Contributions to the Study of Lithospheric Deformation and Seismicity in Stable Continental Regions

---

Tom New

Supervisor: **Dr Alastair Sloan**

University of Cape Town

Abstract

Recently, the field of geophysics has seen increasing recognition of the unique character of deformation and seismicity in stable continental regions (SCRs). However several important questions remain understudied. What controls the locations of earthquakes in SCRs? How well do observations, in SCRs, of elastic strain accumulation and release correlate with each other? How well do they correlate with stresses and geological proxies for rheological variation?

The ultimate goal of this study was to better understand stable continental regions like southern Africa, where large earthquakes occur despite not being near plate boundaries, for example the 2017 M_w 6.5 earthquake in Moiyabana, Botswana. One way of studying the stress and strain in stable continental regions is by understanding the surface deformation of the region. This deformation is easily studied using global navigation satellite system (GNSS) velocity data. One of the biggest difficulties when it comes to GNSS data is that it isn't collected on a regular grid, but rather as irregular data points that need to be interpolated. This research investigated multiple interpolation methods and recommended two methods that best replicate the original velocity field (using a well populated dataset from Southeast Asia). These interpolated GNSS data can then be used to determine deviatoric strain in a region, which can in turn be fed into numerical stress models.

However, limited GNSS data exist across southern Africa, and therefore topographic data was used to calculate the gravitational potential energy, and in turn the body stress and deviatoric stress for the region. This study also investigated how this deviatoric stress (or deviatoric strain) can be more accurately calculated on a spherical rather than a flat surface, which is particularly important over large study areas. Across southern Africa, data show that deviatoric stress lined up with stress data within mobile belts. This suggests that in these weaker mobile belt crust (such as the Namaqua-Natal and Damara-Chobe belts), gravitational collapse is the dominant driver of deformation, which is in line with conclusions that have been made in previous literature. In other regions, deviatoric stress vectors and stress data do not coincide and therefore there are other forces at play. These observations are obviously restricted by limited data coverage; it remains an open question if areas that have increased deviatoric stress due to gravitational collapse, which are also aligned with the orientation of weak zones, will have elevated strain in the long term.

Acknowledgements

I would like to thank Alastair Sloan for his scientific suggestions and collaboration, as well as his advice when hashing out problems with regard to the Python code which was written over the course of the project. Thanks also to the two anonymous reviewers for their helpful comments, and whose advice has improved the quality of this work. Immense gratitude goes to Morgan for her emotional support, technical help with LaTeX, and proof-reading. I'd also like to thank my parents for their understanding over the course of this journey, and one of my grandmothers for taking every opportunity to remind me how long it has taken me to reach the end of said journey (you know which one you are).

Contents

1 Introduction

1.1 Continental tectonics

Although the theory of continental drift was famously ridiculed for decades, it started to become accepted in the mid-20th century, supported by evidence from paleomagnetic observations (Runcorn, 1965), oceanic geology and mid-ocean ridge spread (Heezen, 1960), and volcanic island arcs (Coats, 1950). The theory was refined into what is now called plate tectonics and has become widely accepted (McKenzie, 1966; McKenzie & Parker, 1967). It has done a remarkable job of explaining the behaviour of this aspect of the earth's dynamics, with extraordinary predictive power. In this formulation, most of the earth's outermost layers are rigid, and deformation is taken up within a few tens of kilometers of plate boundaries.

However, this mostly-rigid plate model in which there is relative motion of large (or several smaller) plates cannot adequately explain observations in continental regions (McKenzie, 1972, 1976; Molnar & Tapponnier, 1975). It is now widely accepted that zones of continental collision can undergo rapid, distributed lithospheric deformation (e.g. Toussaint, Burov and Avouac, 2004). Modelling the relationship between stress, rheology, and strain in such areas is of interest as these are some of the most rapidly deforming places on Earth, providing a window into past and future continental evolution. They are also important from a seismic hazard perspective, since these regions are notorious for frequent (and often unexpected) earthquakes (Figure 1.1), which can cause significant loss of human life and inflict enormous damage to property and infrastructure (England & Jackson, 2011).

The distributed deformation in these zones have been successfully modelled by considering the expected strain within a homogeneous viscous fluid acted on by stresses from plate boundaries, as well as body forces arising from the distribution of gravitational potential energy within the deforming region (England, Houseman & Nocquet, 2016; Walters, England & Houseman, 2017). Such studies typically model the stresses by assuming that the lithosphere acts as a thin sheet (i.e. that vertical gradients of horizontal velocity are negligible) and that variations in surface topography are isostatically compensated at depth (England & Houseman, 1986). The distribution and orientation of the strain field calculated from the stresses can then be compared to observations of strain from, for example, long-term Global Navigation Satellite System (GNSS) velocity measurements (e.g. Zhang et al., 2004), the distribution of earthquake moment release (e.g. Heidbach et al., 2010; Selvaggi, 1998; Zarifi, Nilfouroushan and Raeesi, 2014), and even geological estimates of long-term fault slip rates (e.g. Khodaverdian, Zafarani and Rahimian, 2015). Details of some of these methods are discussed in Section 1.2, and this approach has been found to result in a good match with the observed strain patterns in regions such as Tibet (England & Houseman, 1986; Houseman & England, 1986; Zhao et al., 2015), Anatolia (England, Houseman & Nocquet, 2016; Khodaverdian, Zafarani & Rahimian, 2015), and New Zealand (Beanland & Haines, 1998; Hirschberg, Lamb & Savage, 2019; Lamb, 2015). These studies have generally found that the observed strain pattern can be successfully recreated with surprisingly homogeneous rheology within a particular area of distributed deformation, often with the addition of rigid microplates embedded into the weaker region (England & Houseman, 1986).

1.2 Finite-element crustal deformation modelling

First introduced by Tapponnier and Molnar (1976, 1977), an early approach to modelling the problem of lithospheric deformation involved assuming plane horizontal stress, and treating continental collision as the encroachment of a rigid indenter into a plastic deforming medium. Slip-lines are computed from boundary conditions, and it was postulated that the orientations of the slip-lines would predict the orientation of strike-slip faults in the collision zone. While this model saw limited qualitative success, the authors themselves recognised it had significant limitations: (1) slip-lines are instantaneous while strike-slip faults may have accommodated hundreds of kilometers movement over thousands of years, and (2) the assumption of plane horizontal stress is inappropriate when the crust has been significantly thinned or thickened.

A model pioneered by England and McKenzie (1982) and later improved upon by Houseman and England (1986) tackled both of these issues. By modelling the lithosphere as a thin sheet of, ultra-viscous, non-Newtonian fluid, time-dependant deformation can be computed by solving the Navier-Stokes equations of

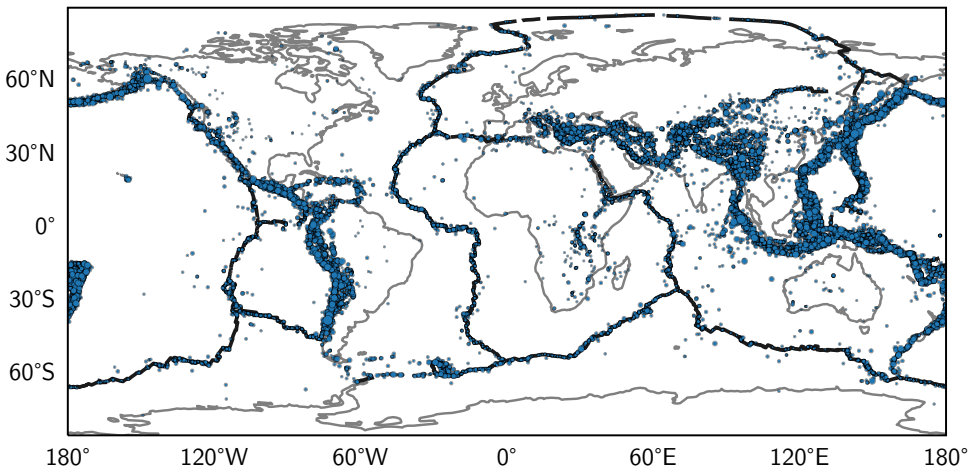


Figure 1.1: The ISC-GEM earthquake catalogue of events larger than M_w 5, from 1904–2014 (Bondár et al., 2015). While seismicity (and associated strain) is concentrated on plat boundaries, there is still significant intraplate seismicity.

fluid dynamics using finite-element methods. Additionally, body stresses are incorporated into the model which allow it to account for lateral variations in crustal thickness. The body stresses are the gradient of the gravitational potential energy of the crust, which is assumed to be in isostatic equilibrium. Despite the improvements made over previous models, this approach still had significant problems when applied to real geology (see, for example, the work of England and Houseman (1986)).

Firstly, a rheology had to be assumed, because at the time GNSS was not yet available for civilian use, and so no GNSS velocity networks existed with which strain rates and boundary conditions could be calculated. Therefore, the rheology of the model was constrained mainly by laboratory experiments. In recent years, modelling systems of intraplate deformation has improved significantly, most notably with large, dense networks of GNSS stations which measure velocities of continental plates. From these GNSS velocities, deviatoric strain rates can be calculated, which can then be compared to the deviatoric stresses produced by modelling the lithosphere as a fluid. Such measurements allow tight enough constraints on real systems that models can actually be solved for the rheology of the area (England, Houseman & Nocquet, 2016; Walters, England & Houseman, 2017), albeit a uniform rheology where deviations from the strain rate are treated as areas where the rheology differs from normal rather than a failing of the modelling technique itself.

Secondly, the earth is not flat (Pigafetta, 1536). For study areas which have a large latitudinal range, the simplification of flat geometry can lead to significant distortions between the model and reality. Recent studies have carried out some parts of the calculation on the surface of the sphere instead of approximating a flat earth, however in other parts of the calculation the flat earth simplification is still used.

1.3 Stable continental regions

Until relatively recently, the prevailing wisdom held that stable continental regions (SCRs) experienced little to no deformation, despite evidence of large seismic events within continental regions dating back centuries. One famous example is the New Madrid earthquakes of 1811–1812, which may have been as large as M_w 7.5, and their associated aftershocks (Hough et al., 2000). Over the past 50 years, recognition of such events has increased, due to the introduction and subsequent improvement of the global seismograph network; a recent notable example is the M_w 7.6 Bhuj earthquake in India on 26th January 2001 (Bodin & Horton, 2004). In geodynamic modelling, India is usually regarded as a rigid body indenting on Eurasia (England & Houseman, 1986), which makes this earthquake particularly noteworthy. Although far less studied, southern Africa has a similar history of relatively high magnitude seismic events. Examples include: 2017 M_w 6.5, Moiyabana, Botswana (Kolawole et al., 2017); 1969 M_w 6.3, Ceres-Tulbagh, South Africa (Green & Bloch, 1971); 1912

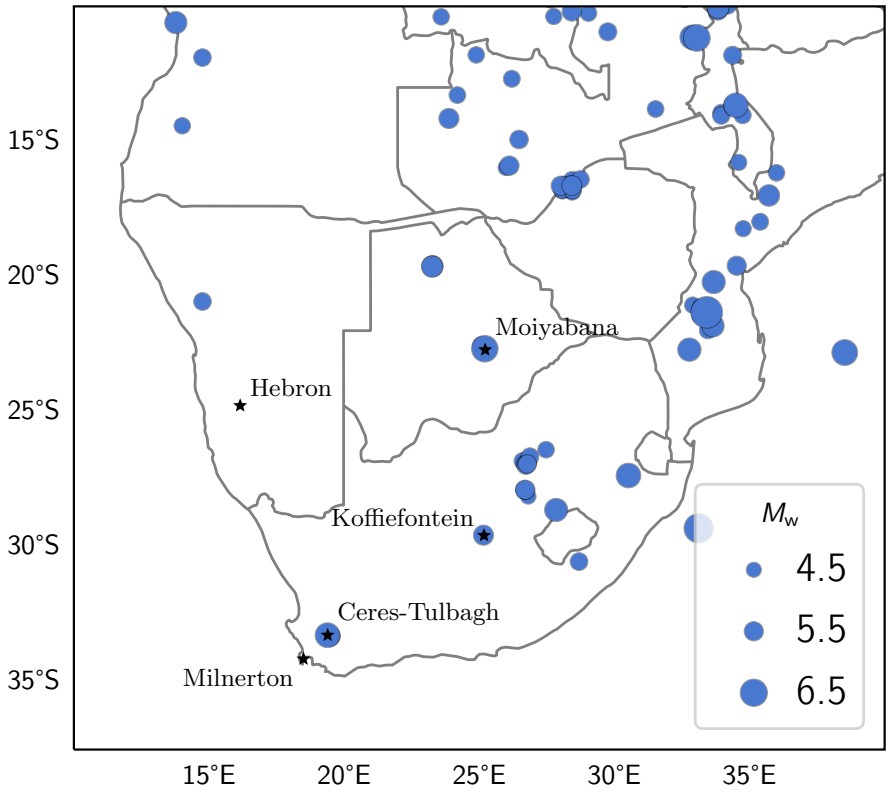


Figure 1.2: The ISC-GEM catalogue of seismicity in southern Africa from 1904–2014 (Bondár et al., 2015). Marked with stars are the locations of especially notable events.

M_w 6.2, Koffiefontein, South Africa (Strasser et al., 2015); 1976 M_w 5.8, Koffiefontein, South Africa (Strasser et al., 2015); 1809 M_w 5.0 to 5.9, Milnerton, South Africa (Von Buchenröder, 1830); and geologically recent reactivation of the Hebron fault, Namibia (White et al., 2009). The locations of these events are shown in Figure 1.2. Since earthquakes are events in which stress is released and converted into strain, this shows that there is indeed deformation in SCRs.

Deformation of tectonic plates is ultimately caused by forces acting upon them. The main sources of these forces are lateral variations in gravitational potential energy (Ghosh, Holt & Flesch, 2009; Rey, Vanderhaeghe & Teyssier, 2001), mantle traction (Morgan, 1972; Runcorn, 1962; Turcotte & Oxburgh, 1972), ridge push (Artyushkov, 1973; Hales, 1969; Jacoby, 1970; Lliboutry, 1969; Orowan, 1964), slab pull (Elsasser, 1971; Royden, 1993; Spence, 1987), and bending stresses within the plate (Craig, Copley & Jackson, 2014; Stauder, 1968). However, these are not the only factors that control the expression of strain; the rheology of the deforming rocks is also very important. For example, weakened zones in stable continental regions have significantly higher seismicity than continental cratons (Craig et al., 2011; Mooney, Ritsema & Hwang, 2012; Morley, 2010).

1.4 New contributions

This work aims to contribute towards some important questions regarding deformation in SCRs:

1. What controls the locations of earthquakes in SCRs?
2. How well do observations, in SCRs, of elastic strain accumulation (GNSS velocity data) and release (earthquakes) correlate with each other?

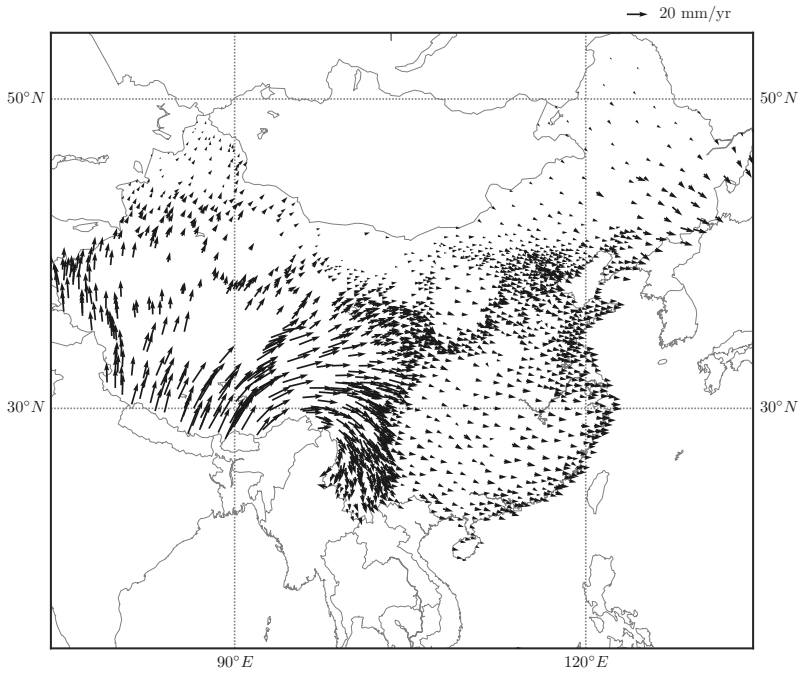


Figure 1.3: The Eurasian study area and GNSS velocities (Zhao et al., 2015). The cause of deformation in this region is the largely rigid Indian plate moving northwards into the Eurasian lithosphere. Coordinates are degrees latitude and longitude, and velocities are relative to stable Eurasia.

3. How well do the items above correlate with stresses and geological proxies for rheological variation?

Three bodies of work are presented in this dissertation which each contribute towards answering these questions by providing a basis for a future comparison of stresses and strain in SCR regions which is beyond the scope of this thesis.

1.4.1 Quantitative comparison of the effectiveness of common interpolation methods

The first contribution is a quantitative comparison of commonly used interpolation methods for the interpolation of GNSS data onto a regular grid. This is a necessary stage in converting such strain constraints into a format that can be used to compare to the thin-sheet modelling discussed in Section 1.2, and also has other applications such as identifying high strain rate regions for the purposes of seismic hazard investigations. Previous papers have introduced and used interpolation methods which combine weightings for data points dependent on a combination of their distance from the point to be interpolated to and their orientation in two-dimensional space (England, Houseman & Nocquet, 2016; Walters, England & Houseman, 2017; Zhao et al., 2015). Zhao et al. (2015) introduce two distance weighting schemes and two 2D orientation weighting schemes. For the distance weighting, they propose a weighting that has the functional form of a Gaussian curve and one with the functional form of an inverse square with respect to distance. For the 2D orientation, two more methods are proposed. The first method finds the Voronoi tessellation of the data points in Euclidean space, and then increases their weighting according to their Voronoi cell area (see Section 2.1.1). The second method calculates what proportion of 360° is taken up by each data point from the perspective of each interpolation point (see Section 2.1.2).

However, these authors only provide a qualitative justification for their choice of interpolation method. In this work a quantitative comparison is carried out of the six permutations of interpolation strategies: Gaussian-Voronoi, Gaussian-azimuthal, Gaussian only, inverse square-Voronoi, inverse square-azimuthal, and inverse square only. GNSS velocities from the India-Eurasia collision zone (Figure 1.3) will be used as the

data set with which these interpolation methods will be tested. The India-Eurasian collision zone is an example of rapidly deforming continental lithosphere, with relative velocities exceeding 20 mm/yr in some areas. This rapid deformation is driven by the Indian plate moving northwards into Eurasia, and is primarily expressed through numerous and large earthquakes (Dal Zilio et al., 2018). As with most large-scale GNSS data sets, the distribution of the receiving stations of this network is distinctly non-uniform; geodesists tend to put stations where it easy to physically get to. This presents a problem when trying to use the data for modelling, as modelling is vastly simplified if the data to which the model is compared is spaced evenly. Additionally, in this study the Voronoi tesselation and azimuthal weight determination will both be improved to be carried out on the surface of the sphere instead of computing them with a Euclidean geometry.

1.4.2 A method to calculate 2D deviatoric tensor fields under a spherical geometry

The second contribution extends the spherical earth model to the calculation of horizontal deviatoric stress or strain fields (Section 2.2), and comparing simple deviatoric tensor fields calculated with a Euclidean and spherical geometry (Section 3.2). This requires invoking the mathematical field of differential geometry, which is concerned with the study of smooth surfaces using (in this case) differential calculus. The proof of the techniques used to perform the necessary calculations will not be included in this work due to their length (and would ultimately be a rephrasing of textbooks such as Spivak (1999) which explain it more thoroughly). The use of these techniques in order to derive the results for the specific application of calculating deviatoric tensor fields on the surface of a sphere will be provided. Unlike the Universal Transverse Mercator (UTM) coordinate system, this method does not model the earth as an oblate spheroid, but instead as a perfect sphere. However, the effect of this is expected to be small relative to the distortion caused by moving too far away from the centre of a UTM zone over very large study areas, and in principle this technique could be extended to an oblate spheroid.

1.4.3 Tectonic stresses in southern Africa due to gravitational collapse

The third contribution is an analysis of the contribution of gravitational collapse to strain in southern Africa. In contrast to rapidly deforming continental regions (for example the east Asian region in Figure 1.3), southern Africa has relative velocities of less than 1.0 mm/yr (Hackl et al., 2011; Saria et al., 2013), and is considered an SCR (Calais et al., 2016), except for the East African Rift (EAR) system, which is viewed as a developing plate boundary (albeit a relatively slowly deforming one at its southern end, Saria et al., 2013).

Seismicity in SCRs have often been assumed to follow similar patterns to more active regions, where there is much greater data coverage due to thousands of recorded events per year, dense seismographic networks, and strain rates derived from high quality GNSS measurements. In these more active areas, strain is localised on long-lived active faults which builds up until a failure threshold is reached, and is then released in an earthquake. Active faults can therefore be identified through the analysis of instrumental and historic seismic catalogues, and even by directly measuring interseismic strain accumulation with GNSS sensors or satellite-based Interferometric Synthetic Aperture Radar (InSAR) sensors (Cavalié et al., 2008; Ruegg et al., 2009; Wright, Parsons & Fielding, 2001). Furthermore, southern Africa does not have a clear, singly predominant cause of stress (Figure 1.4). One candidate for a major cause of stress is the pair of rapidly diverging plate boundaries of the Mid-Atlantic Ridge to the east and the African-Antarctic ridge to the south west. Alone, these divergent boundaries would cause a compressive *ridge-push* stress (Forsyth & Uyeda, 1975). However several key observations do not fit with the simple picture of compressive stress from plate boundaries controlling strain. Most importantly, the majority of earthquakes and faults in the region record normal faulting, which implies horizontal extension and vertical shortening. Measurements from deformation of boreholes have been noted by Bird et al. (2006) to also not align with the strain that would be expected were deformation primarily driven by tectonic rifting, and has been called the Wegener stress anomaly. Southern Africa is unusually elevated compared to other stable continental regions, so it is possible that gravitational collapse is a driving stress of the observed extensional deformation.

There may also be significant differences in the character of seismicity between active regions and SCRs (Calais et al., 2016). In the rapidly deforming regions from which this model of seismicity is derived, the recurrence interval of earthquakes on these active faults is usually geologically short—tens of years to thousands

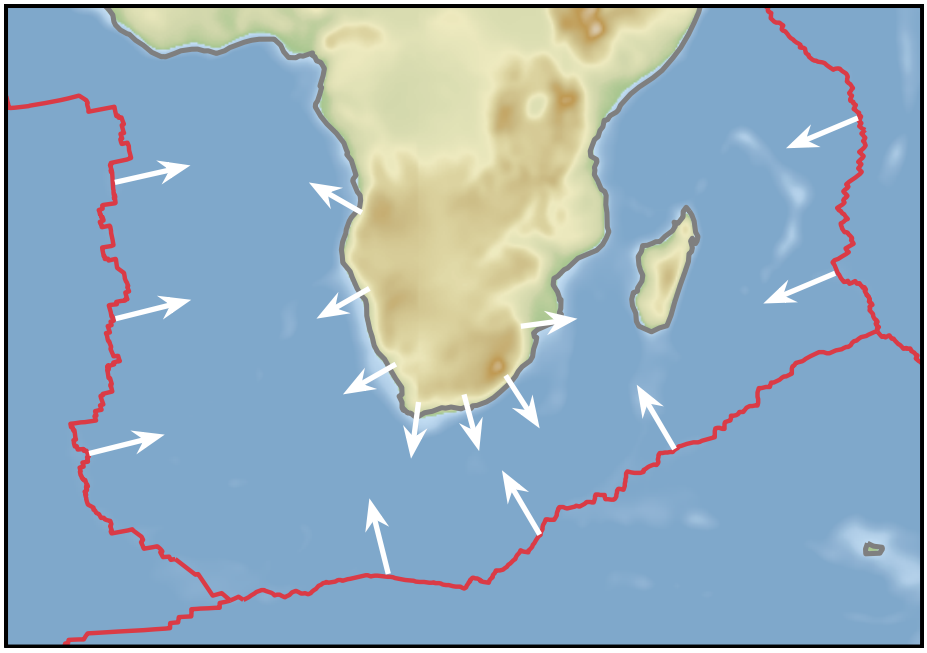


Figure 1.4: The divergent plate boundary system surrounding southern Africa, and southern African topography. White arrows show the general direction of stresses arising from the plate boundaries and from gravitational collapse due to the unusually high topography in the region ($\simeq 1000\,\mathrm{m}$ above sea level).

of years. This is not true of large earthquakes in SCRs, where many of the faults associated with higher magnitude earthquakes show no evidence for past events within the last few million years (Calais et al., 2016, 2010; Craig et al., 2016). Indeed, recent seismological studies in SCRs support the idea that the paradigm for earthquake mechanics may need to be reconsidered for continental interiors, challenging the conventional understanding of recurrence intervals (Calais et al., 2016; Liu & Stein, 2016). These works suggest that existing faults are critically stressed through very long term elastic strain accumulation throughout the entire region, and that these faults are then often triggered by small stress perturbations associated with other regional earthquakes, and even anthropogenic factors such as deep mining or hydrological changes. This model suggests that it may not be possible to identify the location of future events in SCR regions, as they are likely to be triggered by small stress changes, and are not especially likely to reoccur in regions which have been previously seismically active as the long-term strain reservoir may have been depleted there.

A starting point for considering which paradigms may be more useful for understanding the stress patterns and seismic hazard of southern Africa is to compare the geodetic constraints on internal strain within the stable plate, the moment release of instrumental earthquakes in the region, and the variation in stress expected from the topography of the region. While GNSS velocity measurements could be considered the premier method by which to assess deformation, not every region of interest has a high resolution GNSS network. In such areas (like southern Africa), it is especially important not to neglect other methods of analysing strain. Boreholes offer one alternative method of gaining insight into stress that occurs over (geologically) short periods of time. By measuring the change in shape over time of the cross section of a borehole, the principle horizontal stress axes can be determined. However, they have some drawbacks:

1. They contain no information about the magnitude of deformation, which is obviously a necessity to estimate strain rates and use them to calculate slip rates or to model the dynamics of deformation.

2. The time resolution of these purely directional strain measurements is limited by the frequency with which the boreholes are observed.

3. While the rock in which the borehole is drilled is comparatively isotropic (and so reflects the principle stress ellipsoid), on the crustal scale one would expect strain to be significantly modulated by the

orientation of preexisting weak zones.

Earthquakes offer another method of gaining insight into strain rates, but naturally using them comes with another unique set of challenges. While they offer both magnitude and direction information about strain, they are instantaneous events, so transient stress events may distort the macroscopic picture, the most notable example in southern Africa being mining (Brandt et al., 2005). Another challenge in southern Africa is that seismograph networks are sparse in many places (good coverage in South Africa, but poor coverage elsewhere). As previously mentioned, earthquakes in SCRs are sporadic and strongly controlled by rheology (Craig et al., 2011; Mooney, Ritsema & Hwang, 2012; Morley, 2010), which adds additional difficulty when using them to infer stress (McKenzie, 1969). It should be noted, however, that although the East African Rift is not generally considered part of an SCR, it is still strongly controlled by the location and orientation of the Proterozoic mobile belts (Craig et al., 2011). This means that the spatial distribution of strain will not be easy to identify through the measurement of interseismic strain accumulation which is likely to be below that detectable by GNSS surveys (Calais et al., 2010; Calais & Stein, 2009) and even instrumental, historic and paleoseismic earthquake records may be a poor indicator of future activity.

The deviatoric stress due to gravitational collapse will be modelled assuming Airy isostasy (Airy, 1855), using topography data from the ETOPO1 1 Arc-Minute Global Relief Model (Amante & Eakins, 2009). As this is a preliminary study, only the body stresses (i.e. gravitational collapse) will be considered in the stress model. The modelled body stresses will be compared to other data: quantitatively compared to the World Stress Map (WSM) database (Heidbach et al., 2018) and earthquake moment release calculated from the Bulletin of the International Seismological Centre (ISC) (Storchak et al., 2020, 2017); and qualitatively compared to known zones of structural weakness (Corner & Durrheim, 2018).

2 Methods

2.1 Comparison of interpolation methods

GNSS velocity networks can offer a uniquely detailed source of data about both small scales (kilometers to tens of kilometers), such as individual faulting events, and deformation of the earth's lithosphere over length scales of hundreds of kilometers (over which the lithosphere may be reasonably modelled as an ultra viscous fluid). A particular advantage of this measurement method is the potential for very high temporal resolution when operated continuously over many years. In the usual course of operation, a monument need only be visited as often as its battery needs replacing. On the other hand, large-scale GNSS networks are usually spaced irregularly, as the spatial locations of individual monuments is typically controlled by ease of access (e.g. next to major roads). In order to use such data in finite-element models, it needs to be regularised onto an evenly spaced grid using an interpolation method.

Formally, interpolation is a method of estimating intermediate values within the range of a set of discrete existing data points, with the condition that the values of existing data points are preserved. This definition is perfectly sufficient in one dimension where the range of a set of data is well defined; if $X = \{x_1, \ldots, x_n\}$ is the set of positions of the data points, then their range is the interval $[\min(X), \max(X)]$. Interpolation takes place within this range, and extrapolation takes place outside it. Now suppose the data points have positions defined in two coordinates, x and y. A naïve generalisation from the previous one dimensional example to this two dimensional system might stipulate that interpolation takes place in the interval $[(\min(X), \min(Y)), (\max(X), \max(Y))]$, where Y is the corresponding set to X for the y coordinate.

Figure 2.1 reveals this generalisation to be unsatisfactory. Consider the black rectangular box drawn around the data, which is the region of interpolation according to the definition given above. The area within the turquoise polygon is obviously a region of interpolation, and the region outside the black rectangle is obviously a region of extrapolation. However, the line between interpolation and extrapolation is blurred the region within the red rectangle. Towards the bottom of the red rectangle, it could justifiably be called interpolation, but towards the top of the rectangle, it is something closer to extrapolation. The turquoise polygon is *a* concave hull of the data points, and using this as the definition for the region of interpolation is tempting. Unfortunately, *the* concave hull of a set of points is poorly defined (Figueira, 2018), as its concavity must be specified by a parameter which determines how aggressively the algorithm eliminates concave regions. Choosing this parameter poorly could result in areas being excluded where, by eye, one might consider it obvious interpolation, or vice versa. In order to avoid this ambiguity, the range of interpolation is defined here as the convex hull of the data, but the shortfall of this definition should be kept in mind.

Commonly used interpolation methods—such as linear, polynomial, and spline interpolation—focus on interpolating the data between the nearest few points. Indeed, in the strict mathematical sense, interpolation methods force the generated function to pass precisely through every existing data point. This is problematic for this study's application for two reasons. First, current generation GNSS data can have high uncertainties in the measured velocities—so high in some cases that the uncertainty is larger than the measurement itself. Thus, allowing the generated function to vary from data is desired. Second, in some cases, only the long wavelength behaviour of the velocity data is of interest. When modelling large-scale deformation, local phenomena not supported by the lithosphere, such as slope instability, flexure due to hydrological loading, thermal expansions, or post-seismic deformation, are not of interest as they do not contribute to the overall, long-term, lithosphere-scale deformation.

An approach introduced by Shen et al. (2015), which has become popular in recent years (England, Houseman & Nocquet, 2016; Walters, England & Houseman, 2017), is to simultaneously smooth and interpolate the GNSS velocities. This smoothing consists of two parts; a proximity weighting, which down-weights distant points, and a density weighting, which down-weights closely packed points. The motivation for the former is straightforward: sources of stress are expressed as strain more strongly closer to the source of stress, so distant points predict strain rates less well than those which are closer. The motivation for the latter is the typically heterogeneous distribution of GNSS stations. As previously mentioned, GNSS stations are often placed where it is easy to get to, which results in many stations close together, and a few stations very isolated from the rest. In a case where an interpolation point is equidistant from an isolated data point and a cluster of densely packed points, a distance weighting alone would be significantly biased towards the

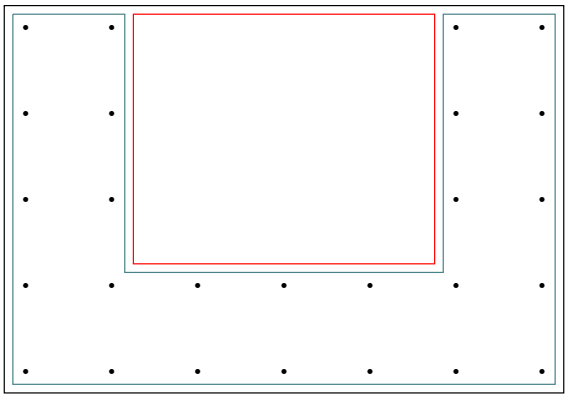


Figure 2.1: Possible definitions of interpolation for concave data sets. The area within the turquoise polygon is clearly a region of interpolation, and the area outside the black rectangle is clearly a region of extrapolation. The area within the red rectangle could be justifiably called either, highlighting the difficulty of defining interpolation for concave data sets.

dense points (Figure 2.2). Since the velocity field varies in space, the isolated point contains more information about the velocity field than an individual densely packed point, so the isolated point is up-weighted. It is worth noting that this is not interpolation in the strict mathematical sense, but this is the terminology that has become popular in the literature.

Unfortunately, this approach of combining a distance and a density weighting has been used in the literature without justification for its use instead of simpler and more common methods (for example a distance weighting alone). Furthermore, in their paper Shen et al. (2015) propose two methods by which to perform the distance weighting, and two methods by which to perform the density weighting, but do not provide more than a brief qualitative justification for why to choose one combination of methods over another. This work will quantitatively analyse the relative performance of the 4 combinations of weighting methods, as well as the two distance weightings on their own. First though, the implementation of these methods will be described in detail.

In this work, the index i will always refer to the i-th data point and the index j will always refer to the j-th interpolation point. For example, $\boldsymbol{v}_i$ means the velocity of the i-th data point. The proposed interpolation scheme is written as a sum over the values of the data $\boldsymbol{v}_i$ multiplied by a weighting coefficient G_{ij}

$$\boldsymbol{v}_j = \boldsymbol{v}(\boldsymbol{x}_j) = \sum_{i=1}^{n} \boldsymbol{v}(\boldsymbol{x}_i) G(\boldsymbol{x}_i, \boldsymbol{x}_j) = \boldsymbol{v}_i G_{ij}, \tag{2.1}$$

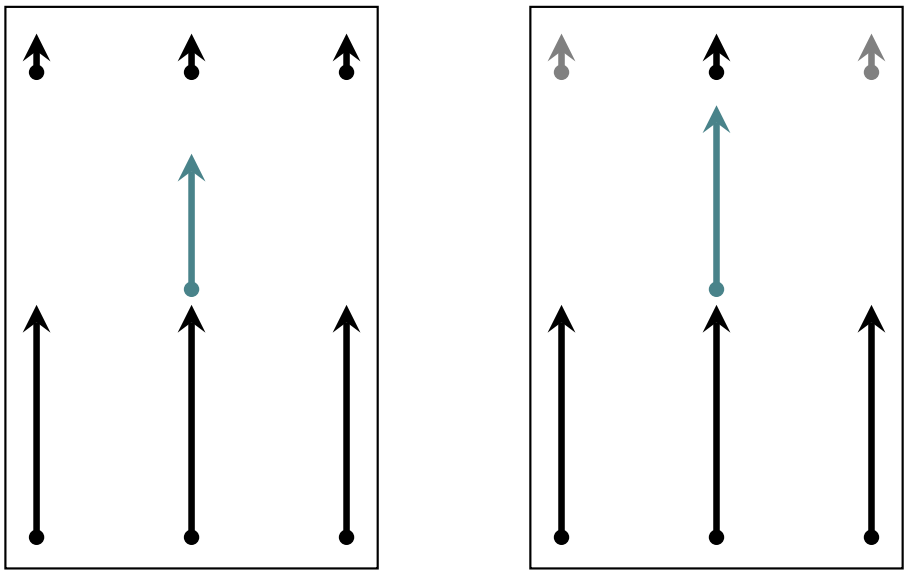
where $\boldsymbol{x}_j$ are the coordinates of the point to be interpolated to, $\boldsymbol{x}_i$ are the spatial locations of the data, and n is the total number of data points. The weighting coefficient is comprised of two components; the distance coefficient D which depends on the distance between the coordinates of the interpolation point and the data point, and the density coefficient A which depends on the density of the data points.

The performance of these two distance weighting schemes and two density weighting schemes introduced by Shen et al. (2015) will be examined. The two distance weighting schemes follow the form of a Gaussian curve and an inverse square curve (Figure 2.3), and the formulae for them are given by

$$D(\boldsymbol{x}_i, \boldsymbol{y}_j) = \exp\left(-\frac{{d_{ij}}^2}{{S_j}^2}\right), \tag{2.2}$$

$$D(\boldsymbol{x}_i, \boldsymbol{y}_j) = \left(1 + \frac{{d_{ij}}^2}{{S_j}^2}\right)^{-1}. \tag{2.3}$$

In both equations, $\boldsymbol{x}_i$ is the coordinate of the i-th data point, $\boldsymbol{y}_j$ is the coordinate of the j-th interpolation point, d_{ij} is the great circle distance between the two, and S_j is the so-called smoothing parameter, which controls how rapidly data points are down-weighted as their distance from the interpolation point increases.



(a) Even distribution.

(b) Uneven distribution.

Figure 2.2: The resulting interpolated velocity vector (turquoise) of a linearly decreasing vector field which arises when there is an even distribution of velocity measurements (2.2a) and an uneven distribution where the greyed-out vectors have not been included in the interpolation (2.2b).

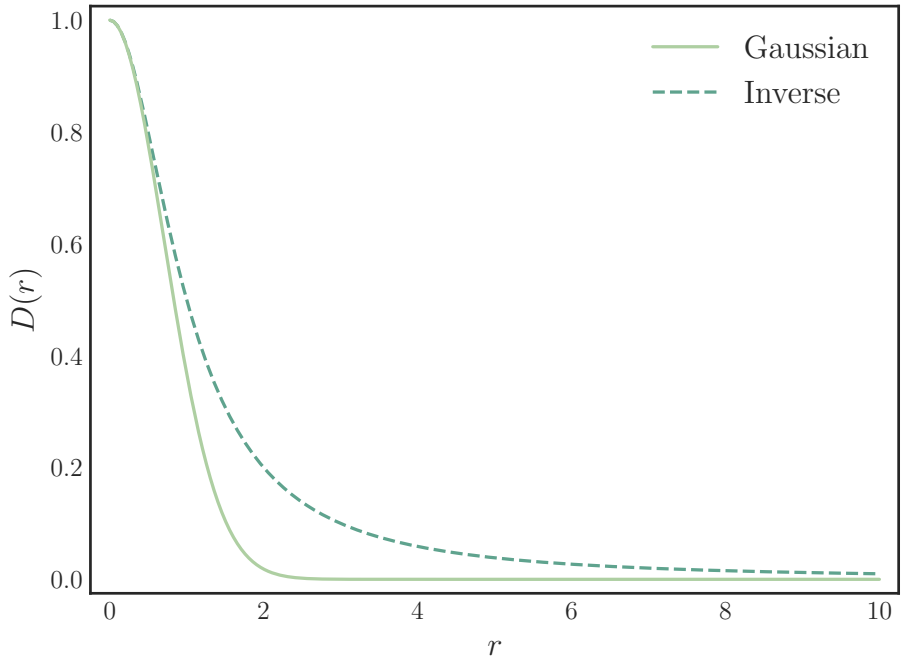


Figure 2.3: The functional forms of the proposed distance weightings (S=1).

The great circle distance between two points is defined as the shortest path between them on the surface of the sphere on which they lie. This may be calculated using the haversine formula, (Van Brummelen, 2013)

$$d_{ij} = 2r \arcsin\left(\sqrt{\left(1 - \frac{\cos(\varphi_j - \varphi_i)}{2}\right) + \sin(\varphi_i)\sin(\varphi_j)\left(1 - \frac{\cos(\lambda_j - \lambda_i)}{2}\right)}\right), \tag{2.4}$$

which has been modified to use spherical coordinates (r, λ, φ) instead of latitude and longitude (φ differs by $\pi/2$ between the two). By convention, the longitude is denoted by λ, which has been maintained to avoid confusion with the azimuthal span θ.

The smoothing parameter S_j introduced in (2.2) and (2.3) requires special attention. The density weights A_{ij} are fixed by the geometry of the problem (i.e. the spatial locations of the data points and the interpolation points), so the only degrees of freedom left in the problem are the S_j. Shen et al. (2015) suggest a method by which a set of smoothing parameters $S = \{S_1, S_2, \ldots, S_n\}$ could be chosen. Let the dimensionless quantity W_j be the sum of the weighting coefficients for an interpolation point such that

$$W_j = \sum_{i=1}^{n} Gij = \sum_{i=1}^{n} D_{ij} A_{ij}. \tag{2.5}$$

Next, let W' be some fixed value. It is then possible to use a root finding algorithm at each interpolation point to find the value S'_j such that $W_j = W'$. Note that by fixing W' and finding the appropriate set S' to match, an overall trend on the values of the S'_j is induced. That is, when the distribution of data near the j-th interpolation point is very dense, S'_j will be smaller than when the data distribution is sparse. It should be straightforward to see that as the chosen value for W' is increased, so does the median value $\bar{S}$ of the smoothing parameters.

Actually performing the root finding requires some care. The first problem is that the functions suggested for D_{ij} are not well-behaved at $S_j = 0$. Secondly, while W' may be arbitrarily small, it is not guaranteed that there exists an S'_j satisfies $W_j = W'$, so the root-finding algorithm cannot be guaranteed to converge. Additionally, there is no strict upper bound on S_j, but in practice choosing something several orders of magnitude higher than the target value of $\bar{S}$ is sufficient. Combined with an adequately small lower bound, this allows the use of fast and stable *bounded* root-finding algorithms such as the classic Brent's method (Brent, 1973), which is chosen here. By this process, the problem of choosing n parameters (the S'_j) has been reduced to choosing just one parameter, namely W'. For some applications, for example the kind of lithosphere deformation modelling carried out by England, Houseman and Nocquet (2016), a larger value of W' would be preferred. This would correspond to a larger smoothing distance. On the other hand, if one was interested in short length-scale applications strain across an individual fault, a small value of W' would be better. Nevertheless, a choice of an exact number for W' and corresponding value of the median value of S still needs to be made. The approach of Shen et al. (2015) leaves the choice to the user (in other words, eyeballing it). While it is possible in principle to run the finite-element deformation model multiple times with various values of W' and minimise the error in the strain rate output, to the author's knowledge this has not been done in the literature, and is beyond the scope of this work.

The motivation for the density weighting is more subtle—the value of the information carried by a data point increases as the density of data points in its local area decreases. To illustrate this, consider the two cases in Figure 2.2, where the the vectors in black sample a velocity field of uniform direction and linearly decreasing magnitude from South to North. In both cases, the vector coloured in turquoise is interpolated from the vectors coloured in black using the Gaussian method described previously. In Figure 2.2a, the interpolation method correctly predicts the vector field at the intermediate location. However in Figure 2.2b, the magnitude of the interpolated vector is strongly biased towards the data points to the South.

Two schemes are proposed to account for this effect, the Voronoi method and the azimuthal method. The weights in the former method scale depending on the local density of measurements (and are independent of the interpolation point), while the weights of the latter depend only on the azimuthal coverage (in two-dimensional (2D) polar coordinates) of each data point. This means that the two methods are essentially 2D and 1D in their calculation of the density weighting respectively.

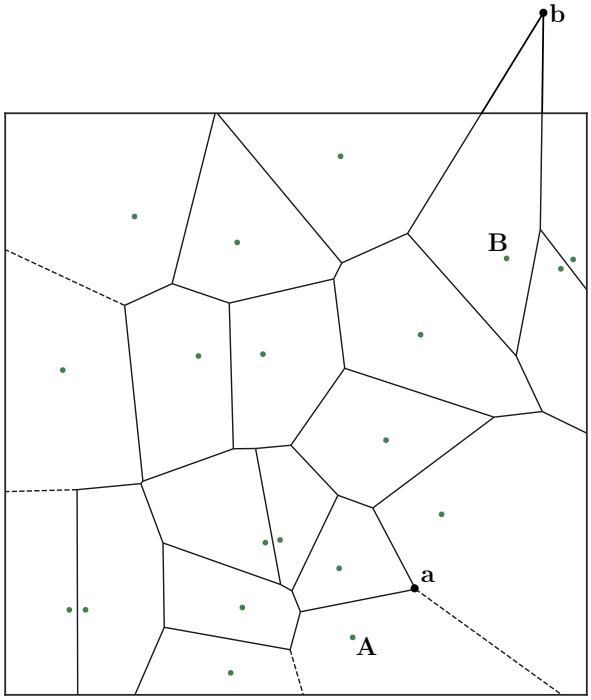


Figure 2.4: The Voronoi tessellation of 20 points on $\mathbb{R}^2$ (Euclidean metric). Solid lines indicate finite length Voronoi edges. Dashed lines indicate infinitely long Voronoi edges, which will never intersect with another Voronoi edge (e.g. the vertex **a**). Some finite-length Voronoi edges are artificially shortened by the border around the figure, and will converge at some point outside border (e.g. the vertex **b**).

2.1.1 Voronoi density weighting

A Voronoi tessellation (Figure 2.4) is a partitioning of a space into regions based on the distances from a set of *generators* (points within the space). The set of all points which are closer to generator x_i than any other generator defines the generator's *Voronoi cell*. The Voronoi tessellation of a set of points is dual to its Delaunay triangulation (Weisstein, 2015b), a special triangulation such that none of the generating points is inside the circumcircle of any triangle in the triangulation. Each data point is then weighted by the area of its Voronoi cell. Previously, the Voronoi tessellation was carried out in a Euclidian geometry (Shen et al., 2015), but in this work, it will instead be done on the surface of a sphere. Directly computing the Voronoi tessellation on the surface of the sphere is challenging, but the Delaunay triangulation of points on the surface of the sphere is equivalent to the (3D) convex hull, which is easily computed.

Unfortunately, there are two problems with Voronoi weighting. The first is that, while calculating the Voronoi tessellation on Euclidean geometry is relatively straightforward, these methods cannot be easily generalised to other surfaces in three-dimensional space. Fortunately the surface of the earth is well approximated by a sphere, and the Voronoi tessellation on a sphere can be calculated in a fairly straightforward way. First, the 3D convex hull of the generators is calculated—this is the same as their Delaunay triangulation on the sphere (Caroli et al., 2010). This is converted into a 3D Delaunay tetrahedralisation by including the centre of the sphere as a fourth vertex of each symplex of the convex hull. The circumcenters of all tetrahedra in the system are calculated and projected to the surface of the sphere, producing the Voronoi vertices. Neighbour information from the Delaunay tetrahedralisation is then used to order the Voronoi regions' vertices around each generator.

Since the Voronoi edges are great circles, the Voronoi cell surface areas a_i can be computed by decomposing them into spherical triangles, calculating the surface area of each, and adding them together. L'Huilier's

theorem states

$$\tan\left(\frac{1}{4}E\right) = \sqrt{\tan\left(\frac{1}{2}s\right)\tan\left[\frac{1}{2}(s-b)\right]\tan\left[\frac{1}{2}(s-c)\right]\tan\left[\frac{1}{2}(s-d)\right]}, \tag{2.6}$$

where E is the spherical excess, which is the surface area of a spherical triangle (Weisstein, 2015a). The semiperimeter s is defined $s = \frac{1}{2}(b+c+d)$ where b, c, and d are the side-lengths of the spherical triangle. The density weighting A_{ij} for the i-th data point is then

$$A_{ij} = \frac{nE_i}{\sum_{k=1}^{n} E_k}, \tag{2.7}$$

where n is the total number of data points. Note that the index j is only present for bookkeeping purposes—the weighting does not actually depend on the positions of the interpolation points j, since the Voronoi cell areas are fixed entirely by the spatial distribution of data points.

The second problem is that at the edge of the network of data points Voronoi cell areas are greatly exaggerated. This is easier to understand with a flat coordinate system (Figure 2.4). Towards the edge of the network, all the Voronoi cell borders eventually converge on dashed lines, and these dashed lines extend to infinity (see the vertex labeled **a**). This is because there are no further generating points to interfere with the line that is equidistant from these points on the exterior. In turn, this means the corresponding Voronoi cell areas are infinite (see the cell labeled **A**). On the surface of the sphere, it is impossible for these lines to extend to infinity (they would eventually converge at some point on the opposite side of the sphere), however the Voronoi cell areas are still much larger than they would be if they were in the interior of the network.

There are several options of various complexity to solve this problem. One option is to draw a border around the network and use it as the missing edge (or edges) of Voronoi cells. However the location of this edge is entirely arbitrary, and not consistent between points on the edge of the network. Another option is to disregard all the data points which have the pseudo-infinite edges as part of their Voronoi cell border, creating a smaller network where every data point has a well defined Voronoi cell area. For data sets with only tens of points removing the exterior data points could result in the network being cut in number by half or more. Additionally, this still leaves some cells with unrealistically large areas, even if they are not pseudo-infinite (see the cell labelled **B**). Shen et al. (2015) approximate the Voronoi cell area for these problematic data points by calculating the area of a spherical cap with radius equal to half the median distance to neighbouring points (for example the dashed circle around **C** in Figure 2.5).

However, some way to determine which points have an unreasonable Voronoi area is still needed. Shen et al. (2015) suggest that Voronoi cells with an area exceeding twice the area of the spherical cap have their area replaced by that of the spherical cap. This solution is unsatisfactory for the following reason: if the spherical cap is a good approximation for what the Voronoi area would have been had the data point been on the interior of the network, it does not make sense to replace the area only if it is twice as big as it is estimated to be! Since this is an approximation, it is preferable that at the very least it usually underestimate the weighting (i.e., err on the side of caution) rather than overestimate it by as much as a factor of two (see the point **D** in Figure 2.5). Remember, the Voronoi cell area is itself only an estimator of the local density of data points.

Unfortunately, simply altering the condition to be a limit equal to the spherical cap is insufficient, as this incorrectly flags many points ($> 50\,\%$) which can be seen by eye to be on the interior of the network of data points (see the point **E** in Figure 2.5). Instead, points which are on the edge of the network are manually identified, and the spherical cap condition is only applied to these points. However, instead of replacement when the Voronoi area exceeds twice the area of the spherical cap, the Voronoi area is replaced whenever it is higher than one multiple of the area of the spherical cap. While it feels in some way less satisfactory than a completely automated solution (and certainly more work), it avoids both the problem of over-weighting exterior points and accidentally flagging interior points.

2.1.2 Azimuthal density weighting

The second density weighting method which will be investigated is the azimuthal span of the data. At the j-th interpolation site, the azimuthal span of a data point θ_{ij} for the i-th data point is

$$\theta_{ij} = \arccos\left(\hat{\boldsymbol{r}}_{i+1,j} \cdot \hat{\boldsymbol{r}}_{i-1,j}\right), \tag{2.8}$$

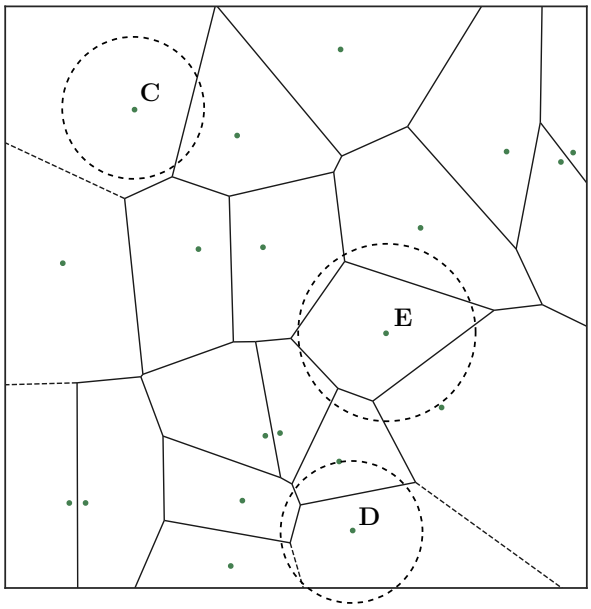


Figure 2.5: The Voronoi tessellation of 20 points on $\mathbb{R}^2$ (Euclidean metric). Solid lines indicate finite length Voronoi edges. Dashed lines indicate infinitely long Voronoi edges. Points on the edge of a network with unrealistically large Voronoi cells should have their area weightings replaced with a better estimate (for example, the point **C**). The point **D** is an example of where the proposed method of Shen et al. (2015) for dealing with edge points is unsatisfactory (the area of the Voronoi cell is unrealistically large, but the area of the circle would *not* be used). The point **E** is an example of where changing the condition would incorrectly replace a Voronoi area with the area of a circle.

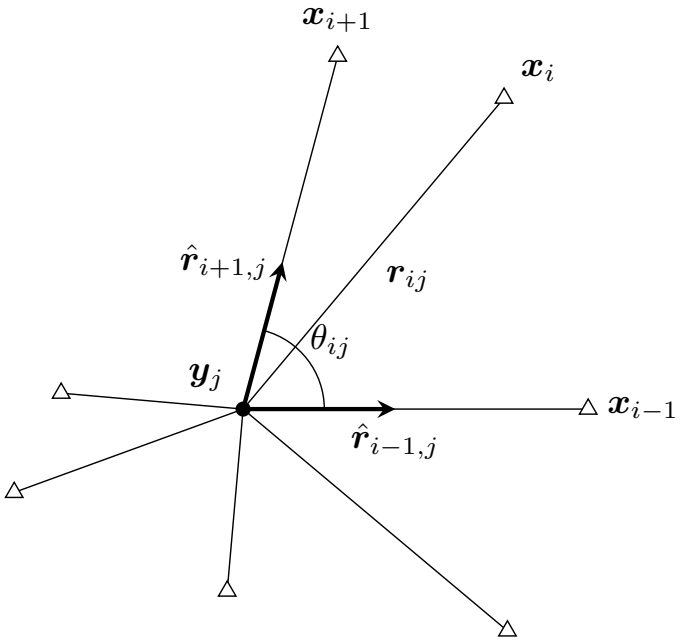


Figure 2.6: The azimuthal span can be used to quantify the spatial density of data points. The azimuthal span (θ_{ij}) of the i-th data point for the j-th interpolation point. The angle used for the azimuthal weighting of a data point is $\theta_{ij}/2$.

where $\hat{\boldsymbol{r}}_{i,j}$ is the unit vector pointing from the j-th interpolation point to the i-th data point. Note that the data points are re-indexed for each interpolation point by increasing azimuthal angle. In other words, the azimuthal span of $\boldsymbol{x}_i$ is the angle between the data points which are *azimuthally adjacent* to it (Figure 2.6). The azimuthal density weighting for a data point is then

$$A_{ij} = \frac{n\theta_{ij}}{4\pi}. \tag{2.9}$$

Note that in general the azimuthal density weight of a data point is not the same for different interpolation points. Imagine standing at the interpolation point $\boldsymbol{y}_1$ in Figure 2.7, and that the data point $\boldsymbol{x}_1$ has an azimuthal span of $\theta_{\boldsymbol{x}_1\boldsymbol{y}_1} = 1^\circ$ with $\boldsymbol{x}_0$ and $\boldsymbol{x}_2$. If you move 100 m Southeast to $\boldsymbol{y}_2$, your view of the relative positions of the three data points will change, which in turn changes the angles between them, and thus the azimuthal span will change. Indeed, if you move far enough, the azimuthal ordering of the data points will change too. In contrast, the Voronoi density weightings are the same for every interpolation point, because the Voronoi cell of a data point depends only absolute positions of all the data points.

The azimuthal span method has a significant drawback. At the edge of a data network, a large proportion of the 360 deg view will be empty of data points (Figure 2.8). This means that two data points will have an enormous azimuthal span compared to the rest of the data points, distorting the interpolation scheme. No acknowledgement of this problem is given in the existing literature, so in this work the azimuthal span will be tested as presented by Shen et al. (2015).

2.1.3 Cross-validation error

Three methods are used to assess the effectiveness of these interpolation schemes. The first is an adaptation of the jackknife resampling method. For each combination of distance weighting and density weighting, the velocity of each data point is predicted by using the rest of the data set, for a range of values of $\bar{S}$. The

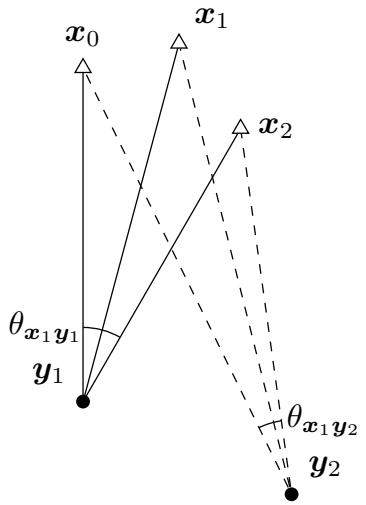


Figure 2.7: Unlike the Voronoi method, with the azimuthal span method the density weighting of a data point also depends on the position of the interpolation point. Moving an interpolation point changes the azimuthal span of a data point. $\theta_{\boldsymbol{x}_1\boldsymbol{y}_1} \neq \theta_{\boldsymbol{x}_1\boldsymbol{y}_2}$.

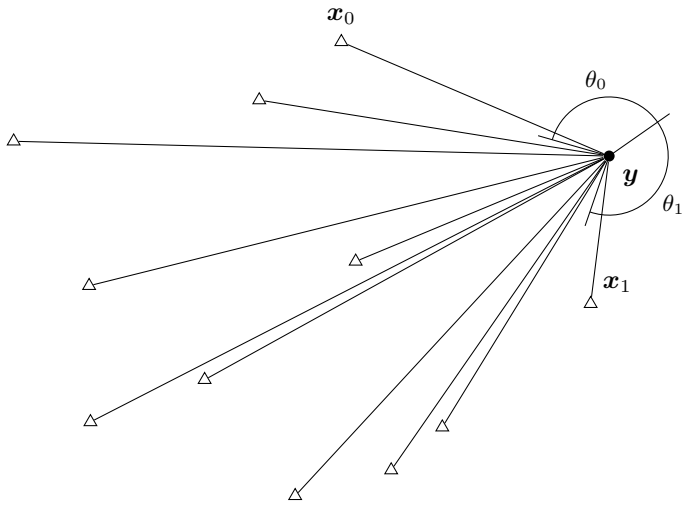


Figure 2.8: A drawback of the azimuthal span method is that for the interpolation point $\boldsymbol{y}$, the azimuthal span of the data points $\boldsymbol{x}_0$ and $\boldsymbol{x}_1$ are far larger than for the other data points, because the interpolation point is outside the the convex hull of the data points.

predicted value and the measured value are compared by calculating the root mean square error (RMSE),

$$RMSE(\boldsymbol{v}) = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(\boldsymbol{y}_i - \boldsymbol{x}_i)^2}, \tag{2.10}$$

where $\boldsymbol{x}_i$ is the i-th measured velocity and $\boldsymbol{y}_i$ is the i-th interpolated velocity at the same spatial location as $\boldsymbol{x}_i$. The RMSE is chosen as an estimator because of its familiarity, simplicity, and its sensitivity to large outliers.

The jackknife approach has been used in other fields to analyse the performance of interpolation methods (Hofstra et al., 2008), and it has the advantage of using samples from a real GNSS velocity field and a real distribution of data points. On the other hand, it is not a perfect analogy to how interpolation methods would be used in the real world; instead of interpolating the velocity field onto a regular grid, it is interpolating it back onto the irregularly spaced original data set. This causes the jackknife approach to be more suited to data with short-range correlation, as the interpolation scheme is attempting to use the data to predict the data itself. This presumably makes it more suited to evaluating interpolation schemes for the purposes of short wavelength geophysical modelling (for example, the strain rate across a single fault) rather than long wavelength modelling (deformation over large subsections of tectonic plates).

2.1.4 Known-field error

To combat the aforementioned disadvantages, in the second analysis method an artificial velocity field is created using a known equation that *resembles* a real velocity field (Figure 2.9a). These equations are chosen due to their first-order similarity to the GNSS velocity data observed in India-Eurasia collision zone. It is then sampled using a realistic distribution of data points taken from the spatial distribution of a data from Zhao et al. (2015) (Figure 2.9b). The equations for this vector field are

$$\begin{aligned} v_\varphi &= 60\varphi' \exp(-10\varphi'^2) + \frac{12}{\exp(\varphi'/3)+1} - 2.2, \\ v_\lambda &= 8\lambda' \exp(-\varphi'/1.3) + 4\exp(1.5\lambda' + 1.2\varphi')\left(40\varphi' \exp(-40\varphi') - 10\varphi'^2 \exp(-10\varphi^2)\right), \end{aligned} \tag{2.11}$$

where $\varphi' \equiv \varphi - \frac{7}{4}$ and $\lambda' \equiv \lambda - 1$. The particular form of this vector field has been chosen to vary smoothly over the length scales of around a hundred kilometers, as would be expected for the results of thin sheet modelling (discussed in Section 1.2). The artificial data points are then interpolated onto a regular grid using each interpolation method, and compared to the value of the velocity field at each grid point by calculating the RMSE over a range of values of $\bar{S}$,

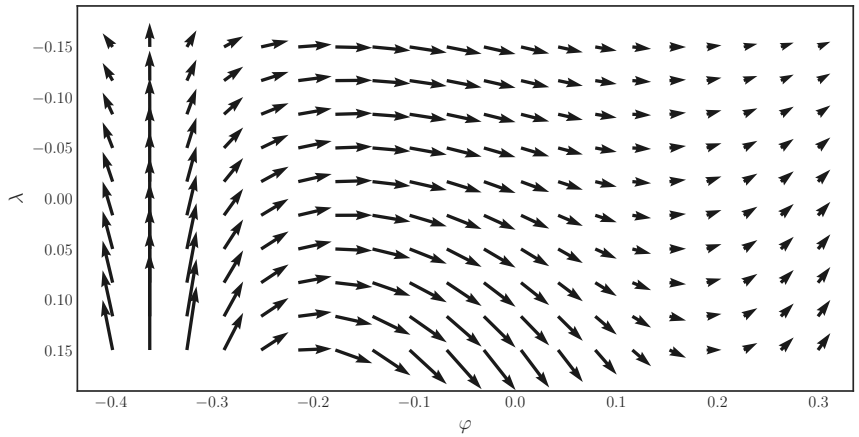
$$RMSE(\boldsymbol{v}) = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(\boldsymbol{y}_i - \boldsymbol{x}_i)^2}, \tag{2.12}$$

where $\boldsymbol{x}_i$ is the velocity of the artificial vector field (2.11) at an interpolation point, and $\boldsymbol{y}_i$ is the velocity at the same point interpolated using the data in Figure 2.9b. The chosen interpolation points are a grid of 20×10 evenly spaced points in the $\varphi\lambda$-plane, with $\varphi \in [1.40, 2.05]$ and $\lambda \in [0.85, 1.15]$. This corresponds to a spacing between grid points of 234 km.

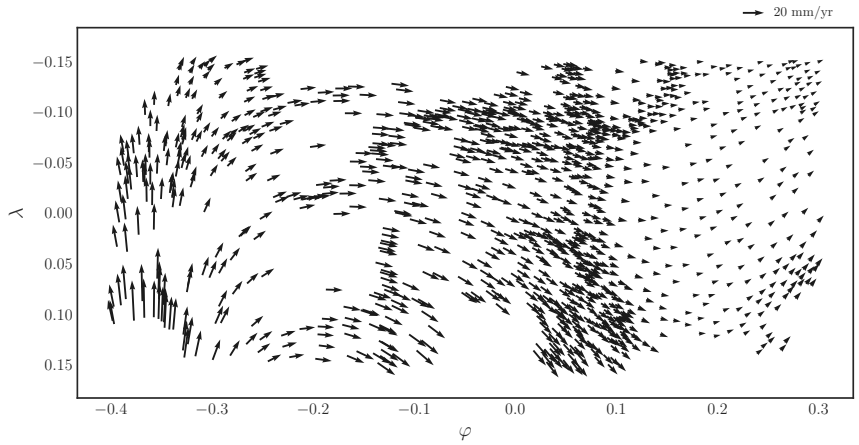
This approach has the advantage of much more closely approximating the process of interpolation as would be used in a real modelling problem. However, it is obviously not derived from a real set of GNSS velocity data. Furthermore, unlike real data, this artificial data is a perfect sampling of the underlying velocity field, with no measurement error.

2.1.5 Noise sensitivity error

Finally, the third method introduces random Gaussian noise to the values of the known velocity field. This is to test the sensitivity of the interpolation methods to source data which does not perfectly sample the true velocity field. This should more closely model a real data set, while still being able to compare the



(a) The known vector field defined by (2.11), sampled on a regular grid for visualisation.



(b) The locations at which the known vector field was sampled to generate data points.

Figure 2.9: The known vector field and corresponding artificial data used for the known field and noise sensitivity tests.

interpolation result to known velocity field. Of the three testing regimes in this work, this one has the fewest drawbacks. While there is no guarantee a real data set has measurement error which follows a Gaussian distribution, calculations of measurement error almost universally assume a Gaussian distribution, so this is a reasonable assumption for this thesis.

The random noise is added to the velocities with a Gaussian distribution with standard deviation equal to the mean standard error of the GNSS data from Zhao et al. (2015). The data is then interpolated to the same regular grid as in the previous test, and compared to the true value of the artificial velocity field at each grid point, again by calculating the RMSE over a range of values of $\bar{S}$. This process is repeated 50 times for each value of $\bar{S}$, and the arithmetic mean of the RMSE at each value is taken as the final value.

2.2 Calculation of 2D deviatoric tensor fields on a spherical earth

Previous works (England, Houseman & Nocquet, 2016; Walters, England & Houseman, 2017) have assumed a flat earth when calculating horizontal deviatoric stress and strain

$$\dot{\varepsilon}_{ij} = \frac{1}{2}\left(\frac{\partial f_i}{\partial x^j} + \frac{\partial f_j}{\partial x^i}\right), \tag{2.13}$$

where i and j denote arbitrary basis vectors of a coordinate system (in this case the basis vectors to choose from are eastings and northings), and $\frac{\partial f_i}{\partial x^j}$ means the derivative in the j-th direction of the i-th component of a vector field $\mathbf{f}$. However, in typical study areas, the true length of a degree of longitude can change significantly. Consider the study area in Walters, England and Houseman (2017), which ranges from 26°N to 44°N and 40°E to 62°E. At 44°N, one degree of longitude has a length on the surface of the (spherical) earth of 80 206.16 m, while at 26°N a degree of longitude has length 100 117.68 m, an increase of 24.8 %. Because Earth is an oblate spheroid, the length of a degree of latitude also changes with latitude, however the change is much smaller; 111 112.21 m at 44°N and 110 787.98 m at 26°N, an increase of just 0.29 %.

This provides strong motivation to account for the curvature of a spherical earth (which causes the change in the length of a degree of longitude) in as many places as possible. The change caused by the earth being an *oblate spheroid* is smaller by two orders of magnitude, and is beyond the scope of this thesis.

To calculate the equation for deviatoric stress on the surface of a sphere, basic differential geometry techniques are required. Differential geometry is the branch of mathematics concerned with applying the techniques of differential calculus (among others) to study curves, surfaces, and manifolds. However, in order to follow the mathematics in the following section, a quick explanation of Einstein summation notation is necessary. If there are matching indices on the same side of an equation, it indicates the presence of a sum over that index,

$$a^i b_i = \sum_i a^i b_i. \tag{2.14}$$

If an index is free (that is, it appears only once on one side of the equation) it must also appear once (and only once) on the other side of the equation. Free indices represent an entry in a tensor. This could be a number in a matrix, or even another tensor.

As an aside, in this section, some indices in (2.14) are superscript and some subscript. In Einstein notation, these are informally referred to as upper and lower indices, and represent *contravariant* and *covariant* vectors respectively. In general, sums must be performed over one upper and one lower index.

First, the metric tensor g is introduced. The metric tensor is the generalisation of the Euclidean dot product to vector spaces that are defined by non-flat geometries. Indeed, the metric tensor *defines* the geometry of a space. The metric tensor is symmetric, which means it is invertible, which in turn means it can be inserted, along with its inverse, to perform a sum over two upper or two lower indices. For 3-dimensional Cartesian coordinates $\mathbb{R}^3$, the metric tensor is

$$g_{ij} = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix}_{ij} = I_{ij}. \tag{2.15}$$

The inverse of g_{ij} is $g^{ij} = I^{ij}$. For the surface of the sphere S^2, the metric and inverse metric tensors are

$$g_{ij} = \begin{pmatrix} r^2 & 0 \\ 0 & r^2 \sin^2\theta \end{pmatrix}_{ij}, \qquad g^{ij} = \begin{pmatrix} \frac{1}{r^2} & 0 \\ 0 & \frac{1}{r^2 \sin^2\theta} \end{pmatrix}^{ij}. \tag{2.16}$$

Next the *Christoffel symbols* $\Gamma^i{}_{jk}$ are introduced. Christoffel symbols make up an array of variables that encode how coordinates change with respect to each other. In the case of S^2 the surface of a sphere, parameterised by colatitude and longitude, the length of one degree of colatitude depends on the colatitude itself. Inspecting the metric tensor, this dependence is seen to be encoded in the bottom right entry $g_{\phi\phi} = r^2 \sin^2\theta$. From (5) in Chapter 4, Part D in Spivak (1999), the Christoffel symbols can be defined in terms of the metric tensor

$$\Gamma^i{}_{kj} = \frac{1}{2} g^{ih} \left(\frac{\partial g_{hk}}{\partial x^j} + \frac{\partial g_{hj}}{\partial x^k} - \frac{\partial g_{kj}}{\partial x^h} \right). \tag{2.17}$$

Finally, the *covariant derivative* is introduced. The covariant derivative is the generalisation of the gradient from scalar fields to vector (or tensor) fields. From (4.6.4) in (Weinberg, 1972), the covariant derivative of a contravariant tensor f^i is

$$f^i{}_{;j} = \frac{\partial f^i}{\partial x^j} + f^k \Gamma^i{}_{kj}. \tag{2.18}$$

The conventional partial derivatives in (2.13) are now replaced with covariant derivatives (and then some manipulation of Einstein notation) to obtain

$$\dot{\varepsilon}^j{}_j = \frac{1}{2} \left(f^i{}_{;j} + g^{i\beta} g_{j\alpha} f^\alpha{}_{;\beta} \right). \tag{2.19}$$

Before tackling the deviatoric stress on the surface of the sphere, it should be confirmed that equation (2.19) reduces to equation (2.13) by using the metric tensor $g = I_2$ for $\mathbb{R}^2$. Since all the components of g for $\mathbb{R}^2$ are constant, all the derivative terms in equation (2.17) will be 0, and so the Christoffel symbols for $\mathbb{R}^2$ are 0, and therefore the covariant derivative on $\mathbb{R}^2$ is the same as the conventional partial derivative.

Computing the Christoffel symbols for S^2 is not difficult, but even with only two coordinates carrying out the algebra is a lengthy process, and not worth including here. The eight Christoffel symbols are

$$\Gamma^\theta{}_{ij} = \begin{pmatrix} 0 & 0 \\ 0 & -\sin\theta\cos\theta \end{pmatrix}_{ij}, \qquad \Gamma^\phi{}_{ij} = \begin{pmatrix} 0 & \frac{\cos\theta}{\sin\theta} \\ \frac{\cos\theta}{\sin\theta} & 0 \end{pmatrix}_{ij}. \tag{2.20}$$

Thus, the four components of the deviatoric stress tensor are given by

$$\begin{aligned} \dot{\varepsilon}^\theta{}_\theta &= \frac{\partial f^\theta}{\partial x^\theta}, & \dot{\varepsilon}^\theta{}_\phi &= \frac{1}{2} \left(\frac{\partial f^\theta}{\partial x^\phi} - f^\phi \cos\theta \sin\theta + \frac{\partial f^\phi}{\partial x^\phi} + f^\theta \frac{\cos\theta}{\sin\theta} \right), \\ \dot{\varepsilon}^\phi{}_\theta &= \dot{\varepsilon}^\theta{}_\phi, & \dot{\varepsilon}^\phi{}_\phi &= \frac{\partial f^\phi}{\partial x^\phi} + f^\theta \frac{cos\theta}{\sin\theta}. \end{aligned} \tag{2.21}$$

Once the equations in (2.21) have been calculated, little effort is needed to use them in place of (2.13). The equations (2.21) expect values in units of radians, and so any values in terms of distance must first be converted (for example a velocity field in mm/yr should be converted to rad s^{-1}). The conversion to angular units for northward vectors is

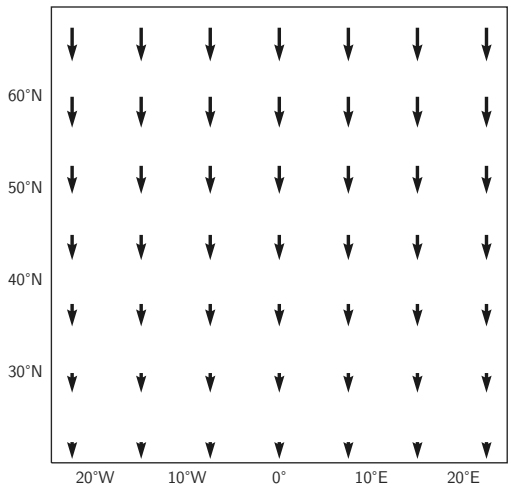
$$f^\theta = \frac{f^y}{r}, \tag{2.22}$$

and for eastward vectors

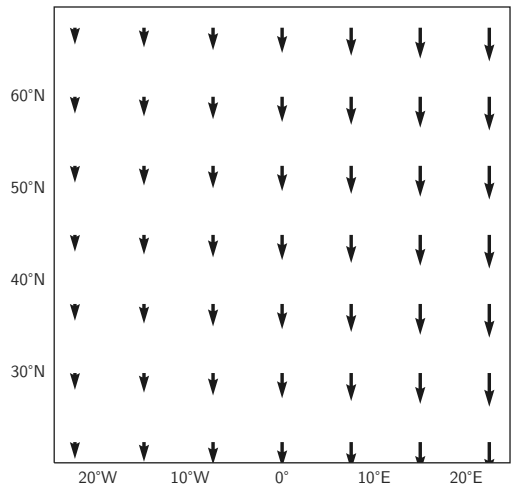
$$f^\phi = \frac{f^x}{r \sin(\theta)}, \tag{2.23}$$

where f^y and f^x are northings and southings, r is the radius of the earth, and θ is the angle between the location of the vector and the north pole.

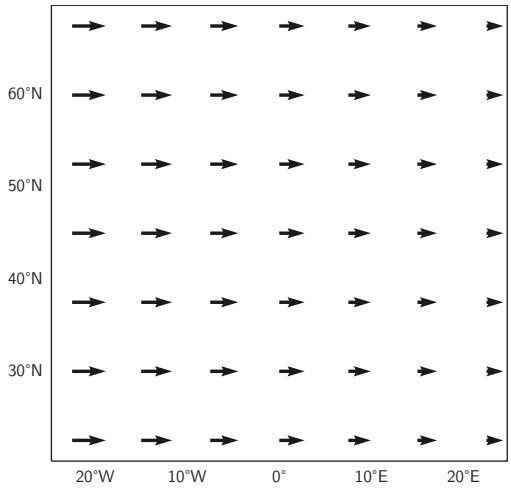
In order to determine any difference made, the deviatoric tensor field will be calculated for four simple velocity fields (Figure 2.10) using the spherical correction and without the spherical correction. In a latitude-longitude coordinate system, these vector fields represent pure compression in the latitudinal direction (Figure



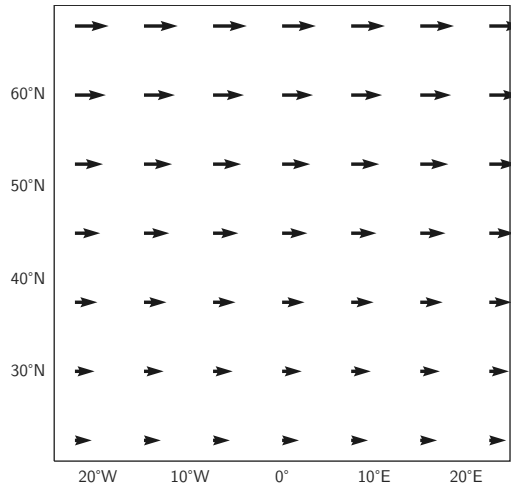
(a) Vector field of latitudinal compression.



(b) Vector field of latitudinal shear.



(c) Vector field of longitudinal compression.



(d) Vector field of longitudinal shear.

Figure 2.10: Compression and shear vector fields in the latitudinal and longitudinal directions for which the associated deviatoric tensor fields will be calculated with and without correction for spherical geometry. All vector fields have a minimum value of 1 mm/yr and a maximum value of 2 mm/yr.

2.10a), pure shear in the latitudinal direction (Figure 2.10b), pure compression in the longitudinal direction (Figure 2.10c), and pure shear in the longitudinal direction (Figure 2.10d). Due to the simplicity of these vector fields, it should be straightforward to note the effect of spherical geometry on the calculation of deviatoric tensor fields.

As a further test, the equations above will be applied to a more realistic situation. The deviatoric strain from GPE is calculated using the set of equations (2.21) for a small test region of real topography replicated at a latitude of 10°S and 35°S. These latitudes correspond to the North and South extents of the southern African study region. The discrepancies in the deviatoric stress at each point are then compared between the two latitudes, and then to the deviatoric stress calculated using equation (2.13). Since the distortion of the geodetic coordinate system is greatest towards the poles, the discrepancy between the deviatoric stresses calculated from equations (2.21) and (2.13) is expected to be largest at 35°S.

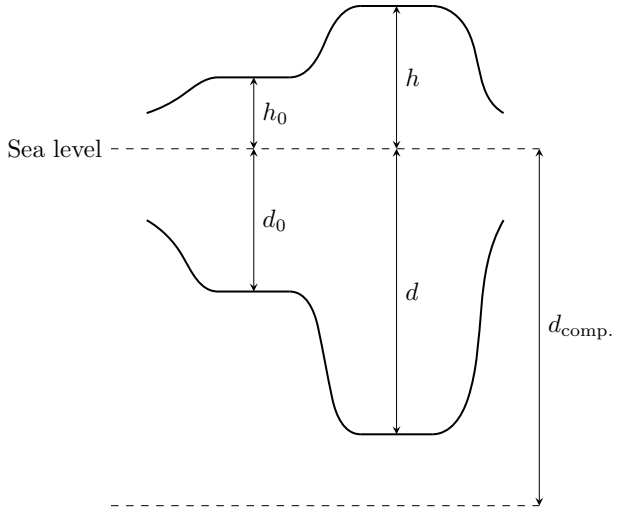


Figure 2.11: It is possible to estimate the thickness of the crust by combining surface elevation data with Pascal's principle. Here, h is height above sea level, d is the depth below sea level of the crustal root, and $d_{\text{comp.}}$ is some depth that is deeper than any crustal root such that mantle is present. The subscript 0 denotes quantities for a reference column.

2.3 Tectonic stresses in southern Africa due to gravitational collapse

In calculating the stresses due to body forces of the lithosphere, it is assumed that the lithosphere is in isostatic equilibrium. Isostatic equilibrium is the state of balance between the upper layer of mantle material and the less dense crust which sits above it. Pascal's law states that the pressure at a given depth in a fluid is the same. At some sufficient depth, the mantle is hot enough that it can deform fluidly, and so Pascal's law must hold in the absence of tractions applied to the base of the lithosphere due to mantle upwelling (often referred to as dynamic uplift). The primary mechanism of isostatic compensation is a thicker crustal root which counterbalances higher topography on the surface, which would otherwise subside due to insufficient buoyancy, which is called *Airy isostasy* (Airy, 1855). Since the crust is less dense than the mantle, thicker crust must be compensated by higher topography at the surface in order for the total mass in a column of material above a point (and thus pressure at the point) to remain constant. If the lithosphere is in isostatic balance via this mechanism, it is straightforward to calculate its gravitational potential energy at a given latitude and longitude. *Pratt isostasy* can also be achieved by having layers of constant thickness, but with lateral changes of density within the layers (Pratt, 1855). This is thought to contribute to topography variation in regions with lateral changes in the geotherm (Hasterok & Chapman, 2007), but this thesis assumes that Airy isostasy is the dominant mechanism in southern Africa.

The elevation of the surface can be used to estimate the thickness of the crust, and therefore the body forces caused by gravity acting on differences in topography (Figure 2.11). Let d_{c} be a constant depth which is deeper than the crustal root for all latitudes and longitudes. Over geological timescales, the mantle deforms like a fluid, and so at d_{c} Pascal's principle applies (it is therefore called the depth of isostatic compensation). It states that in an interconnected fluid, the pressure at a given depth is constant. The pressure at the base of a column of material is given by

$$p = g\rho(h + d) = g\rho S, \tag{2.24}$$

where g is the gravitational constant, ρ is the density of the column of material, and $h + d$ is its height (Pascal, 1663). If a column is made up of multiple materials layered on top of one another, they can be treated as separate columns and their individual pressures summed to find the total pressure at the bottom. Given the thickness and altitude of a reference column of crust, it is possible to estimate the thickness of any column of crust, if its altitude is known. From (2.24), for the two columns of lithosphere in (Figure 2.11) the following

is true

$$\begin{aligned}
\rho_\mathrm{c}(h_0 + d_0) + \rho_\mathrm{m}(d_\mathrm{c} - d_0) &= \frac{p}{g} = \rho_\mathrm{c}(h + d) + \rho_\mathrm{m}(d_\mathrm{c} - d) \\
\rho_\mathrm{c} S_0 + \rho_\mathrm{m}(d_\mathrm{c} - d_0) &= \rho_\mathrm{c} S + \rho_\mathrm{m}(d_\mathrm{c} - d) \\
\rho_\mathrm{c}(S - S_0) &= \rho_\mathrm{m}(d_\mathrm{c} - d_0 - d_\mathrm{c} + d) \\
\rho_\mathrm{c}(S - S_0) - \rho_\mathrm{m}(S - S_0) &= \rho_\mathrm{m}(d - d_0) - \rho_\mathrm{m}(S - S_0) \\
-(\rho_\mathrm{m} - \rho_\mathrm{c})(S - S_0) &= \rho_\mathrm{m}(d - d_0) - \rho_\mathrm{m}(S - S_0) \\
S - S_0 &= \frac{\rho_\mathrm{m}}{\rho_\mathrm{m} - \rho_\mathrm{c}}(h - h_0),
\end{aligned} \tag{2.25}$$

where $S_0 = h_0 + d_0$ and $S = h + d$ are the crustal thicknesses of the reference column and the column of interest respectively (Haxby & Turcotte, 1978). For computations involving topography, a Digital Elevation Model (DEM) is often used. In order to account for the added mass of water above oceanic crust, the ETOPO1 1 Arc-Minute Global Relief Model was chosen, which is a 1 arc-minute DEM that incorporates bathymetry as well as topography (Amante & Eakins, 2009).

The simplifying assumption is made that there are negligible vertical gradients in horizontal velocity. Measurements of the deformation of the lithosphere can only be taken in the very upper layers of the crust, which show elastic deformation. Bird and Piper (1980) and Houseman and England (1986) take the view that this elastic deformation follows the ductile strain of the underlying lithosphere except during and shortly after seismic events. The stress caused by the gravitational potential energy of the lithosphere is the negative gradient of its gravitational potential energy E_G at a point in the thin-sheet model, which in turn depends on its thickness. The change in gravitational potential energy is

$$\Delta E_\mathrm{G} = g\rho_\mathrm{c}\,(h - h_0)\left(S_0 + \frac{1}{2}\rho_\mathrm{m}\frac{h - h_0}{\rho_\mathrm{m} - \rho_\mathrm{c}}\right), \tag{2.26}$$

where g is $9.81\,\mathrm{m\,s^{-2}}$, the acceleration due to gravity at Earth's surface; ρ_c is $2800\,\mathrm{kg\,m^{-3}}$, the assumed density of the crust; ρ_m is $3300\,\mathrm{kg\,m^{-3}}$; and S_0 is 35 km, the depth of the crust of the reference column. This equation is obtained by integrating (2.25) with respect to elevation h after Haxby and Turcotte (1978).

The gradient of the weight of the lithosphere is calculated in spherical coordinates with r assumed to be constant. In other words, the gradient is calculated in two coordinates which parameterise the surface of a sphere

$$\boldsymbol{F} = \boldsymbol{\nabla} E_\mathrm{G} = \frac{1}{r}\frac{\partial E_\mathrm{G}}{\partial \theta}\hat{\theta} + \frac{1}{r\sin\theta}\frac{\partial E_\mathrm{G}}{\partial \phi}\hat{\phi}. \tag{2.27}$$

The horizontal deviatoric stress due to GPE is then calculated using the set of equations derived in the previous section:

$$\begin{aligned}
\dot{\varepsilon}^{\theta}{}_{\theta} &= \frac{\partial f^\theta}{\partial x^\theta}, & \dot{\varepsilon}^{\theta}{}_{\phi} &= \frac{1}{2}\left(\frac{\partial f^\theta}{\partial x^\phi} - f^\phi \cos\theta \sin\theta + \frac{\partial f^\phi}{\partial x^\phi} + f^\theta \frac{\cos\theta}{\sin\theta}\right), \\
\dot{\varepsilon}^{\phi}{}_{\theta} &= \dot{\varepsilon}^{\theta}{}_{\phi}, & \dot{\varepsilon}^{\phi}{}_{\phi} &= \frac{\partial f^\phi}{\partial x^\phi} + f^\theta \frac{cos\theta}{\sin\theta}.
\end{aligned} \tag{2.28}$$

With these results, qualitative and quantitative comparisons with deviatoric stress due to GPE will be made to measurements from the WSM (Heidbach et al., 2018), and to total moment release in the seismic record from 1976–2010 in the ISC Bulletin (Storchak et al., 2020, 2017). For the latter, the moment release will be smoothed with a Gaussian distance weighting with a smoothing diameter of 100 km.

3 Results

3.1 Comparison of interpolation methods

All six permutations of interpolation method (Gaussian only, inverse square only, Gaussian-Voronoi, inverse square-Voronoi, Gaussian-azimuthal, and inverse square-azimuthal) were successfully implemented and used to plot interpolated velocity fields from the Tibetan GNSS velocity data, for $W' \in \{0.1, 1, 10, 100\}$, where W' is the sum of the weighting coefficients D_{ij} and A_{ij}. These values of W' correspond to different values of $\bar{S}$ (the median smoothing parameter) for each interpolation method, and can be found in Table 3.1. Figures 3.1 to 3.4 show the interpolated fields for these values of W'. As expected, the interpolated fields with larger values of W' preserve short-wavelength behaviour less than those with small values for W'.

A brief qualitative assessment shows that the Gaussian-Voronoi weighting and Gaussian only weighting schemes look the most reasonable (even in regions of extrapolation), in particular because it reproduces the curving and spreading of original velocity field around the border with Myanmar and Laos at around 25°N 95°E (red ellipse in Figure 3.5). On the other hand, the vector fields for the azimuthal weighting show significant discontinuities in vector magnitude at the same location, with adjacent interpolation points not smoothly transitioning and instead abruptly changing direction. This is because of the problem highlighted earlier (Figure 2.8), where the directions of the vectors, even in regions of interpolation, are influenced heavily by data points with overlarge density weightings when interpolating to a point close to the edge of the spatial range of the data. Lastly, for the same value of W', the inverse square weighting smooths more aggressively compared to the Gaussian weighting, regardless of density weighting.

As can be seen from these figures, qualitatively comparing the performance of the various combinations of interpolation method is not an easy task. Crucially, there is no good way to compare them for the same value of $\bar{S}$ or W' because, as described earlier (Section 2.1.3), the meaning of $\bar{S}$ (on which W' depends) is not consistent between the functional forms of the Gaussian and inverse square distance weighting methods. As another method of comparison, the results of the cross-validation, known field, and noise sensitivity tests are presented.

Table 3.1: The values of $\bar{S}$ for each value of W' for each interpolation method.

		$\bar{S}$ (km)			
Density method	Distance method	$W' = 0.1$	$W' = 1$	$W' = 10$	$W' = 100$
Gaussian	Voronoi	69.7	103.0	216.5	690.6
Gaussian	Azimuthal	74.2	110.2	185.0	359.9
Gaussian	None	82.2	136.5	306.9	813.5
Inverse square	Voronoi	8.0	25.3	84.6	348.3
Inverse square	Azimuthal	4.7	15.0	49.0	172.2
Inverse square	None	9.6	30.8	102.4	391.3

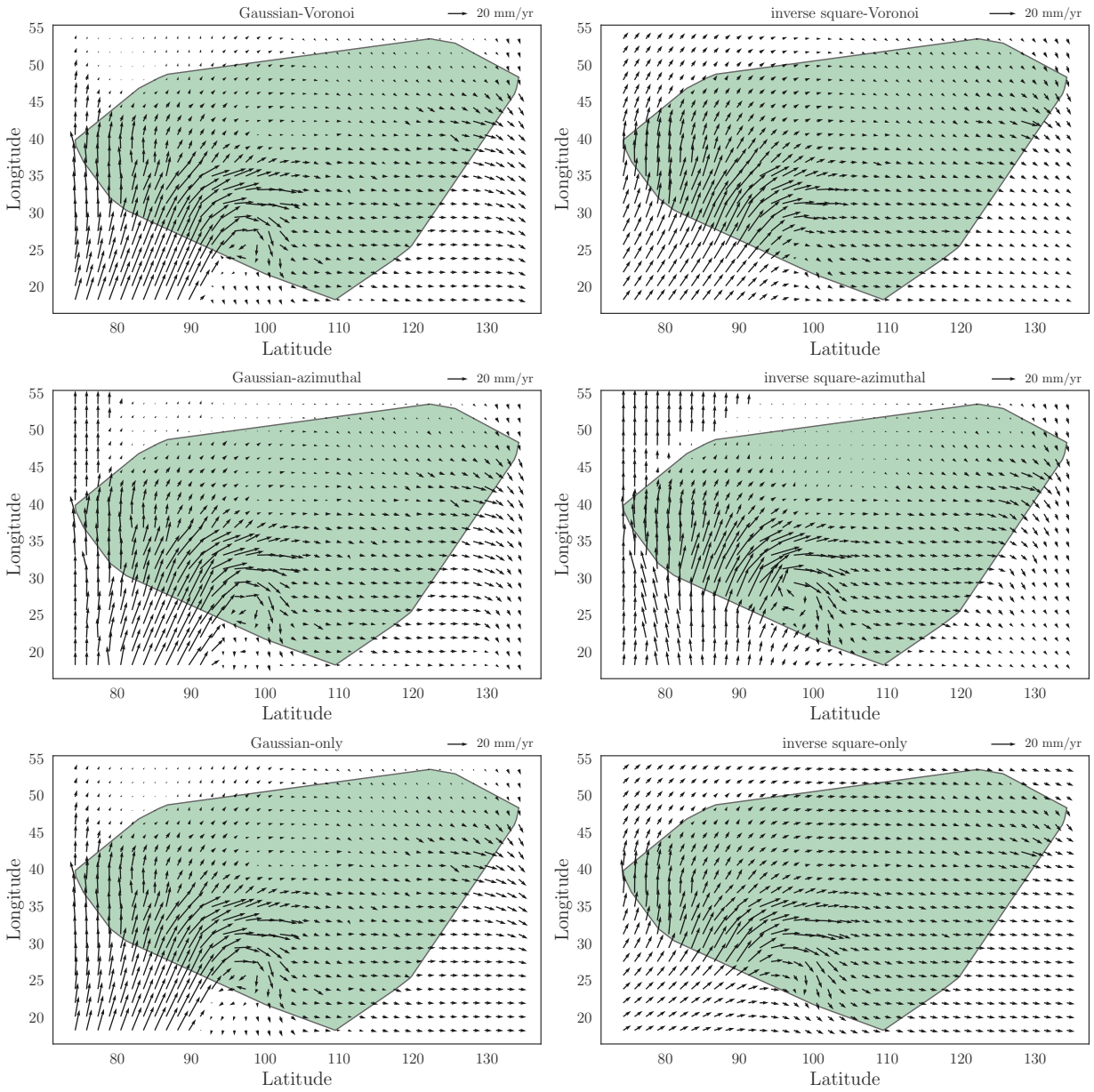


Figure 3.1: The interpolated vector field for each of the interpolation schemes for $W' = 0.1$. The region highlighted in green is the convex hull of the data and the region of interpolation.

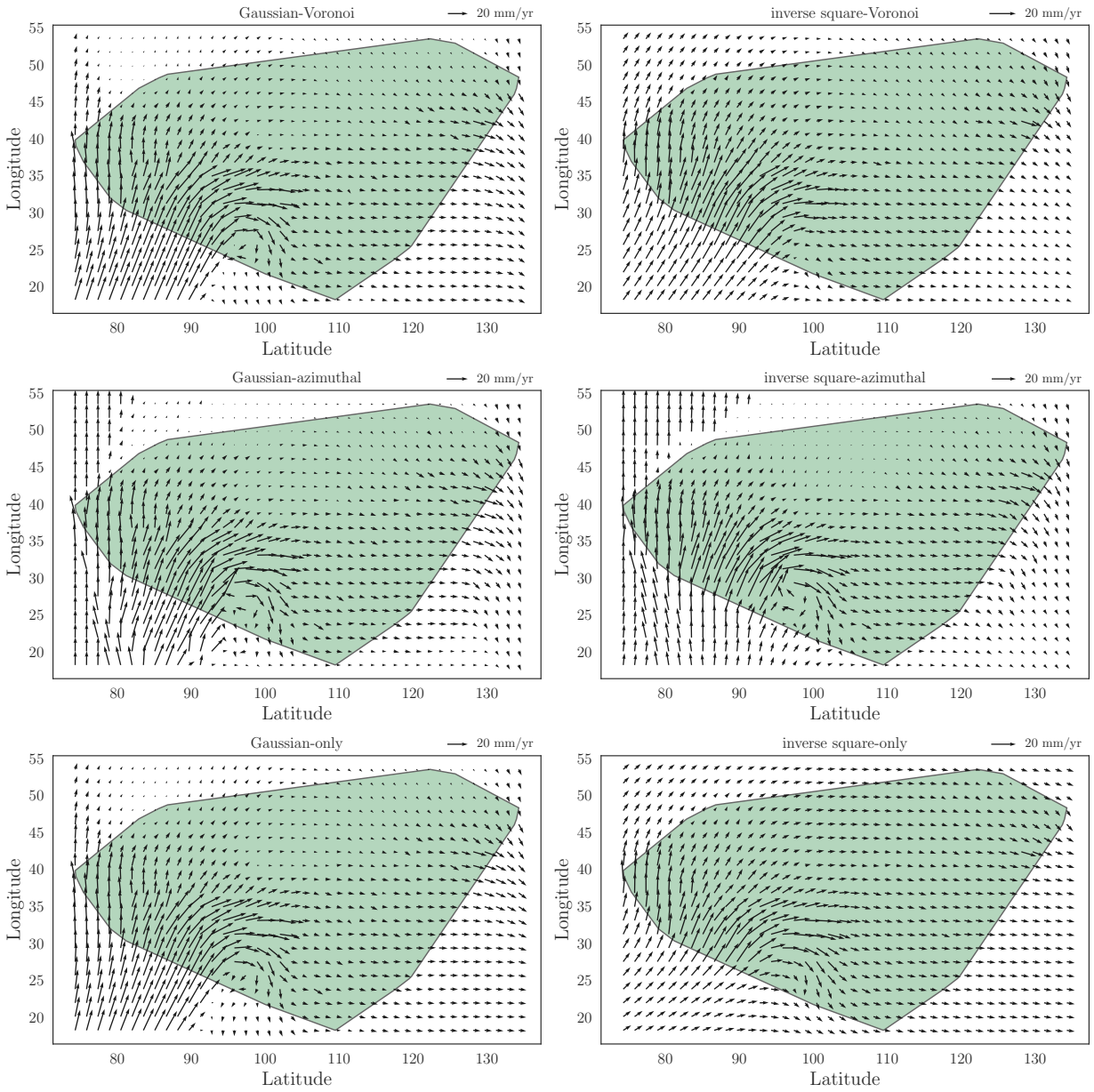


Figure 3.2: The interpolated vector field for each of the interpolation schemes for $W' = 1$. The region highlighted in green is the convex hull of the data and the region of interpolation.

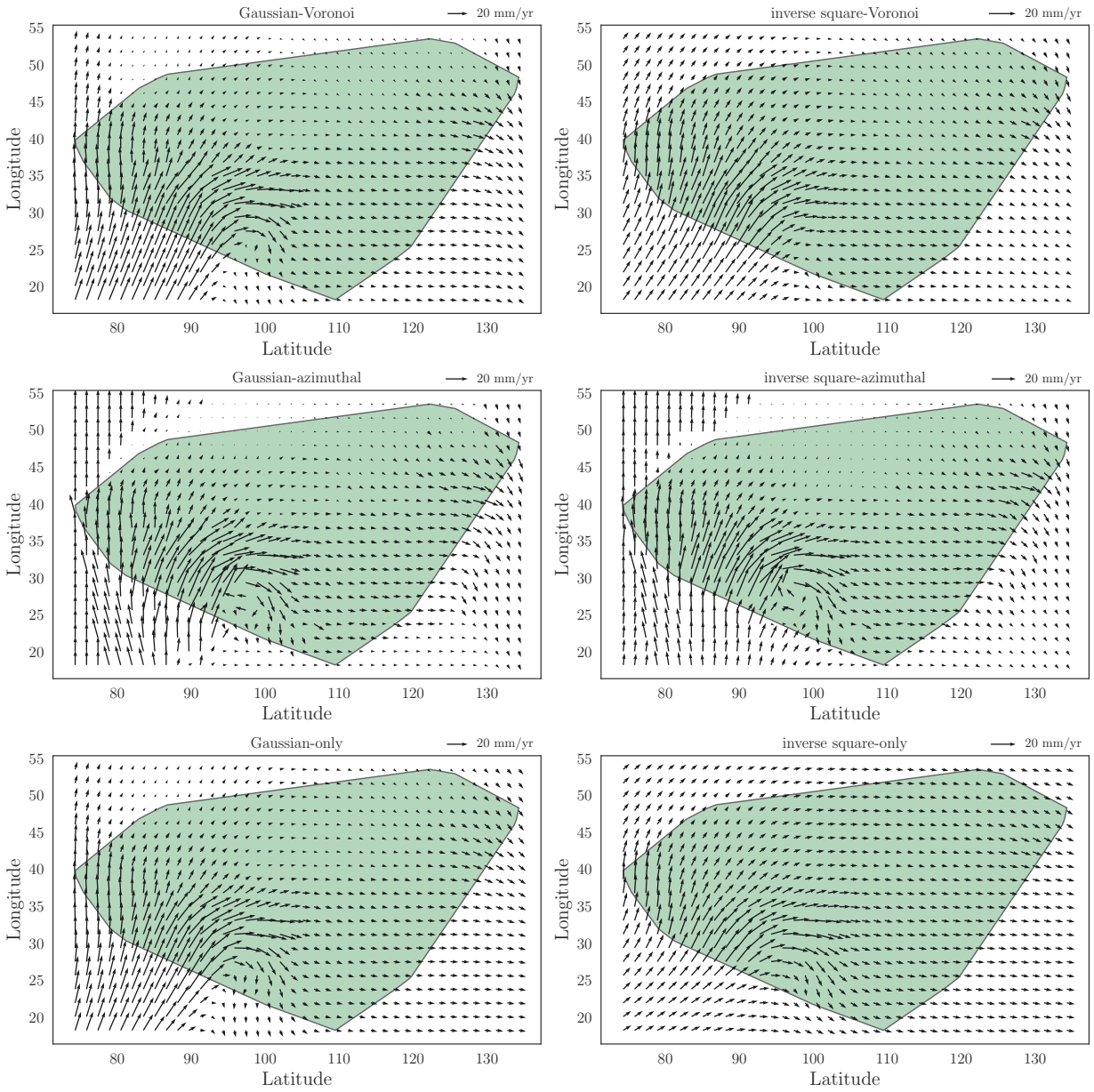


Figure 3.3: The interpolated vector field for each of the interpolation schemes for $W' = 10$. The region highlighted in green is the convex hull of the data and the region of interpolation.

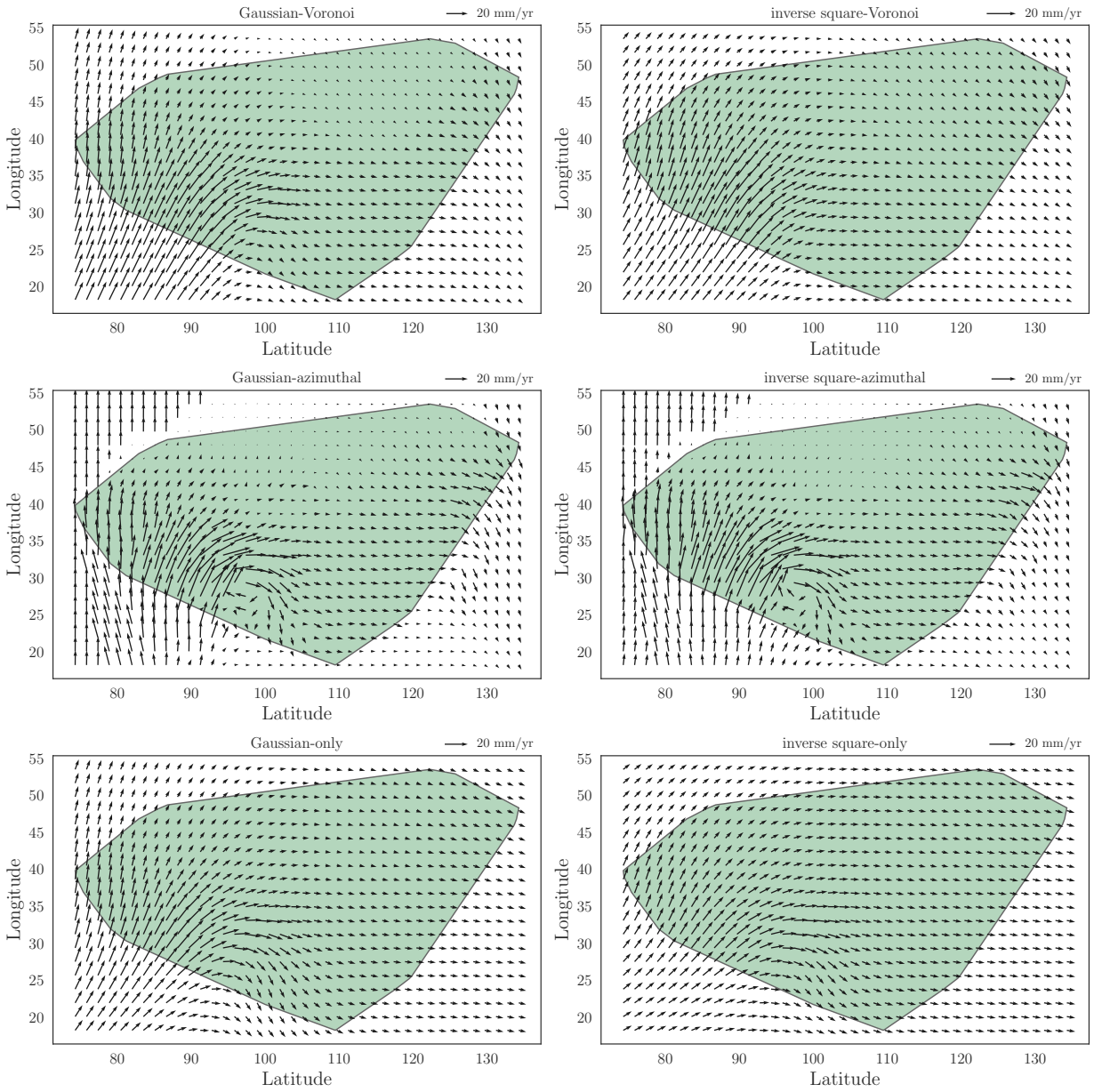


Figure 3.4: The interpolated vector field for each of the interpolation schemes for $W' = 100$. The region highlighted in green is the convex hull of the data and the region of interpolation.

3.1.1 Cross-validation error

Using a similar approach to that taken in other fields (Hofstra et al., 2008), the GNSS velocity data was self-predicted by cross-validation at each data point with each of the six interpolation methods, and the misfit quantified by the RMSE (see Section 2.1.3).

In general, the focal points of greatest mean absolute error for cross-validation were where a significant change in the local density of observations was coupled with the local velocity field changing rapidly (see for example Figure 3.6). It is also interesting that places with the highest measurement density do not, as a rule, have the lowest root mean square error. This could be due to the fact that a high local density of observations means that there is a greater chance of a very close data point dominating the interpolation with a velocity which disagrees with the measured value at the cross-validation point.

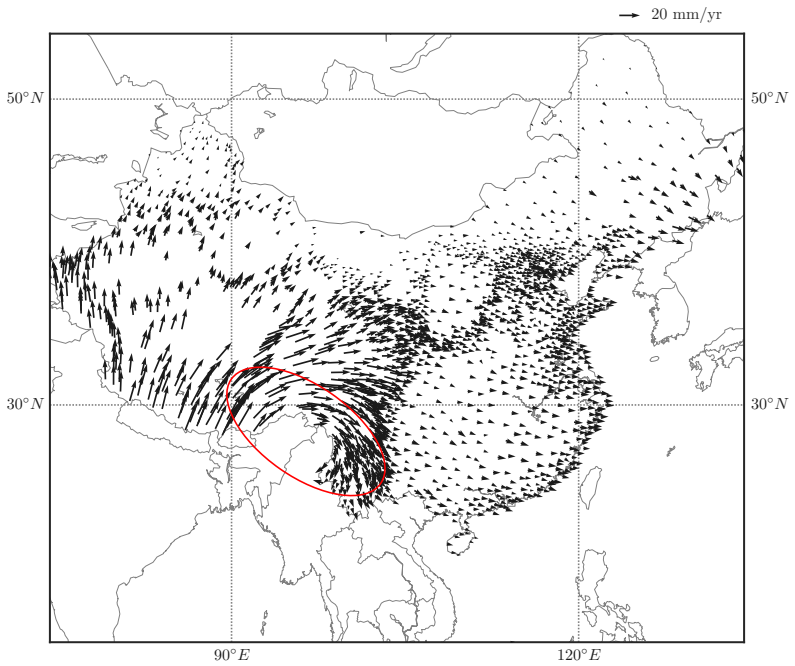


Figure 3.5: The GNSS velocities in the Eurasian study area (Zhao et al., 2015). Coordinates are degrees latitude and longitude, and velocities are relative to stable Eurasia. The red ellipse highlights the curving and spreading of the velocity field which should ideally be preserved by the interpolation methods.

Figure 3.7 shows the root mean square error in cross-validation for each interpolation method as $\bar{S}$ varies. None of the inverse square weighting schemes converge on a minimal value for $RMSE(\boldsymbol{v})$, and have a much higher misfit than their Gaussian counterparts. Of the methods with a Gaussian distance weighting, the Gaussian-azimuthal method has by far the highest $RMSE(\boldsymbol{v})$. The remaining two methods, Gaussian only and Gaussian-Voronoi, perform comparably in this test. In the range $30\,\text{km} < \bar{S} < 150\,\text{km}$ the Gaussian only method has a notably lower $RMSE(\boldsymbol{v})$ than the Gaussian-Voronoi method. At very high and very low values of $\bar{S}$, the Gaussian-Voronoi method performs slightly better.

Table 3.2 shows the minimal total errors for the various interpolation schemes along with the values of W' and $\bar{S}$ for which these minima occur. With a Gaussian distance weighting, the azimuthal interpolation scheme had a significantly higher minimal root mean square error (1.627 mm/yr) compared to both the Voronoi method (1.334 mm/yr) and no density method (1.281 mm/yr). The Voronoi scheme performed better, but had a minimal mean absolute error that was still approximately 5 % worse than no density weighting at all. In addition to the much higher mean absolute error for almost all values of $\bar{S}$, the inverse square distance

Table 3.2: Optimised values for cross-validation. A dash for the values of W'_{opt} and $\bar{S}$ indicates no optimal value was found by the minimisation algorithm, in which case the listed value for $RMSE(\boldsymbol{v})$ is the value to which it tended as $W', \bar{S} \to 0$.

Density method	Distance method	W'_{opt}	$\bar{S}_{\text{opt}}$ (km)	$RMSE(\boldsymbol{v})$ (mm/yr)
Voronoi	Gaussian	1.100	57.24	1.334
Azimuthal	Gaussian	0.273	23.23	1.627
None	Gaussian	2.083	55.64	1.281
Voronoi	Inverse square	—	—	3.157
Azimuthal	Inverse square	—	—	1.951
None	Inverse square	—	—	1.980

Table 3.3: Optimised values for known field error. A dash for the values of W'_{opt} and $\bar{S}$ indicates no optimal value was found by the minimisation algorithm, in which case the listed value for $RMSE(\boldsymbol{v})$ is the value to which it tended as $W', \bar{S} \to 0$.

Density method	Distance method	W'_{opt}	$\bar{S}$ (km)	v_{RMSE} (mm/yr)
Voronoi	Gaussian	0.015	23.5	0.50
Azimuthal	Gaussian	0.010	23.6	0.47
None	Gaussian	0.005	22.0	0.44
Voronoi	Inverse square	—	—	3.09
Azimuthal	Inverse square	—	—	2.98
None	Inverse square	—	—	1.79

weightings were unable to reach convergence—the minimisation scheme simply approached $\bar{S} = 0$ until the tolerance limit was hit. This is obviously problematic because the minimum error being at a smoothing distance of 0 km implies that the method is unsuitable for interpolating long range behaviour.

3.1.2 Known-field error

When comparing the interpolated velocity field to the known field, the locations of the most major discrepancies were wherever the interpolation methods had to do any extrapolation across concave regions where there were no data points. This can be seen, for example, in the relatively high absolute error in the lower central portion and the left hand side of Figure 3.8 (where there are no data points in Figure 2.9).

Figure 3.9 shows the mean absolute error in cross-validation for each interpolation method as $\bar{S}$ varies. For most values of $\bar{S}$, the Gaussian only interpolation scheme has a lower root mean square error than the Gaussian-Voronoi method. Between 55 to 100 km, the Gaussian-Voronoi method is slightly better than the Gaussian only method. The minimum misfit for the Gaussian only method was 0.432 mm/yr at $\bar{S} = 73.2$ km. The Gaussian-Voronoi method had a slightly worse minimum misfit of 0.453 mm/yr at $\bar{S} = 65.7$ km.

As with cross-validation, the inverse square methods perform significantly worse than their Gaussian counterparts for all values of $\bar{S}$, and they also fail to converge on a minimum misfit, approaching $\bar{S} = 0$ as the misfit decreases. For very low values of $\bar{S}$ the Gaussian-azimuthal performs similarly to the Gaussian-Voronoi method. However for $\bar{S} > 25$ km, the Gaussian-azimuthal method performs even worse and more erratically than in the cross-validation test.

Table 3.3 shows the minimum mean absolute error, and the values for the weighting threshold and average smoothing distance at which they occur. As with the results from Section 3.1.1, the Gaussian-Voronoi method had a worse minimum misfit value (0.50 mm/yr) than a pure Gaussian scheme (0.44 mm/yr). Also as with cross-validation error, the azimuthal weighting scheme is far worse than either of its competitors (although it has a similar minimum RMSE of 0.47 mm/yr). With this test, it can be seen more easily why the azimuthal scheme performs more poorly for most values of $\bar{S}$. Whenever the interpolation method has to perform any sort of extrapolation, two data points will be given exceptionally high azimuthal weights (e.g. the lower central portions of the velocity fields in Figure 3.2). As these may not necessarily be representative of the velocity everywhere in the extrapolation region, the huge weights throw off the value significantly (explained in Figure 2.8).

As with cross-validation error, all inverse square weighting schemes failed to converge. Figure A.5 shows how the methods which have no minimal value perform notably worse for $0 < W' \lesssim 10$. It also shows how beyond $W' \simeq 10$, the error grows very rapidly. Extremely high values for W' have not been plotted, however it should be relatively intuitive that the error would asymptotically approach some large value as $W' \to \infty$. This is because at some point, the mean smoothing distance would be so large that all interpolated vectors would be the same.

3.1.3 Noise sensitivity error

Figure 3.10 shows the mean relative error according to the noise sensitivity test for various values of $\bar{S}$. As before, the Gaussian only scheme has the lowest total misfit for most values of $\bar{S}$, with the Gaussian-Voronoi

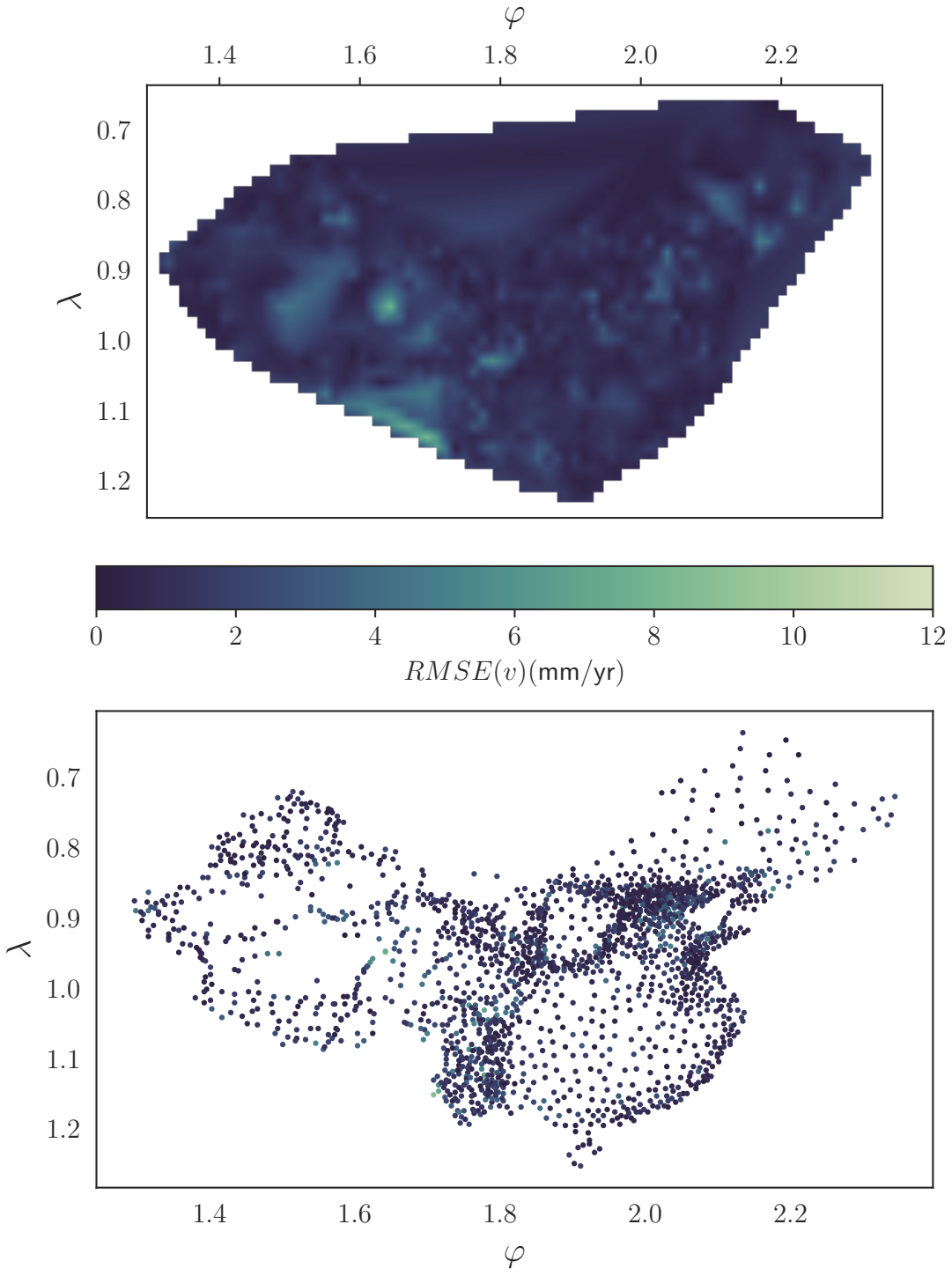


Figure 3.6: The magnitude of the vector field of differences $RMSE(\boldsymbol{v}_j)$ for the Gaussian-Voronoi interpolation scheme according to the cross-validation testing method ($W' = 1.100$, $\bar{S} = 57.24\,\text{km}$). The upper plot uses a naïve interpolation method between points for ease of visualisation, while the lower plot shows the true value of $RMSE(\boldsymbol{v}_j)$ at each data point.

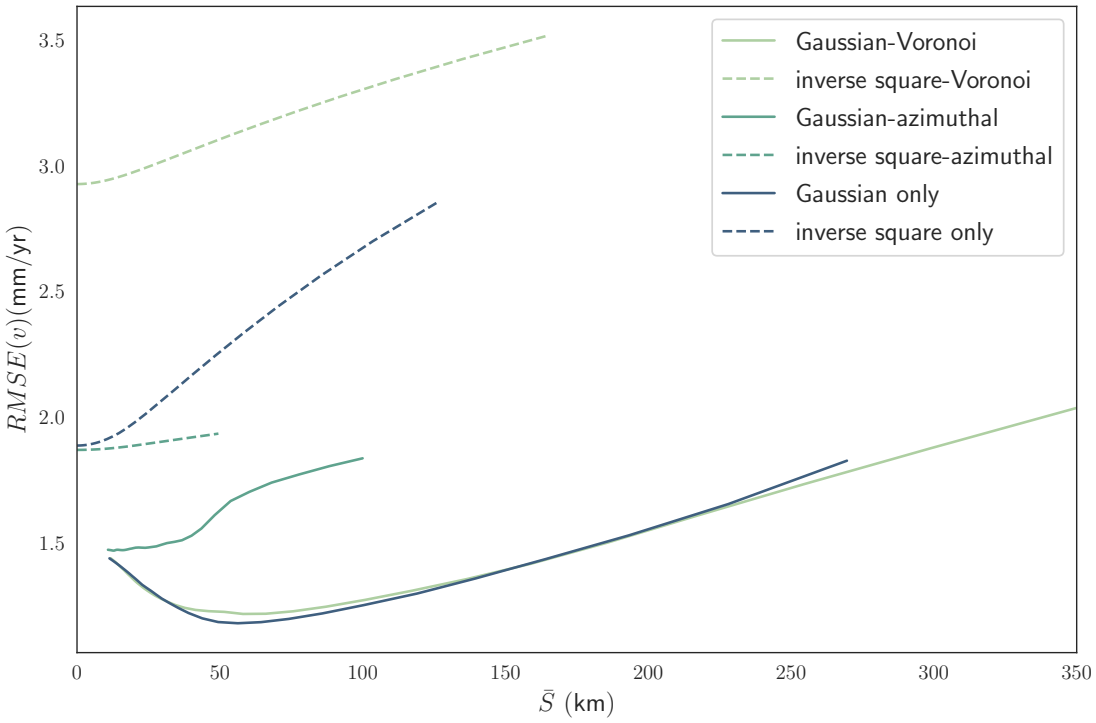


Figure 3.7: The mean absolute error $RMSE(\boldsymbol{v})$ as a function of $\bar{S}$ for the various weighting schemes (for cross-validation).

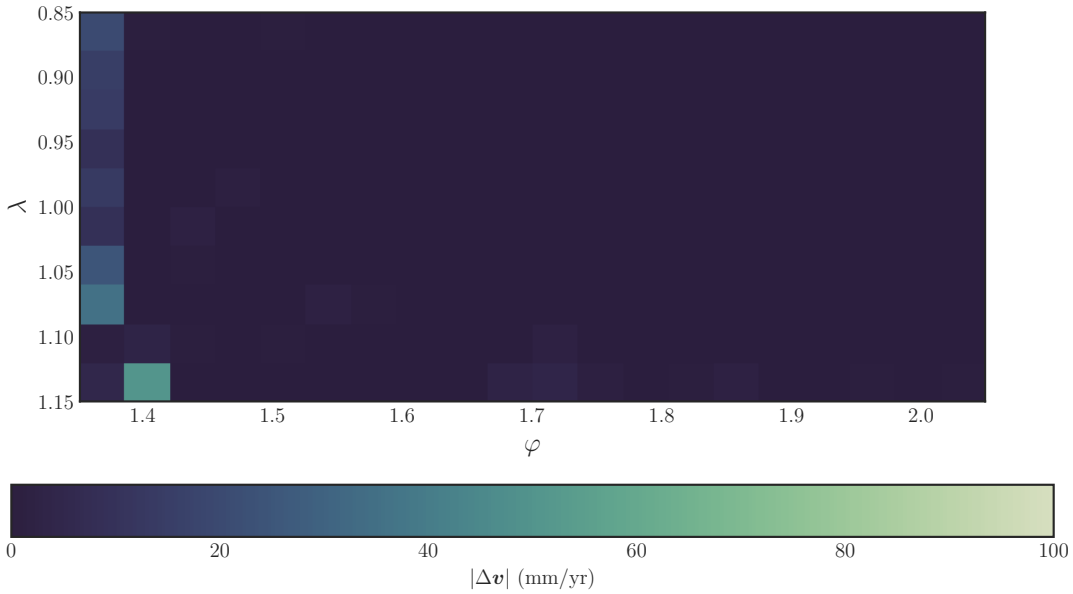


Figure 3.8: The magnitude of the vector field of differences $|\Delta \boldsymbol{v}_j|$ for the Gaussian-Voronoi interpolation scheme according to the known field testing method ($W' = 0.015$, $\bar{S} = 23.5\,\mathrm{km}$).

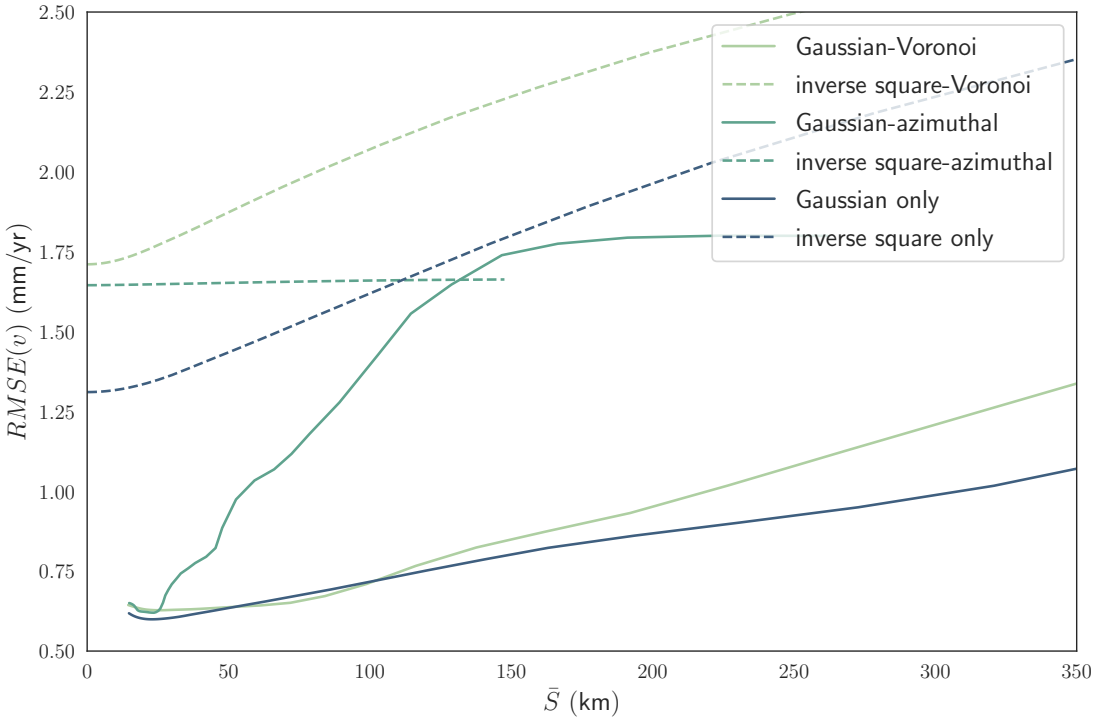


Figure 3.9: The mean absolute error $RMSE(\boldsymbol{v})$ as a function of $\bar{S}$ for the various weighting schemes (for known-field error).

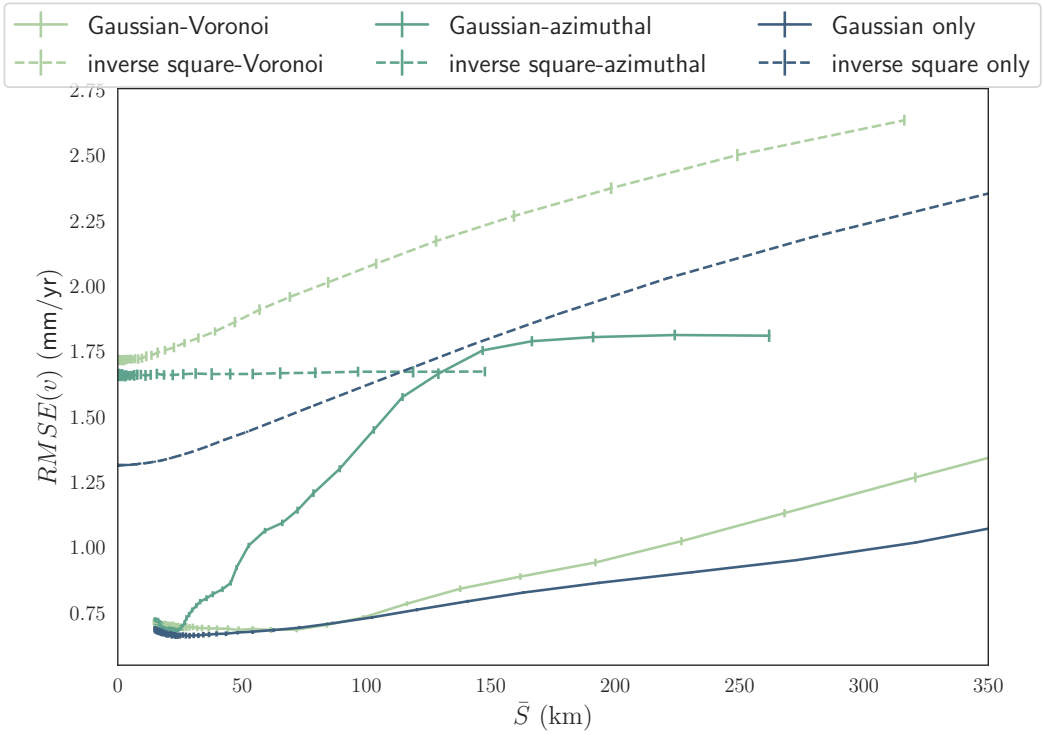


Figure 3.10: The mean over 20 runs of the mean absolute error $RMSE(\boldsymbol{v})$ as a function of $\bar{S}$ for the various weighting schemes (for noise sensitivity). Shown are the 1σ error bars.

scheme being slightly better for a small range of values. In this test however, the Gaussian-Voronoi scheme performed worse relative to the Gaussian only test, performing better over an even narrower range of values than before (60 to 80 km).

Results for the other schemes are effectively identical to the known-field test. The Gaussian-azimuthal scheme is just as erratic, with error rapidly increasing as $\bar{S}$ increases, but being competitive (in terms of misfit) at extremely low values of $\bar{S}$. The inverse square methods again perform worse than their Gaussian counterparts, and have no minimum misfit, with $RMSE(v)$ decreasing as $\bar{S}$ approaches 0. Next, the results of the new method by which to calculate spherical 2D tensor fields will be presented.

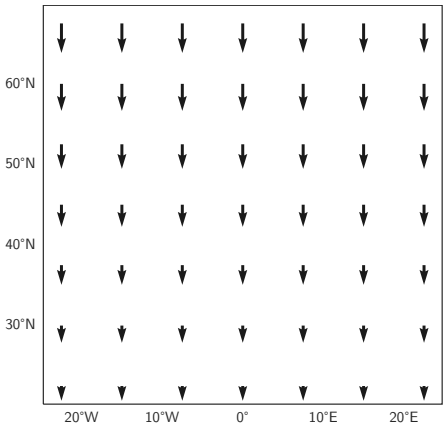
3.2 Calculation of 2D deviatoric tensor fields on a spherical earth

The images in Figure 3.11 show the simple latitudinally (North-South) oriented vector fields and the associated 2D deviatoric tensor fields calculated for a flat geometry and spherical geometry. Figure 3.12 shows the same for longitudinally (East-West) oriented vector fields. Calculating the 2D deviatoric tensor field with a flat geometry reproduces the expected behaviour. Figure 3.11c is non-zero only in the latitudinal (North-South) direction, and shows compression. Similarly, Figure 3.12c is non-zero only in the longitudinal (East-West) direction, and again, is pure compression.

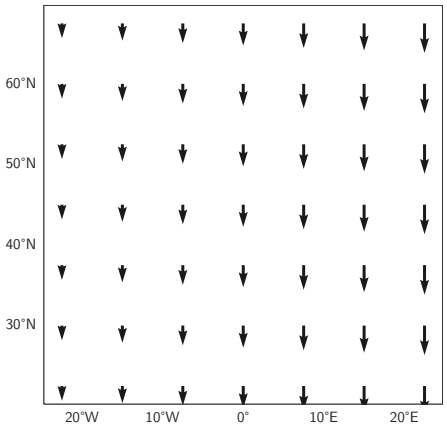
The explanation for the behaviour of the deviatoric tensor field calculated under a spherical geometry requires careful attention. In one sense, the behaviour is entirely expected and completely predictable. On the other hand, looking at latitude-longitude coordinate systems drawn on Euclidean axes is so normalised it can be difficult to incorporate the inherently spherical nature into a mental conception of the situation. The behaviour of the tensor field in Figure 3.11e is the easiest to understand. Imagine standing on the surface of the earth at 68°N 0°E and facing South. The vector arrows in Figure 3.11a located at 68°N 8°E and 68°N 8°W are not actually parallel to the vector arrow at your feet, and in fact point slightly away from you. As you walk South, the vector arrows on either side of you will become gradually more and more parallel, and the rate at which this change occurs also decreases. This means that the vector field in Figure 3.11a which superficially shows pure latitudinal compression actually hides an aspect of longitudinal (East-West) compression, which grows larger towards the poles.

This effect of the gradual change in angle between longitude lines explains some aspects of the results for the other simple vector fields as well. For example, in Figure 3.11f, the deviatoric tensors start out relatively close to pure shear at 25°N, but become progressively more dominated by compression towards the poles as lines of longitude become less parallel. However, longitudinally (East-West) oriented vector fields are more complicated to dissect, because there is the additional effect of the length of a degree of longitude shortening as latitude increases (see (2.23)). This effect is observable even in the calculations which use flat geometry—Figures 3.12c and 3.12d show an increase in shearing in the deviatoric tensor field as latitude increases. When the spherical equations are used instead (Figures 3.12e and 3.12f), this effect is of course still present.

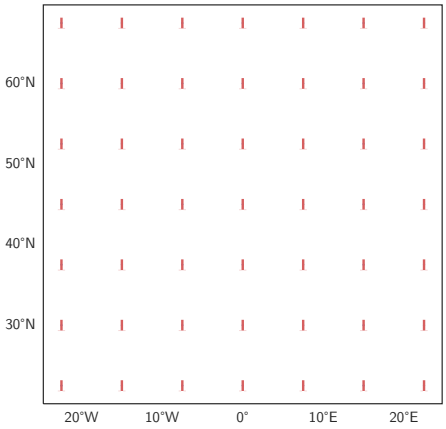
However, the longitudinal (East-West) component of the coordinate system has an analogous effect to the deparallelisation of the latitudinal (North-South) component. Except at 0°N, lines of latitude are *small circles* (circles where the plane of the circle does not pass through the centre of the sphere on whose surface they lie). Straight lines on the surface of a sphere are, by definition, *great circles*; this means that lines of latitude (being small circles) are in fact curved, which induces a natural shear on the longitudinal (East-West) element of the coordinate system. This can be seen in Figures 3.12e and 3.12f, where the amount of shear increases faster as latitude increases than in comparison to their counterparts Figures 3.12c and 3.12d, respectively. The causes of these effects of spherical geometry can be neatly summed up in a single sentence: *lines of longitude are straight lines which are not parallel to each other, while lines of latitude are parallel to each other, but are curved.* The use of this new calculation method with real data will be presented in the following section, where the results are presented of the calculation of the 2D deviatoric stress due to body forces in southern Africa.



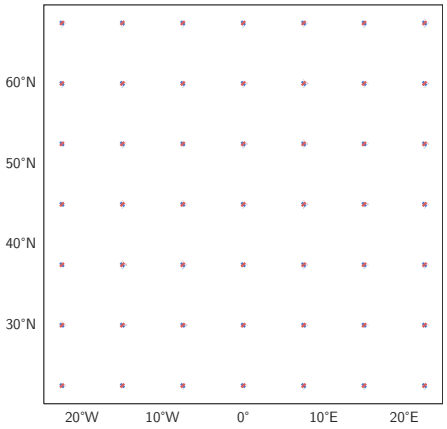
(a) Vector field comprised of latitudinal compression.



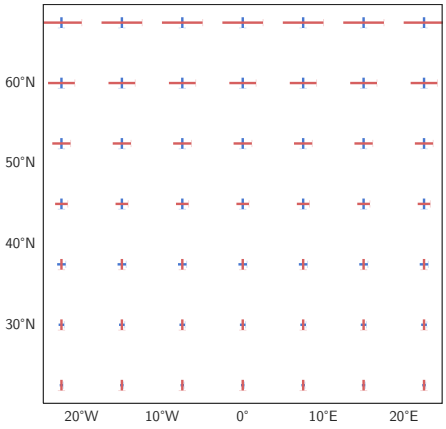
(b) Vector field comprised of latitudinal shear.



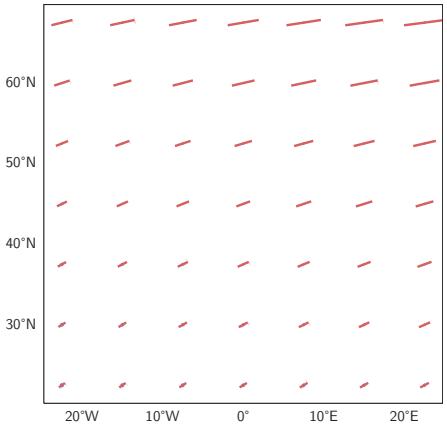
(c) The deviatoric tensor field of 3.11a under flat geometry.



(d) The deviatoric tensor field of 3.11b under flat geometry.

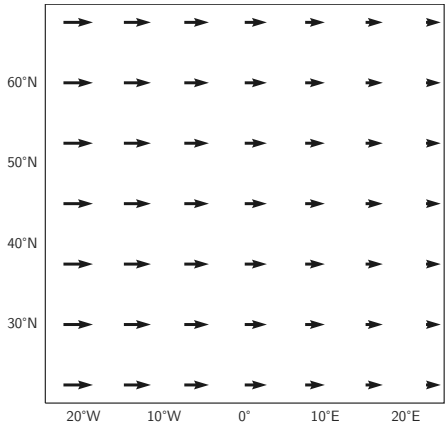


(e) The deviatoric tensor field of 3.11a under spherical geometry.

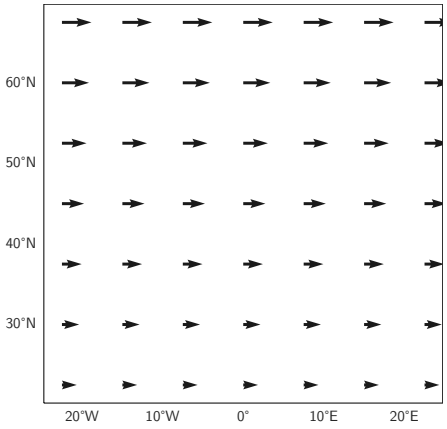


(f) The deviatoric tensor field of 3.11b under spherical geometry.

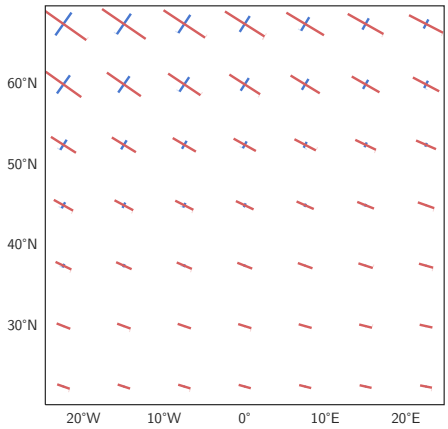
Figure 3.11: Compression and shear vector fields in the latitudinal direction and the associated deviatoric tensor fields calculated with and without correction for spherical geometry. Red bars indicate the direction and relative magnitude of greatest compression, blue bars indicate least compression.



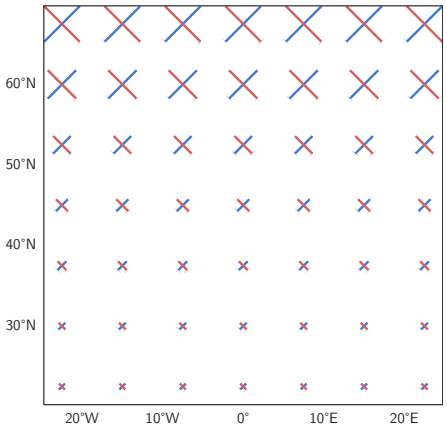
(a) Vector field comprised of longitudinal compression.



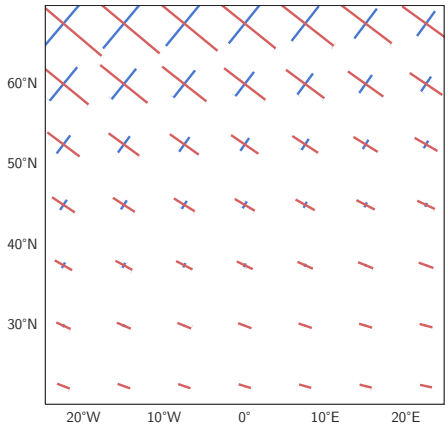
(b) Vector field comprised of longitudinal shear.



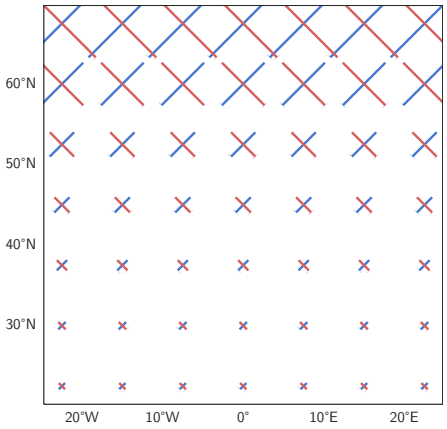
(c) The deviatoric tensor field of 3.12a under flat geometry.



(d) The deviatoric tensor field of 3.12b under flat geometry.



(e) The deviatoric tensor field of 3.12a under spherical geometry.



(f) The deviatoric tensor field of 3.12b under spherical geometry.

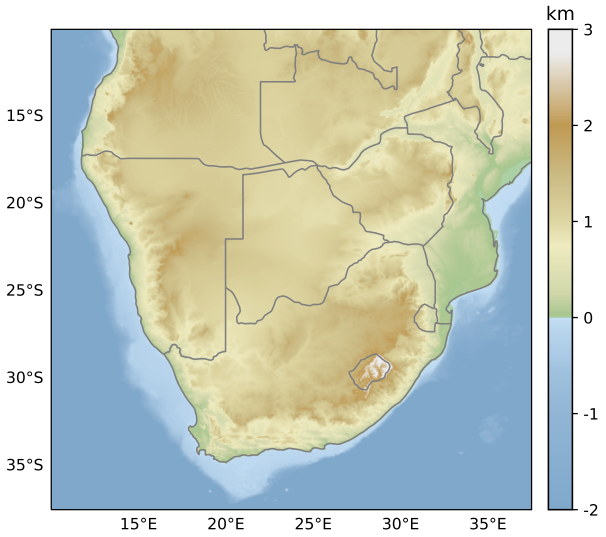
Figure 3.12: Compression and shear vector fields in the longitudinal direction and the associated deviatoric tensor fields calculated with and without correction for spherical geometry.

3.3 Tectonic stresses in southern Africa due to gravitational collapse

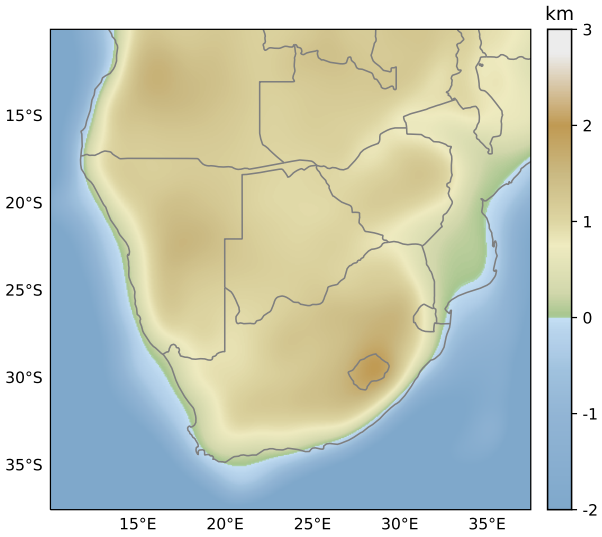
Figure 3.13a shows the topography of southern Africa according to the ETOPO1 1 Arc-Minute Global Relief Model (Amante & Eakins, 2009). Figure 3.13b shows the topography after a Gaussian filter with radius 100 km has been applied. Small details such as ordinary river valleys have been lost, but large topographical features in the region are still present, for example the mountain range running North-South near the Namibian coast, and the very large Fish River Canyon. Figure 3.13c shows the gravitational potential energy due to the thickness of the crust which is estimated from the smoothed topography in Figure 3.13b. Apart from the change in colour scheme, the two are virtually identical as the GPE is directly proportional to the crustal thickness, which is itself proportional to the topographic height (due to the assumption of Airy isostasy). The vector field in Figure 3.13d shows the body stresses caused by the GPE.

Figure 3.14 shows the horizontal deviatoric stress due to the GPE of the crust in southern Africa using the correction derived in Section 2.3 for coordinates on the surface of a sphere. Figure 3.15 shows orientation of the maximum horizontal compression (black bars) of *in situ* stress measurements from the WSM compared to the orientation of maximum horizontal compression (red bars) due to gravitational collapse. It is worth noting that for an ideal pure Andersonian normal fault, the maximum horizontal compression will be the orientation of σ_2, and it will be parallel to the orientation of the fault strike (Anderson, 1905). In most places, there is good alignment between high quality (A or B rated) measurements from the WSM and the orientation of major shear zones. In general, there is fairly poor alignment between high quality WSM measurements and the alignment of $\sigma_{1,\mathrm{H}}$ from the GPE of the topography. However, in certain locations, there is very good alignment between the orientation of all three: shear zones, WSM measurements, and deviatoric stress due to GPE. Areas marked with stars in Figure 3.15 are examples where the match between the three is excellent: the Hebron fault, at the intersection of the Namaqua-Natal Belt and the Damara-Chobe Belt; east of the Congo Craton; northeastern Magondi-Gweta belt, southern-central Namaqua-Natal Belt; and central Damara-Chobe Belt (Corner & Durrheim, 2018). The implications of these alignments (or lack thereof) will be discussed below in Section 4.3.

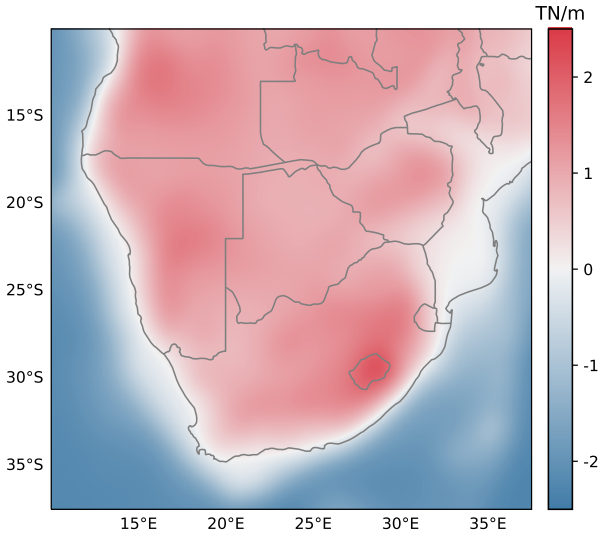
Figure 3.16 shows the seismic moment release in southern Africa from the ISC Bulletin from 1976–2010. This is then overlaid by the deviatoric stress due to gravitational collapse. Generally, the most seismically active regions do not coincide with regions where there is high deviatoric stress, however the moment release in these regions are often dominated by high-magnitude, low-frequency events (e.g. the 2017 M_{w} 6.5, Moiyabana earthquake (Kolawole et al., 2017)). In western Namibia, there is both somewhat elevated seismicity and high deviatoric strain due to gravitational collapse.



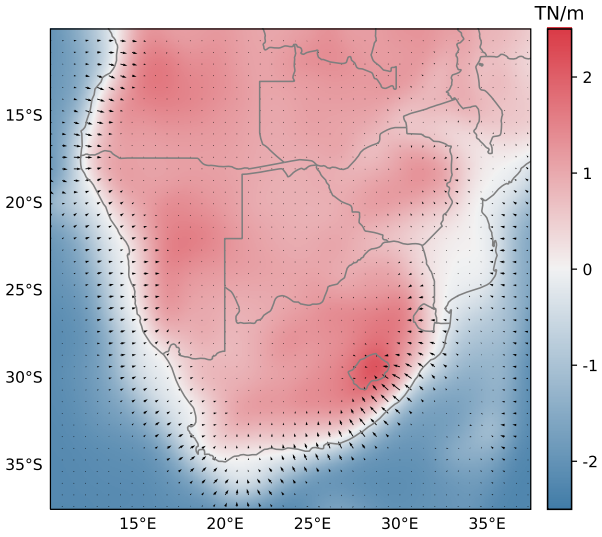
(a) The topography of southern Africa from ETOPO1 (Amante & Eakins, 2009).



(b) The topography smoothed with a 100 km Gaussian filter.



(c) The GPE inferred from smoothed topography.



(d) The body stress due to inferred GPE.

Figure 3.13: The intermediate steps necessary to calculate the deviatoric stress caused by the gravitational potential energy of the crust.

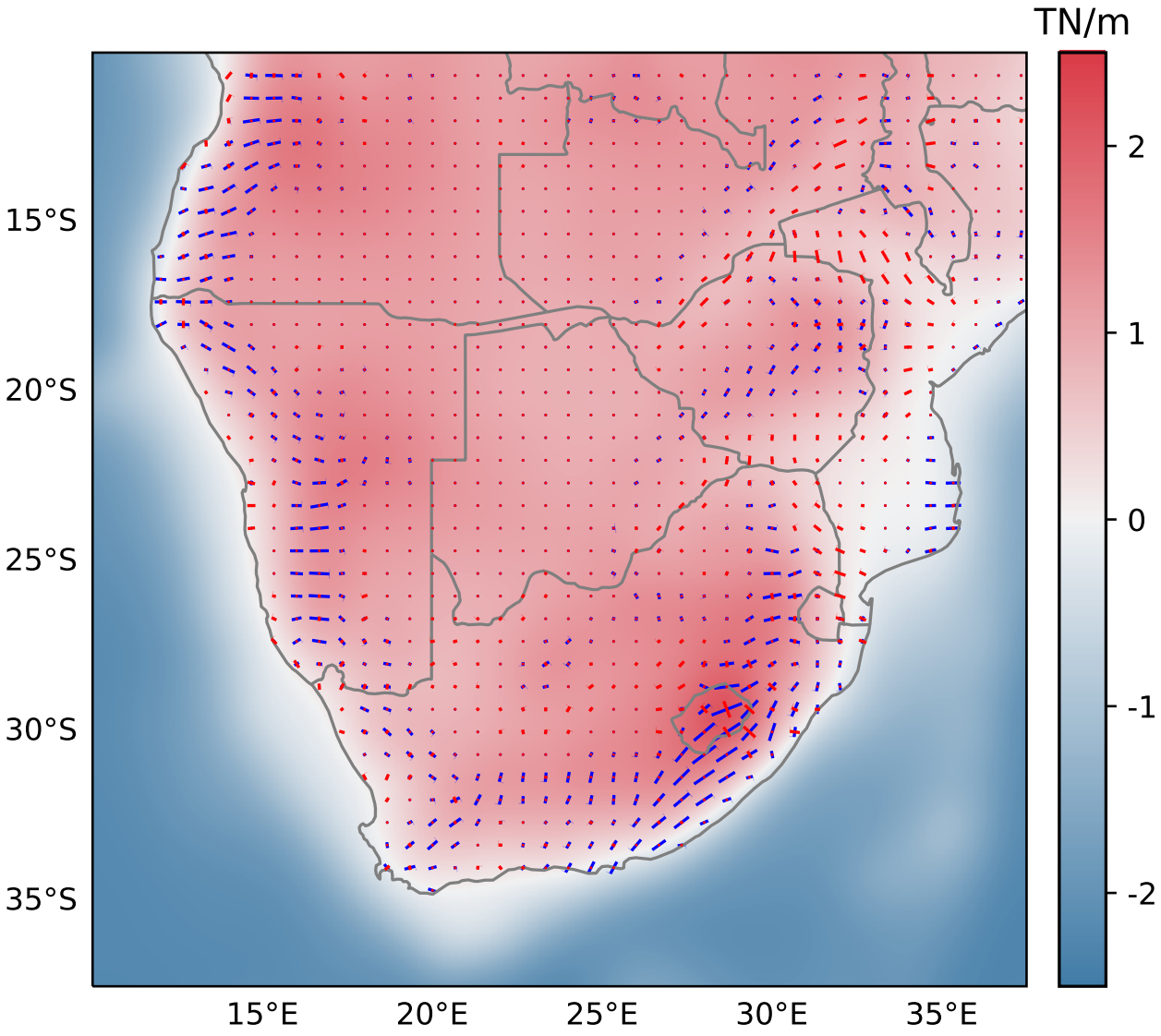


Figure 3.14: The deviatoric stress caused by the gravitational potential energy of the crust. Calculated using the curvature correction.

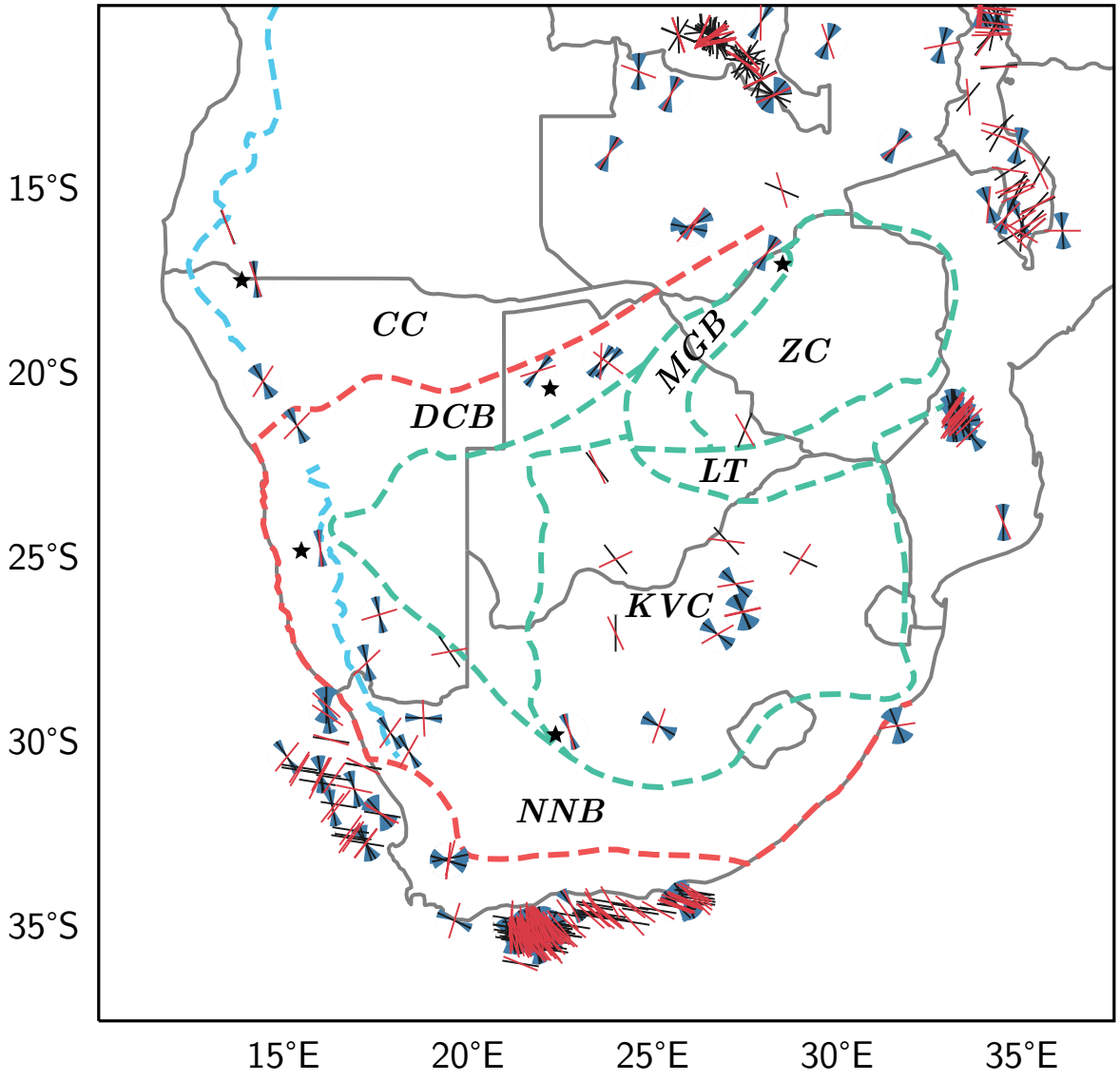


Figure 3.15: The maximum horizontal deviatoric stress sampled at locations where the World Stress Map (WSM) has *in situ* stress measurements (Heidbach et al., 2018). WSM measurements are black lines, with 1σ confidence intervals shown by the blue wedges. The red lines show the deviatoric stress from GPE. Dashed lines and initialisms show known shear zones and cratons, after Corner and Durrheim (2018): *KVC* Kaapvaal Craton; *DCB* Damara-Chobe Belt; *LT* Limpopo Terrane; *NNB* Namaqua-Natal Belt; *ZC* Zimbabwe Craton; *CC* Congo Craton; *MGB* Magondi-Gweta Belt. Stars mark select strain measurements which agree well with the GPE-derived deviatoric stress and orientation of shear zones.

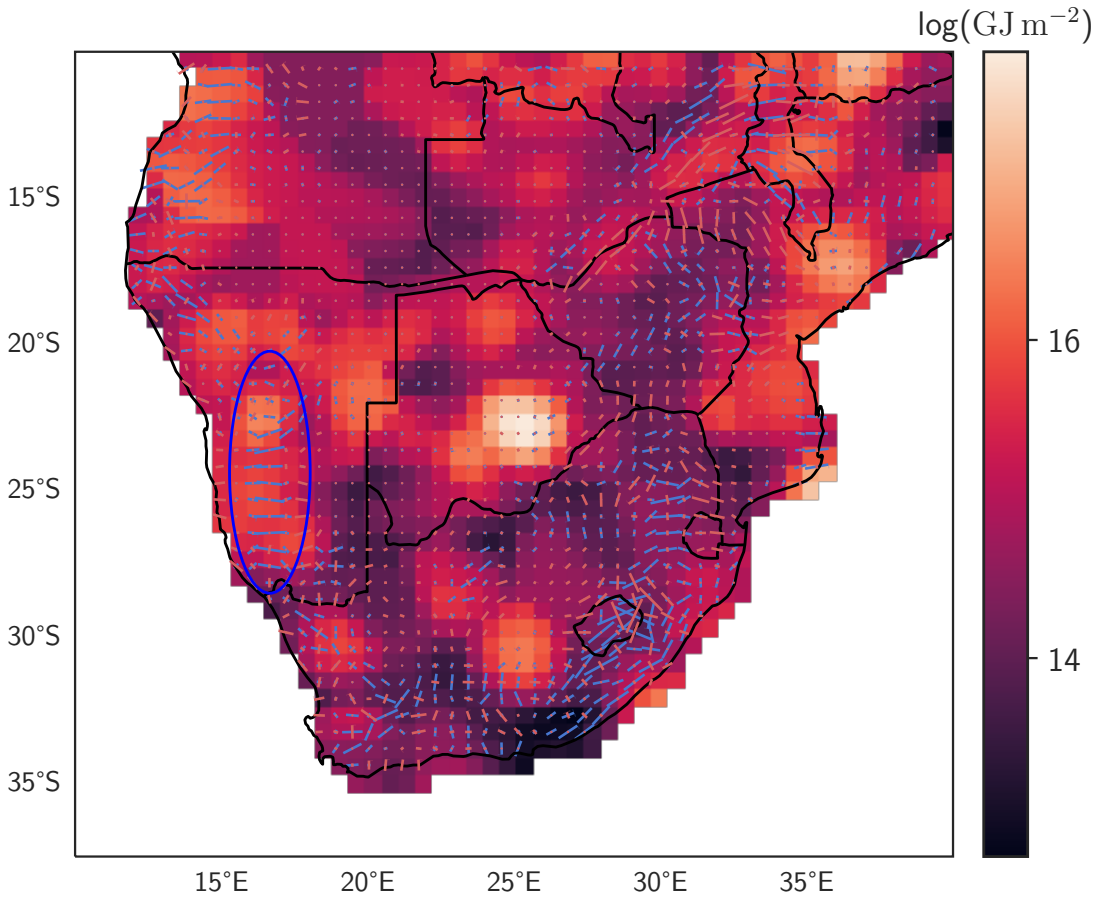


Figure 3.16: Strain release in the ISC Bulletin earthquake record, smoothed with a Gaussian filter over 100 km and overlaid by GPE-derived deviatoric stress. The blue ellipse highlights an region of elevated topography in western Namibia where there is high deviatoric stress due to body forces and there is also relatively high strain release.

4 Discussion

4.1 Comparison of interpolation methods

Three tests (cross validation, comparison to a known field, and noise sensitivity) were used to evaluate the performance of six weighting schemes for interpolation: Gaussian-Voronoi, Gaussian-azimuthal, Gaussian only, inverse square-Voronoi, inverse square-azimuthal, and inverse square only. Across the three tests, only the Gaussian-Voronoi and Gaussian methods had consistently low *and* stable root mean square errors. Figure 4.1 shows the results for these weighting schemes for the cross-validation and known field tests. For $\bar{S}$ between 40 to 120 km in the cross-validation test, the Gaussian-Voronoi method performed slightly worse. For smaller values of $\bar{S} \lesssim 40$ km, and values of $\bar{S} \gtrsim 120$ km, the Gaussian-Voronoi scheme shows slight improvement over the Gaussian only method. In the known field test, the methods are more closely comparable over the range 40 to 120 km, and for larger values of $\bar{S}$, the Gaussian only method is significantly better.

The cross-validation test uses real-world data as both the source for interpolation and for the root mean square misfit comparison. However this does not mean it is the most representative of real world conditions of interpolation; in practice, it is unlikely that interpolation would need be done onto an irregular pattern of interpolation points. Figure 4.2 shows how the distances from interpolation points to data points are skewed towards smaller separations for the cross-validation test compared to the known field and noise sensitivity tests. Note that cross-validation is probably less representative of the usual separation between data and interpolation points, because in cross-validation the interpolation points are also the data points, which would not be the case when carrying out interpolation for use in a model. A possible extension of this test would be to use only a subsample (or subsamples) of the data points as interpolation points using a density threshold. On the other hand, the velocity field used in the known field and noise sensitivity tests is artificial, and created to represent a smooth velocity field representing long wavelength deformation. This means it does not have discontinuous features incorporated in it that would be present in a real data set. Such features would include large short wavelength changes in velocity due to the presence of a major fault, postseismic or coseismic deformation, monument instability, hydrologically induced subsidence, etc. Thus, it is difficult to make a case for either test being a solely appropriate evaluation method and the three tests should be considered together.

With this in mind, the Gaussian only and Gaussian-Voronoi methods are by far the best performing of the six interpolation methods tested. Of these two, there is a reasonable theoretical argument as to why one might prefer the Gaussian-Voronoi method. In regions where there is a large gradient of data point density, it is easy to see that, on paper, the inclusion of a density weighting should improve the quality of interpolation (Figure 2.2). However this hypothesis is not borne out by the quantitative analysis in this work. In all three tests, there was very little difference between the two methods, and for the lengths of median smoothing parameter typically chosen in the literature, the Gaussian only method performed the same or slightly better depending on the test. It is also important to recognise that each of these tests was performed with one set of data only. The RMSE for the Gaussian only and Gaussian-Voronoi methods are very similar to each other in all three tests (except for very large values of $\bar{S}$), so it is quite likely that for different data sets the precise shape of the error curves could change which method is better at a specific value of $\bar{S}$. In other words, these three tests probably did not reveal a statistically significant difference between the two best performing methods (Gaussian-Voronoi and Gaussian only), but showed that the remaining methods are significantly worse (Gaussian-Azimuthal, inverse square-Voronoi, inverse square only, and inverse square-azimuthal).

Previous works which use these interpolation methods to smooth GNSS data sets almost always choose values for the smoothing parameter of $\bar{S} = 100$ km, with the justification that 100 km is the distance across which the modelled deformation is expected to occur, because this is roughly the thickness of the isostatically supported lithosphere (England, Houseman & Nocquet, 2016; Walters, England & Houseman, 2017). A larger value of $\bar{S}$ certainly increases the smoothness of the interpolated data. However, Figure 2.3 shows that, for the same value of the smoothing parameter $\bar{S}$, the choice of distance weighting function $D(r)$ dramatically changes how many data points at a specific distance will influence the value of interpolated data. While this statement might seem trivial in retrospect, the consequence of this is that a choice of smoothing parameter of $\bar{S} = 100$ km for an interpolation method does not in general correspond to the data being smoothed on the 100 km length scale. Whatever distance weighting method is chosen, a choice of $\bar{S}$ still needs to be made. One

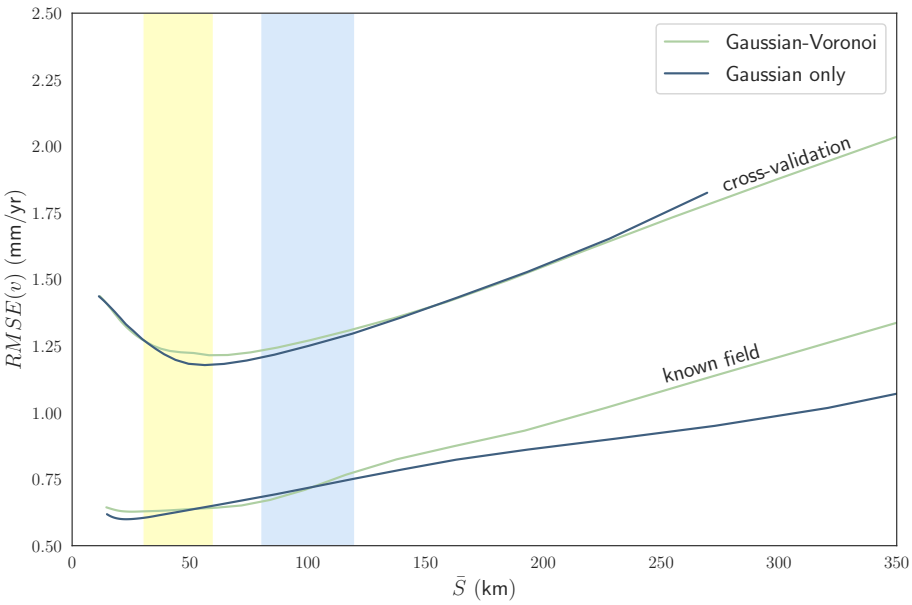


Figure 4.1: The mean absolute error $RMSE(\boldsymbol{v})$ as a function of $\bar{S}$ for the best performing weighting schemes (for the cross-validation and known field tests). The highlighted regions show appropriate ranges for $\bar{S}$ when doing crustal scale modelling (blue) and local hazard assessment (yellow).

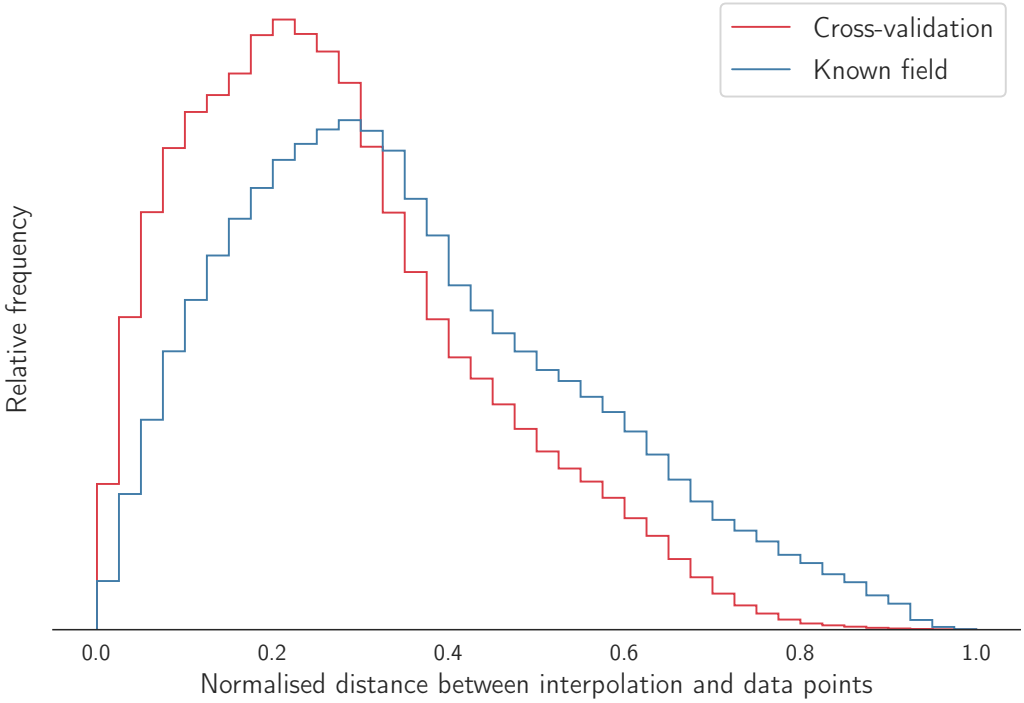


Figure 4.2: Histogram of distances between interpolation and data points for the cross-validation and known field tests. As the data sets cover differently sized areas, the distance between data points has been normalised.

possibility is to use cross-validation minimisation to find a value of $\bar{S}$ which is able to reproduce the original velocity field with as little error as possible, but this may not result in appreciable smoothing and so would only be appropriate for short wavelength modelling. For longer wavelength modelling, one might simply apply the interpolation method to their data for a range of values of $\bar{S}$ and choose one which qualitatively smooths over the desired length scales.

The theoretical and quantitative analysis in this thesis suggests that, of the six interpolation methods proposed by Shen et al. (2015), four can be easily discounted as unfit for purpose (Gaussian-Azimuthal, inverse square-Voronoi, inverse square only, and inverse square-azimuthal). The remaining two methods (Gaussian-Voronoi and Gaussian only) appear to perform similarly well according to this analysis, despite the theoretical advantages of including a sophisticated density weighting. Additionally, future works may wish to implement the extensions of these interpolation methods to a spherical geometry (see Section 2.1 and Appendices B.1, B.3, and B.4).

4.1.1 Limitations and further study

The known field and noise sensitivity tests used a vector field which is not completely analogous to a real-world velocity field for two main reasons. Firstly, and most obviously, it is simply artificially constructed, and so even though the defining equations were chosen to closely represent a real tectonic deformation event, there is no guarantee that real-world tectonics could produce a velocity field with the same long-wavelength behaviour. Secondly, it does not include any short-wavelength behaviour, for example non-elastic deformation caused by faulting. More generally, this work only explores three tests for the efficacy of the interpolation methods, only uses one method to calculate the interpolation error, and only tests them with one spatial distribution of data points.

Additionally, there is another commonly used interpolation and smoothing method in geoscience for data on the surface of a sphere: spherical harmonic analysis. Essentially, this is a spherical extension of Fourier analysis, and in just the same way can be used to smooth an irregular set of data by filtering higher frequencies in spherical harmonic space (Muir & Tkalčić, 2015). It has previously been proposed as a method to aid modelling global tectonic deformation (Grafarend, 1986), however resolving spatial scales of 100 km would require using harmonics up to very high degree, and the computational expense of solving the resulting system of equations would be prohibitive. This may be resolved by instead using Slepian functions, which are a regional version of spherical harmonics. These are constructed from a linear combination of the normal spherical harmonics in such a manner that they are very close to zero outside a region of interest (Plattner & Simons, 2014; Simons, Dahlen & Wieczorek, 2006). Another similar method is wavelet frame analysis, which may be more well suited to irregularly spaced data sets (Chambodut et al., 2005).

Further work could test these interpolation methods (including Slepian functions and wavelet analysis) with many artificial velocity fields, sample the velocity field with multiple distributions of data points, and implementing other error calculation methods (for example ones used by Hofstra et al., 2008). In particular, these additional artificial velocity fields and data distributions could more directly test situations where the more sophisticated interpolation methods are expected to improve the interpolation. This could be done by considering areas (real or artificial) which have the characteristics of Figure 2.2b.

4.2 Spherical corrections to 2D deviatoric tensor fields

These results show that there are non-trivial inaccuracies when calculating tensor derivatives of vector fields on the surface of the earth using a flat coordinate system. A flat coordinate system masks two systematic errors which get larger with increasing latitude: 1. inherent compression in the latitudinal (north-south) component of a deviatoric tensor field, and 2. inherent shear in the longitudinal (east-west) component of a deviatoric tensor field. The cause of these errors is that, on the surface of a sphere, *lines of longitude are straight lines which are not parallel to each other, while lines of latitude are parallel to each other, but are curved.* Using a spherical surface coordinate system (i.e. latitude-longitude) when computing the derivatives eliminates both of these errors.

Ultimately, the impact of the spherical correction to the calculation of deviatoric stress is highly dependant on both the general latitude and the range of latitudes of the region of interest. Polar regions are more

distorted than equatorial ones for the same latitude span, because the size of this distortion also increases with latitude. The absolute size in these effects can be large, even for equatorial regions or small latitude spans, but the *relative change* over the latitude span can be very small (the relative change is somewhat more important, as the absolute size of the effects is essentially a constant systematic error). For example, a compressional 1 to 2 mm/yr south velocity field over the rough latitude range of South Africa (20°S to 35°S) has a longitudinal compression factor that is almost 20 % as large as the latitudinal component, but the relative change across the region is only 0.7 %. Including the spherical correction is straightforward—it is just a conversion of easting and northing velocities from meters per second to radians per second, and the addition of an expression to each term of the deviatoric stress tensor.

Previous thin sheet modelling studies have aimed to allow a very broad comparison between stresses and strain rates across large areas. They have made sweeping assumptions, often including very simplified rheological models, and have aimed to demonstrate the broad congruence between expected viscous flow in the mid-lower crust and upper mantle and the deformation of the elastic crust (England & Houseman, 1986), or to estimate the average mantle rheology over large areas (England, Houseman & Nocquet, 2016; Walters, England & Houseman, 2017). The comparatively small and gradual changes associated with the analysis described here are very unlikely to make a meaningful difference at these resolutions. However, as the density and quality of data grows further (e.g. GNSS measurements, seismometer networks, and thermal constraints on variations in rheologically relevant parameters), and in turn there are more detailed models which aim to discriminate variations in rheology, these improvements may become important. Since the coordinate system of a single UTM zone is not necessarily ideal for the purpose of large scale modelling, and integrating multiple UTM zones together requires significantly more effort than the solution presented here, this formulation may be a viable alternative when modelling over areas significantly larger than a single UTM zone.

4.3 GPE contributions to southern African strain

The deviatoric stress data shown in Figure 3.14 make sense when compared to the topography from which they are derived. In particular, the Great Escarpment (Truswell, 1977) is fairly well defined by a band of increased horizontal extension, and the flat interior region by very low magnitude deviatoric stress. However, it is clear from Figure 3.15 that in many regions of southern Africa, gravitational potential energy of the crust is not the only controlling factor of the expression of strain. Figure 4.3 shows the distribution of angles between deviatoric stress inferred from GPE and measurements from the WSM (Heidbach et al., 2018). A large number of the WSM measurements which do not match the deviatoric stress from GPE have low measurement quality rating (i.e., high uncertainty in the orientation of σ_1).

In many places, high quality WSM measurements do not match well the orientations of stress derived from GPE. Within the Kaapvaal Craton the agreement is very poor (for example the Koffiefontein earthquake west of Lesotho), but many of these events may indicate the role of mining in modifying the local stress state (Brandt et al., 2005). Agreement with the Ceres-Tulbagh earthquake (a strike-slip event rather than a normal faulting event) is also poor—again, this event took place on a preexisting structure which likely formed during the formation of the Cape Fold Belt (Smit et al., 2015). On the other hand, there are some places where the match between orientation of GPE derived stress and the WSM is fairly good. One such place is the Hebron fault in southwestern Namibia, where 2D-σ_1 for the GPE derived deviatoric stress is aligned to within 5° of 2D-σ_1 of the fault (Salomon et al., 2022; White et al., 2009). In the southwest branch of the East African Rift (Daly et al., 2020) the agreement is good in places where the stresses also line up with the mobile belts, but poor elsewhere. The location marked by a star at the southwest edge of the Kaapvaal Craton matches well, but again agrees with the craton boundary.

Generally, there is good alignment between the orientation of WSM measurements and the orientation of preexisting fabric throughout the region, which suggests that it is the fabric which is the dominant controlling factor in most places, rather than small variations in the stress orientations due to gravitational collapse. This result is perhaps to be expected, because in the more rapidly evolving main branch of the EAR, previous studies have reached similar conclusions. That is, that while stress is predominantly caused by gravitational collapse, the orientation of preexisting fabric strongly dominates the expression of stress (Craig et al., 2011; Morley, 2010; Williams et al., 2019). Indeed, Morley (2010) finds that the orientation of stress itself is modified by the weak zones in a similar manner to that which occurs near the San Andreas Fault (Sbar et al.,

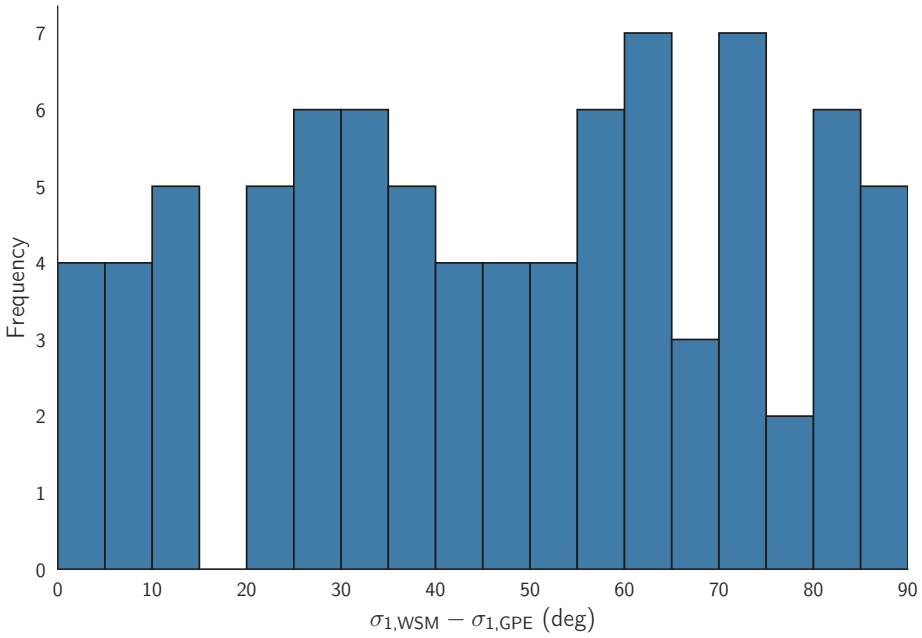


Figure 4.3: The distribution of the angles between *in situ* WSM measurements and the principal stress axis due to GPE, with only high quality measurements included (grades A, B, and C).

1979). This effect could plausibly explain much of the deviation between the orientation of the modelled GPE derived stress and WSM measurements which was found in southern Africa.

In order to further investigate the relationship (or lack thereof) between topography, stress, and strain, the earthquake record was examined. If there is a strong causal relationship between topography in a particular area and the deformation present there, one might expect that earthquakes are concentrated where there are sharp gradients in the topography (which in the simple model considered would coincide with gradients in gravitational energy which must be supported by increased shear stresses). Figure 3.16 shows the distribution of strain release present in the earthquake record in southern Africa overlain by the GPE derived deviatoric stress. In western Namibia, there is a concentration of strain release that coincides with the high deviatoric stress caused by the region of elevated topography that runs North-South (see the blue ellipse in Figure 3.16). In many other places with high strain release there is relatively poor correlation. There are also many places where there is a high magnitude of GPE derived deviatoric stress, and very little strain release. However this latter fact does not mean it is unlikely that the topography is driving deformation. Southern Africa is an SCR, and SCR faults have very long recurrence intervals (if such a concept even applies in this setting (Calais et al., 2016)), and the earthquake record is simply not long enough to make up a significant portion of the recurrence interval for an SCR. Figure 4.4 shows the magnitude-frequency relationship in southern Africa in the ISC Bulletin from 1976–2010. There is clearly incompleteness in the record below $M_{\mathrm{w}} \simeq 3$. The Gutenberg–Richter law states that earthquake magnitudes are distributed exponentially, and it can be used to characterise the completeness of the seismic record (Gutenberg & Richter, 1954). However, it is very difficult to determine where (if anywhere) the relation holds. If it holds for $3 \leq M_{\mathrm{w}} \leq 5$, then it doesn't obviously hold for $M_{\mathrm{w}} > 5$, and vice versa. These areas with high GPE derived stress may only accumulate enough energy to cause an earthquake in many thousands of years time, especially if there are no preexisting structural weaknesses to be reactivated.

Many of the hotspots in moment release coincide with either mining, especially in the central Kaapvaal Craton near Pretoria (Brandt et al., 2005), or with large historical events such as the Moiyabana and Koffiefontein earthquakes. This implies that aftershocks of geologically recent events (and those events themselves) dominate the energy release budget, and so Figure 3.16 shows where earthquakes have happened to occur within the last 50 years, rather the long term pattern of strain. Nevertheless, if the mining areas

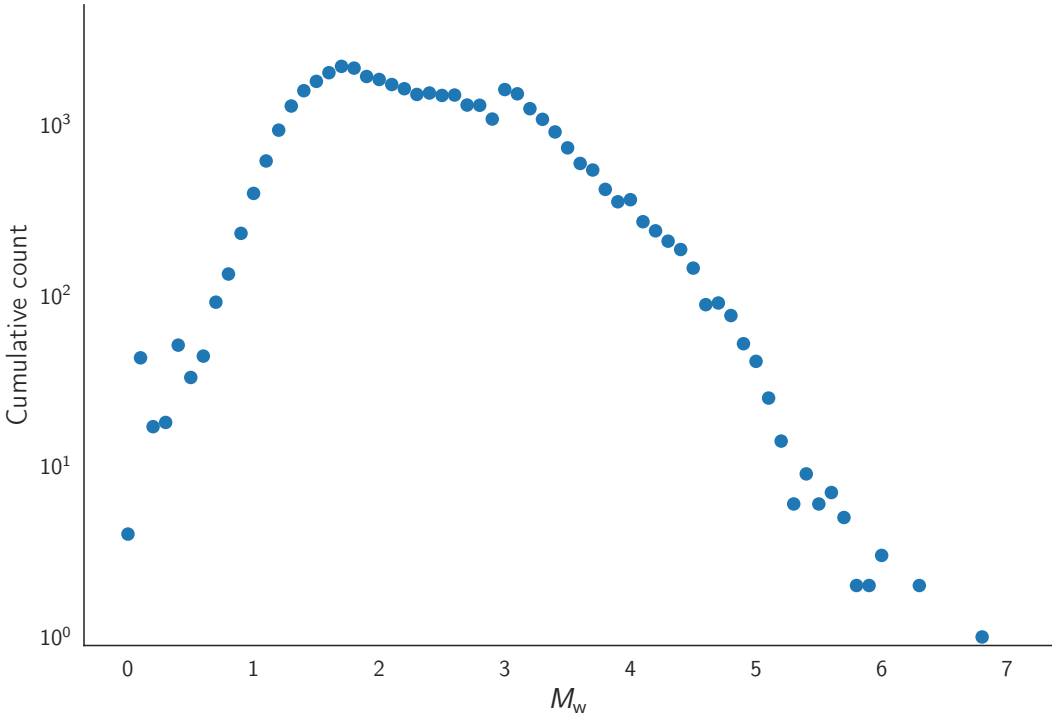


Figure 4.4: The magnitude-frequency relationship of the seismic record in southern Africa from the ISC Bulletin (Storchak et al., 2020, 2017).

are ignored there is notably more activity in the coastal plains of Mozambique and the northeast-southwest trending mobile belts in the north of Zimbabwe and Botswana than the interior of the Zimbabwe Craton, suggesting that rheology may play a significant role.

Overall then, the evidence presented in this thesis suggests that stress due to gravitational collapse is likely the driving source of stress in southern Africa, but that the orientation of preexisting weak zones is the most important in controlling the orientation of strain, a conclusion which is supported by previous work in the region (Craig et al., 2011; Morley, 2010; Williams et al., 2019). Bird et al. (2006) propose that stress in the western area of southern Africa (the Wegener stress anomaly) arises from resistance to the motions of the African and Somalian plates. The results from this analysis suggests that the cause is gravitational collapse, however this stress source was not modelled by Bird et al. (2006), and thus it could still be a combination of both factors.

4.3.1 Limitations and further study

There are several sources of stress which this analysis does not account for—most notably traction forces at the base of the lithosphere, and ridge push from the surrounding plate boundaries, which have been hypothesised as potential causes of southern African stress (Bird et al., 2006). A more thorough investigation could account for these forces, but would require a far more advanced modelling approach. This work also assumes Airy isostatic equilibrium. It is possible that, for example, thermal erosion of the lithosphere has changed the regime in some places to include Pratt isostasy, or that dynamic uplift is occurring. Obviously, if the topographic features of southern Africa are *not* supported by Airy isostasy, it will worsen the accuracy in the calculation of the magnitude of the associated body stresses. It should however be noted that while the magnitude of the gradients in gravitational energy would change if a different form of support were to be

assumed, the direction of the gradients should remain the same. So, for example, if dynamic support were to be assumed, larger gradients (and so larger magnitude stresses) would be expected, but their orientation would remain unchanged. This means the qualitative comparisons performed here would produce similar results. Obviously a composite model including a range of different mechanisms of support would complicate this to some extent. Other processes may also be important. For example, Jackson and McKenzie (2022) model the relationship between gravity anomalies with no topographic expression and earthquakes in Australia, and a similar modelling approach could be applied in southern Africa in the future.

4.4 Conclusions

The theoretical and quantitative analysis in this work suggests that, of the six interpolation methods proposed by Shen et al. (2015), the Gaussian-Azimuthal, inverse square-Voronoi, inverse square only, and inverse square-azimuthal methods can be easily discounted as unfit for purpose. The remaining two methods (Gaussian-Voronoi and Gaussian only) perform well according to this analysis, but despite the theoretical advantages of including a sophisticated density weighting, no significant differences in performance between these two methods were found. Future works may consider implementing the extensions of these interpolation methods to a spherical geometry which were presented in this thesis.

Previous thin sheet modelling studies have aimed to allow a very broad comparison between stresses and strain rates across large areas. They have made sweeping assumptions, often including very simplified rheological models, and have aimed to demonstrate the broad congruence between expected viscous flow in the mid-lower crust and upper mantle and the deformation of the elastic crust (England & Houseman, 1986), or to estimate the average mantle rheology over large areas (England, Houseman & Nocquet, 2016; Walters, England & Houseman, 2017). The comparatively small and gradual changes associated with using the coordinate system of a single UTM zone over such large areas are very unlikely to make a meaningful difference at these resolutions. However, as the density and quality of data grows further, and in turn there are more detailed models which aim to discriminate variations in rheology, these improvements may become more important. Since integrating multiple UTM zones together requires significantly more effort than using the spherical method presented here, this formulation may be a viable alternative when modelling over areas significantly larger than a single UTM zone.

The analysis presented in this work has found that in southern Africa there is very good alignment between the orientation of WSM measurements and the orientation of ancient mobile belts, and that in a few key locations these orientations also match the modelled orientation of stress due to gravitational collapse (Figure 3.15). Such places include the Hebron fault in southwestern Namibia, where 2D-σ_1 for the GPE derived deviatoric stress is aligned to within 5° of 2D-σ_1 of the fault (Salomon et al., 2022; White et al., 2009); parts of the southwest branch of the East African Rift (Daly et al., 2020); and the southwest edge of the Kaapvaal Craton. It also appears that the seismic moment release in southern Africa (Figure 3.16) is dominated by mining (Brandt et al., 2005), and a few key historical earthquakes: M_{w} 6.5 Moiyabana (Kolawole et al., 2017); M_{w} 6.3 Ceres-Tulbagh (Green & Bloch, 1971); and M_{w} 5.8, Koffiefontein. When mining areas are excluded, there is notably more activity in the coastal plains of Mozambique and the northeast-southwest trending mobile belts in the north of Zimbabwe and Botswana than the interior of the Zimbabwe Craton.

Overall, this suggests that stress due to gravitational collapse is likely a driving source of stress in southern Africa, but that the orientation of preexisting weak zones is the most important factor controlling the orientation of strain, a conclusion which is supported by previous work in the region (Craig et al., 2011; Morley, 2010; Williams et al., 2019). Bird et al. (2006) propose that stress in the western area of southern Africa (the Wegener stress anomaly) arises from resistance to the motions of the African and Somalian plates. The results from this analysis suggests that the cause is gravitational collapse, however this stress source was not modelled by Bird et al. (2006), and thus it could still be a combination of both factors. It remains an open question if areas that have increased GPE derived deviatoric stress which is also aligned with the orientation of weak zones will have elevated strain in the long term.

References

Airy, G.B. (1855). III. On the computation of the effect of the attraction of mountain-masses, as disturbing the apparent astronomical latitude of stations in geodetic surveys. *Philosophical Transactions of the Royal Society of London* 145:101–104. DOI: 10.1098/rstl.1855.0003.

Amante, C. & Eakins, B.W. (2009). *ETOPO1 1 Arc-Minute Global Relief Model: Procedures, data sources and analysis*. NOAA Technical Memorandum NESDIS NGDC-24. National Geophysical Data Center, NOAA. DOI: 10.7289/V5C8276M. (Visited on 19/10/2018).

Anderson, E.M. (1905). The dynamics of faulting. *Transactions of the Edinburgh Geological Society* 8(3):387–402. DOI: 10.1144/transed.8.3.387.

Artyushkov, E.V. (1973). Stresses in the lithosphere caused by crustal thickness inhomogeneities. *Journal of Geophysical Research* 78(32):7675–7708. DOI: 10.1029/JB078i032p07675.

Beanland, S. & Haines, J. (1998). The kinematics of active deformation in the North Island, New Zealand, determined from geological strain rates. *New Zealand Journal of Geology and Geophysics* 41(4):311–323. DOI: 10.1080/00288306.1998.9514813.

Bird, P., Ben-Avraham, Z., Schubert, G., Andreoli, M. & Viola, G. (2006). Patterns of stress and strain rate in southern Africa. *Journal of Geophysical Research: Solid Earth* 111(B8). DOI: 10.1029/2005JB003882.

Bird, P. & Piper, K. (1980). Plane-stress finite-element models of tectonic flow in southern California. *Physics of the earth and planetary interiors* 21(2-3):158–175. DOI: 10.1016/0031-9201(80)90067-9.

Bodin, P. & Horton, S. (2004). Source parameters and tectonic implications of aftershocks of the M_w 7.6 Bhuj earthquake of 26 January 2001. *Bulletin of the Seismological Society of America* 94(3):818–827. DOI: 10.1785/0120030176.

Bondár, I., Engdahl, E.R., Villaseñor, A., Harris, J. & Storchak, D. (2015). ISC-GEM: Global Instrumental Earthquake Catalogue (1900–2009), II. Location and seismicity patterns. *Physics of the Earth and Planetary Interiors* 239:2–13. DOI: 10.1016/j.pepi.2014.06.002.

Brandt, M.B.C., Bejaichund, B., Kgaswane, E.M., Hattingh, E. & Roblin, D.L. (2005). Seismic history of southern Africa. *Seismological series of the Council for Geoscience*.

Brent, R.P. (1973). *Algorithms for Minimization Without Derivatives*. Prentice-Hall. Chap. Chapter 4: An Algorithm with Guaranteed Convergence for Finding a Zero of a Function. DOI: 10.1109/TAC.1974.1100629.

Calais, E., Camelbeeck, T., Stein, S., Liu, M. & Craig, T.J. (2016). A new paradigm for large earthquakes in stable continental plate interiors. *Geophysical Research Letters* 43(20):10–621. DOI: 10.1002/2016GL070815.

Calais, E., Freed, A., Van Arsdale, R. & Stein, S. (2010). Triggering of New Madrid seismicity by late-Pleistocene erosion. *Nature* 466(7306):608–611. DOI: 10.1038/nature09258.

Calais, E. & Stein, S. (2009). Time-variable deformation in the New Madrid seismic zone. *Science* 323(5920):1442–1442. DOI: 10.1126/science.1168122.

Caroli, M., Castro, P.M.M. de, Loriot, S., Rouiller, O., Teillaud, M. & Wormser, C. (2010). Robust and Efficient Delaunay Triangulations of Points on Or Close to a Sphere. In *Experimental Algorithms*. Ed. by P. Festa. Berlin, Heidelberg: Springer Berlin Heidelberg, 462–473. DOI: 10.1007/978-3-642-13193-6_39.

Cavalié, O., Lasserre, C., Doin, M.-P., Peltzer, G., Sun, J., Xu, X. & Shen, Z.-K. (2008). Measurement of interseismic strain across the Haiyuan fault (Gansu, China), by InSAR. *Earth and Planetary Science Letters* 275(3-4):246–257. DOI: 10.1016/j.epsl.2008.07.057.

Chambodut, A., Panet, I., Mandea, M., Diament, M., Holschneider, M. & Jamet, O. (2005). Wavelet frames: An alternative to spherical harmonic representation of potential fields. *Geophysical Journal International* 163(3):875–899. DOI: 10.1111/j.1365-246X.2005.02754.x.

Coats, R.R. (1950). *Volcanic activity in the Aleutian Arc*. US Government Printing Office.

Corner, B. & Durrheim, R.J. (2018). An integrated geophysical and geological interpretation of the southern African lithosphere. In *Geology of Southwest Gondwana*. Springer, 19–61. DOI: 10.1007/978-3-319-68920-3_2.

Craig, T.J., Calais, E., Fleitout, L., Bollinger, L. & Scotti, O. (2016). Evidence for the release of long-term tectonic strain stored in continental interiors through intraplate earthquakes. *Geophysical Research Letters* 43(13):6826–6836. DOI: 10.1002/2016GL069359.

Craig, T.J., Copley, A. & Jackson, J. (2014). A reassessment of outer-rise seismicity and its implications for the mechanics of oceanic lithosphere. *Geophysical Journal International* 197(1):63–89. DOI: 10.1093/gji/ggu013.

Craig, T.J., Jackson, J.A., Priestley, K. & McKenzie, D.P. (2011). Earthquake distribution patterns in Africa: Their relationship to variations in lithospheric and geological structure, and their rheological implications. *Geophysical Journal International* 185(1):403–434. DOI: 10.1111/j.1365-246X.2011.04950.x.

Dal Zilio, L., Dinther, Y. van, Gerya, T.V. & Pranger, C.C. (2018). Seismic behaviour of mountain belts controlled by plate convergence rate. *Earth and Planetary Science Letters* 482:81–92. DOI: 10.1016/j.epsl.2017.10.053.

Daly, M.C., Green, P., Watts, A.B., Davies, O., Chibesakunda, F. & Walker, R. (2020). Tectonics and landscape of the Central African Plateau and their implications for a propagating Southwestern Rift in Africa. *Geochemistry, Geophysics, Geosystems* 21(6):e2019GC008746. DOI: 10.1029/2019GC008746.

Elsasser, W.M. (1971). Sea-floor spreading as thermal convection. *Journal of Geophysical Research* 76(5):1101–1112. DOI: 10.1029/JB076i005p01101.

England, P.C. & Houseman, G.A. (1986). Finite strain calculations of continental deformation: 2. Comparison with the India-Asia collision zone. *Journal of Geophysical Research: Solid Earth* 91(B3):3664–3676. DOI: 10.1029/JB091iB03p03664.

England, P.C., Houseman, G.A. & Nocquet, J.-M. (2016). Constraints from GPS measurements on the dynamics of deformation in Anatolia and the Aegean. *Journal of Geophysical Research: Solid Earth* 121(12):8888–8916. DOI: 10.1002/2016JB013382.

England, P.C. & Jackson, J. (2011). Uncharted seismic risk. *Nature Geoscience* 4(6):348–349. DOI: 10.1038/ngeo1168.

England, P.C. & McKenzie, D.P. (1982). A thin viscous sheet model for continental deformation. *Geophysical Journal International* 70(2):295–321. DOI: 10.1111/j.1365-246X.1982.tb04969.x.

Figueira, J.P. (2018). *The Concave Hull: Creating a cluster border using a K-nearest neighbors approach.* URL: https://towardsdatascience.com/the-concave-hull-c649795c0f0f.

Forsyth, D. & Uyeda, S. (1975). On the relative importance of the driving forces of plate motion. *Geophysical Journal International* 43(1):163–200. DOI: 10.1111/j.1365-246X.1975.tb00631.x.

Ghosh, A., Holt, W.E. & Flesch, L.M. (2009). Contribution of gravitational potential energy differences to the global stress field. *Geophysical Journal International* 179(2):787–812. DOI: 10.1111/j.1365-246X.2009.04326.x.

Grafarend, E.W. (1986). Three-dimensional deformation analysis: global vector spherical harmonic and local finite element representation. *Tectonophysics* 130(1-4):337–359. DOI: 10.1016/0040-1951(86)90124-1.

Green, R.W.E. & Bloch, S. (1971). The Ceres, South Africa, earthquake of September 29, 1969: I. Report on some aftershocks. *Bulletin of the Seismological Society of America* 61(4):851–859. DOI: 10.1785/BSSA0610040851.

Gutenberg, B. & Richter, C.F. (1954). *Seismicity of the earth and associated phenomena.* Princeton University Press.

Hackl, M., Malservisi, R., Hugentobler, U. & Wonnacott, R. (2011). Estimation of velocity uncertainties from GPS time series: Examples from the analysis of the South African TrigNet network. *Journal of Geophysical Research: Solid Earth* 116(B11). DOI: 10.1029/2010JB008142.

Hales, A.L. (1969). Gravitational sliding and continental drift. *Earth and Planetary Science Letters* 6(1):31–34. DOI: 10.1016/0012-821X(69)90156-3.

Hasterok, D. & Chapman, D.S. (2007). Continental thermal isostasy: 1. Methods and sensitivity. *Journal of Geophysical Research: Solid Earth* 112(B6). DOI: 10.1029/2006JB004663.

Haxby, W.F. & Turcotte, D.L. (1978). On isostatic geoid anomalies. *Journal of Geophysical Research: Solid Earth* 83(B11):5473–5478. DOI: 10.1029/JB083iB11p05473.

Heezen, B.C. (1960). The rift in the ocean floor. *Scientific American* 203(4):98–114.

Heidbach, O., Rajabi, M., Cui, X., Fuchs, K., Müller, B., Reinecker, J., Reiter, K., Tingay, M. et al. (2018). The World Stress Map database release 2016: Crustal stress pattern across scales. *Tectonophysics* 744:484–498. DOI: 10.1016/j.tecto.2018.07.007.

Heidbach, O., Tingay, M., Barth, A., Reinecker, J., Kurfeß, D. & Müller, B. (2010). Global crustal stress pattern based on the World Stress Map database release 2008. *Tectonophysics* 482(1-4):3–15. DOI: 10.1016/j.tecto.2009.07.023.

Hirschberg, H.P., Lamb, S. & Savage, M.K. (2019). Strength of an obliquely convergent plate boundary: Lithospheric stress magnitudes and viscosity in New Zealand. *Geophysical Journal International* 216(2):1005–1024. DOI: 10.1093/gji/ggy477.

Hofstra, N., Haylock, M., New, M., Jones, P. & Frei, C. (2008). Comparison of six methods for the interpolation of daily, European climate data. *Journal of Geophysical Research: Atmospheres* 113(D21). DOI: 10.1029/2008JD010100.

Hough, S.E., Armbruster, J.G., Seeber, L. & Hough, J.F. (2000). On the modified Mercalli intensities and magnitudes of the 1811–1812 New Madrid earthquakes. *Journal of Geophysical Research: Solid Earth* 105(B10):23839–23864. DOI: 10.1029/2000JB900110.

Houseman, G.A. & England, P.C. (1986). Finite strain calculations of continental deformation: 1. Method and general results for convergent zones. *Journal of Geophysical Research: Solid Earth* 91(B3):3651–3663. DOI: 10.1029/JB091iB03p03651.

Jackson, J. & McKenzie, D.P. (2022). The exfoliation of cratonic Australia in earthquakes. *Earth and Planetary Science Letters* 578:117305. DOI: 10.1016/j.epsl.2021.117305.

Jacoby, W.R. (1970). Instability in the upper mantle and global plate movements. *Journal of Geophysical Research* 75(29):5671–5680. DOI: 10.1029/JB075i029p05671.

Khodaverdian, A., Zafarani, H. & Rahimian, M. (2015). Long term fault slip rates, distributed deformation rates and forecast of seismicity in the Iranian Plateau. *Tectonics* 34(10):2190–2220. DOI: 10.1002/2014TC003796.

Kolawole, F., Atekwana, E.A., Malloy, S., Stamps, D.S., Grandin, R., Abdelsalam, M.G., Leseane, K. & Shemang, E.M. (2017). Aeromagnetic, gravity, and Differential Interferometric Synthetic Aperture Radar analyses reveal the causative fault of the 3 April 2017 M_w 6.5 Moiyabana, Botswana, earthquake. *Geophysical Research Letters* 44(17):8837–8846. DOI: 10.1002/2017GL074620.

Lamb, S. (2015). Kinematics to dynamics in the New Zealand Plate boundary zone: Implications for the strength of the lithosphere. *Geophysical Journal International* 201(2):552–573. DOI: 10.1093/gji/ggv027.

Liu, M. & Stein, S. (2016). Mid-continental earthquakes: Spatiotemporal occurrences, causes, and hazards. *Earth-Science Reviews* 162:364–386. DOI: 10.1016/j.earscirev.2016.09.016.

Lliboutry, L. (1969). Sea-floor spreading, continental drift and lithosphere sinking with an asthenosphere at melting point. *Journal of Geophysical Research* 74(27):6525–6540. DOI: 10.1029/JB074i027p06525.

McKenzie, D.P. (1966). The viscosity of the lower mantle. *Journal of Geophysical Research* 71(16):3995–4010. DOI: 10.1029/JZ071i016p03995.

— (1969). The relation between fault plane solutions for earthquakes and the directions of the principal stresses. *Bulletin of the Seismological Society of America* 59(2):591–601. DOI: 10.1785/BSSA0590020591.

— (1972). Active tectonics of the Mediterranean region. *Geophysical Journal International* 30(2):109–185. DOI: 10.1111/j.1365-246X.1972.tb02351.x.

— (1976). Can plate tectonics describe continental deformation? In *International Symposium on the Structural History of the Mediterranean Basins*. Éditions TECHNIP, Paris, 189–196.

McKenzie, D.P. & Parker, R.L. (1967). The North Pacific: An example of tectonics on a sphere. *Nature* 216(5122):1276–1280. DOI: 10.1038/2161276a0.

Molnar, P. & Tapponnier, P. (1975). Cenozoic tectonics of Asia: Effects of a continental collision. *Science* 189(4201):419–426.

Mooney, W.D., Ritsema, J. & Hwang, Y.K. (2012). Crustal seismicity and the earthquake catalog maximum moment magnitude M_{cmax} in stable continental regions (SCRs): Correlation with the seismic velocity of the lithosphere. *Earth and planetary science letters* 357:78–83. DOI: 10.1016/j.epsl.2012.08.032.

Morgan, W.J. (1972). Deep mantle convection plumes and plate motions. *AAPG bulletin* 56(2):203–213. DOI: 10.1306/819A3E50-16C5-11D7-8645000102C1865D.

Morley, C.K. (2010). Stress re-orientation along zones of weak fabrics in rifts: An explanation for pure extension in 'oblique' rift segments? *Earth and Planetary Science Letters* 297(3-4):667–673. DOI: 10.1016/j.epsl.2010.07.022.

Muir, J.B. & Tkalčić, H. (2015). A method of spherical harmonic analysis in the geosciences via hierarchical Bayesian inference. *Geophysical Journal International* 203(2):1164–1171. DOI: 10.1093/gji/ggv361.

Orowan, E. (1964). Continental drift and the origin of mountains. *Science* 146(3647):1003–1010.

Pascal, B. (1663). *Traitez de l'équilibre des liqueurs et de la pesanteur de la masse de l'air*. Paris: Chez Guillaume Desprez.

Pigafetta, A. (1536). *Primer Viaje en Torno del Globo*. Princeton University Press.

Plattner, A. & Simons, F.J. (2014). Spatiospectral concentration of vector fields on a sphere. *Applied and Computational Harmonic Analysis* 36(1):1–22. DOI: 10.1016/j.acha.2012.12.001.

Pratt, J.H. (1855). I. On the attraction of the Himalaya Mountains, and of the elevated regions beyond them, upon the plumb-line in India. *Philosophical Transactions of the Royal Society of London* 145:53–100. DOI: 10.1098/rstl.1855.0002.

Rey, P., Vanderhaeghe, O. & Teyssier, C. (2001). Gravitational collapse of the continental crust: Definition, regimes and modes. *Tectonophysics* 342(3-4):435–449. DOI: 10.1016/S0040-1951(01)00174-3.

Royden, L.H. (1993). The tectonic expression slab pull at continental convergent boundaries. *Tectonics* 12(2):303–325. DOI: 10.1029/92TC02248.

Ruegg, J.C., Rudloff, A., Vigny, C., Madariaga, R., De Chabalier, J.B., Campos, J., Kausel, E., Barrientos, S. et al. (2009). Interseismic strain accumulation measured by GPS in the seismic gap between Constitución and Concepción in Chile. *Physics of the Earth and Planetary Interiors* 175(1-2):78–85. DOI: 10.1016/j.pepi.2008.02.015.

Runcorn, S.K. (1962). Towards a theory of continental drift. *Nature* 193(4813):311–314. DOI: 10.1038/193311a0.

— (1965). Palaeomagnetic comparisons between Europe and North America. *Philosophical Transactions of the Royal Society of London. Series A, Mathematical and Physical Sciences* 258(1088):1–11.

Salomon, G.W., New, T., Muir, R.A., Whitehead, B., Scheiber-Enslin, S., Smit, J., Stevens, V., Kahle, B. et al. (2022). Geomorphological and geophysical analyses of the Hebron fault, SW Namibia: implications for stable continental region seismic hazard. *Geophysical Journal International*. DOI: 10.1093/gji/ggab466.

Saria, E., Calais, E., Altamimi, Z., Willis, P. & Farah, H. (2013). A new velocity field for Africa from combined GPS and DORIS space geodetic solutions: Contribution to the definition of the African reference frame (AFREF). *Journal of Geophysical Research: Solid Earth* 118(4):1677–1697. DOI: 10.1002/jgrb.50137.

Sbar, M.L., Engelder, T., Plumb, R. & Marshak, S. (1979). Stress pattern near the San Andreas fault, Palmdale, California, from near-surface in situ measurements. *Journal of Geophysical Research: Solid Earth* 84(B1):156–164. DOI: 10.1029/JB084iB01p00156.

Selvaggi, G. (1998). Spatial distribution of horizontal seismic strain in the Apennines from historical earthquakes. *Annals of Geophysics* 41(2). DOI: 10.4401/ag-4334.

Shen, Z.-K., Wang, M., Zeng, Y. & Wang, F. (2015). Optimal interpolation of spatially discretized geodetic data. *Bulletin of the Seismological Society of America*. DOI: 10.1785/0120140247.

Simons, F.J., Dahlen, F.A. & Wieczorek, M.A. (2006). Spatiospectral concentration on a sphere. *SIAM review* 48(3):504–536. DOI: 10.1137/S0036144504445765.

Smit, L., Fagereng, Å., Braeuer, B. & Stankiewicz, J. (2015). Microseismic activity and basement controls on an active intraplate strike-slip fault, Ceres–Tulbagh, South Africa. *Bulletin of the Seismological Society of America* 105(3):1540–1547. DOI: 10.1785/0120140262.

Spence, W. (1987). Slab pull and the seismotectonics of subducting lithosphere. *Reviews of Geophysics* 25(1):55–69. DOI: 10.1029/RG025i001p00055.

Spivak, M. (1999). *A Comprehensive Introduction to Differential Geometry*. 3rd ed. Vol. 2. Publish or Perish.

Stauder, W. (1968). Mechanism of the Rat Island earthquake sequence of February 4, 1965, with relation to island arcs and sea-floor spreading. *Journal of Geophysical Research* 73(12):3847–3858. DOI: 10.1029/JB073i012p03847.

Storchak, D.A., Harris, J., Brown, L., Lieser, K., Shumba, B. & Di Giacomo, D. (2020). Rebuild of the Bulletin of the International Seismological Centre (ISC)—part 2: 1980–2010. *Geoscience Letters* 7(1):1–21. DOI: 10.1186/s40562-020-00164-6.

Storchak, D.A., Harris, J., Brown, L., Lieser, K., Shumba, B., Verney, R., Di Giacomo, D. & Korger, E.I.M. (2017). Rebuild of the Bulletin of the International Seismological Centre (ISC), part 1: 1964–1979. *Geoscience Letters* 4(1):1–14. DOI: 10.1186/s40562-017-0098-z.

Strasser, F.O., Albini, P., Flint, N.S. & Beauval, C. (2015). Twentieth century seismicity of the Koffiefontein region (Free State, South Africa): Consistent determination of earthquake catalogue parameters from mixed data types. *Journal of Seismology* 19(4):915–934. DOI: 10.1007/s10950-015-9503-2.

Tapponnier, P. & Molnar, P. (1976). Slip-line field theory and large-scale continental tectonics. *Nature* 264(5584):319–324. DOI: 10.1038/264319a0.

Tapponnier, P. & Molnar, P. (1977). Active faulting and tectonics in China. *Journal of Geophysical Research* 82(20):2905–2930. DOI: 10.1029/JB082i020p02905.

Toussaint, G., Burov, E. & Avouac, J.-P. (2004). Tectonic evolution of a continental collision zone: A thermomechanical numerical model. *Tectonics* 23(6). DOI: 10.1029/2003TC001604.

Truswell, J.F. (1977). *The geological evolution of South Africa*. Purnell.

Turcotte, D.L. & Oxburgh, E.R. (1972). Mantle convection and the new global tectonics. *Annual Review of Fluid Mechanics* 4(1):33–66. DOI: 10.1146/annurev.fl.04.010172.000341.

Van Brummelen, G.R. (2013). *Heavenly Mathematics: The Forgotten Art of Spherical Trigonometry*. Princeton University Press. DOI: 10.1515/9781400844807.

Von Buchenröder, W.L. (1830). An account of earthquakes which occurred at the Cape of Good Hope during the month of December 1809 etc. *South African Quarterly Journal*:18–25.

Walters, R.J., England, P.C. & Houseman, G.A. (2017). Constraints from GPS measurements on the dynamics of the zone of convergence between Arabia and Eurasia. *Journal of Geophysical Research: Solid Earth* 122(2):1470–1495. DOI: 10.1002/2016JB013370.

Weinberg, S. (1972). *Gravitation and Cosmology: Principles and Applications of the General Theory of Relativity*. Wiley.

Weisstein, E.W. (2015a). *L'Huilier's Theorem*. URL: http://mathworld.wolfram.com/LHuiliersTheorem.html.

— (2015b). *Voronoi Diagram*. URL: http://mathworld.wolfram.com/VoronoiDiagram.html.

White, S., Stollhofen, H., Stanistreet, I.G. & Lorenz, V. (2009). Pleistocene to recent rejuvenation of the Hebron Fault, SW Namibia. *Geological Society, London, Special Publications* 316(1):293–317. DOI: 10.1144/SP316.18.

Williams, J.N., Fagereng, Å., Wedmore, L.N., Biggs, J., Mphepo, F., Dulanya, Z., Mdala, H. & Blenkinsop, T. (2019). How do variably striking faults reactivate during rifting? Insights from southern Malawi. *Geochemistry, Geophysics, Geosystems* 20(7):3588–3607. DOI: 10.1029/2019GC008219.

Wright, T., Parsons, B. & Fielding, E. (2001). Measurement of interseismic strain accumulation across the North Anatolian Fault by satellite radar interferometry. *Geophysical Research Letters* 28(10):2117–2120. DOI: 10.1029/2000GL012850.

Zarifi, Z., Nilfouroushan, F. & Raeesi, M. (2014). Crustal stress map of Iran: Insight from seismic and geodetic computations. *Pure and Applied Geophysics* 171(7):1219–1236. DOI: 10.1007/s00024-013-0711-9.

Zhang, P.-Z., Shen, Z.-K., Wang, M., Gan, W., Burgmann, R., Molnar, P., Wang, Q., Niu, Z. et al. (2004). Continuous deformation of the Tibetan Plateau from global positioning system data. *Geology* 32(9):809–812. DOI: 10.1130/G20554.1.

Zhao, B., Huang, Y., Zhang, C., Wang, W., Tan, K. & Du, R. (2015). Crustal deformation on the Chinese mainland during 1998–2014 based on GPS data. *Geodesy and Geodynamics* 6(1):7–15. DOI: 10.1016/j.geog.2014.12.006.

Appendices

A Additional figures

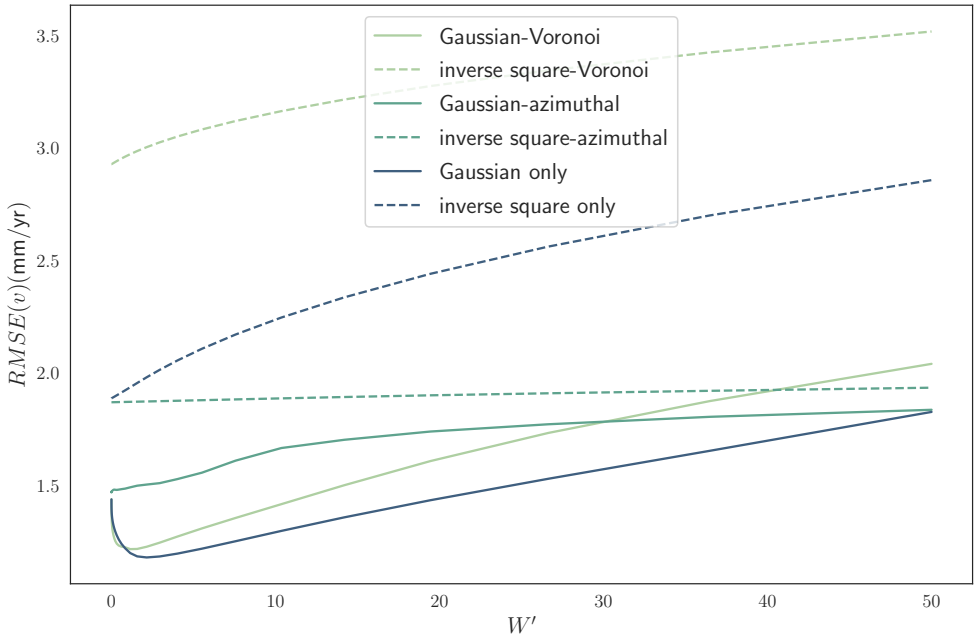


Figure A.1: The mean error $RMSE(\boldsymbol{v})$ as a function of W' for the various weighting schemes (for cross-validation).

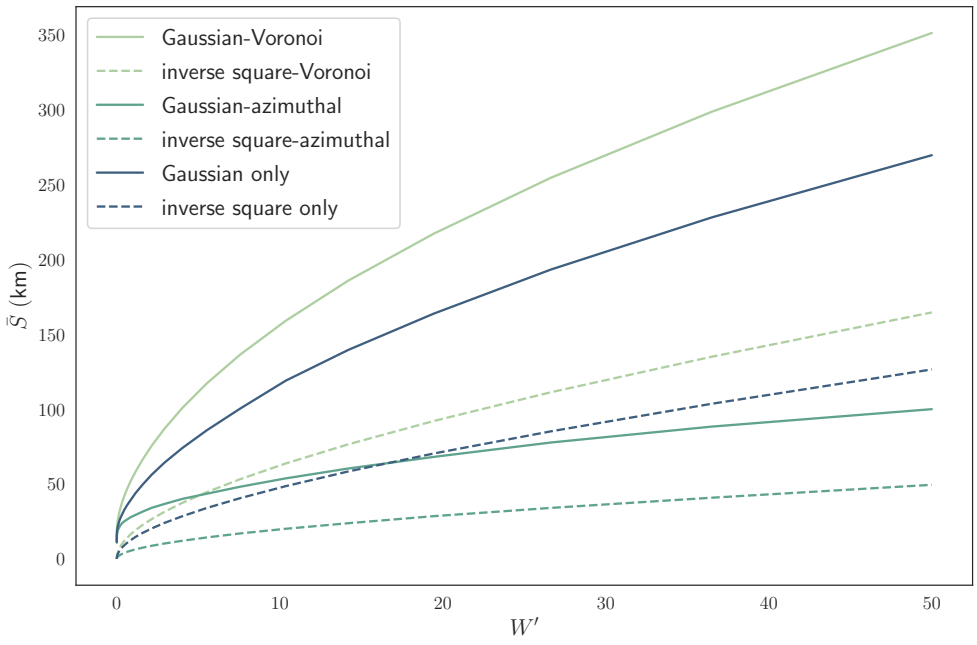


Figure A.2: The change in the average smoothing parameter $\bar{S}$ with respect to W' in the cross-validation test.

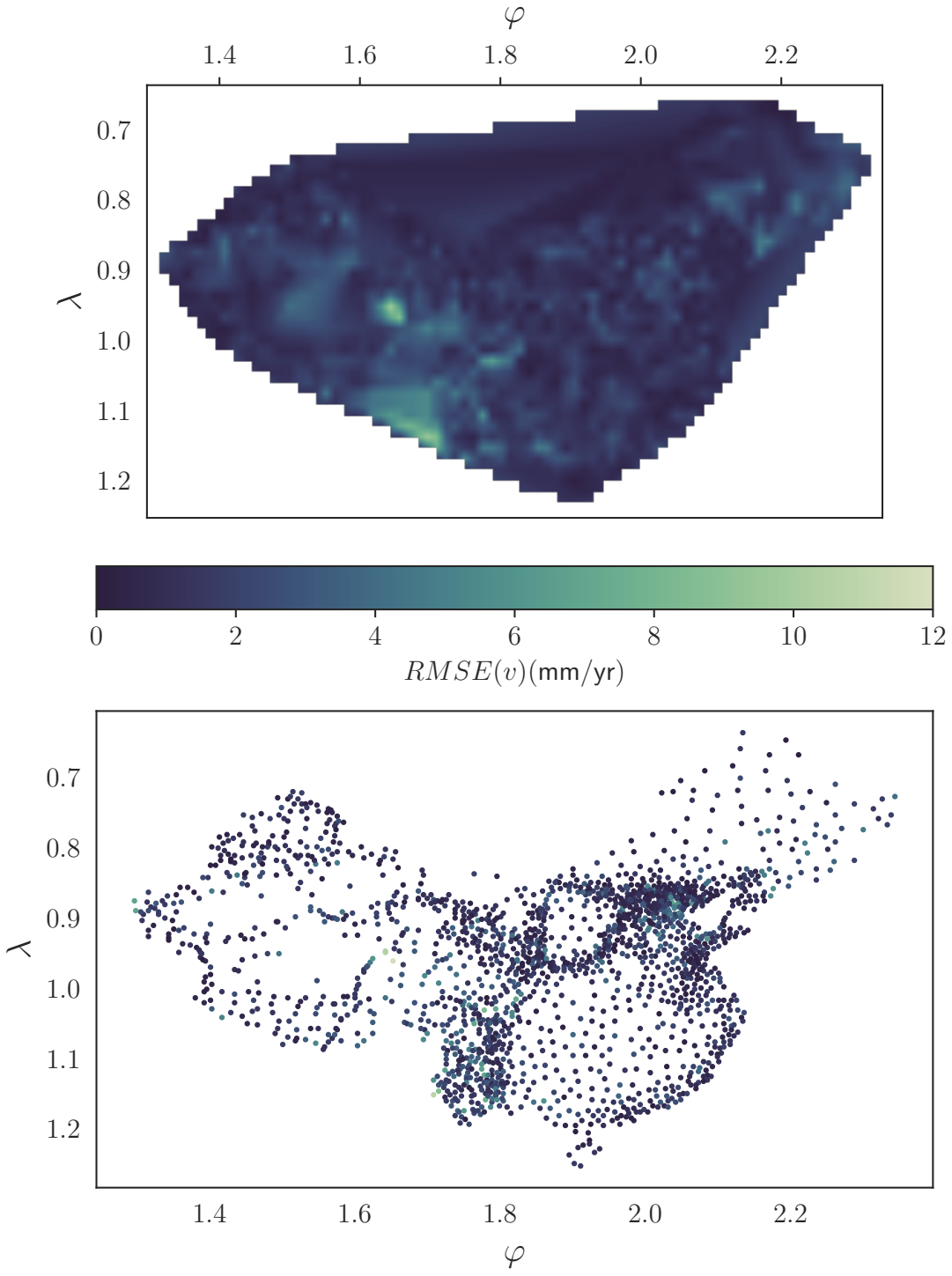


Figure A.3: The magnitude of the vector field of differences $|\Delta \boldsymbol{v}_j|$ for the Gaussian-azimuthal interpolation scheme according to the cross-validation testing method ($W' = 0.492$, $\bar{S} = 28.9\,\mathrm{km}$). The upper plot uses a naïve interpolation method between points for ease of visualisation, while the lower plot shows the true values of $|\Delta \boldsymbol{v}_j|$ at each data point.

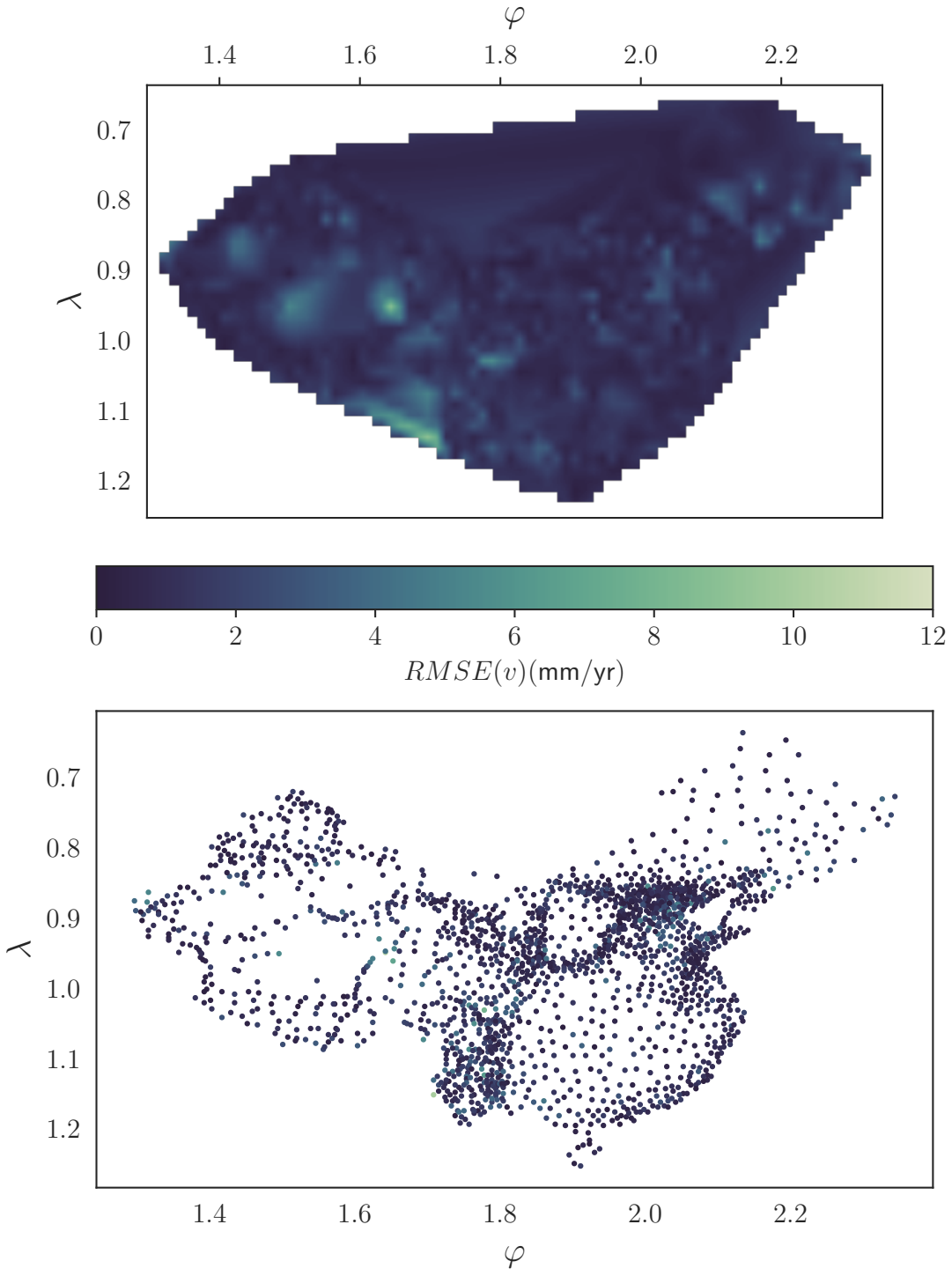


Figure A.4: The magnitude of the vector field of differences $|\Delta \boldsymbol{v}_j|$ for the Gaussian only interpolation scheme according to the cross-validation testing method ($W' = 2.159$, $\bar{S} = 64.3\,\mathrm{km}$). The upper plot uses a naïve interpolation method between points for ease of visualisation, while the lower plot shows the true values of $|\Delta \boldsymbol{v}_j|$ at each data point.

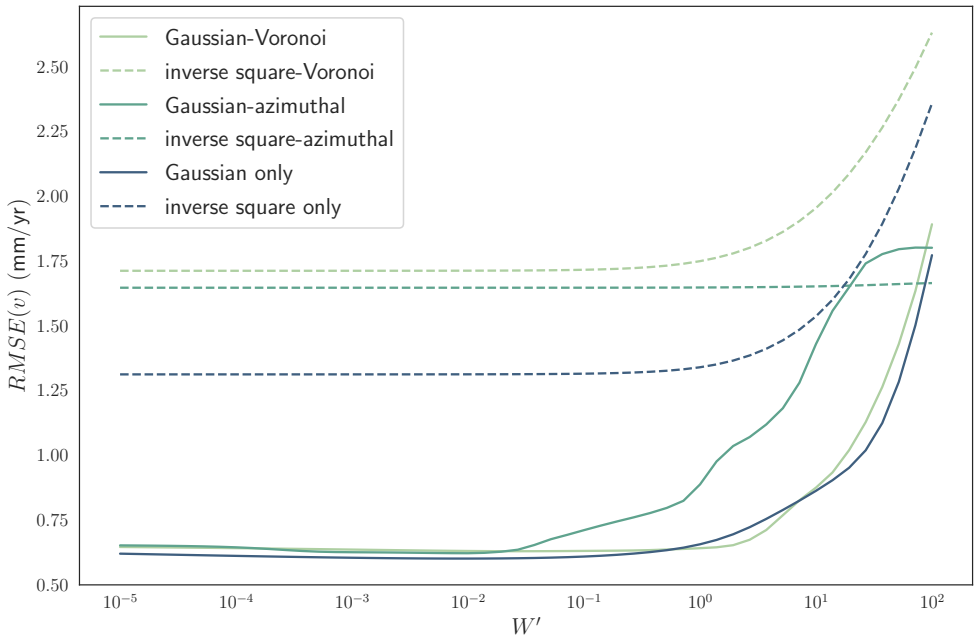


Figure A.5: The mean absolute error $RMSE(\boldsymbol{v})$ as a function of W' for the various weighting schemes (for known-field error).

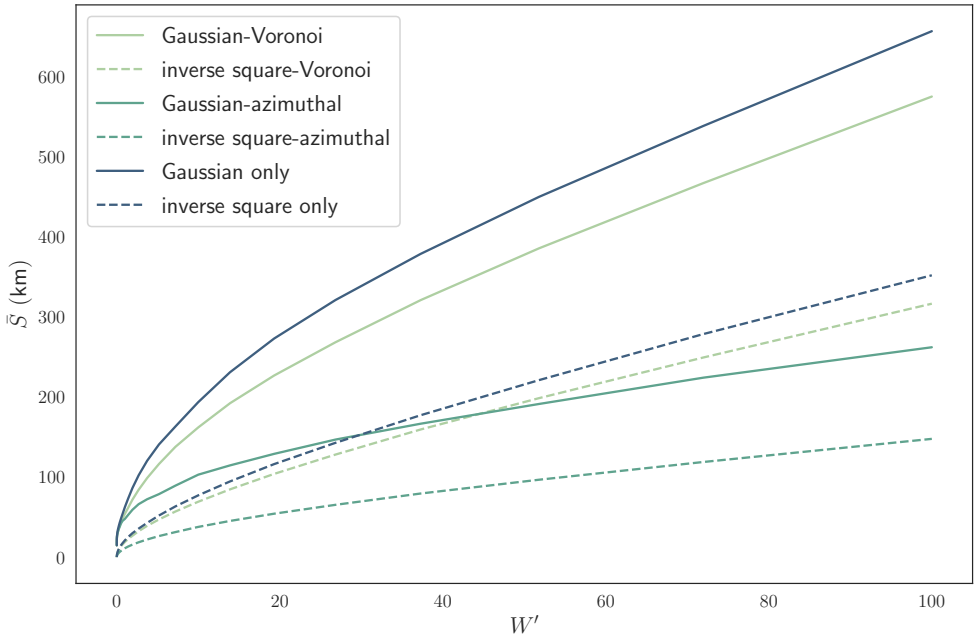


Figure A.6: The change in the average smoothing parameter $\bar{S}$ with respect to W' in the known-field test.

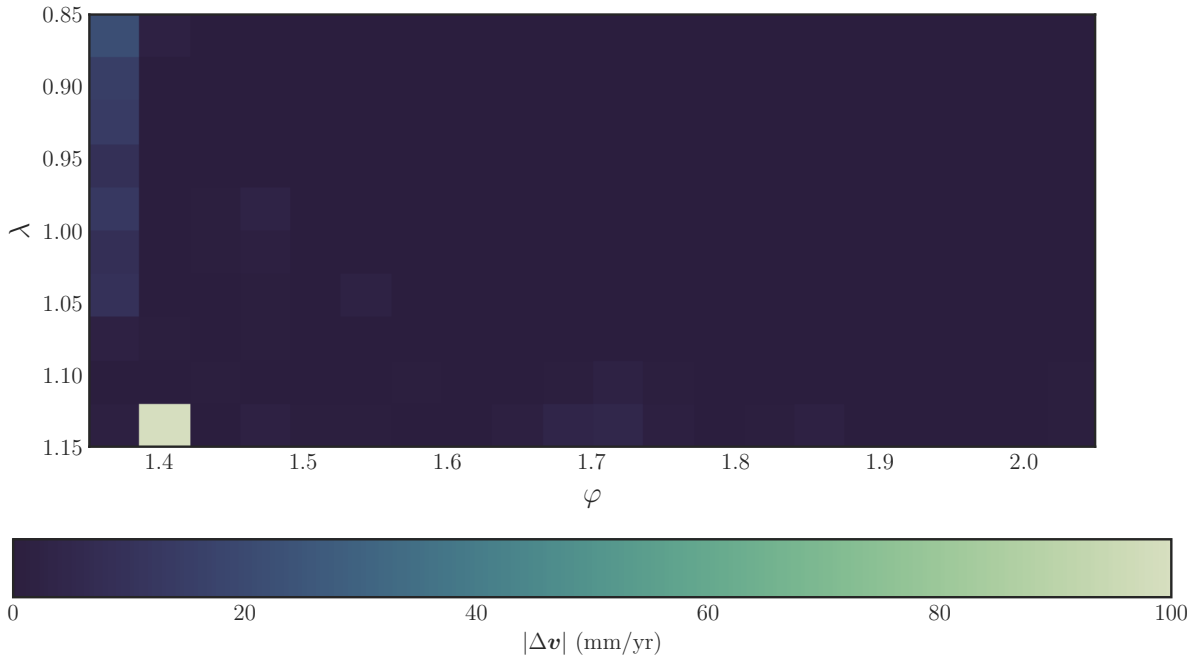


Figure A.7: The magnitude of the vector field of differences $|\Delta \boldsymbol{v}_j|$ for the Gaussian-azimuthal interpolation scheme according to the known-field testing method ($W' = 0.000014$, $\bar{S} = 7.84\,\mathrm{km}$).

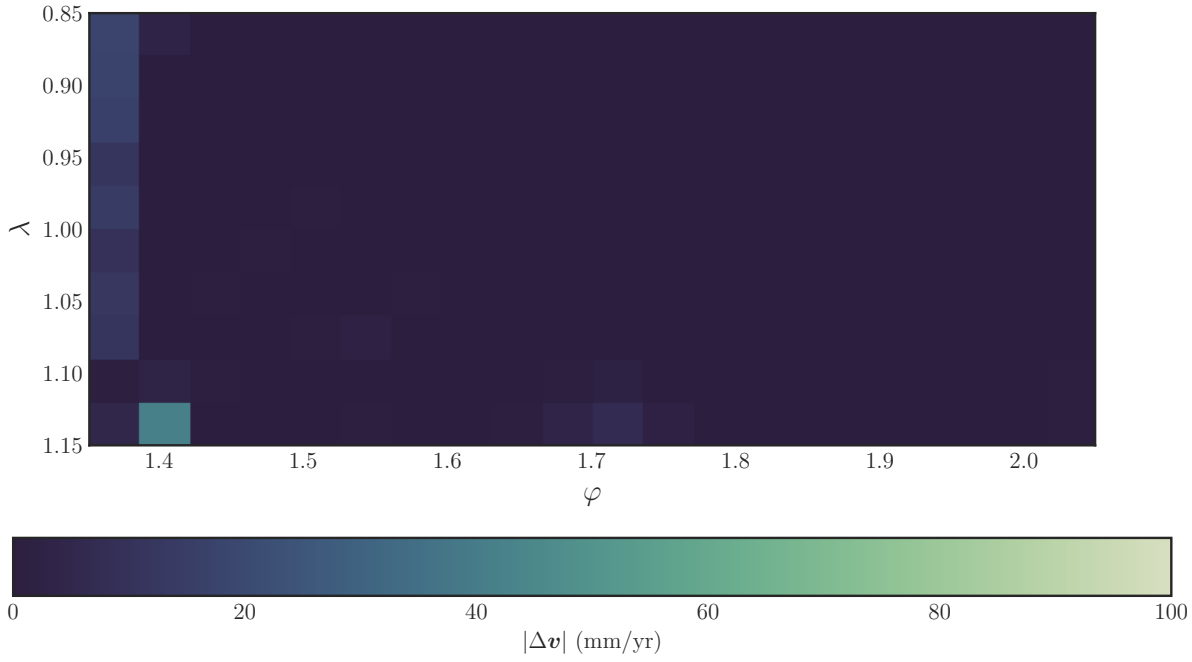


Figure A.8: The magnitude of the vector field of differences $|\Delta \boldsymbol{v}_j|$ for the Gaussian only interpolation scheme according to the known-field testing method ($W' = 0.353$, $\bar{S} = 22.0\,\mathrm{km}$).

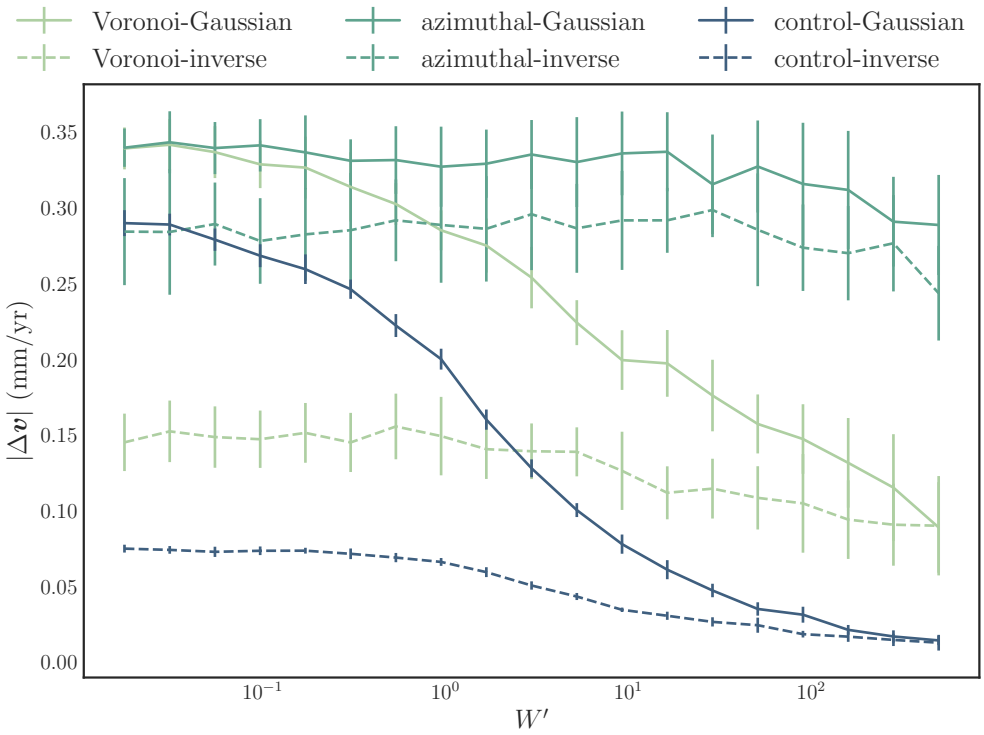


Figure A.9: The mean over 20 runs of the mean absolute error $RMSE(\boldsymbol{v})$ as a function of W' for the various weighting schemes (for noise sensitivity). Shown are the 1σ error bars.

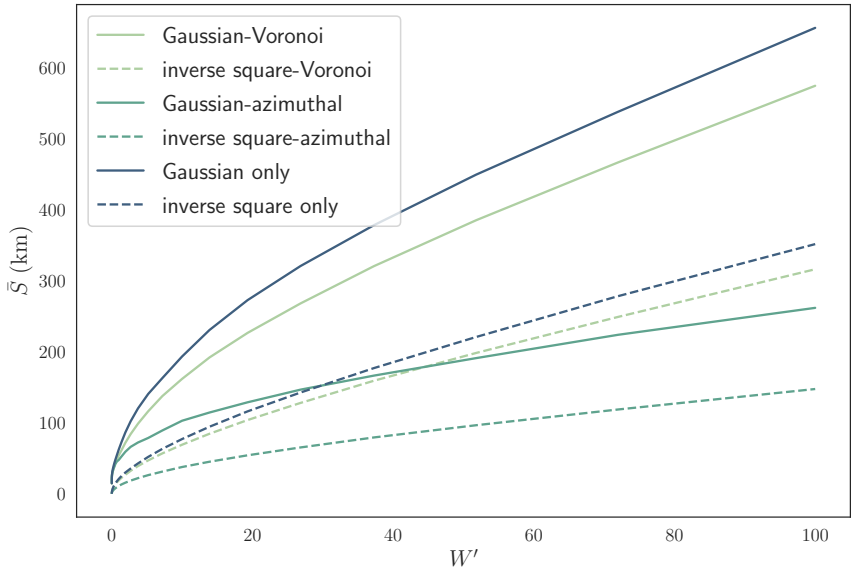


Figure A.10: The change in the average smoothing parameter $\bar{S}$ with respect to W' in the noise sensitivity test.

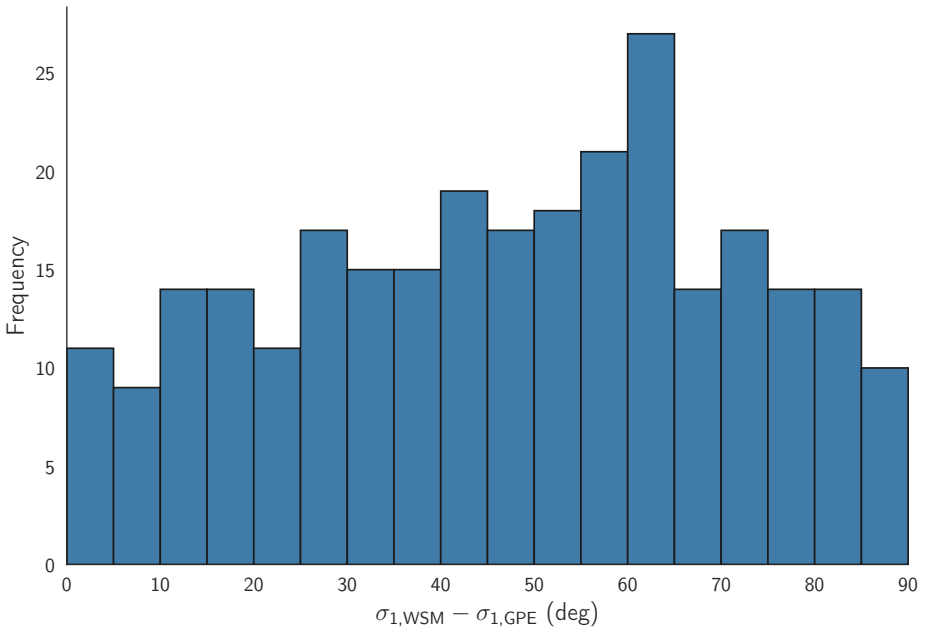


Figure A.11: The distribution of the angles between *in situ* WSM measurements and the principal stress axis due to GPE.

B Python code

B.1 Voronoi cell weighting

```
# other people's stuff
import numpy as np
from scipy.spatial import SphericalVoronoi
from scipy.spatial.distance import pdist
import math

# my stuff
import Coordinates.coordinate_transforms as ct
import Distance.distance as dw
from .edges import Edge
from .edges import Edges

# stuff for graphs
import matplotlib.pyplot as plt
import seaborn as sns
from matplotlib import rc
from mpl_toolkits.mplot3d import Axes3D
from mpl_toolkits.mplot3d.art3d import Poly3DCollection
from mpl_toolkits.mplot3d import proj3d
from matplotlib import colors

def calculate_surface_area_of_spherical_cap(theta, sphere_radius):
    # get area of spherical cap given angle between top of cap and base of
        cap
    return 2 * np.pi * sphere_radius**2 * (1 - np.cos(theta))

def calculate_angle_between_vectors(point_1, point_2):
    cos_theta = np.dot(point_1, point_2) / (np.linalg.norm(point_1) * np.
       linalg.norm(point_2))
    return np.arccos(cos_theta)

def calculate_surface_area_of_a_spherical_Voronoi_polygon(
   array_ordered_Voronoi_polygon_vertices,sphere_radius):
    '''Calculate the surface area of a polygon on the surface of a sphere.
        Based on equation provided here: http://mathworld.wolfram.com/
       LHuiliersTheorem.html
    Decompose into triangles, calculate excess for each'''
    #have to convert to unit sphere before applying the formula
    spherical_coordinates = ct.convert_cartesian_array_to_spherical_array(
       array_ordered_Voronoi_polygon_vertices)
    spherical_coordinates[:,0] = 1.0
    array_ordered_Voronoi_polygon_vertices = ct.
       convert_spherical_array_to_cartesian_array(spherical_coordinates)
    #handle nearly-degenerate vertices on the unit sphere by returning an
       area close to 0 -- may be better options, but this is my current
       solution to prevent crashes, etc.
    #seems to be relatively rare in my own work, but sufficiently common
       to cause crashes when iterating over large amounts of messy data
    if pdist(array_ordered_Voronoi_polygon_vertices).min() < (10 ** -7):
```

```
        return 10 ** -8
    else:
        n = array_ordered_Voronoi_polygon_vertices.shape[0]
        #point we start from
        root_point = array_ordered_Voronoi_polygon_vertices[0]
        totalexcess = 0
        #loop from 1 to n-2, with point 2 to n-1 as other vertex of
           triangle
        # this could definitely be written more nicely
        b_point = array_ordered_Voronoi_polygon_vertices[1]
        root_b_dist = dw.
           calculate_haversine_distances_between_cartesian_points(
           root_point, b_point, 1.0)
        for i in 1 + np.arange(n - 2):
            a_point = b_point
            b_point = array_ordered_Voronoi_polygon_vertices[i+1]
            root_a_dist = root_b_dist
            root_b_dist = dw.
               calculate_haversine_distances_between_cartesian_points(
               root_point, b_point, 1.0)
            a_b_dist = dw.
               calculate_haversine_distances_between_cartesian_points(
               a_point, b_point, 1.0)
            s = (root_a_dist + root_b_dist + a_b_dist) / 2
            totalexcess += 4 * math.atan(math.sqrt( math.tan(0.5 * s) *
               math.tan(0.5 * (s-root_a_dist)) * math.tan(0.5 * (s-
               root_b_dist)) * math.tan(0.5 * (s-a_b_dist))))
        return totalexcess * (sphere_radius ** 2)


def calculate_spherical_Voronoi_weighting(spherical_coord_array,
   edge_points, return_vertices_for_plotting=False):
    '''
    Take shape (N,3) spherical_coord_array and calculates their spherical
       voronoi weightings

    Since points on the edge of the network have poorly defined Voronoi
       cells, the area weighting is instead the area of the spherical cap
       defined by the average distance to neighbouring points.
    '''
    coord_array = np.copy(spherical_coord_array)
    n = len(coord_array)
    # print(coord_array.shape)
    coord_array[:,0] = np.ones_like(coord_array[:,0]) # put the
       coordinates onto the unit sphere
    # print(coord_array.shape)
    opposite_point = -1 * ct.convert_spherical_array_to_cartesian_array(np
       .mean(coord_array, axis=0)) # create a point on the opposite side
       of the sphere
    coord_array = ct.convert_spherical_array_to_cartesian_array(
       coord_array) # from here on we'll only need coord_array in
       cartesian coordinates
    coord_array = np.concatenate( (coord_array, opposite_point[None]) ) #
       add the opposite point to the end of the array
```

```
point_ids = np.zeros(n) # make point id array

# print(coord_array.shape)

centre = np.array([0, 0, 0])
radius = 1.0
sv = SphericalVoronoi(coord_array, radius, centre)
sv.sort_vertices_of_regions()
regions = np.copy(sv.regions)
areas = np.empty(n)
radii = np.zeros(n)

number_of_border_points = len(edge_points[edge_points == 1.0])
number_of_interior_points = len(edge_points[edge_points == 0.0])
number_of_cap_points = 0

coordinates_2D = spherical_coord_array[:,1:]
vertices_2D = ct.convert_cartesian_array_to_spherical_array(sv.
   vertices)[:,2:0:-1]
# check whether each voronoi cell is a border or interior point
for p, i in zip(regions, range(n)):
    voronoi_area =
       calculate_surface_area_of_a_spherical_Voronoi_polygon(sv.
       vertices[p], sphere_radius=1.0)
    p = Edges(vertices_2D[p]) # get the edges of the voronoi region
    areas[i] = voronoi_area

    # if the ith point is an edge point, determine if the spherical
       cap area is necessary
    if edge_points[i]:
        # find the other data points which share an edge with this
           data point and where the straight line joining the points
           intersects the shared edge
        shared_edge_indices = []
        neighbours = []
        for q, j in zip(regions, range(n)):

            q = Edges(vertices_2D[q]) # get the edges of the jth
               region
            shared_edge = p.share_edge(q, return_edge=True) # check if
                an edge is shared
            shared_edge_indices.append(j)
            if shared_edge:
                # # if an edge is shared, check whether it intersects
                   the connecting line
                # connecting_line = Edge.from_coordinates(
                   coordinates_2D[i][1], coordinates_2D[i][0],
                   coordinates_2D[j][1], coordinates_2D[j][0])

                # if shared_edge.intersect(connecting_line):
                #     # if it intersects the connecting line, it is
                   considered a neighbour
                #     neighbours.append(j)
                neighbours.append(j)
```

```
            neighbour_angles = []
            xy_distances = []
            for j in neighbours:
                neighbour_angles.append(calculate_angle_between_vectors(
                    coord_array[i], coord_array[j]))
                xy_distances.append(np.linalg.norm(coordinates_2D[j] -
                    coordinates_2D[i]))
            cap_area = calculate_surface_area_of_spherical_cap(np.median(
                neighbour_angles)/2, sphere_radius=1.0)

            if voronoi_area > cap_area:
                areas[i] = cap_area
                number_of_cap_points += 1
                point_ids[i] = 1.0
                radii[i] = np.sqrt(cap_area) / np.pi

    print('number␣of␣interior␣points:', number_of_interior_points)
    print('number␣of␣border␣points:', number_of_border_points)
    print('number␣of␣cap␣points:', number_of_cap_points)
    # areas = areas / np.sum(areas)
    # print(np.sum(areas))

    if return_vertices_for_plotting:
        return areas, coord_array[:-1], sv.vertices, sv.regions, point_ids
            , radii

    return areas
```

B.2 Azimuthal weighting

```
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from matplotlib import rc
from mpl_toolkits.mplot3d import Axes3D
from mpl_toolkits.mplot3d.art3d import Poly3DCollection
from mpl_toolkits.mplot3d import proj3d
from matplotlib import colors

def calculate_azimuthal_weights(interpolation_coordinate,
    data_location_coordinates):
    '''
    Finds the azimuthal weight of data_location_coordinates with respect
        to interpolation_coordinate
    '''

    def translate_coords(transl_to, points):
        '''
        Translates the (cartesian) coordinate system of all the points, to
            be centred
        at some point. Returns an array of the translated points.
        '''

        points -= transl_to

        return points


    def rotate_coords(rotate_to, points):
        '''
        Rotates the (cartesian) coordinate system of points, s.t. the z-
            axis
        lines up with the vector to be rotated to. Returns an array of the
            rotated
        points.
        '''

        rotate_to_hat = rotate_to / np.linalg.norm(rotate_to)
        z_hat = np.array([0., 0., 1.])

        omega_hat = np.cross(rotate_to_hat, z_hat)
        theta = np.arccos(np.dot(z_hat, rotate_to_hat))
        omega = omega_hat * theta
        omega_hats = np.ones_like(points) * omega_hat

        # need to check behaviour of numpy.cross with array of vectors
        a = points * np.cos(theta)
        b = np.cross(omega_hat, points) * np.sin(theta)
        c = omega_hats * np.expand_dims(np.tensordot(omega_hat, points,
            axes=(0,1)),0).T * (1 - np.cos(theta))
        rot_points = a + b + c
```

```
    return rot_points

def sort_by_angle(points):
    '''
    Projects points to the (x,y)-plane and sorts them by increasing
       polar angle.
    Returns a direct (the sorted angles) and indirect (indices) sort.
    '''

    angles = np.arctan2(points[:,1], points[:,0])
    sorting_indices = np.argsort(angles)
    sorted_angles = angles[sorting_indices]

    return sorted_angles, sorting_indices

def get_weights(sorted_angles):
    '''
    Finds the azimuthal weighting of each point. Returns an array of
       weights (in
    radians).
    '''

    sorted_angles_rolled = np.roll(sorted_angles, 1)
    diffs = sorted_angles - sorted_angles_rolled
    diffs[0] = sorted_angles[0] + (2*np.pi - sorted_angles[-1])

    below_weights = diffs
    above_weights = np.roll(below_weights, -1)

    weights = (below_weights + above_weights) * len(sorted_angles) /
       (4 * np.pi)

    return weights

interpolation_coordinate = np.copy(interpolation_coordinate)
data_location_coordinates = np.copy(data_location_coordinates)

data_location_coordinates = translate_coords(interpolation_coordinate,
    data_location_coordinates)
data_location_coordinates = rotate_coords(interpolation_coordinate,
   data_location_coordinates)

sorted_data_angles, sorting_indices = sort_by_angle(
   data_location_coordinates)
azimuthal_weights = get_weights(sorted_data_angles)
unsorting_indices = np.argsort(sorting_indices)

return azimuthal_weights[unsorting_indices]
```

B.3 Distance weighting

```
import numpy as np
import math
from scipy.optimize import newton, brentq

import Coordinates.coordinate_transforms as ct

import matplotlib.pyplot as plt
from matplotlib import cm
import seaborn as sns
# import shapely.geometry as geometry
# from descartes import PolygonPatch
from matplotlib import rc

def calculate_haversine_distance_between_spherical_points(
    cartesian_array_1, cartesian_array_2, sphere_radius):
    '''
    Calculate the haversine-based distance between two points on the
       surface of a sphere. Should be more accurate than the arc cosine
       strategy. See, for example: http://en.wikipedia.org/wiki/
       Haversine_formula
    '''
    spherical_array_1 = ct.convert_cartesian_array_to_spherical_array(
       cartesian_array_1)
    spherical_array_2 = ct.convert_cartesian_array_to_spherical_array(
       cartesian_array_2)
    lambda_1 = spherical_array_1[1]
    lambda_2 = spherical_array_2[1]
    phi_1 = spherical_array_1[2]
    phi_2 = spherical_array_2[2]
    # we rewrite the standard Haversine slightly as long/lat is not the
       same as spherical coordinates - phi differs by pi/4
    spherical_distance = 2.0 * sphere_radius * math.asin(math.sqrt( ((1 -
       math.cos(phi_2-phi_1))/2.) + math.sin(phi_1) * math.sin(phi_2) * (
       (1 - math.cos(lambda_2-lambda_1))/2.)  ))

    return spherical_distance


def calculate_haversine_distances_between_spherical_points(
    cartesian_array_1, cartesian_array_2, sphere_radius):
    '''
    Calculate the haversine-based distance between two points on the
       surface of a sphere. Should be more accurate than the arc cosine
       strategy. See, for example: http://en.wikipedia.org/wiki/
       Haversine_formula
    '''
    spherical_array_1 = ct.convert_cartesian_array_to_spherical_array(
       cartesian_array_1)
    spherical_array_2 = ct.convert_cartesian_array_to_spherical_array(
       cartesian_array_2)
    lambda_1 = spherical_array_1[...,1]
```

```
    lambda_2 = spherical_array_2[...,1]
    phi_1 = spherical_array_1[...,2]
    phi_2 = spherical_array_2[...,2]
    # we rewrite the standard Haversine slightly as long/lat is not the
       same as spherical coordinates - phi differs by pi/4
    spherical_distance = 2.0 * sphere_radius * np.arcsin(np.sqrt( ((1 - np
       .cos(phi_2-phi_1))/2.) + np.sin(phi_1) * np.sin(phi_2) * ( (1 - np.
       cos(lambda_2-lambda_1))/2.)  ))

    return spherical_distance


def gaussian_weighting(distance_array, S):
    '''
    Take shape (N,) array of distances and returns their Gaussian
       weighting with smoothing factor S
    '''

    return np.exp(- distance_array**2 / S**2)


def inverse_weighting(distance_array, S):
    '''
    Take shape (N,) array of distances and returns their inverse weighting
        with smoothing factor S
    '''

    return (1 + ((distance_array**2) / (S**2)))**(-1)


def calculate_optimal_distance_weights(data_coords, interp_coord,
   density_weights, distance_method, weighting_threshhold, return_S=False)
   :
    '''
    Calculates the distance weightings Gij for a set of data to an
       interpolation point j.

    Parameters
    ----------
    data_coords : numpy.ndarray of shape (n,3)
        The coordinates of the data that should be interpolated to the
           location of `interp_coord`. Should be Cartesian coordinates on
           the surface of a sphere.
    interp_coord : numpy.ndarray of shape (3,)
        The coordinates of the point that `data_coords` should be
           interpolated to. Should be Cartesian coordinates on the surface
            of a sphere.
    density_weights : numpy.ndarray of shape (n,)
        The density weights of `data_coords` for the given `interp_coord`.
    distance_method : {'g', 'i'}
        The fuctional form that should be used for distance weighting.
           Options are a Gaussian or an inverse curve of the form 1/(x+1).
    weighting_threshhold : float
        The maximum value of the sum of the all the interpolation weights.
```

```
    return_S : bool, optional
        Whether or not to return the smoothing parameter S. Default False.
    '''

    assert (distance_method in ['g', 'i']), 'Invalid distance weighting
        method specified. Must be \'g\' or \'i\'.'
    if distance_method == 'g':
        distance_weighting = gaussian_weighting
    else:
        distance_weighting = inverse_weighting

    radius = ct.convert_cartesian_array_to_spherical_array(interp_coord)
        [0]

    def calculate_weights(smoothing_parameter):

        distances = calculate_haversine_distances_between_spherical_points
            (interp_coord, data_coords, sphere_radius=radius)
        distance_weights = distance_weighting(distances,
            smoothing_parameter)
        weights = density_weights * distance_weights

        return weights


    def calculate_total_weighting(smoothing_parameter):
        '''
        Calculates Wj - Wt, the difference between the sum of the
            weighting coefficients Gij and the weighting threshhold Wt, for
             a given smoothing paramter Sj and deinsity weights Aij.
        '''

        return np.sum(calculate_weights(smoothing_parameter)) -
            weighting_threshhold

    # Sjs = np.linspace(1e-8, 0.1)
    # print(Sjs)
    # for Sj in Sjs:
    #     print(calculate_total_weighting(Sj))

    smoothing_parameter = brentq(calculate_total_weighting, 0.000001,
        400750000)

    if return_S:
        return calculate_weights(smoothing_parameter), smoothing_parameter
    return calculate_weights(smoothing_parameter)


if __name__ == '__main__':

    sns.set_color_codes(palette='muted')
    sns.set_style('white')
    plt.rc('text', usetex = True)
    plt.rcParams['text.latex.preamble'] = r'\usepackage{amssymb, amsmath,
```

```
    bm}␣\usepackage{siunitx}␣\usepackage[T1]{fontenc}␣\usepackage{
    sansmath}␣\sansmath'
plt.rc('legend', **{'fontsize':15})
plt.rc('axes', **{'labelsize':15})
plt.rc('xtick', **{'labelsize':12})
plt.rc('ytick', **{'labelsize':12})
colors = sns.cubehelix_palette(8, start=.5, rot=-.75)

fig, ax = plt.subplots(1, 1, figsize=(6,3))
x = np.linspace(0, 10, num=200)
gauss = gaussian_weighting(x, 1)
inverse = inverse_weighting(x, 1)
ax.plot(x, gauss, zorder=2, label=r'Gaussian', color=colors[1])
ax.plot(x, inverse, zorder=1, label=r'Inverse', color=colors[3],
    linestyle='--')
ax.set_xlabel(r'$␣r␣$')
ax.set_ylabel(r'$␣D␣$')
ax.set_xlim(-0.25,10.25)
ax.set_ylim(-0.025,1.025)
ax.legend(loc='upper␣right')
fig.tight_layout()
plt.savefig('./images/distance_curves.pdf')
plt.close()
```

B.4 Exterior and interior points

```
import numpy as np

class Edge(object):
    """docstring for Edge."""
    def __init__(self, v1, v2):
        # print(v1)
        self.v1 = v1
        self.v2 = v2
        # self.xs = [v1[0], v2[0]]
        # self.ys = [v1[1], v2[1]]

    @classmethod
    def from_list(cls, l):
        l = np.array(l).flatten().tolist()
        v1 = l[:2]
        v2 = l[2:]
        return cls(v1, v2)

    @classmethod
    def from_coordinates(cls, v1x, v1y, v2x, v2y):
        v1 = [v1x, v1y]
        v2 = [v2x, v2y]
        return cls(v1, v2)

    def __eq__(self, other):
        if isinstance(other, self.__class__):
            return (((self.v1 == other.v1) and (self.v2 == other.v2)) or
                ((self.v1 == other.v2) and (self.v2 == other.v1)))
        return NotImplemented

    def __ne__(self, other):
        if isinstance(other, self.__class__):
            return not self.__eq__(other)
        return NotImplemented

    def __repr__(self):
        return '[{0}, {1}]'.format(self.v1, self.v2)

    def intersect(self, other):
        """
        Fuction to check whether the edge 'other' intersects with this
            edge. For more details, see https://www.geeksforgeeks.org/check
            -if-two-given-line-segments-intersect/.
        """

        # get the four vertices for readability
        v1, v2 = self.v1, self.v2
        w1, w2 = other.v1, other.v2

        # find the 4 orientations required for the general and special
            cases.
```

```
        o1 = orientation(v1, v2, w1)
        o2 = orientation(v1, v2, w2)
        o3 = orientation(w1, w2, v1)
        o4 = orientation(w1, w2, v2)

        # general case
        if ((o1 != o2) and (o3 != o4)):
            return True

        # special cases
        # v1, v2, and w1 are colinear and w1 lies on v1v2
        if ((o1 == 0) and on_segment(v1, v2, w1)):
            return True
        # v1, v2, and w2 are colinear and w2 lies on v1v2
        if ((o2 == 0) and on_segment(v1, v2, w2)):
            return True
        # w1, w2, and v1 are colinear and v1 lies on w1w2
        if ((o3 == 0) and on_segment(w1, w2, v1)):
            return True
        # w1, w2, and v2 are colinear and v2 lies on w1w2
        if ((o4 == 0) and on_segment(w1, w2, v2)):
            return True

        # no intersection between self and other
        return False

class Edges(object):
    """docstring for edge."""
    def __init__(self, vertices):
        # print(vertices)
        vertices = vertices.tolist()
        self.edges = []
        for i in range(len(vertices)):
            if i == range(len(vertices))[-1]:
                self.edges.append(Edge(vertices[i], vertices[0]))
                break
            self.edges.append(Edge(vertices[i], vertices[i+1]))

    @classmethod
    def from_edge_list(cls, l):
        vertices = []
        for edge in l:
            vertices.append(edge.v1)
        return cls(np.array(vertices))

    def share_edge(self, other, return_edge=False):
        if isinstance(other, self.__class__):
            if self == other:
                return False
            matching_edge = False
            edge = False
            for my_edge in self.edges:
                for its_edge in other.edges:
                    if return_edge and my_edge == its_edge:
```

```
                        edge = my_edge
                    matching_edge = (matching_edge or my_edge == its_edge)
            if return_edge:
                return edge
            return matching_edge
        return NotImplemented

    def __eq__(self, other):
        if isinstance(other, self.__class__):
            return self.edges == other.edges
        return NotImplemented

    def __ne__(self, other):
        if isinstance(other, self.__class__):
            return not self.__eq__(other)
        return NotImplemented

    def __len__(self):
        return len(self.edges)

    def __repr__(self):
        r = [[edge.v1, edge.v2] for edge in self.edges]
        return repr(r)

def on_segment(a, b, c):
    """
    Fuction to check, for three colinear vertices a, b, c, if c lies on
       the edge ab. Works for all c, a, b.
    """
    if ((c[0] <= max(a[0], b[0])) and (c[0] >= min(a[0], b[0])) and (c[1]
       <= max(a[1], b[1])) and (c[0] >= min(a[1], b[1]))):
        return True
    return False

def orientation(a, b, c):
    """
    Function to find the orientation of an ordered triplet of points (a, b
       , c).
    Returns:
    0: Colinear points
    1: Clockwise points
    -1: Anticlockwise points

    See https://www.geeksforgeeks.org/orientation-3-ordered-points/ for
       details of below formula.
    """
    val = ((b[1] - a[1]) * (c[0] - b[0])) - ((b[0] - a[0]) * (c[1] - b[1])
       )
    if (val > 0):
        return 1 # clockwise
    elif (val < 0):
        return -1 # anticlockwise
    else:
        return 0 # colinear
```

```
if __name__ == "__main__":
    ab = Edge.from_coordinates(0,0,1,0)
    bc = Edge.from_coordinates(1,0,1,1)
    cd = Edge.from_coordinates(1,1,0,1)
    da = Edge.from_coordinates(0,1,0,0)
    box1 = Edges.from_edge_list([ab,bc,cd,da])


    be = Edge.from_coordinates(1,0,2,0)
    ef = Edge.from_coordinates(2,0,2,1)
    fc = Edge.from_coordinates(2,1,1,1)
    cb = Edge.from_coordinates(1,1,1,0)
    box2 = Edges.from_edge_list([be,ef,fc,cb])

    fb = Edge.from_coordinates(2,1,1,0)
    triangle = Edges.from_edge_list([be,ef,fb])

    connecting_line = Edge.from_coordinates(0.5,0.5,1.5,0.5)

    shared_edge_box2 = box1.share_edge(box2, return_edge=True)
    print(shared_edge_box2)
    print(box1.share_edge(triangle, return_edge=True))

    print(shared_edge_box2.intersect(connecting_line))
```

B.5 2D deviatoric tensor fields

```
import gc
import numpy as np
from scipy.signal import fftconvolve
from scipy.ndimage import fourier_gaussian
from numpy import linalg as la


# for plotting
import matplotlib.pyplot as plt
import seaborn as sns

class Surface(object):
    '''
    the surface class defines a surface on a segment of a spherical shell.
        it
    contains a numpy meshgrid of coordinates in theta and phi, and a
    corresponding array of the surface values at each coordinate pair.

    should passed as [rs, thetas, phis] (or equivalent ndarray)
    '''

    def __init__(self, surface, longitudes, colatitudes, r, degs, lat, wc)
        :
        '''Unused constructor.'''

        self.surface = np.copy(surface)
        self.r = r
        self.thetas = np.copy(longitudes)
        self.phis = np.copy(colatitudes)
        self.degs = degs
        self.lat = lat
        self.wc = wc

        self.dtheta = longitudes[0, 1] - longitudes[0, 0]
        self.dphi = colatitudes[1, 0] - colatitudes[0, 0]

        self.colatitudes = np.degrees(colatitudes)
        self.latitudes = 90 - np.degrees(colatitudes)
        self.longitudes = np.degrees(longitudes)

    @classmethod
    def from_array(cls, data, r=6371000, angle_measure='degrees',
        polar_coordinate='latitude', water_corrected=False):
        '''
        Creates an instance of Surface from an array of data and
            coordinates.
        '''

        surface = np.array(data[0])
        longitudes = np.array(data[1])
        polar_angles = np.array(data[2])
```

```
        rads = ['rad', 'rads', 'radian', 'radians']
        degs = ['deg', 'degs', 'degree', 'degrees']
        if angle_measure in rads:
            degs = False
        elif angle_measure in degs:
            longitudes = np.radians(longitudes)
            polar_angles = np.radians(polar_angles)
            degs = True
        else:
            print("angle must be 'radians' or 'degrees'!")
            quit()

        lats = ['lat', 'latitude']
        colats = ['colat', 'colatitude']
        if polar_coordinate in lats:
            polar_angles = np.pi/2 - polar_angles
            lat = True
        elif polar_coordinate in colats:
            lat = False
        else:
            print("coords must be 'latitude' or 'colatitude'!")
            quit()

        return cls(surface, longitudes, polar_angles, r, degs, lat,
           water_corrected)

    def smooth_fft2(self, sigma=50000):
        # get what size sigma*n_sigma translates to
        ## get the middle of the region
        phi = self.phis[int(len(self.phis)//2), 0]

        Delta_phi = sigma/self.r
        m = np.ceil(Delta_phi/np.abs(self.dphi)).astype('int')

        Delta_theta = sigma/(self.r*np.sin(phi))
        n = np.ceil(Delta_theta/np.abs(self.dtheta)).astype('int')

        surface_ = np.fft.fft2(self.surface)
        result = fourier_gaussian(surface_, sigma=(n, m))
        result = np.fft.ifft2(result)
        self.surface = result.real
        return self.surface

    def smooth_fft(self, sigma=50000, n_sigma=4):
        # get what size sigma*n_sigma translates to
        ## get the middle of the region
        phi = self.phis[int(len(self.phis)//2), 0]

        Delta_phi = sigma/self.r
        m = np.ceil(Delta_phi/np.abs(self.dphi)).astype('int')
        m_max = np.ceil(n_sigma*Delta_phi/np.abs(self.dphi)).astype('int')

        Delta_theta = sigma/(self.r*np.sin(phi))
```

```
        n = np.ceil(Delta_theta/np.abs(self.dtheta)).astype('int')
        n_max = np.ceil(n_sigma*Delta_theta/np.abs(self.dtheta)).astype('
            int')

        # expand in_array to fit edge of kernel
        pad_sizes = ((m_max, m_max), (n_max, n_max))
        padded_array = np.pad(self.surface, pad_sizes, 'symmetric')

        # build kernel
        xs, ys = np.mgrid[-n_max:n_max + 1, -m_max:m_max + 1]
        g = np.exp(-(xs**2 / n**2 + ys**2 / m**2)/2)
        g = (g / g.sum()).astype(self.surface.dtype)

        # do the Gaussian blur
        self.surface = fftconvolve(padded_array, g, mode='same')[m_max:-
            m_max, n_max:-n_max]
        return self.surface

    def compress_water(self, rho_water=1000, rho_crust=2800):
        '''
        Compresses the water that sits above the elevation from bathometry
            data into a column of rock.

        Parameters
        ----------
        rho_water: The density of the water (in kg/m^3). Default values is
            1000 kg/m^3.
        rho_crust: The assumed density of the crust (in kg/m^3). Default
            values is 2800 kg/m^3.
        '''

        if not self.wc:
            bathometry_data = self.surface[self.surface < 0]
            compressed_water_depths = bathometry_data*(1 - rho_water/
                rho_crust)
            self.surface[self.surface < 0] = compressed_water_depths
            self.wc = True
        else:
            print('This␣surface␣has␣already␣been␣corrected␣for␣the␣
                presence␣of␣water.␣If␣you␣want␣to␣do␣it␣again␣for␣some␣
                reason,␣you␣can␣override␣this␣by␣manually␣setting␣Surface.
                wc␣to␣False.')
        return self.surface

    def calculate_gpe(self, h_0=100, S_0=3.5e4, rho_crust=2800, rho_mantle
        =3300, g=9.81):
        '''
        Calculates the gravitational potential energy due to topography.

        Parameters
        ----------
        h_0: The height above sea-level of the reference column to which
            GPE is calculated (in meters). Default value of 100 m.
        S_0: The depth of the crust for the reference column (in meters).
```

```
        Default value of 35 km.
    rho_crust: The assumed density of the crust (in kg/m^3). Default
        values is 2800 kg/m^3.
    rho_mantle: The assumed density of the mantle (in kg/m^3). Default
         values is 3300 kg/m^3.
    '''

    # calculates the GPE relative to the reference column, see England
        's Iran paper for details
    self.Delta_gpe = g*rho_crust*(self.surface - h_0)*(S_0 + 0.5*
        rho_mantle*(self.surface - h_0)/(rho_mantle - rho_crust))
    self.h_0, self. S_0, self.rc, self.rm, self.g = h_0, S_0,
        rho_crust, rho_mantle, g
    self.gpe_calculated = True

    return self.Delta_gpe

def calculate_gpe_stress(self, L=1e5):
    '''
    Calculates the stress in spherical goordinates due to the
        gravitational potential energy due to topography.

    Parameters
    L: The assumed thickness of the lithosphere (in meters). Default
        value of 100 km.
    '''
    if not self.gpe_calculated:
         print('GPE has not been calculated for this surface yet which
             is required to calculate the stress due to GPE. Doing that
             now.')
         self.calculate_gpe()

    self.L = L

    # fairly self-explanatory: calculate the stress on the surface of
        the sphere (see England's Iran paper for eqn)
    dgpe_dphi, dgpe_dtheta = np.gradient(self.Delta_gpe, self.dtheta,
        self.dphi)
    self.theta_stresses = dgpe_dtheta/(self.r*L*np.sin(self.phis))
    self.phi_stresses = dgpe_dphi/(self.r*L)

    return self.theta_stresses, self.phi_stresses

def calculate_deviatoric_vectors(self, geometry='spherical'):
    '''
    Calculates the deviatoric stress or strain tensor on the surface
        of the sphere of a vector field,
    then calculates the vectors which correspond to the maximum and
        minimum compressive stresses/strains.

    This entails calculating the Christoffel symbols from the metric
        of the surface of a sphere,
    and using these to calculate the correction terms of the covariant
         derivative which are zero
```

```
for Euclidian geometry.
'''

assert (str(geometry).lower() in ['spherical', 'cartesian', '
    euclidian']), "Argument␣geometry␣must␣be␣\'spherical\'␣or␣\'
    euclidian\'."

print(self.thetas)

if geometry == "spherical":
    # calculate the covariant derivative of stress on the surface
        of the sphere
    f11 = -np.gradient(self.theta_stresses, self.dtheta, axis=1)
    f12 = -np.gradient(self.theta_stresses, self.dphi, axis=0) -
        self.phi_stresses*np.cos(self.thetas)*np.sin(self.thetas)
    f21 = -np.gradient(self.phi_stresses, self.dtheta, axis=1) +
        self.phi_stresses*np.cos(self.thetas)/np.sin(self.thetas)
    f22 = -np.gradient(self.phi_stresses, self.dphi, axis=0) +
        self.theta_stresses*np.cos(self.thetas)/np.sin(self.thetas)
else:
    # calculate the covariant derivative of stress on the surface
        of a flat geometry
    f11 = -np.gradient(self.theta_stresses, self.dtheta, axis=1)
    f12 = -np.gradient(self.theta_stresses, self.dphi, axis=0)
    f21 = -np.gradient(self.phi_stresses, self.dtheta, axis=1)
    f22 = -np.gradient(self.phi_stresses, self.dphi, axis=0)

# calculate the deviatoric stress
epsilon_11s = np.copy(f11)
epsilon_12s = 0.5*(f12 + f21)
epsilon_21s = np.copy(epsilon_12s)
epsilon_22s = np.copy(f22)

# reformat the array so that it's an n_theta by n_phi array of 2
    by 2 tensors
epsilon_tensors = np.array([epsilon_11s, epsilon_12s, epsilon_21s,
    epsilon_22s]).transpose(1,2,0)
epsilon_tensors = epsilon_tensors.reshape((epsilon_tensors.shape
    [0], epsilon_tensors.shape[1], 2, 2))

# diagonalising the epsilon tensors is fairly straightforward
    using the eigenvalues
# but we can easily get principle stress axes from the
    eigenvectors of the epsilon tensors
evals, evecs = la.eigh(epsilon_tensors)
e11s = evals[:,:,0,None]*evecs[:,:,:,0]
e22s = evals[:,:,1,None]*evecs[:,:,:,1]

# we want to separate the e_11s and e_22s into principle strain
    axes
# sigma 1 should be the largest of the eigenvalues, so we get an
    array that is True where ev_1 > ev_2
c = evals[:,:,0] > evals[:,:,1]
# we should be able to use numpy.where to extract first the vector
```

```
            components for sigma_1, and then for sigma_2
        self.sigma1s = np.where(c[:,:,None], e11s, e22s)
        self.sigma2s = np.where(c[:,:,None], e22s, e11s)
        return self.sigma1s, self.sigma2s

def calculate_deviatoric_tensor_field(vector_field, geometry='spherical'):
    '''
    Calculates the deviatoric tensor field of a vector field on the
       surface of the sphere, then calculates the vectors which correspond
        to the maximum and minimum compressive stresses/strains.

    This entails calculating the Christoffel symbols from the metric of
       the surface, and using these to calculate the correction terms of
       the covariant derivative (which are zero for Euclidian geometry,
       and nonzero for the surface of a sphere).

    Parameters
    ----------
    vector_field : array_like of shape (n,m,4)
        The vector field from which to calculate the tensor field. Should
           be an array-like with shape (n,m,4) in spherical coordinates.
           In other words, an array containing a numpy meshgrid of n theta
           -coordinates and m phi-coordinates, where each position in the
           meshgrid contains a quadruplet of (theta, phi, theta_velocity,
           phi_velocity).
    geometry : {'spherical', 'euclidian'}, default 'spherical'
        Whether to assume a spherical or Euclidian (flat) geometry when
           calculating the deviatoric tensor field.
    '''

    assert (str(geometry).lower() in ['spherical', 'cartesian', 'euclidian
       ']), "Argument␣geometry␣must␣be␣\'spherical\'␣or␣\'euclidian\'."


    vector_field = np.copy(np.transpose(vector_field, axes=(2,0,1)))
    # if not geometry == 'spherical': print(vector_field[0])
    thetas = vector_field[0]
    dtheta = thetas[1,0] - thetas[0,0]
    theta_vectors = vector_field[2]
    phis = vector_field[1]
    dphi = phis[0,1] - phis[0,0]
    phi_vectors = vector_field[3]

    # print(theta_vectors)
    # print(phi_vectors)

    # calculate the covariant derivative of stress on the surface of the
       sphere
    if geometry == 'spherical':
        f11 = -np.gradient(theta_vectors, dtheta, axis=0)
        f12 = -np.gradient(theta_vectors, dphi, axis=1) - phi_vectors*np.
           cos(thetas)*np.sin(thetas)
        f21 = -np.gradient(phi_vectors, dtheta, axis=0) + phi_vectors*np.
           cos(thetas)/np.sin(thetas)
```

```
    f22 = -np.gradient(phi_vectors, dphi, axis=1) + theta_vectors*np.
        cos(thetas)/np.sin(thetas)
else:
    f11 = -np.gradient(theta_vectors, dtheta, axis=0)
    f12 = -np.gradient(theta_vectors, dphi, axis=1)
    f21 = -np.gradient(phi_vectors, dtheta, axis=0)
    f22 = -np.gradient(phi_vectors, dphi, axis=1)

if geometry == 'spherical': print(f11)
if geometry == 'spherical': print(f12)
if geometry == 'spherical': print(f21)
if geometry == 'spherical': print(f22)
if geometry == 'spherical': print()

# calculate the deviatoric stress
epsilon_11s = np.copy(f11)
epsilon_12s = 0.5*(f12 + f21)
epsilon_21s = np.copy(epsilon_12s)
epsilon_22s = np.copy(f22)

# reformat the array so that it's an n_theta by n_phi array of 2 by 2
    arrays
epsilon_tensors = np.array([epsilon_11s, epsilon_12s, epsilon_21s,
    epsilon_22s]).transpose(1,2,0)
epsilon_tensors = epsilon_tensors.reshape((epsilon_tensors.shape[0],
    epsilon_tensors.shape[1], 2, 2))

if geometry == 'spherical': print(epsilon_tensors[:,0])
if geometry == 'spherical': print()

# diagonalising the epsilon tensors is fairly straightforward using
    the eigenvalues
# but we can easily get principle stress axes from the eigenvectors of
     the epsilon tensors
evals, evecs = la.eigh(epsilon_tensors)
e11s = evals[:,:,0,None]*evecs[:,:,:,0]
e22s = evals[:,:,1,None]*evecs[:,:,:,1]

if geometry == 'spherical': print(evals[:,0])
if geometry == 'spherical': print()
if geometry == 'spherical': print(evecs[:,0])
if geometry == 'spherical': print()

# we want to separate the e_11s and e_22s into principle strain axes
# sigma 1 should be the largest of the eigenvalues, so we get an array
     that is True where ev_1 > ev_2
c = evals[:,:,0] > evals[:,:,1]
# we should be able to use numpy.where to extract first the vector
    components for sigma_1, and then for sigma_2
sigma1s = np.where(c[:,:,None], e11s, e22s)
sigma2s = np.where(c[:,:,None], e22s, e11s)

if geometry == 'spherical': print(sigma1s[:,0])
if geometry == 'spherical': print()
```

```
    if geometry == 'spherical': print(sigma2s[:,0])
    if geometry == 'spherical': print()


    # mags = la.norm(epsilon_tensors, axis=3)

    # if geometry == 'spherical': print(mags[:,0])
    # if geometry == 'spherical': print()

    # c = mags[...,0] > mags[...,1]
    # if geometry == 'spherical': print(c)
    # if geometry == 'spherical': print()

    # sigma1s = np.where(c[:,:,None], epsilon_tensors[:,:,0],
        epsilon_tensors[:,:,1])
    # sigma2s = np.where(c[:,:,None], epsilon_tensors[:,:,1],
        epsilon_tensors[:,:,0])

    # if geometry == 'spherical': print(sigma1s[:,0])
    # if geometry == 'spherical': print()
    # if geometry == 'spherical': print(sigma2s[:,0])
    # if geometry == 'spherical': print()

    print()
    return np.transpose([sigma1s, sigma2s], axes=(1,2,0,3))

if __name__ == '__main__':
    m, n = 3, 3
    x, y = np.meshgrid(np.linspace(-15, 15, num=n), np.linspace(-15, 15,
        num=m))
    surface_1 = np.ones((m,n))
    surface_1[:,int(n/2):] = surface_1[:,int(n/2):] * -1
    test_data = np.array([surface_1, x, y])
    print(test_data[0], end='\n\n')
    test_surface = Surface.from_array(test_data, angle_measure='deg',
        polar_coordinate='lat')
    test_surface.compress_water()
    print(test_surface.surface)
    test_surface.smooth_fft(sigma=50000)
    print(test_surface.surface)

    ax = sns.heatmap(test_surface.surface)
    plt.show()
```

B.6 Simple test fields for 2D deviatoric tensor fields

```
import numpy as np
import os
from itertools import combinations
from scipy.interpolate import griddata
from scipy.optimize import minimize_scalar
from scipy.special import gamma
from sklearn.metrics import mean_squared_error

import Coordinates.coordinate_transforms as ct
import Distance.distance_weightings as dw
import CrossValidate.cross_validate as cv
import KnownField.known_field as kf
import surface as sf

import matplotlib.pyplot as plt
from matplotlib import cm
import seaborn as sns
from matplotlib import rc
import cartopy.crs as ccrs
import cartopy.feature as feature
import gdal
import matplotlib.pyplot as plt
import matplotlib.colors as mcolors
import matplotlib.ticker as mticker
from mpl_toolkits.axes_grid1 import make_axes_locatable
import seaborn as sns
from cartopy.io.shapereader import Reader
from cartopy.feature import ShapelyFeature
from matplotlib.patches import Wedge
friendly_colours = sns.diverging_palette(240, 10)
unfriendly_colours = ["#F05354", "#48BEA1", "#52C8EB"]

sns.set_color_codes(palette='muted')
sns.set_style('white')
plt.rc('text', usetex = True)
plt.rc('legend', **{'fontsize':15})
cmap = sns.cubehelix_palette(start=0.5, rot=-.75, reverse=True, as_cmap=
    True)

def dist_to_ang(velocities, r=6.371e6):

    # expects shape (n, 4) array where n is the number of coordinate
       triplets to process, and 4 is theta, phi, theta vel, phi vel in
       meters and meters per second respectively

    velocities = np.copy(velocities)
    velocities[...,2] = velocities[...,2] / r
    velocities[...,3] = velocities[...,3] / (r * np.sin(velocities[...,0])
       )

    return velocities
```

```
def theta_comp(spherical_coordinates, gridshape):

    # expects shape (n, 3) array where n is the number of coordinate
       triplets to process, and 3 is r, theta, phi

    coords = np.copy(spherical_coordinates)
    coords = np.transpose(coords) # transpose the data so it's shorter to
       work with all the phi/theta data at once

    phi_vels = np.zeros_like(coords[2])

    theta_vels = np.linspace(2, 1, num=gridshape[0])
    theta_vels = np.tile(theta_vels, (gridshape[1], 1))
    theta_vels = theta_vels.T.flatten()

    return np.array([theta_vels, phi_vels]).T

def phi_comp(spherical_coordinates, gridshape):

    # expects shape (n, 3) array where n is the number of coordinate
       triplets to process, and 3 is r, theta, phi

    coords = np.copy(spherical_coordinates)
    coords = np.transpose(coords) # transpose the data so it's shorter to
       work with all the phi/theta data at once

    theta_vels = np.zeros_like(coords[1])

    phi_vels = np.linspace(2, 1, num=gridshape[1])
    phi_vels = np.tile(phi_vels, (gridshape[0], 1))
    phi_vels = phi_vels.flatten()

    return np.array([theta_vels, phi_vels]).T

def theta_shear(spherical_coordinates, gridshape):

    # expects shape (n, 3) array where n is the number of coordinate
       triplets to process, and 3 is r, theta, phi

    coords = np.copy(spherical_coordinates)
    coords = np.transpose(coords) # transpose the data so it's shorter to
       work with all the phi/theta data at once

    phi_vels = np.zeros_like(coords[2])

    theta_vels = np.linspace(1, 2, num=gridshape[1])
    theta_vels = np.tile(theta_vels, (gridshape[0], 1))
    theta_vels = theta_vels.flatten()

    return np.array([theta_vels, phi_vels]).T

def phi_shear(spherical_coordinates, gridshape):
```

```
    # expects shape (n, 3) array where n is the number of coordinate
       triplets to process, and 3 is r, theta, phi

    coords = np.copy(spherical_coordinates)
    coords = np.transpose(coords) # transpose the data so it's shorter to
       work with all the phi/theta data at once

    theta_vels = np.zeros_like(coords[1])

    phi_vels = np.linspace(2, 1, num=gridshape[0])
    phi_vels = np.tile(phi_vels, (gridshape[1], 1))
    phi_vels = phi_vels.T.flatten()

    return np.array([theta_vels, phi_vels]).T

# make a grid for interpolation
thetas = np.linspace(11*np.pi/18, 25*np.pi/36, num=7)
phis = np.linspace(11*np.pi/18, 25*np.pi/36, num=7)
grid = np.meshgrid(thetas, phis)
grid.insert(0, np.ones_like(grid[0]) * 6.378e6) # add in the r coordinate
grid = np.array(grid)
grid = np.transpose(grid) # this makes it so that grid[i,j] returns the ij
   -th coordinate triplet of (r,theta,phi)
coordinates = np.reshape(grid, (grid.shape[0]*grid.shape[1], grid.shape
   [2])) # flatten the "grid" part for compatibility with other functions

velocities, field_name = theta_comp(coordinates, grid.shape), "
   theta_compression"
# velocities, field_name = theta_shear(coordinates, grid.shape), "
   theta_shear"
# velocities, field_name = phi_comp(coordinates, grid.shape), "
   phi_compression"
# velocities, field_name = phi_shear(coordinates, grid.shape), "phi_shear"

velocities = np.transpose(np.reshape(velocities, (grid.shape[0],grid.shape
   [1],2))) # restore the "grid"
velocities = np.transpose([*np.meshgrid(thetas, phis), *velocities]) # add
    the coordinates back
mps_vels = np.copy(velocities)
velocities = dist_to_ang(velocities) # convert meters per second to
   radians per second
tensor_field = sf.calculate_deviatoric_tensor_field(velocities)
flat_tensor_field = sf.calculate_deviatoric_tensor_field(velocities,
   geometry='euclidian')

latitudes = np.rad2deg(velocities[...,1])
longitudes = 90 - np.rad2deg(velocities[...,0])

fig, ax = plt.subplots(1, 1, subplot_kw={'projection': ccrs.PlateCarree()
   })
gl = ax.gridlines(draw_labels=True)
gl.top_labels = False
gl.xlines = False
gl.right_labels = False
```

```
gl.ylines = False
ax.quiver(latitudes, longitudes, mps_vels[...,3], -mps_vels[...,2])
plt.savefig(os.path.join('figures', 'simple_field', field_name + '.pdf'),
    bbox_inches='tight', pad_inches=0)
plt.close()

scale, width = 5e-7, 3e-1
fig, ax = plt.subplots(1, 1, subplot_kw={'projection': ccrs.PlateCarree()
    })
gl = ax.gridlines(draw_labels=True)
gl.top_labels = False
gl.xlines = False
gl.right_labels = False
gl.ylines = False
ax.quiver(latitudes, longitudes, tensor_field[...,0,1], -tensor_field
    [...,0,0],
           units='xy', angles='xy', zorder=2, color='r', pivot="mid",
              headaxislength=0, headlength=0, scale=scale, width=width)
ax.quiver(latitudes, longitudes, tensor_field[...,1,1], -tensor_field
    [...,1,0],
           units='xy', angles='xy', zorder=1, color='b', pivot="mid",
              headaxislength=0, headlength=0, scale=scale, width=width)
plt.savefig(os.path.join('figures', 'simple_field', field_name + "_dev_sph
    " + '.pdf'), bbox_inches='tight', pad_inches=0)
plt.close()

fig, ax = plt.subplots(1, 1, subplot_kw={'projection': ccrs.PlateCarree()
    })
gl = ax.gridlines(draw_labels=True)
gl.top_labels = False
gl.xlines = False
gl.right_labels = False
gl.ylines = False
ax.quiver(latitudes, longitudes, flat_tensor_field[...,0,1], -
    flat_tensor_field[...,0,0],
           units='xy', angles='xy', zorder=2, color='r', pivot="mid",
              headaxislength=0, headlength=0, scale=scale, width=width)
ax.quiver(latitudes, longitudes, flat_tensor_field[...,1,1], -
    flat_tensor_field[...,1,0],
           units='xy', angles='xy', zorder=1, color='b', pivot="mid",
              headaxislength=0, headlength=0, scale=scale, width=width)
plt.savefig(os.path.join('figures', 'simple_field', field_name + "
    _dev_flat" + '.pdf'), bbox_inches='tight', pad_inches=0)
plt.close()
```

B.7 Deviatoric stress due to gravitational collapse

```
import sys, getopt, json, copy, os
import numpy as np
from osgeo import ogr, osr, gdal
from surface import Surface

# for plotting
import cartopy.crs as ccrs
import cartopy.feature as feature
import matplotlib.pyplot as plt
import matplotlib.ticker as mticker
from mpl_toolkits.axes_grid1 import make_axes_locatable
import seaborn as sns
pal = sns.diverging_palette(240, 10, as_cmap=True)

np.seterr(all='raise')

'''program to get geotiff (.tif) elevation data and convert it to
    spherical
coordinates.

the first command line argument following the .py program should be the
    source
file path, the second argument should be the target file path, e.g.:

    $ python gtif2sph ./source ./target

if you want detailed information, insert verbose as an argument:

    $ python gtif2sph ./source ./target verbose

to get full precision when printing verbose output, add precise as an
    argument.
use the projection argument to show the projection when printing verbose
    output.
use the tag r to get the radius from the centre of the earth instead of
    height
above sea level.

tom new
20/08/2018'''

def array_to_gtiff(data, save_path, transform, projection):
    """
    Array > GTiff
    Save a GeoTiff from a C order array.
    """

    driver = gdal.GetDriverByName('GTiff')
    rows, cols = data.shape
    out_dataset = driver.Create(save_path, cols, rows, 1, gdal.GDT_Float32
        )
```

```
    out_dataset.SetGeoTransform(transform)
    out_dataset.SetProjection(projection)
    out_dataset.GetRasterBand(1).WriteArray(data)
    out_dataset.FlushCache()  # Write to disk.

# get command line aruments
args = copy.deepcopy(sys.argv)

opts, args = getopt.getopt(sys.argv[1:], "d:l:n:p:s:t:vc", ["dem=", "
    limits=", "plot=", "precise", "projection", "save=", "ticks=", "verbose
    ", "cart"])
n, limit, plot, precise, projection, save, tick_spacing, verbose, c = 40,
    False, False, False, False, False, False, False, ""
for opt, arg in opts:
    if opt in ("-d", "--dem"):
        dem_path = arg
    elif opt in ("-l", "--limits"):
        limit = True
        limits = arg.strip("[]").split(",")
        limits = np.array(limits, dtype=float)
    elif opt == '-n':
        n = int(arg)
    elif opt in ("-p", "--plot"):
        plot = True
        plot_path = arg
    elif opt == '--precise':
        precise = True
    elif opt == '--projection':
        projection = True
    elif opt in ("-s", "--save"):
        save_path = arg
        save = True
    elif opt in ("-t", "--ticks"):
        tick_spacing = float(arg)
    elif opt in ("-v", "--verbose"):
        verbose = True
    elif opt in ("-c", "--cart"):
        c = "_c"

# open the file
if verbose: print('Reading file...')
dataset = gdal.Open(dem_path, gdal.GA_ReadOnly)
if verbose: print('Done.')


# show size and projection info
if verbose:
    print("Driver: {}/{}".format(dataset.GetDriver().ShortName,
                                 dataset.GetDriver().LongName))
    print("Size is {} x {} x {} (x by y by layers)".format(dataset.
        RasterXSize,
                                           dataset.RasterYSize,
                                           dataset.RasterCount))
    if projection: print("Projection is {}".format(dataset.GetProjection()
```

```
        ))
projection = dataset.GetProjection()

# get origin (top left) and step size
geotransform = dataset.GetGeoTransform()
if verbose:
    if geotransform:
        if precise:
            print("Origin: {} deg N, {} deg E".format(geotransform[3],
                geotransform[0]))
            print("Pixel Size: {} deg, {} deg".format(geotransform[5],
                geotransform[1]))
        else:
            print("Origin: {:.3g} deg N, {:.3g} deg E".format(geotransform
                [3], geotransform[0]))
            print("Pixel Size: {:.3g} deg, {:.3g} deg".format(geotransform
                [5], geotransform[1]))
origin = np.array([geotransform[0], geotransform[3]])
spacing = np.array([geotransform[1], geotransform[5]])

# get elevation data
raster_band = dataset.GetRasterBand(1)
heights = raster_band.ReadAsArray()

# create array of h (or r), theta, phi data
npixels = np.array([dataset.RasterXSize, dataset.RasterYSize])
longitudes = np.arange(1, dataset.RasterXSize + 1) * spacing[0] + origin
    [0]
latitudes = np.arange(1, dataset.RasterYSize + 1) * spacing[1] + origin[1]
longitudes, latitudes = np.meshgrid(longitudes, latitudes)
data = np.array([heights, longitudes, latitudes])

data = Surface.from_array(data, angle_measure='deg', polar_coordinate='lat
    ') # make it a Surface
original_topo = np.copy(data.surface)
data.compress_water() # compress water to rock
sigma = 50000
data.smooth_fft(sigma=sigma) # smooth it


data.calculate_gpe() # calculate GPE
data.calculate_gpe_stress() # calculate the stresses
if c:
    data.calculate_deviatoric_vectors(geometry='Cartesian')
else:
    data.calculate_deviatoric_vectors()

if not limit: limits = [longitudes[0,0], longitudes[0,-1], latitudes
    [-1,0], latitudes[0,0]]
imlimits = [longitudes[0,0], longitudes[0,-1], latitudes[-1,0], latitudes
    [0,0]]

if plot:
    fig, ax = plt.subplots(1, 1, figsize=(6,7), subplot_kw={'projection':
```

```
        ccrs.PlateCarree()})
    gl = ax.gridlines(draw_labels=True)
    gl.xlabels_top = False
    gl.xlines = False
    gl.ylabels_right = False
    gl.ylines = False
    ax.set_extent(limits, crs=ccrs.PlateCarree())
    if tick_spacing:
        gl.xlocator = mticker.MultipleLocator(tick_spacing)
        gl.ylocator = mticker.MultipleLocator(tick_spacing)
    ax.add_feature(feature.COASTLINE, edgecolor='tab:gray', zorder=1)
    ax.add_feature(feature.BORDERS, edgecolor='tab:gray', zorder=1)
    im = ax.imshow(data.Delta_gpe / 1e12, extent=imlimits, cmap=pal,
        interpolation='none', origin='upper', zorder=0)

    ax.quiver(data.longitudes[n:-n:n,n:-n:n], data.latitudes[n:-n:n,n:-n:n
        ],
            data.theta_stresses[n:-n:n,n:-n:n], -data.phi_stresses[n:-n:n,
                n:-n:n],
            units='xy', angles='xy', scale=1.75e2, zorder=2)


    divider = make_axes_locatable(ax)
    cax = divider.new_horizontal(size="5%", pad=0.15, axes_class=plt.Axes)
    fig.add_axes(cax)
    plt.colorbar(im, cax=cax)
    cax.set_title('TN/m')
    plt.savefig(plot_path)
    plt.close()

if save:
    if verbose: print('Saving␣as␣NPY...')
    np.savez(save_path + "_" + str(int(sigma/500)), longitudes=data.
        longitudes, latitudes=data.latitudes, topography=original_topo,
        smooth_topo=data.surface, Delta_gpe=data.Delta_gpe, theta_stresses=
        data.theta_stresses, phi_stresses=data.phi_stresses, sigma1s=data.
        sigma1s, sigma2s=data.sigma2s)
    if verbose: print('Done.')
```