec=-16, dxy=10, cx=None, cy=None, cz=None):
    """
    Pseudo tensor calculation

    Parameters
    ----------
    magval : numpy array
        magnetic TMI values
    inc : float
        inclination
    dec : float
        inclination
    dxy : float
        cube dimension

    Returns
    -------
    ptensor : dictionary
        pseudo tensor values

    """
    if cx is None:
        cx, cy, cz = dircos(inc, dec)

    pwidth = max(magval.shape)
    magval2 = np.pad(magval, pwidth, 'linear_ramp')

    fft = np.fft.fft2(magval2)
    kx = np.fft.fftfreq(fft.shape[1], d=dxy)
    ky = np.fft.fftfreq(fft.shape[0], d=dxy)

    ptensor = {}

    kx, ky = np.meshgrid(kx, ky)
    k = np.sqrt(kx**2+ky**2)
    dterm = (1j*(cx*kx+cy*ky)+cz*k)

    dterm[dterm == 0] = -np.finfo(float).eps
    k[k == 0] = np.finfo(float).eps
    kx[kx == 0] = np.finfo(float).eps
    ky[ky == 0] = np.finfo(float).eps
```

```
    out1 = fft*kx*1j/dterm
    out1[np.isnan(out1)] = 0.
    out2 = np.fft.ifft2(out1)
    ptensor['x'] = out2[pwidth:-pwidth, pwidth:-pwidth].real

    out1 = fft*ky*1j/dterm
    out1[np.isnan(out1)] = 0.
    out2 = np.fft.ifft2(out1)
    ptensor['y'] = out2[pwidth:-pwidth, pwidth:-pwidth].real

    out1 = fft*k/dterm
    out1[np.isnan(out1)] = 0.
    out2 = np.fft.ifft2(out1)
    ptensor['z'] = out2[pwidth:-pwidth, pwidth:-pwidth].real
    FBz = out1

    out1 = -2*np.pi*kx**2/k*FBz
    out1[np.isnan(out1)] = 0.
    out2 = np.fft.ifft2(out1)
    ptensor['xx'] = out2[pwidth:-pwidth, pwidth:-pwidth].real

    out1 = -2*np.pi*ky**2/k*FBz
    out1[np.isnan(out1)] = 0.
    out2 = np.fft.ifft2(out1)
    ptensor['yy'] = out2[pwidth:-pwidth, pwidth:-pwidth].real

    out1 = 2*np.pi*k*FBz
    out1[np.isnan(out1)] = 0.
    out2 = np.fft.ifft2(out1)
    ptensor['zz'] = out2[pwidth:-pwidth, pwidth:-pwidth].real

    out1 = -2*np.pi*kx*ky/k*FBz
    out1[np.isnan(out1)] = 0.
    out2 = np.fft.ifft2(out1)
    ptensor['xy'] = out2[pwidth:-pwidth, pwidth:-pwidth].real

    out1 = 2*np.pi*1j*kx*FBz
    out1[np.isnan(out1)] = 0.
    out2 = np.fft.ifft2(out1)
    ptensor['xz'] = out2[pwidth:-pwidth, pwidth:-pwidth].real

    out1 = 2*np.pi*1j*ky*FBz
    out1[np.isnan(out1)] = 0.
    out2 = np.fft.ifft2(out1)
    ptensor['yz'] = out2[pwidth:-pwidth, pwidth:-pwidth].real

    return ptensor


def pseudo_limits():
    """
    Shows limits of pseudo tensor calculations, in that they cannot account
    for remanence.

    Parameters
    ----------
    None

    Returns
    -------
    None
    """

    inc = 60
    dec = -30

    Tz = []
    oTz = []
    magval = []

    tcube = None
    for mstr in [0, 0.323]:
```

```
            del tcube
            tcube = TensorCube()
            tcube.inc = inc
            tcube.dec = dec
            tcube.mstrength = mstr
            tcube.minc = -40
            tcube.mdec = 20
            tcube.susc = 0.01
            tcube.hintn = 28000.
            tcube.azim = 90
#            tcube.dxy = 5

            tcube.calc_all()

            prof = int(tcube.magval.shape[0]//2)

            dxy = tcube.dxy

            magval.append(tcube.magval)
            Tz.append(tcube.bz)

        # calculate artificial tensor

            ptensor = pseudo_tensor(tcube.magval, inc, dec, dxy,
                                    tcube.cx, tcube.cy, tcube.cz)

            oTz.append(ptensor['z'])

            print(tcube.cx, tcube.cy, tcube.cz)
#            magval2 = tcube.cx*tcube.bx+tcube.cy*tcube.by+tcube.cz*tcube.bz

        xcoords = np.arange(0, tcube.rc, tcube.dxy)
        x1, x2 = tcube.u
        y1, y2 = 0, -20  # tcube.v

        plt.figure(figsize=(8, 4))
        plt.subplot(121)
        plt.title('(a) No remanence', loc='left')
        plt.ylim(-50, 200)
        ply = plt.Line2D([x1, x1, x2, x2, x1], [y2, y1, y1, y2, y2], color='k')
        plt.gca().add_line(ply)
        plt.plot(xcoords, magval[0][prof], '-.', label='$B_{tmi}$')
        plt.plot(xcoords, Tz[0][prof], '.', label='$B_{z}$')
        plt.plot(xcoords, oTz[0][prof], label='Derived $B_{z}$')
        plt.ylabel('(nT)')
        plt.xlabel('Distance (m)')
        plt.legend(loc='upper left')

        plt.subplot(122)
        plt.ylim(-50, 200.)
        plt.title('(b) Remanence', loc='left')
        ply = plt.Line2D([x1, x1, x2, x2, x1], [y2, y1, y1, y2, y2], color='k')
        plt.gca().add_line(ply)
        plt.plot(xcoords, magval[1][prof], '-.', label='$B_{tmi}$')
        plt.plot(xcoords, Tz[1][prof], '.', label='$B_{z}$')
        plt.plot(xcoords, oTz[1][prof], label='Derived $B_{z}$')
        plt.ylabel('(nT)')
        plt.xlabel('Distance (m)')
        plt.legend(loc='upper left')

        plt.tight_layout()
        plt.show()


    def tests():
        """
        Test program for calculation of tensor cube data.

        Parameters
        ----------
        None

        Returns
```

```
-------
None
"""

tcube = TensorCube()
tcube.calc_all()

tcube.plot_grav()
tcube.plot_mag()

pt = pseudo_tensor(tcube.magval)

plt.figure(figsize=(8, 8))

plt.subplot(4, 3, 1)
plt.imshow(pt['x'], vmin=-500, vmax=500)
plt.title('x')
plt.subplot(4, 3, 2)
plt.imshow(pt['y'], vmin=-500, vmax=500)
plt.title('y')
plt.subplot(4, 3, 3)
plt.imshow(pt['z'], vmin=-500, vmax=500)
plt.title('z')
plt.subplot(4, 3, 4)
plt.imshow(pt['xx'], vmin=-10, vmax=10)
plt.title('xx')
plt.subplot(4, 3, 8)
plt.imshow(pt['yy'], vmin=-10, vmax=10)
plt.title('yy')
plt.subplot(4, 3, 12)
plt.imshow(pt['zz'], vmin=-10, vmax=10)
plt.title('zz')
plt.subplot(4, 3, 5)
plt.imshow(pt['xy'], vmin=-10, vmax=10)
plt.title('xy')
plt.subplot(4, 3, 9)
plt.imshow(pt['yz'], vmin=-10, vmax=10)
plt.title('yz')
plt.subplot(4, 3, 6)
plt.imshow(pt['xz'], vmin=-10, vmax=10)
plt.title('xz')

plt.subplot(4, 3, 10)
a, b, g = dircos(-62., -16.)

nmagval = a*pt['x']+b*pt['y']+g*pt['z']
plt.imshow(nmagval, vmin=-500, vmax=500)

plt.tight_layout()
plt.show()

dxy = tcube.dxy

# Now we begin the tensor stuff

Txx = np.ma.masked_invalid(tcube.bxx)
Txy = np.ma.masked_invalid(tcube.bxy)
Txz = np.ma.masked_invalid(tcube.bxz)
Tyy = np.ma.masked_invalid(tcube.byy)
Tyz = np.ma.masked_invalid(tcube.byz)
Tzz = np.ma.masked_invalid(tcube.bzz)

cx = tcube.cx
cy = tcube.cy
cz = tcube.cz

Htmi_x = cx*Txx+cy*Txy+cz*Txz
Htmi_y = cx*Txy+cy*Tyy+cz*Tyz
Htmi_z = cx*Txz+cy*Tyz+cz*Tzz

Asp = np.sqrt(Htmi_x**2 + Htmi_y**2 + Htmi_z**2)

Txxy, Txxx = gradientO4(Txx, dxy)
```

```
Tyyy, Tyyx = gradientO4(Tyy, dxy)
Tzzy, Tzzx = gradientO4(Tzz, dxy)
Txyy, Txyx = gradientO4(Txy, dxy)
Txzy, Txzx = gradientO4(Txz, dxy)
Tyzy, Tyzx = gradientO4(Tyz, dxy)

Txxz = Txzx
Tyyz = Tyzy
Tzzz = -(Txxz+Tyyz)
Txyz = Tyzx
Txzz = Tzzx
Tyzz = Tzzy

As_x = (Htmi_x*(cx*Txxx + cy*Txyx + cz*Txzx) +
        Htmi_y*(cx*Txyx + cy*Tyyx + cz*Tyzx) +
        Htmi_z*(cx*Txzx + cy*Tyzx + cz*Tzzx))

As_y = (Htmi_x*(cx*Txxy + cy*Txyy + cz*Txzy) +
        Htmi_y*(cx*Txyy + cy*Tyyy + cz*Tyzy) +
        Htmi_z*(cx*Txzy + cy*Tyzy + cz*Tzzy))

As_z = (Htmi_x*(cx*Txxz + cy*Txyz + cz*Txzz) +
        Htmi_y*(cx*Txyz + cy*Tyyz + cz*Tyzz) +
        Htmi_z*(cx*Txzz + cy*Tyzz + cz*Tzzz))

As2p = np.sqrt(As_x**2 + As_y**2 + As_z**2)/Asp

N = 0.0
distancep = (N+1)*Asp/As2p

xcoords = np.arange(0, tcube.rc, tcube.dxy)
x1, x2 = tcube.u
y1, y2 = tcube.v
z1, z2 = tcube.w
z2 = -300

extent = (0, tcube.rc, 0, tcube.rc)

plt.figure(figsize=(6, 6))
plt.subplot(2, 2, 1)
ax = plt.gca()
im = plt.imshow(Asp, extent=extent)
divider = make_axes_locatable(ax)
ply = plt.Line2D([x1, x1, x2, x2, x1], [y2, y1, y1, y2, y2], color='k')
plt.gca().add_line(ply)
plt.xticks([0, tcube.rc])
plt.yticks([0, tcube.rc], rotation='vertical')
plt.ylabel('Distance (m)')
plt.xlabel('Distance (m)')
plt.title('(a)', loc='left')
cax = divider.append_axes("right", size="5%", pad=0.05)
plt.title('$As_{1}$ (nT/m)', size='medium')
plt.colorbar(im, cax=cax)

plt.subplot(2, 2, 2)
ax = plt.gca()
im = plt.imshow(As2p, extent=extent)
ply = plt.Line2D([x1, x1, x2, x2, x1], [y2, y1, y1, y2, y2], color='k')
plt.gca().add_line(ply)
plt.xticks([0, tcube.rc])
plt.yticks([0, tcube.rc], rotation='vertical')
plt.ylabel('Distance (m)')
plt.xlabel('Distance (m)')
plt.title('(b)', loc='left')
divider = make_axes_locatable(ax)
cax = divider.append_axes("right", size="5%", pad=0.05)
plt.title(r'$As_{2} (\mathrm{nT/m}^2)$', size='medium')
plt.colorbar(im, cax=cax)

plt.subplot(2, 2, 3)
ax = plt.gca()
im = plt.imshow(-distancep, vmax=0, extent=extent)
ply = plt.Line2D([x1, x1, x2, x2, x1], [y2, y1, y1, y2, y2], color='k')
```

```
    plt.gca().add_line(ply)
    ply = plt.Line2D([0, tcube.rc], [tcube.rc/2, tcube.rc/2], color='k',
                     ls='dashed')
    plt.gca().add_line(ply)
    plt.xticks([0, tcube.rc])
    plt.yticks([0, tcube.rc], rotation='vertical')
    plt.ylabel('Distance (m)')
    plt.xlabel('Distance (m)')
    plt.title('(c)', loc='left')
    divider = make_axes_locatable(ax)
    cax = divider.append_axes("right", size="5%", pad=0.05)
    plt.title('Depth (m)', size='medium')
    plt.colorbar(im, cax=cax, ticks=[0, -50, -100, -150, -200])

    plt.subplot(2, 2, 4)
    plt.plot(xcoords, -distancep[20])
    ply = plt.Line2D([x1, x1, x2, x2], [z2, z1, z1, z2], color='k')
    plt.gca().add_line(ply)
    plt.xlabel('Distance (m)')
    plt.ylabel('Depth (m)', labelpad=-10)
    plt.yticks([0, -200])
    plt.title('(d)', loc='left')

    plt.tight_layout()
    plt.show()


def remanence():
    """
    Test program for calculation of remanence.

    Parameters
    ----------
    None

    Returns
    -------
    None
    """
    cmap = plt.cm.jet

    inc = 60
    dec = -30
    minc = 50
    mdec = -20
    mstrength = 0.323
    lmstrength = 0.
    susc = 0.01
    hintn = 28000.

    print('inclination: ', inc)
    fa, fb, fc = dircos(inc, dec, 90)
    ma, mb, mc = dircos(minc, mdec, 90)

# This is A/m
    mr = lmstrength*np.array([ma, mb, mc])
    mi = susc*hintn*np.array([fa, fb, fc])/(400*np.pi)

    m3 = mr+mi
    m = np.sqrt(m3 @ m3)
    m3 /= m
    cx1, cy1, cz1 = m3
    m1 = m

    mr = mstrength*np.array([ma, mb, mc])
    mi = susc*hintn*np.array([fa, fb, fc])/(400*np.pi)
    m3 = mr+mi
    m = np.sqrt(m3 @ m3)
    m3 /= m
    cx2, cy2, cz2 = m3
    m2 = m

##############################
```

```
    tcube1 = TensorCube()
    tcube1.inc = inc
    tcube1.dec = dec
    tcube1.mstrength = lmstrength
    tcube1.minc = minc
    tcube1.mdec = mdec
    tcube1.susc = susc
    tcube1.hintn = hintn
    tcube1.u = [90, 100]
    tcube1.v = [0, 400]
#     tcube1.v = [100, 300]
    tcube1.w = [-20, -3000]
    tcube1.rc = 400

    tcube1.init_grids()
    tcube1.calc_all()
    N = 1

    tcube2 = TensorCube()
    tcube2.inc = inc
    tcube2.dec = dec
    tcube2.mstrength = mstrength
    tcube2.minc = minc
    tcube2.mdec = mdec
    tcube2.susc = susc
    tcube2.hintn = hintn
    tcube2.u = [290, 300]  # [250, 350]
    tcube2.v = tcube1.v  # [150, 250]
    tcube2.w = tcube1.w
    tcube2.rc = 400

    tcube2.init_grids()
    tcube2.calc_all()

##################################

    print('rmi:', lmstrength*400*np.pi)
    print('kH:', tcube1.susc*tcube1.hintn)
    print('mi:', tcube1.susc*tcube1.hintn/(400*np.pi),
          'mr:', lmstrength,
          'mt:', m)

    ########################
    Tx = tcube1.bx
    Ty = tcube1.by
    Tz = tcube1.bz
    Txx = tcube1.bxx
    Txy = tcube1.bxy
    Txz = tcube1.bxz
    Tyy = tcube1.byy
    Tyz = tcube1.byz
    Tzz = tcube1.bzz
    magval = tcube1.magval

    dxy = tcube1.dxy

    cx, cy, cz = cx1, cy1, cz1
    Hx = (Txx*(Tx+cx*hintn)+Txy*(Ty+cy*hintn)+Txz*(Tz+cz*hintn))/(magval+hintn)
    Hy = (Txy*(Tx+cx*hintn)+Tyy*(Ty+cy*hintn)+Tyz*(Tz+cz*hintn))/(magval+hintn)
    Hz = (Txz*(Tx+cx*hintn)+Tyz*(Ty+cy*hintn)+Tzz*(Tz+cz*hintn))/(magval+hintn)

    Tx = tcube2.bx
    Ty = tcube2.by
    Tz = tcube2.bz
    Txx = tcube2.bxx
    Txy = tcube2.bxy
    Txz = tcube2.bxz
    Tyy = tcube2.byy
    Tyz = tcube2.byz
    Tzz = tcube2.bzz
    magval = tcube2.magval

    cx, cy, cz = cx2, cy2, cz2
```

```
    Hx += (Txx*(Tx+cx*hintn)+Txy*(Ty+cy*hintn)+Txz*(Tz+cz*hintn))/(magval+hintn)
    Hy += (Txy*(Tx+cx*hintn)+Tyy*(Ty+cy*hintn)+Tyz*(Tz+cz*hintn))/(magval+hintn)
    Hz += (Txz*(Tx+cx*hintn)+Tyz*(Ty+cy*hintn)+Tzz*(Tz+cz*hintn))/(magval+hintn)

    Tx = tcube1.bx+tcube2.bx
    Ty = tcube1.by+tcube2.by
    Tz = tcube1.bz+tcube2.bz
    Txx = tcube1.bxx+tcube2.bxx
    Txy = tcube1.bxy+tcube2.bxy
    Txz = tcube1.bxz+tcube2.bxz
    Tyy = tcube1.byy+tcube2.byy
    Tyz = tcube1.byz+tcube2.byz
    Tzz = tcube1.bzz+tcube2.bzz
    magval = tcube1.magval+tcube2.magval

    plt.imshow(magval)
    plt.show()

    cx2a, cy2a, cz2a = cx2, cy2, cz2
    cx2, cy2, cz2 = cosines_from_tensor3(Hx, Hy, Hz, magval, Txx, Tyy,
                                         Txy, Txz, Tyz, hintn, Tx, Ty, Tz)

    As, As2 = As_calcs(Hx, Hy, Hz, dxy)

    Hilbx = ss.hilbert(magval, axis=1).imag
    Hilby = ss.hilbert(magval, axis=0).imag
    As0 = np.sqrt(magval**2 + Hilbx**2 + Hilby**2)

    Hilbx = ss.hilbert(Tz, axis=1).imag
    Hilby = ss.hilbert(Tz, axis=0).imag
    Asz0 = np.sqrt(Tz**2 + Hilbx**2 + Hilby**2)

    Asx, Asx2 = As_calcs(Txx, Txy, Txz, dxy)
    Asy, Asy2 = As_calcs(Txy, Tyy, Tyz, dxy)
    Asz, Asz2 = As_calcs(Txz, Tyz, Tzz, dxy)

########################
    r1 = -N*As0/As
    r2 = -N*Asz0/Asz

    print('####################')
    print('depth:', tcube1.w[0], r1[19, 9], r2[19, 9])

########################
    width = tcube1.u[1]-tcube1.u[0]
    depth = tcube1.w[0]
    print('width:', width)


    mt = As0**2/(As*2*width*(1-cy2**2)*100)
    mt2 = Asz*depth**2/(width*200**np.sqrt(1-cy2**2))
    mt2a = Asz0**2/(Asz*width*200*np.sqrt(1-cy2**2))
    mt2b = Asz0**2/(As*width*200)

    mt2 = mt2b

    print('cx1:', cx1, cx2[19, 9])
    print('cy1:', cy1, cy2[19, 9])
    print('cz1:', cz1, cz2[19, 9])

    print('cx2:', cx2a, cx2[19, 29])
    print('cy2:', cy2a, cy2[19, 29])
    print('cz2:', cz2a, cz2[19, 29])

    susc = mt2*400*np.pi/tcube1.hintn

    print('inc:', inc, 'minc:', minc)
    print('dec:', dec, 'mdec:', mdec)
    print('mt (real):', m1, 'mt:', mt[19, 9], 'mt2a', mt2a[19, 9],
          'mt2', mt2[19, 9])
    print('2mt (real):', m2, 'mt:', mt[19, 29], 'mt2a', mt2a[19, 29],
          'mt2', mt2[19, 29])
    print('susc:', tcube1.susc, susc[19, 9])
```

```
    print('2susc:', tcube1.susc, susc[19, 29])

    mt[:, :] = mt2

    plt.figure(figsize=(9, 6))
    vmin = np.median(cx2)-np.std(cx2)*2
    vmax = np.median(cx2)+np.std(cx2)*2

    ax = plt.gca()
    plt.title(r'$\alpha_{t}$')
    plt.imshow(cx2, extent=(0, 400, 0, 400), cmap=cmap, vmin=vmin, vmax=vmax)
    plt.colorbar()
    plt.ylabel('Northings (m)')
    plt.xlabel('Eastings (m)')

    x1, x2 = tcube1.u
    y1, y2 = tcube1.v
    ply = plt.Line2D([x1, x1, x2, x2], [y2, y1, y1, y2], color='k')
    ax.add_line(ply)

    x1, x2 = tcube2.u
    y1, y2 = tcube2.v
    ply = plt.Line2D([x1, x1, x2, x2], [y2, y1, y1, y2], color='k')
    ax.add_line(ply)
    plt.show()

    plt.figure(figsize=(9, 6))
    vmin = np.median(cy2)-np.std(cy2)*2
    vmax = np.median(cy2)+np.std(cy2)*2

    ax = plt.gca()
    plt.title(r'$\beta_{t}$')
    plt.imshow(cy2, extent=(0, 400, 0, 400), cmap=cmap, vmin=vmin, vmax=vmax)
    plt.colorbar()
    plt.ylabel('Northings (m)')
    plt.xlabel('Eastings (m)')

    x1, x2 = tcube1.u
    y1, y2 = tcube1.v
    ply = plt.Line2D([x1, x1, x2, x2], [y2, y1, y1, y2], color='k')
    ax.add_line(ply)

    x1, x2 = tcube2.u
    y1, y2 = tcube2.v
    ply = plt.Line2D([x1, x1, x2, x2], [y2, y1, y1, y2], color='k')
    ax.add_line(ply)
    plt.show()

    plt.figure(figsize=(9, 6))
    vmin = np.median(cz2)-np.std(cz2)*2
    vmax = np.median(cz2)+np.std(cz2)*2

    ax = plt.gca()
    plt.title(r'$\gamma_{t}$')
    plt.imshow(cz2, extent=(0, 400, 0, 400), cmap=cmap, vmin=vmin, vmax=vmax)
    plt.colorbar()
    plt.ylabel('Northings (m)')
    plt.xlabel('Eastings (m)')

    x1, x2 = tcube1.u
    y1, y2 = tcube1.v
    ply = plt.Line2D([x1, x1, x2, x2], [y2, y1, y1, y2], color='k')
    ax.add_line(ply)

    x1, x2 = tcube2.u
    y1, y2 = tcube2.v
    ply = plt.Line2D([x1, x1, x2, x2], [y2, y1, y1, y2], color='k')
    ax.add_line(ply)
    plt.show()

################################################################################

    rc = tcube1.rc
```

```
extent = (0, rc, 0, rc)
x1, x2 = tcube1.u
y1, y2 = tcube1.v

plt.figure(figsize=(8, 8))

ax = plt.subplot(4, 3, 1)
plt.ylabel('Distance (m)')
plt.xlabel('Distance (m)')
im = plt.imshow(Tx, extent=extent)

plt.xticks([0, rc])
plt.yticks([0, rc], rotation='vertical')
plt.title('(a)', loc='left')
divider = make_axes_locatable(ax)
cax = divider.append_axes("right", size="5%", pad=0.05)
plt.title('$B_{x}$ (nT)', size='medium')
plt.colorbar(im, cax=cax)

ax = plt.subplot(4, 3, 2)
plt.ylabel('Distance (m)')
plt.xlabel('Distance (m)')
im = plt.imshow(Ty, extent=extent)

plt.xticks([0, rc])
plt.yticks([0, rc], rotation='vertical')
plt.title('(b)', loc='left')
divider = make_axes_locatable(ax)
cax = divider.append_axes("right", size="5%", pad=0.05)
plt.title('$B_{y}$ (nT)', size='medium')
plt.colorbar(im, cax=cax)

ax = plt.subplot(4, 3, 3)
plt.ylabel('Distance (m)')
plt.xlabel('Distance (m)')
im = plt.imshow(Tz, extent=extent)

plt.xticks([0, rc])
plt.yticks([0, rc], rotation='vertical')
plt.title('(c)', loc='left')
divider = make_axes_locatable(ax)
cax = divider.append_axes("right", size="5%", pad=0.05)
plt.title('$B_{z}$ (nT)', size='medium')
plt.colorbar(im, cax=cax)

ax = plt.subplot(4, 3, 4)
plt.ylabel('Distance (m)')
plt.xlabel('Distance (m)')
im = plt.imshow(Txx, extent=extent)

plt.xticks([0, rc])
plt.yticks([0, rc], rotation='vertical')
plt.title('(d)', loc='left')
divider = make_axes_locatable(ax)
cax = divider.append_axes("right", size="5%", pad=0.05)
plt.title('$B {xx}$ (nT/m)', size='medium')
plt.colorbar(im, cax=cax)

ax = plt.subplot(4, 3, 8)
plt.ylabel('Distance (m)')
plt.xlabel('Distance (m)')
im = plt.imshow(Tyy, extent=extent)

plt.xticks([0, rc])
plt.yticks([0, rc], rotation='vertical')
plt.title('(g)', loc='left')
divider = make_axes_locatable(ax)
cax = divider.append_axes("right", size="5%", pad=0.05)
plt.title('$B_{yy}$ (nT/m)', size='medium')
plt.colorbar(im, cax=cax)

ax = plt.subplot(4, 3, 12)
plt.ylabel('Distance (m)')
```

```
    plt.xlabel('Distance (m)')
    im = plt.imshow(Tzz, extent=extent)

    plt.xticks([0, rc])
    plt.yticks([0, rc], rotation='vertical')
    plt.title('(j)', loc='left')
    divider = make_axes_locatable(ax)
    cax = divider.append_axes("right", size="5%", pad=0.05)
    plt.title('$B_{zz}$ (nT/m)', size='medium')
    plt.colorbar(im, cax=cax)

    ax = plt.subplot(4, 3, 5)
    plt.ylabel('Distance (m)')
    plt.xlabel('Distance (m)')
    im = plt.imshow(Txy, extent=extent)

    plt.xticks([0, rc])
    plt.yticks([0, rc], rotation='vertical')
    plt.title('(e)', loc='left')
    divider = make_axes_locatable(ax)
    cax = divider.append_axes("right", size="5%", pad=0.05)
    plt.title('$B_{xy}$ (nT/m)', size='medium')
    plt.colorbar(im, cax=cax)

    ax = plt.subplot(4, 3, 9)
    plt.ylabel('Distance (m)')
    plt.xlabel('Distance (m)')
    im = plt.imshow(Tyz, extent=extent)

    plt.xticks([0, rc])
    plt.yticks([0, rc], rotation='vertical')
    plt.title('(h)', loc='left')
    divider = make_axes_locatable(ax)
    cax = divider.append_axes("right", size="5%", pad=0.05)
    plt.title('$B_{yz}$ (nT/m)', size='medium')
    plt.colorbar(im, cax=cax)

    ax = plt.subplot(4, 3, 6)
    plt.ylabel('Distance (m)')
    plt.xlabel('Distance (m)')
    im = plt.imshow(Txz, extent=extent)

    plt.xticks([0, rc])
    plt.yticks([0, rc], rotation='vertical')
    plt.title('(f)', loc='left')
    divider = make_axes_locatable(ax)
    cax = divider.append_axes("right", size="5%", pad=0.05)
    plt.title('$B_{xz}$ (nT/m)', size='medium')
    plt.colorbar(im, cax=cax)

    ax = plt.subplot(4, 3, 10)
    plt.ylabel('Distance (m)')
    plt.xlabel('Distance (m)')
    dtmp = magval
    im = plt.imshow(dtmp, extent=extent)

    plt.xticks([0, rc])
    plt.yticks([0, rc], rotation='vertical')
    plt.title('(i)', loc='left')
    divider = make_axes_locatable(ax)
    cax = divider.append axes("right", size="5%", pad=0.05)
    plt.title('$B_{tmi}$ (nT)', size='medium')
    plt.colorbar(im, cax=cax)

    plt.tight_layout()
    plt.show()

##############################################################################
    azim = 90
    mr, minc, mdec = rem_from_cosines(cx2, cy2, cz2, mt, tcube1.susc, hintn,
                                      inc, dec, azim)

    print('inc:', tcube2.inc)
```

```
    print('dec:', tcube2.dec)
    print('minc:', tcube2.minc, minc[19, 29])
    print('mdec:', tcube2.mdec, mdec[19, 29])
    print('mr:', tcube2.mstrength, mr[19, 29])

    title = ['Total Magnetisation', 'Remanent Magnetisation']
    clabel = 'A/m'
    cmap = plt.cm.jet
    minstd = 1
    maxstd = 1
    extent = (0, 400, 0, 400)

    mask = np.ma.make_mask_none(mt.shape)+True
    mask[1:-1, 1:-1] = False

    mt = np.ma.array(mt, mask=mask)
    mr = np.ma.array(mr, mask=mask)

    for i, pdat in enumerate([mt, mr]):
        vmin = np.ma.median(pdat)-np.ma.std(pdat)*minstd
        vmax = np.ma.median(pdat)+2*np.ma.std(pdat)*maxstd

        plt.figure(figsize=(9, 6))
        plt.title(title[i])
        plt.imshow(pdat, extent=extent, cmap=cmap, vmin=vmin, vmax=vmax)
        plt.colorbar().set_label(clabel)
        plt.ylabel('Northings (m)')
        plt.xlabel('Eastings (m)')

        x1, x2 = tcube1.u
        y1, y2 = tcube1.v
        ply = plt.Line2D([x1, x1, x2, x2], [y2, y1, y1, y2], color='k')
        plt.gca().add_line(ply)

        x1, x2 = tcube2.u
        y1, y2 = tcube2.v
        ply = plt.Line2D([x1, x1, x2, x2], [y2, y1, y1, y2], color='k')
        plt.gca().add_line(ply)

        plt.tight_layout()
        plt.show()


def interp(filename):
    """
    Implementation of the interpretatin routine in PhD

    Parameters
    ----------
    filename : string
        input PyGMI model filename. This contains all raster data in it.

    Returns
    -------
    None
    """

    N = 1  # dyke
    fht = 50  # flying height
    w = 100  # this is the dike width
    numclasses = 10


# Import model file
    lmod = importmod3d(filename)

    maxdepth = lmod.numz*lmod.d_z

    maxdepth = 500

    inc = lmod.lith_list['Generic 1'].finc
    dec = lmod.lith_list['Generic 1'].fdec
    hintn = lmod.lith_list['Generic 1'].hintn
```

```
    magval = lmod.griddata['Magnetic Dataset'].data.data
    tmp = lmod.griddata['Magnetic Dataset']
    magextent = (tmp.tlx, tmp.tlx+tmp.cols*tmp.xdim, tmp.tly-tmp.rows*tmp.ydim,
                 tmp.tly)
    dtmval = lmod.griddata['DTM Dataset'].data.data
    tmp = lmod.griddata['DTM Dataset']
    dxy = lmod.griddata['Magnetic Dataset'].xdim

# subtract surface from dataset

    rows, cols = magval.shape
    y, x = np.mgrid[:rows, :cols]
    c = polyfit2d(x, y, magval, [2, 2])
    f = polynomial.polyval2d(x, y, c)

    magval -= f

# calculate artificial tensor

    ptensor = pseudo_tensor(magval, inc, dec, dxy)

    Tx = ptensor['x']
    Ty = ptensor['y']
    Tz = ptensor['z']
    Txx = ptensor['xx']
    Tyy = ptensor['yy']
    Tzz = ptensor['zz']
    Txy = ptensor['xy']
    Txz = ptensor['xz']
    Tyz = ptensor['yz']

    plt.figure(figsize=(9, 6))

    ax = plt.gca()
    plt.ylabel('Northings (m)')
    plt.xlabel('Eastings (m)')

    plt.title('Total Magnetic Intensity')
    plt.imshow(magval+f, extent=magextent, vmin=-150,
               vmax=400)
    plt.colorbar().set_label('nT')

    ply = plt.Line2D([magextent[0]+15520, magextent[0]+15520],
                     [magextent[2], magextent[3]], color='k', ls='dashed')
    ax.add_line(ply)

    plt.tight_layout()
    plt.show()

    plt.figure(figsize=(9, 6))
    plt.title('Polynomial Surface')
    plt.imshow(f, extent=magextent)
    plt.colorbar().set_label('nT')

    plt.tight layout()
    plt.show()

    plt.figure(figsize=(9, 6))
    plt.title('Total Magnetic Intensity - Polynomial Surface')
    plt.imshow(magval, extent=magextent, vmin=-100, vmax=150)
    plt.colorbar().set_label('nT')

    plt.tight_layout()
    plt.show()

    plt.figure(figsize=(8, 8))
    ax = plt.subplot(4, 3, 10)

    plot2(magval, '(i) $B_{tmi}$', '(nT)', minstd=2.0, maxstd=2.5,
          extent=magextent, show=False)
    ply = plt.Line2D([15520, 15520], [0, 36705], color='k', ls='dashed')
    ax.add_line(ply)
```

```
plt.subplot(4, 3, 1)
plot2(Tx, '(a) $B_{x}$', '(nT)', minstd=1, maxstd=2.5,
      extent=magextent, show=False)
plt.subplot(4, 3, 2)
plot2(Ty, '(b) $B_{y}$', '(nT)', minstd=1, maxstd=2.5,
      extent=magextent, show=False)
plt.subplot(4, 3, 3)
plot2(Tz, '(c) $B_{z}$', '(nT)', minstd=1, maxstd=2.5,
      extent=magextent, show=False)
plt.subplot(4, 3, 4)
plot2(Txx, '(d) $B_{xx}$', '(nT/m)', minstd=1, maxstd=2.5,
      extent=magextent, show=False)
plt.subplot(4, 3, 8)
plot2(Tyy, '(g) $B_{yy}$', '(nT/m)', minstd=1, maxstd=2.5,
      extent=magextent, show=False)
plt.subplot(4, 3, 12)
plot2(Tzz, '(j) $B_{zz}$', '(nT/m)', minstd=1, maxstd=2.5,
      extent=magextent, show=False)
plt.subplot(4, 3, 5)
plot2(Txy, '(e) $B_{xy}$', '(nT/m)', minstd=1, maxstd=2.5,
      extent=magextent, show=False)
plt.subplot(4, 3, 6)
plot2(Txz, '(f) $B_{xz}$', '(nT/m)', minstd=1, maxstd=2.5,
      extent=magextent, show=False)
plt.subplot(4, 3, 9)
plot2(Tyz, '(h) $B_{yz}$', '(nT/m)', minstd=1, maxstd=2.5,
      extent=magextent, show=False)
plt.tight_layout()
plt.show()

cx, cy, cz = dircos(inc, dec)
Hx = (Txx*(Tx+cx*hintn)+Txy*(Ty+cy*hintn)+Txz*(Tz+cz*hintn))/(magval+hintn)
Hy = (Txy*(Tx+cx*hintn)+Tyy*(Ty+cy*hintn)+Tyz*(Tz+cz*hintn))/(magval+hintn)
Hz = (Txz*(Tx+cx*hintn)+Tyz*(Ty+cy*hintn)+Tzz*(Tz+cz*hintn))/(magval+hintn)

As, As2 = As_calcs(Hx, Hy, Hz, dxy)
Asz, Asz2 = As_calcs(Txz, Tyz, Tzz, dxy)
Asy, Asy2 = As_calcs(Txy, Tyy, Tyz, dxy)
Asx, Asx2 = As_calcs(Txx, Txy, Txz, dxy)

As0 = As0_calc(magval)
Asx0 = As0_calc(Tx)
Asy0 = As0_calc(Ty)
Asz0 = As0_calc(Tz)

r1 = -N*As0/As + fht
rx12 = -(N+1)*Asx/Asx2 + fht
ry12 = -(N+1)*Asy/Asy2 + fht
rz12 = -(N+1)*Asz/Asz2 + fht
rx01 = -N*Asx0/Asx + fht
ry01 = -N*Asy0/Asy + fht
rz01 = -N*Asz0/Asz + fht

r1[r1 < -maxdepth] = -maxdepth
r1[r1 > 0] = 0

idat = r1
rp = ss.argrelmax(idat, order=16, axis=0)

F = lmod.lith_list['Generic 1'].hintn
c = np.sin(np.deg2rad(lmod.lith_list['Generic 1'].finc))**2
k = 4*np.pi*As0**2/(As*2*F*c*w)

plt.figure(figsize=(9, 6))
plt.ylabel('Northings (m)')
plt.xlabel('Eastings (m)')
plt.title('$As_{0}$')
plt.imshow(As0, extent=magextent, vmin=-30, vmax=280)
plt.colorbar().set_label('nT')
plt.tight_layout()
plt.show()
```

```
plt.figure(figsize=(9, 6))
plt.ylabel('Northings (m)')
plt.xlabel('Eastings (m)')
plt.title('$As_{1}$')
plt.imshow(As, extent=magextent, vmin=-0.2, vmax=1.3)
plt.colorbar().set_label('nT/m')
plt.tight_layout()
plt.show()

plt.figure(figsize=(9, 6))
plt.ylabel('Northings (m)')
plt.xlabel('Eastings (m)')
plt.title('$-r$')
plt.imshow(r1, extent=magextent, vmin=-500, vmax=0, cmap=plt.cm.jet_r)
plt.colorbar().set_label('m')
plt.tight_layout()
plt.show()

plt.figure(figsize=(9, 6))
plt.ylabel('Northings (m)')
plt.xlabel('Eastings (m)')
plt.title('$-r$')
plt.imshow(rz12, extent=magextent, vmin=-500, vmax=0, cmap=plt.cm.jet_r)
plt.colorbar().set_label('m')
plt.tight_layout()
plt.show()

plt.figure(figsize=(9, 6))
plt.ylabel('Northings (m)')
plt.xlabel('Eastings (m)')
plt.title('$k$')
plt.imshow(k, extent=magextent, vmin=-0.1, vmax=0.9)
plt.colorbar().set_label('SI Units')
plt.tight_layout()
plt.show()

plt.figure(figsize=(8, 8))
plt.subplot(3, 3, 1)
plot2(rx12, '(a)', 'Depth (m)', minstd=-400, maxstd=0,
      extent=magextent, cmap=plt.cm.jet_r, notstd=True, show=False)
plt.subplot(3, 3, 2)
plot2(ry12, '(b)', 'Depth (m)', minstd=-400, maxstd=0,
      extent=magextent, cmap=plt.cm.jet_r, notstd=True, show=False)
plt.subplot(3, 3, 3)
plot2(rz12, '(c)', 'Depth (m)', minstd=-400, maxstd=0,
      extent=magextent, cmap=plt.cm.jet_r, notstd=True, show=False)
plt.subplot(3, 3, 4)
plot2(rx01, '(d)', 'Depth (m)', minstd=-400, maxstd=0,
      extent=magextent, cmap=plt.cm.jet_r, notstd=True, show=False)
plt.subplot(3, 3, 5)
plot2(ry01, '(e)', 'Depth (m)', minstd=-400, maxstd=0,
      extent=magextent, cmap=plt.cm.jet_r, notstd=True, show=False)
plt.subplot(3, 3, 6)
plot2(rz01, '(f)', 'Depth (m)', minstd=-400, maxstd=0,
      extent=magextent, cmap=plt.cm.jet_r, notstd=True, show=False)
plt.subplot(3, 3, 7)
plot2(r1, '(g)', 'Depth (m)', minstd=-400, maxstd=0.,
      extent=magextent, cmap=plt.cm.jet_r, notstd=True, show=False)
plt.subplot(3, 3, 9)
plot2(k, '(h)', 'SI units', minstd=1, maxstd=2.5,
      extent=magextent, show=False)
plt.tight_layout()
plt.show()

tmp = lmod.griddata['Magnetic Dataset']
plt.figure(figsize=(7, 6))
plt.title('Peak Locations')
plt.plot(tmp.tlx+rp[1]*tmp.xdim, tmp.tly-rp[0]*tmp.ydim, 'k.')
plt.ylabel('Northings (m)')
plt.xlabel('Eastings (m)')
plt.axis('equal')
plt.xlim([magextent[0], magextent[1]])
plt.ylim([magextent[2], magextent[3]])
```

```
plt.tight_layout()
plt.show()

db = DBSCAN(eps=7, algorithm='brute').fit(np.transpose(rp))
labels = db.labels_
newlabels = (labels != -1)

for i in np.unique(labels):
    if i == -1:
        continue
    elif np.sum(labels == i) < 10:
        newlabels[labels == i] = False
    else:
        newlabels[labels == i] = True

labels = labels[newlabels]
rp = (rp[0][newlabels], rp[1][newlabels])
k = k[rp]

rall = depth_from_tensor(Tx, Ty, Tz, Txx, Tyy, Tzz, Txy, Txz, Tyz, N,
                         fht, magval, dxy, Hx, Hy, Hz)

means = []
meds = []
for i in rall:
    plt.title('depths:'+i)
    depth = rall[i]

    w1 = depth[rp]
    w1 = np.ma.masked_invalid(w1)
    w1 = w1[~w1.mask]
    w1 = w1.flatten()

    width, A0, A1 = width_from_tensor(Hx, Hy, Hz, Txx, Tyy, Tzz, Txy, Txz,
                                      Tyz, fht, dxy, depth, magval, hintn,
                                      Tx, Ty, Tz, cx, cy, cz)

    plt.title('widths:'+i)
    w1 = width[rp]
    w1 = np.ma.masked_invalid(w1)
    w1 = w1[~w1.mask]
    w1 = w1.flatten()

    print('widths:'+i+' mean:', w1.mean(), 'std:', w1.std())
    print('widths:'+i+' median:', np.median(w1))

    means.append(w1.mean())
    meds.append(np.median(w1.std()))

print('width std', np.mean(meds))
print('width mean', np.mean(means))

plt.figure(figsize=(7, 6))
plt.title('Peak Locations')
plt.plot(tmp.tlx+rp[1]*tmp.xdim, tmp.tly-rp[0]*tmp.ydim, 'k.')
plt.ylabel('Northings (m)')
plt.xlabel('Eastings (m)')
plt.axis('equal')
plt.xlim([magextent[0], magextent[1]])
plt.ylim([magextent[2], magextent[3]])
plt.tight layout()
plt.show()

plt.figure(figsize=(9, 6))
plt.title('Peak Locations with Susceptibility')
plt.scatter(tmp.tlx+rp[1]*tmp.xdim, tmp.tly-rp[0]*tmp.ydim, c=k,
            cmap=plt.cm.jet)
plt.ylabel('Northings (m)')
plt.xlabel('Eastings (m)')
plt.axis('equal')
plt.colorbar().set_label('SI units')
plt.xlim([magextent[0], magextent[1]])
plt.ylim([magextent[2], magextent[3]])
```

```
    plt.tight_layout()
    plt.show()

    km = KMeans(n_clusters=numclasses).fit(np.expand_dims(k, 1))
    klabels = km.labels_
    kmc = km.cluster_centers_

    plt.figure(figsize=(7, 5))
    plt.title('Susceptibility Distribution')

    N, bins, patches = plt.hist(k, 100)

    fracs = np.zeros_like(N)
    for i in np.unique(klabels):
        kmin = k[klabels == i].min()
        kmax = k[klabels == i].max()
        filt = np.logical_and(bins >= kmin, bins <= kmax)
        fracs[filt[:-1]] = i

    norm = colors.Normalize(fracs.min(), fracs.max())

    for thisfrac, thispatch in zip(fracs, patches):
        color = plt.cm.jet(norm(thisfrac))
        thispatch.set_facecolor(color)

    plt.ylabel('counts')
    plt.xlabel('k (SI units)')
    sm = plt.cm.ScalarMappable(cmap=plt.cm.jet, norm=norm)
    sm._A = []
    cbar = plt.colorbar(sm, ticks=np.unique(klabels))
    cbar.ax.set_yticklabels((np.unique(klabels)+1).astype(str).tolist())

    cbar.set_label('Class Number')

    plt.tight_layout()
    plt.show()

    print('Susceptibilities: ', kmc)

# remember that r1 is negative below
    depths = ((dtmval.max()-(dtmval[rp]+r1[rp]))/lmod.d_z).astype(int)
    depths[depths >= lmod.numz] = -1

    for i, dep in enumerate(depths):
        yy = int(lmod.numy-rp[0][i]*lmod.numy/dtmval.shape[0])
        xx = int(rp[1][i]*lmod.numx/dtmval.shape[1])

        if dep != -1:
            lmod.lith_index[xx, yy, dep:] = klabels[i]+1

    for i, _ in enumerate(kmc):
        lmod.lith_list['Generic '+str(i+1)] = \
            copy.deepcopy(lmod.lith_list['Generic 1'])
        lmod.lith_list['Generic '+str(i+1)].lith_index = i+1
        lmod.lith_list['Generic '+str(i+1)].susc = kmc[i]

# Save model
    lmod.mlut = {0: [170, 125, 90], 2: [255, 0, 0], 1: [0, 0, 255],
                 3: [0, 255, 0]}

    filename = filename[:-4]+'_out.npz'
    emod = ExportMod3D(None)
    emod.ifile = filename
    emod.indata['Model3D'] = [lmod]

    emod.lmod = lmod
    emod.ifile = str(filename)
    emod.ext = filename[-3:]
    emod.savemodel()

    newmag = calc_field(lmod, magcalc=True)

    ztmp = gridmatch(lmod, 'Magnetic Dataset', 'Calculated Magnetics')
```

```
    dtmp = newmag['Magnetic Dataset']

    xcoords = np.arange(0, dtmp.ydim*dtmp.rows, dtmp.ydim)+dtmp.ydim/2

    plt.figure(figsize=(8, 8))
    plt.subplot(2, 1, 1)
    plt.title('(a)', loc='left')
    plt.ylabel('$B_{tmi}$ (nT)')
    plt.xlabel('Distance (m)')

    col = int(15520/40)
    plt.plot(xcoords, ztmp.data[::-1, col])
    plt.plot(xcoords, dtmp.data[::-1, col], '.')

    plt.subplot(2, 1, 2)
    plt.title('(b)', loc='left')
    plt.ylabel('Depth (m)')
    plt.xlabel('Distance (m)')
    plt.ylim([-500, 0])

    depths = r1[::-1, col]

    # Suspicious depths can be further filtered out here
    depths[73:90] = -500
    depths[584:595] = -500

    rp = ss.argrelmax(depths, order=4)[0]

    plt.plot(xcoords, np.ma.masked_equal(depths, -500))
    for i in rp:
        plt.plot([xcoords[i], xcoords[i]], [-500, depths[i]], 'r')

    plt.show()


def As0_calc(idata):
    """
    Calculates the 0 order analytic signal using hilbert transforms

    Parameters
    ----------
    idata : numpy array
        grid of magnetic data

    Returns
    -------
    As0 : numpy array
        zero order analytic signal

    """

    Hilbx = ss.hilbert(idata, axis=1).imag
    Hilby = ss.hilbert(idata, axis=0).imag
    As0 = np.sqrt(idata**2 + Hilbx**2 + Hilby**2)

    return As0


def depth_from_tensor(Tx, Ty, Tz, Txx, Tyy, Tzz, Txy, Txz, Tyz, N,
                      fht, magval, dxy, Hx, Hy, Hz):
    """
    Calculates depth to source from tensor values

    Parameters
    ----------
    Tx : numpy array
        Tx component
    Ty : numpy array
        Ty component
    Tz : numpy array
        Tz component
    Txx : numpy array
        Txx component
```

```
    Tyy : numpy array
        Tyy component
    Tzz : numpy array
        Tzz component
    Txy : numpy array
        Txy component
    Txz : numpy array
        Txz component
    Tyz : numpy array
        Tyz component
    N : integer
        N value for depth, 0 for steps and 1 for dykes
    fht : float
        flying height
    magval : numpy array
        TMI array
    dxy : float
        grid cell size
    Hx : numpy array
        x derivative of TMI
    Hy : numpy array
        y derivative of TMI
    Hz : numpy array
        z derivative of TMI

    Returns
    -------
    r : python dictionary
        dictionary of different calculated depths
    """

    As0 = As0_calc(magval)
    Asx0 = As0_calc(Tx)
    Asy0 = As0_calc(Ty)
    Asz0 = As0_calc(Tz)

    As, As2 = As_calcs(Hx, Hy, Hz, dxy)
    Asx, Asx2 = As_calcs(Txx, Txy, Txz, dxy)
    Asy, Asy2 = As_calcs(Txy, Tyy, Tyz, dxy)
    Asz, Asz2 = As_calcs(Txz, Tyz, Tzz, dxy)

    Asall0 = np.sqrt(Asx0**2+Asy0**2+Asz0**2)
    Asall1 = np.sqrt(Asx**2+Asy**2+Asz**2)
    Asall2 = np.sqrt(Asx2**2+Asy2**2+Asz2**2)

    r = {}
    r['1'] = -N*As0/As+fht
    r['12'] = -(N+1)*As/As2+fht

    r['a01'] = -N*Asall0/Asall1+fht
    r['a12'] = -(N+1)*Asall1/Asall2+fht

    r['x12'] = -(N+1)*Asx/Asx2+fht
    r['y12'] = -(N+1)*Asy/Asy2+fht
    r['z12'] = -(N+1)*Asz/Asz2+fht

    r['x01'] = -N*Asx0/Asx+fht
    r['y01'] = -N*Asy0/Asy+fht
    r['z01'] = -N*Asz0/Asz+fht

    return r


def width_from_tensor(Hx, Hy, Hz, Txx, Tyy, Tzz, Txy, Txz, Tyz, fht,
                      dxy, depth, TMI, Ba, Tx, Ty, Tz, cx, cy, cz):
    """
    Calculates dyke width from tensor values.


    Parameters
    ----------
    Hx : numpy array
        x derivative of TMI
```

```
Hy : numpy array
    y derivative of TMI
Hz : numpy array
    z derivative of TMI
Txx : numpy array
    Txx component
Tyy : numpy array
    Tyy component
Tzz : numpy array
    Tzz component
Txy : numpy array
    Txy component
Txz : numpy array
    Txz component
Tyz : numpy array
    Tyz component
fht : float
    flying height
depth : numpy array
    depth grid or value
TMI : numpy array
    TMI data
Tx : numpy array
    Tx component
Ty : numpy array
    Ty component
Tz : numpy array
    Tz component
cx : numpy array
    alpha direction cosine
cy : numpy array
    beta direction cosine
cz : numpy array
    gamma direction cosine


Returns
-------
width : numpy array
    grid of widths
A0 : numpy array
    A0 analytic signal
A1 : numpy array
    A1 analytic signal
"""

depth2 = np.abs(depth)+np.abs(fht)

Txxy, Txxx = gradientO4(Txx, dxy)
Tyyy, Tyyx = gradientO4(Tyy, dxy)
Tzzy, Tzzx = gradientO4(Tzz, dxy)
Txyy, Txyx = gradientO4(Txy, dxy)
Txzy, Txzx = gradientO4(Txz, dxy)
Tyzy, Tyzx = gradientO4(Tyz, dxy)

Txxz = Txzx
Tyyz = Tyzy
Tzzz = -(Txxz+Tyyz)
Txyz = Tyzx
Txzz = Tzzx

D = Tx+cx*Ba
E = Ty+cy*Ba
F = Tz+cz*Ba

Htmi_zz = ((Txzz*D+Tzzy*E+Tzzz*F+Txz**2+Tyz**2+Tzz**2)/(TMI+Ba) -
           (Txz*D+Tyz*E+Tzz*F)**2/(TMI+Ba)**3)

Htmi_yz = ((Txyz*D+Tyyz*E+Tzzy*F+Txz*Txy+Tyz*Tyy+Tzz*Tyz)/(TMI+Ba) -
           (Txz*D+Tyz*E+Tzz*F)*(Txy*D+Tyy*E+Tyz*F)/(TMI+Ba)**3)

Htmi_xz = ((Txxz*D+Txyz*E+Tzzx*F+Txz*Txx+Tyz*Txy+Tzz*Txz)/(TMI+Ba) -
           (Txz*D+Tyz*E+Tzz*F)*(Txx*D+Txy*E+Txz*F)/(TMI+Ba)**3)
```

```
    A0 = np.sqrt(Hx**2+Hy**2+Hz**2)
    A1 = np.sqrt(Htmi_xz**2+Htmi_yz**2+Htmi_zz**2)
    c1 = A1/A0

    width = 2*np.sqrt((2*depth2/c1-depth2**2))

    return width, A0, A1


def susc_from_tensor(Tx, Ty, Tz, Txx, Tyy, Tzz, Txz, Tyz, magval, hintn,
                     width, depth1, fht, Hx, Hy, Hz, cy, dxy, Txy):
    """
    Calculates susceptibility and magnetisation from tensor values

    Parameters
    ----------
    Tx : numpy array
        Tx component
    Ty : numpy array
        Ty component
    Tz : numpy array
        Tz component
    Txx : numpy array
        Txx component
    Tyy : numpy array
        Tyy component
    Tzz : numpy array
        Tzz component
    Txz : numpy array
        Txz component
    Tyz : numpy array
        Tyz component
    magval : numpy array
        TMI array
    hintn : float
        ambient field strength in nT
    width : numpy array or float
        dyke width
    depth1 : numpy array
        dyke depth
    fht : float
        flying height
    Hx : numpy array
        x derivative of TMI
    Hy : numpy array
        y derivative of TMI
    Hz : numpy array
        z derivative of TMI
    cy : numpy array
        direction cosine - alpha for ENU/END or beta for NED
    dxy : float
        grid cell size
    Txy : numpy array
        Txy component

    Returns
    -------
    k : python dictionary
        dictionary of different calculated magnetisations and susceptibilities
    """
    depth = np.abs(depth1)+np.abs(fht)

    As0 = As0_calc(magval)
    Asx0 = As0_calc(Tx)
    Asy0 = As0_calc(Ty)
    Asz0 = As0_calc(Tz)

    As, As2 = As_calcs(Hx, Hy, Hz, dxy)
    Asx, Asx2 = As_calcs(Txx, Txy, Txz, dxy)
    Asy, Asy2 = As_calcs(Txy, Tyy, Tyz, dxy)
    Asz, Asz2 = As_calcs(Txz, Tyz, Tzz, dxy)
```

```
    Asall0 = np.sqrt(Asx0**2+Asy0**2+Asz0**2)
    Asall1 = np.sqrt(Asx**2+Asy**2+Asz**2)

    F = hintn
    c = 1-cy**2

    k = {}
    k['mt01'] = As0**2/(200*As*width*c)
    k['mtz1d'] = Asz*depth**2/(width*200*np.sqrt(c))
    k['mtz01'] = Asz0**2/(200*Asz*width*np.sqrt(c))
    k['mtz0d'] = Asz0*depth/(width*200*np.sqrt(c))
    k['mtall'] = Asall0**2/(200*Asall1*width*c)

    k['k01'] = k['mt01']*4*np.pi*100/F
    k['kz1d'] = k['mtz1d']*4*np.pi*100/F
    k['kz01'] = k['mtz01']*4*np.pi*100/F
    k['kz0d'] = k['mtz0d']*4*np.pi*100/F
    k['kall'] = k['mtall']*4*np.pi*100/F

    return k


def cosines_from_tensor(Hx, Hy, Hz, Txx, Tyy, Tzz, Txy, Txz, Tyz):
    """
    Direction cosines from tensor, first approx equation.

    Parameters
    ----------
    Hx : numpy array
        x derivative of TMI
    Hy : numpy array
        y derivative of TMI
    Hz : numpy array
        z derivative of TMI
    Txx : numpy array
        Txx component
    Tyy : numpy array
        Tyy component
    Tzz : numpy array
        Tzz component
    Txy : numpy array
        Txy component
    Txz : numpy array
        Txz component
    Tyz : numpy array
        Tyz component

    Returns
    -------
    cx : numpy array
        alpha direction cosine
    cy : numpy array
        beta direction cosine
    cz : numpy array
        gamma direction cosine
    """

    warnings.filterwarnings('ignore')
    cx2 = (Hx*(Tyz**2 - Tyy*Tzz) +
           Hy*(Txy*Tzz - Txz*Tyz) +
           Hz*(Txz*Tyy - Txy*Tyz))/(Txx*Tyz**2 - Txx*Tyy*Tzz + Txy**2*Tzz -
                                    2*Txy*Txz*Tyz + Txz**2*Tyy)

    cy2 = (Hx*(Txz*Tyz - Txy*Tzz) +
           Hy*(Txx*Tzz - Txz**2) +
           Hz*(Txy*Txz - Txx*Tyz))/(Txx*Tyy*Tzz - Txx*Tyz**2 - Txy**2*Tzz +
                                    2*Txy*Txz*Tyz - Txz**2*Tyy)

    cz2 = (Hx*(Txy*Tyz - Txz*Tyy) +
           Hy*(Txy*Txz - Txx*Tyz) +
           Hz*(Txx*Tyy - Txy**2))/(Txx*Tyy*Tzz - Txx*Tyz**2 - Txy**2*Tzz +
                                   2*Txy*Txz*Tyz - Txz**2*Tyy)
```

```
    return (cx2, cy2, cz2)


def cosines_from_tensor2(tx, ty, f, bxx, byy, bxy, bxz, byz, bx, by, bz):
    """
    Direction cosines from tensor, second approx equation.

    Parameters
    ----------
    tx : numpy array
        x derivative of TMI
    ty : numpy array
        y derivative of TMI
    f : numpy array
        TMI data
    bxx : numpy array
        Txx component
    byy : numpy array
        Tyy component
    bzz : numpy array
        Tzz component
    bxy : numpy array
        Txy component
    bxz : numpy array
        Txz component
    byz : numpy array
        Tyz component
    bx : numpy array
        Tx component
    by : numpy array
        Ty component
    bz : numpy array
        Tz component

    Returns
    -------
    cx : numpy array
        alpha direction cosine
    cy : numpy array
        beta direction cosine
    cz : numpy array
        gamma direction cosine
    """

    warnings.filterwarnings('ignore')
    cx2 = (bxy*byz*f - bxy*bz*ty + bxz*by*ty - bxz*byy*f - by*byz*tx +
           byy*bz*tx)/(bx*bxy*byz - bx*bxz*byy - bxx*by*byz + bxx*byy*bz -
                       bxy**2*bz + bxy*bxz*by)
    cy2 = (-bx*bxz*ty + bx*byz*tx - bxx*byz*f + bxx*bz*ty + bxy*bxz*f -
           bxy*bz*tx)/(bx*bxy*byz - bx*bxz*byy - bxx*by*byz + bxx*byy*bz -
                       bxy**2*bz + bxy*bxz*by)
    cz2 = (bx*bxy*ty - bx*byy*tx - bxx*by*ty + bxx*byy*f - bxy**2*f +
           bxy*by*tx)/(bx*bxy*byz - bx*bxz*byy - bxx*by*byz + bxx*byy*bz -
                       bxy**2*bz + bxy*bxz*by)

    return (cx2, cy2, cz2)


def cosines_from_tensor3(tx, ty, tz, f, bxx, byy, bxy, bxz, byz, B, bx, by,
                         bz):
    """
    Direction cosines from tensor, full equation.

    Parameters
    ----------
    tx : numpy array
        x derivative of TMI
    ty : numpy array
        y derivative of TMI
    tz : numpy array
        z derivative of TMI
    f : numpy array
        TMI data
```

```
    bxx : numpy array
        Txx component
    byy : numpy array
        Tyy component
    bzz : numpy array
        Tzz component
    bxy : numpy array
        Txy component
    bxz : numpy array
        Txz component
    byz : numpy array
        Tyz component
    B : float
        ambient field strength
    bx : numpy array
        Tx component
    by : numpy array
        Ty component
    bz : numpy array
        Tz component

    Returns
    -------
    cx : numpy array
        alpha direction cosine
    cy : numpy array
        beta direction cosine
    cz : numpy array
        gamma direction cosine
    """

    warnings.filterwarnings('ignore')
    cx2 = (-B*bxx*bxy*ty + B*bxx*byy*tx - B*bxy*byy*ty - B*bxy*byz*tz +
           B*bxz*byy*tz - B*bxz*byz*ty + B*byy**2*tx + B*byz**2*tx -
           bx*bxx**2*byy + bx*bxx*bxy**2 - bx*bxx*byy**2 - bx*bxx*byz**2 +
           bx*bxy**2*byy + 2*bx*bxy*bxz*byz - bx*bxz**2*byy - bxx*bxy*f*ty +
           bxx*byy*f*tx - bxy*byy*f*ty - bxy*byz*f*tz + bxz*byy*f*tz -
           bxz*byz*f*ty + byy**2*f*tx +
           byz**2*f*tx)/(B*(bxx**2*byy - bxx*bxy**2 + bxx*byy**2 +
                            bxx*byz**2 - bxy**2*byy - 2*bxy*bxz*byz +
                            bxz**2*byy))
    cy2 = (B*bxx**2*ty - B*bxx*bxy*tx + B*bxx*byy*ty + B*bxx*byz*tz -
           B*bxy*bxz*tz - B*bxy*byy*tx + B*bxz**2*ty - B*bxz*byz*tx -
           bxx**2*by*byy + bxx**2*f*ty + bxx*bxy**2*by - bxx*bxy*f*tx -
           bxx*by*byy**2 - bxx*by*byz**2 + bxx*byy*f*ty + bxx*byz*f*tz +
           bxy**2*by*byy + 2*bxy*bxz*by*byz - bxy*bxz*f*tz - bxy*byy*f*tx -
           bxz**2*by*byy + bxz**2*f*ty -
           bxz*byz*f*tx)/(B*(bxx**2*byy - bxx*bxy**2 + bxx*byy**2 +
                             bxx*byz**2 - bxy**2*byy - 2*bxy*bxz*byz +
                             bxz**2*byy))
    cz2 = (-B*bxx*byy*tz + B*bxx*byz*ty + B*bxy**2*tz - B*bxy*bxz*ty -
           B*bxy*byz*tx + B*bxz*byy*tx - bxx**2*byy*bz + bxx*bxy**2*bz -
           bxx*byy**2*bz - bxx*byy*f*tz - bxx*byz**2*bz + bxx*byz*f*ty +
           bxy**2*byy*bz + bxy**2*f*tz + 2*bxy*bxz*byz*bz - bxy*bxz*f*ty -
           bxy*byz*f*tx - bxz**2*byy*bz +
           bxz*byy*f*tx)/(B*(bxx**2*byy - bxx*bxy**2 + bxx*byy**2 +
                             bxx*byz**2 - bxy**2*byy - 2*bxy*bxz*byz +
                             bxz**2*byy))

    return (cx2, cy2, cz2)


def rem_from_cosines(cx2, cy2, cz2, mt, susc, hintn, inc, dec, azim):
    """
    Remanence calculations


    Parameters
    ----------
    cx2 : numpy array
        alpha direction cosine
    cy2 : numpy array
        beta direction cosine
```

```
    cz2 : numpy array
        gamma direction cosine
    mt : numpy array
        total magnetisation
    susc : numpy array
        susceptibility
    hintn : float
        ambient field strength in nT
    inc : float
        inclination of inducing field
    dec : float
        declination of inducing field
    azim : float
        angle between x direction and North.

    Returns
    -------
    mstr : numpy array
        remanent magnetisation
    minc : numpy array
        remanent inclination
    mdec : numpy array
        remanent declination
    """
    f1 = susc*hintn
    fa, fb, fc = dircos(inc, dec, azim)

    A = cx2*mt - f1*fa/(400*np.pi)
    B = cy2*mt - f1*fb/(400*np.pi)
    C = cz2*mt - f1*fc/(400*np.pi)

    mdec = -2*np.arctan2((A + np.sqrt(A**2 + B**2)), B)
    mdec = np.rad2deg(mdec)-180+azim
    mdec[mdec > 180] -= 360
    mdec[mdec < -180] += 360

    tmp1 = -np.sqrt(A**2+B**2+C**2)+np.sqrt(A**2+B**2)
    tmp2 = C
    minc = -2*np.arctan2(tmp1, tmp2)
    minc = np.rad2deg(minc)

    minc[minc > 90] -= 360
    minc[minc < -90] += 360

    ma, mb, mc = dircos(minc, mdec, azim)

    mstr = np.sqrt(A**2+B**2+C**2)

    return (mstr, minc, mdec)


def tallafwd():
    """
    Simple forward model for simulating tallawang anomaly. See section 6.3.2

    Parameters
    ----------
    None

    Returns
    -------
    None
    """

    hintn = 56701.6
    inc = -63.0575
    dec = 11.47
    susc = 2.5
    mstr = 30
    minc = -70
    mdec = -60
    dxy = 5
    fht = 0
```

```
depth = 20
width = 10

tcube = TensorCube()
tcube.susc = susc
tcube.mstrength = mstr
tcube.inc = inc
tcube.dec = dec
tcube.minc = minc
tcube.mdec = mdec
tcube.hintn = hintn
tcube.height = 0
tcube.dxy = dxy
tcube.u = [100-width, 100]
cpnt = int(np.mean(tcube.u)/dxy)
azim = tcube.azim

xblocks = []
yblocks = []
for offset in np.arange(depth, 150, dxy):

    print('offset', offset)
    xoff = (offset-depth)*20/(150-depth)
    tcube.u = [102-width-xoff, 102-xoff]
    tcube.v = [0, 150]
    tcube.w = [-offset, -(offset+dxy)]
    tcube.rc = 150

    xblocks.append(tcube.u)
    yblocks.append(tcube.w)

    tcube.calc_all()
    coords = tcube.xyall

    if offset == depth:
        Txx = tcube.bxx
        Txy = tcube.bxy
        Txz = tcube.bxz
        Tyy = tcube.byy
        Tyz = tcube.byz
        Tzz = tcube.bzz
        Tx = tcube.bx
        Ty = tcube.by
        Tz = tcube.bz
        magval = tcube.magval
    else:
        Txx = Txx + tcube.bxx
        Txy = Txy + tcube.bxy
        Txz = Txz + tcube.bxz
        Tyy = Tyy + tcube.byy
        Tyz = Tyz + tcube.byz
        Tzz = Tzz + tcube.bzz
        Tx = Tx + tcube.bx
        Ty = Ty + tcube.by
        Tz = Tz + tcube.bz
        magval = magval + tcube.magval

print('mt (tcube)', tcube.mt/100)
cx = tcube.cx
cy = tcube.cy
cz = tcube.cz

Hx = (Txx*(Tx+cx*hintn)+Txy*(Ty+cy*hintn)+Txz*(Tz+cz*hintn))/(magval+hintn)
Hy = (Txy*(Tx+cx*hintn)+Tyy*(Ty+cy*hintn)+Tyz*(Tz+cz*hintn))/(magval+hintn)
Hz = (Txz*(Tx+cx*hintn)+Tyz*(Ty+cy*hintn)+Tzz*(Tz+cz*hintn))/(magval+hintn)

As, As2 = As_calcs(Hx, Hy, Hz, dxy)
Asz, Asz2 = As_calcs(Txz, Tyz, Tzz, dxy)
Asy, Asy2 = As_calcs(Txy, Tyy, Tyz, dxy)
Asx, Asx2 = As_calcs(Txx, Txy, Txz, dxy)

As0 = As0_calc(magval)
Asx0 = As0_calc(Tx)
```

```
    Asy0 = As0_calc(Ty)
    Asz0 = As0_calc(Tz)

    plt.figure(figsize=(8, 6))
    plt.subplot(2, 1, 1)
    plt.plot(coords, magval[8], label='TMI calculated')
    plt.xlim(0, 150)
    plt.ylabel('(nT)')
    plt.xlabel('Distance (m)')
    plt.yticks(rotation='vertical', va='center')
    plt.axvline(x=97, c='r', ls='dashed')
    plt.legend()

    plt.subplot(2, 1, 2)
    plt.xlim(0, 150)
    plt.ylabel('Depth (m)')
    plt.xlabel('Distance (m)')
    for i, _ in enumerate(xblocks):
        x1, x2 = xblocks[i]
        y1, y2 = yblocks[i]
        plt.plot([x1, x2, x2, x1, x1], [y1, y1, y2, y2, y1], 'k')

    plt.tight_layout()
    plt.show()

    N = 1
    depths = depth_from_tensor(Tx, Ty, Tz, Txx, Tyy, Tzz, Txy, Txz, Tyz, N,
                               fht, magval, dxy, Hx, Hy, Hz)

    print('depth fixed')
    depth = abs(depth)
    tmp = width_from_tensor(Hx, Hy, Hz, Txx, Tyy, Tzz, Txy, Txz, Tyz, fht,
                            dxy, depth, magval, hintn, Tx, Ty, Tz, cx, cy,
                            cz)
    widths, A0, A1 = tmp

#    print('width fixed')
    width = np.zeros_like(widths)+widths[15].min()

    tmp = cosines_from_tensor3(Hx, Hy, Hz, magval, Txx, Tyy, Txy, Txz,
                               Tyz, hintn, Tx, Ty, Tz)
    cx2, cy2, cz2 = tmp

    k = susc from tensor(Tx, Ty, Tz, Txx, Tyy, Tzz, Txz, Tyz, magval, hintn,
                         width, depth, fht, Hx, Hy, Hz, cy2, dxy, Txy)

    mt1 = tcube.mt/100

    # Susceptibility and magnetisation can be fixed for the remanence
    # calculations.
    susc1 = susc
    mt1 = 114  # 140 is the accurate value, and 114 is the innacurate value

    tmp = rem_from_cosines(cx2, cy2, cz2, mt1, susc1, hintn, inc, dec, azim)
    mstr, minc2, mdec2 = tmp

    print('Center point', cpnt*dxy)
    print('Width', width[cpnt, cpnt])
    print('cx', cx, cx2[cpnt, cpnt])
    print('cy', cy, cy2[cpnt, cpnt])
    print('cz', cz, cz2[cpnt, cpnt])
    print('mdec2', mdec, mdec2[cpnt, cpnt])
    print('minc2', minc, minc2[cpnt, cpnt])
    print('mstr2', mstr[cpnt, cpnt])

    coords = tcube.xyall
    cpnt = magval.shape[0]//2

# Convert the EN to NE axes.
    Asx, Asy = Asy, Asx
    Asx0, Asy0 = Asy0, Asx0
    Asx2, Asy2 = Asy2, Asx2
    Txx, Tyy = Tyy, Txx
```

```
    Tyz, Txz = Txz, Tyz

    r = depths
    r['x01'], r['y01'] = r['y01'], r['x01']
    r['x12'], r['y12'] = r['y12'], r['x12']

# Plot data
    i = cpnt
    cpnt = 97  # black dashed line on plots
    minc = minc2
    mdec = mdec2
    r = depths

    dxy = coords[1]-coords[0]
    icpnt = int(round(cpnt/dxy))
    s = slice(icpnt-1, icpnt+1)
    s2 = slice(icpnt-2, icpnt+2)
    s3 = slice(int(60/dxy), int(130/dxy))
    s4 = slice(5, -1)
    s = s4
    s2 = s4

    # Plot the tensor components
    plt.figure(figsize=(8, 8))
    plt.subplot(331)
    plt.title('(a) $T_{xx}$', loc='left')
    plt.ylabel('(nT/m)')
    plt.xlabel('Distance (m)')
    plt.plot(coords, Txx[i])
    plt.yticks(rotation='vertical', va='center')

    plt.subplot(332)
    plt.title('(b) $T_{xy}$', loc='left')
    plt.ylabel('(nT/m)')
    plt.xlabel('Distance (m)')
    plt.plot(coords, Txy[i])
    plt.yticks(rotation='vertical', va='center')

    plt.subplot(333)
    plt.title('(c) $T_{xz}$', loc='left')
    plt.ylabel('(nT/m)')
    plt.xlabel('Distance (m)')
    plt.plot(coords, Txz[i])
    plt.yticks(rotation='vertical', va='center')

    plt.subplot(335)
    plt.title('(d) $T_{yy}$', loc='left')
    plt.ylabel('(nT/m)')
    plt.xlabel('Distance (m)')
    plt.plot(coords, Tyy[i])
    plt.yticks(rotation='vertical', va='center')

    plt.subplot(336)
    plt.title('(e) $T_{yz}$', loc='left')
    plt.ylabel('(nT/m)')
    plt.xlabel('Distance (m)')
    plt.plot(coords, Tyz[i])
    plt.yticks(rotation='vertical', va='center')

    plt.subplot(339)
    plt.title('(f) $T {zz}$', loc='left')
    plt.ylabel('(nT/m)')
    plt.xlabel('Distance (m)')
    plt.plot(coords, Tzz[i])
    plt.yticks(rotation='vertical', va='center')

    plt.tight_layout()
    plt.show()

    plt.figure(figsize=(8, 8))
    plt.subplot(221)
    plt.title('(a)', loc='left')
    plt.plot(coords[s4], As0[i][s4], label='$As_{0}$')
```

```
plt.plot(coords[s4], Asy0[i][s4], label='$As_{y0}$')
plt.plot(coords[s4], Asz0[i][s4], label='$As_{z0}$')
plt.xlabel('Distance (m)')
plt.ylabel('(nT)')
plt.legend()
plt.yticks(rotation='vertical', va='center')
plt.axvline(x=cpnt, c='k', ls='dashed')
plt.xlim(coords[0], coords[-1])

plt.subplot(222)
plt.title('(b)', loc='left')
plt.plot(coords[s4], As[i][s4], label='$As_{1}$')
plt.plot(coords[s4], Asy[i][s4], label='$As_{y1}$')
plt.plot(coords[s4], Asz[i][s4], label='$As_{z1}$')
plt.xlabel('Distance (m)')
plt.ylabel('(nT/m)')
plt.legend()
plt.yticks(rotation='vertical', va='center')
plt.axvline(x=cpnt, c='k', ls='dashed')
plt.xlim(coords[0], coords[-1])

plt.subplot(223)
plt.title('(c)', loc='left')
plt.plot(coords[s], r['1'][i][s], label=r'$\frac{N As_{0}}{As_{1}}$')
plt.plot(coords[s], r['y01'][i][s], label=r'$\frac{N As_{y0}}{As_{y1}}$')
plt.plot(coords[s], r['z01'][i][s], label=r'$\frac{N As_{z0}}{As_{z1}}$')
plt.plot(coords[s], r['a01'][i][s], label=r'$\frac{N As_{xyz0}}{As_{xyz1}}$')
plt.axvline(x=cpnt, c='k', ls='dashed')
plt.xlabel('Distance (m)')
plt.ylabel('Depth (m)')
plt.xlim(coords[0], coords[-1])
plt.ylim(-25, 0)
plt.legend(loc=2, prop={'size': 12})
plt.yticks(rotation='vertical', va='center')

plt.tight_layout()
plt.show()

plt.figure(figsize=(8, 8))
plt.subplot(221)
plt.title('(a)', loc='left')
plt.plot(coords[s4], As[i][s4], label='$As_{1}$')
plt.plot(coords[s4], Asy[i][s4], label='$As_{y1}$')
plt.plot(coords[s4], Asz[i][s4], label='$As_{z1}$')
plt.xlabel('Distance (m)')
plt.ylabel('(nT/m)')
plt.legend()
plt.yticks(rotation='vertical', va='center')
plt.axvline(x=cpnt, c='k', ls='dashed')
plt.xlim(coords[0], coords[-1])

plt.subplot(222)
plt.title('(b)', loc='left')
plt.plot(coords[s4], As2[i][s4], label='$As_{2}$')
plt.plot(coords[s4], Asy2[i][s4], label='$As_{y2}$')
plt.plot(coords[s4], Asz2[i][s4], label='$As_{z2}$')
plt.xlabel('Distance (m)')
plt.ylabel('(nT/m$^2$)')
plt.legend()
plt.yticks(rotation='vertical', va='center')
plt.axvline(x=cpnt, c='k', ls='dashed')
plt.xlim(coords[0], coords[-1])

plt.subplot(223)
plt.title('(c)', loc='left')
plt.plot(coords[s], r['12'][i][s],
         label=r'$\frac{(N+1) As_{1}}{As_{2}}$')
plt.plot(coords[s], r['y12'][i][s],
         label=r'$\frac{(N+1) As_{y1}}{As_{y2}}$')
plt.plot(coords[s], r['z12'][i][s],
         label=r'$\frac{(N+1) As_{z1}}{As_{z2}}$')
plt.plot(coords[s], r['a12'][i][s],
         label=r'$\frac{(N+1) As_{xyz1}}{As_{xyz2}}$')
```

```
plt.axvline(x=cpnt, c='k', ls='dashed')
plt.xlabel('Distance (m)')
plt.ylabel('Depth (m)')
plt.xlim(coords[0], coords[-1])
plt.ylim(-30, 0)

plt.legend(loc=2, prop={'size': 12})
plt.yticks(rotation='vertical', va='center')

plt.tight_layout()
plt.show()

plt.figure(figsize=(8, 8))
plt.subplot(221)
plt.title('(a)', loc='left')
plt.plot(coords[s4], A0[i][s4], label='$A_{0}$')
plt.xlabel('Distance (m)')
plt.ylabel('(nT/m)')
plt.legend()
plt.yticks(rotation='vertical', va='center')
plt.axvline(x=cpnt, c='k', ls='dashed')
plt.xlim(coords[0], coords[-1])

plt.subplot(222)
plt.title('(b)', loc='left')
plt.plot(coords[s4], A1[i][s4], label='$A_{1}$')
plt.xlabel('Distance (m)')
plt.ylabel('(nT/m$^2$)')
plt.legend()
plt.yticks(rotation='vertical', va='center')
plt.xlim(coords[0], coords[-1])
plt.axvline(x=cpnt, c='k', ls='dashed')
plt.xlim(coords[0], coords[-1])

plt.subplot(223)
plt.title('(c)', loc='left')
plt.plot(coords[s4], widths[i][s4],
         label=r'$2 \sqrt{\frac{2d}{c}-d^2}$')
plt.xlabel('Distance (m)')
plt.ylabel('Width (m)')
plt.xlim(coords[0], coords[-1])
plt.legend(prop={'size': 12})
plt.axvline(x=cpnt, c='k', ls='dashed')
plt.yticks(rotation='vertical', va='center')

plt.tight_layout()
plt.show()

lmt01 = r'$\frac{As_{0}^2}{As_{1} 200 w c}$'
lmtz1d = r'$\frac{As {z1}d^2}{As {1} 200 w \sqrt{c}}$'
lmtz01 = r'$\frac{As_{z0}^2}{As_{z1} 200 w \sqrt{c}}$'
lmtz0d = r'$\frac{As_{z0}d}{200 w \sqrt{c}}$'
lmtall = r'$\frac{As_{xyz0}^2}{As_{xyz1} 200 w \sqrt{c}}$'
lk01 = r'$\frac{4 \pi As_{0}^2}{As_{1} 2 B_{a} w c}$'
lkz1d = r'$\frac{4 \pi As_{z1}d^2}{As_{1} 2 B_{a} w \sqrt{c}}$'
lkz01 = r'$\frac{4 \pi As {z0}^2}{As {z1} 2 B {a} w \sqrt{c}}$'
lkz0d = r'$\frac{4 \pi As_{z0}d}{2 B_{a} w \sqrt{c}}$'
lkall = r'$\frac{4 \pi As_{xyz0}^2}{As_{xyz1} 2 B_{a} w \sqrt{c}}$'

plt.figure(figsize=(8, 4))
plt.subplot(121)
plt.title('(a) $Susceptibility$', loc='left')
plt.ylabel('(SI)')
plt.xlabel('Distance (m)')
plt.plot(coords[s3], k['k01'][i][s3], label=lk01)
plt.plot(coords[s3], k['kz1d'][i][s3], label=lkz1d)
plt.plot(coords[s3], k['kz01'][i][s3], label=lkz01)
plt.plot(coords[s3], k['kz0d'][i][s3], label=lkz0d)
plt.plot(coords[s3], k['kall'][i][s3], label=lkall)
plt.xlim(coords[0], coords[-1])
plt.legend(loc=2, prop={'size': 13})
plt.yticks(rotation='vertical', va='center')
plt.axvline(x=cpnt, c='k', ls='dashed')
```

```
plt.subplot(122)
plt.title('(b) $Magnetisation$', loc='left')
plt.ylabel('(A/m)')
plt.xlabel('Distance (m)')
plt.plot(coords[s3], k['mt01'][i][s3], label=lmt01)
plt.plot(coords[s3], k['mtz1d'][i][s3], label=lmtz1d)
plt.plot(coords[s3], k['mtz01'][i][s3], label=lmtz01)
plt.plot(coords[s3], k['mtz0d'][i][s3], label=lmtz0d)
plt.plot(coords[s3], k['mtall'][i][s3], label=lmtall)
plt.xlim(coords[0], coords[-1])
plt.legend(loc=2, prop={'size': 13})
plt.yticks(rotation='vertical', va='center')
plt.axvline(x=cpnt, c='k', ls='dashed')

plt.tight_layout()
plt.show()

plt.figure(figsize=(8, 5))
plt.subplot(121)
plt.title('(a)', loc='left')
plt.plot(coords[s4], cx2[i][s4], label=r'$\alpha$')
plt.plot(coords[s4], cy2[i][s4], label=r'$\beta$')
plt.plot(coords[s4], cz2[i][s4], label=r'$\gamma$')
plt.xlabel('Distance (m)')
plt.ylabel('(direction cosine)')
plt.ylim(-1.2, 1.)
plt.legend()
plt.yticks(rotation='vertical', va='center')
plt.axvline(x=cpnt, c='k', ls='dashed')
plt.xlim(coords[0], coords[-1])

plt.subplot(122)
plt.title('(b)', loc='left')
plt.plot(coords[s2], minc[i][s2], label='inc')
plt.plot(coords[s2], mdec[i][s2], label='dec')
plt.xlabel('Distance (m)')
plt.ylabel('(degrees)')
plt.yticks(rotation='vertical', va='center')
plt.axvline(x=cpnt, c='k', ls='dashed')
plt.xlim(coords[0], coords[-1])
plt.legend()

plt.tight layout()
plt.show()

# First do inclination

cx1, cy1, cz1 = dircos(inc, dec, azim)
xvals = coords
yvals = magval[i]

plt.figure(figsize=(8, 4))
plt.subplot(121)
plt.title('(a)', loc='left')
plt.ylabel('(nT)')
plt.xlabel('Distance (m)')
plt.yticks(rotation='vertical', va='center')

cmagn1 = np.sqrt(cx1**2+cy1**2+cz1**2)

uvals = np.zeros_like(coords) + np.sqrt(cx1**2+cy1**2)/cmagn1
vvals = np.zeros_like(coords) + cz1/cmagn1

Q1 = plt.quiver(xvals, yvals, uvals, vvals, color='r', headwidth=5)

cmagn2 = np.sqrt(cx2[i]**2+cy2[i]**2+cz2[i]**2)

uvals = np.zeros_like(coords) + np.sqrt(cx2[i]**2+cy2[i]**2)/cmagn2
vvals = np.zeros_like(coords) + cz2[i]/cmagn2

Q2 = plt.quiver(xvals, yvals, uvals, vvals, color='y', headwidth=5)
```

```
    plt.quiverkey(Q1, 0.1, .9, 1, 'I (ambient field)', labelpos='E')
    plt.quiverkey(Q2, 0.1, .8, 1, 'I (resultant)', labelpos='E')

    # Now declination

    plt.subplot(122)
    plt.ylabel('(nT)')
    plt.xlabel('Distance (m)')
    plt.yticks(rotation='vertical', va='center')
    plt.title('(b)', loc='left')

    cmagn1 = np.sqrt(cx1**2+cy1**2)
    cmagn2 = np.sqrt(cx2[i]**2+cy2[i]**2)

    if azim == 90:
        uvals = np.zeros_like(coords) + cx1/cmagn1
        vvals = np.zeros_like(coords) - cy1/cmagn1
    else:
        uvals = np.zeros_like(coords) + cy1/cmagn1
        vvals = np.zeros_like(coords) + cx1/cmagn1

    Q1 = plt.quiver(xvals, yvals, uvals, vvals, color='r', headwidth=5)

    if azim == 90:
        uvals = np.zeros_like(coords) + cx2[i]/cmagn2
        vvals = np.zeros_like(coords) - cy2[i]/cmagn2
    else:
        uvals = np.zeros_like(coords) + cy2[i]/cmagn2
        vvals = np.zeros_like(coords) + cx2[i]/cmagn2

    Q2 = plt.quiver(xvals, yvals, uvals, vvals, color='y', headwidth=5)

    plt.quiverkey(Q1, 0.1, 0.9, 1, 'D (ambient field)', labelpos='E')
    plt.quiverkey(Q2, 0.1, 0.8, 1, 'D (resultant)', labelpos='E')

    plt.tight_layout()
    plt.show()

    inc = np.rad2deg(np.arctan(cz2[i][icpnt] /
                               np.sqrt(cx2[i][icpnt]**2+cy2[i][icpnt]**2)))
    dec = np.rad2deg(np.arctan(cy2[i][icpnt]/cx2[i][icpnt]))+azim
    print('field inc', inc)
    print('field dec', dec)


# This section calls the program. You can use either of the two routines below.
if __name__ == "__main__":
    try:
        sys.path.index(PyGMIPATH)
    except ValueError:
        sys.path.append(PyGMIPATH)

#    tests()
#    remanence()
#    interp(r'.\data2\licht1.npz')
    tallafwd()

    print('Finished!')
    winsound.PlaySound('SystemQuestion', winsound.SND_ALIAS)
```