ABSTRACT

Title of Thesis:	CONSTRAINTS ON THE LITHOSPHERIC STRUCTURE OF MID OCEAN RIDGES FROM OCEANIC CORE COMPLEX MORPHOLOGY
	Mark Oscar Larson, Master of Science, 2016
Thesis Directed By:	Professor Laurent G. J. Montési, Department of Geology

The Mid-oceanic ridge system is a feature unique to Earth. It is one of the fundamental components of plate tectonics and reflects interior processes of mantle convection within the Earth. The thermal structure beneath the mid-ocean ridges has been the subject of several modeling studies. It is expected that the elastic thickness of the lithosphere is larger near the transform faults that bound mid-ocean ridge segments. Oceanic core complexes (OCCs), which are generally thought to result from long-lived fault slip and elastic flexure, have a shape that is sensitive to elastic thickness. By modeling the shape of OCCs emplaced along a ridge segment, it is possible to constraint elastic thickness and therefore the thermal structure of the plate and how it varies along-axis.

This thesis builds upon previous studies that utilize thin plate flexure to reproduce the shape of OCCs. I compare OCC shape to a suite of models in which elastic thickness, fault dip, fault heave, crustal thickness, and axial infill are systematically varied. Using a grid search, I constrain the parameters that best reproduce the bathymetry and/or the slope of ten candidate OCCs identified along the

12°—15°N segment of the Mid-Atlantic Ridge. The lithospheric elastic thicknesses that explains these OCCs is thinner than previous investigators suggested and the fault planes dip more shallowly in the subsurface, although at an angle compatible with Anderson's theory of faulting. No relationships between model parameters and an oceanic core complexes location within a segment are identified with the exception that the OCCs located less than 20km from a transform fault have slightly larger elastic thickness than OCCs in the middle of the ridge segment.

CONSTRAINTS ON THE LITHOSPHERIC STRUCTURE OF MID OCEAN RIDGES FROM OCEANIC CORE COMPLEX MORPHOLOGY

by

Mark Oscar Larson

Thesis submitted to the Faculty of the Graduate School of the
University of Maryland, College Park, in partial fulfillment
of the requirements for the degree of
Master of Science
2016

Advisory Committee:

Professor Laurent G. J. Montési, Chair

Professor Wenlu Zhu

Professor Deborah Smith

Professor Karen Prestegaard

Acknowledgements

I would like to especially thank my primary advisor, Laurent Montési for his guidance throughout my graduate school process. He was supportive of the many directions my research went. Wenlu Zhu deserves credit for additional support through much of it. I would like to extend gratitude for Debbie Smith's enthusiasm and expertise through my oceanic core complex study. I thank Karen Prestegaard for her time and feedback as a committee member.

I have made many friends while at UMD but special consideration must be extended to my office mate Hailong Bai, with whom I have had many enlightening conversations about research and philosophy of science. James Dottin has provided much support as we have journeyed together through our master degrees.

I must acknowledge Kyoko Okino, and Masakazu Fujii of the University of Tokyo, for their support while I was on a research stay through the NSF/JSPS Summer Institutes.

Lastly I would like to extend great gratitude for Maggie McAdam for her endless support and patience through the end of it all.

Table of Contents

List of Tables

List of Figures

1 Introduction

Large low-angle detachment faults were recognized at mid-ocean ridges in the early 1980's (Dick et al., 1981). These long-lived detachments expose plutonic and metamorphic rocks that are rarely seen at mid-oceanic ridges, which together form oceanic core complexes (OCCs) (Dick et al., 1981, Tucholke et al., 1998). Initially identified at transform boundaries or in correlation with inter-segment discontinuities, OCCs have now been found worldwide and, in particular, are ubiquitous at slow-spreading ridges (Carbotte et al., 2015 and references therein). In the 12º - 15º N segment of the mid-Atlantic ridge, OCCs occur at all locations within the segment (Smith et al., 2008). In this thesis, I use the topographic profile of OCCs to constrain the elastic thickness of the oceanic lithosphere and search for correlations between elastic thickness and the location of oceanic core complexes within a segment.

1.1 Mid-Ocean Ridges

Mid-ocean ridges grant insight into fundamental processes in the interior and the exterior of the Earth. An expression of convection in the mantle, they accommodate divergence between tectonic plates and are where much of the oceanic crust is formed (Turcotte & Oxburgh, 1967; Schubert et al., 2001). The mid-ocean ridges, originally thought to be the primary mover of the plates, are now thought to be a minor, if not insignificant, driver of plate motion (Turcotte & Oxburgh, 1967; Conrad & Lithgow-Bertelloni, 2002). The relative contribution of the upwelling mantle at the mid-ocean ridges to the plate-scale force balance is now mostly accepted to be negligible, with slab pull, basal drag, and gravitational sliding away

from topographically high mantle (also called ridge push) being the dominant plate-driving forces (Conrad & Lithgow-Bertelloni, 2002). The mid-ocean ridges sit atop this convection in the mantle, and respond to the forces by either filling the space with magma via dikes, or else normal faulting (MacDonald, 1982). The fact that the mid-ocean ridges are mostly passive boundaries is crucial to understanding how the plates diverge, and in turn what the response of the mantle to their movements will be (McKenzie, 1969).

1.1.1 Classification of Mid-Ocean Ridges

Three classes of mid-ocean ridge, fast, intermediate, and slow, have been defined according to their spreading rate (Macdonald, 1988). A fourth class, ultraslow, and another intermediate class lying between slow and ultraslow have since been proposed, with varying acceptance, resulting in a classification system composed of three 'primary classes' and two 'intermediate classes' of mid-ocean ridge (Dick et al., 2003). Researchers have identified many relationships between spreading rate, morphology and rock chemistry (e.g. Carbotte et al., 2015). The faster the spreading ridge the more linear the ridges axis and the less interrupted it is by discontinuities (Figure 1.1 and Figure 1.2; Carbotte et al., 2015 and references therein; Dick et al., 2003). Crustal thickness also increases slightly from slow to intermediate spreading rate, although it may decrease slightly at the fastest spreading rates (White et al., 2001).

The distribution of ages of the oceanic crust highlights the variety of spreading rates, and varying asymmetry of the ridges spreading rates around the world (Figure 1.1; Müller et al. 1997; 2008). Local asymmetrical spreading (Müller et

al., 1998; 2008) has been proposed as a primary result of oceanic core complex development (Escartin et al., 2008). In Figure 1.1, large scale asymmetrical spreading can be seen at nearly every single ridge, however the scale is too large to see local asymmetry.

Fast spreading ridges open at a greater than ~80 mm yr^{-1} full-spreading rate (FSR). They have a pronounced axial high bounded by normal faults, and topographically form an inverted 'V,' (i.e. axial high) centered on the ridge axis (see Figure 1.2 a). Intermediate spreading ridges exist between ~50-80 mm yr^{-1} (FSR) they are characterized by pronounced abyssal hills, and moderate axial valleys (see Figure 1.2 b). Slow spreading ridges spread between ~20 and 55 mm yr^{-1} FSR, have deep rift valleys, but variable relief (400 - 2500 m). The second Intermediate class of Dick et al. (2003) lies below 20 mm yr^{-1}, and above 15 mm yr^{-1} FSR. Ultraslow spreading ridges spread more slowly than ~20 mm yr^{-1} FSR. Ultraslow ridges have long amagmatic segments that are spread by tectonic activity. These ultraslow spreading segments can be separated by focused centers of magmatic output. These magmatic centers may reflect the shape of deep 'permeability barriers' that often focus melt to offsets in a ridge axis (Standish et al., 2008; Montési et al., 2011). An alternative hypothesis for these focused magma centers along ultraslow ridges is that the mantle is highly heterogeneous and does not melt at typical mantle potential temperatures (Liu et al., 2008; Zhao & Dick, 2013). The morphology of ridges with faster spreading rates may be sensitive to mantle heterogeneity or the thermal structure beneath the ridge, because of the higher intensity of plate-driven flow (Dick et al., 2003).

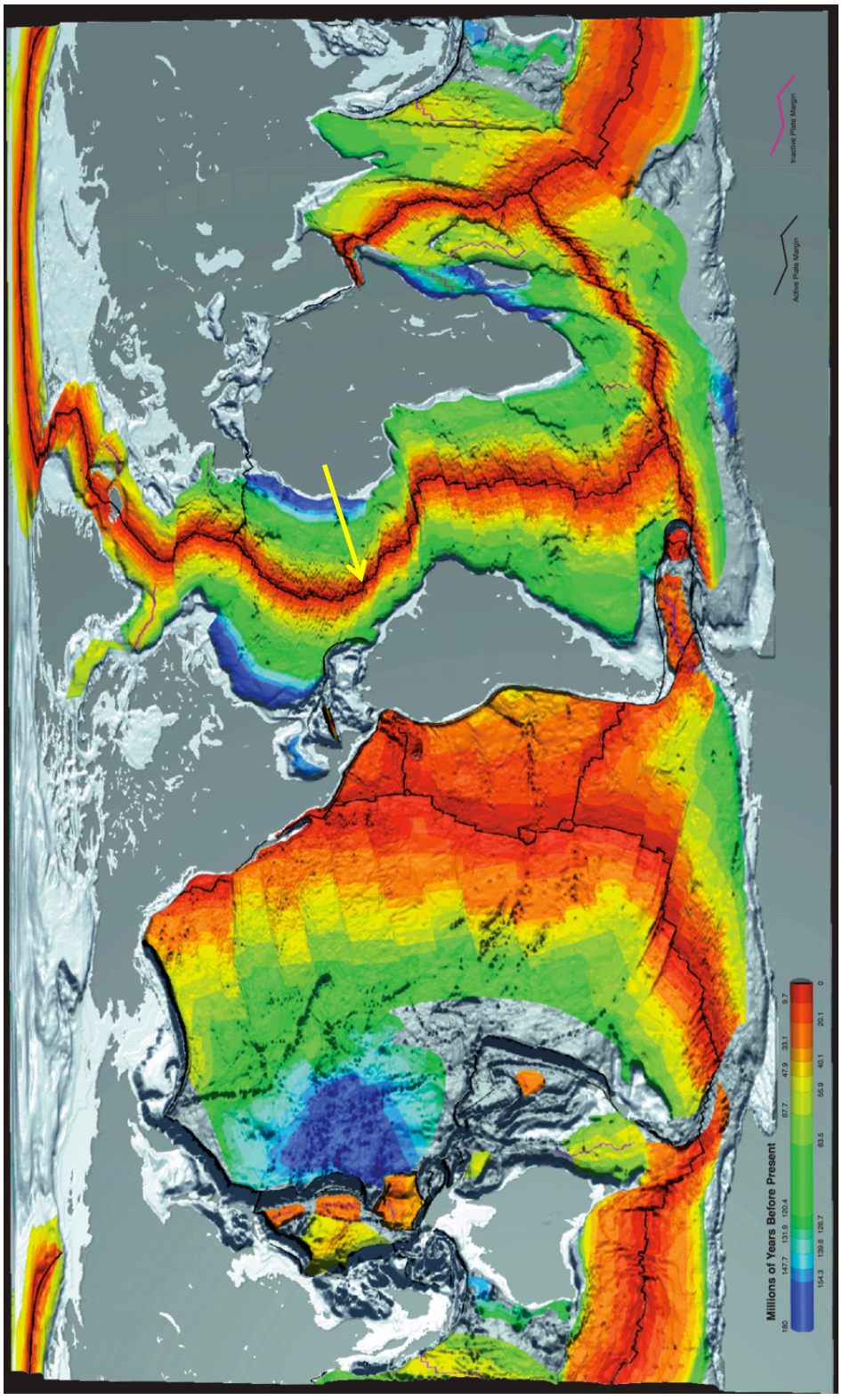


Figure 1.1: Map showing ages of the oceanic crust. Yellow arrow points to study area of this thesis. (Taken from Muller et al., 1996).

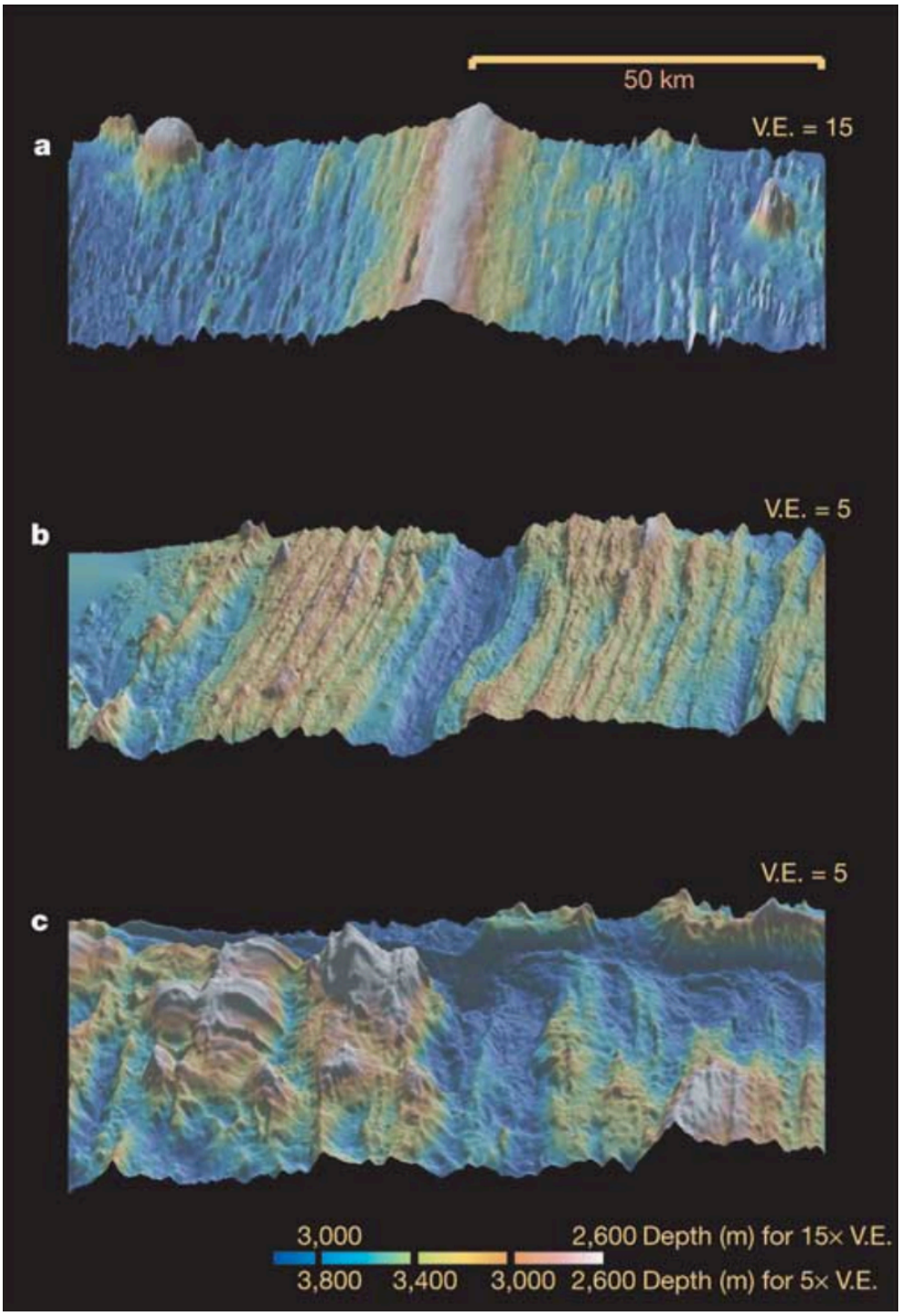


Figure 1.2: Bathymetric images of prototypical examples of three main spreading rates. a) Fast-spreading East Pacific Rise, b) Intermediate-spreading Southeast Indian Ridge, c) Slow-spreading Mid-Atlantic Ridge. (Taken from Buck et al., 2005).

1.1.2 Discontinuities along the ridge system

Along-axis ridge discontinuities are offsets along mid-ocean ridges that are classified into second, third, and fourth order discontinuities on the basis of their size and morphology (Macdonald et al., 1988). First-order discontinuities are transform faults, and they separate ridge segments by more than 30 km (Carbotte et al., 2015 and references therein). Second-order discontinuities include long-lived overlapping spreading centers and non-transform offsets, which have been shown to have a characteristic morphology that evolves through time (Carbotte and MacDonald, 1992). Second-order discontinuities separate ridge segments from one another by 2-30 km (Perram et al., 1993). Their genesis and evolution are still not completely understood, but their initiation appears to be related to a change in tension direction, due to either changes in plate motion vectors, or magma emplacement (Perram et al., 1993; Carbotte and MacDonald, 1992). Third-order discontinuities are smaller and less long-lived versions of second order discontinuities, and have no off-axis signature (Macdonald et al., 1988). Fourth-order discontinuities are even smaller, often not resolvable with most bathymetric survey. Otherwise, they are similar to second and third order discontinuities and are possibly related to magma intrusions that break the linearity of the ridges (Macdonald et al., 1988).

Discontinuities are generally accepted to be products of lithospheric processes (e.g. non-uniformly diverging plates; far field stress changes) rather than products of a heterogeneous asthenosphere (e.g. thermal, chemical, velocity), although geochemical variations do coincide with structural discontinuities (Carbotte et al., 2004; 2015).

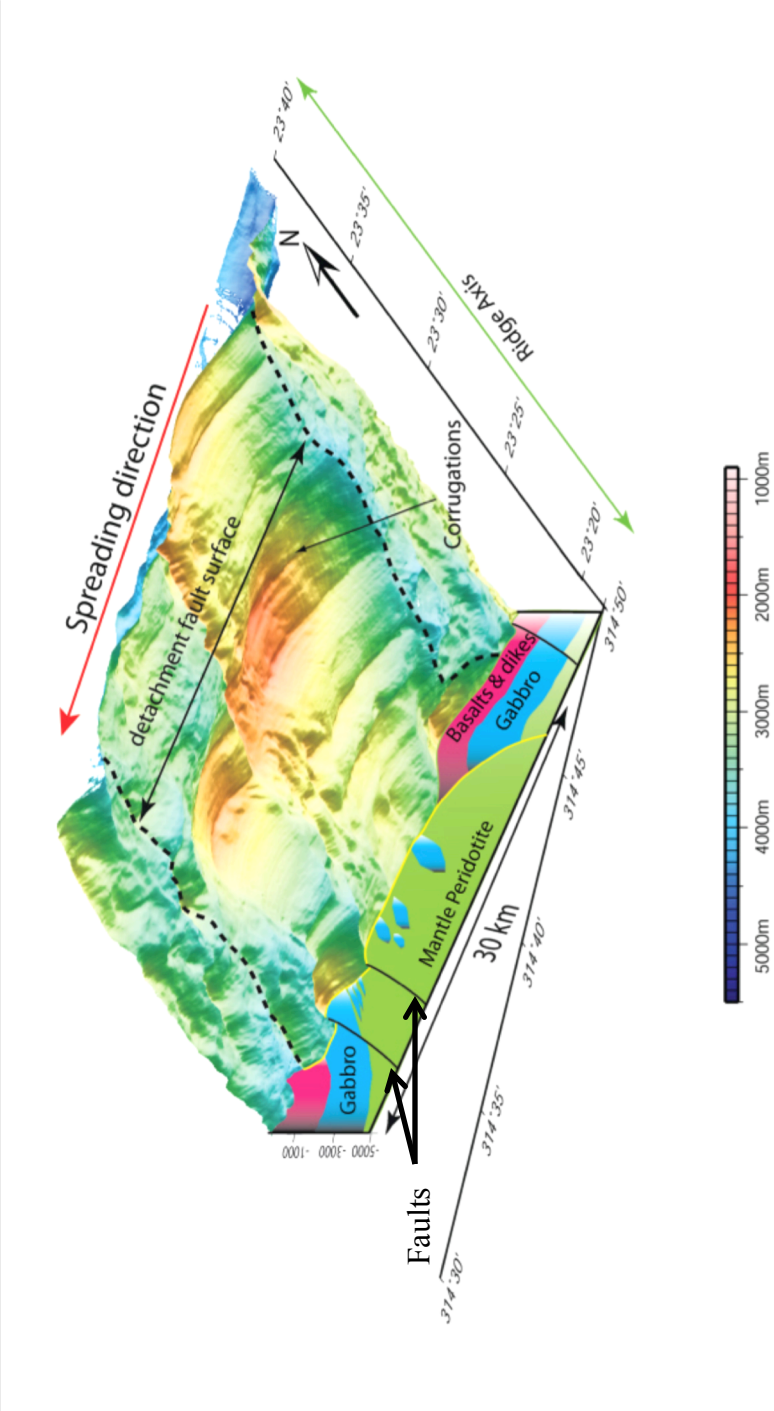


Figure 1.3: Perspective view of the Kane Oceanic core complex at the Mid-Atlantic ridge. Note the two faults shown have broken the exposed footwall surface. (Taken from NOAA Okeanos Explorer Program, MCR Expedition 2011)

At slow spreading ridges, oceanic core complexes are very often associated with a first or second order discontinuity, especially the inside corner of transform faults (Dick et al., 1981; Tucholke et al., 1998; Carbotte et al., 2015). Oceanic core complexes are also associated with second order discontinuities between ridge segments (Carbotte et al., 2015). These regions are where stress fields are complicated, the stress orientations vary, and the lithosphere is thicker and less ductile than at the center of segments (Shaw & Lin, 1996). This explains in part where oceanic core complexes are spatially located, that is, where tectonic forces overcome magmatic ones. Smith et al., (2008) describe the 13-15°N segment of the Mid-Atlantic ridge as being dominated by OCCs, showing how important these features can be. I will discuss how the morphology of these OCCs is possibly linked to variations in the elastic thickness of the oceanic lithosphere.

1.1.3 Core Complex Observations

Oceanic core complexes gained much attention in the 1990s when models to describe their formation were aided by higher resolution bathymetry, sample collection, and seafloor drilling (Mutter and Karson, 1992; Cann et al., 1997; Cannat et al., 1997). Since then, over fifty core complexes have been documented along many ocean ridges, and their role in accommodating spreading has been shown to be as great 100%, although most estimates place it between 50% and 80% (Tucholke et al., 2008; Macleod et al., 2008). Determining the composition of core complexes has been the purpose of many cruises, and the questions of their overall composition as well as the intensity of internal deformation remain debated (Carbotte et al., 2015). Currently, it is accepted that OCCs are composed of a significant amount of gabbro

(implying the presence of melt), serpentinized peridotite, and mantle peridotite (Figure 1.3; Cannat et al., 1997). The gabbro corresponds mainly to uplifted and exposed preexisting lower crust, although some of the gabbro may come from periodic emplacement into a peridotitic composition (Cannat et al., 1997).

Long-lived detachment faults characteristically roll over as their footwalls are exposed (Figure 1.4; Buck, 1988). This surface is usually "corrugated" with low amplitude ripples appearing to be scratched into it, and may be controlled by spreading rate and or melt supply, as OCCs at Ultraslow ridges have been found containing no corrugations (Tucholke et al., 2008; Sauter et al., 2013). The source of the corrugations is unknown, but may be related to irregular brittle-ductile transition (Tucholke et al., 2008). The presence of serpentine along the detachment surface and the timing of its development with respect to fault slip may play a key role in the development of core complexes. Serpentine may be the primary reason that the fault surface is able to rotate to a low angle while continuing to slip (Cannat et al., 2009).

Oceanic core complexes are mainly found at slow spreading ridges, with some examples at ultraslow spreading centers, and a few at intermediate spreading centers (Smith et al, 2006; Hayman et al., 2011). Forming an OCC may require a precise balance between tectonic and magmatic activity (Buck et al., 2005; Tucholke et al., 2008; Olive & Behn, 2013): tectonic spreading is expected to be between 50% and 70% of the total divergence when OCCs form (Tucholke et al., 2008). However, this expected range has not been supported by recent research (Sauter et al., 2013; Mallows & Searle, 2012). Sauter et al. (2013) report long-lived detachments leading to OCCs that formed when tectonic spreading was much greater than 80%. Mallows

& Searle, (2012) report tectonic spreading between 25% and 44% for their analysis of the MAR between 12.5º N and 14º N, a subsection of the 12º - 15º N segment.

Oceanic core complexes are identifiable by their characteristic shape in profile and in map view, by their corrugated surfaces, the presence of serpentine and gabbro in the corrugations, and of basalt at the breakaway. A simple model for their growth is shown in Figure 1.4. The fault surface dips initially at 60^{o}. As heave increases the growing height of the footwall cannot be supported by the elastic lithosphere and begins to flatten out, while the active fault surface remains dipping at ~60^{o}. After deformation, the exposed footwall takes on the characteristic shape seen in the last frame.

Three possible scenarios of mid-ocean ridge morphology are shown in (Error! Reference source not found.). In a) fault initiation develops, and is proposed to be the same for c) – d). In b) the fault continues with no interruption until a pristine OCC surface has formed. In c) faults move off axis and are replaced by younger faults before they form OCCs. Finally, in d) slices of upper crust are carried atop a continuously slipping footwall. Scenario d) is indistinguishable from c) in bathymetry maps, but its underlying structure is actually similar to b), that is, case d) includes a fully formed OCC.

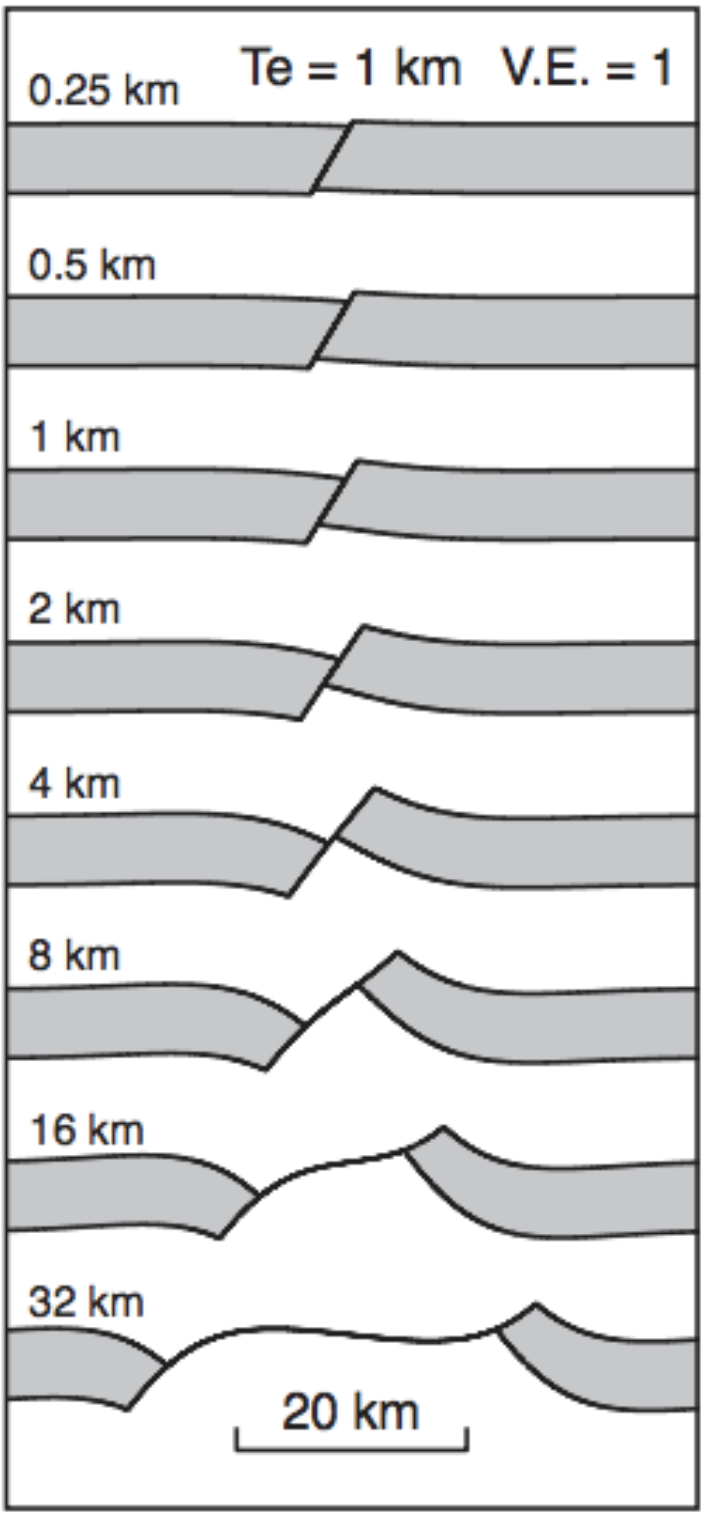


Figure 1.4: Model of Oceanic Core Complex evolution. Each profile is labeled by the heave of the fault. T_e is the elastic thickness of the lithosphere. (Taken from Schouten et al., 2010).

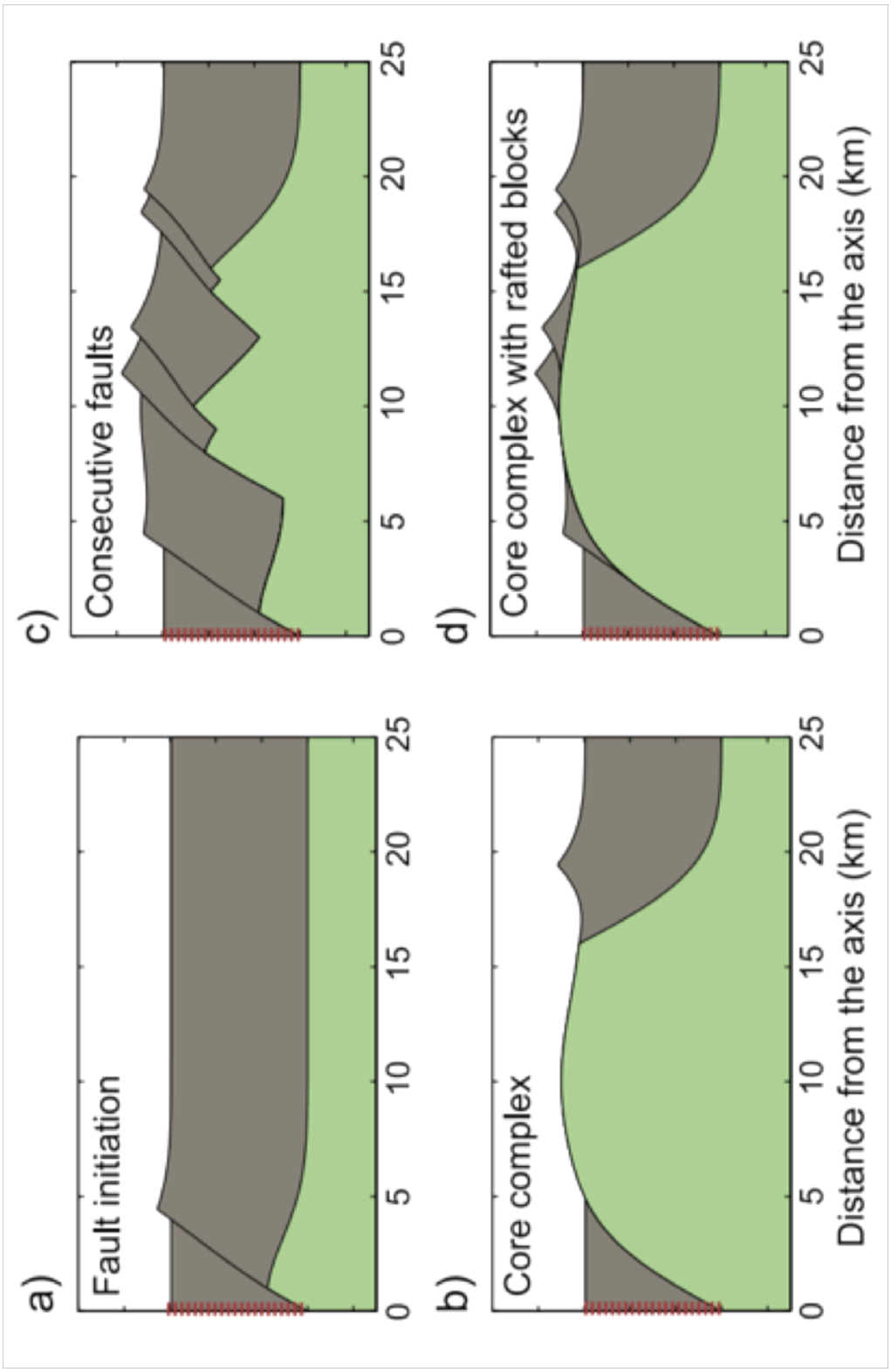


Figure 1.5: Models of off axis features at mid-ocean ridges.

1.2 OCC Modeling

Core complexes have been studied on land since the early 20th century (e.g. Daly, 1912; Misch, 1960; Wheeler, 1963; Coney, 1980). However, their importance for the development of large extensional provinces like the Basin and Range was not fully grasped until the early 1980s (Wernicke and Burchfiel, 1982). They were understood to expose the 'core' of the crust and are abundant in the mountain belts in Western North America during extension. Numerous models have been proposed to explain the emplacement of core complexes (Lister & Davis, 1989). The specific model that I use in this research was proposed by Buck (1988).

The lithosphere of the Earth and other planets has long been modeled as a thin elastic plate (Schubert et al., 2001). Roger Buck (1988) applied this concept to continental core complex formation, and showed that it was possible to bring up blocks of material, and have them passively ride along a fault that is undergoing continuous slipping. This modeling led to a differential equation that describes the deflection of an elastic plate and the resulting topography of a large-offset fault (see 2.2.2).

The same model can be applied to oceanic core complexes (Smith et al., 2008; Schouten et al., 2010). The results of deMartin et al. (2007), based on an analysis of seismic activity near the TAG seamount and associated OCC, suggest that the detachment surface roots ~6.5 km beneath the ridge axis and that the fault begins at a 60^{o} dip. The detachment fault reaches the surface about 3.5 km from the axis, which defines the termination of the OCC. Then, the slope of the core complex fault decreases progressively away from the termination (Figure 1.6 b; Figure 1.7 inset)

and steepens again at the end of the OCC, called the breakaway. By comparing the slope of the detachment surface and the elastic model of Buck (1988), Schouten et al. (2010) concluded that an effective elastic thickness of 0.5 – 1.0 km is typical of the mantle lithosphere beneath oceanic core complexes (Figure 1.6)

A few caveats limit the comparison of observed and modeled core complexes. The modeling I utilize assumes no post formation deformation, no sediment cover, and no passively carried rider blocks. For this reason, portions of an OCC where these assumptions are clearly not verified cannot be used to evaluate the model. For example, regions of an OCC with a rider block (as shown Error! Reference source not found. d) are not directly comparable with the model, which ignores such blocks. Deformation can also obscure the initial morphology of an OCC. For example, faulting can interrupt OCC morphology by effectively translating a portion of the OCC down and away from the rest of the OCC (Figure 1.7). One final way that OCC surfaces can be modified after formation is that erosive forces can carry sediment cover, volcanic deposits, or even entire portions of the OCC from one region to another in the form of slump blocks. All of these processes need to be identified before modeling an OCC and, absent the possibly of correcting for the modification, the affected portion of the OCC must be ignored, as I do in this work.

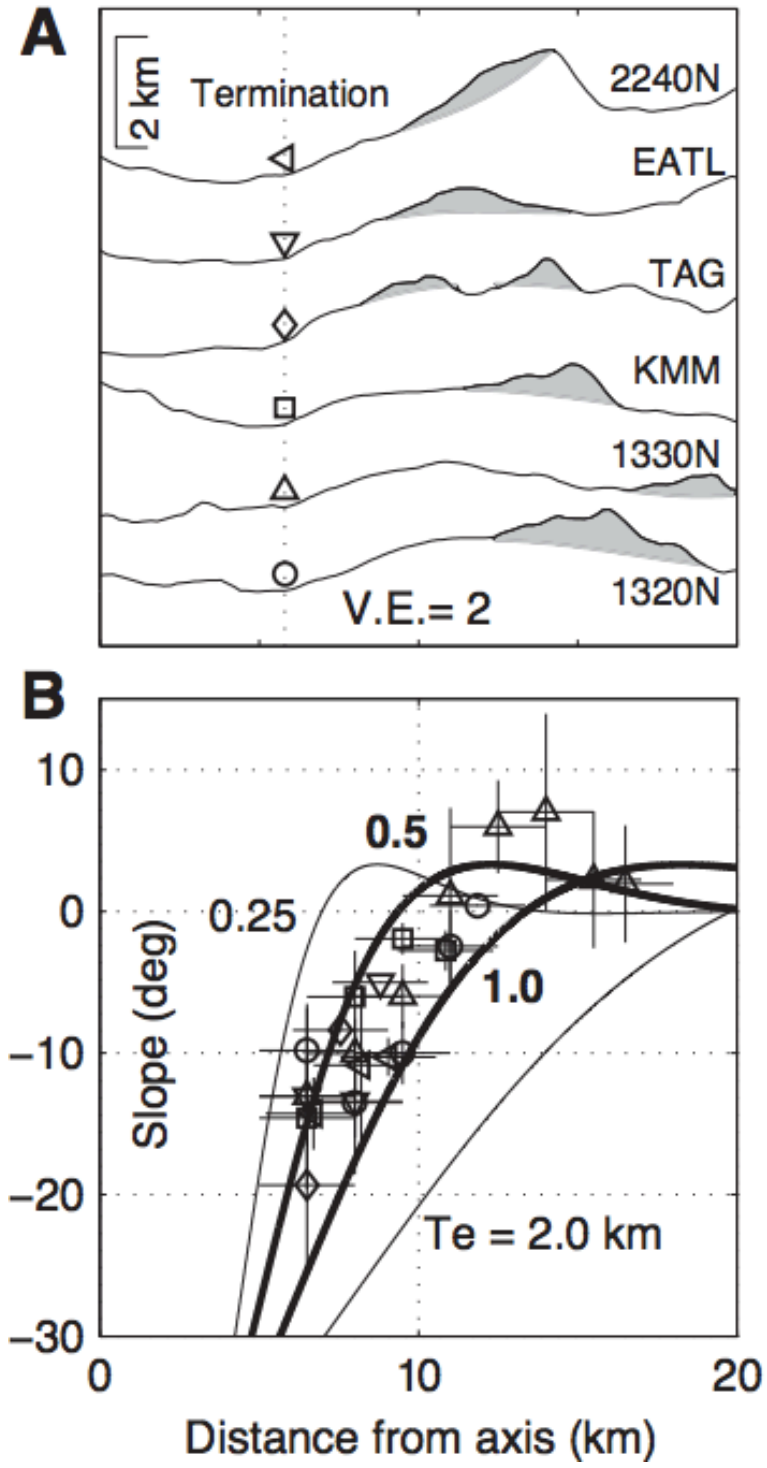


Figure 1.6: A) Bathymetric profiles of six different oceanic core complexes. The shaded regions are interpreted to be rafted blocks, ridge axis is indicated by a dashed line. 2240N-22^{o} 40' N, EATL-Eastern Atlantic, TAG-Transatlantic Geotraverse, KMM-Kane Megamullion, 1330N-13^{o} 30' N, 1320N-13^{o} 20' N, symbols are used in B to denote measured slope values. B) The curves are slopes of the topography models in (a). The markers are values taken from the core complexes in the figure below. (Taken from Schouten et al., 2010).

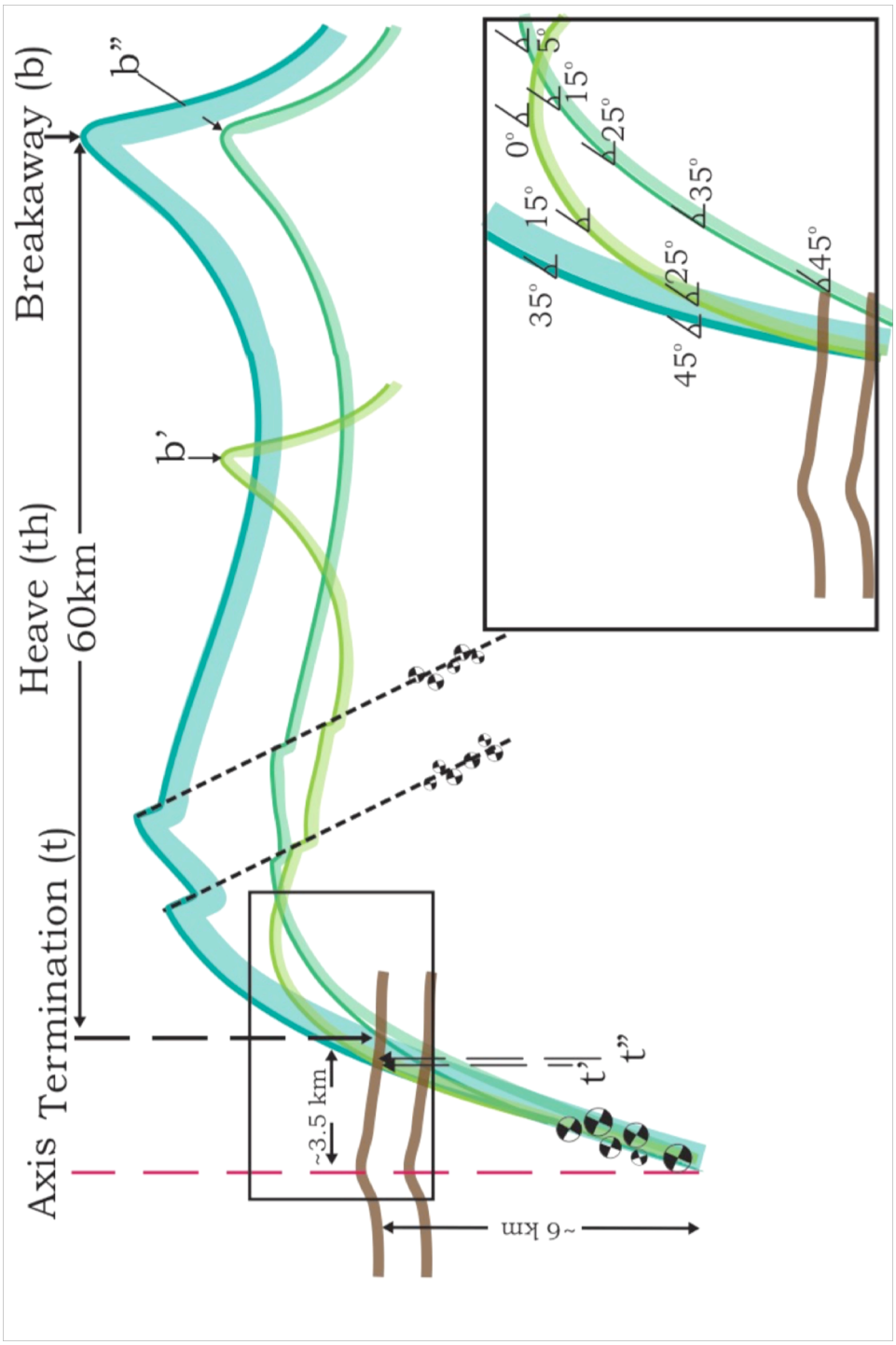


Figure 1.7: Cartoon of bathymetry vs. distance for various lithospheric thicknesses (teal is thick, green and yellow are thin) and heave (heave is indicated by breakaway distance (b)). The focal mechanisms are taken from deMartin et al., 2007, and show normal faulting at the OCC root, and secondary faults at the rollover point. Brown represents basalt, and depending on the axial infill thickness expose the detachment fault at different slope values.

1.3 Study Area: Mid-Atlantic Ridge Segment 12º - 15º N

The Mid-Atlantic Ridge contains the majority of the ocean's identified core complexes. They are so abundant that they can account for around 50% (and up to 70%) of one side of the ridge between 12 º and 35º N (Escartin et al., 2008). By analyzing these core complexes, I can determine whether there are systematic morphological differences related to the position of these features with respect to first and second order ridge axis discontinuities.

My study focuses on the region of the Mid-Atlantic Ridge between the Marathon and Fifteen-Twenty Fracture zones (Figure 1.8). I take advantage of high-resolution multi-beam bathymetry data collected along the Mid-Atlantic Ridge between 12° and 15° N by a number of cruises (Smith, 2013; Fujiwara, 1998; Bougalt, 1993). Intense seismic activity is confined to the northern and southern portions of the ridge segment, and correlate with OCCs (Smith et al., 2008). A few V-shaped bathymetric depressions are noticeable, and may be related to a mini-hot-spot that has been proposed in the segment at ~14.25º N (Dosso et al., 1991).

This region contains five previously identified on-axis OCCs as well as many off-axis OCCs (Smith et al., 2006, 2008; Cannat et al., 1997; Mallows & Searle, 2012). I consider here five new potential OCCs along the Mid-Atlantic ridge between 12º and 15º N as well as the five OCCs previously recognized there and three classical OCCs, Dante's Dome, TAG, and the Kane Megamullion (KMM). The new candidates OCCs are indicated on Figure 1.8. I refer to the OCCs and candidate OCCs in the study area from North to South as Features 1 through 10.

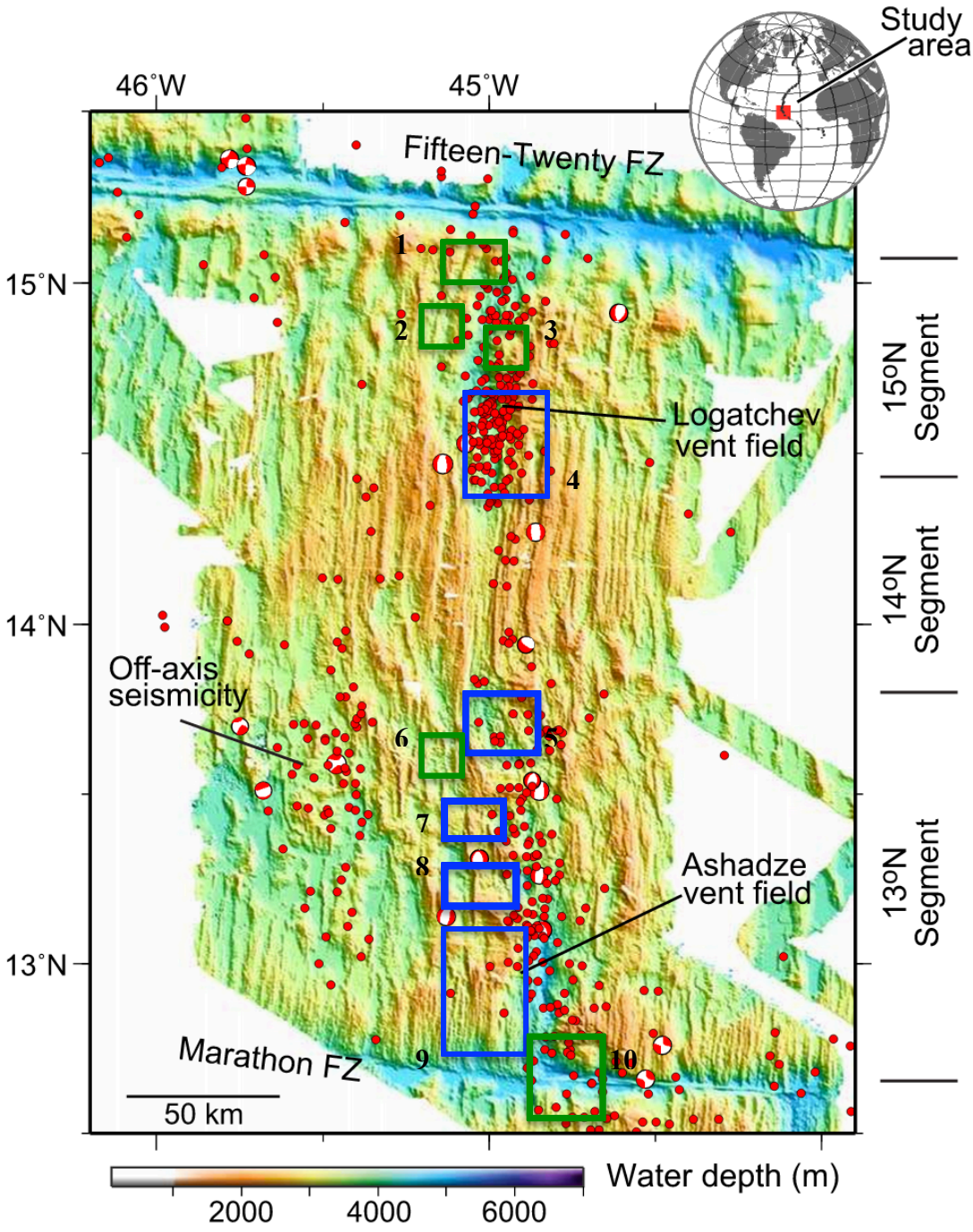


Figure 1.8: Bathymetric map showing the region where the preliminary analysis was conducted. Core complexes are boxed (blue = previous studies, green = this study). Red dots show earthquake epicenters including focal mechanism where available (modified after Smith et al., 2008).

1.3.1 Geological Observations

Three main rock types are found at mid-ocean ridges, each related to a specific aspect of oceanic crust accretion. 1) Basalts form by extrusive magmatism at mid-ocean ridges. When they cover the axial valley, they suggest a robust magma supply, and when they are accompanied by minor offset faults (i.e. abyssal hills), they indicate a mostly magmatically spreading ridge (MacDonald et al.,1988). 2) Gabbros and dolerites, basaltic intrusive equivalents, can be thought of as basalts that do not reach the surface. They compose the lower oceanic crust and imply that magma supply was sufficient to create lower crust. 3) Serpentines and peridotites, which often occur together represent upper mantle rocks. This occurs most easily at large-offset transform faults, depressions such as Hess Deep, and at ultra-slow spreading centers (Dick et al., 2003). Lower crustal and upper mantle lithologies can only be exposed at the seafloor due to tectonic activity and are commonly associated with detachment faulting within the inside-corner area of first- and second-order ridge discontinuities (e.g. Dick et al., 1981; Tucholke & Lin, 1994; Cann et al., 1997). Serpentine and peridotites are more common when magma supply is low, in particular at ultraslow spreading centers (Dick et al., 2003).

The region of the Mid-Atlantic Ridge between the Marathon and Fifteen-Twenty Fracture zones has been well surveyed lithologically and geophysically (Picazo et al., 2012; Mallows & Searle, 2012; Smith et al., 2006, 2008; Fujiwara et al., 2003; Cannat et al., 1997). A comprehensive study of the geology of the $13^{o} – 13^{o}$ 50’ N region is given in Mallows & Searle (2012). A geologic map of their interpretations is shown in Figure 1.9. Volcanic deposits fill the axial region and follow the

bathymetric contours created by the OCCs 1329, 1330, and 1348. This creates an axial valley that is variable in width, with the two southern core complexes emerging very near the ridge axis. This observation supports the hypothesis proposed by Mallows & Searle (2012), and MacLeod et al. (2009), that oceanic core complex fault terminations migrate further or closer to the ridge as the relative rate of tectonic spreading decreases or increases.

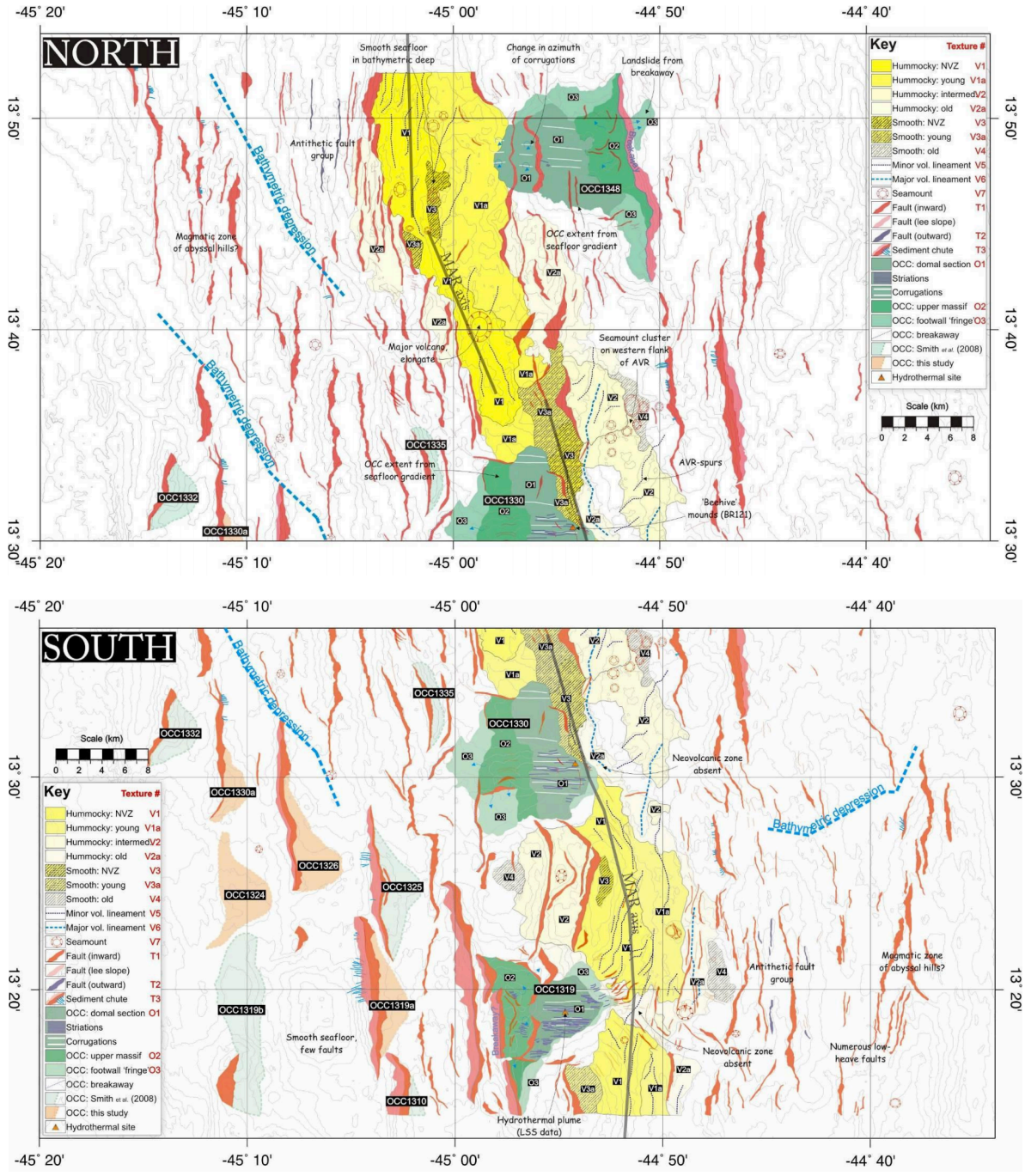


Figure 1.9: Geological maps of the North and South regions studied by Mallows & Searle (2012).

1.3.2 Hydrothermal Vent Fields

The ridge segment between 12º and 15º N hosts two well-studied hydrothermal vent fields; the Logachev vent field and the Ashadze vent field (Petersen et al., 2009; Ondréas et al., 2012). The Logachev vent field and associated ultramafic exposures are located at ~14.75º N (Figure 1.8). Petersen et al. (2009) identify extensive debris flows, which have obscured the original morphology of the Logachev region. Features 3 and 4 in this study encompass much of the Logachev region. Feature 3 is the youngest, and least disturbed portion of the Logachev detachment. Feature 4 is associated with the most elevated breakaway near Logachev, and has experienced substantial slumping as well as hosting hydrothermal vents. This extensive modification likely explains why this feature is not well represented by the model as we'll see later.

The Ashadze vent field is located ~ 13º N and, like Logachev, is a very large feature in the 12º - 15º N segment. It was analyzed in detail by Ondreas et al. (2012), using RESON SeaBat 7125 multibeam echo sounder data with a resolution down to 10 m. Two Remotely Operated Vehicle (ROV) dives collected high-resolution bathymetry covering ~3 km^2 of the youngest portions of the Ashadze vent field (Ondreas et al., 2012). Of the ~3 km^2 a large slump block ~0.38 km^2 in area was determined to have dropped ~300 m by a series of scarps (Ondreas et al., 2012). In addition to the slump block, hydrothermal vents, and craters covered much of the area (Ondreas et al., 2012). Feature 9 in this study is associated with the Ashadze vent field. The deformation processes may explain the lack of a distinct breakaway on the feature.

1.3.3 Thermal Structure Beneath mid-Ocean ridges

Previous work has proposed that segments along the mid-ocean ridge system each have their own thermal contour, with heat being focused towards segment centers by a combination of upwelling at segment centers, cooling at segment ends in contact with older plates, and hydrothermal circulation driven by magmatic flux (Figure 1.10; Phipps Morgan & Forsyth, 1988; Behn et al., 2004; Fontaine et al., 2008). It might be expected that the elastic thickness of the lithosphere, which reflects the thermal structure, would increase towards segment end.

The volcanic segment in the center of the 12º - 15º N supersegment may have a more complex thermal structure due to the presence of a possible hotspot. Dosso et al., (1991) analyzed dredge samples from 10º - 17º N on the mid-Atlantic ridge, and found eleveated light rare earth elements (LREEs) at 14º N Latitude. They proposed that a hotspot exists in the center of the 12º - 15º N super-segment. Using their geochemical analysis they constrained the emergence of the hotspot to ~18 Ma (Dosso et al., 1991). This can explain the robust volcanism evidenced by the linear basaltic abyssal hills which are seen at the center of the supersegment.

The thermal anomaly from this proposed micro hotspot's would be superposed on the normal segment-scale thermal structure, creating elevated temperatures somewhat North of the segment center, and away from the ends (Figure 1.10). A goal of this thesis is investigate whether this temperature structure can be resolved by analyzing the many OCCs in the super-segment.

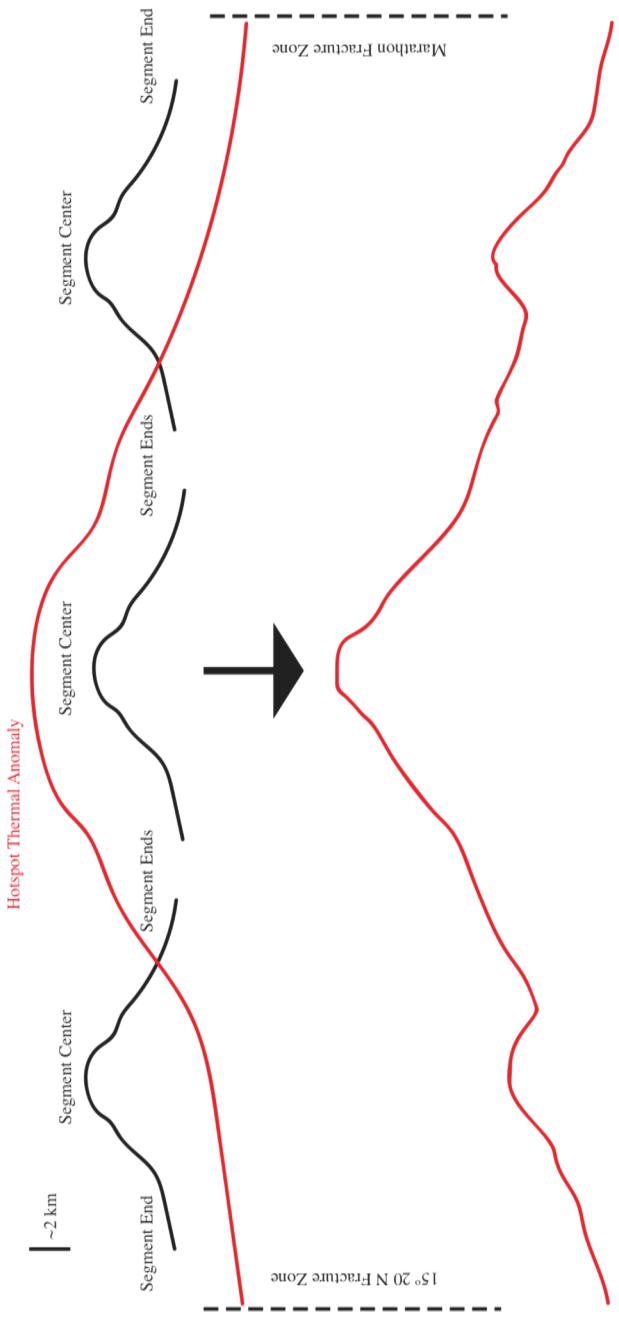


Figure 1.10: Schematic thermal anomalies beneath the three segments between the 15°20N and Marathon fracture zones separating the effects of a proposed hotspot at 14°N and and the “normal” segment-centered heat flow maxima..

2 Methods

Oceanic core complexes have been shown to be confined to specific portions of the mid-ocean ridges. Oceanic core complexes have been successfully modeled, and have been used to assign an average value for elastic thickness. However, an average obscures the unique properties of individual core complexes. It is unclear whether there is a link between oceanic core complex morphology and the underlying temperature structure of the lithosphere.

The goal of this project is to better constrain the properties of the lithosphere, such as elastic thickness, relative to the segmentation of the ridge axis using the morphology of OCCs. In order to do this, I 1) developed a procedure to identify possible OCCs from a bathymetry map, 2) created a suite of synthetic bathymetric profiles based upon elastic thin plate theory, 3) compared the synthetic profiles to the observations in order to constrain the properties of the oceanic lithosphere at each candidate OCC.

This section first describes the methods used for identifying OCCs and choosing 2D profiles for each feature. Then, the model equations and parameters are presented, followed by an overview of the inversion technique adopted to determine a best-fit model for each feature.

2.1 Identification of Oceanic Core Complexes

The features of OCCs are characteristic curvature, back-tilted breakaways, abrupt termination, corrugations, and lower crust and mantle lithology. The first requirement to identify and characterize OCCs is the availability of bathymetric data

of sufficient resolution to show surface corrugations, breakaways, and terminations. Generally this means the data should be gridded to ~100 m or higher resolution (Escartin et al., 2008).

The bathymetric map of the study area gridded to ~100 m and then processed using MATLAB to help identify features that exhibit distinctive properties typical of OCCs: 1) a breakaway that I identify in a bathymetric slope map as a steep outward dipping slope; 2) a corrugated surface, which runs perpendicular to the breakaway; 3) a ridge-perpendicular bathymetric profile featuring a characteristic roll over.

2.1.1 Bathymetry and slope maps

The bathymetry datasets were provided by Dr. Deborah K. Smith, and include multibeam data from Multibeam Bathymetry Surveys: KN210-05 – 2013, PI Smith, Debbie; KN182L03 – 2005, PI Lemmond, Peter; YK 98–05–1998, PI Fujiawara, Toshiya; Faranaut Cruise 1993 – Bougalt, H. They were combined and gridded using the Generic Mapping Tools (Wessel & Smith, 2013). To facilitate the identification of OCCs, the bathymetric maps and associated slopes are filtered using Fast Fourier Transform (FFT).

The bathymetric map was bandwidth-filtered using FFT to remove long-wavelength (>50 km Longitudinally, and >20 km Latitudinally) and short-wavelength (<5 km Longitudinally and <1 km Latitudinally) signals. This removes most of the variation of the map, leaving OCCs to stand out (Figure 2.1). While most OCCs are confined to these dimensions, there is at least one counter example, the Godzilla megamullion, in the South China Sea, which is about 100 km long, in ridge

perpendicular length (Ohara et al., 2001). However none of the OCCs located along the MAR and specifically in the 12º - 15º N segment are anywhere near 100 km long.

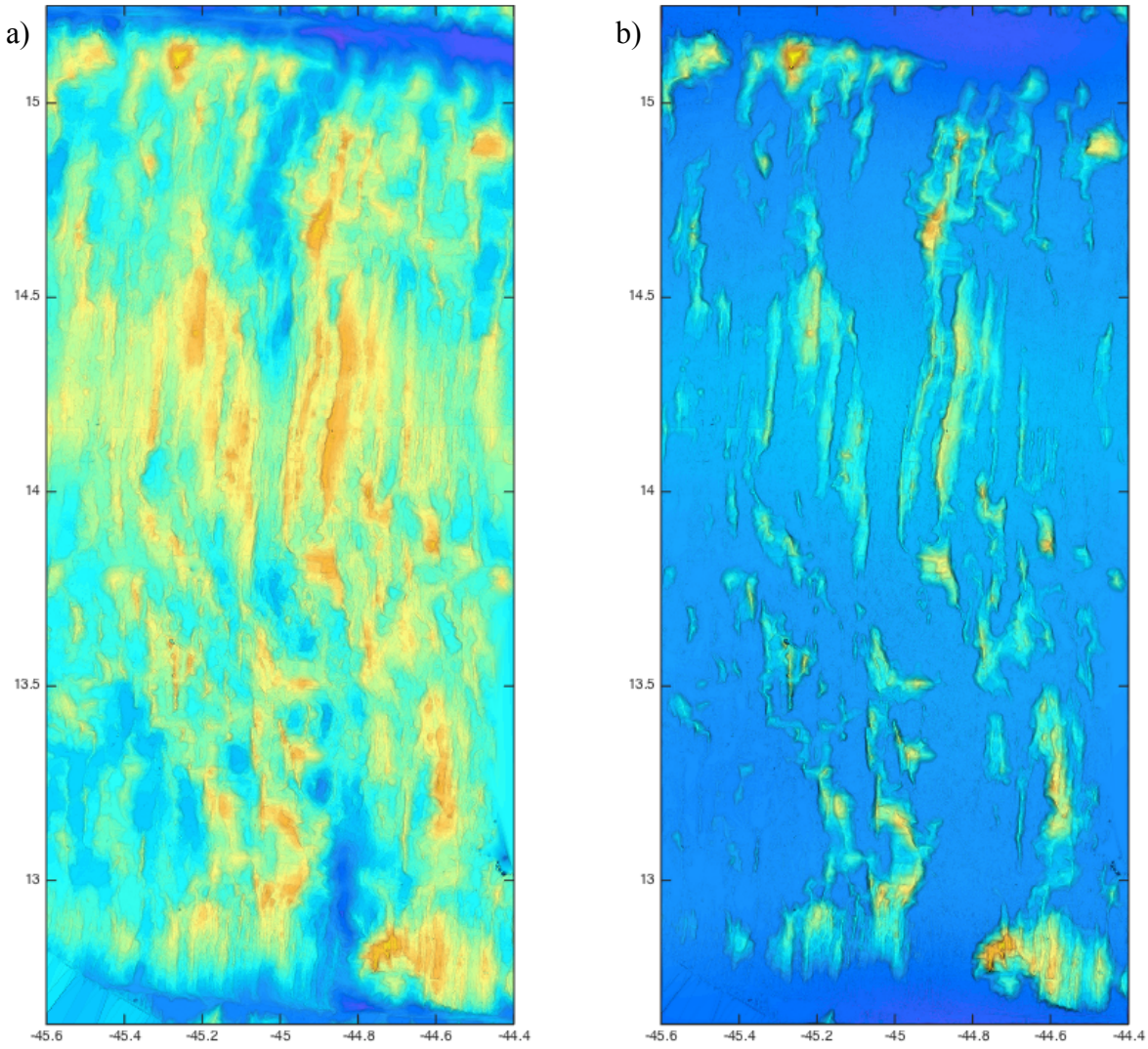


Figure 2.1: Original (a), and bandpass filtered (b) bathymetric map. Note how the OCCs stand out after removing long and short wavelengths of the region. Filtering was conducted in both the ridge parallel and ridge perpendicular directions.

To refine the identification of OCCs I analyze a map of the slopes of the region. MATLAB is used to create a slope map by taking the directional derivative of the bathymetry in spreading-parallel direction. The spreading direction I use is from Kyoko Okino's plate motion calculator using the MORVEL model for relative plate motion (DeMets et al., 2007). This map is then separated into outward- and inward-dipping slopes in accordance to their position with respect to the predefined ridge axis. The map is then filtered to only show slope values greater than 15º and less than 60º (Figure 2.2). Connected regions of high slope with lengths greater than 45 km are removed from the map because they are interpreted as abyssal hills, following Macdonald et al. (1992).

The inward-dipping slopes are assigned negative values. OCC footwalls can dip as steeply as ~45º degrees, and as shallowly as ~0º. To find connected regions (potential breakaways) that cover the corrugated surfaces I filter values out of the slope map that are greater than 0º and less than -45º. This technique was applied in the ridge perpendicular direction to identify corrugations. The corrugations are less well connected than inward and outward dipping surfaces, however, and so this map was less helpful than other topographic feature maps in identifying OCCs.

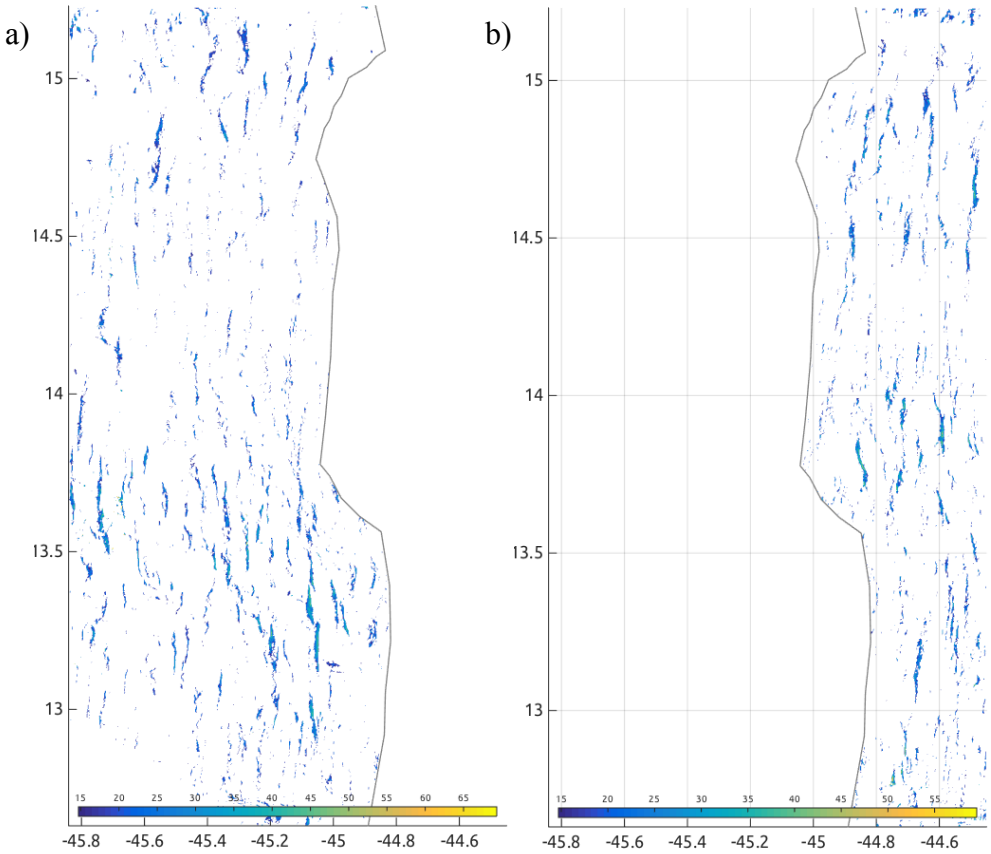


Figure 2.2: Map of outward-dipping slope of the study area filtered by angle (>15º, <60º). The linear regions about 30 km in length are associated with identified OCC breakaways

2.1.2 OCC profile extraction

The flexural models that I construct are two-dimensional (2D). Therefore, to compare model results with measured profiles, I must select representative 2D profiles through the features of interest. This is achieved through a combination of automated and manual techniques.

Each bathymetric profile is taken perpendicular to spreading direction according to MORVEL for absolute plate motion (DeMets, 2007). Three strategies are adopted to select bathymetric profiles over each candidate OCC: 1) Measure the bathymetric profile over the center of the feature in spreading parallel direction, which usually coincides with the cusp of the termination in plan view; 2) Measure over the part of the feature evaluated to be least modified, that is, where there has been the least amount of slumping and faulting as determined by the geology, and profile features (most pristine); 3) Starting at highest elevation or greatest back-slope value. This last strategy is usually coincident with the first.

The next step is to determine where the profiles of the features are sampling the original OCC, and where post-formation processes obscure the surface of the OCC. Figure 1.7 shows that faulting and slumping may break the original structure of OCCs in several places. Therefore, only the least modified portion of each profile is compared to the flexural models. The exact length of undisturbed OCC surface is different for each profile, and is chosen by eye. I look for evidence of faulting, and slumping events that heavily modify the original structure of the OCC. Which of the profiles described above is chosen to represent each OCC is determined on a case-by-case basis by visual inspection.

2.2 Flexural Model

In this thesis, I model the shape of OCCs using the response of a thin elastic plate to the load produced by a finite fault offset. The initial topography is defined by the offset of a fault breaking the surface of the seafloor and is parameterized by fault dip, fault offset, and fault position with respect to the ridge axis. The load corresponds to the buoyancy generated by the fault offset. The initial topography changes as a result of elastic flexure, which depends on the flexural rigidity.

2.2.1 Model setup

The initial topography is constructed by connecting two parallel lines, representing the broken seafloor, linked by a diagonal line that represents the fault surface (Figure 2.3). The initial topography is characterized by four parameters (Table 2.1): fault heave, fault angle, crustal thickness, and axial infill thickness.

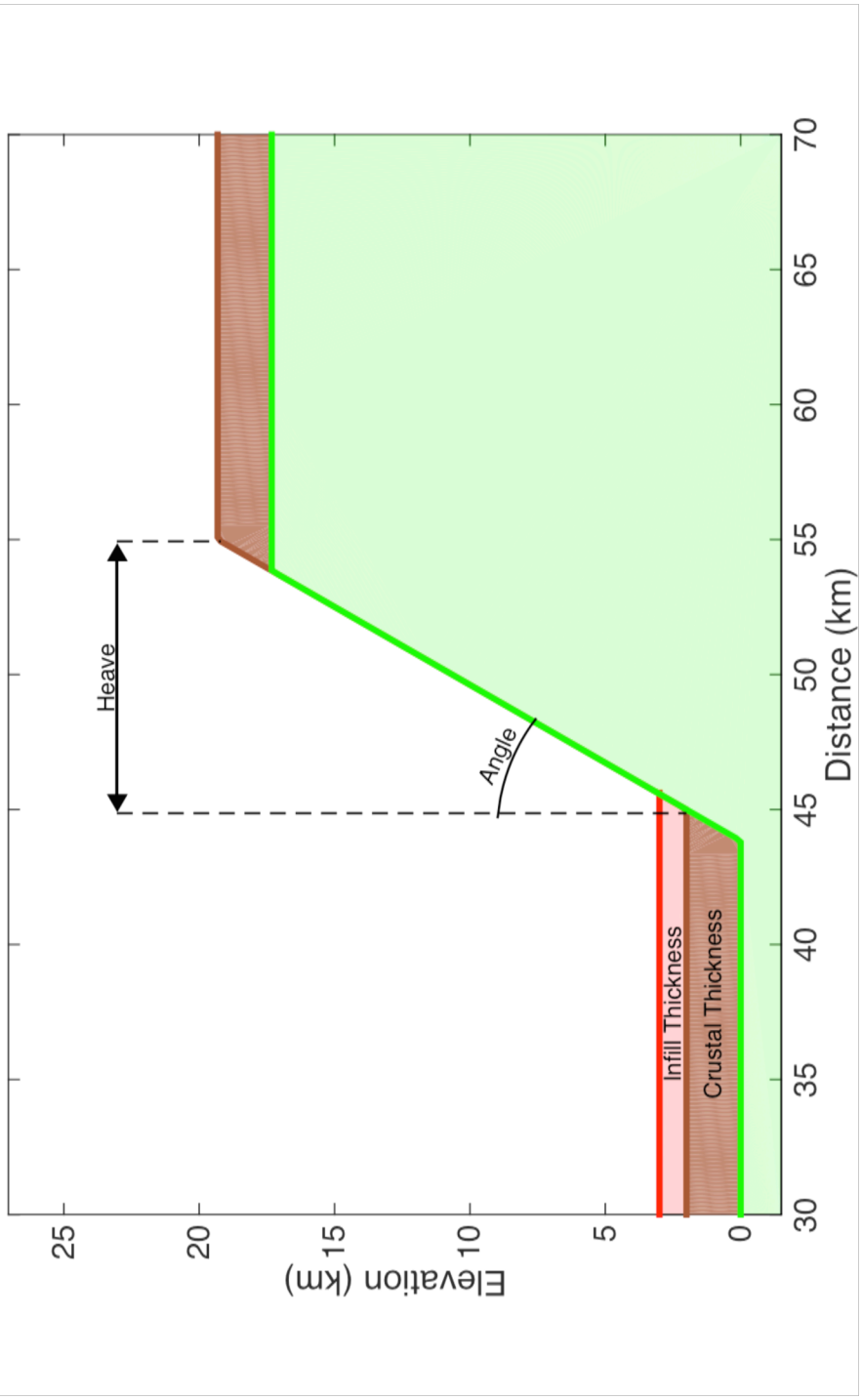


Figure 2.3: The initial configuration of the model. For this Figure: Infill Thickness is 1km; Crustal Thickness is 2 km; Angle is 60º; Heave is 10 km. The model assumes that the fault heave increased until this point, without flexing

2.2.2 Elastic plate equation

The load generated by fault offset deforms a thin elastic plate that describes the oceanic lithosphere. Thin plate theory is based on Equation 1, which relates vertical displacements, *w(x)*, with loads *q(x)* (Turcotte and Schubert, 2014)

$$q = D\frac{d^4w}{dx^4} + \Delta\rho_d g w \qquad (1)$$

where D is the flexural rigidity of the plate, $\Delta\rho_d$ is density contrast above and below the plate, g is gravitational acceleration (9.8 $m \cdot s^{-2}$), and d^4w/dx^4, is the biharmonic operator. Equation (1) is a fourth order ordinary differential equation.

The flexural rigidity of a plate is given by:

$$D = \frac{ET_e^3}{12(1-v^2)} \qquad (2)$$

where E is Young's modulus, v is Poisson's ratio, and T_e is the effective lithospheric thickness. The model assumes a constant D, although, in reality, the lithosphere would cool, and become stiffer as it moves away from the ridge.

The topographic load that may form an OCC can be spectrally decomposed into a sequence of periodic loads that displace the initially flat sea floor. The 4th order flexural equation is easily solved for a periodic load. The resulting deflections can be summed (backward Fourier-transformed) to predict the overall plate deflection and final topography.

A harmonic topography of amplitude q_0 and wavelength λ generates a load

$$q = q_0 \sin\frac{2\pi x}{\lambda} \tag{3}$$

Equation (1) then becomes,

$$q_0 \sin\frac{2\pi x}{\lambda} = D\frac{d^4 w}{dx^4} + \Delta\rho_d g w \tag{4}$$

Since the load is periodic, it can be shown that the solution is also periodic with wavelength λ, giving

$$w = w_0 \sin\frac{2\pi x}{\lambda} \tag{5}$$

Substituting equation (5) into equation (4), and solving for w_0, we find

$$w_0 = \frac{q_0}{\Delta\rho_d + \frac{D}{g}(\frac{2\pi}{\lambda})^4} \tag{6}$$

Equation (6) links the deflection of the plate with the amplitude and wavelength of a harmonic topographic load.

It is instructive to examine two end-member cases. First, if the wavelength is sufficiently short, that is,

$$\lambda << 2\pi\left(\frac{D}{g}\right)^{\frac{1}{4}} \tag{7}$$

Then we are left with,

$$w \approx 0 \tag{8}$$

In this case, the deflection is vanishingly small. The elastic plate behaves rigidly and supports the short-wavelength load without deforming. If however, the wavelength is sufficiently long, that is,

$$\lambda >> 2\pi\left(\frac{D}{g}\right)^{\frac{1}{4}} \tag{9}$$

Then equation (6) becomes

$$w = w_{0\infty} = \frac{q_0}{\Delta\rho_d} \tag{10}$$

The topography is isostatically compensated, without a contribution from elastic flexure. This deflection is the maximum deflection that can be achieved in this model. We may now plug in the density contrasts, and the crust-water and mantle-crust interfaces,

$$q_0 = [(\rho_c - \rho_w)h_c + (\rho_m - \rho_c)h_m]g \tag{11}$$

where ρ_c is the density of the crust, ρ_w is the density of seawater, ρ_m is the density of the mantle, and h_c is the topography of the crust-seawater interface, h_m is the topography of the mantle-crust interface.

The denominator of Equation (10) is,

$$\Delta\rho_d = (\rho_c - \rho_w) + (\rho_m - \rho_c) = \rho_m - \rho_w \tag{12}$$

Substituting Equations (11) and (12) into Equation (10) gives,

$$w_{0\infty} = \frac{(\rho_c - \rho_w)h_c + (\rho_m - \rho_c)h_m}{\rho_m - \rho_w} \tag{13}$$

The ratio between the deflection at finite wavelength and the isostatic deflection is called the compensation, *C*,

$$C = \frac{w_0}{w_{0\infty}} \tag{14}$$

Substituting Equations (6) and (11) into Equation (13) gives,

$$C = \frac{\rho_m - \rho_w}{\rho_m - \rho_w + \frac{D}{g}(\frac{2\pi}{\lambda})^4} \quad (15)$$

We can see that C does not depend on the amplitude of topography but only on the relation between wavelength and flexural rigidity whereas $w_{0\infty}$ depends on the amplitude of topography, not on wavelength. Solving them separately then, can be more intuitive.

The topographic load defined in Figure 2.4 is Fourier-transformed into a series of harmonic loads. Given D, I calculate the compensation C associated with each wavelength and the isostatic deflection associated with the topographic amplitude. The deflection at this wavelength is giving by

$$w_0 = Cw_{0\infty} \quad (16)$$

The set of deflections w_0 is backward-Fourier-transformed into the physical domain to give the deflection produced by the actual fault-induced topography. The sum of that deflection and the original topography provides the modeled OCC topography.

2.3 Inversion routine

The comparison between the model and observations is done by calculating the misfit between a 2D profile (Section 2.1.2) and a series of model profiles produced by varying systematically the input parameters to the elastic flexure model (Section 2.2, see **Table 2.1**). The comparison is limited by destructive processes that modify OCC shapes and by the three-dimensional character of faulting and flexure. Nevertheless, it is possible to provide constraints on the model parameters that best explain the shape of real OCCs.

2.3.1 Forward model series

In these models, the densities, Young's modulus, Poisson's ratio, and gravity are fixed to *a priori* values. I use a mantle density of 3300 $kg \cdot m^{-3}$, crustal density of 3000 $kg \cdot m^{-3}$, a sea water density of 1030 $kg \cdot m^{-3}$, Young's Modulus of 10^{11} Pa, Poisson ratio of 0.25, and gravity value of 9.81 $kg \cdot m^{-2}$ (Schouten et al., 2008).

In addition to the parameters that enter the equations and directly affect the shape of the deflection, the model profiles also depend on one more parameter, the offset distance between the fault and the ridge axis. Offset simply corresponds to a translation of the model profile in the cross-axis directions. Although the inversion constrains the horizontal offset of the profile, the results presented below report rooting depth, which is simply related to offset through fault dip but it is more intuitive to interpret geologically. Thus, the total parameter dimensions that I am testing are elastic thickness T_e, fault angle α, fault heave fh, crustal thickness T_c, axial infill thickness T_{if}, and fault root offset x_0.

I calculated a suite of models for use in comparing to the observed profiles. The models are calculated for a range of values for each parameter (summarized in Table 2.1). I do construct the models in a large structure of matrices by looping through each varied parameter. I then compare each model's topography and slope values to the topography and slope values of the OCCs. In order to only consider models that fit reality, I sometimes create a subset of the entire suite, to use for specific parameters, these are 'Semi-fixed,' as I only consider a smaller ranged subset of the full parameter range. That is, only consider the values one spacing beyond and below the measured value for each feature.

Table 2.1: Range of parameters tested.

Parameter	Symbol	Values	Interval	Semi-fixed?
Elastic Thickness	T_e	**0.1 – 2.0 km**	**0.1 km**	**NO**
Fault Angle	α	**45^o - 75^o**	**5^o**	**NO**
Fault heave	**fh**	**1 – 20 km**	**0.5 km**	**YES**
Crustal Thickness	T_c	**0, 1 – 6 km**	**3 km**	**NO**
Axial Infill Thick	T_{if}	**0, 2 km**	**-**	**NO**
Fault root offset	x_0	**-3 – 3 km**	**0.25 km**	**YES**

The shape of the footwalls produced by selected forward models is shown in Figure 2.4 – Figure 2.7 to show the variability of model results.

In Figure 2.4, the elastic thickness is varied for a fixed heave of 10 km and a fault dip of 60°. We see how the thicker plates support higher topography. For such a heave, the footwall rolls over to become almost horizontal only for the elastic thicknesses less than ~200m. In that case, the slopes displays a characteristic two-hump shape profile with three inflexion points (see as extreme values of the slope profile in Figure 2.4b) whereas for the thicker elastic plates, the slope features a single inflection point roughly halfway through the detachment surface.

In Figure 2.5, the heave is systematically varied for two values of the elastic thickness, while fault dip remains at 60°. It is clear that the rollover is more pronounced for larger heave, as the topographic load otherwise increases. The critical heave required for rollover increases with elastic thickness. The topographic profile near the termination becomes invariant to heave once the rollover is fully developed. However, the heave of the OCCs studied here is small enough that this limiting case is unlikely to be realized. Therefore, we cannot make the assumption of a large heave limit, as Schouten et al. (2010) did.

Figure 2.6 shows the effect of the offset parameter, which trivially translates the profile laterally.

The effect of varying fault dip is shown in Figure 2.7 for a fixed value of elastic thickness and heave. As a shallower fault builds topography less quickly than a steep fault, the resulting topography is more subdued at lower values of fault dip, at least when the heave is short and/or the elastic thickness is high. Both shallow fault

dip and thin elastic plate result in subdued topography, but the shapes are easily differentiated in the slope profile: the detachment initiated by a shallowly dipping fault features a single inflection point. Figure 2.7b also shows that the maximum slope increases with fault dip.

The final two parameters that enter the model, crustal thickness and infill thickness, have a relatively minor effect on bathymetric and slope profiles. In most cases, it is not possible to constraint their values. Therefore, most results shown below assume $T_c = T_{if} = 0$ unless otherwise noted.

Two general aspects of the forward models considered here are worth noting. First, the roll-over becomes constant after certain values of heave (usually ~30 km), this finding allows my grid search to not need to consider values > ~30 km for heave. Second, the parameters often trade-off between each other, especially heave and elastic thickness. For that reason, we conduct restricted grid search in which the heave is fixed to the geological observed value, leaving only fault angle and elastic thickness to be varied systematically.

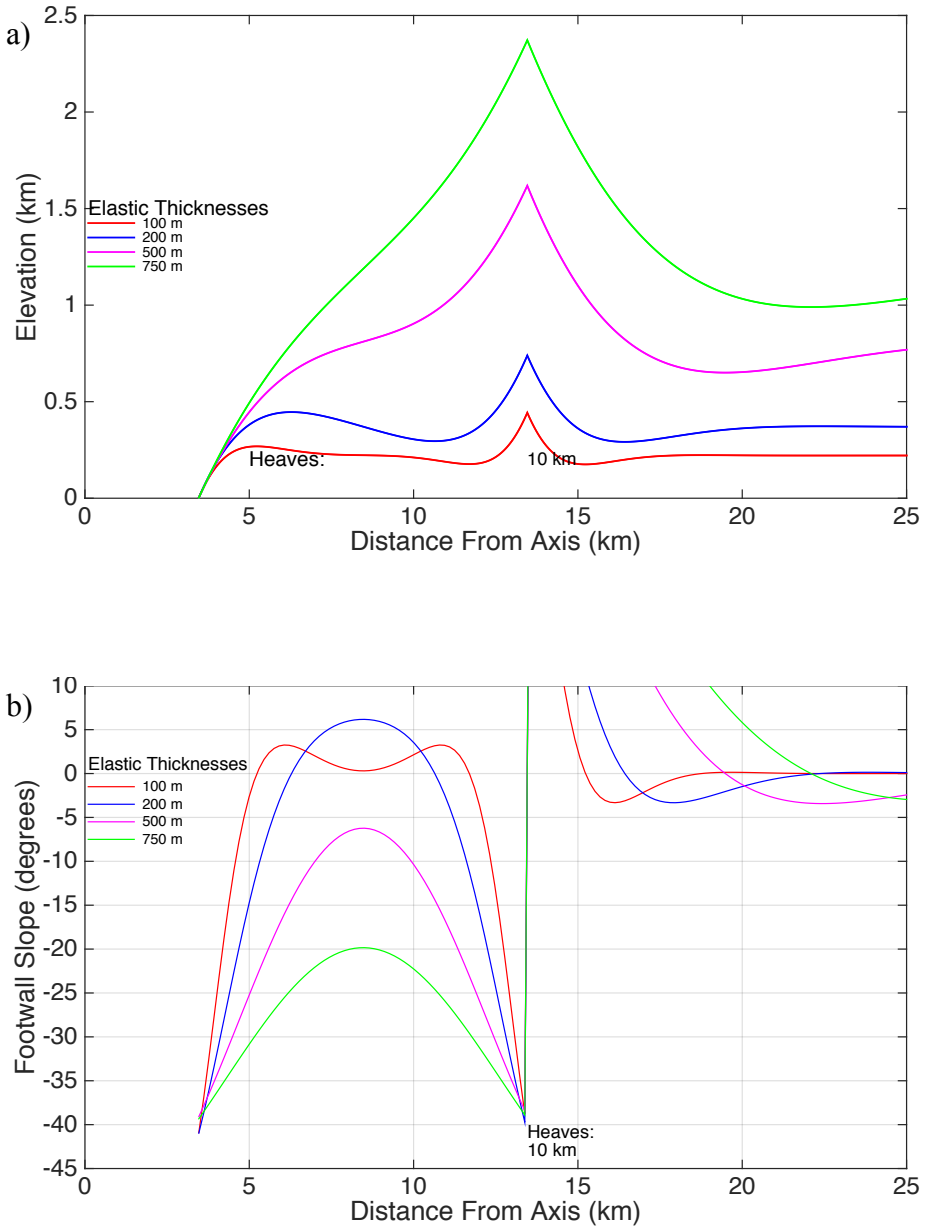


Figure 2.4: Modeled OCC topography (a) and slope (b) profiles to show how elastic thickness affects the models. For T_e = 0.1 km (red), 0.2 km (blue), 0.5 km (magenta), 0.75 km (green). A heave of 10 km, angle of 60º, infill and crustal thicknesses of 0 km.

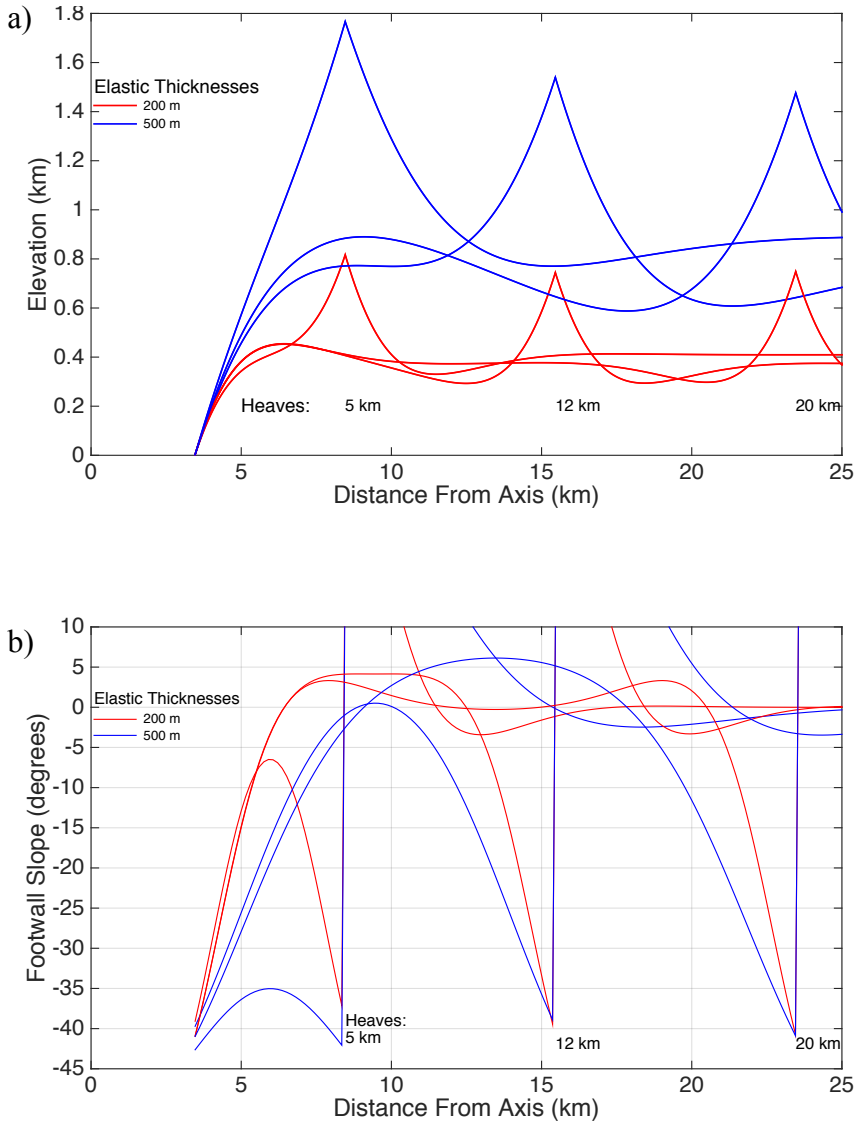


Figure 2.5: Modeled OCC topography (a) and slope (b) profiles to show how heave affects the models, for various values of elastic thickness. For T_e = 0.2 km (red), 0.5 km (blue). Heaves of 5 km, 12 km, and 20 km, angles of 60º, infill and crustal thicknesses of 0 km.

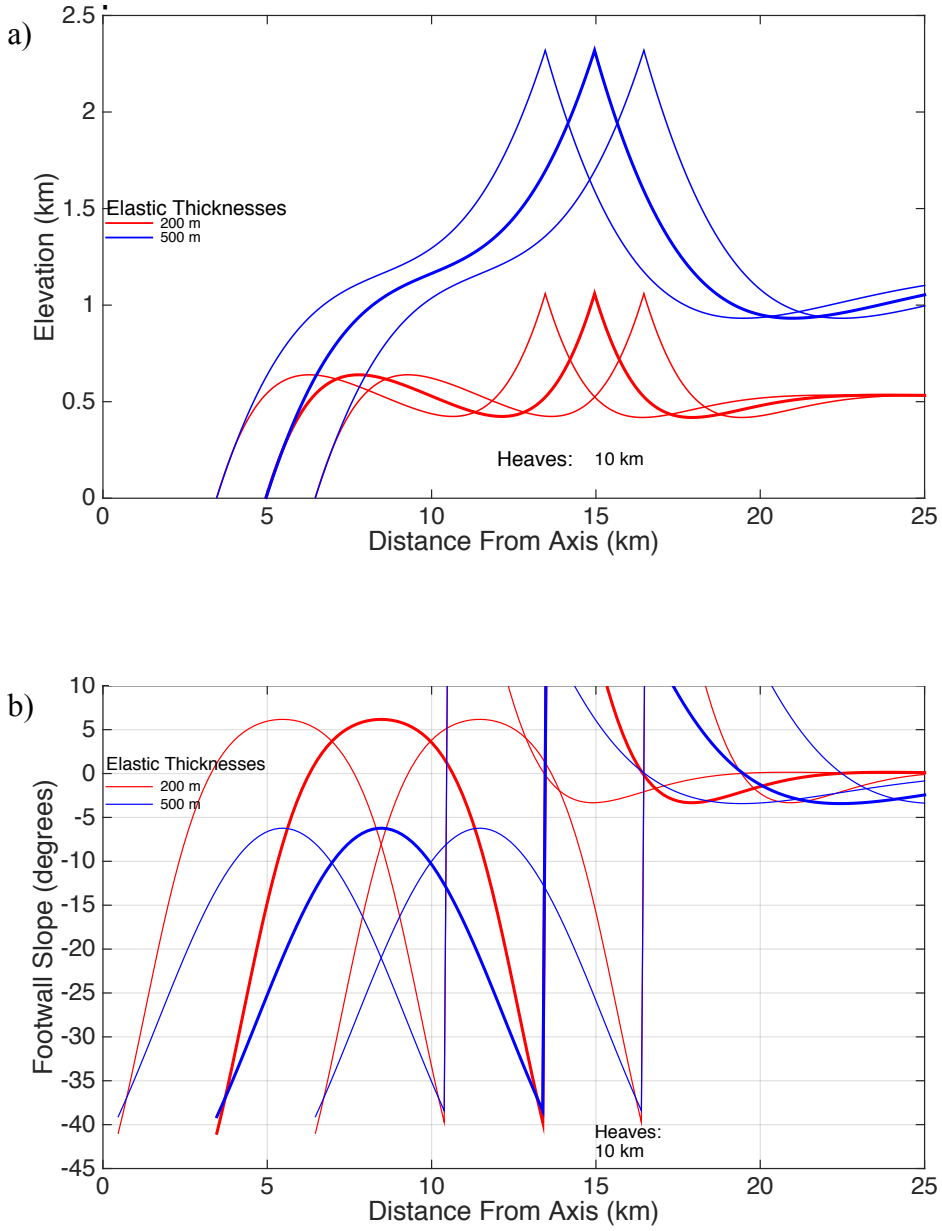


Figure 2.6: Modeled OCC topography (a) and slope (b) profiles to show how offset affects the models, with offsets of -1.5, 0, and 1.5 km. For T_e = 0.2 km (red), 0.5 km (blue), a heave of 10 km, angle of 60º, infill and crustal thicknesses of 0 km

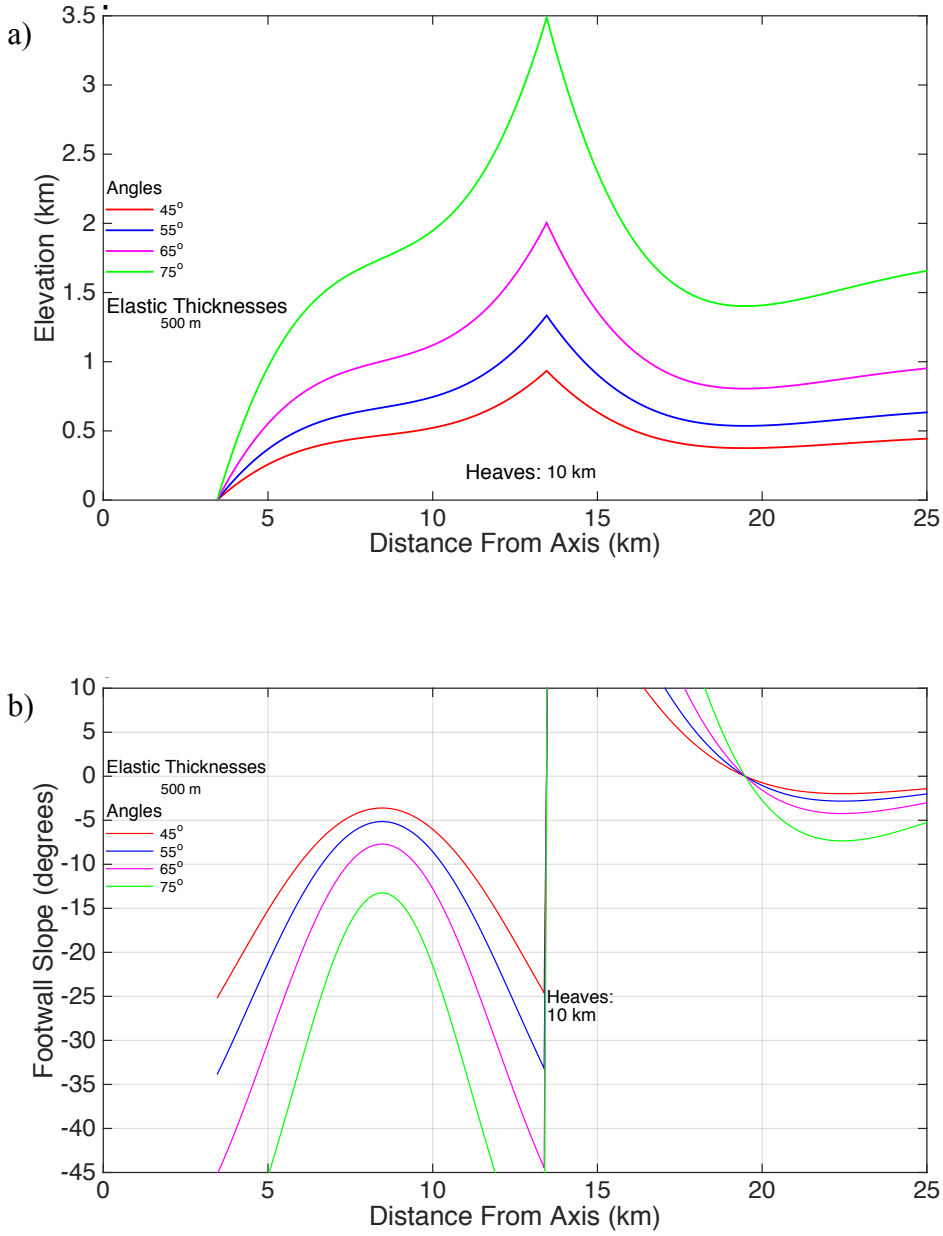


Figure 2.7: Modeled OCC topography (a) and slope (b) profiles to show how angle affects the models. Figure shows variation of angles of 45º, 55º, 65º, and 75º. T_e = 0.5 km, a heave of 10 km, infill and crustal thicknesses of 0 km.

2.3.2 Misfit

Constraints on elastic thickness and other parameters are obtained by comparing actual topographic and slope profiles to the predictions of a series of models in which input parameters are systematically varied. This systematic grid search approach is repeated for bathymetry and slope. Schouten et al. (2010) compared a similar model to ours with only slope values. However slope is derived from bathymetric data and is inherently noisier than the original bathymetric data because any deflection in bathymetry is amplified by taking its derivative. Thus the grid search is repeated using slope only, bathymetry only, and then a combination of both.

The goodness of fit is evaluated using the chi-square

$$\chi^2 = \sum_{i=1}^{n} \left[\frac{d(x_i) - m(x_i)}{\sigma} \right]^2 \qquad (17)$$

where $m(x_i)$ is the value of the model function at sampling point i, located at the x_i horizontal position, $d(x_i)$ is the corresponding value of the data point, and σ is the associated error. The error as assigned based on the variability observed on the profiles. I used an error of 4° for the slope and 100 m for the topography.

For each dataset (slope, bathymetry, or combination of both) I defined the best fitting model as the one for which the chi-squared misfit measure is the smallest. For the combined data, I vary the weight given to the misfit in bathymetry and the misfit in slope and search for the model with the lowest combined misfit. Often, that model is similar to the best-fit model considering only bathymetry, although this is not

always the case. The three best models give an idea of the range of conditions that explain relatively well the observations.

2.4 Test Case walk-through

As an illustration of the feature identification and modeling procedure, I describe here in detail the analysis of an OCC located around 13°50' on the Eastern side of the ridge in the study area. This feature was called OCC1350 in Smith et al. (2008) and OCC1348 by Mallows & Searle (2012).

2.4.1 Profile Picking

Starting from the regional bathymetric map of Figure 2.1, I see a continuous region of steep back slope around 13°50'N. Zooming in (Figure 2.8), I see in map view a feature that displays all the characteristics of an OCC, especially a U- shaped termination, corrugations on the exposed footwall, a gradually increasing slope on the footwall, and a steeper dipping breakaway. I now have a feature of interest in plan view over which I now need to draw a profile.

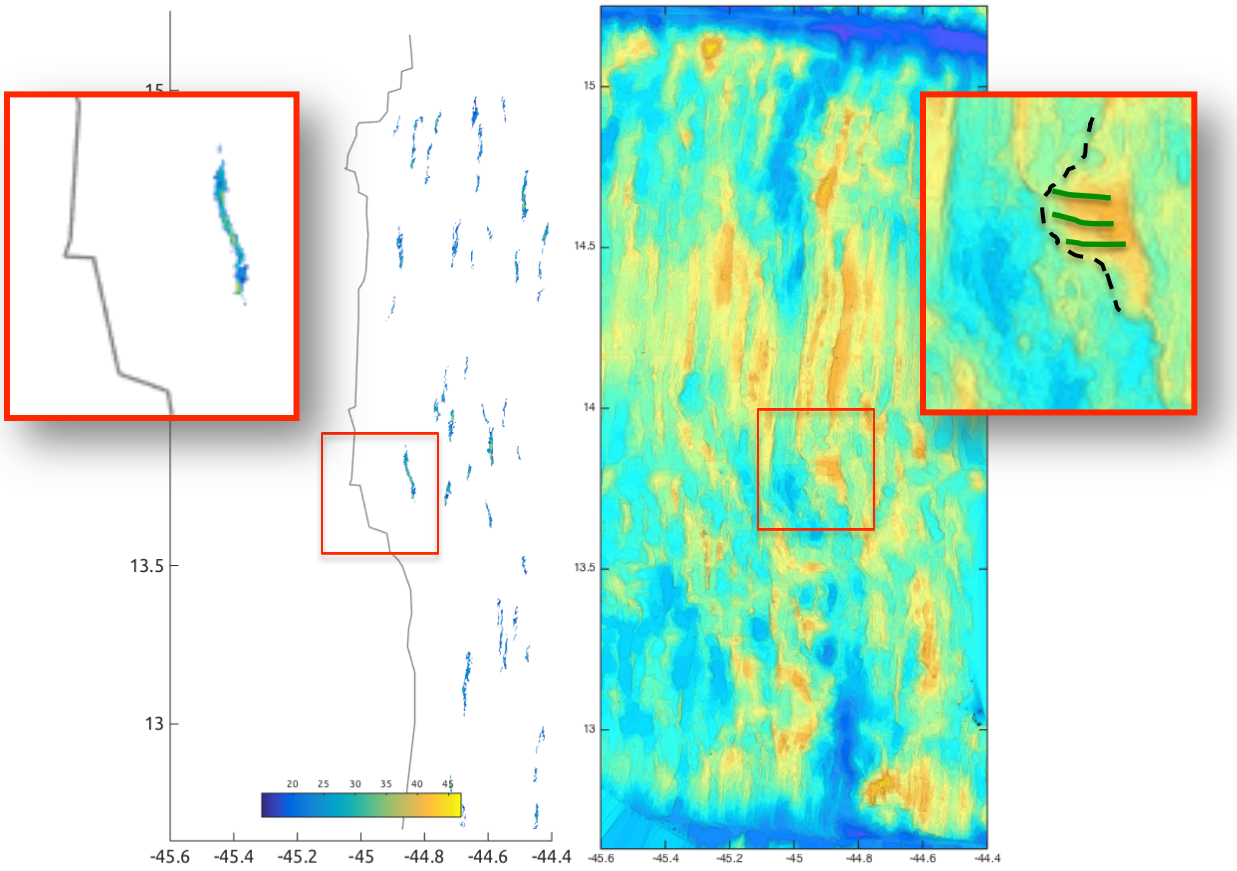


Figure 2.8: Filtered slope map, with identified back slope boxed and as inset. Bathymetry map showing the same feature boxed and as inset. In the inset, a U-shaped termination (black dashed line) and corrugations (green lines) are traced.

The next step is to zoom and in inspect the feature in detail, determining where faults, slumps or other post-formation deformations may have obscured the original surface. In this particular example, OCC1350 was included in a detailed study of the southern portion of the 12 – 15ºN region (Mallows & Searle, 2012), which resulted in a geologic map (Figure 2.9). As is seen in the mapping, a major fault cuts through roughly the middle of OCC1350, and this provides me with strong reasoning to only consider the exposed footwall to the West of the fault. Additionally, the footwall portions identified as "OCC: footwall 'fringe'" by Mallows & Searle, (2012) should be avoided (see Figure 1.9). Thus the region indicated by the black dashed line (Figure 2.8) is where I choose to draw a profile.

Consulting the bathymetric map of the OCC further as well as three-dimensional renditions generated in Matlab (Figure 2.10), I identify and test a few profiles within the region circled in Figure 2.10. Figure 2.11 shows profiles drawn at regular intervals over the entire feature to illustrate the more characteristic OCC regions of feature 5. Green is the best profile in the image, and closely-spaced profiles are drawn around it in the next step.

The same process is followed for each feature. Not all OCCs have the benefit of being in the area that Mallows & Searle, (2012) mapped in detail, but their interpretation of features assisted me when I was looking at OCCs outside of their study area.

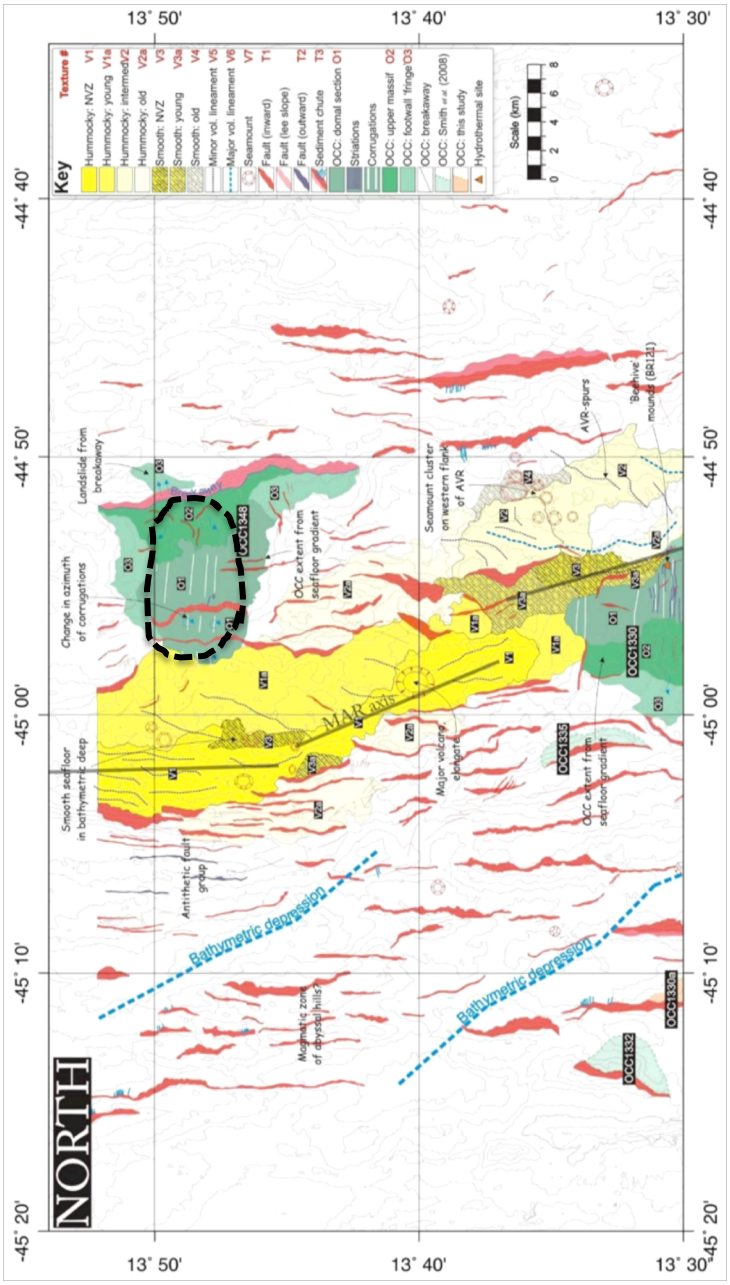


Figure 2.9: Geologic map of a portion of my study region, including OCC1350 (OCC1348, here). Yellow indicates axial region, greens indicate OCCs, shades and symbols show morphological differences. Red are fault surfaces, Note the many ridge-parrallel faults that cut through OCC1350. The region circled by dashed black line is the most pristine, and where I choose profile from. Figure modified after Mallows & Searle, (2012)

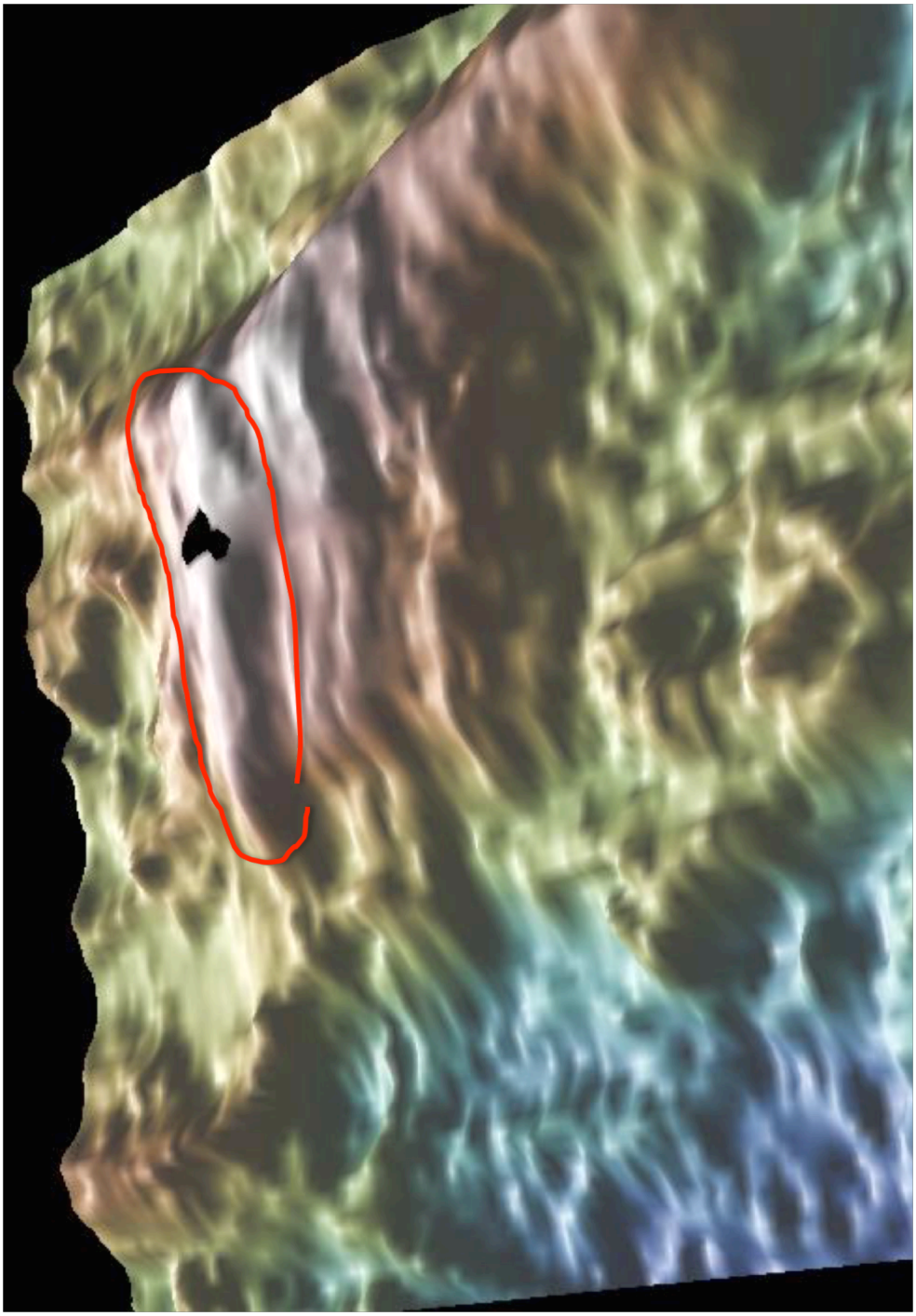
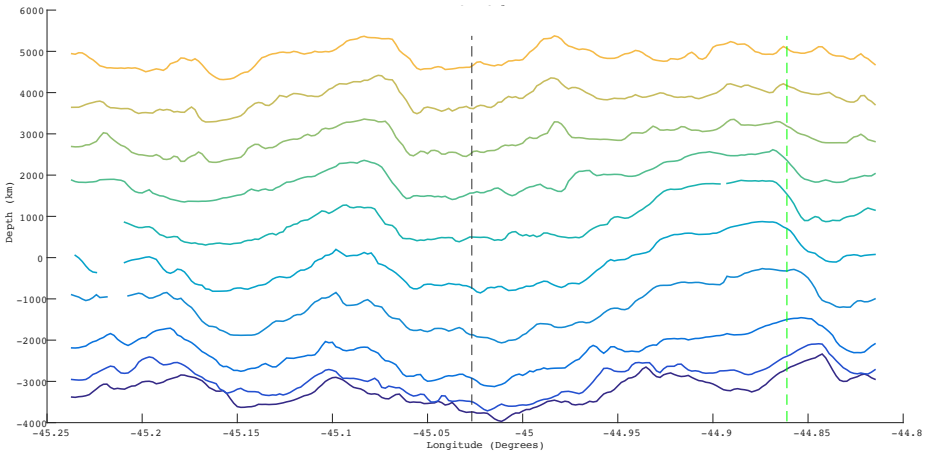


Figure 2.10: Three-dimensional ‘fly-over’ view of OCC1350. This view, aided by lighting changes, allows for good picking of profile locations. The red circle indicates the large corrugation which profiles are drawn over.



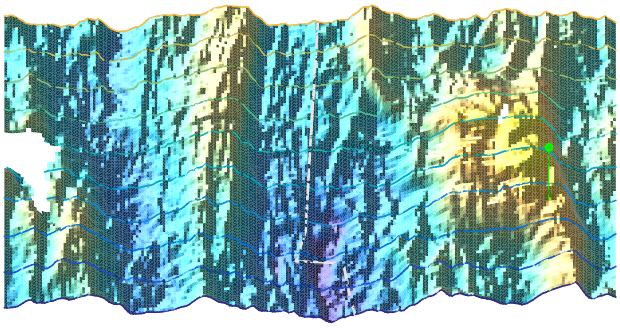


Figure 2.11: Plots of bathymetric test profiles over feature 5.

2.4.2 Inversion

I illustrate the inversion process using a synthetic profile to show the degree to which the inversion procedure successfully recovers the input parameters. The synthetic profile has parameters: T_e = 1000 m, fh = 12 km, α = 65º, T_c = 1 km, T_{if} = 1 km, and has random deviations starting halfway along the profile. The scatter represents surface modification processes such as slumping. Being able to state that an elastic model as input will be best matched by its own parameters in spite of that noise is an important test of the inversion process.

The synthetic profile with and without noise is shown in Figure 2.12, where the black outline highlights the region selected for calculation of the misfit. Figure 2.13 shows how the profile compares with a selection of forward elastic models.

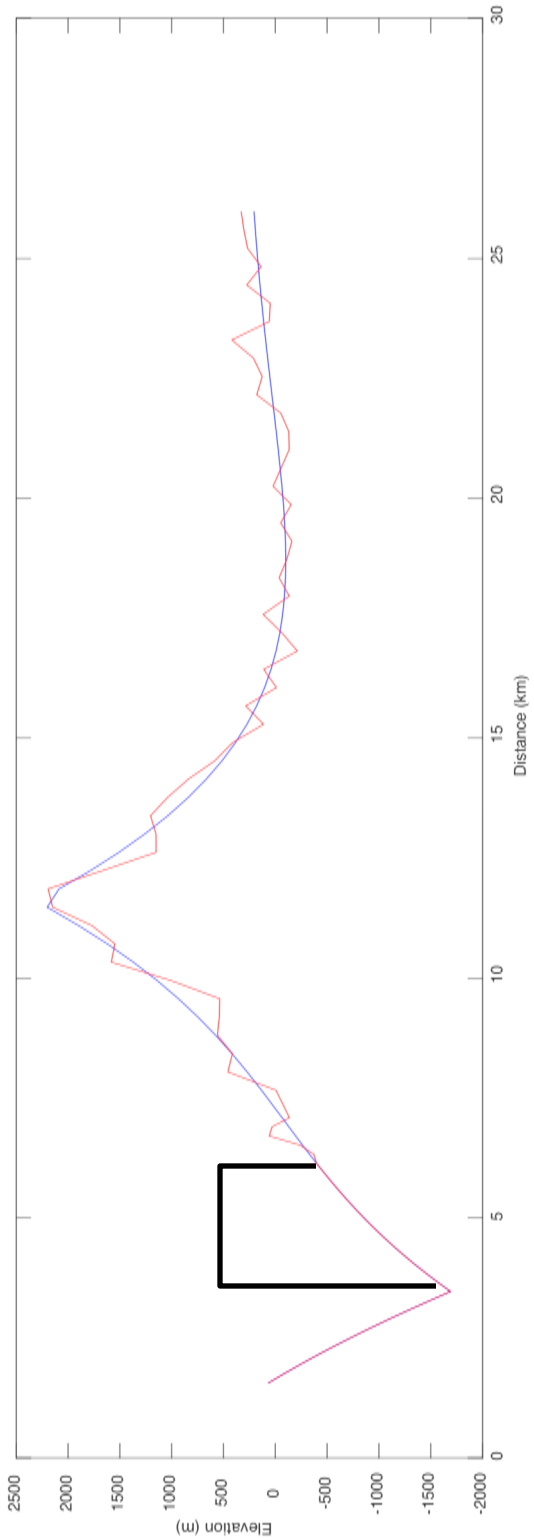


Figure 2.12: Synthetic profile (blue), and same profile with random noise added beyond a distance that I determined from consulting actual OCC profiles. This simulates a faulted and slumped footwall that no longer retains a useful morphology for comparison to flexural models.

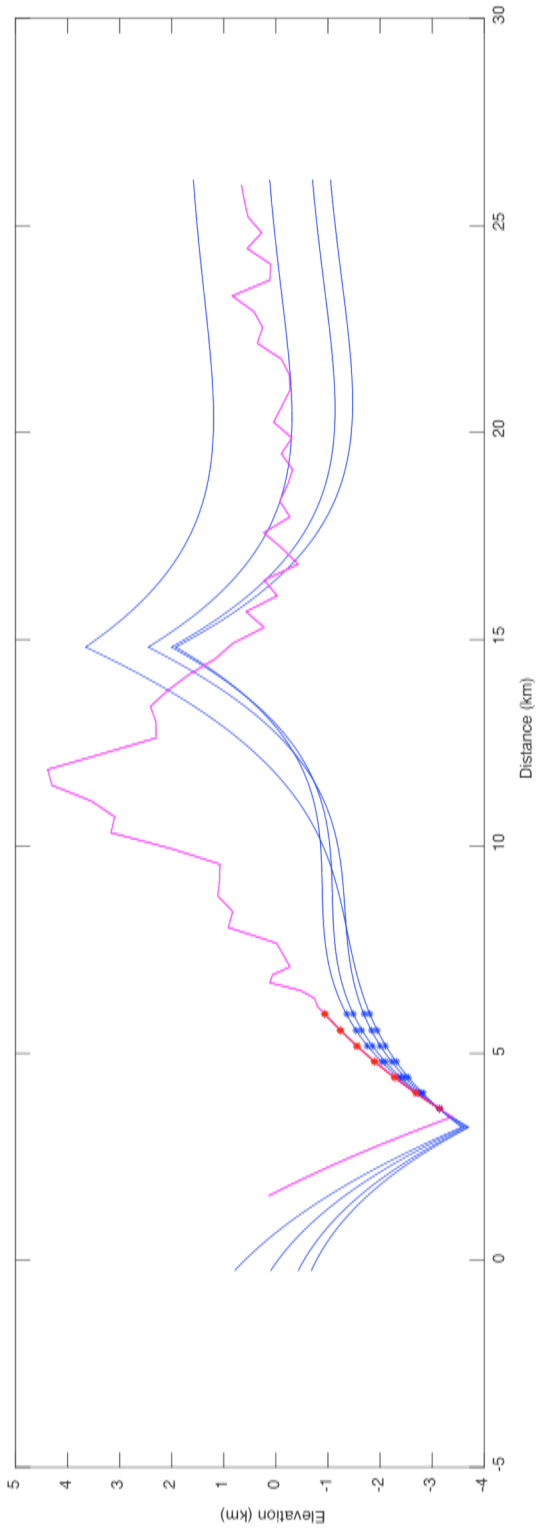


Figure 2.13: Synthetic profile (magenta) compared to each flexural model. Input points for the profile are shown by red dots, blue asterisks are the points on the models which the red dots are compared with. A difference is calculated for each point pair and summed. This sum is the misfit for each profile.

After all models have been compared to the synthetic profile, I must find the best fitting one. I first identify the model that produces the smallest misfit value using bathymetry only or slope only. These two models can be quite different though, and as such, a criterion for an "overall best fitting" model is needed. This model uses a combined measure of misfit that is the weighted average of the misfit in bathymetry and the misfit in slope. Figure 2.14 shows the slope misfit and bathymetry misfit associated with the best-combined fitting model for each value of the weight between 0 and 1. I then choose the model that produces the point nearest to the origin as the best combined misfit model (Figure 2.14).

The misfit values are always much smaller when considering slope data. This is because of the order of magnitude difference in the slope values and the topography values.

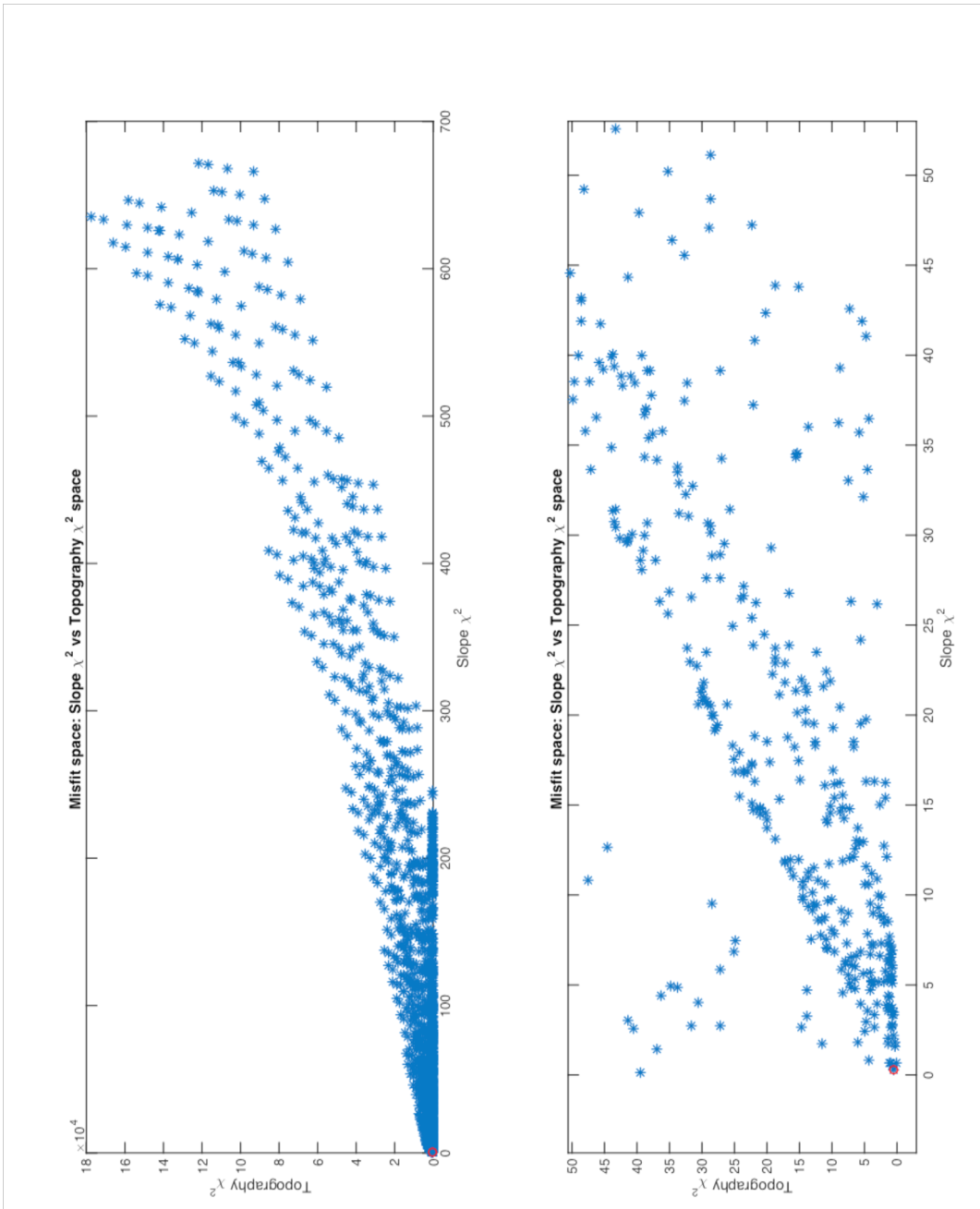


Figure 2.14: Misfit space plot of chi-squares (top), and the same plot, but zoomed in nearer to the origin (second from top) to show the best fitting model (red circle) by distance function determination.

3 Results

3.1 Profiles

This section presents the bathymetric profiles for each of the features considered in this study. For each feature, I show the profile location as well as a perspective view of the entire feature in order to highlight its characteristics. My method of identifying OCCs selected five previously identified features, as well as five new features in the 12º - 15º N mid-Atlantic ridge segment. I only considered features that are near the ridge axis, because active features are needed in order to constrain the properties I am trying to constrain. Older features are more heavily modified, and the cooling of the plate likely affects their curvature.

3.1.1 Candidate OCC in the 12°-15°N segment

The main study area that I consider is the 12°-15°N segment of the Mid-Atlantic ridge. I identify ten candidate OCCs, numbered 1 through 10 from North to South (Figure 1.9). Six are on the West side of the ridge (North American plate) and the other four on the East side (Eurasian plate). Features 4, 5, 7, 8, and 9 were previously identified as OCCs, with Feature 4 part of the Logatchev complex, Feature 9 associated with the Ashadze vent field, and Features 5,7, and 8 labeled as OCC1348, OCC1330, OCC1320, respectively, by Mallows & Searle (2012).

Feature 1 is not a well-formed OCC by many standards. It lacks a feature-wide U-shaped termination, and may be behind another much smaller OCC, which is very close to the axis and just north of the profile for feature 1 (Figure 3.1). However, corrugations can be seen in the region that the profile is taken (Figure 3.1a). The

breakaway slopes ~25º on average. The axis takes a quick jog in front of feature 1, which may explain why the feature's breakaway appears somewhat rotated compared to the spreading direction and represents an uncertainty when determining the fault rooting depth. My profile runs between a region of small corrugations, and a region that seems to lack them. Many places were good candidates for a profile as little large-scale deformation is apparent; my ultimate choice for profile was to take one that went through the middle of the feature.

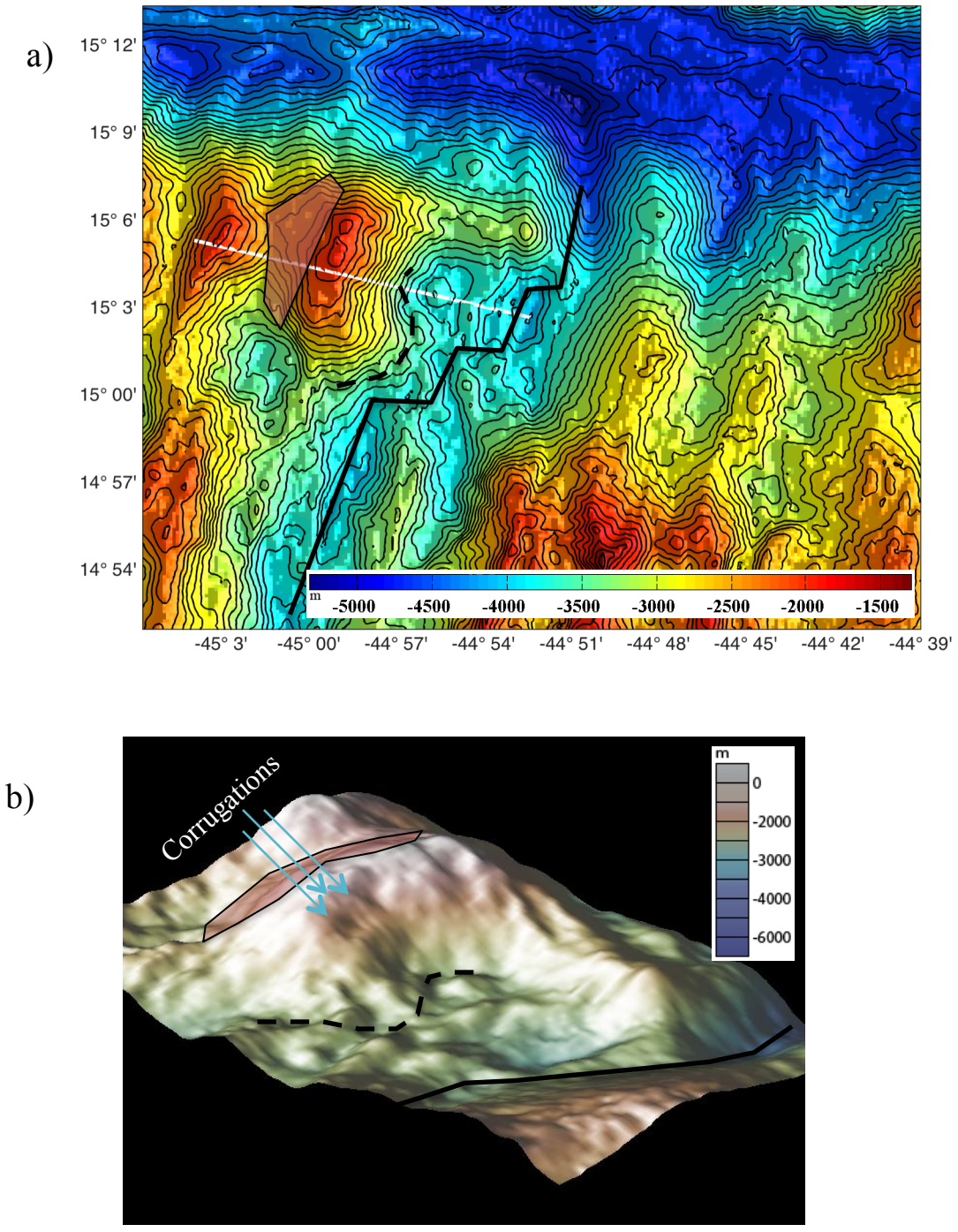
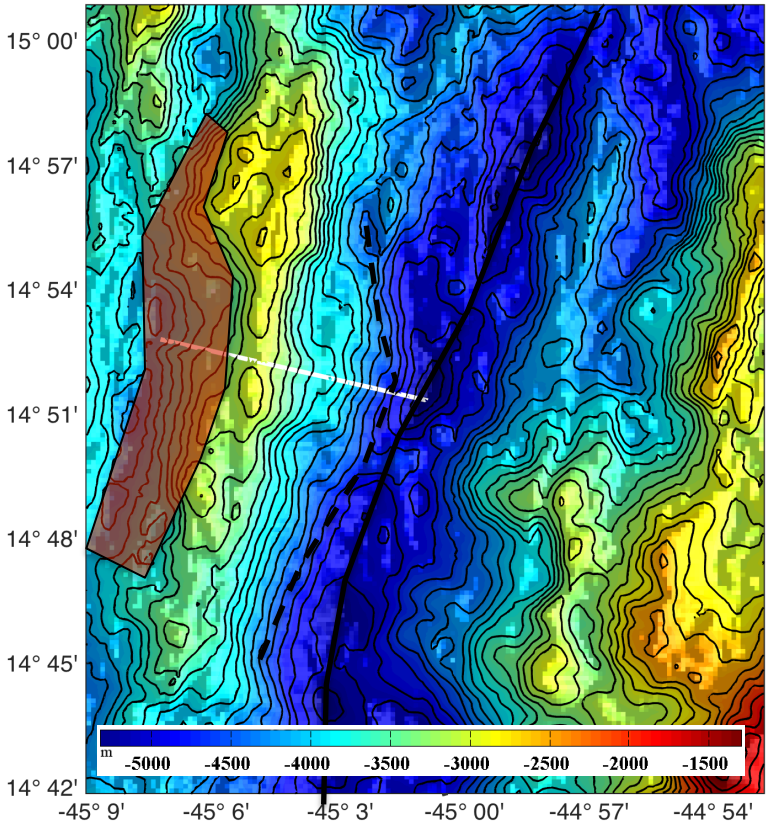


Figure 3.1: Map (a) and perspective (b) views of feature 1 with breakaway (red shaded region), termination (dashed line), axis (black line), and profile location (white line).

Feature 2 has an obvious, although broader than usual, U-shaped termination (Figure 3.2), which can best be seen in the perspective view of Figure 3.2. I place my profile very near to the center of the most convex region of the feature, near where the termination is closest to the ridge axis. The profile goes over what I interpret to be a corrugation, with narrow troughs on either side of it. Large slump features exist ~2km south and ~2.5 km north of my profile. The breakaway that I chose for this feature is the red shadowed region in both Figure 3.2a & Figure 3.2b. A linear ridge and trough run through much of the structure between the identified breakaway and the axis, which is probably a post formation fault, but is possibly a younger breakaway.

a)



b)

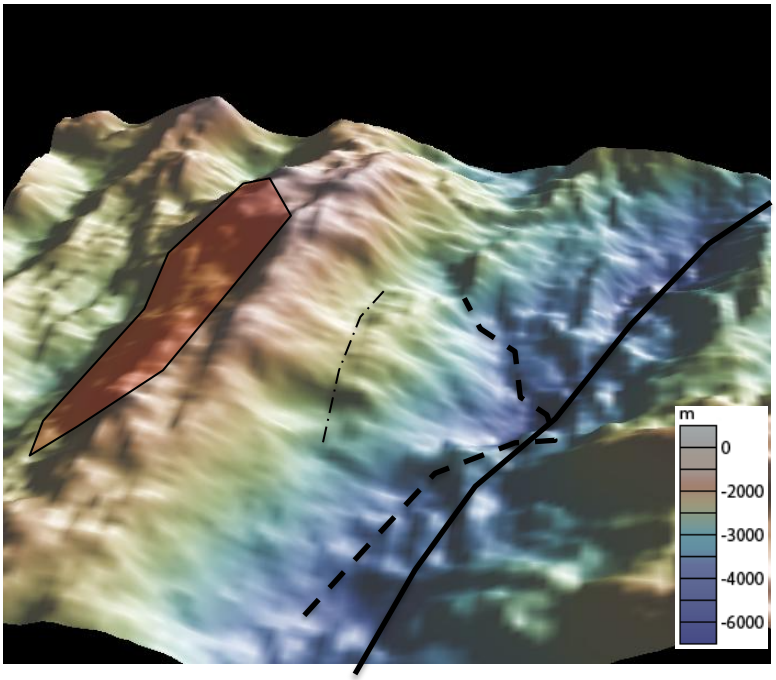
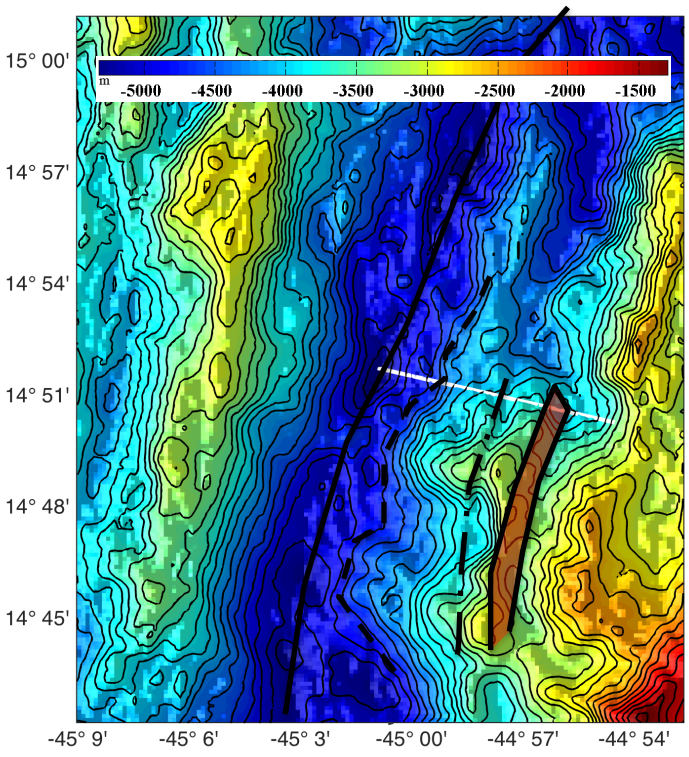


Figure 3.2: Map (a) and perspective (b) views of feature 2, with breakaway (red shaded region), termination (dashed line), axis (black line), and profile location (white line), and possible breakaway or fault (dash dot).

Feature 3 is located directly across the ridge axis from Feature 2. Its profile has all the characteristics of a well-formed OCC. In map view, the feature is associated with possibly two other OCCs, which I did not model in this project (Figure 3.3). Feature 3, Feature 4, and the associated possible OCCs are all within the Logachev hydrothermal vent field. Unlike Feature 4 however, Feature 3 appears to be largely unaffected by hydrothermal and gravity-driven deformation. As in many of the features, a trough cuts through the middle of Feature 3 and could be a fault or a younger breakaway. My profile runs through the northern edge of the feature. This feature does not appear to have undergone much slumping although a smooth draping of sediment has obscured the termination in most places.

a)



b)

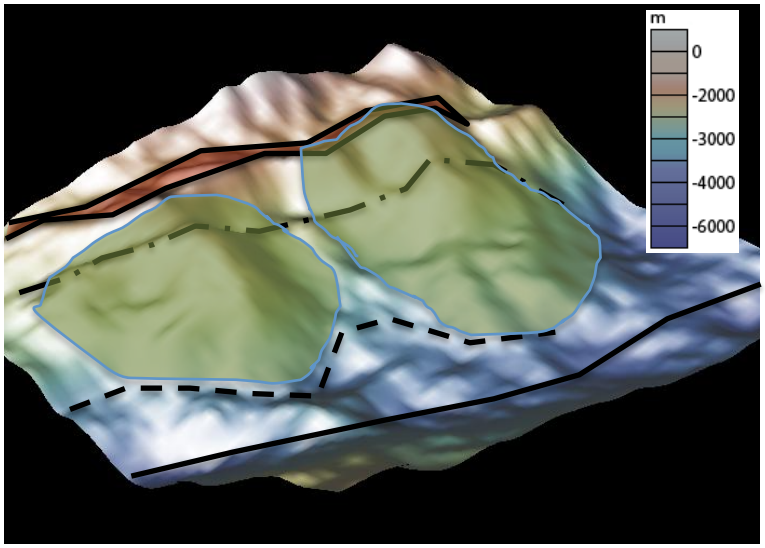


Figure 3.3: Map (a) and perspective (b) views of feature 3, with breakaway (red shaded region), termination (dashed line), axis (black line), profile location (white line), fault feature (dash dot line), and two possible OCCs associated with the breakaway (green regions).

Like Feature 3, Feature 4 is associated with the Logachev vent field. In Figure 3.4, it can be seen that the majority of the profile extends beyond the breakaway of Feature 3. However, because of the ridge axis undergoes a left-lateral offset between the two features, Feature 4 remains on-axis. Even though slumping and other deformation took place on Feature 4, corrugations can easily be seen in Figure 3.4. My profile samples the most convex portion of the surface, and also goes over the smoothest part of the convex region. The edge of what is likely the breakaway of feature 3 lies in front of the termination of Feature 4.

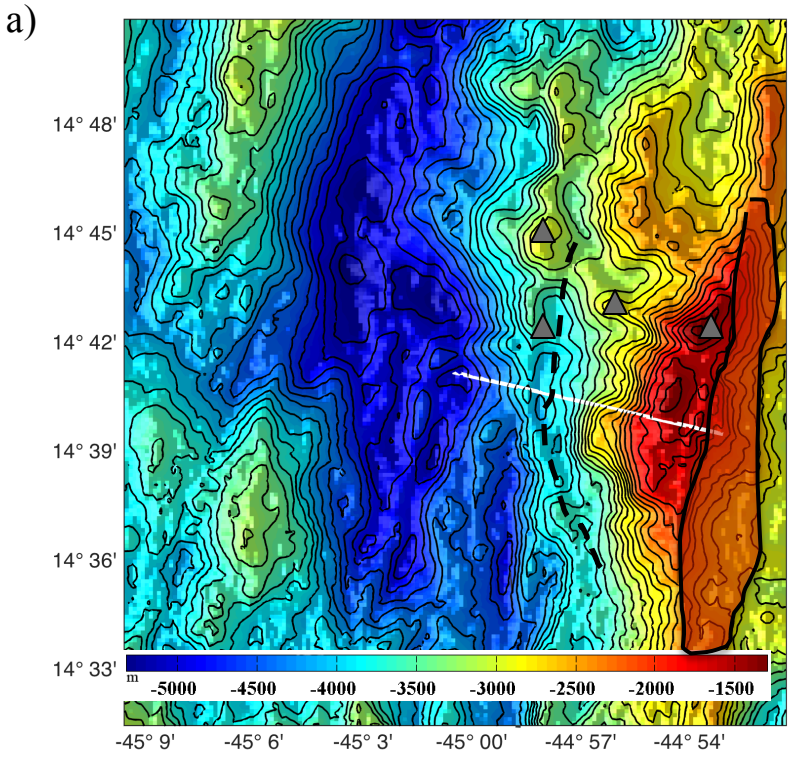


Figure 3.4: Map (a) and perspective (b) views of feature 4, showing termination (dashed line), breakaway (red shaded region), corrugations (teal arrows), vent fields (grey triangles), and profile loation (white line).

Feature 5 has a broad U-shaped termination, with a breakaway that spans a longer distance (Figure 3.5). A sharp inflection connects the breakaway ridge and the termination on the southern portion of the OCC. This is reminiscent of the structure of Feature 1 and may have been created by massive slumping. Alternatively, the feature may have originated as two different detachments that later merged. Less slip took place on the southern half of the surface. Two faults cut through Feature 5 (Figure 3.5). and made it difficult to interpret models of the structure. My profile lies just to the north of the apex of the most prominent and largest corrugation. I avoid the very center of the corrugation because of the missing data (white region in Figure 3.5) and because a profile drawn over that region has a few mini troughs that are caused by small slumping or faulting events. The portion of the profile nearest to the ridge axis is chosen to be modeled, but of note is the observation that little slumping has occurred on the rest of this particular profile.

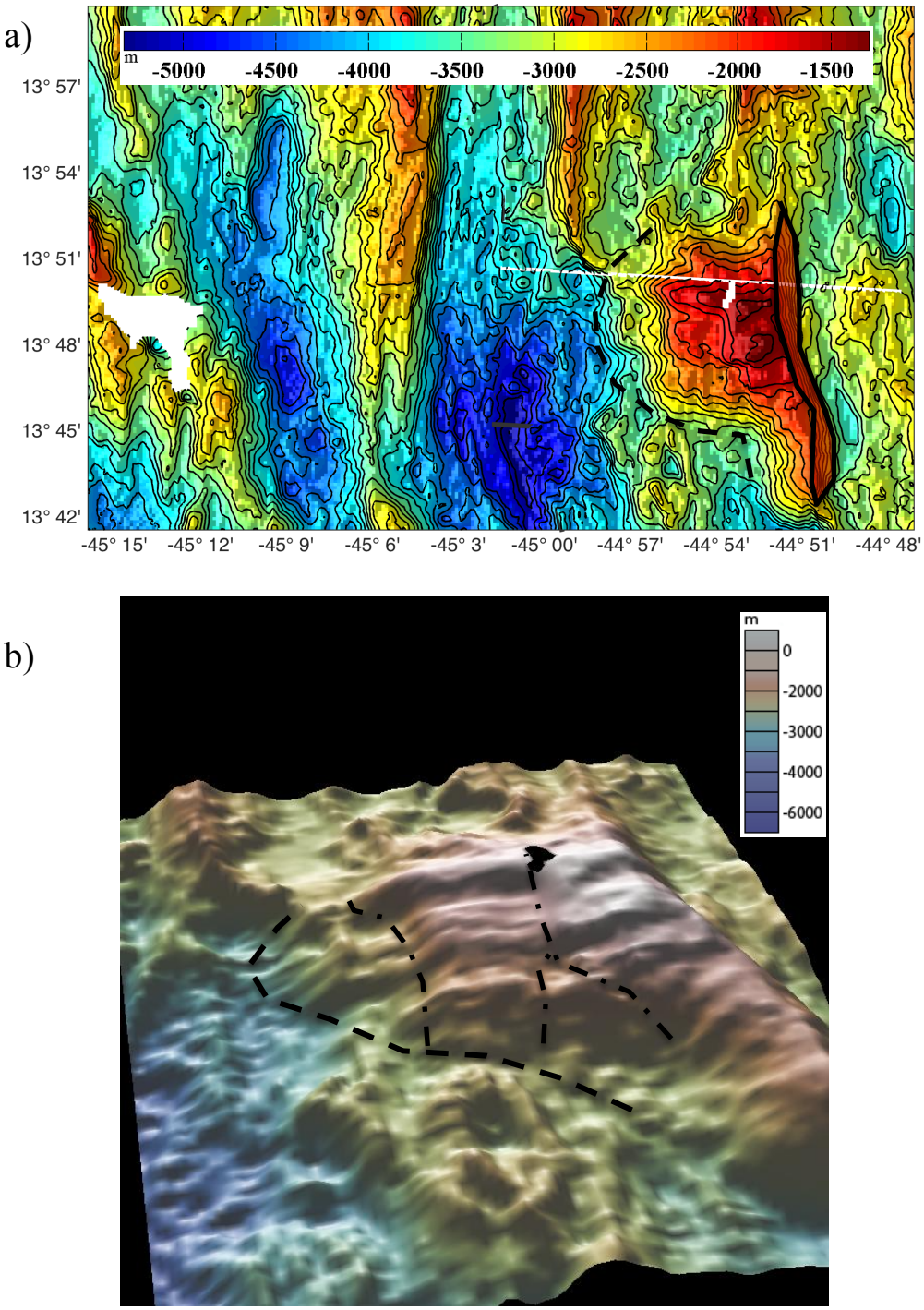


Figure 3.5: Map (a) and perspective (b) views of feature 5 showing termination (dashed line), breakaway (red shaded region), post formation faults (dash-dotted lines), and profile (white line).

Feature 6 is the smallest OCC I identified in the study area. Its termination almost intersects the ridge axis, and has a measured heave of ~1.5 km (Figure 3.6). The profile I draw for this feature goes directly through the center. For such a small and young feature no other options exist. I identified a larger OCC ~5 km further off axis, whose breakaway is highlighted in both Figure 3.6 a&b as the larger red shaded region. Both candidate OCCs are small features, although their breakaways dip at a typical ~25º and they display characteristic U-shaped terminations. This morphology is different from that of a seamount and makes them distinct enough even though they lack corrugations. For comparison, a seamount lies directly on the ridge axis in Figure 3.6 a&b.

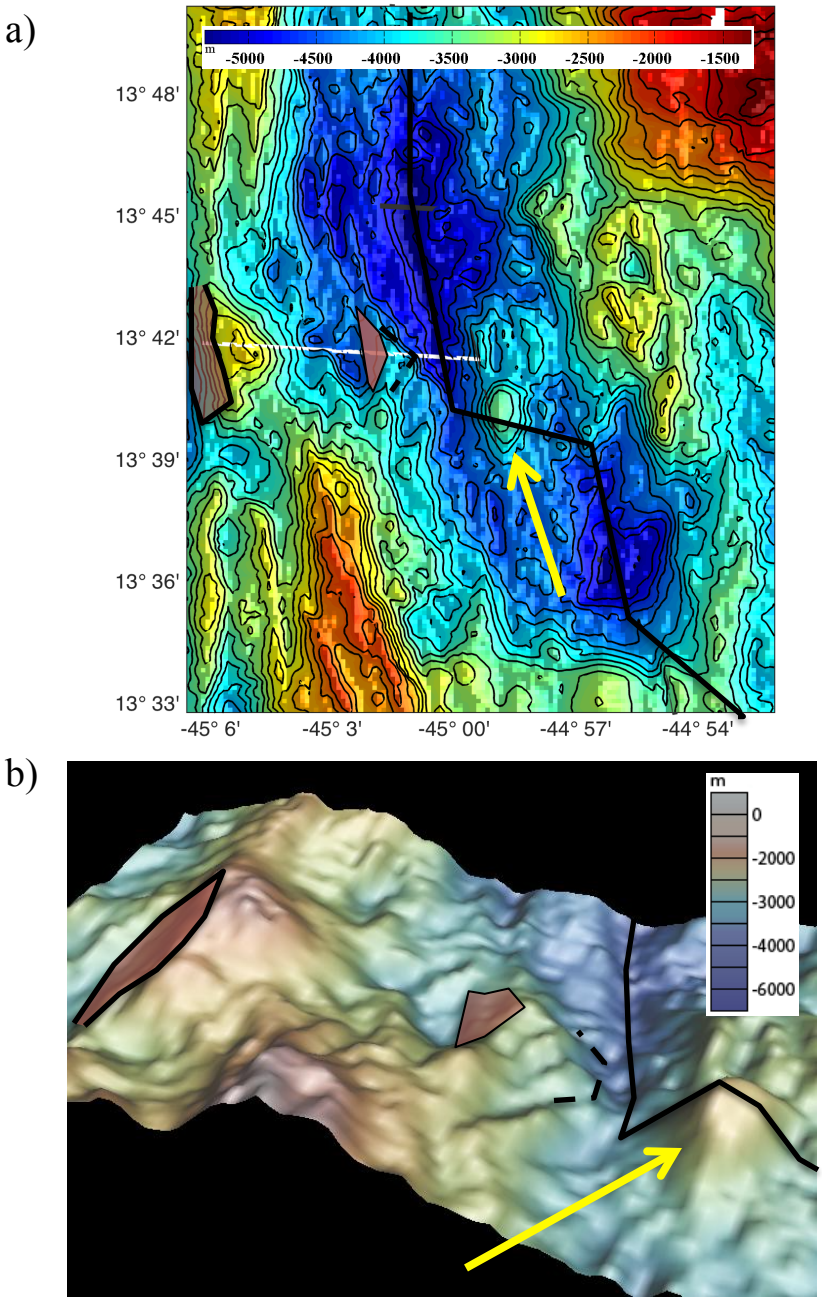


Figure 3.6: Map (a) and perspective (b) views of feature 6 showing axis (solid line), termination (dashed line), breakaways (red shaded regions), seamount (yellow arrow), and profile (white line).

Feature 7 has been previously identified as an OCC by Smith et al. (2008) and Mallows & Searle (2012). My profile follows a smooth surface just to the north of a small trough and avoids the prominent central corrugation because of faulting and slumping that is mostly confined to the southern portion of the OCC (Figure 3.7). There is uncertainty as to the location of its breakaway, which may occur at either of the two locations I identified in (Figure 3.7 b). Inversions using heave values of both of these breakaways produce models that have small misfits, however the breakaway nearer to the axis fits within a more realistic structure overall. The breakaway that lies further from the axis is highly irregular, and disrupted, and may have been moved away from the axis as a relatively cohesive unit. That would explain why the breakaway nearest the axis only dips ~15º, which is rather shallow. About three corrugations can be seen over the central portion of this OCC. They are much wider than the corrugations seen on Feature 8 (Figure 3.8) but narrower than those seen on Feature 5 (Figure 3.5).

The location of the ridge axis around this OCC is quite uncertain, with at least three possibilities proposed in the literature (Smith et al., 2008; Fujiwara et al., 2002; Mallows & Searle, 2012). I use the most recent and detailed from Mallows & Searle, (2012), which places Feature 7 very close to the ridge axis, which is key to their model of OCC life cycle. Fujiwara et al. (2002) and Smith et al. (2008) locate the axis ~2 km further away, which significantly affects the depth of my model fault root but not the misfit values.

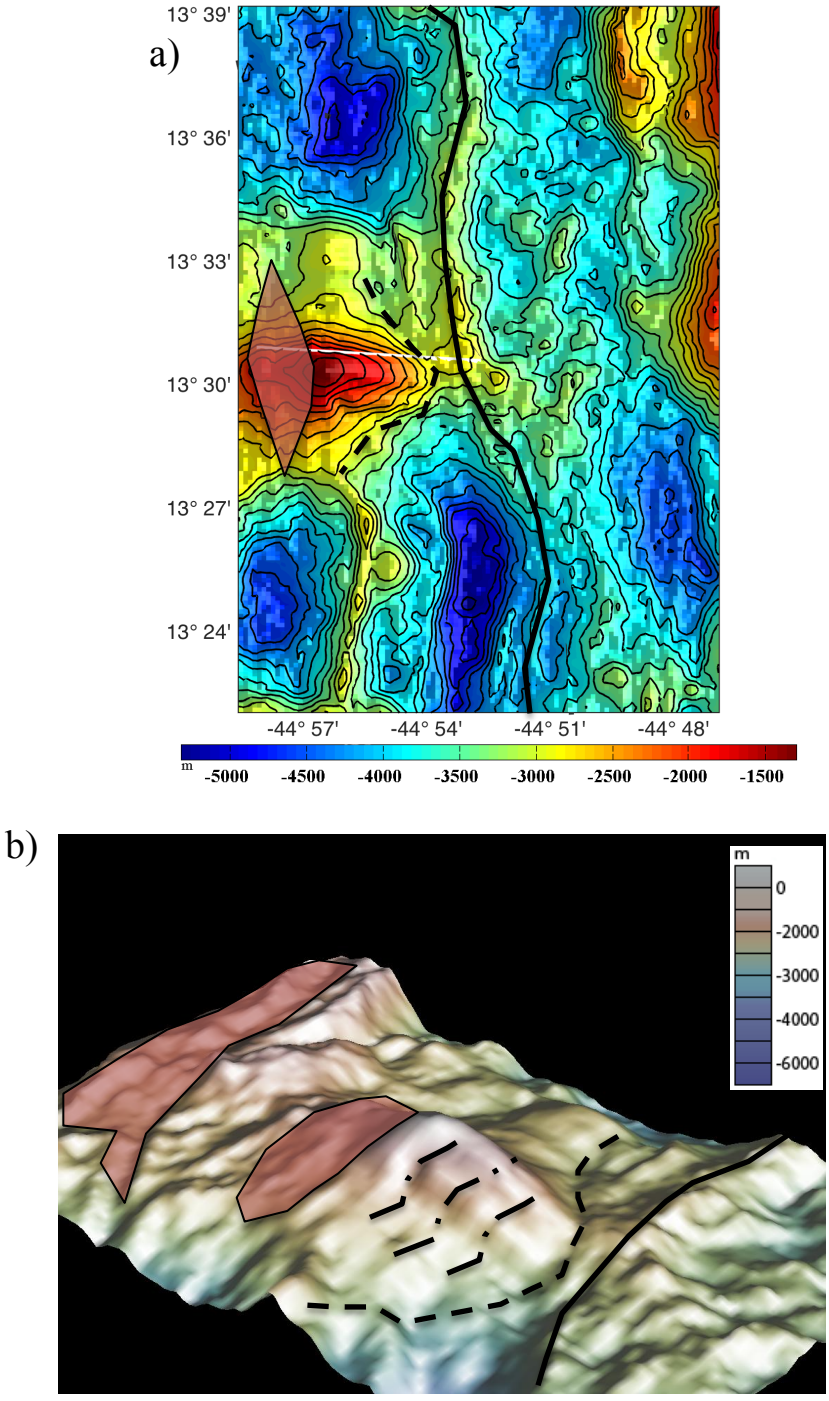


Figure 3.7: Map (a) and perspective (b) views of feature 7 showing axis (solid line), termination (dashed line), faults (dashed –dotted lines), breakaway (red shaded region), and profile location (white line).

Feature 8 has an extremely pristine appearance and well-developed corrugations over much of its exhumed surface (Figure 3.8). The upper portion can be described as rubbly and could be a large set of volcanic rider blocks or slumps. According to elastic flexure theory, a series of volcanic rider blocks would cause the OCC to sink in the region containing the rider blocks, causing buoyancy-driven uplift in the region nearer to the axis. Translating the breakaway as a slump block would likewise result in a change in the features overall morphology. None of these modifications is included in the elastic model that I use for this project. Instead, a greater uncertainty should be considered when comparing the profile to models without rider blocks and slumps.

My profile goes over the southern edge of the prominent central bulge of this OCC. I choose this profile as opposed to going over the center of the feature because it avoids the large sediment/slump pile that is concentrated near the northern central portion of the termination (purple shaded region, Figure 3.8).

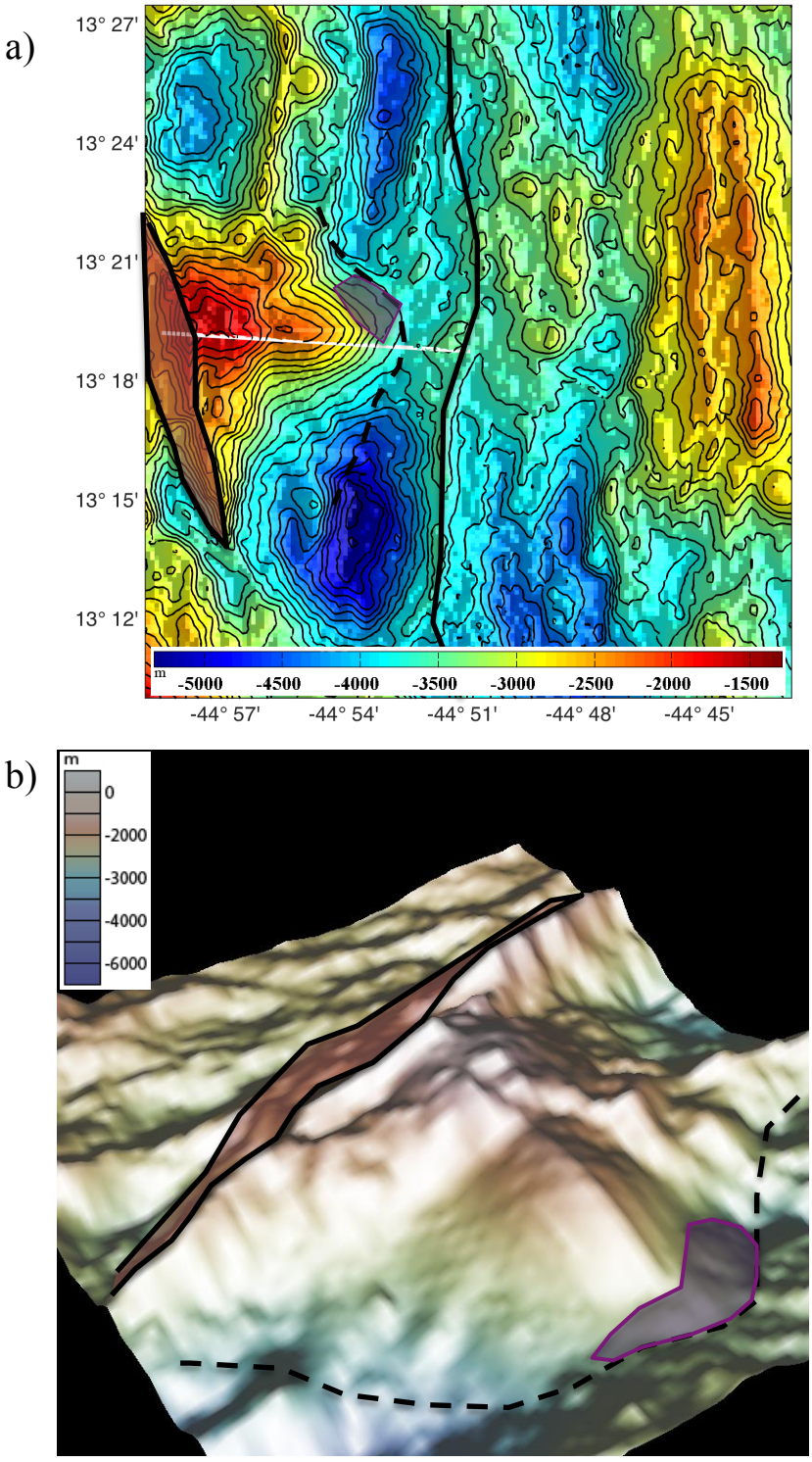


Figure 3.8: Map (a) and perspective (b) views of feature 8 showing axis (solid line), termination (dashed line), breakaway (red shaded region), profile location (white line), and slumps (purple shaded region).

The Ashadze vent field and associated OCC is collocated with Feature 9. The Ashadze region is shown in Figure 3.9: Map view of feature 9 showing perimeter of Ashadze region (orange dash), axis (solid line), termination (dashed line), breakaways (red shaded region), vent fields (grey triangles), and profile location (white line). and is the most complex region of the ridge segment. As mentioned in 1.3.2 this region has undergone extensive deformation and surface alteration. The profile I picked is associated with a prominent relatively large-scale convex structure that contains two corrugations. A concave structure to the North of the profile and many hummocks to the South of the profile are likely slumped blocks. Thus my profile likely samples the only pristine region of the entire exhumed surface. Whether this feature is a seperate OCC or merely the youngest portion of a long-lived detachment is unclear. There exists near Feature 9 a clear breakaway ~20 km from the axis, a region containing corrugations, and a sharp U-shaped ridge from which many of the slump blocks originated and which itself may be another breakaway (Figure 3.9: Map view of feature 9 showing perimeter of Ashadze region (orange dash), axis (solid line), termination (dashed line), breakaways (red shaded region), vent fields (grey triangles), and profile location (white line).; Ondreas et al., 2012). Fixed heave inversions assume that the detachment stops at the breakaway nearest to the ridge axis although free-heave inversions favor a heave that would put the breakaway ~20 km off-axis.

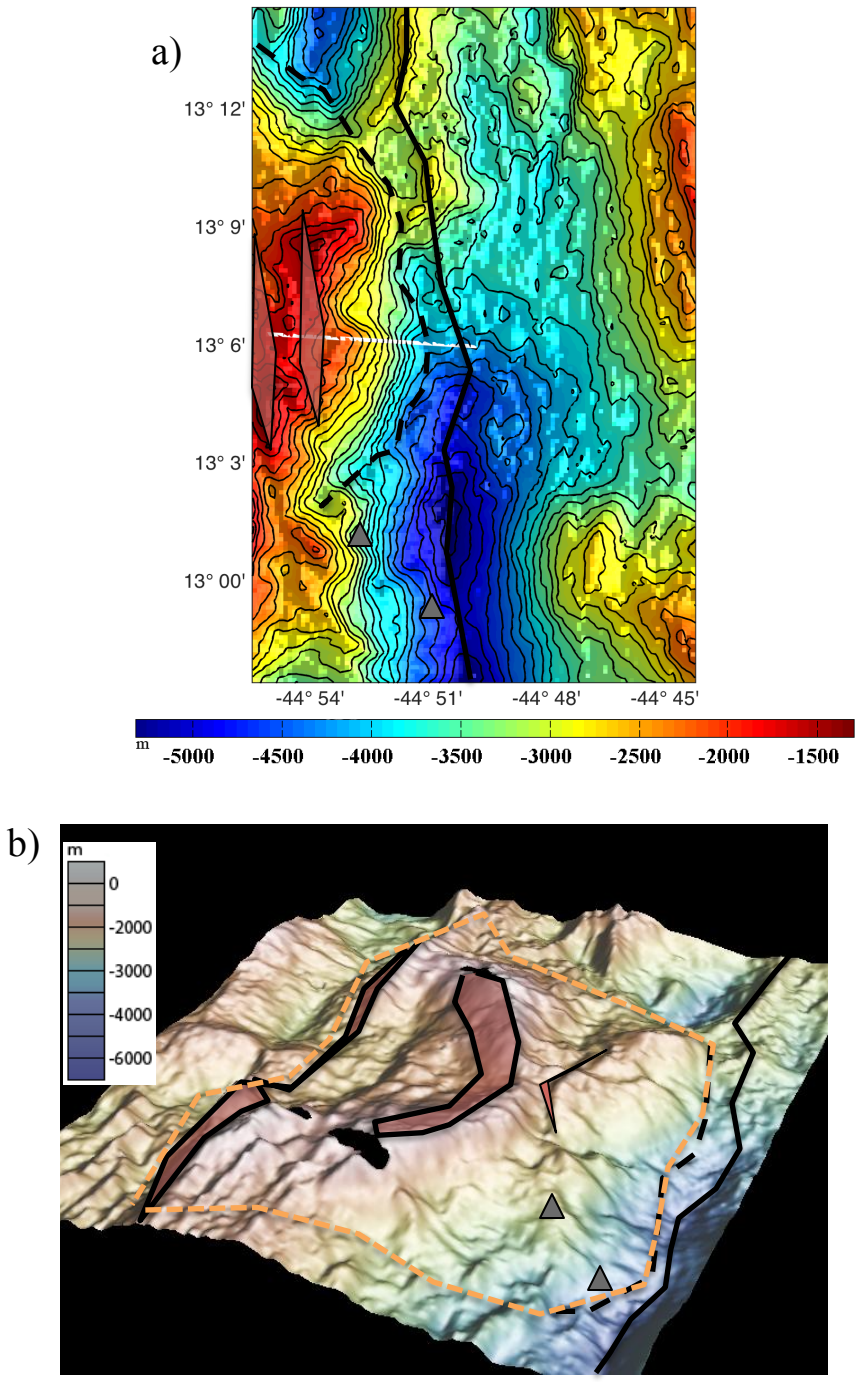


Figure 3.9: Map view of feature 9 showing perimeter of Ashadze region (orange dash), axis (solid line), termination (dashed line), breakaways (red shaded region), vent fields (grey triangles), and profile location (white line).

Many of the most famous OCCs occur at inside corners of segments along the Mid-Atlantic ridge (Cannat et al., 1997; Dick et al., 1981; Tucholke et al., 1997). Feature 10 lies at the southern portion of my study area at the segment's inside corner. It does not have a single clear breakaway. Instead, the region beyond its breakaway ridge is composed of many linear ridges that could simply be created by successive normal faults, as seen at the center of the segment at ~14º N (Figure 1.8). Regardless of what those features are, the region axis-ward of the breakaway ridge of my feature is devoid of any similar linear ridges. Instead it hosts a few sediment chutes that lie between what may be corrugations (Figure 3.10: Map (a) and perspective (b) views of feature 10 showing axis (solid line), termination (dashed line), faults (dashed –dotted lines), and breakaways (red shaded regions).). There is one possible fault that cuts Feature 10's surface. Otherwise the surface appears draped in sediment. The profile I selected for Feature 10 avoids several slump locations identified on the surface as well as the central part of the feature. Instead, the profile is drawn over a large corrugation that lies on the northern portion of the feature (Figure 3.10).

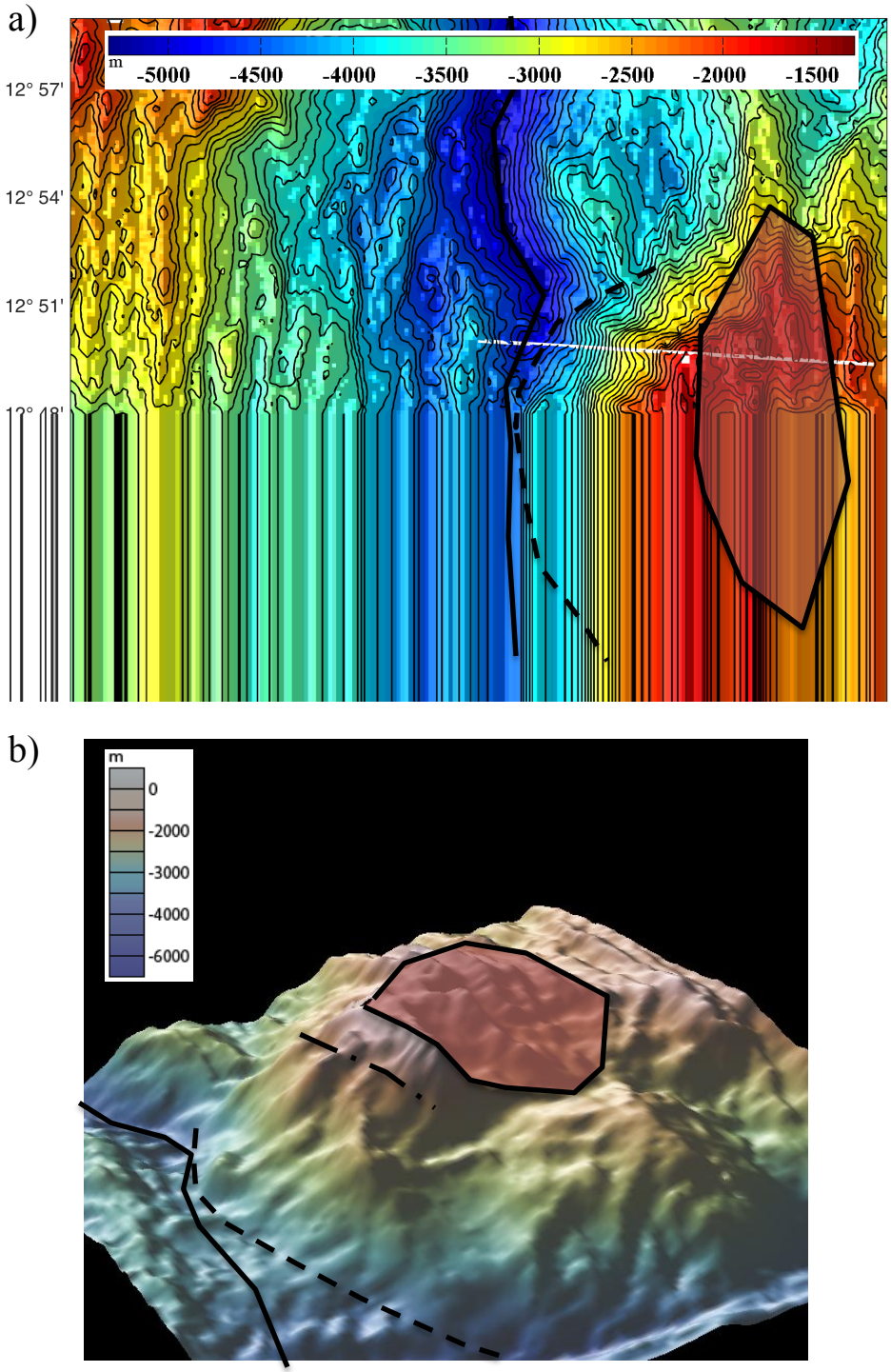


Figure 3.10: Map (a) and perspective (b) views of feature 10 showing axis (solid line), termination (dashed line), faults (dashed –dotted lines), and breakaways (red shaded regions).

3.1.2 Prototypical OCCs

In addition to the ten Features identified as candidate OCCs in the 12°-15°N segment, I consider three well-documented OCCs to evaluate the success of the elastic model in capturing the shape of a long-lived detachment. The three prototypical OCCs I selected are Dante's Dome, the TAG core complex, and KMM (Tucholke et al., 1998, 2001; Escartín, 2008; MacLeod et al., 2009). All three are located along the Mid-Atlantic Ridge, although well to the North of the study area (Figure 3.11). Dante's Dome is very near TAG core complex. They were both chosen because of their well-documented characteristics, as well as proximity to one another. KMM is located on the inside corner of the north end of the segment directly south of the transform fault separating the first order segment containing TAG core complex, and Dante's Dome.

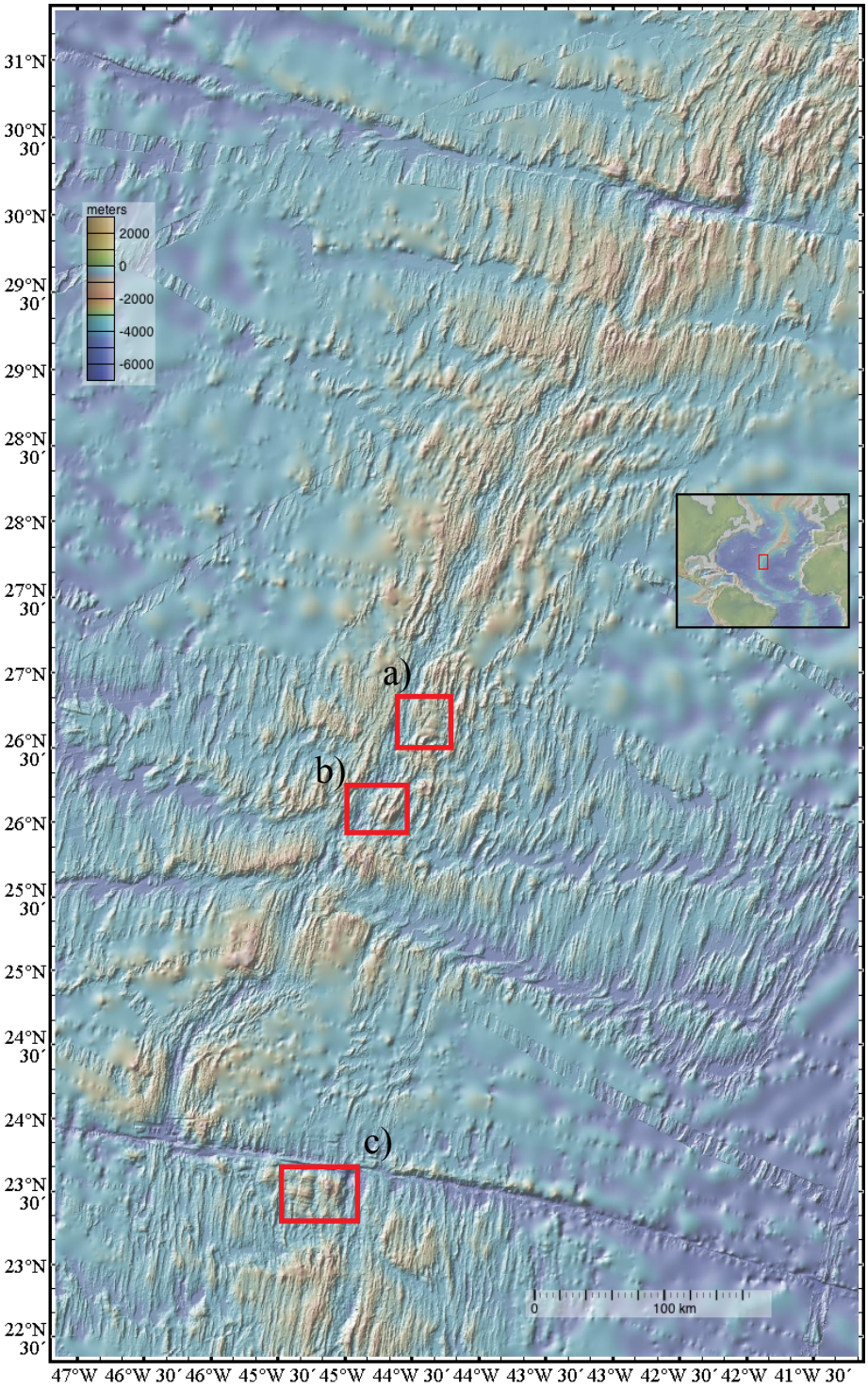


Figure 3.11: Bathymetric map of the TAG segment, showing locations of Dante's Dome (a), TAG core complex (b), and Kane Megamullion (c).

The topographic profile for Dante's Dome is taken over the southern part of the OCC (Figure 3.12). The profile goes over corrugations for about 20 km. However, most of that surface is interpreted as being currently inactive (Tucholke et al., 2001). A newer breakaway can be identified about 6 km into the topographic profile (smaller red shaded region, Figure 3.12). Therefore, the model is fit only to the detachment ridge-ward of that younger breakaway with a heave of 4 km. A much larger ridge lies to the North and East of my profile, and doesn't appear to have the same corrugations associated with it, Dante's Dome region may be replaced entirely by non-OCC style spreading regime at this point in time.

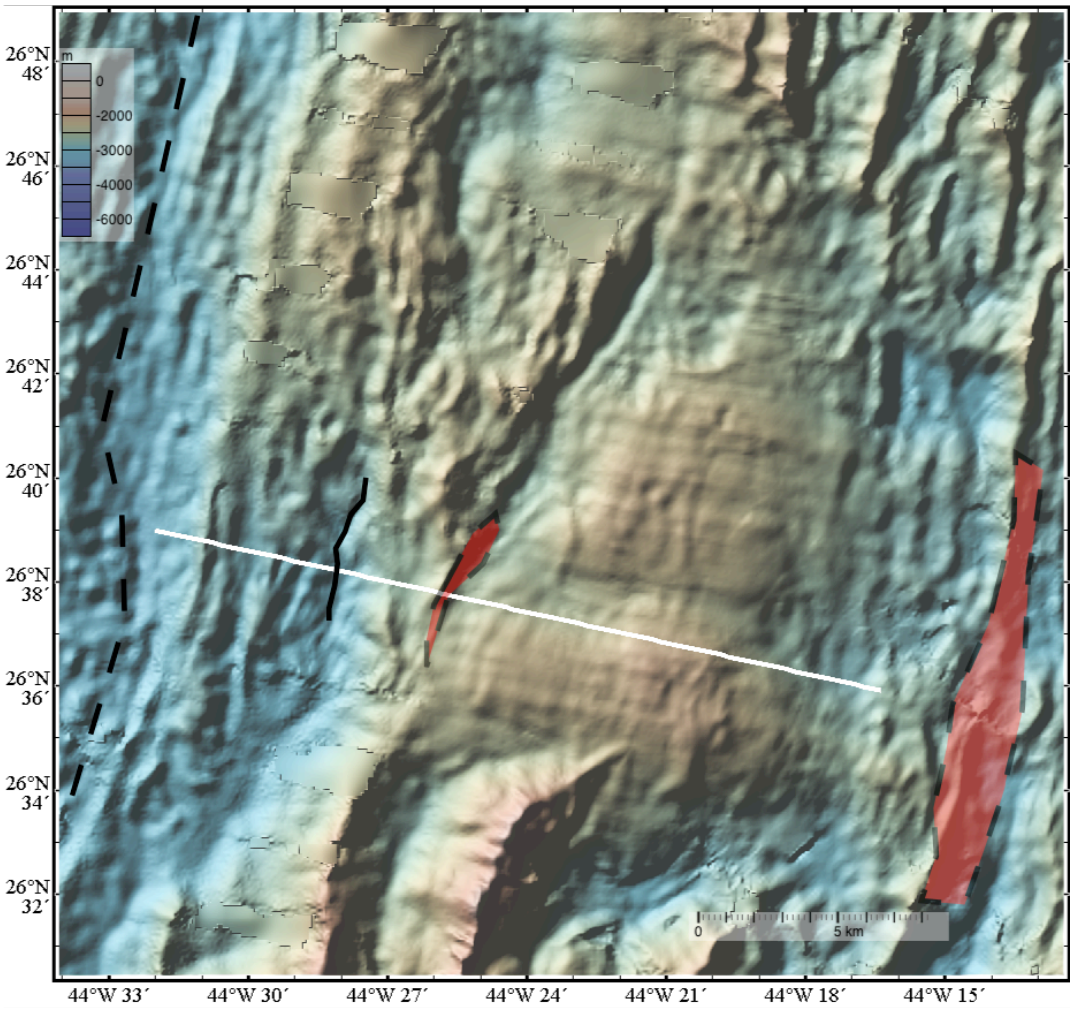


Figure 3.12: Map view of Dante's Dome showing axis (dashed black line), termination (solid black line) profile location (white line), and breakaways (shaded red regions).

The TAG core complex is located on-axis in the center of the ridge segment. The TAG core complex appears to have up to four separate breakaways (Figure 3.13). They appear to be spaced about every 5 km. I choose the 10 km breakaway as the furthest extent of the feature because in profile it has a characteristic surface. Beyond 10 km the feature appears to be covered in sediment chutes and rubble, suggesting extensive post-formation deformation beyond my choice of breakaways. I draw my profile over the place where the breakaway's U-shapes are nearest to the axis. My profile captures two possible breakaways that can be seen in Figure 3.13 as the red shaded regions. When doing fixed-heave inversions I use the breakaway nearer to the ridge axis, because the surface looks undisturbed, however, free-heave inversions prefer the further breakaway. It is unclear whether the nearer breakaway is then a rider block or a breakaway. Higher resolution bathymetry data could help discriminate between these two scenarios.

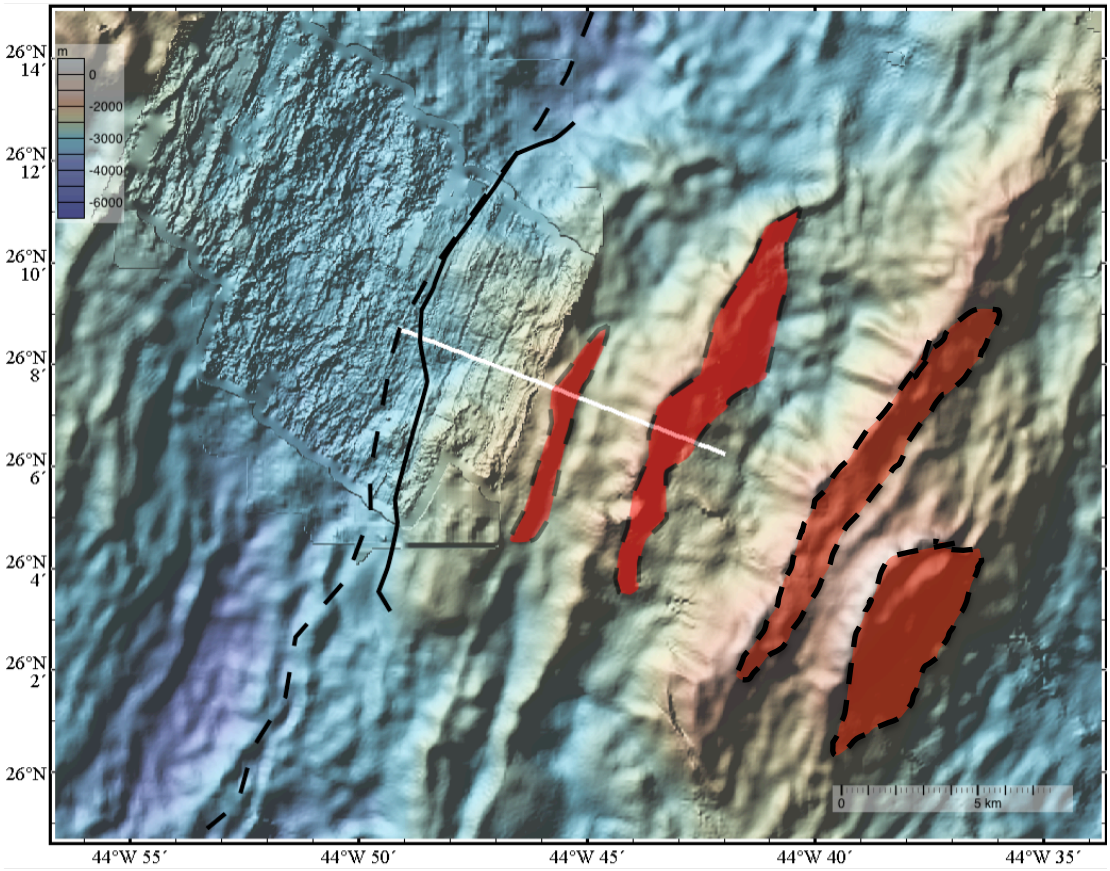


Figure 3.13: Map view of TAG core complex showing axis (dashed black line), termination (solid black line), and profile location (white line), and breakaways (shaded red regions).

The Kane Megamullion is an off-axis core complex (Figure 3.14). It was chosen because it is has been extensively described in the literature and because it lies on an inside corner, next to a transform boundary. Its corrugated surface extends for ~40 km in ridge parallel direction and has a very undulatory termination, and several breakaways that may or may not have been connected in the past. I focus on the central dome, with a distinct U-shaped termination, and a clear breakaway. Beyond my chosen breakaway exists more corrugations, which suggest that my choice of breakaway may be a rider block. Although no clear breakaway exists beyond that corrugated surface. The exhumed footwall is quite pristine across the entire OCC, so I draw my profile through the central portion, along a medium width corrugation.

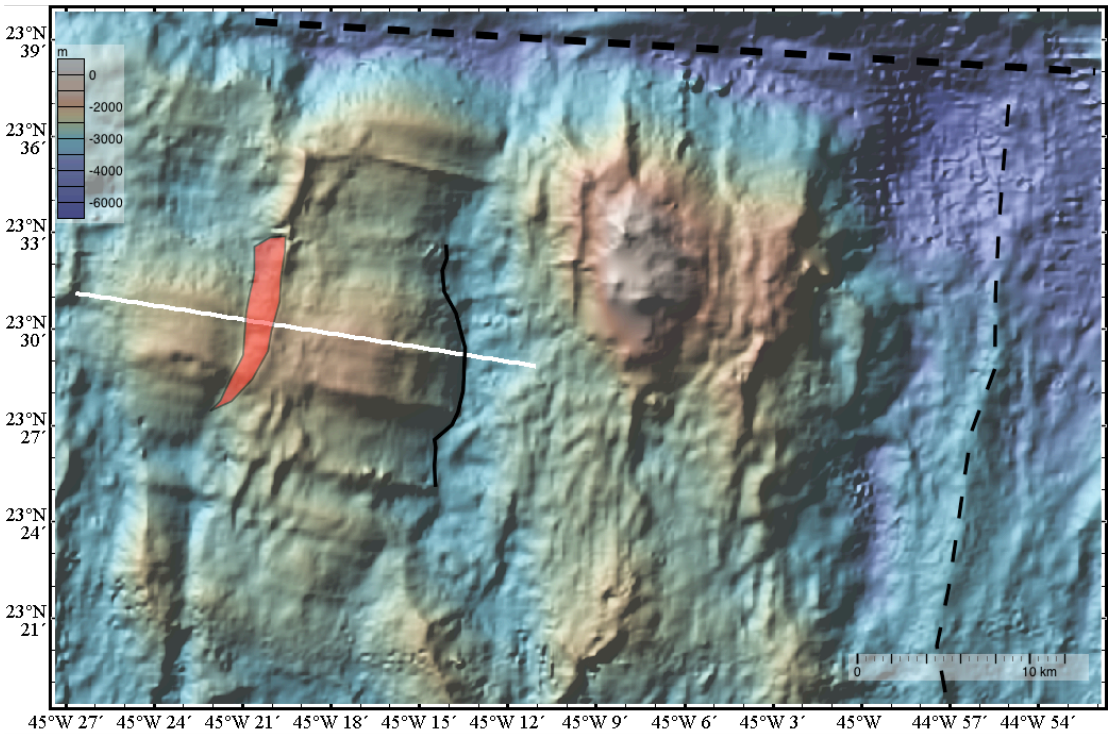


Figure 3.14: Map view of Kane megamullion showing axis (dashed black line) and transform boundary (thick black line), termination (solid black line), profile location (white line), and breakaway (shaded red regions).

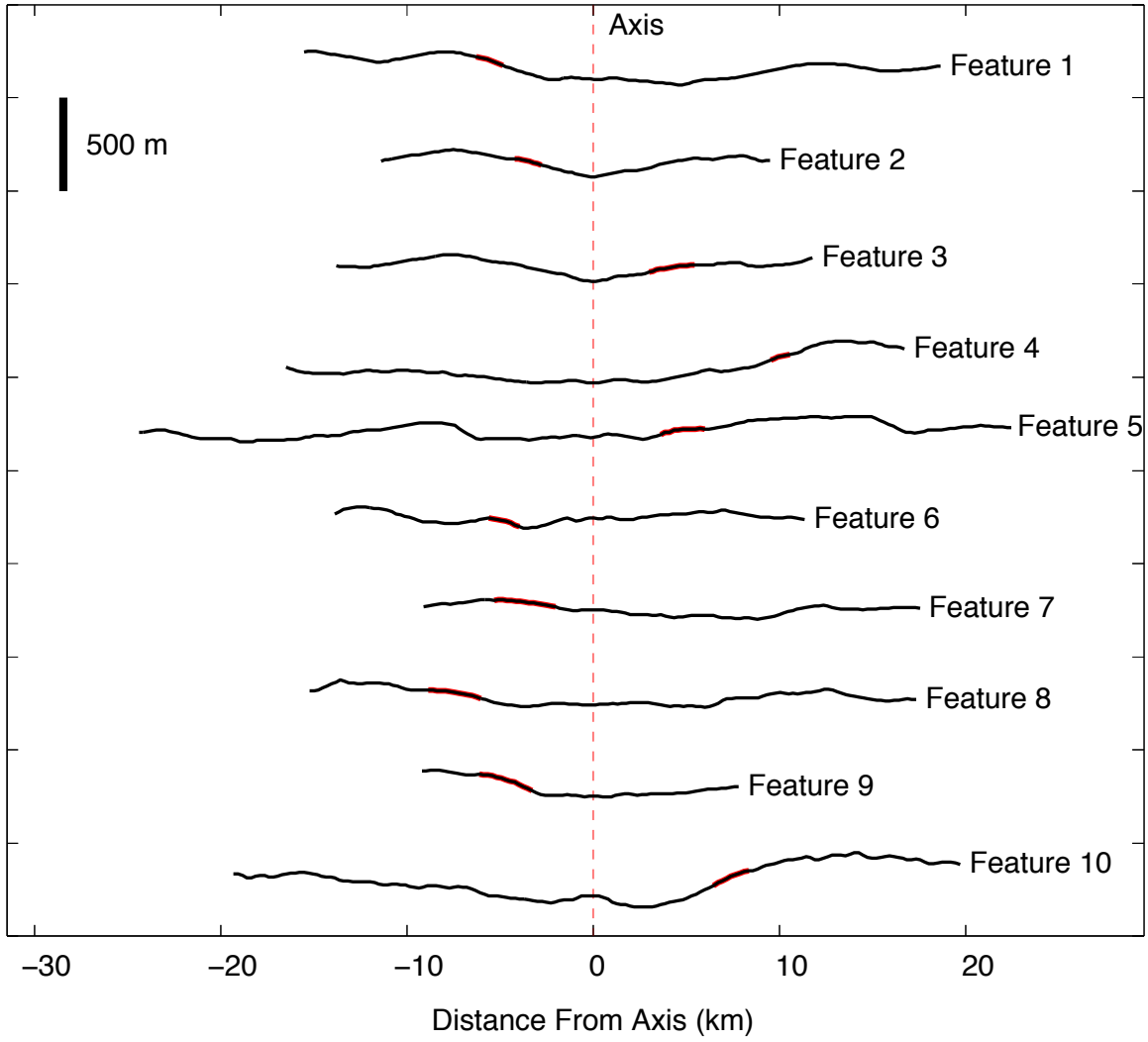


Figure 3.15: Chosen profiles of each feature, with modeled region highlighted in red.

Table 3.1: Details for each feature, and chosen profile.

Feature	Heave (km)	Fitted Length (km)	Number of points	Corrugations Present	Clear Break away	Discovery Reference	Distance to transform (km)
Dante's	3.6	1.64	15	Yes	No	Tucholke et al. (1998)	350
KMM	10.2	5.45	18	Yes	No	Tucholke et al. (1998)	20
TAG	4.4	1.94	8	Yes	No	Zonenshain et al. (1989)	220
1	6	1.43	8	Yes	Yes	This study	16.49
2	7.3	1.43	8	Yes	No	This study	37.99
3	5.5	2.46	13	No	Yes	This study	37.30
4	6.4	1.03	6	Yes	Yes	Fujiwara et al. (2003)	56.49
5	11.6	2.36	11	Yes	No	Smith et al. (2006)	132.87
6	2.0	1.65	8	No	Yes	This study	115.82
7	4.8	3.30	15	Yes	No	Smith et al. (2006)	95.49
8	9.8	2.83	13	Yes	No	Smith et al. (2006)	73.64
9	3.7	2.84	13	Yes	No	Cannat et al. (1995)	50.09
10	8.7	1.89	9	Yes	No	This study	20.50

3.2 Inversion results

The topographic profiles of each OCC were modeled using elastic plate flexure. I compare here the topographic and slope profiles of each best fit model against the observation and report the model parameters associated with each feature. First, I present matches to three prototypical OCCs, Dante's Dome, KMM, and the TAG core complex. Then, I report the results for the ten OCCs proposed in the 12°-15° segment of the MAR.

The elastic plate model depends on elastic plate thickness, T_e, crustal thickness, T_c, axial infill thickness T_{if}, fault heave h, fault angle α, and depth of faulting (rooting depth) z_r. The models were found to be essentially insensitive to T_c and T_{if}. Therefore, these parameters will not be discussed further. In all the models presented here, fault heave was set to the observed value to avoid selecting models associated with a local misfit minimum and reduce parameter spaces. However, in a few cases where the location of the breakaway is ambiguous, we compare these results with inversions in which heave is a free parameter. The mechanical properties of the plate are fixed. The important inversion results are therefore T_e, α, and z_r. Inversion results are compiled in Table 3.2 and Table 3.5. Table 3.2 shows the results for a fixed heave, and Table 3.5 shows the results of full grid search or free heave inversions.

Only the footwall is considered in the misfit measure. The hanging wall is likely modified by ridge axis processes and therefore is not reliably a component of the model whereas the footwall is being carried away from the axis. Surface modification processes such as slumping; riding blocks; volcanism; and secondary

faulting can modify the appearance of the detachment surface. Therefore, I fit the model only to portions of the OCC that are interpreted as relatively pristine on bathymetric maps and profiles.

3.2.1 Prototypical OCCs

The preferred elastic thickness for Dante's Dome is 100 to 200 m depending on whether topography or slope is fit (Figure 3.16). The fault dips 50° to 55° and is rooted at 3.0 to 3.5 km depth. The models over-predict the topography near the breakaway, possibly because of slumping. The fit is conducted only over the region of the OCC that is interpreted as pristine based on map-view and profile bathymetry.

Because of the presence of an older, currently inactive detachment, I also model the profile without a constraint on the heave. In that case (Figure 3.16c&d) the preferred heave is 12.5 to 15.5 km, with an elastic thickness of 400 to 500 m, and a fault dip of 70º. The elastic thicknesses of the free heave inversions are higher than when the heave is fixed to 4km and the angle becomes more variable (see Table 3.5).

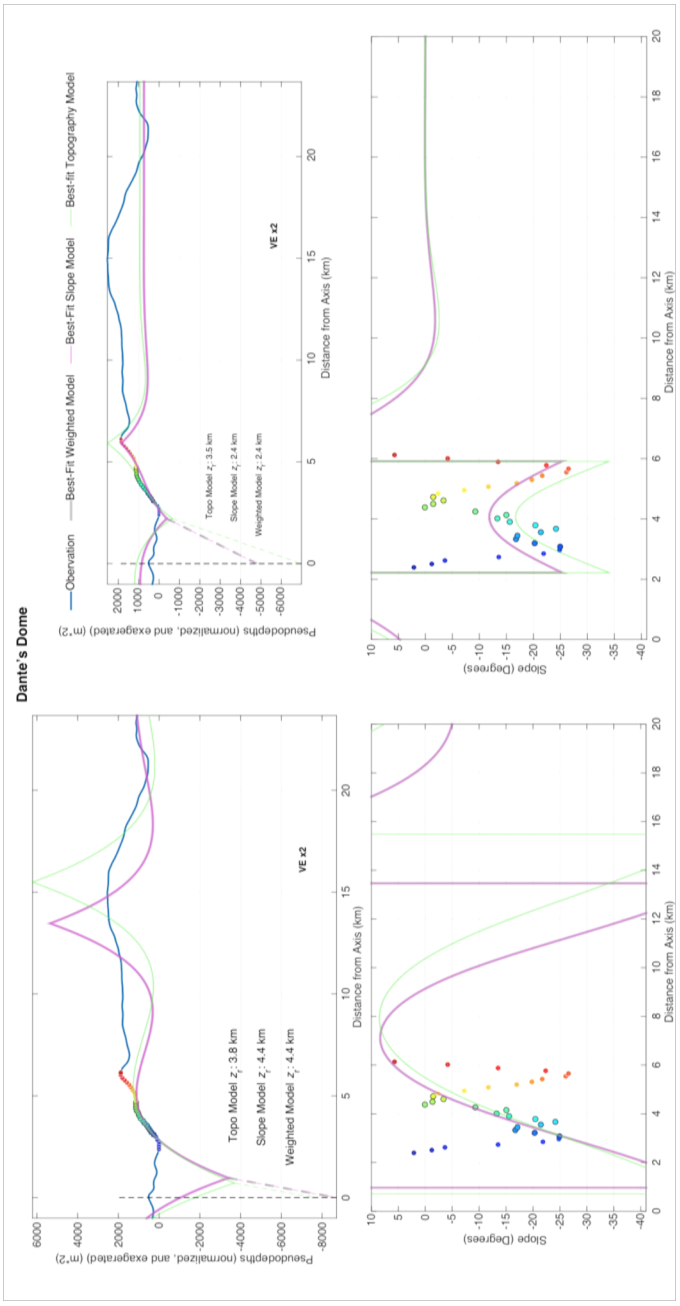


Figure 3.16: Bathymetric (a, b) and slope (b, c) profiles of Dante's Dome (thick blue line). The best fit models from fixed-heave (a, c) and free-heave (b, d) inversions are indicated a bathymetry-only (green), slope-only (pink) and combined (grey) misfit measures. The black circled colored-coded points indicate the portion of the profiles used for fitting, with the color representing distance from the axis.

Fixing the heave of the TAG core complex to the 4 km that correspond to the newer detachment, the best-fit models have an elastic thickness between 100 and 200 m, similar to Dante's Dome (Figure 3.17). The fault dips about 55° and is rooted about 2 km beneath the ridge axis. The smaller rooting depth compared to Dante's dome is related to the proximity of the termination to the ridge axis. If the fault rooted at 3km depth, it would have to cross the ridge axis.

If the heave is left free, the inversion results for the TAG core complex favor a heave of 5 to 10.5 km (Figure 3.17c&d; Table 3.5). In that case, the younger breakaway would be interpreted as a rafted block. However, the model topography provides only a poor match to the observed shape. Therefore, I consider that the active detachment is the newer of the two and that the older potential breakaway is unrelated to the current core complex.

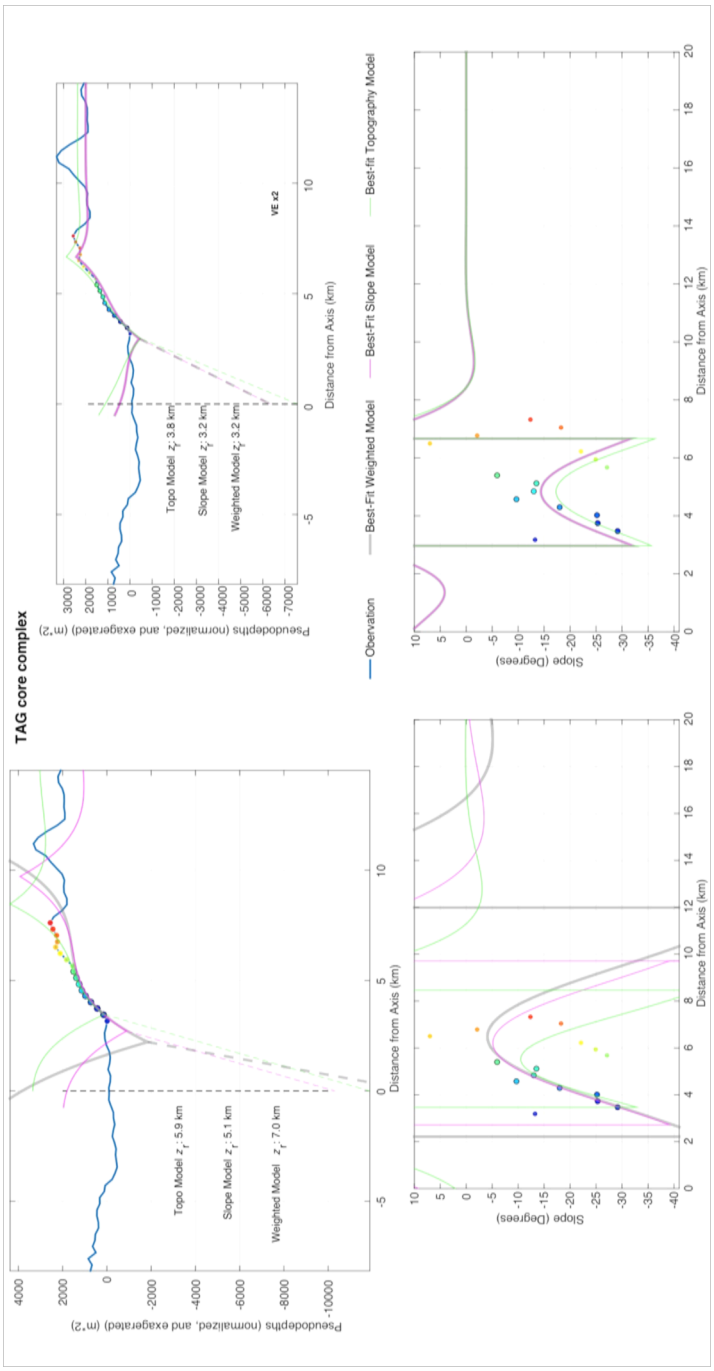


Figure 3.17: Bathymetric (a, b) and slope (b, c) profiles of TAG core complex (thick blue line). The best fit models from fixed-heave (a, c) and free-heave (b, d) inversions are indicated as bathymetry-only (green), slope-only (pink) and combined (grey) misfit measures. The black circled colored-coded points indicate the portion of the profiles used for fitting, with the color representing distance from the axis.

The bathymetric and slope profiles of KMM are best fit by an elastic thickness of 500 m and a fault dip of 45° to 50° (Figure 3.18). Here again, the breakaway is less pronounced in the megamullion than in the elastic models. The depth to which the fault is rooted, or equivalently the offset between the termination and the ridge axis, are irrelevant for this feature as it is inactive and was carried away from the ridge axis.

In all three prototypical OCCs, elastic models provide a reasonable fit to the topography and slope profiles with the observed heave. Remarkably, the elastic thickness for Kane megamullion is more than twice that of the other two examples. It is possible that this difference is due to the position of the KMM along a transform fault while the other cases are in a ridge center. The compilation of results along the 12-15°N MAR segment should provide a new test of this idea. However, it is also possible that the elastic thickness at Kane Megamullion is due to its different age. For example, KMM might have deform once off-axis as the lithosphere thickened and cooled. Fault dip results are consistent for all three core complexes.

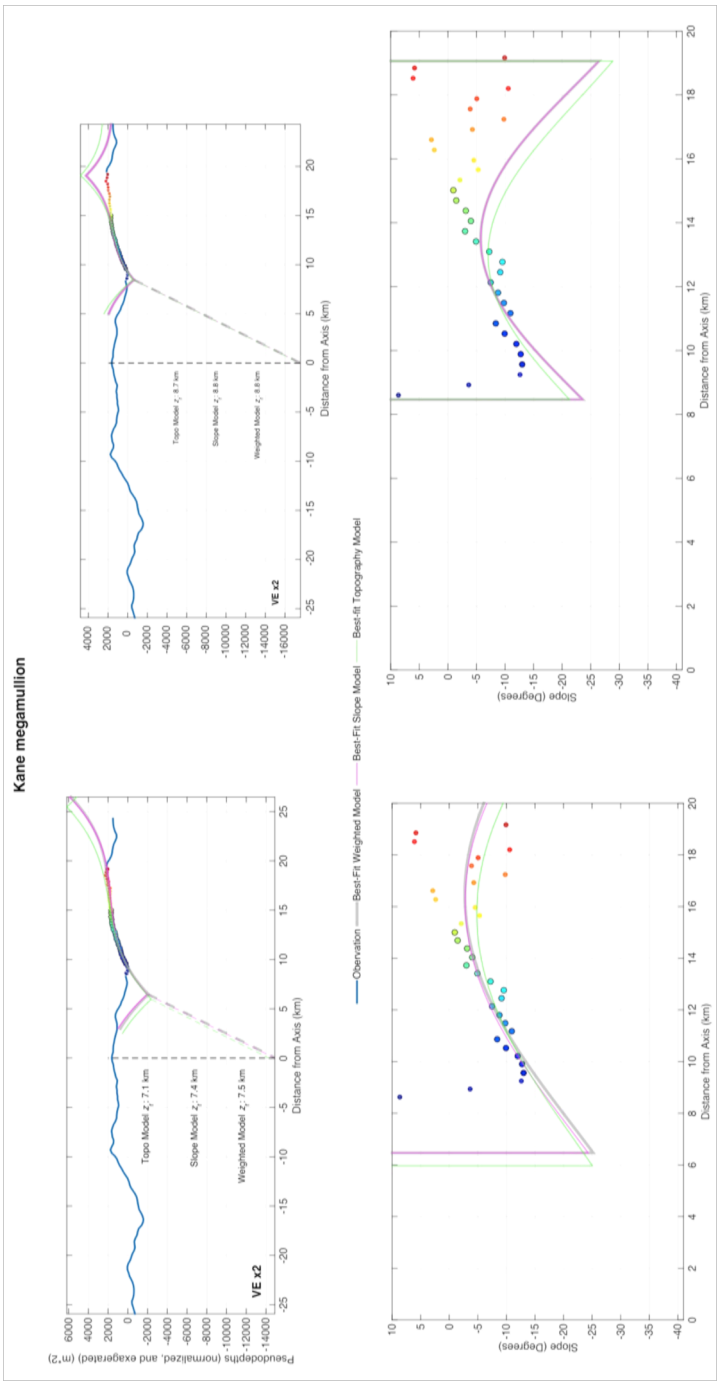


Figure 3.18: Bathymetric (a, b) and slope (b, c) profiles of Kane Megamullion (thick blue line). The best fit models from fixed-heave (a, c) and free-heave (b, d) inversions are indicated a bathymetry-only (green), slope-only (pink) and combined (grey) misfit measures. The black circled colored-coded points indicate the portion of the profiles used for fitting, with the color representing distance from the axis.

3.2.2 Mid-Atlantic Ridge 12 – 15º N Features

I now present the best fits to the ten OCCs proposed for the 12°—15°N region of the Mid-Atlantic Ridge. All ten fit some of the criteria of an OCC, as discussed in Methods. However, the elastic plate models have various degree of success in fitting the observed profiles. The assigned errors for some of the OCCs best-fit models are greater then σ_1, σ_2, or even σ_3. Even in the best cases, there remains systematic differences between models and observations, especially regarding the amplitude or location of the breakaway, and in some case, it can be questioned whether an elastic plate model is at all relevant to explain the observed feature.

For example, Feature 10 displays a drastic relief change but no distinct breakaway. It may correspond to a fault that has not or will not roll over or the breakaway has been modified beyond recognition. Nevertheless, it still can grant insight into the properties of the Earth at mid-oceanic ridges, although that insight must be considered with due caution.

One of the most apparent findings of the research is that fitting the bathymetric profile of a candidate OCC is typically more successful than fitting a slope profile. For example, in Feature 1, the bathymetric profile fits quite well to the eye (Figure 3.19) but the corresponding slope profile is further from model predictions, likely because of the presence of a talus slope at the base of the feature. This is one reason the fit in topography space is usually preferred when discussing the results.

A 500 m elastic thickness model with a fault dipping 45º best fits the bathymetry of Feature 1. The rooting depth for the associated fault in at 2.3 km.

Interestingly the best-fit model is the same for the slope, topography, and the weighted spaces. This is unique amongst the features analyzed. The worst fitting part of the profile is the sharp peak associated with the breakaway. It is missing entirely from the topography profile, and has been eroded or slumped off out of the profiles transect. A final possibility is that this is a misidentified feature.

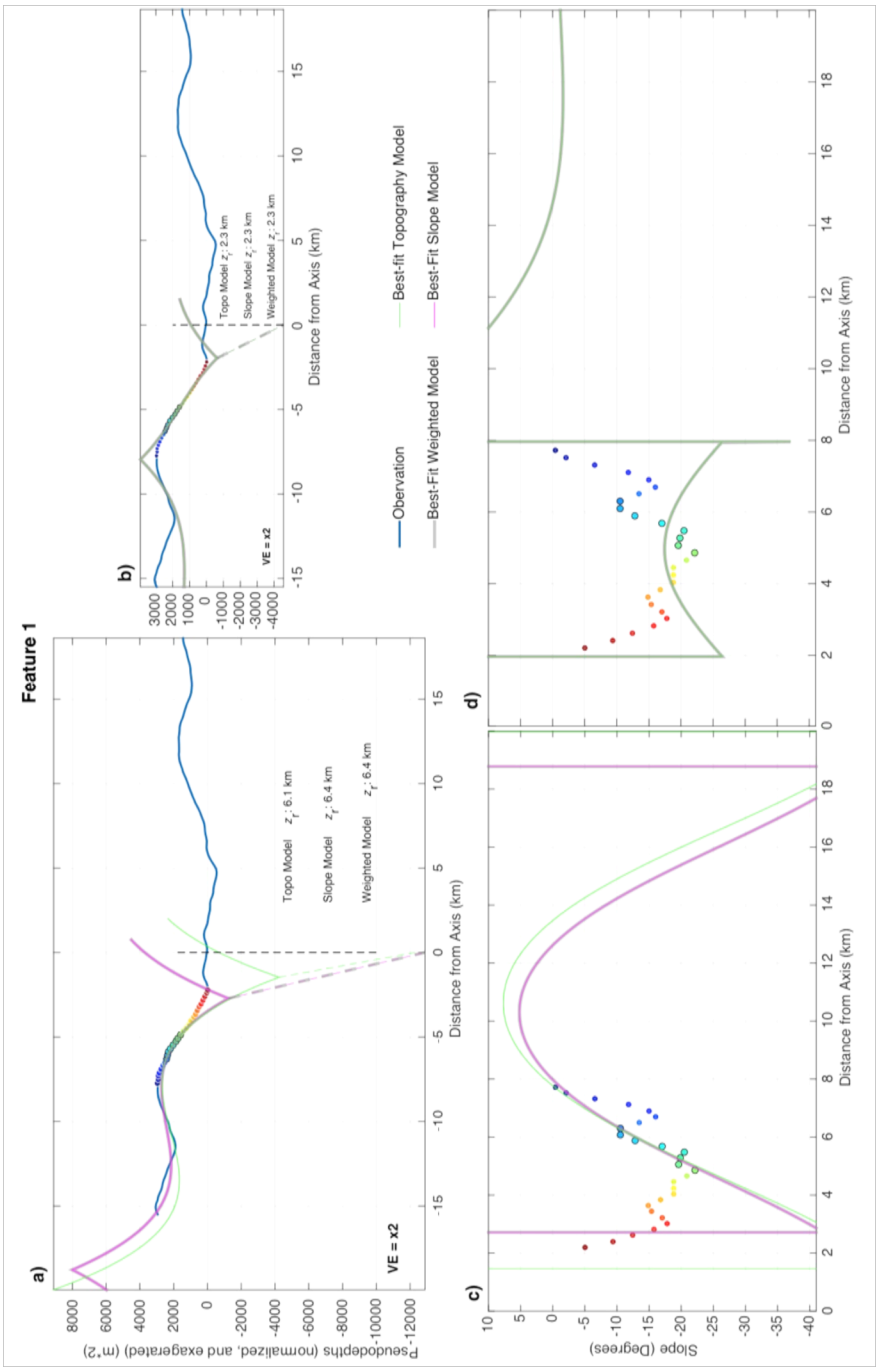


Figure 3.19: Bathymetric (a, b) and slope (b, c) profiles of Feature 1 (thick blue line). The best fit models from fixed-heave (a, c) and free-heave (b, d) inversions are indicated a bathymetry-only (green), slope-only (pink) and combined (grey) misfit measures. The black circled colored-coded points indicate the portion of the profiles used for fitting, with the color representing distance from the axis.

The models that fit Feature 2 are the most variable of any of the features (Figure 3.20). The elastic models always display more curvature than observed. A somewhat flatter surface would be observed if the elastic thickness was much greater than considered here, but in that case, the 10° slope of the detachment would be the original fault dip. The models favored by my inversion scheme imply an elastic thickness of 100 m to 400 m and a fault dipping between 45º and 60º. As the termination is at the axis, the fault would root at 0.1 km to 0.8 km.

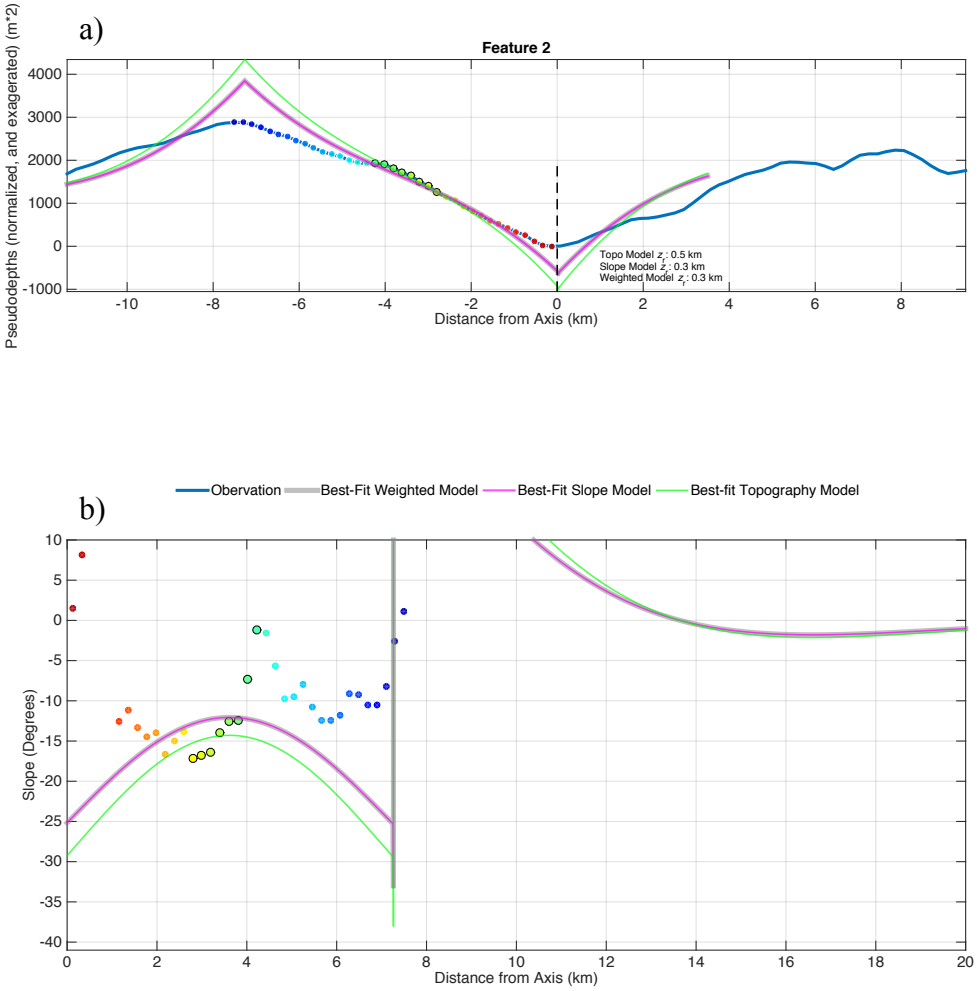


Figure 3.20: Bathymetric (a) and slope (b) profiles of Feature 2 (thick blue line). The best fit models from fixed-heave inversions are indicated a bathymetry-only (green), slope-only (pink) and combined (grey) misfit measures. The black circled colored-coded points indicate the portion of the profiles used for fitting, with the color representing distance from the axis.

Feature 3 (Figure 3.21). is directly north of the previously identified Logachev core complex (Feature 4 in this thesis). Feature 3 is nearer to the ridge axis than Logachev, and has distinct morphology from that of Logachev. Whereas Logachev's elevation change is drastic and lacks a distinct breakaway, Feature 3 has the characteristic roll-over and a more apparent breakaway. It looks like KMM in profile view. The best-fit models for Feature 3 all agree that an elastic thickness of 300 m, a fault dip of 45º and a fault root 2.4 km beneath the axis. Again, we see that the prominent peaked breakaway of the models over-estimate that of the topography. And again post-formational processes of faulting, slumping, and erosion may explain why the breakaway is subdued in the topographic profile.

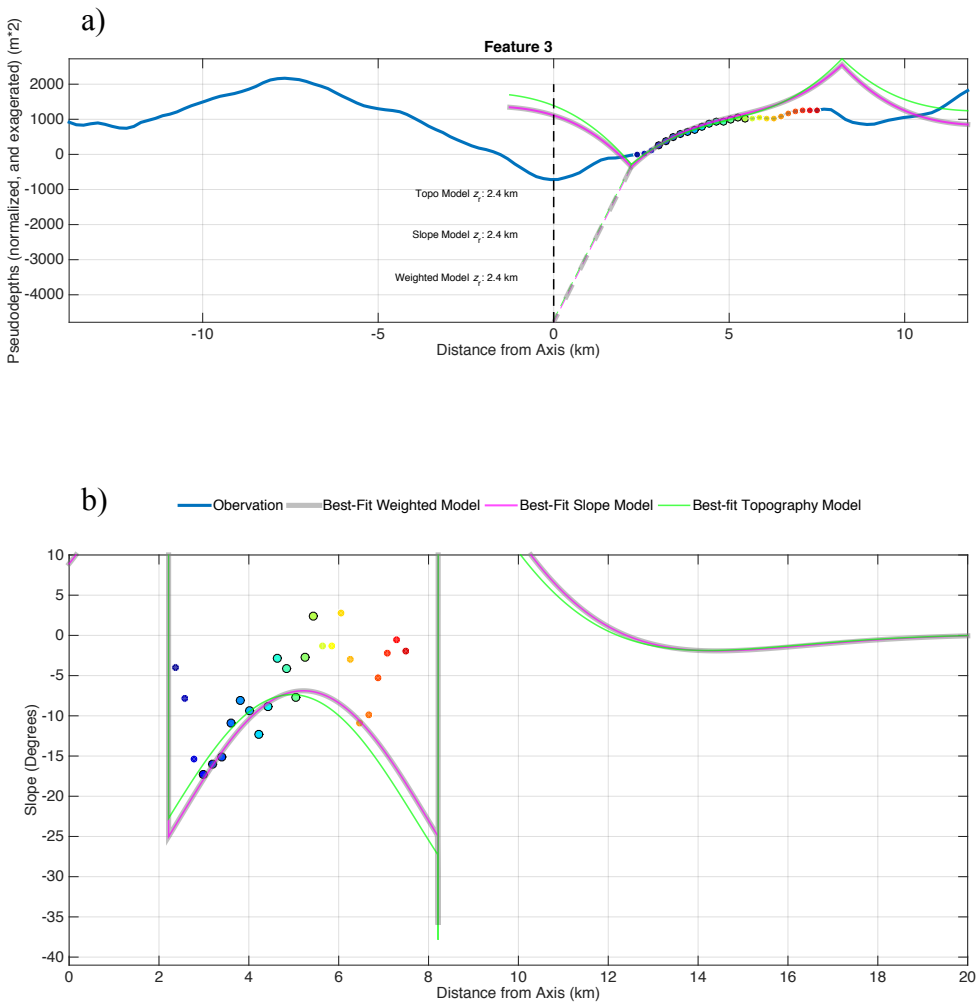


Figure 3.21: Bathymetric (a) and slope (b) profiles of Feature 3 (thick blue line). The best fit models from fixed-heave inversions are indicated a bathymetry-only (green), slope-only (pink) and combined (grey) misfit measures. The black circled colored-coded points indicate the portion of the profiles used for fitting, with the color representing distance from the axis.

Logachev is a well-documented OCC, which hosts hydrothermal vents (Petersen et al., 2009). In this thesis it is Feature 4. In map view it exhibits the distinct U-shape of the termination. It has also experienced significant post-formation deformation (Petersen et al., 2009). This may explain why the models do not fit the feature well (Figure 3.22). Like Feature 2, the models display more significant convexity than observed. The best-fit model suggests that the elastic thickness is 400 to 500 m, and that the fault dips 45º to 55º. This places the fault's root 7.2 to 10.3 km, which is deeper than rooting depth of 6.5 km found at TAG core complex by deMartin et al. (2007).

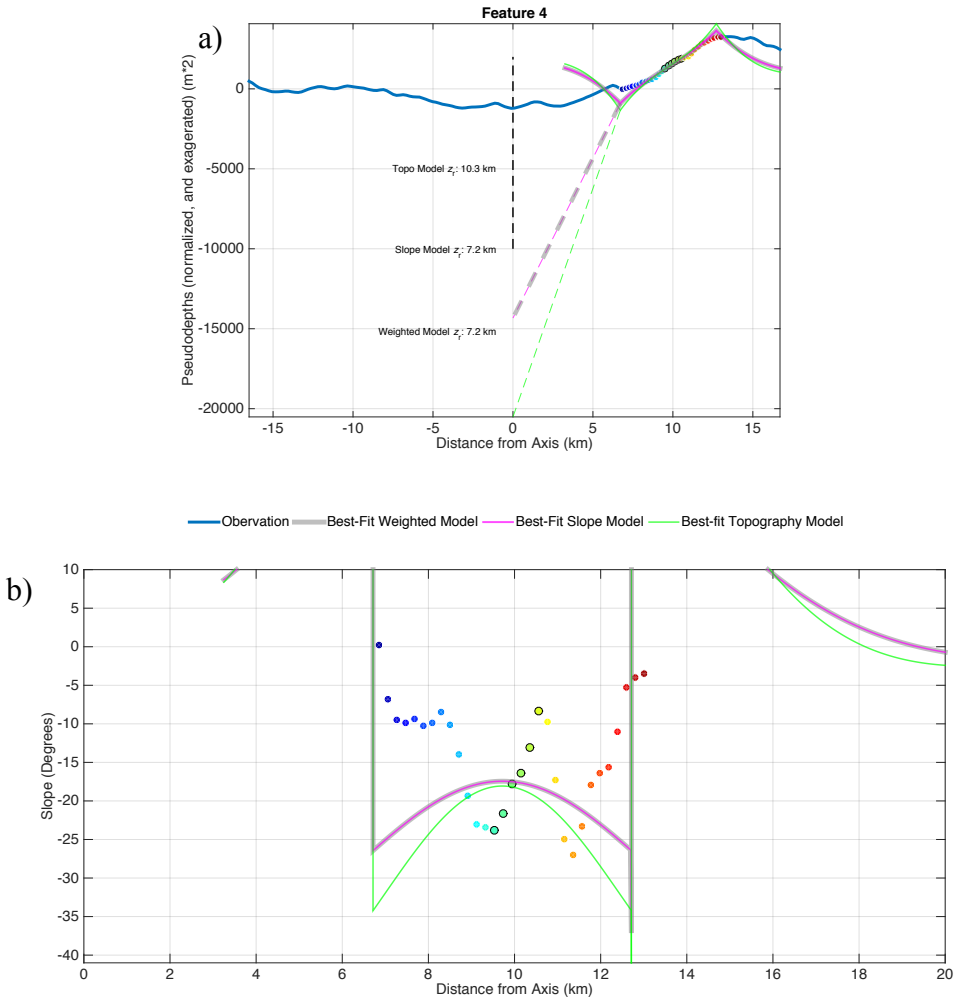


Figure 3.22: Bathymetric (a) and slope (b) profiles of Feature 4 (thick blue line). The best fit models from fixed-heave inversions are indicated a bathymetry-only (green), slope-only (pink) and combined (grey) misfit measures. The black circled colored-coded points indicate the portion of the profiles used for fitting, with the color representing distance from the axis.

Best-fit models for Feature 5 suggest an elastic thickness of 200 m, and a fault that dips 50º to 55º. This places the fault root at 3.6 to 4.4 km. The misfit values are very low for this feature regardless of the inversion method. However, there are important differences between models and observations when the profile is extended beyond the points where the fit is evaluated (Figure 3.23). Feature 5 displays a distinctive double-hump morphology in both topography space and slope space. The second hump begins ~7 km from the ridge axis, and is evidently absent in the best-fit model. Motion on a single fault followed by flexure of an elastic plate cannot produce the second hump. One possible explanation is that post-exhumation faulting could have drastically modified this feature. Another possibility is that the feature has a large rider block or sequence of rider blocks, as suggested by Mallows & Searle (2012). Lastly, the feature I actually model could be a new OCC. In this case, the breakaway of the new feature is eroded, or perhaps hasn't yet developed. Results for inversions with free heave suggest a heave of 9.5 km to 15 km, comparable to the measured heave of 11.5 km. In addition, an elastic thickness of 200 m also fits the free-heave inversion best, with angles of 55º to 60º, only slightly steeper than the fixed heave angles.

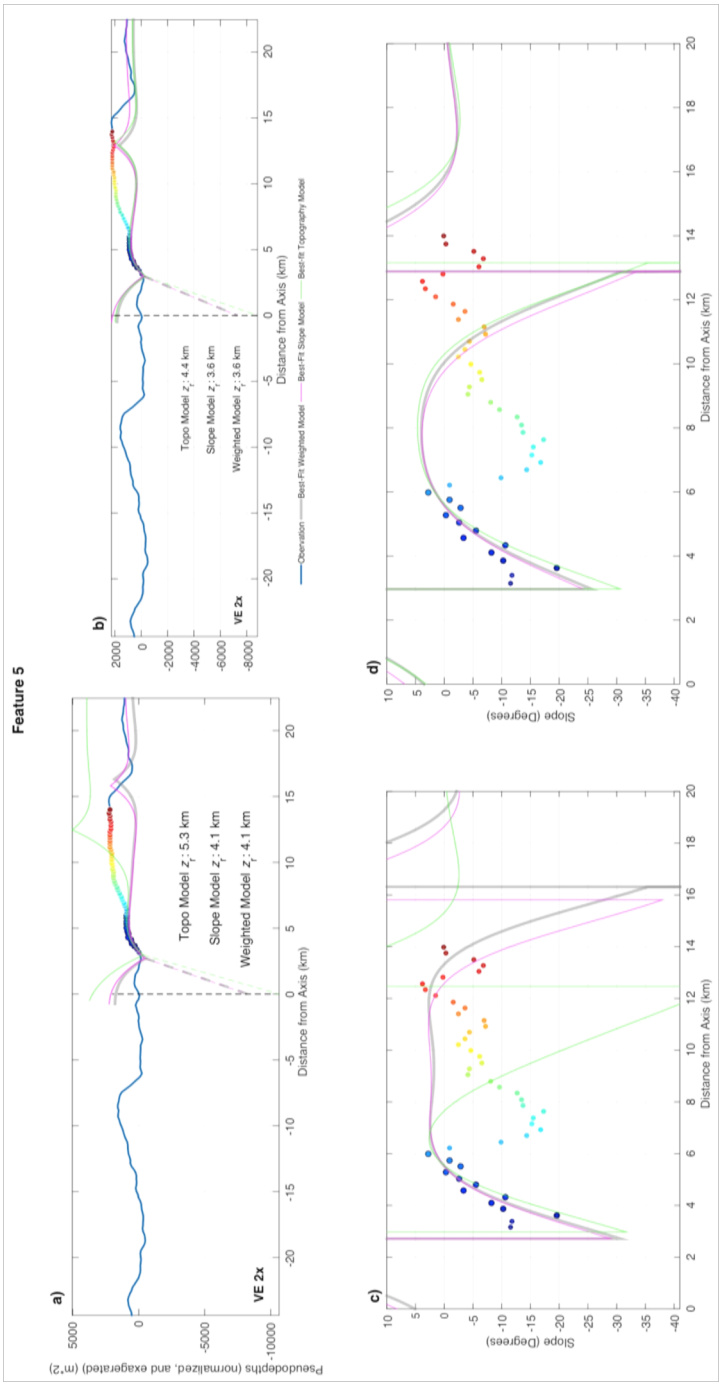


Figure 3.23: Bathymetric (a, b) and slope (b, c) profiles of Feature 5 (thick blue line). The best fit models from fixed-heave (a, c) and free-heave (b, d) inversions are indicated as bathymetry-only (green), slope-only (pink) and combined (grey) misfit measures. The black circled colored-coded points indicate the portion of the profiles used for fitting, with the color representing distance from the axis.

Feature 6 is unique from the other features in this study in that it has an extremely small heave (2 km), and seems to require an even smaller elastic thickness than I set as the threshold (Figure 3.24). For this feature more than any of the other candidate OCCs, the entire bathymetric profile is well represented by the model. In particular, the agreement is good not only on the exposed detachment but also beyond the breakaway and between the termination and the ridge axis. However, the detachment surface does not have the inflection expected from the elastic models (Figure 3.24). This feature requires an elastic thickness of 100 m, the lowest value of those tested. The model fault dips between 45º and 50º, and the fault roots at 3.6 to 4.3 km depth.

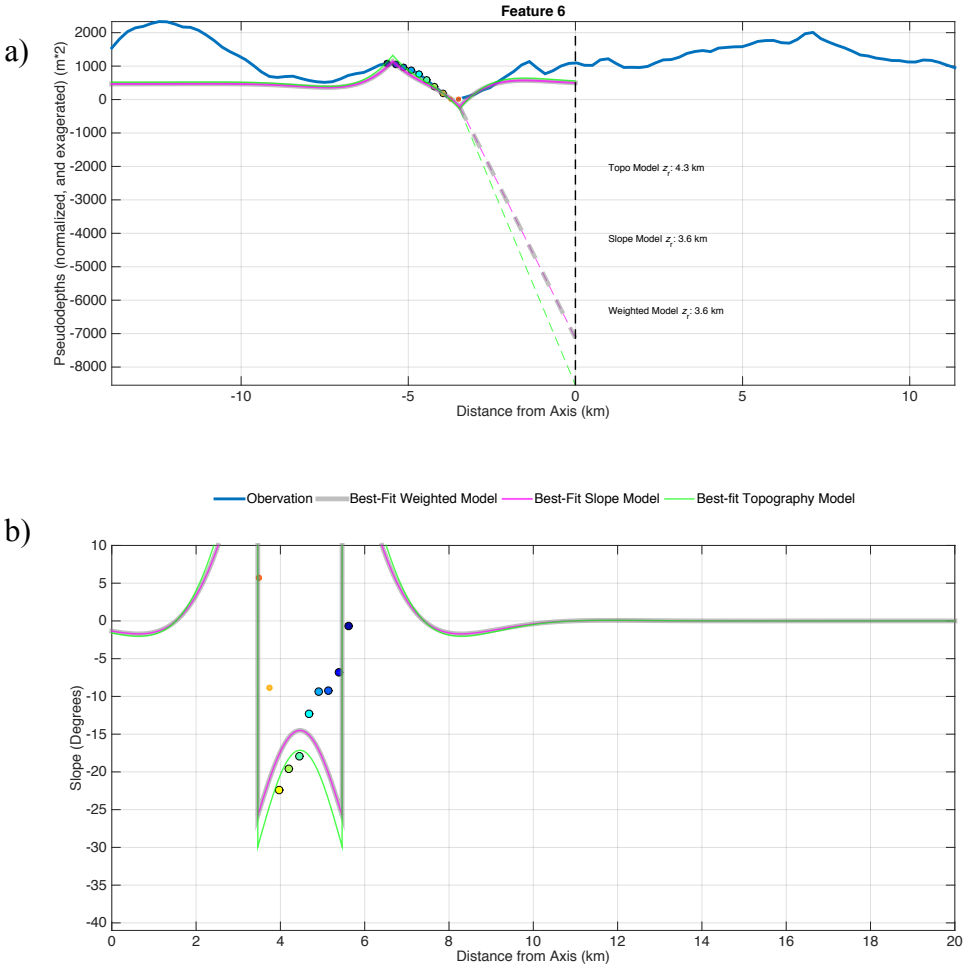


Figure 3.24: Bathymetric (a) and slope (b) profiles of Feature 6 (thick blue line). The best fit models from fixed-heave inversions are indicated a bathymetry-only (green), slope-only (pink) and combined (grey) misfit measures. The black circled colored-coded points indicate the portion of the profiles used for fitting, with the color representing distance from the axis.

Features 7 & 8 (Figure 3.25 and Figure 3.26) are both previously identified OCCs (Smith et al., 2008; Mallows and Searle, 2012), and along with OCC1350 Feature 5, are the inspiration for the project. However, all three of these features are quite distinct, yet do share similarities. Feature 7 lacks the prominent titled breakaway that Features 8 and 5 display. In fact, it has been proposed that Feature 7's breakaway may be located ~10 km further from ridge axis than my current profile displays. I interpret the heave to be only 5 km.

Modeling with a heave of 5 km provides a good fit to the observed feature (Figure 3.25). The overestimation of the bathymetry at the breakaway would imply that a significant volume of material has been moved. Best-fit model parameters for Feature 7 are 100 to 200 m elastic thickness, which agrees with the elastic thickness of feature 5. The model faults dip between 45º and 50º, and with being so close to the ridge axis, would root at a depth of 1.1 to 1.4 km.

The elastic thickness of best-fit results for the free-heave inversion is more than two times that of the fixed heave inversion, being 400 to 600 m thick. The heaves are 15 to 16 km, and associated with a potential breakaway much further from the axis but close to the pronounced ridge sometimes argued to be a second breakaway (Figure 3.7: Map (a) and perspective (b) views of feature 7 showing axis (solid line), termination (dashed line), faults (dashed –dotted lines), breakaway (red shaded region), and profile location (white line).). It is remarkable that these models with large heave capture well the portion of the seafloor that dips away from the axis to the west of the detachment surface that I identified as Feature 7. The angles of all best fit models are similar and lie between 45º and 50º.

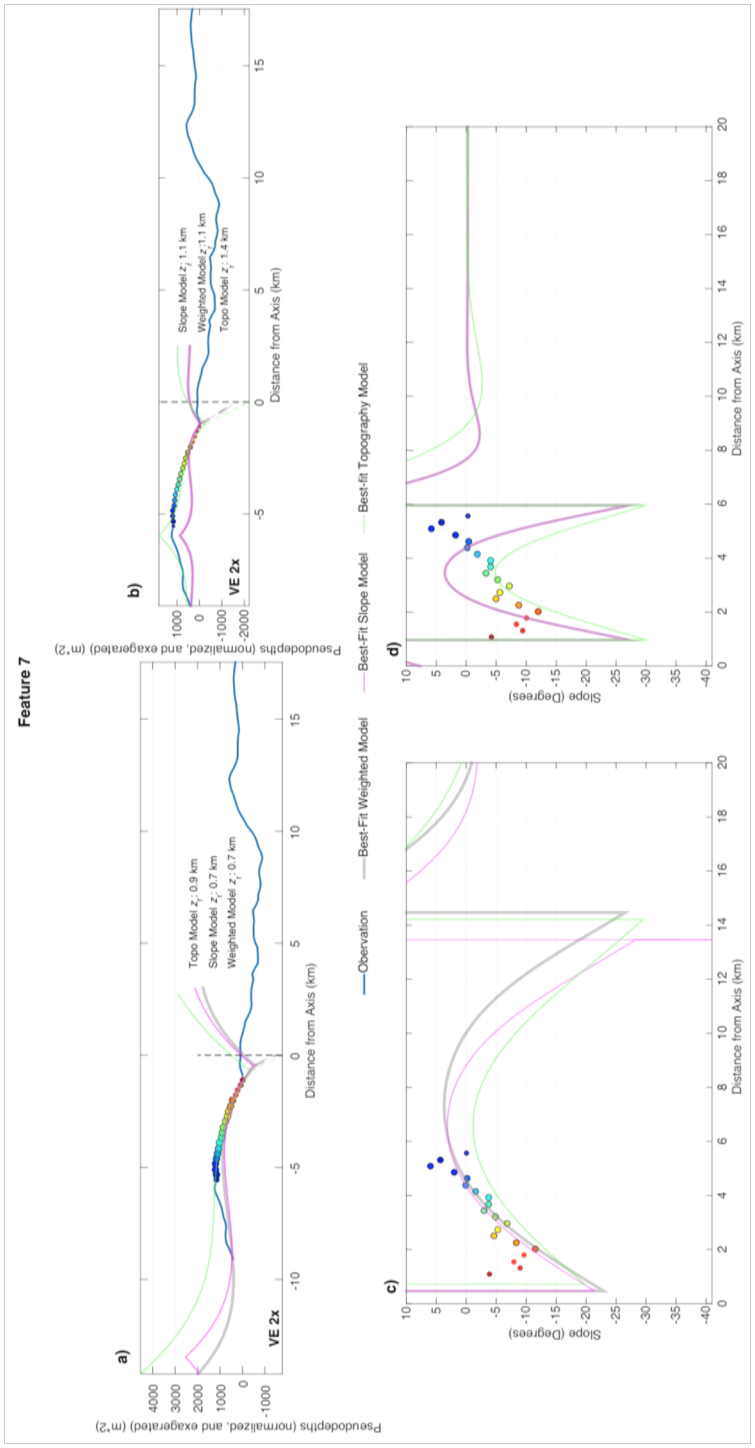


Figure 3.25: Bathymetric (a, b) and slope (b, c) profiles of Feature 7 (thick blue line). The best fit models from fixed-heave (a, c) and free-heave (b, d) inversions are indicated as bathymetry-only (green), slope-only (pink) and combined (grey) misfit measures. The black circled colored-coded points indicate the portion of the profiles used for fitting, with the color representing distance from the axis.

Features 8 & 5 share a ‘double-hump’ morphology, though in either feature it is displayed in distinct ways (Figure 3.26, Figure 3.23). Feature 8 has a very small misfit in all model domains, along the fitted points, although further upslope, the relief of the breakaway is over predicted in the best fit models. Feature 8 is the only candidate OCC for which the slope and combined inversions yield a more realistic model outside of the fitted points than the bathymetric inversion. The best-fit models suggest an elastic thickness of 400 to 500 m, and variable fault dips from 55º to 70º. These put the fault’s root 11.9 to 6.2 km beneath the axis. The faulting of the exposed surface for Feature 8 is extensive as documented by Mallows and Searle (2012) and could explain the double hump morphology and the small but sharp breakaway peak.

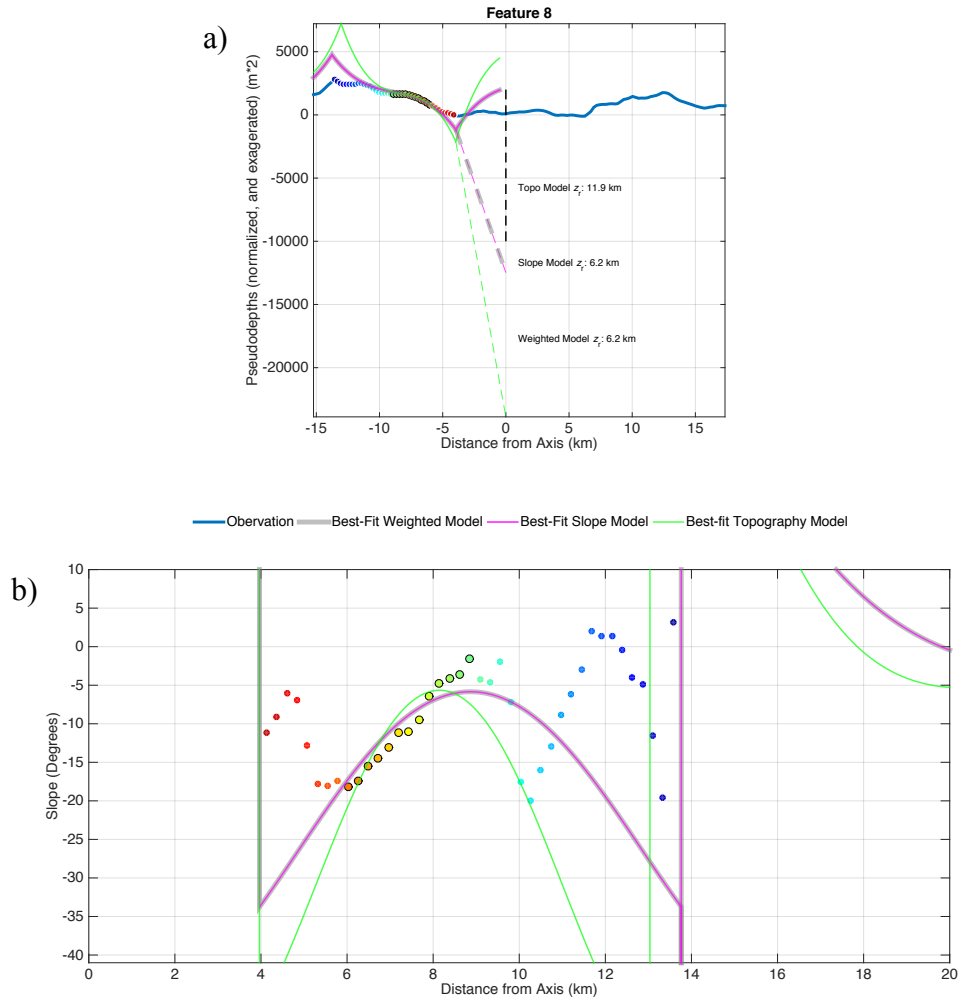


Figure 3.26: Bathymetric (a) and slope (b) profiles of Feature 8 (thick blue line). The best fit models from fixed-heave inversions are indicated a bathymetry-only (green), slope-only (pink) and combined (grey) misfit measures. The black circled colored-coded points indicate the portion of the profiles used for fitting, with the color representing distance from the axis.

Best-fit models to Feature 9 with a fixed heave of 3.7 km agree on an elastic thickness of 300 m, a fault dip of 45º, and a fault root 2.6 km beneath the axis 9 (Figure 3.27). The bathymetric profile, although concave downward, does not display the full rollover characteristic of OCCs. Therefore, it is possible that the breakaway identified here could be a new feature forming in front of an older OCC, producing a situation similar to Dantes' Dome. Alternatively, it could be a post-exhumation fault. Free-heave inversions can distinguish between these possibilities. Free-heave inversions favor a heave of 19 to 20 km, which is five times that of the measured heave, supporting the idea that the detachment surface extends beyond the map corrugated surface. Additionally, the fault dip and elastic thickness are much greater for the free-heave inversion than the fixed heave inversion, picking values of 60º to 70º and elastic thicknesses between 500 and 700 m. Corrugations can be seen on Ashadze extending ~25 km from the axis. If this entire surface is one long-lived OCC then my free-heave inversions more accurately describe the feature. The surface is so broken though and modified that it is difficult to ascertain which is correct.

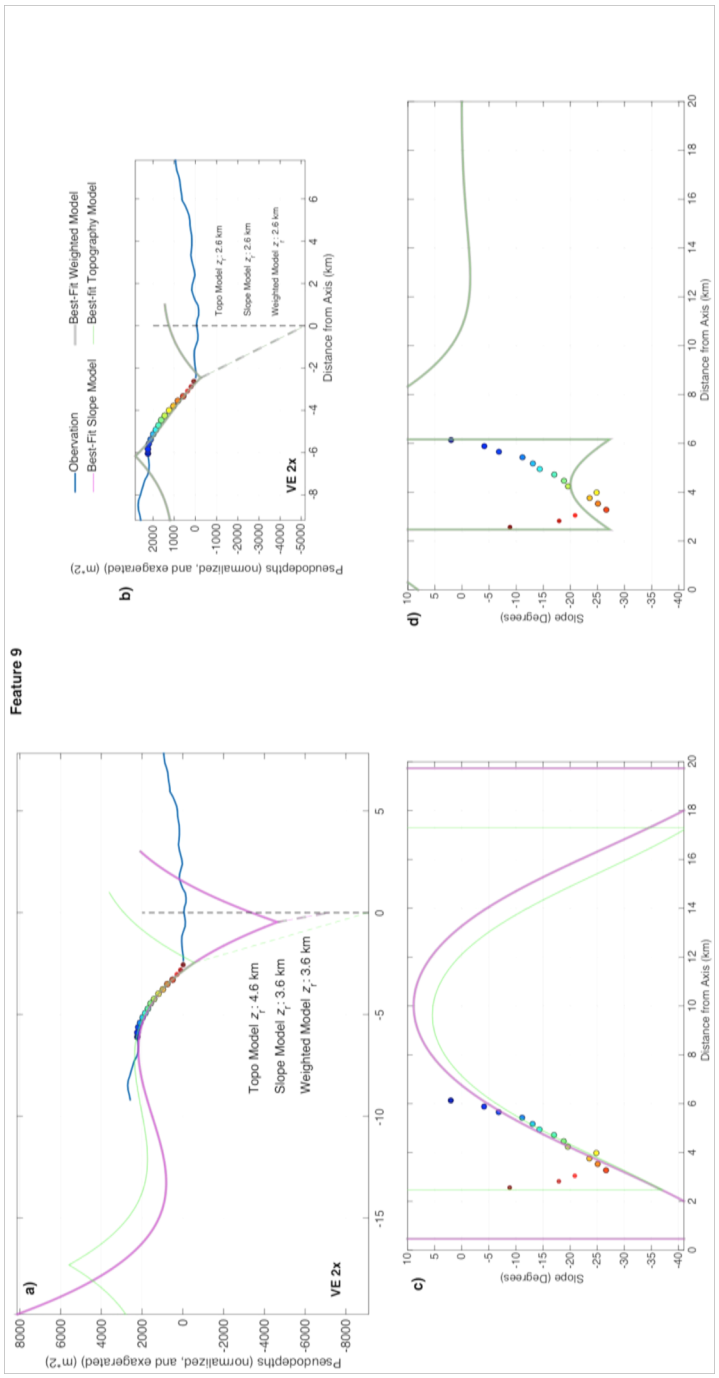


Figure 3.27: Bathymetric (a, b) and slope (b, c) profiles of Feature 9 (thick blue line). The best fit models from fixed-heave (a, c) and free-heave (b, d) inversions are indicated as bathymetry-only (green), slope-only (pink) and combined (grey) misfit measures. The black circled colored-coded points indicate the portion of the profiles used for fitting, with the color representing distance from the axis.

Feature 10 is located at the inside corner of the southern end of the ridge segment in this study area (Figure 3.28). It lacks a corrugated surface, and a strong rollover, but has significant vertical displacement (fault throw) from the axial valley and has a steep breakaway, though the breakaway surface is not smooth. The feature requires the highest elastic thickness (1 km) of any feature identified in this map, which may be attributed to the location of this feature at the end of segment, a similar configuration to Feature 1 and KMM.

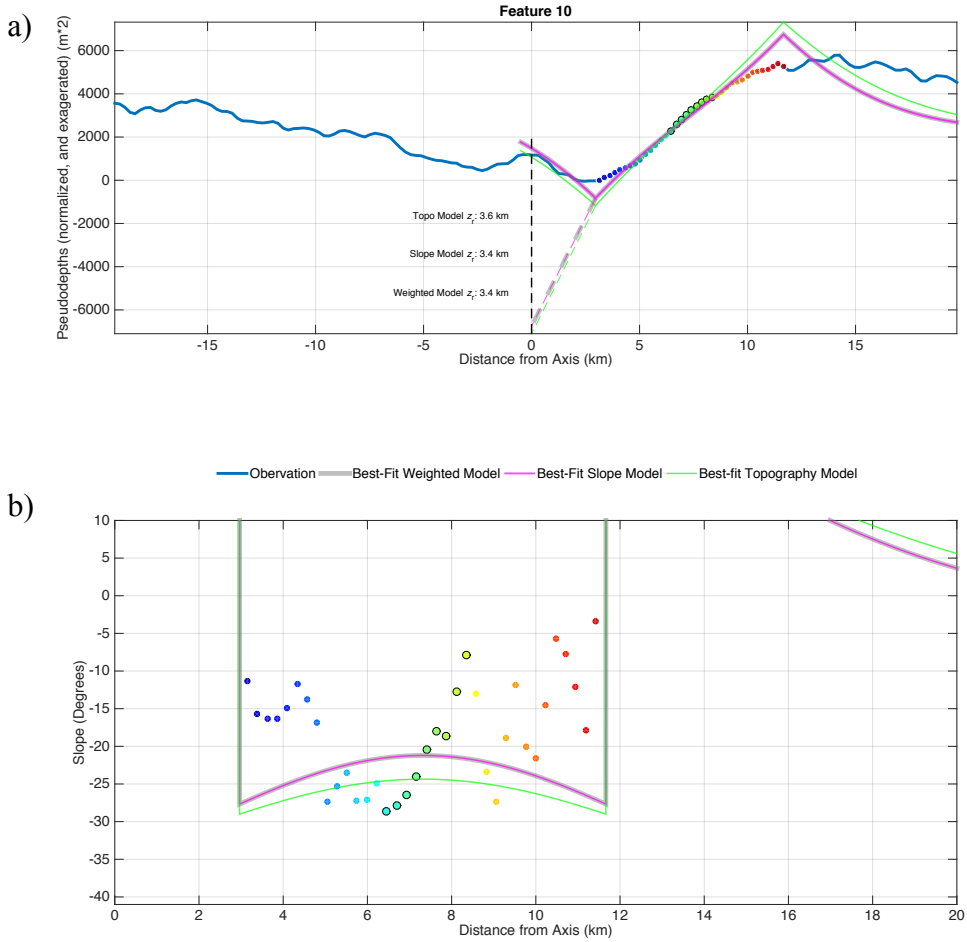


Figure 3.28: Bathymetric (a) and slope (b) profiles of Feature 10 (thick blue line). The best fit models from fixed-heave inversions are indicated a bathymetry-only (green), slope-only (pink) and combined (grey) misfit measures. The black circled colored-coded points indicate the portion of the profiles used for fitting, with the color representing distance from the axis.

Table 3.2: Inversion results for the best fit models by slope method with fixed heave.

Feature	Infill (km)	Angle	Crust (km)	Depth (km)	T_e (km)
Dante's	0	55º	0	3.5	0.2
KMM	0	45º	0	-	0.5
TAG	0	55º	0	1.9	0.2
1	0	45º	0	2.3	0.5
2	0	45º	6	0.1	0.1
3	0	45º	0	2.4	0.3
4	0	45º	0	7.2	0.5
5	2	50º	0	3.6	0.1
6	0	45º	0	3.6	0.1
7	0	45º	0	1.1	0.1
8	0	60º	1	7.6	0.5
9	0	45º	0	2.6	0.3
10	0	45º	0	3.3	1

Table 3.3: Inversion results for the best fit models by topography method with fixed heave.

Feature	Infill (km)	Angle	Crust (km)	Depth (km)	T_e (km)
Dante's	0	50º	6	3.9	0.1
KMM	0	50º	3	-	0.5
TAG	0	50º	6	1.6	0.1
1	0	45º	0	2.3	0.5
2	0	60º	0	0.8	0.4
3	0	45º	1	2.4	0.3
4	0	55º	0	10.3	0.5
5	2	55º	1	4.4	0.1
6	0	50º	0	4.3	0.1
7	0	50º	0	1.4	0.2
8	2	55º	3	11.9	0.4
9	0	45º	0	2.6	0.3
10	0	50º	0	4.2	1

Table 3.4: Inversion results for the best fit models by weighted method with fixed heave.

Feature	Infill (km)	Angle	Crust (km)	Depth (km)	T_e (km)
Dante's	0	55º	0	3.5	0.2
KMM	0	45º	0	-	0.5
TAG	0	55º	0	1.9	0.2
1	0	45º	0	2.3	0.5
2	0	45º	6	0.1	0.1
3	0	45º	0	2.4	0.3
4	0	45º	0	7.2	0.5
5	2	50º	0	3.6	0.1
6	0	45º	0	3.6	0.1
7	0	45º	0	1.1	0.1
8	0	70º	0	6.2	0.5
9	0	45º	0	2.6	0.3
10	0	45º	0	3.3	1

Table 3.5: Inversion results for the best fit models by slope method with heave as a free parameter.

Feature	Heave (km)	Angle	Infill (km)	Crust (km)	Depth (km)	T_e (km)
Dante's	12.5	70º	0	1	4.4	0.4
KMM	20	45º	0	1	-	1.2
TAG	5	60º	0	0	5.1	0.3
1	17	65º	2	3	6.4	0.6
2	13	60º	0	1	2.1	0.4
3	17.5	60º	2	1	2.0	0.6
4	19	70º	0	0	19.8	0.5
5	14.5	55º	2	1	4.1	0.2
6	14.5	70º	2	0	6.5	0.4
7	15	45º	2	1	0.7	0.4
8	19.5	60º	0	0	6.5	0.7
9	20	70º	2	0	3.6	0.7
10	18.5	70º	0	1	12.1	0.6

Table 3.6: Inversion results for bestfit results by topography method with heave as a free parameter.

Feature	Heave (km)	Angle	Infill (km)	Crust (km)	Depth (km)	T_e (km)
Dante's	15.5	70º	2	1	3.8	0.5
KMM	19.5	45º	0	0	-	1.3
TAG	7	60º	0	3	5.9	0.2
1	18.5	70º	0	1	6.1	0.7
2	11.5	50º	2	3	2.4	0.3
3	14	55º	0	1	2.0	0.6
4	14.5	70º	0	0	19.7	0.6
5	9.5	60º	0	6	5.3	0.2
6	13.5	70º	2	3	7.4	0.4
7	15.5	45º	2	3	0.9	0.6
8	20	65º	0	6	7.8	0.7
9	16	60º	2	1	4.6	0.5
10	13	55º	0	1	7.9	0.4

Table 3.7: Inversion results for bestfit results by weighted method with heave as a free parameter.

Feature	Heave (km)	Angle	Infill (km)	Crust (km)	Depth (km)	T_e (km)
Dante's	12.5	70º	0	1	4.4	0.4
KMM	20	45º	0	0	-	1.2
TAG	10.5	70º	2	6	7.0	0.4
1	17	65º	2	3	6.4	0.6
2	13	60º	0	1	2.1	0.4
3	17.5	60º	2	1	2.0	0.6
4	19	70º	0	0	19.8	0.5
5	15	55º	2	0	4.1	0.2
6	14	70º	2	3	6.9	0.4
7	16	45º	2	0	0.7	0.4
8	19	60º	0	0	6.5	0.7
9	20	70º	2	0	3.6	0.7
10	18.5	70º	0	1	12.1	0.6

4 Discussion

4.1 OCC morphology

The portion of the detachment surface closest to the termination is usually fit well by the elastic model. This portion of the detachment is the most recently exposed surface, and the least modified by post-exposure deformation, erosion, and the occurrence of rider blocks. Features 3, 5, 7, and 8 are particularly well fit by the model. Of these four features, three were previously identified as OCCs by Smith et al. (2008) and Mallows & Searle (2012). Features 4 and 9 were also previously identified as OCCs, but are among the worst fitted by the model. The bathymetry of Feature 9 is more concave than can be produced in the model. Feature 4 is convex near the termination, which cannot be reproduced in the model and may reflect surface modification processes. Features 4 and 9 have extensively documented hydrothermal vent fields, which may be the cause of the model discrepancy. Hydrothermal circulation might hasten the modification of the feature by providing lubrication to fault surfaces, damaging bedrock and make it more susceptible to erosion, and generating surface deposits that would appear as noise in the bathymetric data.

In general, my model results do not fit the breakaway regions well. The breakaways have often been moved or removed (Smith et al., 2008; Mallows & Searle, 2012), either in one large slump block (e.g. Feature 5), or in many slumping events (e.g. Feature 7). Evidence of slumping is ubiquitous on the footwall surface (Mallows & Searle, 2012). Slumps add noise to the profile, which prevents accurate

fits to the topography and makes the slope estimates unreliable, as noise is amplified when taking the first derivative of a profile.

Many of the OCCs I model are immediately preceded by another (or multiple) OCC(s) (e.g. Figure 3.9, Figure 3.12, Figure 3.13, Figure 3.14, Figure 3.5, Figure 3.7, Figure 3.3). In some cases a clear breakaway does not exist between the successive OCCs (e.g. Figure 3.5, Figure 3.7). In other cases the possible breakaway may be a rider block (e.g. Figure 3.9, Figure 3.12, Figure 3.13, Figure 3.14). Faulting may even be the cause of the interruption in some of these cases (e.g. Figure 3.5**,** Figure 3.9, Figure 3.12). In all cases it is unclear whether these OCCs share a breakaway. If these features are not part of the same detachment it means that very little activity was taking place between OCC events, similar to the scenario Sauter et al. (2013) identified at the Ultraslow-spreading Southwest Indian Ridge. If the features do share a breakaway, then the free-heave inversion results should be favored for most of the features, although in some of these cases, many of the models deviate staggeringly from model predictions beyond the first possible breakaway (e.g. Figure 3.25, Figure 3.27). Further study is needed to distinguish between these two possibilities.

As mentioned previously, crustal and infill thickness values do not significantly change the model results, though of note, most modeling prefers 0 km thick crust. Likewise, a 0 km thick infill is preferred, except in two cases for fixed-heave inversions (Features 5 and 8). In free-heave inversions, positive infill thicknesses are chosen about half of the time. This is because the misfits are much smaller, and therefore the minute effects of crustal and infill thicknesses can be resolved.

4.2 Comparison to previous studies

This project finds that the elastic thickness of OCCs is on average 340 m from fixed heave inversions and 540 m when heave is not held fixed. Although elastic thicknesses as high as 1300 m are compatible with the Kane Megamullion, many of the smaller features in the study area are compatible with elastic thicknesses as low as 100 m. These values are significantly less than obtained in previous investigations (e.g. Smith et al., 2008; Schouten et al., 2010; Reston and Ranero, 2011). Schouten et al. (2010) and Smith et al. (2008) report an average of ~750 m, with all values lying between 500 and 1000 m. Reston and Ranero (2011) scale previous investigator's modeling, instead of recalculating the models and report elastic thicknesses for OCCs between 380 and 1000 m. One difference between my work and these studies is that I modeled features with large vertical relief and relatively short heave, which may not be sensitive to other effects than elastic plate flexure. The median values for this thesis' inversions are 300 m for using a fixed heave, and 500 m with a free heave, with modes of 100 m and 400 m respectively. By using heaves that are identified from the bathymetry, a much lower elastic thickness is needed to fit the morphology of the features analyzed. The observed heaves are also smaller than those favored by the free-heave inversion. It should be noted that load on the plate increases with heave. A relatively small produces a small load on the elastic lithosphere, which can bend only if its elastic thickness is small. Therefore, the same thin plate implied by fixed-heave inversions may reflect the reduced load associated with the short heave. Smith et al. (2008) and Schouten et al. (2010) used a heave of 60 km for all elastic thicknesses modeled, which is clearly inappropriate for the features of interest here.

It is of some note that the free heave inversions fit the modeled portions of the OCC features better than the fixed heave inversions in almost all cases. In my proposal for this thesis, before I considered a heave other than 60 km, the best fits showed a much stronger trend of increasing elastic thickness away from transform boundaries. It is also important to note that the modeled portions of almost all of the features was restricted to the profile before any rollover, which is also before any significant post-formation deformation could take place. The models may simply not fit the portions beyond the fitted points because of this deformation, however, other factors may be at play here. For example, it could be that breakaways I have identified are simply small perterbations interrupting a very long lived and often obscured 'heaving fault.' This would explain why a heave of 60 km, as Schouten et al. (2010) used, fit the model at all, and quite well in fact. It could also be that the portions of the OCCs beyond the youngest exposed sections are sensitive to the regional characteristics of other features, and have largely lost the flexural response responsible to the OCC formation.

4.3 Fault Mechanics

The fault angles that my models prefer are at the lower threshold for those considered in the modeling. My average angles for a fixed heave inversion are 50º and for a free heave 60º, with the lowest values I report around 45º. Anderson's theory of faulting predicts that fault dip θ_c should be

$$\theta_c = 45^\circ + \frac{\phi}{2} \qquad (17)$$

where ϕ is the angle of internal friction (Davis et al., 2012). Furthermore, the angle of internal friction μ is related to the coefficient of friction ϕ as

$$\phi = \arctan(\mu) \qquad (18)$$

For most rocks, μ is between 0.3 and 0.8 (Byerlee, 1978). Therefore, ϕ is typically between 20° and 40° and the dip of a normal fault should be between 55° and 65°. The angles I obtain are at the lower threshold for Andersonian faulting, as 45° would require a coefficient of friction of 0.

The coefficients of friction for lizardite and antigorite, rocks that are observed along the detachment surface of OCCs, are ~0.55 and ~0.59 respectively (Moore and Lockner, 2010). This gives values for the angle of internal friction ranging from 28º to 30º, which grant critical angles of 59º to 60º. This angle is measured relative to the horizontal, and result in normal faults. Friction coefficients as low as 0.1 are possible when the shear is high and the fault zone has an intense fabric (Collettini et al., 2010). The presence of a pore fluid can also decreases the relation between friction coefficient and fault dip. Therefore, fault dips between 45° and 60° are most compatible with Anderson's theory of faulting.

deMartin et al. (2007) suggest an angle of 70° at the TAG OCC from earthquake hypocenters (Figure 4.1) in agreement with the analysis presented here, which favors fault dip between 50º and 70º, although at the higher end of the results. Fixed-heave model results suggest angles of 50° to 55° and free-heave models 60º to 70º. The free-heave model are more consistent with deMartin et al., (2007) but not with Anderson's theory of faulting. It is possible that the fault initiates at a very high angle due to the presence of rocks with an anomalously high friction coefficient.

However, this is unlikely, especially as any amount of pore fluid pressure during fault initiation would reduce fault dip.

I note that a shallower dip than 70º would fit the seismic data of deMartin et al. (2007) (Figure 4.1). I have compared my model results to their seismic results, and found that while my exact model configuration does not go through the earthquake hypocenter data, a dip of 55º represents the cloud of seismicity well.

The horizontal offset parameter in my model, x_0, was originally defined to show a deviation from the fault geometry proposed by deMartin et al. (2007). Their fault rooted 6.5 km beneath the axis and, with a ~70° dip, would reach the surface 3.5 km from the ridge axis. My offset is relative to this value. Since my modeling allows the fault angle to change, this offset becomes dependent on other variables. As the position of the fault termination is directly observable on the seafloor, a more insightful measure of the fault geometry below the seafloor is the depth at which the fault is rooted beneath the axis, which is a function of horizontal distance and fault dip. For reference, the thickness of the crust at slow-spreading ridges is typically ~5 km, and magma chambers are often at 2 - 3 km depth underneath the axis (Zhou & Dick, 2013; Sinton & Detrick, 1992).

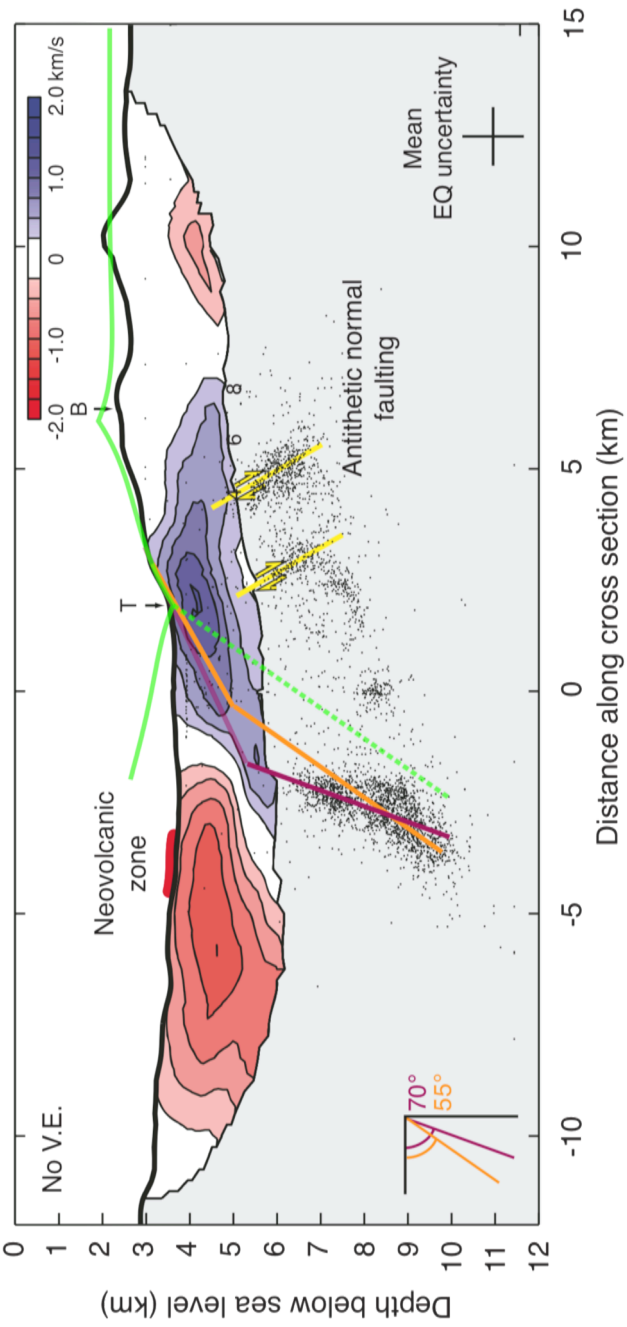


Figure 4.1: Seismic depth image at TAG core complex. Black dots are earthquake epicenters. deMartin et al., (2007) interpreted an ~70º fault (maroon line), I draw an orange line showing a 55º dip. I also overlay my topography inversion model results (green line) [Modified after deMartin et al., 2007]

4.4 Rock Mechanics

Equation 2 shows that the flexural rigidity is not only dependent upon elastic thickness, but also on Young's modulus (E), and Poisson's ratio (v). Because these are properties of the rock, they were not varied in this study, but the Young's modulus of a fractured rock is significantly less than the Young's modulus of an unfractured rock (Gudmundsson, 2011). In fact, the effective Young's modulus (E_e) of a rock is given by,

$$E_e = \left(\frac{1}{E} + \frac{1}{\bar{s}k} \right)^{-1} \qquad (19)$$

where $\bar{s}$ is the fracture spacing, and k is the stiffness of the fractures.

Jiang et al. (2009) show that an unfractured rock with a Young's modulus (E) of 60 GPa, decreases to an effective Young's modulus (E_e) of only 1.2 GPa, with a fracture spacing (s) of 0.5 m, and a fracture stiffness (k) of 2.5 GPa/m. Using a rough order of magnitude decrease in E as a starting point, equation 2 becomes

$$D = \frac{0.1ET_e^3}{12(1-v^2)} \qquad (20)$$

If we consider the effect this can have on T_e (because we have already held E constant), we may formulate it as follows,

$$T_{e_{Effective}}^3 = 0.1T_e^3 \qquad (21)$$

and solving,

$$T_{e_{Effective}} = 2.2T_e \qquad (22)$$

We see that a unaccounted for decrease in Young's modulus in the equation can obscure an increase in elastic thickness. That is, if we had accounted for the decrease

in E near inside corners, we would see larger values of T_e at these locations. This can help to explain why the hypothesis was not well supported by the initial results of the modeling: the increase of elastic thickness trades off against a decrease in Young's modulus to result in an approximately constant flexural rigidity.

One final point of discussion from a rock mechanics view point is the effect that water has on rock strength. As water content increases in a bulk rock, it's strength very typically decreases (Gudmundsson, 2011). Likewise, as fracturing increases in a rock, water content typically increases (Gudmundsson, 2011). This creates a feedback that leads to an overall substantial weakening of the whole rock. Escartin et al., (1997) showed that this feedback leads to high serpentinization of peridotites at the inside corners of transform faults (i.e. super-segment ends). In addition to the decrease in Young's modulus from fracturing, there would be a substantial decrease due to water content increase. In terms of serpentinization the Christensen (1966) showed that for a 30% serpentinized peridotites there was an ~20% decrease in Young's modulus. That experiment did not account for fractures (Chistensen, 1966).

4.5 Geographical variations: Parameters as function of distance from transform

This section concerns the thermal structure of the tectonic segments (centered at 13.25º and 14.9° N), and the super-segment in the 12º - 15º N region (see Figure 1.10). The chemical and topographic anomaly identified by Dosso et al. (1991) may have an affect on the elastic thickness and other properties of the OCCs within the 12º - 15º N super-segment. In my analysis I found that the smallest values for elastic thickness did indeed occur near the center of the ridge super-segment (e.g. Dante's

Dome, features 5 & 6), and that larger values occurred at the inside corners (e.g. feature 10, KMM), although the results overlap within uncertainty. To investigate this initial relationship I plot the elastic thickness associated with each feature in the MAR 12º—15º N super-segment as a function of distance from the nearest transform (Figure 4.2). Plotting the data in this way shows that indeed the smallest values occur near the super-segment center, and large values near the transform faults. This relation is consistent with thermal models of segmented mid-ocean ridges in which mantle temperature is higher near the segment center (Phipps Morgan & Forsyth, 1988; Behn et al., 2004; Fontaine et al., 2008). However, it is noteworthy that only the OCCs within 20 km of a transform display high elastic thickness. The thickness is otherwise fairly uniform. The superposed thermal structure of a normal mid-ocean ridge setting (black lines) and a hotspot's anomaly (red) shown in Figure 1.10 has more than one peak, and does not imply a strictly increasing temperature towards a hotspot's center. This could explain the lack of a strong trend in the elastic thicknesses of the model results shown in Figure 4.2, Figure 4.3, & Figure 4.4. A detailed thermal model of the MAR 12°–15°N super-segment is needed confirm whether the extremely thin elastic plate implied by our modeling and the small extent of the cooling effect associated with the transform are compatible with my observations.

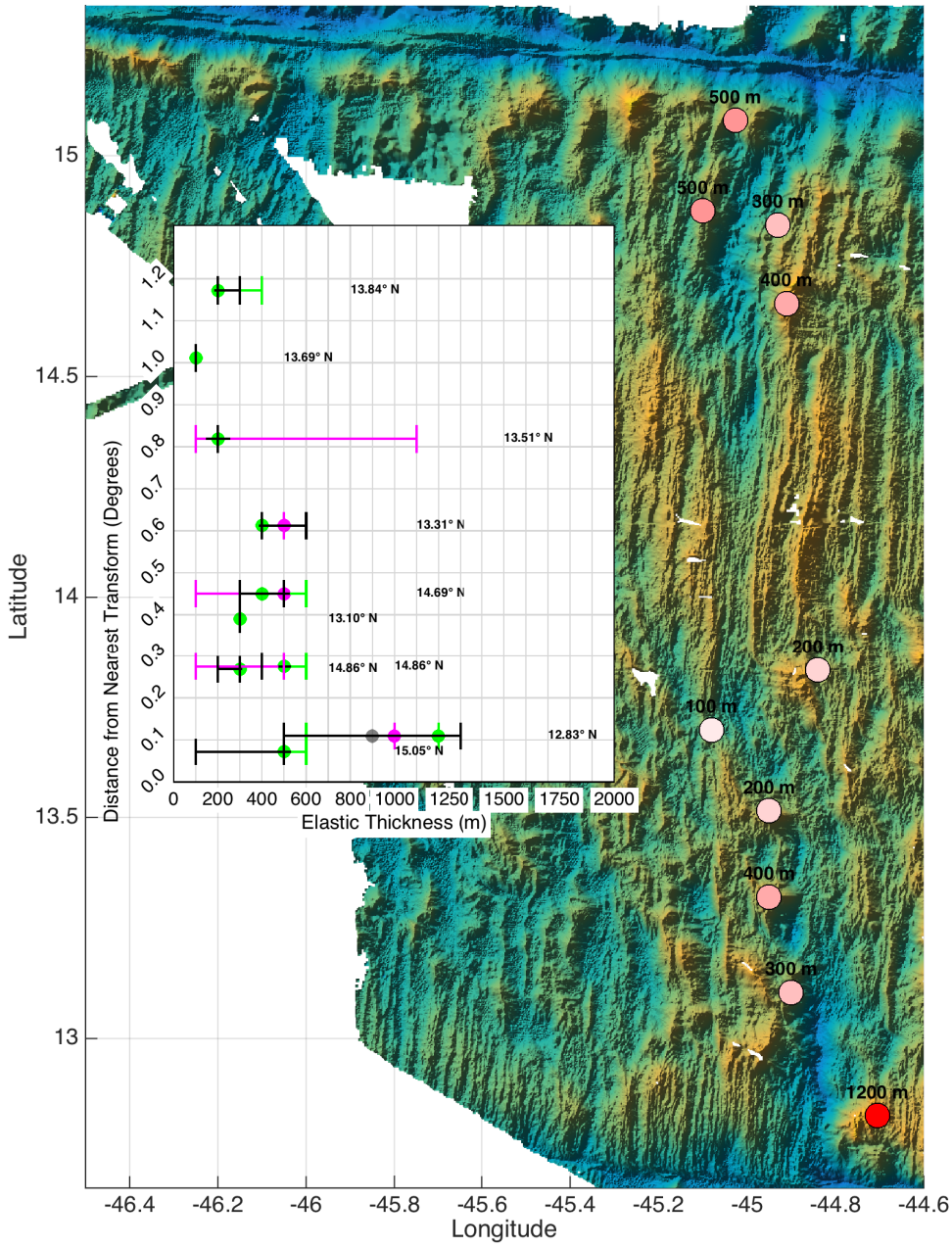


Figure 4.2: Bathymetric map of the 12º - 15º N super-segment with a dot plotted for each feature in the region. Shaded reds indicate the relative elastic thickness of each feature. Inset is a plot of each features distance from the nearest transform vs. its elastic thickness. Green dots are for topography model best fits, grey are for weighted models, and magenta are for slope model best fit elastic thicknesses.

In order to fully investigate any relationship that may exist between OCC characteristics and distance within a ridge super-segment I plot each features best-fit parameters as a function of the feature's distance from the nearest transform for each parameter as well as one-sigma error bars for fixed-heave (Figure 4.3) and free-heave (Figure 4.4) inversions.

With the exception of elastic thickness, no robust trend can be seen in the variation of inversion parameters with distance from the transform. This is actually expected, as there is no know relation between these parameters and the thermal structure of the lithosphere.

The heave measured on each feature is quite variable along the axis (Top left, Figure 4.3). Both the feature with the largest measured heave (Feature 5) and the feature with the smallest measured heave (Feature 6) lie furthest from the transform boundaries. This makes sense, as these OCCs could be at any stage of their growth, and there is no reason to assume that some characteristic of the mantle or lithosphere can be shown by in situ measurement of fault heave. Heave returned by free-heave inversions also do not show a systematic pattern (Figure 4.4, top left). The only feature that would indicate a large heave, Feature 8, was discussed earlier as being unlikely to have a heave that large.

Fault dip likewise shows no discernible trend, especially once uncertainty on the result is taken into consideration. As coefficient of friction does not depend on temperature, no trend should be expected in this parameter.

As stated earlier model results are essentially insensitive to crustal and infill thicknesses. This can be seen readily in both Figure 4.3 and Figure 4.4, where the error bars for almost every feature fill the space.

The depth at which a fault roots is a function of the distance a features termination is from the ridge axis and the angle of the model result. My hypothesis was that rooting depth would be deeper near transform boundaries and shallower at segment centers. Consulting the bottom middle of Figure 4.3 and Figure 4.4 it can be seen that most faults root at about 5 km depth. It may have been possible to have shallower rooting depths if the fault is rooted in the magma chamber, as the center of the super-segment likely features warmer mantle. The proposed 14.5º N mini-hotspot could be the reason that the center of the super-segments would have warmer temperatures.

Elastic thickness results have been discussed for the fixed-heave inversion, however, a few differences exist that can be pointed out when considering free-heave inversions. For one, the elastic thickness of Feature 10 decreases by half. However, the fit of that model result is much worse than the best-fit model for the fixed-heave inversion so the thicker elastic thicknesses reported there are preferred. Importantly, the error bars are skewed to high values so that a uniformly large elastic thickness cannot be ruled out. Fixed-heave inversions consistently return an elastic thickness close to the lower end of the range allowed by free-heave inversions.

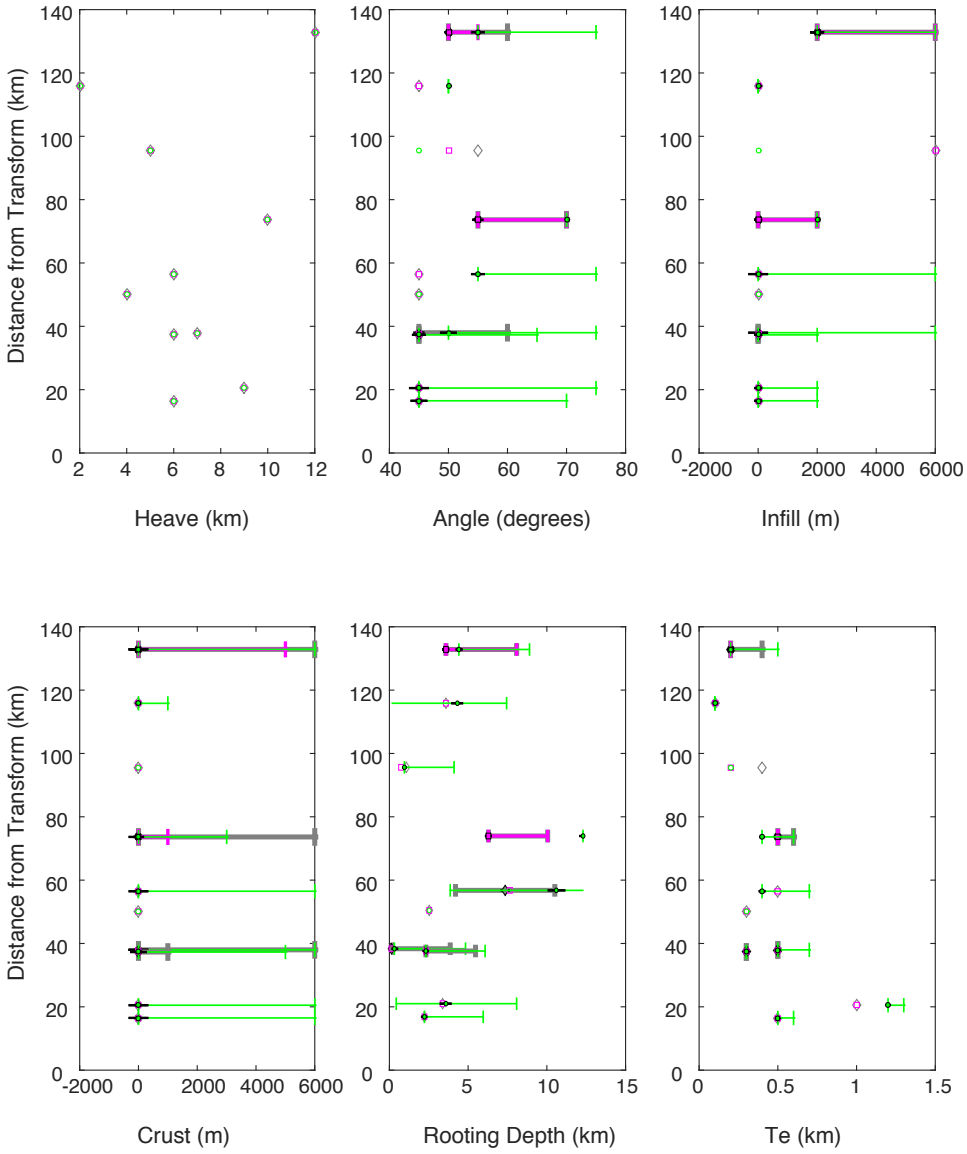


Figure 4.3: Parameters from fixed heave inversions plotted as a function of their distance from the nearest transform boundary. Green dots and lines for topography models, grey diamonds and lines for weighted models, and pink squares and lines for slope models. σ_1 error bars are drawn when available. They are not drawn for rooting depth because of the compounded error from the calculation.

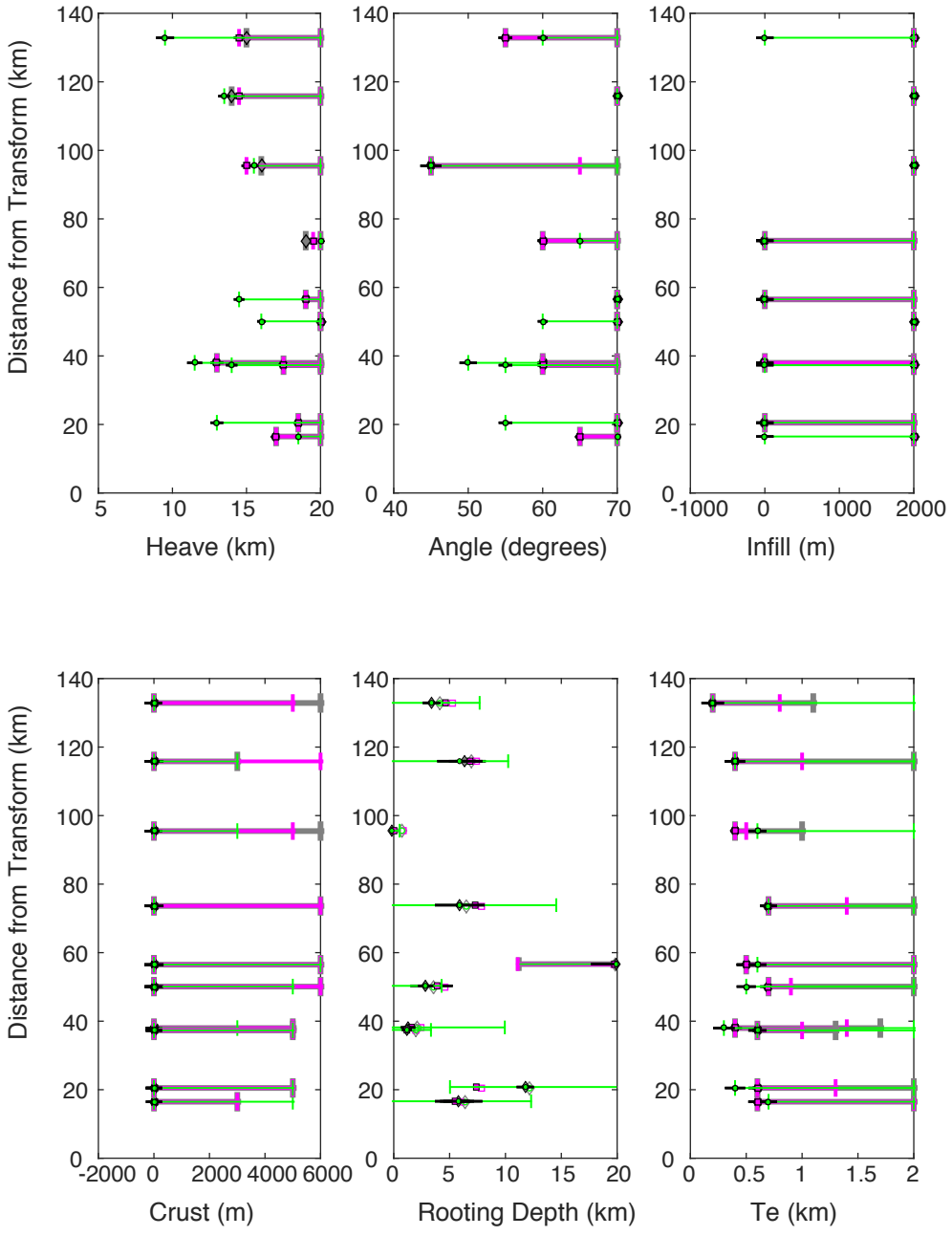


Figure 4.4: Parameters from Free heave inversions plotted as a function of their distance from the nearest transform boundary. Green dots and lines for topography models, grey diamonds and lines for weighted models, and pink squares and lines for slope models. σ_1 error bars are drawn when available. They are not drawn for rooting depth because of the compounded error from the calculation.

5 Conclusion

This thesis investigated the origin of OCC morphology and its variations using an elastic flexure model. While the modeling remained simple in many respects, the conclusions are significantly different from previous research findings. By utilizing a systematic grid search, and considering a much broader range of input variables that includes fault heave and fault dip, the elastic thicknesses suggested beneath oceanic core complexes, and by extension, mid-ocean ridges is much thinner than in previous studies. Fault planes are also suggested to dip more shallowly (~55º) than previously thought (65º –70º, Schouten et al., 2010; DeMartin et al., 2007).

This thesis gives uncertainties to model results and finds that broad ranges of variables are within one-sigma uncertainty. The range of acceptable variables varies significantly across the OCCs modeled. Results of this thesis show that the crustal thicknesses considered do not affect the modeling. Among the variables that do affect the results, this thesis finds no robust relationship between the feature's model variables and the feature's location with respect to the nearest transform boundary. Only the elastic thickness seems to be ~500 m higher in the immediate vicinity of a transform than away from the transform. While this is consistent with the enhanced plate cooling that takes place at the transform, full thermal models are necessary to explore the origin of this variation.

It is clear that this project used an overly simplistic approach to capturing the rock properties along the mid-oceanic ridges. There is significant reason to account for these in future modeling endevours, where the flexural rigidity and the density

inputs of the plate should be assigned based upon the proximity of an OCC to a transform fault, especially if the OCC is at an inside corner.

This thesis, and previous studies do show that treating oceanic core complexes as thin elastic plates is a promising method of analysis. However, the flexure model imperfectly represents certain features, especially the breakaway. Future modeling should treat elastic thicknesses as a distance dependent variable, simulating the cooling of the oceanic crust as the fault slips. Future models should also incorporate more complicated density distributions, to match the layering of the lithosphere of oceanic plates.

6 Appendices

6.1 axisinterpolater.m

```
function [axis_interp,slopeE_Eside,slopeW_Wside,slopeE_Wside,slopeW_Eside] =
axisinterpolater(axis,ll1,ll2,slopeE_Eside,slopeW_Wside,slopeE_Wside,slopeW_Esid
e)
% ll1 is lat or long, ll2 is tother
dothetwist = abs(axis(1,1) - axis(end,1)) > abs(axis(1,2) - axis(end,2));
if dothetwist
   axis = fliplr(axis);
   slopeE_Eside = slopeE_Eside';
   slopeW_Wside = slopeW_Wside';
   slopeW_Eside = slopeW_Eside';
   slopeE_Wside = slopeE_Wside';
end

for n = 1:length(ll2)
   tempEE = zeros(size(ll1));
   tempWW = zeros(size(ll1));
   tempWE = zeros(size(ll1));
   tempEW = zeros(size(ll1));
  for m = 1:2:size(axis,2)
   ay=axis(find(isfinite(axis(:,m+1))),m+1);
   ax=axis(find(isfinite(axis(:,m))),m);
      axis_interp(n,m) = interp1(ay,ax,ll2(n),'linear');
      if isfinite(axis_interp(n,m))
         axis_interp(n,m+1) = ll2(n);
      else
         axis_interp(n,m+1) = NaN;
      end
      [nul intind] = min(abs(ll1 - axis_interp(n,m)));
      if dothetwist
         plot(axis_interp(n,m+1),axis_interp(n,m),'.k')
      else
         plot(axis_interp(n,m),axis_interp(n,m+1),'.k')
      end
      if isfinite(nul)
         % EE
         temp = ll1*0 + 1;
         temp(1:intind) = 0;
         tempEE = tempEE + temp;
         % WW
         temp = ll1*0 + 1;
         temp(intind:end) = 0;
```

```
            tempWW = tempWW + temp;
            % WE
            temp= ll1*0 + 1;
            temp(1:intind) = 0;
            tempWE = tempWE + temp;
            % EW
            temp = ll1*0 + 1;
            temp(intind:end) = 0;
            tempEW = tempEW + temp;
        end
    end
    slopeE_Eside(n,find(tempEE == 0)) = NaN;
    slopeW_Wside(n,find(tempWW == 0)) = NaN;
    slopeW_Eside(n,find(tempWE == 0)) = NaN;
    slopeE_Wside(n,find(tempEW == 0)) = NaN;
    %%%
%     slopeE_Eside(n,1:intind) = NaN;
%     slopeW_Wside(n,intind:end) = NaN;
%     slopeW_Eside(n,1:intind) = NaN;
%     slopeE_Wside(n,intind:end) = NaN;
    %%%
end

if dothetwist
    for m = 1:2:size(axis_interp,2)
        axis_interp(:,m:m+1) = fliplr(axis_interp(:,m:m+1));
    end
    slopeE_Eside = slopeE_Eside';
    slopeW_Wside = slopeW_Wside';
    slopeW_Eside = slopeW_Eside';
    slopeE_Wside = slopeE_Wside';
end
hold off
disp('**********************************************************************
***');
disp('**********************************************************************
***');
disp('**********************************************************************
***');
disp('**********************************************************************
***');
disp(sprintf('Axis has %g segment(s), total of %g points that means %.2f %%
overlap',m/2+.5,numel(find(isfinite(axis_interp)))/2,(1-
((numel(find(isfinite(axis_interp)))/2)/length(ll2)))*100))
```

6.2 axisbreaker.m

```
function broken_axis = axisbreaker(axis)

Ay = axis(:,2);
SegRid.Idx = find(sign(diff(Ay)) > -1);
temp2 = [];
if isempty(SegRid.Idx)
   broken_axis = axis;
else
   if isempty(SegRid.Idx)
      SegRid.Idx = length(Ay);
   else
      SegRid.CrossOvers = Ay(SegRid.Idx-1);
      SegRid.Idx = [1 SegRid.Idx' length(Ay)];
      jk=0;
      for n = 1:(length(SegRid.CrossOvers)+1)
         temp1 = axis*NaN;
         temp1(SegRid.Idx(n)+jk:SegRid.Idx(n+1),:) =
axis(SegRid.Idx(n)+jk:SegRid.Idx(n+1),:);
         temp2 = [temp2  temp1];
         jk = 1;
      end
      broken_axis = temp2;
   end
end
```

6.3 CCslopeLooper.m

```
function [tslope ttopo tdistances topodistances] =
CCslopesLooper(heaves,angles,crusts,infills,depths,tes)
%
% CCslopevs Te.m calculates curves of detachment slope vs distance from the axis as
seen in Schouten et al  (2010) figure2')
% more te s can be added in line 6 of this m-file')
% program derived from GEOLOGYfigure2.m
% CCslopevsTe calls GEOLtestfaultnofhnote.m, flex.m, and dofault from folder
FLEXURE Javier')
% Hans Schouten July 2015
% InpuTs
%   heaves: vector or scalar of horizontal extensions of the fault
%   angles: vecotr of scalar of angles for the fault geometyr
%   crusts: vector or scalar of crustal thicknesses
%   infills: vecotr or scalar of infills thicknesses
%   depths: vector or scalar of depths of fault roots
%   tes: vector of elastic thicknesses in meters
%       default = 6
```

```
% Outputs
%   slopes: matrix of slope values for Te
%   distances: matrix of coresponding y elevations
%
% loop through all Te values
% this loop generates the topography for each Te
% and it finds the slope of the topography
% the topography is not saved, or plotted in this script.

% first initalize the counter
icount=0;

% second cd to where the calculator scripts are
cd mfiles

% now loop through parameters

for fh = heaves
   for an=angles
      for te=tes
         for ct=crusts
            for ift = infills
               icount=icount+1;
               % call the calculator script
               [fyt, ndx, nnx, yt] = slopecalc(fh,te,1,ct,an,1e5,ift);
               % now find offset for assuming the fault begins at
               % depth 6 km and angle = an
%                 depth=6; % km, from Schouten
%                 dx60=depth/tan(an*pi/180);
               % now we find the min value, and we only cut it there.
               % this is because we had doubled the topography
               [yy,ii]=min(fyt(1:end/2));
               yyy=fyt(ii-4540:ii+30000);%fyt(ii:ii+30000-1);
               xxx=ndx*(1:length(yyy));
               % take slope of the topography
               % not sure why we do this
               slope=atan2(diff(yyy),diff(xxx))*180/pi;
               % we make the slopes negative, because they are 'pointing toward' the
ridge
               % axis, That is, the footwall is sloping toward the ridge axis.
               tslope(:,icount)=-slope;
               % topography of the slope data with some of the
               % footwall
               ttopo(:,icount)=fyt(ii-4540:ii+30000-1);
               % full topo here
               %ttopo2(:,icount)=fyt(1:end/2);
```

```
            % distance vector
            tdistances(:,icount) = nnx(1:length(tslope))'/1000;%+dx60;
            topodistances(:,icount) = nnx(1:length(ttopo))'/1000;
        end
    end
  end
end

% then cd back

cd ..
```

6.4 Centroid_spreadingprofile.m

```
function [gp1 gp2 AC TopProf BotProf ] =
Centroid_spreadingprofile(bathy,axis,slopestruct,lat,long,n,SD)
% Centroid_spreadingprofile Multiple Profiles along spreading direction.
% Usage: Centroid_spreadingprofile(bathy,axis,lat,long,slopestruct,n,SD,LBF,RBF)
% Mark Oscar Larson 2015
%%% INPUT
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% bathy:    bathymetry map, can ontain NaNs
% axis:     matrix of axis segments
% lat:      vector of latitudes, must match size(bathy,2)
% long:     vector of longitudes, must match size(bathy,1)
% LL:       sturcture of lat long points from which profiles will be
%             collected
% n:        which lat long point in LL is currently being worked on
% SD:       azimuth of spreading direction
% RBF:      right/east bounding fault
% LBF:      left/west bounding fault
%
%%% OUTPUT
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% gp1 (good Profile):    the x-axis (lat or long) points of profile
% gp2 (good Profile):    the y-axis (lat or long) points of profile
% AC (axis_crossing):     point on the axis that the main profile crosses
%%%
% ACB (axis_crossing):    point on the axis that the bottom profile crosses
% ACT (axis_crossing):    point on the axis that the top profile crosses
%%%
% TopProf:          A profile parallel to the good profile, but above it
% BotProf:          A profile parallel to the good profile, but below it
```

```
%%%  unassigned -mol
% LBF_crossing:     point on the left bounding fault profile intersection
% RBF_crossing:     point on the right bounding fault profile intersection


% first, find which orientation axis is
%if abs(axis(1,1)-axis(end,1)) < abs(axis(1,2)-axis(end,2))
if sum(abs(diff(axis(isfinite(axis(:,1)),1)))) < sum(abs(diff(axis(isfinite(axis(:,2)),2))))
   MA = 1;
   MM = 2;
   l1 = long;
   l2 = lat;
else
   MA = 2;
   MM = 1;
   l1 = lat;
   l2 = long;
end

% set up the stuff
mid = (slopestruct.rps(n).Centroid);
TopMid = slopestruct.rps(n).MajorAxisTop;
BotMid = slopestruct.rps(n).MajorAxisBot;
TopMid1 = TopMid(MA);
TopMid2 = TopMid(MM);
BotMid1 = BotMid(MA);
BotMid2 = BotMid(MM);
mid1ix = round(mid(MA));
mid2ix = round(mid(MM));
mid1 = l1(mid1ix);
mid2 = l2(mid2ix);

% do axes stuff
[ax1 ax2] = find_crossing_line(axis,MA,MM,mid1,mid2ix);
% [LBF1 LBF2] = find_crossing_line(LBF,MA,MM,mid1,mid2ix);
% [RBF1 RBF2] = find_crossing_line(RBF,MA,MM,mid1,mid2ix);

% find which side of ridge starting point is on
% so we multiply our profile distances accordingly
if mid1 > ax1(mid2ix)
   % to da right!
   toda1 = 2.25;
   toda2 = 0.25;
else
   % to da left!
   toda1 = 0.25;
```

```
   toda2 = 2.25;
end
%%
sino = sind(90-SD);
coso = cosd(90-SD);
sico(MA) = sino;
sico(MM) = coso;
slope = (tand(90-SD));
% Now other side of profile properties
sino2 = sind(90-SD+180);
coso2 = cosd(90-SD+180);
sico2(MA) = sino2;
sico2(MM) = coso2;
% find y axis intersections
b = mid2 - slope*mid1;
bB = BotMid2 -slope*BotMid1;
bT = TopMid2 -slope*TopMid1;
testy = slope*ax1 + b;
testyB = slope*ax1 + bB;
testyT = slope*ax1 + bT;
% find the index for our axis point
% as well as left and right bounding faults
[nul axidx] = min( abs( abs(testy) - abs(ax2) ) );
[nul axidxB] = min( abs( abs(testyB) - abs(ax2) ) );
[nul axidxT] = min( abs( abs(testyT) - abs(ax2) ) );
% [nul LBFidx] = min( abs( abs(slope*LBF1 + b) - abs(LBF2) ) );
% [nul RBFidx] = min( abs( abs(slope*RBF1 + b) - abs(RBF2) ) );
% find distance to this point
length2axis = ll2m([ax1(axidx) mid1],[ax2(axidx) mid2]);
hypo = sqrt((ax1(axidx) - mid1)^2 + (ax2(axidx) - mid2)^2);

%%
% figure(10)
% clf
%
% imshade(long,lat, bathy);
% %surf(long,lat,bathybu)
% hold on
% plot(ax1,ax2,'-','Color',[.8 .8 .8],'LineWidth',1.3)
% shading interp
% view([0 90])
% plot(ax1,testy,'--k')
% plot(mid1,mid2,'b*')
% plot(ax1(axidx),ax2(axidx),'r*')
% %%
% plot(gp1,gp2,'-r')
```

```
%%
% make sure profile is sufficient length
% if length2axis*2<minproflength
%     hypo = hypo*(minproflength/length2axis);
% end
extentA = [sign(slope) -1].*hypo.*toda1.*sico + [mid2 mid1];
extBotA = [sign(slope) -1].*hypo.*toda1.*sico + [BotMid2 BotMid1];
extTopA = [sign(slope) -1].*hypo.*toda1.*sico + [TopMid2 TopMid1];
% and other side of profile
extentB = [sign(slope) -1].*hypo.*toda2.*sico2 + [mid2 mid1];
extBotB = [sign(slope) -1].*hypo.*toda2.*sico2 + [BotMid2 BotMid1];
extTopB = [sign(slope) -1].*hypo.*toda2.*sico2 + [TopMid2 TopMid1];


% make the profiles contain 'sufficiently' many points
% find out how big?
% maybe later
% [ nul idx1 ] = min( abs( abs(l1) - abs(extentA(2)) ) );
% [ nul idx2 ] = min( abs( abs(l1) - abs(extentA(2)) ) );
% [ nul idx3 ] = min( abs( abs(l1) - abs(extentA(2)) ) );
% [ nul idx4 ] = min( abs( abs(l1) - abs(extentA(2)) ) );
%
clear good_profile
% since ll2m([13 13],[-44 -44.001845955]) == 200.0000
% use spacing increment of 0.001845955
%%% Old way
% good_profile(:,2) = linspace(extentA(1),extentB(1),1000);
% good_profile(:,1) = linspace(extentA(2),extentB(2),1000);
% BotProf(:,2) = linspace(extBotA(1),extBotB(1),1000);
% BotProf(:,1) = linspace(extBotA(2),extBotB(2),1000);
% TopProf(:,2) = linspace(extTopA(1),extTopB(1),1000);
% TopProf(:,1) = linspace(extTopA(2),extTopB(2),1000);
try good_profile(:,2) = extentA(1):0.00001845955:extentB(1) ;
catch good_profile(:,2) = fliplr(extentB(1):0.00001845955:extentA(1) );
end
good_profile(:,1) = linspace(extentA(2),extentB(2),length(good_profile(:,2)));

try BotProf(:,2) = extBotA(1):0.00001845955:extBotB(1);
catch BotProf(:,2) = fliplr(extBotB(1):0.00001845955:extBotA(1));
BotProf(:,1) = linspace(extBotA(2),extBotB(2),length(BotProf(:,2)));
end
try TopProf(:,2) = extTopA(1):0.00001845955:extTopB(1);
catch TopProf(:,2) = fliplr(extTopB(1):0.00001845955:extTopA(1));
TopProf(:,1) = linspace(extTopA(2),extTopB(2),length(TopProf(:,2)));
end
```

```
% check if all inside for profile
l2lim = [l2(1) l2(end)];
l1lim = [l1(1) l1(end)];
good_profile = lineinside(l1lim,l2lim,good_profile);
BotProf = lineinside(l1lim,l2lim,BotProf);
TopProf = lineinside(l1lim,l2lim,TopProf);


% set output
AC = [ax1(axidx) ax2(axidx)];
ACB = [ax1(axidxB) ax2(axidxB)];
ACT = [ax1(axidxT) ax2(axidxT)];
% RBFcrossing = [RBF1(axidx) RBF2(axidx)];
% LBFcrossing = [LBF1(axidx) LBF2(axidx)];
gp1 = good_profile(:,1);
gp2 = good_profile(:,2);
% test
% findbestprofilelength(axis_interp,slopeE_Wcc,lat,long,469)
```

6.5 directional_fftfiltermap.m

```
function [FILTMAP]=
directional_fftfiltermap(MAP,LAT,LONG,CUTOFFX,GRADATIONX,CUTOFFY,
GRADATIONY)
[numrows numcolumns]=size(MAP);
% if numrows or numcolumns not divisible by 2, fix it
oldrows = numrows;
oldcolumns = numcolumns;
if mod(numrows,2)
   numrows = numrows+1;
end
if mod(numcolumns,2)
   numcolumns = numcolumns+1;
end

% find distance covered in each dimension
longx=ll2m([LAT(1) LAT(1)],[LONG(1) LONG(end)])*1e-3;
longy=ll2m([LAT(1) LAT(end)],[LONG(1) LONG(1)])*1e-3;

% create vectors for plotting spectrum
X = -[-longy:longy/numrows*2:0];
Y = -[-longx:longx/numcolumns*2:0];

% convert cutoffs (in wavelengths duh) into frequencies
```

```
WHY=1/CUTOFFY;          %smaller cut-off frequency (1/wavelength in Km) e. g.
0.01, i.e. 1/100 Km
SHY=1/GRADATIONY;          %greater cut-off frequency (1/wavelength in Km) e.
g. 0.012, i.e. 1/83.3 Km
WHX=1/CUTOFFX;
SHX=1/GRADATIONX;

% throw map into fft in Y
fftMAPY = fft(MAP,numrows,1);
FILTMAPY = fft_me(fftMAPY,numrows,longy,WHY,SHY,oldrows,oldcolumns);
% Throw this result into fft for X
fftMAPYX = fft(FILTMAPY,numcolumns,2);
FILTMAPYX =
fft_me(fftMAPYX,numcolumns,longx,WHX,SHX,oldrows,oldcolumns);

% Now do the reverse
fftMAPX = fft(MAP,numcolumns,2);
FILTMAPX =
fft_me(fftMAPX,numcolumns,longx,WHX,SHX,oldrows,oldcolumns);
% Throw this result into fft for X
fftMAPXY = fft(FILTMAPX,numrows,1);
FILTMAPXY =
fft_me(fftMAPXY,numrows,longy,WHY,SHY,oldrows,oldcolumns);

sum(sum(abs(FILTMAPXY-FILTMAPYX)));
sum(sum(abs(FILTMAPXY)));
sum(sum(abs(FILTMAPYX)));
sum(sum(abs(FILTMAPX)));
sum(sum(abs(FILTMAPY)));
FILTMAP = FILTMAPXY;



% Plot The Things
figure
clf
subplot(2,2,1)
surf(LONG,LAT,FILTMAPYX)   %and the spectrum is drawn only for visualization
view([0 90]);shading interp;colorbar;axis equal
lightangle(-90,1e-3)
lightangle(0,1e-3)
xlim([LONG(1) LONG(end)])
ylim([LAT(1) LAT(end)])
title('Y then X Combine Filtered Map') ...Bathymetry map') %this is the title of the
new graph
```

```
subplot(2,2,2)
surf(LONG,LAT,FILTMAPY)
view([0 90]);shading interp;colorbar;axis equal
lightangle(-90,1e-3)
lightangle(0,1e-3)
xlim([LONG(1) LONG(end)])
ylim([LAT(1) LAT(end)])
title(sprintf('Map filtered in Y- direction only\n cosine filtered Cut: %.1f
%.1f',CUTOFFY,GRADATIONY))

subplot(2,2,4)
surf(LONG,LAT,FILTMAPX)
view([0 90]);shading interp;colorbar;axis equal
lightangle(-90,1e-3)
lightangle(0,1e-3)
xlim([LONG(1) LONG(end)])
ylim([LAT(1) LAT(end)])
title(sprintf('Map filtered in X-direction only\n cosine filtered Cut: %.1f
%.1f',CUTOFFX,GRADATIONX)) %this is the title of the new graph

subplot(2,2,3)
surf(LONG,LAT,FILTMAPXY)   %and the spectrum is drawn only for visualization
view([0 90]);shading interp;colorbar;axis equal
lightangle(-90,1e-3)
lightangle(0,1e-3)
xlim([LONG(1) LONG(end)])
ylim([LAT(1) LAT(end)])
title('X then Y Combine Filtered Map')

return

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% SUBFUNCTION
%%%% The real fft filtering goes on here.
%%
function FILTeMAP =
fft_me(FFTMAP,NUM_CR,Tlength,HighCut,LowCut,oldrows,oldcolumns)

for fg=1:((NUM_CR/2)+1);
   frequency(fg)=sqrt(((fg-1)/Tlength)^2);
end

%the matrix of the negative frequencies is also computed
```

```
frequency2=fliplr(frequency);
frequencytotal=[frequency frequency2];
frequencytotal(1)=[];
frequencytotal(end)=[];

frequencytotalplot=frequencytotal.*(2*pi);  %the frequency (1/wavelength) matrix is
transformed to wavenumber (2*pi/wavelength) matrix
% and now filtered
FILTER=zeros(size(FFTMAP));     %the filter matrix is set to zero
for f=1:size(FFTMAP,1);
  for g=1:size(FFTMAP,2);
      if  length(frequencytotal) == size(FFTMAP,2)
         fg = g;
         DIRECTION = 2;
      else
         fg = f;
         DIRECTION = 1;
      end
    if frequencytotal(fg)<HighCut
    FILTER(f,g)=1;
    elseif frequencytotal(fg)<LowCut
    FILTER(f,g)=.5.*(1+cos((((2*pi)*frequencytotal(fg))-
(2*pi*HighCut))/(2*(LowCut-HighCut))));
    else
    FILTER(f,g)=0;
    end
  end;
end;

% finally apply the filter, and calculate the ifft
FILTeMAP = -abs(ifft(FILTER.*FFTMAP,NUM_CR,DIRECTION));
FILTeMAP = FILTeMAP(1:oldrows,1:oldcolumns);
return


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%% OLD WAY
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%
%%% Now fileiter in Xx
% %the matrix with the frequencies of every harmonic is computed
% clear frequencyX frequencyX2 frequencytotalplotX filterX
% for g=1:((numcolumns/2)+1);
%    frequencyX(g)=sqrt(((g-1)/longx)^2);
% end
```

```
%
%
% %the matrix of the negative frequencies is also computed
% frequencyX2=fliplr(frequencyX);
% frequencytotalX=[frequencyX frequencyX2];
% frequencytotalX(1)=[];
% frequencytotalX(end)=[];
%
% frequencytotalplotX=frequencytotalX.*(2*pi);  %the frequency (1/wavelength)
matrix is transformed to wavenumber (2*pi/wavelength) matrix
% % and now filtered
% filterX=zeros(size(fftMAPX));      %the filter matrix is set to zero
% for f=1:size(fftMAPX,1);
%    for g=1:size(fftMAPX,2);
%       if frequencytotalX(g)<WHX
%       filterX(f,g)=1;
%       elseif frequencytotalX(g)<SHX
%       filterX(f,g)=.5.*(1+cos((((2*pi)*frequencytotalX(g))-(2*pi*WHX))/(2*(SHX-
WHX))));
%       else
%       filterX(f,g)=0;
%       end
%    end;
% end;
%
% % finally apply the filter, and calculate the ifft
% FILTMAPX = -abs(ifft(filterX.*fftMAPX,numcolumns,2));
% FILTMAPX = FILTMAPX(1:oldrows,1:oldcolumns);
% figure(2)
% clf
% surf(FILTMAPX)
% axis equal
% lightangle(-90,1e-3)
% view(0,90);shading interp;colorbar
% title('X filter')
%%%%%%%%%% IN Y
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%
%% Filitere in Y s
% %a vector with the frequencies of every harmonic is computed
% for f=1:((numrows/2)+1);
%    frequencyY(f)=sqrt(((f-1)/longy)^2);
% end
%
%
```

```
% %the matrix of the negative frequencies is also computed
% frequencyY2=fliplr(frequencyY);
% frequencytotalY=[frequencyY frequencyY2];
% frequencytotalY(1)=[];
% frequencytotalY(end)=[];
% frequencytotalplotY=frequencytotalY.*(2*pi);  %the frequency (1/wavelength)
matrix is transformed to wavenumber (2*pi/wavelength) matrix
% % and now filtered
% filterY=zeros(size(fftMAPY));      %the filter matrix is set to zero
% for f=1:size(fftMAPY,1)
%    for g=1:size(fftMAPY,2);
%       if frequencytotalY(f)<WHY
%       filterY(f,g)=1;
%       elseif frequencytotalY(f)<SHY
%       filterY(f,g)=0.5.*(1+cos((((2*pi)*frequencytotalY(f))-(2*pi*WHY))/(2*(SHY-
WHY))));
%       else
%       filterY(f,g)=0;
%       end
%    end;
% end;
% % finally apply the filter, and calculate the ifft
% FILTMAPY = -abs(ifft(fftMAPY.*filterY,numrows,1));
% FILTMAPY = FILTMAPY(1:oldrows,1:oldcolumns);
% figure(1)
% clf
% surf(FILTMAPY)
% axis equal
% lightangle(0,1e-2)
% view(0,90);shading interp;colorbar
% title('Y filter')
```

6.6 dofaultWinfill.m

```
function [nx,yt,ym,ndx]=dofaultWinfill(an,fh,lp,dx,ct,ift)

% [nx,yt,ym,ndx]=dofault(an,fh,lp,dx,ct)
%
%%% INPUT
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% an - fault angle in degrees
% fh - fault horizontal displacement in m
% lp - length of profile in m
% dx - Min dx (the array is recalculated for power-of-two length)
% ct - crustal thickness in m
% ift - infill thickness in m
```

```
%%% OUTPUT
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% nx - new x array
% yt - seafloor topography
% ym - mantle topography
% ndx - spacing of new x-array

%%% calculate some scalars
try     dan=deg2rad(an); % degree to radians
catch   dan=an*pi/180;
end
% fh is horizontal distance of fault, and so the vertical uplife is
fv=fh*tan(dan); % Vertical uplift of the fault
%
%  ^  |       /
%  |  |     t/
%  |  |    l/
% fv  |   u/
%  |  |  a/
%  |  | f/
%  v  | /
%     ---------
%     <-- fh -->

ch=ct/tan(dan); % Horizontal distance of contact of Moho w/ fault w/respect to
surface fault break
%
%           /| ^
%         t/ | |
%        l/  | ct
%       u/   | |
%      a/    | v
%     f/<-ch->
%     /
%
%
inh=ift/tan(dan);
%
%              /
%       <-inh->/t
%       ^     /l
%       |    /u
%      ift  /a
%       |  /f
%       \//
```

```
%  -------'
%
%%
%%% dx is spacing, profile length is symetrical about x=0
x=-lp/2:dx:lp/2; % x array
% if lenght of array is not a power of 2, redo it
np2=power2(length(x)); % calculate next power of two length
nx=linspace(min(x),max(x),np2); % recalculate x for 2^n

%%%% make a y-array that is the same length as 'x'
% this is the topography of the crust
yt=zeros(size(nx)); % initialize y variables
% offsetting the crust topography by the crustal thickness gives mantle
% topography
ym=yt-ct;          % shift them

%% now we find the actual intial conditions for seafloor topography
% first find all x values beyond the fault heave
itop=find(nx>=fh/2);
% and offset them to be equal to fault vertical dispalcemnet
yt(itop)=yt(itop)+fv;
% then find the values that are on the fault surface
ifau=find(nx>-fh/2 & nx<fh/2);
% and find their y-values according to trigonometry
yt(ifau)=[nx(ifau)-min(nx(ifau))]*tan(dan);
% now find flat topo
% iinf = find(nx<=-(fh/2 - inh); % this method is after Schouten,
iinf = find(yt<=ift);   % but this method is better
yt(iinf) = ift;

%%% Schematic of resultant topography
%
%                      <--- ifau ----> <------- itop ------->
%                                  ,----------------------
%                                 /
%                                t/
%                               l/
%                              u/
%                             a/
%                            f/
%                           /
%                          /:
%                         / :
%                        /  :
%                       /   :
%   ----------------------  /    :
```

```
%           ^           /    :
%          ift         /     :
%   ----------------------'      :
%                              :
%                             x=0


%% now do for moho topography
% we find the x's beyond the fault, which for the mantle is the fh/2, minus
% the horizontal distance we know from the crust
itop=find(nx>=fh/2-ch);
% offset this by the fault vertical displacement
ym(itop)=ym(itop)+fv;
% go find the points that lie on the fault
ifau=find(nx>-fh/2-ch & nx<fh/2-ch);
% offset them by the trigonometry
ym(ifau)=[nx(ifau)-min(nx(ifau))]*tan(dan)-ct;
% %
% % ym(ifau(1):ifau(100)) = ym(ifau(1));

%%% both topographies now
%                                  ,----------------------
%                                 /  crust
%                      <-- ifau --> /------------------------
%                                 /
%                                /
%     sea water                   /
%                               /    mantle
%                              /
%                             /:
%                            / :
%                           /  :
%                          /   :
%                         /    :
%                        /     :
%                       /      :
%   ----------------------'      :
%     crust            /       :
%   --------------------'        :
%                              :
%                             x=0

%%% last is to find the new spacing after the power of 2 thing
% simply take diff of two ajacent x's
ndx=nx(2)-nx(1);
```

6.7 errorCalc.m

```
function [R_ChiSqX, R_ChiSqTopo] =
errorCalc(fittingtopo,fittingtopodist,fittingrot,fittingdist,fulldist,fulltopo,heaves,TES,t
es,angles,infill,crusts,offdistX,accept_he,accept_of)
%,  R_sqrX, R_sqrY, SumResSqr
% find deviations from averages, which won't change for each R^2 calculation
y_barX = (1/length(fittingrot))*sum(fittingrot);
y_barY = (1/length(fittingdist))*sum(fittingdist);
TotSqX= (fittingrot - y_barX).^2;
TotSqY= (fittingdist - y_barY).^2;
%% initialize residual matrices;
clear R_sqrX
R_sqrX(1:length(heaves),1:length(angles),1:length(infill),1:length(crusts),1:length(of
fdistX), 1:length(tes)) = 1e8;
SumResSqr = R_sqrX;
R_sqrY = R_sqrX;
R_ChiSqX = R_sqrX;
R_ChiSqTopo = R_sqrX;
clear ResSqX ChiSqX res_sqr ResSqY ChiSqTopo
ChiSqTopo = zeros(1,length(fittingtopodist));
ChiSqX = zeros(1,length(fittingdist));
% display error value, this is for chi^2 test
% error = 5; % degrees for slope. this is likely very conservative
errorS = 4;
errorT = 100;
disp(sprintf('Slope Error is %g^o',errorS))
disp(sprintf('Topo Error is %g m',errorT))
tic
aheid = 1;
aofid = 1;
for he = 1:length(heaves) % prev 'te'
   %if aheid <= length(accept_he)
   %if accept_he(aheid) == he
  % aheid = aheid+1;
   disp(sprintf('he = %g',heaves(he)))
     figure(100002);clf
   for teidx =  1:length(tes) %(size(te(he).slope,2))

%       disp(sprintf('/tte = %g',tes(teidx)))
      for ANid = 1:length(angles)
%          disp(sprintf('/tte = %g',angles(ANid)))
         for IFid=1:length(infill)
            for CTid=1:length(crusts)
               for offdistidx = 1:length(offdistX)
```

```
                % if aofid <= length(accept_of)
                % if accept_of(aofid) == offdistidx
                % aofid = aofid+1;
% % %                     disp(sprintf('Offset = %g',offdistX(offdistidx)))
                % offset our distance vector
                % testtedist = te(he).dist + offdistX(offdistidx);
                testtedistTOPO = TES(he,ANid,IFid,CTid).topodist(:,teidx) +
offdistX(offdistidx);%linspace(0,TES(he,ANid,IFid,CTid).dist(end,teidx),length(TES
(he,ANid,IFid,CTid).topo)) + offdistX(offdistidx);
                testtedist =  TES(he,ANid,IFid,CTid).dist(:,teidx) +
offdistX(offdistidx);
                %theseslopes = te(he).slope(:,teidx);
                theseslopes =  TES(he,ANid,IFid,CTid).slope(:,teidx);
                % we need to fix our topo to the elevation of the
                % first point, which is either the fulltopo 1 or
                % end, depending on if on left or right side.
               %  thesetopos =  TES(he,ANid,IFid,CTid).topo(:,teidx) - ...
               %       abs(min(TES(he,ANid,IFid,CTid).topo(:,teidx)) +  abs(min([
fulltopo(1) fulltopo(end)])));
                % to fix according to the chosen pts minimizatoin...
                    [val, idx] = min(abs( testtedistTOPO - min(fittingdist) )); %(2:end-
1)
                thesetopos =  TES(he,ANid,IFid,CTid).topo(:,teidx) - ...                %
2           end-1
                   abs(min(TES(he,ANid,IFid,CTid).topo(idx,teidx)) + abs(min([
fittingtopo(1) fittingtopo(end)])));

                %% now find residual for each point in slope space
                for jkkk = 1:length(fittingdist)
                   % first find the nearest point to the point,
                   [val, idx] = min(abs( testtedist - fittingdist(jkkk)));

                   %minslope = te(he).slope(idx,teidx);
                   minslope = TES(he,ANid,IFid,CTid).slope(idx,teidx);


% % %                     ResSqX(jkkk) = (fittingrot(jkkk) - minslope)^2;
                   ChiSqX(jkkk) = ( ( fittingrot(jkkk) - minslope ) / errorS)^2;
% % %                     res_sqr(jkkk) = (fittingrot(jkkk) - minslope)^2;
% % %                     [val, idx] = min(abs(theseslopes  - fittingrot(jkkk)));
% % %                     ResSqY(jkkk) = (fittingdist(jkkk) - testtedist(idx))^2;
                end
                %sum the residuals
                %R_sqrX(teidx,offdistidx,he) = 1 - sum(ResSqX)/sum(TotSqX);
                %SumResSqr(teidx,offdistidx,he) = sum(res_sqr)/length(fittingdist);
                %R_sqrY(teidx,offdistidx,he) = 1 - sum(ResSqY)/sum(TotSqY);
```

```
                %R_ChiSqX(teidx,offdistidx,he) = sum(ChiSqX);
                % R_sqrX(he,ANid,IFid,CTid,offdistidx,teidx) = 1 -
sum(ResSqX)/sum(TotSqX);
                % SumResSqr(he,ANid,IFid,CTid,offdistidx,teidx) =
sum(res_sqr)/length(fittingdist);
% % %                   R_sqrY(he,ANid,IFid,CTid,offdistidx,teidx) = 1 -
sum(ResSqY)/sum(TotSqY);
                R_ChiSqX(he,ANid,IFid,CTid,offdistidx,teidx) = sum(ChiSqX);
                ChiSqX = ChiSqX*0;
%plot for checking

%     plot(fulldist,fulltopo*2e-3,'m')
%     plot(testtedistTOPO,...
%       thesetopos*2e-3,'b')
%        hold on
%     plot(fittingtopodist,fittingtopo*2e-3,'r')


                %% find residual for each point in topography space
                for jkkk = 1:length(fittingtopodist)
                   [valT, idxT] = min(abs( testtedistTOPO - fittingtopodist(jkkk)));
                   ChiSqTopo(jkkk) = ( ( fittingtopo(jkkk) - thesetopos(idxT) ) /
errorT)^2;
%   plot(testtedistTOPO(idxT),thesetopos(idxT)*2e-3,'b*')
%   plot(fittingtopodist(jkkk),fittingtopo(jkkk)*2e-3,'r*')

                end
%   drawnow
%   disp(sprintf('For offset: %g , misfit is:
%g',offdistX(offdistidx),sum(ChiSqTopo)))
%   pause(1)
                R_ChiSqTopo(he,ANid,IFid,CTid,offdistidx,teidx) =
sum(ChiSqTopo);
                ChiSqTopo = ChiSqTopo*0;

              %  end
              %  end
             end
          end
        end
      end
   end
   %end
   %end
end
toc
```

6.8 errorcalcandplot.m

```
function Modeled =
errorcalcandplot(fetnum,TES,tes,heaves,crusts,infill,angles,offdistX,ProPick,figstring
)

%% upack the structure
% and assign values
[ val axidx ] =min(abs(ProPick.profdist));
axDepth = ProPick.zs(axidx);
mHeave = ProPick.Bfcc.Heave*1e-3;
idx1 = ProPick.Ffcc.idx1;
idx2 = ProPick.Ffcc.idx2;
try
   fittingrot = ProPick.fittingrot;% [  ProPick.fittingrot 0 ];
catch
   fittingrot =  ProPick.fittingrot';% [  ProPick.fittingrot' 0 ];
end

try
   fittingdist = abs(ProPick.fittingdist)';%[   abs(ProPick.fittingdist)' 0 ];
catch
   fittingdist =abs(ProPick.fittingdist);% [   abs(ProPick.fittingdist) 0 ];
end


try                                          % attempt to make axis depth matter
   fittingtopodist = abs(ProPick.fittingdist)' ; % [ abs(ProPick.fittingdist)' 0 ] ;
catch
   fittingtopodist = abs(ProPick.fittingdist); % [ abs(ProPick.fittingdist) 0 ] ;
end

try
   fittingtopo= ProPick.fittingtopo';%[ ProPick.fittingtopo' axDepth] ;%
ProPick.zs(idx1:idx2);
catch
   fittingtopo=  ProPick.fittingtopo;%[ ProPick.fittingtopo axDepth] ;%
ProPick.zs(idx1:idx2);
end

fulldist=abs(ProPick.profdist(idx1:idx2)*1e-3);
fulltopo=ProPick.zs(idx1:idx2);
```

```
%% keep heave and offset constant
if 0
   % do for heave
   % if only considering measured heave
   Acceptable_heaves =  round(mHeave)*1e3;
   % for a small subset of heaves use this
   %Acceptable_heaves = round(mHeave)*1e3-500:500:round(mHeave)*1e3+500
   for he = 1:length(Acceptable_heaves)
      [v t] = (min(abs(Acceptable_heaves(he) - heaves)));
      accept_he(he) = t;
   end
   heaves = Acceptable_heaves;
   % do for Offset
   % measured offset
    Acceptable_offdistX =   round((min(fulldist) - 3.6)/.25)*.25;
    %%%%%                        fulldist(10) for synthetic
   % small subset of offsets
   %Acceptable_offdistX = Acceptable_offdistX-1:.25:Acceptable_offdistX+1;
   for of = 1:length(Acceptable_offdistX)
      [v t] = (min(abs(Acceptable_offdistX(of) - offdistX)));
      accept_of(of) = t;
   end
   offdistX = Acceptable_offdistX;
   % reset the TES
   TES = TES(accept_he,:,:,:);

   % assign these to the model, cause now they unique
   Modeled.offdistX = Acceptable_offdistX;
   Modeled.heaves = Acceptable_heaves;
   Modeled.crusts = crusts;
   Modeled.tes = tes;
   Modeled.infill = infill;
   Modeled.angles = angles;
else
   accept_he = 1:length(heaves);
   accept_of = 1:length(offdistX);
end
%% for synthetic
if 0
mHeave = heaves(end);
tesidx = 6;
if tesidx == 1
   fittingdist = [TES(end,4,3,2).dist(1:50:500,tesidx)'
TES(end,4,3,2).dist(500:200:1500,tesidx)'
TES(end,4,3,2).dist(2000:2000:24000,tesidx)'
TES(end,4,3,2).dist(24500:200:26000,tesidx)'
```

```
TES(end,4,3,2).dist(26000:50:26700,tesidx)'
TES(end,4,3,2).dist(26700:200:27500,tesidx)'
TES(end,4,3,2).dist(28000:1000:end,tesidx)'];
   fittingrot = [TES(end,4,3,2).slope(1:50:500,tesidx)'
TES(end,4,3,2).slope(500:200:1500,tesidx)'
TES(end,4,3,2).slope(2000:2000:24000,tesidx)'
TES(end,4,3,2).slope(24500:200:26000,tesidx)'
TES(end,4,3,2).slope(26000:50:26700,tesidx)'
TES(end,4,3,2).slope(26700:200:27500,tesidx)'
TES(end,4,3,2).slope(28000:1000:end,tesidx)'];
elseif tesidx == 6
   fulltopo = [TES(end,4,3,2).topo(4542-250*10:250:4541+5000,tesidx)'
TES(end,4,3,2).topo(4541+5500:500:4541+22000,tesidx)'
TES(end,4,3,2).topo(4541+23000:500:end,tesidx)' ];
   fulltopodist =  [TES(end,4,3,2).topodist(4542-250*10:250:4541+5000,tesidx)'
TES(end,4,3,2).topodist(4541+5500:500:4541+22000,tesidx)'
TES(end,4,3,2).topodist(4541+23000:500:end,tesidx)' ];

   fulldist =  [TES(end,4,3,2).dist(4542-250*10:250:4541+5000,tesidx)'
TES(end,4,3,2).dist(4541+5500:500:4541+22000,tesidx)'
TES(end,4,3,2).dist(4541+23000:500:end,tesidx)' ];
   fullrot =   [TES(end,4,3,2).slope(4542-250*10:250:4541+5000,tesidx)'
TES(end,4,3,2).slope(4541+5500:500:4541+22000,tesidx)'
TES(end,4,3,2).slope(4541+23000:500:end,tesidx)' ];
end
   fittingdist = fulldist(12:2:25);
   fittingrot = fullrot(12:2:25);% + randn(10,1)'*5;


   fulltopoBU= fulltopo;
   fulltopo(26:end) = fulltopo(26:end) + randn(length(fulltopo(26:end)),1)'*150;
   fittingtopo = fulltopo(12:2:25);
   fittingtopodist = fulltopodist(12:2:25);

plot(fulltopodist,fulltopoBU,'b');hold on
plot(fulltopodist,fulltopo,'r')
end

%%   Big if
if 1
%% misfit calculation
% Now we have our datas distances, and rotations
% loop through a bunch of offsets, and Te curves, find minimum R^2

if 0%fetnum>10
```

```
[R_ChiSqX, R_ChiSqTopo] =
errorCalc(fittingtopo,fittingtopodist,fittingrot,fittingdist,fulldist,fulltopo,heaves,TES,t
es,angles,infill,crusts,offdistX,accept_he,accept_of);
% , R_sqrX, R_sqrY, SumResSqr
end

%%
if 1%fetnum<11
%Modeled = ProPick.ModelwCTApr1;
% Modeled = ProPick.FixAbsHe_smallErr_Apr27;
%R_ChiSqX = ProPick.Model.R_ChiSqx((4/5)^2);
%R_ChiSqX = ProPick.ModelwCTApr1.R_ChiSqx;
%R_ChiSqX = ProPick.FixAbsHe_smallErr_Apr27.R_ChiSqx;
% R_ChiSqTopo = ProPick.Model.R_ChiSqTopo((100/75)^2);
% R_ChiSqTopo = ProPick.ModelwCTApr1.R_ChiSqTopo;
% R_ChiSqTopo = ProPick.FixAbsHe_smallErr_Apr27.R_ChiSqTopo;


Modeled = ProPick.FullGrid_May4;
R_ChiSqX = ProPick.FullGrid_May4.R_ChiSqx;
R_ChiSqTopo = ProPick.FullGrid_May4.R_ChiSqTopo;

end
%% find minimum
%misfitfct = SumResSqr;
%msftstr = 'S/n'; % S/n (Sum Squared Residuals / n)
% % longprofmisfitfct;
% % shortprofmisfitfct;
% % shortprofmisfitfctwNoise;
% % R_ChiSqX = R_ChiSqTopo;
Modeled.R_ChiSqx = R_ChiSqX;
Modeled.R_ChiSqTopo = R_ChiSqTopo;

[bestfitmistfit, idx] = min(R_ChiSqTopo(:)); % max for some misfits
[HEidxT,ANidxT,IFidxT,CTidxT,offdistidxT,teidxT] = ind2sub(size(R_ChiSqTopo),
idx);
Modeled.TopoRawminfitidx = [HEidxT,ANidxT,IFidxT,CTidxT,offdistidxT,teidxT];
minfitidxT=[HEidxT,ANidxT,IFidxT,CTidxT,offdistidxT,teidxT];


[bestfitmistfit, idx] = min(R_ChiSqX(:)); % max for some misfits
[HEidx,ANidx,IFidx,CTidx,offdistidx,teidx] = ind2sub(size(R_ChiSqX), idx);
Modeled.SlopeRawminfitidx = [HEidx,ANidx,IFidx,CTidx,offdistidx,teidx];
minfitidx=[HEidx,ANidx,IFidx,CTidx,offdistidx,teidx];
```

```
[HEsize ANsize IFsize CTsize ODsize TEsize] = size(R_ChiSqX);

%% Now do the wieght function
if 1%fetnum>10
[Modeled.maxTw , Modeled.minTw ,Modeled.Weightedminfitidx] =
misfitweights(Modeled);
end
meanTw = (Modeled.maxTw + Modeled.minTw)/2 ;
minfitidxW = Modeled.Weightedminfitidx;
HEidxW = Modeled.Weightedminfitidx(1);
ANidxW = Modeled.Weightedminfitidx(2);
IFidxW = Modeled.Weightedminfitidx(3);
CTidxW = Modeled.Weightedminfitidx(4);
offdistidxW = Modeled.Weightedminfitidx(5);
teidxW = Modeled.Weightedminfitidx(6);

% keep the ids of the variables that are not constant
if TEsize == 1; minfitidxT(6) = [];minfitidxW(6) = [];minfitidx(6) = [];end
if ODsize == 1; minfitidxT(5) = [];minfitidxW(5) = [];minfitidx(5) = [];end
if CTsize == 1; minfitidxT(4) = [];minfitidxW(4) = [];minfitidx(4) = [];end
if IFsize == 1; minfitidxT(3) = [];minfitidxW(3) = [];minfitidx(3) = [];end
if ANsize == 1; minfitidxT(2) = [];minfitidxW(2) = [];minfitidx(2) = [];end
if HEsize == 1; minfitidxT(1) = [];minfitidxW(1) = [];minfitidx(1) = [];end
%%
if 1%fetnum>10
%% this is where figi2 also comes from
[numDF ys modelfitsbysigma] = plotErrorSurface(R_ChiSqX,    minfitidx,
tes,heaves,angles,infill,crusts,offdistX,HEidx,ANidx,IFidx,CTidx,offdistidx,teidx,spri
ntf('%g Slope',fetnum),figstring);
%%
[numDF ys modelfitsbysigmaT] =
plotErrorSurface(R_ChiSqTopo,minfitidxT,tes,heaves,angles,infill,crusts,offdistX,HE
idxT,ANidxT,IFidxT,CTidxT,offdistidxT,teidxT,sprintf('%g Topo',fetnum),figstring);

%%
[numDF ys modelfitsbysigmaW] = plotErrorSurface((R_ChiSqTopo*meanTw +
R_ChiSqX*(1-
meanTw)),minfitidxW,tes,heaves,angles,infill,crusts,offdistX,HEidxW,ANidxW,IFid
xW,CTidxW,offdistidxW,teidxW,sprintf('%g Weighted by
%.0f',fetnum,meanTw*1e3),figstring);
end
end

%% find plotting stuff
desirespacing=200;
axisdist = 15;
```

```
Nkminc = 1;
% these for the bf dist vs outward rotation
howfar = 3;
BFNkminc = 1;
if isfield(ProPick.Ffcc,'idx1')
   idx1 = ProPick.Ffcc.idx1;
   idx2 = ProPick.Ffcc.idx2;
else
   disp('Pick the left side of the Footwall')
   [idx1 val]=ginput(1);
   [val idx1]=min(abs(ProPick.profdist-idx1));
   disp('Pick the right side of the Footwall')
   [idx2 val]=ginput(1);
   [val idx2]=min(abs(ProPick.profdist-idx2));
   ProPick.Ffcc.idx1 = idx1;
   ProPick.Ffcc.idx2 = idx2;
end

% find new spacing for this profile
for hk = 1:abs(idx1-idx2)
   fidminc(hk) = diff([ProPick.profdist(idx1) ProPick.profdist(idx1+hk)]);
end
[val idxhk] = min(abs(fidminc-desirespacing));
Nkminc =  idxhk;
fidminc = fidminc(idxhk);
%fidkminc = diff([ProPick.profdist(idx1) ProPick.profdist(idx1+Nkminc)]);

% subplot(2,2,spp2)
% CCslopevsTe(ProPick.Bfcc.Heave(fid1)*1e3,abs(ProPick.WBFdist)-3)
% hold on

ProPick.Ffcc.kminc = fidminc;
% we have to recalculate dzdx, using this new spacing,
% and a resampling as given in input
% multiply by 100 to get toi degrees from slope
onEast = sign(ProPick.profdist(ProPick.bsidx1));
%%
if isfield(ProPick,'Sdzdx')
if onEast
   disp('It'' on the east')
   ProPick.Ffcc.OutwardRotation =
ProPick.Sdzdx(idx1:Nkminc:idx2);%gradient(ProPick.zs(idx1:Nkminc:idx2),fidminc
)*100;
   OutwardRotations = ProPick.Ffcc.OutwardRotation;
   Distances = ProPick.profdist([idx1:Nkminc:idx2]);
   Depths = ProPick.zs([idx1:Nkminc:idx2]);
```

```
   xlimit1 = ProPick.profdist(idx1);
   xlimit2 = 1;
else
   disp('It'' on the west')
   ProPick.Bfcc.OutwardRotation =
ProPick.Sdzdx(idx1:Nkminc:idx2);%gradient(ProPick.zs(idx1:Nkminc:idx2),fidminc
)*100;
   OutwardRotations = fliplr(ProPick.Bfcc.OutwardRotation);
   Distances = fliplr(ProPick.profdist([idx1:Nkminc:idx2]));
   Depths = fliplr(ProPick.zs([idx1:Nkminc:idx2])');
   xlimit1 = -1;
   xlimit2 = ProPick.profdist(idx2);
end
ProPick.Distances = Distances;
ProPick.OutwardRotations = OutwardRotations;
%%
else
Distances = ProPick.Distances;
OutwardRotations = ProPick.OutwardRotations*-1;
Depths = fulltopo;
end
countme=1;
PrevRot = 45;
colorN = jet(length(Distances));
PrevDep = -1e100;


fet=fetnum;

printme =1;
doAll = 1;
plotBestfitModel_onSlope;
```

6.9 fftfiltermap.m

```
function [FILTMAP]=fftfiltermap(MAP,LAT,LONG,CUTOFF,GRADATION)
[numrows numcolumns]=size(MAP);
% if numrows or numcolumns not divisible by 2, fix it
oldrows = numrows;
oldcolumns = numcolumns;
if mod(numrows,2)
   numrows = numrows+1;
end
if mod(numcolumns,2)
   numcolumns = numcolumns+1;
end
```

```
longx=ll2m([LAT(1) LAT(1)],[LONG(1) LONG(end)])*1e-3;
longy=ll2m([LAT(1) LAT(end)],[LONG(1) LONG(1)])*1e-3;

% create vectors for plotting spectrum

X = -[-longy:longy/numrows*2:0];
Y = -[-longx:longx/numcolumns*2:0];

WH=1/CUTOFF;          %smaller cut-off frequency (1/wavelength in Km) e. g. 0.01,
i.e. 1/100 Km
SH=1/GRADATION;          %greater cut-off frequency (1/wavelength in Km) e. g.
0.012, i.e. 1/83.3 Km

fftMAP=fft2(MAP,numrows,numcolumns);  %the 2-D FFT of the gravity input
matrix is computed after demeaning
bath_spectrum=abs(fftMAP);  %this computes the amplitude spectrum
%%
%the matrix with the frequencies of every harmonic is computed
for f=1:((numrows/2)+1);
  for g=1:((numcolumns/2)+1);
    frequency(f,g)=sqrt(((f-1)/longx)^2+((g-1)/longy)^2);
  end
end

%the matrix of the negative frequencies is also computed
frequency2=fliplr(frequency);
frequency3=flipud(frequency);
frequency4=fliplr(flipud(frequency));
entero=round(numcolumns/2);
if ((numcolumns/2) - entero)==0
  frequency2(:,1)=[];
  frequency3(1,:)=[];
  frequency4(:,1)=[];
  frequency4(1,:)=[];
  frequencytotal=[frequency frequency2;frequency3 frequency4];
else
  frequencytotal=[frequency frequency2;frequency3 frequency4];
end
frequencytotal(end,:)=[];
frequencytotal(:,end)=[];

frequencytotalplot=frequencytotal.*(2*pi);  %the frequency (1/wavelength) matrix is
transformed to wavenumber (2*pi/wavelength) matrix
%%
%The high-cut filter is constructed
```

```
filter=frequencytotal.*0;      %the filter matrix is set to zero
for f=1:numrows;
  for g=1:numcolumns;
    if frequencytotal(f,g)<WH
    filter(f,g)=1;
    elseif frequencytotal(f,g)<SH
    filter(f,g)=0.5.*(1+cos((((2*pi)*frequencytotal(f,g))-(2*pi*WH))/(2*(SH-WH))));
    else
    filter(f,g)=0;
    end
  end;
end;
% finally apply the filter, and calculate the ifft
FILTMAP = -abs(ifft2(fftMAP.*filter));
FILTMAP = FILTMAP(1:oldrows,1:oldcolumns);
%% Plot The Things
figure
clf
subplot(2,2,3)
surf(filter)
view([0 90]);shading interp;colorbar
title('Amplitude spectrum of the Filter') %this is the title of the new graph

subplot(2,2,1)
surf(log10(bath_spectrum(1:numrows/2, 1:numcolumns/2)))   %and the spectrum is
drawn only for visualization
view([0 90]);shading interp;colorbar
title('Amplitude spectrum of the Observed Bathymetry map') %this is the title of the
new graph

subplot(2,2,2)
surf(frequencytotalplot(1:numrows/2, 1:numcolumns/2))
view([0 90]);shading interp;colorbar
title('"Ideal" Amplitude Spectrum of the Region')

subplot(2,2,4)
imshade(LONG,LAT,FILTMAP)   %and the spectrum is drawn only for visualization
view([0 90]);shading interp;colorbar
title(sprintf('High Pass (>%.0f km) Filtered Map',GRADATION))
```

6.10 flex.m

```
function w=flex(ht,hc,dx,te)
% gives response to load in meters (- down)
% w=flex(ht,hc,dx,te)
```

```
%%% INPUT
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% ht=water-crust interface topo, centered at 0, m (array)
% hc=crust-mantle interface topo, centered at 0, m (array)
% dx=spacing in m
% te= Elastic thickness, in m
%%% OUTPUT
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% w - the response of the plate
% assumes:
% E=1e11;  Young's modulus
% v=.25 Poisson's ratio
% rho water     = 1.03
% rho crust     = 2.7
% rho mantle    = 3.3
% weissel & karner 1989, jgr 94 13919-13950

%E=Emult*1e10; % mlarson change, aug 20 to test different E
E=1e11;               % Youngs modulus
v=.25;               % Poisson's ratio
g=9.81;               % Gravity
rho_w=1030;           % water
rho_c=2700;           % crust
rho_m=3300;           % mantle density
rho_s=3000;           % serpentinite density

% eqn 3.115 from Turcotte & Schubert 3rd Ed.
% isostatic result for ht and hc
% this is units of elevation
st=ht*(rho_c-rho_w)/(rho_m-rho_w);
sc=hc*(rho_m-rho_c)/(rho_m-rho_w);
% these results look like this
%
%
%                    ,------------,
%                  ,             ,
%                 /               \
%               ,                   ,
%              /                     \
%             ,     ,------------,      ,
%            /    ,'              ',     \
% ------------' ,'                  ',  '------------ st
% -----------'                        '------------ sc
```

```
% now add and invert
s=-(sc+st);
% the result is a steeper slope than before
%
% ------------,                       ,------------ s
%             \                     /
%              \                   /
%               \                 /
%                \               /
%                 \             /
%                  \           /
%                   \         /
%                    '-------'

% find some wavelenghts, and their spectrum
[S,k]=jfft(s,dx);
% x = 100000*[1:length(k)]/length(k);
% E = E*(1+x/10);

% find D, a scalar, whic is the rigidity of the plate.
% from Turcotte & Schubert 3rd Ed. eqn. 3.72, also, Buck 1988
D=E.*te^3/(12*(1-v^2));

% find compensation, from Turcotte Schubert 3rd Ed. eqn 3.117
cte=(rho_m-rho_w)*(D/g.*k.^4+((rho_m-rho_w))).^(-1);
% this is a curve that starts at x=0, y = 1,
% and then goes to y=0 as x -> length profile
% curvature is changed with all the factors, but Te is the one we deal with
% -.
%   `
%   :
%   :
%   :
%   :
%    `-----------------------------------------------------------------
% 0

W=cte.*S';
% now time to do the inverse fft, and find the real part
% now we have our w
w=jifft(W);
```

6.11 InversionRoutine.m

% takes structure file from OCCFlexMain.m and does an inversion routine which identifies best fitting models and calculates an error for them. %% set up parameter vectors, and calculate our model structure

```
clear heaves offdistX  angles infill crusts tes

tes =[ 100:100:2000 ];

for heavers = 1:length(Profslopes.profile)
   heaves(heavers) = Profslopes.profile(heavers).pick.Bfcc.Heave;
end
heaves(length(heaves)+1:length(heaves)+3) = [ ...
   DANTESpick.Bfcc.Heave
   KANEpick.Bfcc.Heave
   TAGpick.Bfcc.Heave ];
heaves = sort(heaves);
heaves = unique(round(heaves*1e-2)*1e2);


crusts =[0 1e3 3e3 6e3];
infill = [ 0 2e3 6e3];
angles = [45:5:75];

%%%%% Variables for FULL GRID
% heaves = [2000:500:20000] ;%[1000:500:12000];%[ [1000:1000:10000]
[15000:5000:30000]];
% angles = [45:5:70];
% infill = [0 2e3];
% crusts =[0 1e3 3e3 6e3];
% offdistX =  -3:.25:3;
% tes =[ 100:100:2000 ];

% <45 is impossible because
% critical_angle = ( 90 + internal_friction_angle )/ 2
% >75 impossible because

%%%%%%%%%[HEidx,ANidx,IFidx,CTidx,offdistidx,teidx]%%%%%%%%%
% set up offset vector
offdistX =  -3:.25:3;
%%
if 1
clear TEmodels
TEmodels = TESmaker(heaves,angles,infill,crusts,6,tes)
end
%%
figstring = 'May5_Thesis_FixHe';%'Apr27_firstpt_err4_100_fixAbshe';

for n = 1:length(Profslopes.profile)
   ErrorStructwCrustThick =
errorcalcandplot(n,TEmodels,tes,heaves,crusts,infill,angles,offdistX,...
```

```
        Profslopes.profile(n).pick,figstring);
   %Profslopes.profile(n).pick.SOMETHING = ErrorStructwCrustThick;
end

%% for Dantes, Kane and Tag.

DANTESpick.figstring =   'Dantes_May4_firstpt_err4_100_FullGrid';
KANEpick.figstring =      'Kane_May4_firstpt_err4_100_FullGrid';
TAGpick.figstring =       'TAG_May4_firstpt_err4_100_FullGrid';
time = 1;
for pickpick = [ DANTESpick KANEpick TAGpick]
   ErrorStructwCrustThick =
errorcalcandplot(n+time,TEmodels,tes,heaves,crusts,infill,angles,offdistX,...
      pickpick,pickpick.figstring);
%    if strfind(pickpick.figstring,'TAG')
%       TAGpick.FixHe_May5 = ErrorStructwCrustThick;
%    elseif strfind(pickpick.figstring,'Dantes')
%       DANTESpick.FixHe_May5 = ErrorStructwCrustThick;
%    elseif strfind(pickpick.figstring,'Kane')
%       KANEpick.FixHe_May5 = ErrorStructwCrustThick;
%    end
   time = time+1;
end
% save('KANEpick_May6.mat','KANEpick')
% save('DANTESpick_May6.mat','DANTESpick')
% save('TAGpick_May6.mat','TAGpick')

%%
%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%

%%%%%%%%%%%%%%%%%%%%%%%%%%

%%%%%%%%%%%%%%%%%%%%%%%%%%

%%%%%%%%%%%%%%%%%%%%%%%%%%

%%%%%%%%%%%%%%%%%%%%%%%%%%

%%%%%%%%%%%%%%%%%%%%%%%%%%

%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%
%% find statistics of parameters
%
% offdistX =  -3:.25:3;
% heaves = [2000:500:20000];
for n = 1:length(Profslopes.profile)
```

```
disp('************************************************************')

   % [HEidx,ANidx,IFidx,CTidx,offdistidx,teidx] = ind2sub(size(R_ChiSqX), idx);
   disp([ n n n n n n n n n n n n n n n n ])
%
    Profslopes.profile(n).pick.ModelwCTApr1.heaves = [2000:500:20000];
    Profslopes.profile(n).pick.ModelwCTApr1.angles = [45:5:70];
    Profslopes.profile(n).pick.ModelwCTApr1.infill = [ 0 2e3];
    Profslopes.profile(n).pick.ModelwCTApr1.crusts = [0 1e3 3e3 6e3];
    Profslopes.profile(n).pick.ModelwCTApr1.tes = [ 100:100:2000 ];
    Profslopes.profile(n).pick.ModelwCTApr1.offdistX = offdistX;
%
   % find the region of the feature
   lats(n)  = mean(Profslopes.profile(n).lat);
   cornerlons(n,:) = Profslopes.profile(n).corner1;
   cornerlats(n,:) = Profslopes.profile(n).corner2;
   lons(n)  = mean(Profslopes.profile(n).long);

   % find its best fit parameters
   % heave is fixed, so don't need error
   Fits(n).Var(1).Vals(1) =
heaves(Profslopes.profile(n).pick.ModelwCTApr1.Weightedminfitidx(1));
   Fits(n).Var(1).Vals(2) =
heaves(Profslopes.profile(n).pick.ModelwCTApr1.SlopeRawminfitidx(1));
   Fits(n).Var(1).Vals(3) =
heaves(Profslopes.profile(n).pick.ModelwCTApr1.TopoRawminfitidx(1));
%    Fits(n).Var(1).Vals(1) = Profslopes.profile(n).pick.ModelwCTApr1.heaves*1e-
3;
%    Fits(n).Var(1).Vals(2) = Fits(n).Var(1).Vals(1);
%    Fits(n).Var(1).Vals(3) = Fits(n).Var(1).Vals(1);
   Fits(n).Var(1).Label = 'Heave (km)';
   % for fix heave
   Adepths = [ 2.3 2.3 2.3
            0.3 0.3 0.5
            2.4 2.4 2.4
            7.2 7.5 10.3
            3.6 3.6 4.4
            3.6 3.6 4.3
            1.2 0.9 1.1
            6.2 6.2 11.9
            2.6 2.6 2.6
            3.4 3.4 3.6]';
   % for free heave
   Adepths = [ 6.4 6.1 6.4
            2.1 2.4 2.1
```

```
            2.0 2.0 2.0
            19.8 19.7 19.8
            4.1 5.3 4.1
            6.9 7.4 6.5
            0.7 0.9 0.7
            6.5 7.8 6.5
            3.6 4.6 3.6
            12.1 7.9 12.1]';

   Fits(n).Var(5).Vals(1) = Adepths(1,n);
   Fits(n).Var(5).Vals(2) = Adepths(2,n);
   Fits(n).Var(5).Vals(3) = Adepths(3,n);
   Fits(n).Var(5).Label = 'Rooting Depth (km)';

   for m = 2:length(Profslopes.profile(n).pick.ModelwCTApr1.Weightedminfitidx)
      mIDW = Profslopes.profile(n).pick.ModelwCTApr1.Weightedminfitidx(m);
      mIDS = Profslopes.profile(n).pick.ModelwCTApr1.SlopeRawminfitidx(m);
      mIDT = Profslopes.profile(n).pick.ModelwCTApr1.TopoRawminfitidx(m);
      if m == 1
         disp "doing nothing for Heave"
      elseif m == 2
         Fits(n).Var(m).Vals(1) =
Profslopes.profile(n).pick.ModelwCTApr1.angles(mIDW);
         Fits(n).Var(m).Vals(2) =
Profslopes.profile(n).pick.ModelwCTApr1.angles(mIDS);
         Fits(n).Var(m).Vals(3) =
Profslopes.profile(n).pick.ModelwCTApr1.angles(mIDT);
         Fits(n).Var(m).Label = 'Angle (degrees)';
      elseif m == 3
         Fits(n).Var(m).Vals(1) =
Profslopes.profile(n).pick.ModelwCTApr1.infill(mIDW);
         Fits(n).Var(m).Vals(2) =
Profslopes.profile(n).pick.ModelwCTApr1.infill(mIDS);
         Fits(n).Var(m).Vals(3) =
Profslopes.profile(n).pick.ModelwCTApr1.infill(mIDT);
         Fits(n).Var(m).Label = 'Infill (km)';
      elseif m ==4
         Fits(n).Var(m).Vals(1) =
Profslopes.profile(n).pick.ModelwCTApr1.crusts(mIDW)*1e-3;
         Fits(n).Var(m).Vals(2) =
Profslopes.profile(n).pick.ModelwCTApr1.crusts(mIDS)*1e-3;
         Fits(n).Var(m).Vals(3) =
Profslopes.profile(n).pick.ModelwCTApr1.crusts(mIDT)*1e-3;
         Fits(n).Var(m).Label = 'Crust (km)';
      elseif m == 5
         disp 'doin nothing for offset'
```

```
      elseif m == 6
         Fits(n).Var(m).Vals(1) =
Profslopes.profile(n).pick.ModelwCTApr1.tes(mIDW)*1e-3;
         Fits(n).Var(m).Vals(2) =
Profslopes.profile(n).pick.ModelwCTApr1.tes(mIDS)*1e-3;
         Fits(n).Var(m).Vals(3) =
Profslopes.profile(n).pick.ModelwCTApr1.tes(mIDT)*1e-3;
         Fits(n).Var(m).Label = 'Te (km)';
      end
   end

  % grab the histograms
   disp('    %%%%    ')
   disp('Topo:')
   [testprof(n).numDF testprof(n).ys testprof(n).modelfitsbysigma] =
plotErrorSurface(...
               Profslopes.profile(n).pick.ModelwCTApr1.R_ChiSqTopo,...
               Profslopes.profile(n).pick.ModelwCTApr1.TopoRawminfitidx,...
               Profslopes.profile(n).pick.ModelwCTApr1.tes,...
               Profslopes.profile(n).pick.ModelwCTApr1.heaves,...
               Profslopes.profile(n).pick.ModelwCTApr1.angles,...
               Profslopes.profile(n).pick.ModelwCTApr1.infill,...
               Profslopes.profile(n).pick.ModelwCTApr1.crusts,...
               Profslopes.profile(n).pick.ModelwCTApr1.offdistX,...
            Profslopes.profile(n).pick.ModelwCTApr1.TopoRawminfitidx(1),...HEidx
            Profslopes.profile(n).pick.ModelwCTApr1.TopoRawminfitidx(2),...ANidx
            Profslopes.profile(n).pick.ModelwCTApr1.TopoRawminfitidx(3),...IFidx
            Profslopes.profile(n).pick.ModelwCTApr1.TopoRawminfitidx(4),...CTidx

Profslopes.profile(n).pick.ModelwCTApr1.TopoRawminfitidx(5),...offdistidx
            Profslopes.profile(n).pick.ModelwCTApr1.TopoRawminfitidx(6),...teidx
         ",...fet
         ");

   for m = 1:length(testprof(n).ys)
      Fits(n).errlab{3,m} = testprof(n).ys(m).ylab;
      try
         Fits(n).U1err(3,m) =
max(testprof(n).ys(m).ys(testprof(n).modelfitsbysigma(1).ids(:,m)))-
testprof(n).ys(m).miny;
      catch
         Fits(n).U1err(3,m) = 12.345;
      end
      try
```

```
        Fits(n).U2err(3,m) =
max(testprof(n).ys(m).ys(testprof(n).modelfitsbysigma(2).ids(:,m)))-
testprof(n).ys(m).miny;
      catch
        Fits(n).U2err(3,m) = 12.345;
      end
      try
        Fits(n).U3err(3,m) =
max(testprof(n).ys(m).ys(testprof(n).modelfitsbysigma(3).ids(:,m)))-
testprof(n).ys(m).miny;
      catch
        Fits(n).U3err(3,m) = 12.345;
      end
      try
        Fits(n).L1err(3,m) =
min(testprof(n).ys(m).ys(testprof(n).modelfitsbysigma(1).ids(:,m)))-
testprof(n).ys(m).miny;
      catch
        Fits(n).L1err(3,m) = 12.345;
      end
      try
        Fits(n).L2err(3,m) =
min(testprof(n).ys(m).ys(testprof(n).modelfitsbysigma(2).ids(:,m)))-
testprof(n).ys(m).miny;
      catch
        Fits(n).L2err(3,m) = 12.345;
      end
      try
        Fits(n).L3err(3,m) =
min(testprof(n).ys(m).ys(testprof(n).modelfitsbysigma(3).ids(:,m)))-
testprof(n).ys(m).miny;
      catch
        Fits(n).L3err(3,m) = 12.345;
      end
   end
   %%%%%%%%%%%%%
   %%%%%%%%%%%%%
   %%%%%%%%%%%%%
   %%%%%%%%%%%%%
   disp('    %%%%    ')
   disp('Slope:')
   [testprof(n).numDF testprof(n).ys testprof(n).modelfitsbysigma] =
plotErrorSurface(...
            Profslopes.profile(n).pick.ModelwCTApr1.R_ChiSqx,...
            Profslopes.profile(n).pick.ModelwCTApr1.SlopeRawminfitidx,...
            Profslopes.profile(n).pick.ModelwCTApr1.tes,...
```

```
            Profslopes.profile(n).pick.ModelwCTApr1.heaves,...
            Profslopes.profile(n).pick.ModelwCTApr1.angles,...
            Profslopes.profile(n).pick.ModelwCTApr1.infill,...
            Profslopes.profile(n).pick.ModelwCTApr1.crusts,...
            Profslopes.profile(n).pick.ModelwCTApr1.offdistX,...
         Profslopes.profile(n).pick.ModelwCTApr1.SlopeRawminfitidx(1),...HEidx
         Profslopes.profile(n).pick.ModelwCTApr1.SlopeRawminfitidx(2),...ANidx
         Profslopes.profile(n).pick.ModelwCTApr1.SlopeRawminfitidx(3),...IFidx
         Profslopes.profile(n).pick.ModelwCTApr1.SlopeRawminfitidx(4),...CTidx
         Profslopes.profile(n).pick.ModelwCTApr1.SlopeRawminfitidx(5),...offdistidx
         Profslopes.profile(n).pick.ModelwCTApr1.SlopeRawminfitidx(6),...teidx
      ",...fet
      ");
   for m = 1:length(testprof(n).ys)
      Fits(n).errlab{2,m} = testprof(n).ys(m).ylab;
      try
         Fits(n).U1err(2,m) =
max(testprof(n).ys(m).ys(testprof(n).modelfitsbysigma(1).ids(:,m)))-
testprof(n).ys(m).miny;
      catch
         Fits(n).U1err(2,m) = 12.345;
      end
      try
         Fits(n).U2err(2,m) =
max(testprof(n).ys(m).ys(testprof(n).modelfitsbysigma(2).ids(:,m)))-
testprof(n).ys(m).miny;
      catch
         Fits(n).U2err(2,m) = 12.345;
      end
      try
         Fits(n).U3err(2,m) =
max(testprof(n).ys(m).ys(testprof(n).modelfitsbysigma(3).ids(:,m)))-
testprof(n).ys(m).miny;
      catch
         Fits(n).U3err(2,m) = 12.345;
      end
      try
         Fits(n).L1err(2,m) =
min(testprof(n).ys(m).ys(testprof(n).modelfitsbysigma(1).ids(:,m)))-
testprof(n).ys(m).miny;
      catch
         Fits(n).L1err(2,m) = 12.345;
      end
      try
```

```
        Fits(n).L2err(2,m) =
min(testprof(n).ys(m).ys(testprof(n).modelfitsbysigma(2).ids(:,m)))-
testprof(n).ys(m).miny;
      catch
        Fits(n).L2err(2,m) = 12.345;
      end
      try
        Fits(n).L3err(2,m) =
min(testprof(n).ys(m).ys(testprof(n).modelfitsbysigma(3).ids(:,m)))-
testprof(n).ys(m).miny;
      catch
        Fits(n).L3err(2,m) = 12.345;
      end
   end
   %%%%%%%%%%%%%
   %%%%%%%%%%%%%
   %%%%%%%%%%%%%
   disp('    %%%%    ')
   disp('Weight:')
   [testprof(n).numDF testprof(n).ys testprof(n).modelfitsbysigma] =
plotErrorSurface(...

Profslopes.profile(n).pick.ModelwCTApr1.R_ChiSqTopo*mean([Profslopes.profile(
n).pick.ModelwCTApr1.maxTw,Profslopes.profile(n).pick.ModelwCTApr1.minTw])
+ ...
      Profslopes.profile(n).pick.ModelwCTApr1.R_ChiSqx*(1-
mean([Profslopes.profile(n).pick.ModelwCTApr1.maxTw,Profslopes.profile(n).pick.
ModelwCTApr1.minTw])),...
            Profslopes.profile(n).pick.ModelwCTApr1.Weightedminfitidx,...
            Profslopes.profile(n).pick.ModelwCTApr1.tes,...
            Profslopes.profile(n).pick.ModelwCTApr1.heaves,...
            Profslopes.profile(n).pick.ModelwCTApr1.angles,...
            Profslopes.profile(n).pick.ModelwCTApr1.infill,...
            Profslopes.profile(n).pick.ModelwCTApr1.crusts,...
            Profslopes.profile(n).pick.ModelwCTApr1.offdistX,...
         Profslopes.profile(n).pick.ModelwCTApr1.Weightedminfitidx(1),...HEidx
         Profslopes.profile(n).pick.ModelwCTApr1.Weightedminfitidx(2),...ANidx
         Profslopes.profile(n).pick.ModelwCTApr1.Weightedminfitidx(3),...IFidx
         Profslopes.profile(n).pick.ModelwCTApr1.Weightedminfitidx(4),...CTidx
         Profslopes.profile(n).pick.ModelwCTApr1.Weightedminfitidx(5),...offdistidx
         Profslopes.profile(n).pick.ModelwCTApr1.Weightedminfitidx(6),...teidx
      ',...fet
      ');
   for m = 1:length(testprof(n).ys)
      Fits(n).errlab{1,m} = testprof(n).ys(m).ylab;
      try
```

```
        Fits(n).U1err(1,m) =
max(testprof(n).ys(m).ys(testprof(n).modelfitsbysigma(1).ids(:,m)))-
testprof(n).ys(m).miny;
      catch
        Fits(n).U1err(1,m) = 12.345;
      end
      try
        Fits(n).U2err(1,m) =
max(testprof(n).ys(m).ys(testprof(n).modelfitsbysigma(2).ids(:,m)))-
testprof(n).ys(m).miny;
      catch
        Fits(n).U2err(1,m) = 12.345;
      end
      try
        Fits(n).U3err(1,m) =
max(testprof(n).ys(m).ys(testprof(n).modelfitsbysigma(3).ids(:,m)))-
testprof(n).ys(m).miny;
      catch
        Fits(n).U3err(1,m) = 12.345;
      end
      try
        Fits(n).L1err(1,m) =
min(testprof(n).ys(m).ys(testprof(n).modelfitsbysigma(1).ids(:,m)))-
testprof(n).ys(m).miny;
      catch
        Fits(n).L1err(1,m) = 12.345;
      end
      try
        Fits(n).L2err(1,m) =
min(testprof(n).ys(m).ys(testprof(n).modelfitsbysigma(2).ids(:,m)))-
testprof(n).ys(m).miny;
      catch
        Fits(n).L2err(1,m) = 12.345;
      end
      try
        Fits(n).L3err(1,m) =
min(testprof(n).ys(m).ys(testprof(n).modelfitsbysigma(3).ids(:,m)))-
testprof(n).ys(m).miny;
      catch
        Fits(n).L3err(1,m) = 12.345;
      end
    end

    %%%%%%%%%%%%%
    %%%%%%%%%%%%%
    %%%%%%%%%%%%%
```

```
disp('***************************************************************')
end
%% plot these things

%%%% /Users/marklar/Documents/MATLAB/OCCflex/InversionRoutine
CCs = [-45.0262   15.08
   -45.1 14.875
-44.93     14.843
-44.9100   14.6650
-44.8396   13.8339
-45.0800   13.6980
-44.9500   13.5150
-44.9500   13.3200
-44.9000   13.1040
-44.7053   12.8254];
try
[X,Y,Z]=grdread2('Bathy12_15N.grd');
catch
here = pwd;
cd ..
[X,Y,Z]=grdread2('Bathy12_15N.grd');
cd(here)
end
figure(101)
clf;
surf(X,Y,Z)
view( 0, 90)
shading interp
lightangle(-90,1e-20)
lightangle(180,1e-20)
xlim([-46.5, -44.6])
ylim([12.65, 15.25])
axis equal
% fillshit = ([(Ates(3,:))-LTerr  fliplr(UTerr+(Ates(3,:))) ]);
%    fill(fillshit,[lats fliplr(lats)],'g','facealpha',.25)
hold on
for n = 1:length(cornerlats)
   p=plot(CCs(n,1),CCs(n,2),'o',...
      'MarkerSize',20,...        % 'MarkerFaceColor',1-[0 1 1]*Ates(3,n)/max(Ates(:)),...
      'MarkerFaceColor',1-[0 1 1]*(Fits(n).Var(6).Vals(3)/Fits(10).Var(6).Vals(3)),...
      'MarkerEdgeColor','k')
   text(CCs(n,1),.05+CCs(n,2),sprintf('%.0f m', Fits(n).Var(6).Vals(3)*1e3),...
      'Fontsize',15,'HorizontalAlignment','Center','Fontweight','bold')
   %f=fill(cornerlons(n,:),cornerlats(n,:),1-[0 1
1]*Ates(3,n)/max(Ates(:)),'Facealpha',.75);
```

```
   %f.AmbientStrength=.8;
   %text(lons(n),lats(n),sprintf('%.0f m', Ates(3,n)*1e3),'Fontsize',15)
end
xlabel('Longitude')
ylabel('Latitude')
ax = gca;
ax.FontSize = 18;

%%

bulats=lats;
lats = min(abs([12.65-lats ; 15.2-lats])) + 13.5;
try
bua = a;
catch
   disp 'no a'
end
a = xlim +.2;
latfac = 0.5;
Ates=Ates.*latfac;
fill([a(1)  a(1)+1 a(1)+1 a(1) a(1)],[min(lats)*[.995 .995] max(lats)*[1.01 1.01]
min(lats)*.995],...
       'w',...
       'Linewidth',1,'AmbientStrength',1)
errorbarxy(Ates(3,:)+a(1),lats,LTerr.*latfac,UTerr.*latfac,0,0,{'','g','k'})

 %fillshit = ([(Ates(2,:)+a(1))-LSerr.*latfac  fliplr(USerr.*latfac+(a(1) + Ates(2,:)))
]);
 %   fill(fillshit,[lats fliplr(lats)],'m','facealpha',.25)
errorbarxy(Ates(2,:)+a(1),lats,LSerr.*latfac,USerr.*latfac,0,0,{'','m','k'})
 % fillshit = ([(Ates(1,:)+a(1))-LWerr.*latfac  fliplr(UWerr.*latfac+(Ates(1,:)+a(1)))
]);
 %    fill(fillshit,[lats fliplr(lats)],'k','facealpha',.25)
errorbarxy(Ates(1,:)+a(1),lats,LWerr.*latfac,UWerr.*latfac,0,0,{'','k','k'})
% plot(mean(Ates)+a(1),lats,...
%        '+','Color',[1 0 0 ])%'Linestyle','','Linewidth',5
hold  on

plot(Ates(1,:)+a(1),lats,...
      'o','MarkerFaceColor',[.5 .5 .5 ],'MarkerSize',10,'MarkerEdgeColor',[.5 .5 .5 ])%
'Linestyle','','Marker',,'Linewidth',4.5
plot(Ates(2,:)+a(1),lats,...
       'o','MarkerFaceColor',[1 0 1 ],'MarkerSize',10,'MarkerEdgeColor',[1 0 1
])%'Linestyle','','Marker', ,'Linewidth',2
plot(Ates(3,:)+a(1),lats,...
```

```
      'o','MarkerFaceColor',[0 1 0 ],'MarkerSize',10,'MarkerEdgeColor',[0 1 0 ])
%'Linestyle','','Marker', ,'Linewidth',2
for n = 1:length(Ates)
   text(Ates(1,n)+a(1)+.2+max([UWerr(n) USerr(n)
UTerr(n)]*latfac),lats(n),sprintf('%.2f%c N',bulats(n),char(176)),'Fontweight','bold')
end
%xlabel('Elastic Thickness (m)')
for n = .1:.1:1.9
   line(a(1)+n*.5 * [1 1],[max(lats)*1.01 min(lats)*.995],'color',[.2 .2 .2 .2])
end
for n = [0:.2:1 1.25:.25:2]
   t = text(a(1) +n*.5 ,
min(lats)*.992,sprintf('%.0f',n*1000),'Fontsize',15,'HorizontalAlignment','center');
   t.BackgroundColor = 'w';
end
t = text(a(1) + .5, min(lats)*.988,'Elastic Thickness
(m)','Fontsize',16,'HorizontalAlignment','center');
t.BackgroundColor = 'w';
fact = 1.2/abs(diff([max(lats)*1.01 min(lats)*.995]));
for n = 0:.1:1.2
   line([a(1) a(1)+1],min(lats)*.995*[1 1] + n*fact,'color',[.2 .2 .2 .2])
   t = text(a(1)*1.001,min(lats)*.995 +
n*fact,sprintf('%.1f',n),'Rotation',45,'Fontsize',15,'HorizontalAlignment','center');
   t.BackgroundColor = 'w'
end
text(a(1)*1.003,min(lats)*1.02,'Distance from Nearest Transform
(Degrees)','Rotation',90,'Fontsize',16,'HorizontalAlignment','center')
%t.BackgroundColor = 'w';
a = bua;
lats = bulats;
Ates=Ates/latfac;
%%
fig = figure(202);clf
TFlats = min(abs([12.65-lats ; 15.2-lats])) ;
cd ..
for n = 1:length(lats)
   TFlats(n) = min([    ll2m([12.65 lats(n)],[lons(n) lons(n)]) ...
                ll2m([15.2 lats(n)],[lons(n) lons(n)])    ])*1e-3;
end
cd InversionRoutine/
%%
clear l

colr = [ .5 .5 .5;
       1 0 1
       0 1 0];
```

```
for n = 1:6
   for m = 1:3
      subplot(2,3,n)
      plot(As(n).A(m,:),TFlats,'Linestyle','none','Marker','none');
      l = lsline;
      poly1s(m,n).x = l.XData;
      poly1s(m,n).y = l.YData;
      clf
   end
end
%%
for n = 1:6
   for m = 1:3
      subplot(2,3,n);hold on
      plot(poly1s(m,n).x,poly1s(m,n).y,'Linestyle','-','Color',colr(m,:));
   end
end
%%
for n = 1:length(TFlats)


subplot(2,3,1)
% plot(mean(Aang),TFlats,...
%        'Linestyle','none','Marker','+','Color',[1 0 0 .5])
hold  on
if LWerr(n)==0 && UWerr(n)==0
plot(Aang(1,n),TFlats(n),'Linestyle','none','Marker','d','Color',[.5 .5 .5 .5])
else
plot(Aang(1,n),TFlats(n),'Linestyle','none','Marker','d','Color','k','MarkerFaceColor',[.
5 .5 .5])
end
if LSerr(n)==0 && USerr(n)==0
plot(Aang(2,n),TFlats(n),'Linestyle','none','Marker','s','Color',[1 0 1 .1])
else
plot(Aang(2,n),TFlats(n),'Linestyle','none','Marker','s','Color','k','MarkerFaceColor',[1
0 1])
end
if LTerr(n)==0 && UTerr(n)==0
plot(Aang(3,n),TFlats(n),'Linestyle','none','Marker','o','Color',[0 1 0 .2])
else
plot(Aang(3,n),TFlats(n),'Linestyle','none','Marker','o','Color','k','MarkerFaceColor',[0
1 0 ])
end
ylim([0 max(TFlats)*1.1])
xlim([40 max(Aang(:))])
xlabel('Angle (degrees)')
```

```
ylabel('Distance from Transform Boundary (Degrees Long.)')
h = gca;
h.FontSize = 15;
box on
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
subplot(2,3,2)
% plot(mean(Ates),TFlats,...
%         'Linestyle','none','Marker','+','Color',[1 0 0 .5])
hold  on
if LWerr(n)==0 && UWerr(n)==0
plot(Ates(1,n),TFlats(n),'Linestyle','none','Marker','d','Color',[.5 .5 .5 .5])
else
plot(Ates(1,n),TFlats(n),'Linestyle','none','Marker','d','Color','k','MarkerFaceColor',[.5
.5 .5])
end
if LSerr(n)==0 && USerr(n)==0
plot(Ates(2,n),TFlats(n),'Linestyle','none','Marker','s','Color',[1 0 1 .1])
else
plot(Ates(2,n),TFlats(n),'Linestyle','none','Marker','s','Color','k','MarkerFaceColor',[1
0 1])
end
if LTerr(n)==0 && UTerr(n)==0
plot(Ates(3,n),TFlats(n),'Linestyle','none','Marker','o','Color',[0 1 0 .2])
else
plot(Ates(3,n),TFlats(n),'Linestyle','none','Marker','o','Color','k','MarkerFaceColor',[0
1 0 ])
end
ylim([0 max(TFlats)*1.1])
xlim([0 max(Ates(:))])
xlabel('Elastic Thickness (km)')
h = gca;
h.FontSize = 15;
box on
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
subplot(2,3,3)
% plot(mean(Adepths),TFlats,...
%         'Linestyle','none','Marker','+','Color',[1 0 0 .5],'Linewidth',5)
hold  on
if LWerr(n)==0 && UWerr(n)==0
plot(Adepths(1,n),TFlats(n),'Linestyle','none','Marker','d','Color',[.5 .5 .5 .5])
else
plot(Adepths(1,n),TFlats(n),'Linestyle','none','Marker','d','Color','k','MarkerFaceColor'
,[.5 .5 .5])
end
```

```
if LSerr(n)==0 && USerr(n)==0
plot(Adepths(2,n),TFlats(n),'Linestyle','none','Marker','s','Color',[1 0 1 .1])
else
plot(Adepths(2,n),TFlats(n),'Linestyle','none','Marker','s','Color','k','MarkerFaceColor'
,[1 0 1])
end
if LTerr(n)==0 && UTerr(n)==0
plot(Adepths(3,n),TFlats(n),'Linestyle','none','Marker','o','Color',[0 1 0 .2])
else
plot(Adepths(3,n),TFlats(n),'Linestyle','none','Marker','o','Color','k','MarkerFaceColor'
,[0 1 0 ])
end
ylim([0 max(TFlats)*1.1])
xlim([0 max(Adepths(:))])
xlabel('Rooting Depth (km)')
h = gca;
h.FontSize = 15;
box on
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
subplot(2,3,4)
% plot(mean(Ahe),TFlats,...
%       'Linestyle','none','Marker','+','Color',[1 0 0 .5],'Linewidth',5)
hold  on
if LWerr(n)==0 && UWerr(n)==0
plot(Ahe(1,n),TFlats(n),'Linestyle','none','Marker','d','Color',[.5 .5 .5 .5])
else
plot(Ahe(1,n),TFlats(n),'Linestyle','none','Marker','d','Color','k','MarkerFaceColor',[.5
.5 .5])
end
if LSerr(n)==0 && USerr(n)==0
plot(Ahe(2,n),TFlats(n),'Linestyle','none','Marker','s','Color',[1 0 1 .1])
else
plot(Ahe(2,n),TFlats(n),'Linestyle','none','Marker','s','Color','k','MarkerFaceColor',[1 0
1])
end
if LTerr(n)==0 && UTerr(n)==0
plot(Ahe(3,n),TFlats(n),'Linestyle','none','Marker','o','Color',[0 1 0 .2])
else
plot(Ahe(3,n),TFlats(n),'Linestyle','none','Marker','o','Color','k','MarkerFaceColor',[0
1 0 ])
end
xlabel('Heave (m)')
xlim([0 max(Ahe(:))])
ylim([0 max(TFlats)*1.1])
ylabel('Distance from Transform Boundary (Degrees Long.)')
```

```
h = gca;
h.FontSize = 15;
box on
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
subplot(2,3,5)
% plot(mean(Ain),TFlats,...
%         'Linestyle','none','Marker','+','Color',[1 0 0 .5],'Linewidth',5)
hold  on
if LWerr(n)==0 && UWerr(n)==0
plot(Ain(1,n),TFlats(n),'Linestyle','none','Marker','d','Color',[.5 .5 .5 .5])
else
plot(Ain(1,n),TFlats(n),'Linestyle','none','Marker','d','Color','k','MarkerFaceColor',[.5
.5 .5])
end
if LSerr(n)==0 && USerr(n)==0
plot(Ain(2,n),TFlats(n),'Linestyle','none','Marker','s','Color',[1 0 1 .1])
else
plot(Ain(2,n),TFlats(n),'Linestyle','none','Marker','s','Color','k','MarkerFaceColor',[1 0
1])
end
if LTerr(n)==0 && UTerr(n)==0
plot(Ain(3,n),TFlats(n),'Linestyle','none','Marker','o','Color',[0 1 0 .2])
else
plot(Ain(3,n),TFlats(n),'Linestyle','none','Marker','o','Color','k','MarkerFaceColor',[0 1
0 ])
end
xlabel('Infill (m)')
ylim([0 max(TFlats)*1.1])
xlim([0 max(Ain(:))])
h = gca;
h.FontSize = 15;
box on
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
subplot(2,3,6)
% plot(mean(Acru),TFlats,...
%         'Linestyle','none','Marker','+','Color',[1 0 0 .5],'Linewidth',5)
hold  on
if LWerr(n)==0 && UWerr(n)==0
plot(Acru(1,n),TFlats(n),'Linestyle','none','Marker','d','Color',[.5 .5 .5 .5])
else
plot(Acru(1,n),TFlats(n),'Linestyle','none','Marker','d','Color','k','MarkerFaceColor',[.5
.5 .5])
end
if LSerr(n)==0 && USerr(n)==0
```

```
plot(Acru(2,n),TFlats(n),'Linestyle','none','Marker','s','Color',[1 0 1 .1])
else
plot(Acru(2,n),TFlats(n),'Linestyle','none','Marker','s','Color','k','MarkerFaceColor',[1
0 1])
end
if LTerr(n)==0 && UTerr(n)==0
plot(Acru(3,n),TFlats(n),'Linestyle','none','Marker','o','Color',[0 1 0 .2])
else
plot(Acru(3,n),TFlats(n),'Linestyle','none','Marker','o','Color','k','MarkerFaceColor',[0
1 0 ])
end
xlabel('Crustal Thickness (m)')
ylim([0 max(TFlats)*1.1])
xlim([0 max(Acru(:))])
h = gca;
h.FontSize = 15;
box on
end


%% inverse fft iof profile attempt

Gcolrs=[.5 .5 .5 .5
1 0 1 .1
0 1 0 .2];

colrs=[.5 .5 .5
1 0 1
0 1 0 ];
mker = {'d','s','o'};


%%
TFlats = min(abs([12.65-lats ; 15.2-lats])) ;
cd ..
for n = 1:length(lats)
   TFms(n) = min([    ll2m([12.65 lats(n)],[lons(n) lons(n)]) ...
                ll2m([15.2 lats(n)],[lons(n) lons(n)])    ])*1e-3;
end
cd InversionRoutine/
%%
figure(203);clf;hold on
for m = 1:length(Fits(n).Var)
   subplot(2,3,m);hold on
   for n = 1:length(Fits)
      for j = 1:length(Fits(n).Var(m).Vals)
```

```
        try
           lID = find(strcmp(Fits(n).errlab(j,:),Fits(n).Var(m).Label));
           if Fits(n).L1err(j,lID) == 12.345 || Fits(n).U1err(j,lID) == 12.345
              grab
           end

           X=Fits(n).Var(m).Vals(j);
           Y=TFms(n);
           Xl=Fits(n).L1err(j,lID);
           Xu=Fits(n).U1err(j,lID);
           if Xl - X < 0
              Xl = 0;
           else
              Xl = abs(Xl);
           end

           errorbarxy(X,Y,...
                 Xl,Xu,0,0,{'',colrs(j,:),'k'},(4-j)*1.5,2)

            plot(X,Y,'Linestyle','none','MarkerSize',(6-
j)*1.5,'Marker',mker{j},'Color','k','MarkerFaceColor',colrs(j,:))
         catch
           disp('skippin')
            plot(Fits(n).Var(m).Vals(j),TFms(n),'Linestyle','none','MarkerSize',(6-
j)*1.5,'Marker',mker{j},'Color',Gcolrs(j,:))
        end
%%% for plot with no error
%plot(Fits(n).Var(m).Vals(j),TFms(n),'Linestyle','none','MarkerSize',(6-
j)*1.5,'Marker',mker{j},'Color','k','MarkerFaceColor',colrs(j,:))

         hold on

      end
   end
   if m == 1 | m == 4
      ylabel('Distance from Transform (km)')
   end
        xlabel(Fits(n).Var(m).Label)
   box on
end
%%

for n = 1:length(Profslopes.profile)
   fittingdists(n) = abs(diff([Profslopes.profile(n).pick.fittingdist(1)
Profslopes.profile(n).pick.fittingdist(end)]))
end
```

```
% % % % % % % % % %
% % % % % % % % % %
% % % % % % % % % % E=1e11;                % Youngs modulus
% % % % % % % % % % v=.25;                % Poisson's ratio
% % % % % % % % % % g=9.81;                % Gravity
% % % % % % % % % % rho_w=1030;            % water
% % % % % % % % % % rho_c=2700;            % crust
% % % % % % % % % % rho_m=3300;            % mantle density
% % % % % % % % % % rho_s=3000;            % serpentinite density
% % % % % % % % % % te=1000;
% % % % % % % % % % D=E.*te^3/(12*(1-v^2));
% % % % % % % % % % %%
% % % % % % % % % %
% % % % % % % % % %         st=ht*(rho_c-rho_w)/(rho_m-rho_w);
% % % % % % % % % %         sc=hc*(rho_m-rho_c)/(rho_m-rho_w);
% % % % % % % % % %     s=-(sc+st);
% % % % % % % % % % [S,k]=jfft(s,dx);
% % % % % % % % % %
% % % % % % % % % % cte=(rho_m-rho_w)*(D/g.*k.^4+((rho_m-rho_w))).^(-
1);
% % % % % % % % % %     D=E.*te^3/(12*(1-v^2));
% % % % % % % % % %
% % % % % % % % % % W=cte.*S';
% % % % % % % % % % w=jifft(W);
% % % % % % % % % %
% % % % % % % % % % %% Reverse this
% % % % % % % % % %
% % % % % % % % % % % st=ht*(rho_c-rho_w)/(rho_m-rho_w);
% % % % % % % % % % % sc=hc*(rho_m-rho_c)/(rho_m-rho_w);
% % % % % % % % % % % s=-(sc+st);
% % % % % % % % % % % % find some wavelenghts, and their spectrum
% % % % % % % % % % % [S,k]=jfft(s,dx);
% % % % % % % % % % % % find D, a scalar, whic is the rigidity of the plate.
% % % % % % % % % % % % from Turcotte & Schubert 3rd Ed. eqn. 3.72, also,
Buck 1988
% % % % % % % % % % % D=E.*te^3/(12*(1-v^2));
% % % % % % % % % % % % find compensation, from Turcotte Schubert 3rd
Ed. eqn 3.117
% % % % % % % % % % % cte=(rho_m-rho_w)*(D/g.*k.^4+((rho_m-
rho_w))).^(-1);
% % % % % % % % % % % % this is a curve that starts at x=0, y = 1,
% % % % % % % % % % % % and then goes to y=0 as x -> length profile
% % % % % % % % % % % % curvature is changed with all the factors, but Te is
the one we deal with
% % % % % % % % % % % W=cte.*S';
```

```
% % % % % % % % % % % % now time to do the inverse fft, and find the real
part
% % % % % % % % % % % % now we have our w
% % % % % % % % % % % w=jifft(W);
% % % % % % % % % % %
% % % % % % % % % % % %%% So this?
% % % % % % % % % % [flipud(Profslopes.profile(2).zs(end/2:end-
1));Profslopes.profile(2).zs(end/2:end-1) ]';
% % % % % % % % % % dx=1;
% % % % % % % % % %
h=interp1(1:dx:dx*length(h),h,linspace(1,dx*length(h),4096),'spline','extrap')
% % % % % % % % % % %[W, k] = jfft(h,dx);
% % % % % % % % % % W = fft(h);
% % % % % % % % % % N = length(W);
% % % % % % % % % % dk = 2.*pi/(N.*dx);
% % % % % % % % % % k=dk .* [ 0:N-1]';
% % % % % % % % % % cte=(rho_m-rho_w)*(D/g.*k.^4+((rho_m-rho_w))).^(-
1);
% % % % % % % % % % S = W./cte';
% % % % % % % % % % s=ifft(S);
% % % % % % % % % % subplot(2,1,1)
% % % % % % % % % % plot(Profslopes.profile(2).zs(end/2:end-1))
% % % % % % % % % % subplot(2,1,2)
% % % % % % % % % % plot(real(s(end/2:end)/1e10))
% % % % % % % % % % disp('done')
% % % % % % % % % % %%
% % % % % % % % % % % where
% % % % % % % % % % %  cte / (rho_m-rho_w) = (D/g.*k.^4+(rho_m-
rho_w)).^(-1)
% % % % % % % % % %       (rho_m-rho_w)/cte = D/g.*k.^4+(rho_m-rho_w)
% % % % % % % % % %     (rho_m-rho_w) - (rho_m-rho_w)/cte = D/g.*k.^4
% % % % % % % % % %   ((rho_m-rho_w) - (rho_m-rho_w)/cte )/D =
(g.*k.^4)^.-1
% % % % % % % % % %    D/((rho_m-rho_w) - (rho_m-rho_w)/cte ) =
g.*k.^4
% % % % % % % % % %  D/(g.*((rho_m-rho_w) - (rho_m-rho_w)/cte )) = k.^4
% % % % % % % % % %  (D/(g.*((rho_m-rho_w) - (rho_m-rho_w)/cte
))).^(1/4) = k
```

6.12 jfft.m

```
function [H,k]=ezfft(h,dx)
%[H,k]=jfft(h,dx);
% provides positive-frequency fft of a function.
% vector k is such that dk = 2.*pi/(N*dx)
```

```
%%% INPUT
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% h - vector of topography
% dx - spacing of the topography points in the x direction
%%% OUTPUT
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% H -
%
%

% make sure power of 2. should have already been done though
N=power2(length(h));
huse=h;
%
% Possibly pad huse with last value, out to next power of 2:
if N~=length(h), disp('(ezfft): **Input not power of 2. Padding...'); end
%huse(length(h)+1:N)=interp1([length(h);N],[h(length(h));h(1)],length(h)+1:N);
%huse(length(h)+1:N)=table1([length(h) h(length(h));N h(1)],length(h)+1:N);

% this only does something if N~= length(h)
% otherwise, huse(length(h)...) is an empty matrix
huse(length(h)+1:N)=interp1([length(h) h(length(h))], [N h(1)],length(h)+1:N);


%disp([length(h),N,length(huse)])
%
%%% Define values for positive k
% N*dx should be 2e5
% making dk pi*1e-5
% this is our spacing in frequency space
dk = 2.*pi/(N.*dx);
% then we make a vector of half our length vector
% and scale it to our dk
k=dk .* [ 0:N./2]';
% we take half because we only take half below, of our fft

%%% now take the fast fourier transform of our topography.
% this provides a power spectrum
hfft = fft(huse);
% we are only interested in the first half
H=hfft(1:(N./2 + 1));
%disp(length(H))
```

6.13 jifft.m

```
function [h]=ezifft(H)
%[h]=ezifft(H);
%where H is complex for positive frequencies.
%length of H must be a n+1 where n is a power of 2.
hfft = H;
n=length(H);
if n-1 ~= power2(n-1),
  %disp('(ezifft): length not power2+1');
else
  N=2*(length(H)-1);
  hfft( N:-1:(N/2)+2 ) = conj(hfft(2:N/2));
  h=real(ifft(hfft));
end
```

6.14 ll2m.m

```
function meters = ll2m(lats,lons)
% ll2m Finds distance (in meters) between 2 latitude-longitude pairs
% Explicit: Finds distance between lat(1) lons(1) and lat(2) lons(2)
% example: meters = ll2m(lats,lons)
% Mark Oscar Larson
%%% INPUTS
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%
% lats:     2 latitudes
% lons:     2 longitudes
%%% OUTPUTS
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%
% meters:   distance in meters
%%% Internal
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%
% R:    Radius of the Earth

lat1=lats(1)*pi/180;
lat2=lats(2)*pi/180;
lon1=lons(1)*pi/180;
lon2=lons(2)*pi/180;
R = 6371000; % meters
dellat = abs(lat1-lat2);
dellon = abs(lon1-lon2);
a = sin(dellat/2) * sin(dellat/2) + ...
         cos(lat1) *  cos(lat2) * ...
        sin(dellon/2) * sin(dellon/2);
c = 2 * atan2( sqrt(a), sqrt(1-a) );
```

```
meters = R * c;
```

6.15 misfitweights.m

```
function [maxTw , minTw , Weightedminfitidx] = misfitweights(Misfits)

% first, find the misfits which have a value. Some models were not run, and
% have a 1e6 misfit.
TopoMF = (Misfits.R_ChiSqTopo(find( Misfits.R_ChiSqTopo(:) ~=
max(Misfits.R_ChiSqTopo(:)))));
SlopMF = (Misfits.R_ChiSqx(find( Misfits.R_ChiSqx(:) ~=
max(Misfits.R_ChiSqx(:)))));

% ScaledSlope = Misfits.R_ChiSqx/var(SlopMF);
% ScaledTopo = Misfits.R_ChiSqTopo/var(TopoMF);


testweights = [.0001:.0001:.9999];

idxs = zeros(length(testweights),1);
bestfitmistfits = idxs;
for m = 1:length(testweights)
   [bestfitmistfits(m) , idxs(m) ] = min(Misfits.R_ChiSqx(:)*testweights(m) ...
                                + Misfits.R_ChiSqTopo(:)*(1-testweights(m)));

end
%%
clear distances
distances = sqrt( ([Misfits.R_ChiSqx(:).^ 2 + Misfits.R_ChiSqTopo(:).^ 2]) );
[mindist distID ] =min(distances);
figure(2);clf
subplot(3,1,1)
plot(Misfits.R_ChiSqx(:),Misfits.R_ChiSqTopo(:)','*')
hold on
plot(Misfits.R_ChiSqx(distID),Misfits.R_ChiSqTopo(distID)','ro')
ylabel('Topography \chi^2')
xlabel('Slope \chi^2')
title('Misfit space: Slope \chi^2 vs Topography \chi^2 space')
%%
subplot(3,1,2)
plot(Misfits.R_ChiSqx(:),Misfits.R_ChiSqTopo(:)','*')
hold on
plot(Misfits.R_ChiSqx(distID),Misfits.R_ChiSqTopo(distID)','ro')
xlim([0 500])
ylim([0 100])
```

```
ylabel('Topography \chi^2')
xlabel('Slope \chi^2')
title('Misfit space: Slope \chi^2 vs Topography \chi^2 space')
%%
subplot(3,2,5)
h = histogram(TopoMF);
hold on
histogram(SlopMF,h.BinEdges)
xlabel('Misfit value')
ylabel('Frequency')
title('Histogram of the misfits in each model space')
legend('Topography Misfits','Slope Misfits')
%%

% this always gives Tw = max and Sw = min
% idx = find(bestfitmistfits == min(bestfitmistfits));
%
% Sw = testweights(idx);
% Tw = 1 - Tw;
%
% [HEidx,ANidx,IFidx,CTidx,offdistidx,teidx] = ind2sub(size(Misfits.R_ChiSqx),
idxs(idx));
% Weightedminfitidx = [HEidx,ANidx,IFidx,CTidx,offdistidx,teidx];

% lets try to find the most numerous bestfit
uids = unique(idxs);

for n = 1:length(uids)
   numofmodels(n) = numel(find(idxs == uids(n)));
end

subplot(3,2,6)
bar((uids),log(numofmodels))
title('Histogram of best fit models')
xlabel('Matrix ID')
ylabel('Log_1_0 Frequency')
for n = 1:length(uids)
   text(uids(n)+100,log(numofmodels(n)),sprintf('%g',numofmodels(n)))
end

bfidx = find(bestfitmistfits == min(bestfitmistfits));
Tw = 1 - testweights(bfidx);
text(idxs(bfidx),4,sprintf('Minimun, with Topo_{weight}: %g',Tw))

%%
acceptable = find(idxs == uids(find(numofmodels == max(numofmodels))));
```

```
acceptableID = uids(find(numofmodels == max(numofmodels)));
maxTw = min(testweights(acceptable));
minTw = max(testweights(acceptable));

[HEidx,ANidx,IFidx,CTidx,offdistidx,teidx] = ind2sub(size(Misfits.R_ChiSqx),
acceptableID);
if acceptableID ~= distID
   disp('Most common and Shortest Distance misfits are NOT EQUAL')
   disp('Going with distance model')
   [HEidx,ANidx,IFidx,CTidx,offdistidx,teidx] = ind2sub(size(Misfits.R_ChiSqx),
distID);
end

Weightedminfitidx = [HEidx,ANidx,IFidx,CTidx,offdistidx,teidx];

text(acceptableID+100,8,sprintf('Tw_m_e_a_n = %g \nTw_r_a_n_g_e = %g -
%g',(maxTw+minTw)/2,maxTw,minTw))
```

6.16 OCCFlexMain.m

```
% master script to import .grd file and use semiautomated routine to identify OCC
features
% saves a structure which is then used by inversion

%CALCULATE SLOPE DISTRIBUTION FOR A REGION

% give the name of the grid file
gridfile = 'Bathy12_15N.grd';
% an axis file if you have on, otherwise, you will pick the axis points
Axisfile ='Axis_12_15N_Mallow&Searle2012_supp_Fujiwara2003.txt';
% East and west bounding fault files if you have them,
WestBoundingFaults = 'WBF_Mallow&Searle2012_Aug12.txt';
EastBoundingFaults = 'EBF_Mallow&Searle2012_Aug12.txt';
% a gravity map of the region if you have it.
do_grav = 1; % binary switch
gravgridfile = 'cutrmba.grd';
gravtype ='RMBA'; % RMBA or MBA, etc
% if there is EQ data
EQfname = 'EQ_GMT_Aug18_2015.m'; % if no data, set to ''

% set the subregion, if you want
% otherwise, region will be the entire grid file
long1 =  45.6;
long2 = 44.4;
lat1 = 12.63;
lat2 =  15.25;
```

```
% Spreading direction, I got this from Okino's plate calculator
SD = 102.9;
% these were the coordinates for my spreading direction
% (lat,lon) = (30.10, 317.90) is 23.64 mm/year at an azimuth of N102.9E

% inputs for fft filtering of the region
% set the low cut/highpass filter here
% highpass1 will assign all wavelengths greater than itself a '1' in fft
% domain
% highpass2 will grade sinusoidally from 1 (at highpass1) to 0, at
% highpass2
highpass1x = 100; % these are in units of km roughly
highpass2x = 50;
% Set y direction to zero to uniformly filter
highpass1y = 100;
highpass2y = 50;
% do the filter swith
dohpf =1;
% do the filter twice switch? this removes the large features
removebig=1;
if dohpf
   if highpass2y > 0
      HPFstr = sprintf('HPFxy_%s_%s_%s_%s',num2str(highpass1x),...
                                    num2str(highpass2x),...
                                    num2str(highpass1y),...
                                    num2str(highpass2y));
   else
      HPFstr = sprintf('HPFx%s_%s',num2str(highpass1x),num2str(highpass2x));
   end
else
  HPFstr = 'noHPF';
end

% set cutoff area (in pixels)
% or cutoff length of major axis
% set the other value to 0
cutoff=0;
cutofflength = 30;
cutofflength2 = 5;
connectivity=8;
if cutoff > 0
   CLstring = sprintf('CL%d',cutoff);
else
   CLstring = sprintf('CL%d_%d',cutofflength2,cutofflength);
end
```

```
% set slope filter amounts in degrees
lowslope = 15;
highslope = 70;

% set profile length in km
minimum_prof_length = 80;
% set to 0, to let variable length be decided
orthog_to_ridge = 0;

%%% Save Figures?
print_y_n = 0;
% This is a cuttoff for direction of identified features facing
% give the average spreading direction,
% or whatever you want for west azimuth spreading
% direction
% east will be -180 of that
westaz = 270;
eastaz = westaz-180;
% give two values bounding the east direction
eastlowaz=40;
easthighaz=180;
% and two values bounding the west direction
westlowaz =180;
westhighaz=350;

% make title for figures from provided parameters
disp(sprintf('\n\n:::: Parameters are understood to be ::::'))
app_title = sprintf('%s_%s_SL%.1f_%.1f_Waz%d_%d_Eaz%d_%d_%s',...
        HPFstr,...
        CLstring,...
        lowslope,highslope,...
        westlowaz,westhighaz,...
        eastlowaz,easthighaz,...
        datestr(now,'mm_dd_yyyy'));
disp(app_title)
%% Set up interpolation for Bathy and Grav
% for bathymetry
disp(sprintf(':::: Loading :::: \n \t\t %s ',gridfile))
[long,lat, bathy]=grdread2(gridfile);
disp(':::: File Grid Spacing :::: :::: :::: :::: :::: :::: :::: :::: :::: :::: :::: ::::')
disp(sprintf('\t :: x dimension:\n\t\tMax: %.2f Deg \n\t\tMin: %.2f Deg
\n\t\tSpacing:\t%.4f Deg \n\t\tLat %.2f:\t%.4f m\n\t\tLat %.2f:\t%.4f m',   max(long),
min(long), abs(long(1)-long(2)), min(lat),  ll2m([min(lat) min(lat)],[long(1) long(2)]),
max(lat),  ll2m([max(lat) max(lat)],[long(1) long(2)]) ))
```

```
disp(sprintf('\t :: y dimension:\n\t\tMax: %.2f Deg \n\t\tMin: %.2f Deg
\n\t\tSpacing:\t%.4f Deg \n\t\t\t\t%.4f m', max(lat),  min(lat),  abs(lat(1)-
lat(2)),ll2m([lat(1) lat(2)],[long(1) long(1)]) ))
disp(sprintf('\t :: z dimension:\n\t\tMax: %.2f m \n\t\tMin: %.2f m
',max(max(bathy)),min(min(bathy))))
disp(':::: :::: :::: :::: :::: :::: :::: :::: :::: :::: :::: :::: :::: :::: :::: :::::::')
bathy = double(bathy);
% bathy = bathy(1:5:end,1:5:end);
% long = long(1:5:end);
% lat=lat(1:5:end);
%% do interpolation, this is because some functions do not deal with NaNs
[ F , bathybu , bathy , x_bathy, y_bathy, long, lat ] =
scattInt(bathy,lat,long,long1,long2,lat1,lat2);

% plot it up, for visual comparison
figure(99)
clf
subplot(2,4,1)
imshade(long,lat, bathy);colorbar('SouthOutside')
title('Interpolated Bathy')
subplot(2,4,2)
imshade(long,lat, bathybu);colorbar('SouthOutside')
title('Raw Bathy')
%% Now do for gravity

if do_grav
   % import grav file
   [Glong,Glat, gravy]=grdread2(gravgridfile);
   gravy = double(gravy);
   % interpolate it
   [ G , gravybu , gravy, ~ , ~ , Glong, Glat ] =
scattInt(gravy,Glat,Glong,long1,long2,lat1,lat2);
   % now interpolate to make gravity map same size as bathymetry
   gravy = G(x_bathy,y_bathy);
   % plot the interpolated and the non
   figure(99)
   subplot(2,4,3)
   imshade(long,lat, gravy);colorbar('SouthOutside')
   title(sprintf('Interpolated %s',gravtype))
   subplot(2,4,4)
   imshade(Glong,Glat, gravybu);colorbar('SouthOutside')
   title(sprintf('Raw %s',gravtype))
   % now fft filter it
   HPF_gravy = fftfiltermap(gravy,lat,long,.4,.2);
   figure(99)
   subplot(2,4,8)
```

```
   imshade(Glong,Glat, HPF_gravy);colorbar('SouthOutside')
   title(sprintf('High Pass Filtered %s',gravtype))
end

%%
if dohpf
   if highpass2y > 0
      if removebig == 1
            HPF_bathy_forrem = directional_fftfiltermap(bathy,lat,long,...
               highpass1x,highpass2x,highpass1y,highpass2y);
            bathy2 = bathy - HPF_bathy_forrem;
            HPF_bathy = directional_fftfiltermap(bathy2,lat,long,...
               lowpass1x,lowpass2x,lowpass1y,lowpass2y);

      else
            HPF_bathy = directional_fftfiltermap(bathy,lat,long,...
               highpass1x,highpass2x,highpass1y,highpass2y);

      end


   else
      HPF_bathy = fftfiltermap(bathy,lat,long,highpass1x,highpass2x);
   end
   %bathy = HPF_bathy;
   Fbathy = bathy - HPF_bathy;
   figure(99)
   subplot(2,4,5)
   imshade(long,lat, Fbathy);colorbar('SouthOutside')
   title('Band Pass Filtered Bathy')
end
%% Now plot the results and compare to original
figure(999)
subplot(1,2,1)
imshade(long,lat, bathy);
title('Original')
subplot(1,2,2)
imshade(long,lat, Fbathy);
title(sprintf('Band Pass Filtered Bathy \n %.0f km - %.0f km EW & %.0f km - %.0f
km NS',highpass2x,lowpass1x,highpass2y,lowpass1y)  )
%% do the gradient
% using matlab mapping toolbox function 'gradientm'
% this needs the lat and long, and expect z data in meters
[aspect, slope, gradN, gradE] = gradientm(y_bathy,x_bathy, bathy);

%dirslope = gradN*(1-90/SD) + gradE*(90/SD);
```

```
dirslope = atand(gradN*cosd(SD) + gradE*sind(SD));
slopebu = slope;
slope = dirslope;
[ S ] = scattInt(dirslope,lat,long,long1,long2,lat1,lat2);

[Gaspect, Gslope, GgradN, GgradE] = gradientm(y_bathy,x_bathy, gravy);

% visual check of min max
%min(min(slope))
%max(max(slope))
%min(min(gradN))
%max(max(gradN))
%% plot pretty
dirslope = atand(gradN*cosd(SD)+gradE*sind(SD));
fig104=figure(104);
clf
subplot(1,2,1)
surf(long,lat,slopebu.*isfinite(bathybu));
view(0,90)
axis equal
axis tight
shading interp
% xlim([-44.98 -44.8])
% ylim([13.43 13.58 ])
hold on
title('Slope Map')
%title('Azimuth of Gradient','Fontsize',28,'Fontname','Ubuntu')
subplot(1,2,2)
surf(long,lat,abs(dirslope).*isfinite(bathybu));
view(0,90)
axis equal
axis tight
shading interp
% xlim([-44.98 -44.8])
% ylim([13.43 13.58 ])
hold on
title('Directional Slope Map')


fig104.PaperPositionMode = 'auto';
if print_y_n
print(fig104,sprintf('Azimuth_Prefilter_%s.png',sprintf('%s_%s',...
        HPFstr,...
        datestr(now,'mm_dd_yyyy'))),'-dpng','-r0')
end
```

```
%% Now take care of the axis
try % try loading an axis file, and plot it
   axis1=load(Axisfile);

   Ax = axis1(:,1);
   Ay = axis1(:,2);
   disp(sprintf('<<<<<<<<<< Using Axis data from %s',Axisfile))
   disp('<<<<<<<<<< <<<<<<<<< >>>>>>>>> >>>>>>>>>>')
   fig104=figure(104);
   clf
   surf(long,lat,bathybu);
   shading interp
   view(0 ,90)
   axis equal
   axis tight
   hold on
   plot(Ax,Ay,'r*')
   plot(Ax,Ay,'r--')
catch % otherwise, get user input
   fig104=figure(104);
   clf
   surf(long,lat,bathybu);
   shading interp
   view(0 ,90)
   lightangle(SD,1e-5)
   lightangle(SD+180,1e-5)
   axis equal
   axis tight
   hold on
   disp('<<<<<<<<<< Select Axis by clicking along it')
   disp('<<<<<<<<<< Press the Return key when done selecting points >>>>>>>>>')
   [Ax,Ay] = ginput;
   plot(Ax,Ay,'r*')
   disp('<<<<<<<<<< Axis Accepted, Thank you >>>>>>>>>>')
   disp('<<<<<<<<<< <<<<<<<<<<< >>>>>>>>>>> >>>>>>>>>>')
   savethisaxis = input(sprintf('Want to save this as "%s" ?\n(1 for yes, 0 for no)\n
',Axisfile))
   if savethisaxis
      file = [Ax,Ay];
      save(Axisfile,'file', '-ascii')
   end
end


%% plot EQs,
if strcmp(EQfname,'') ~= 1
```

```
   plotEQs(EQfname,lat1,lat2,long1,long2,SD)
end

%% filter out slopes with 'extreme slopes' & filter out slopes that face certain
directions
% create new copies, so we don't mess with original data
slope2=slope;
gradE2=gradE;
gradN2 =gradN;
aspect2=aspect;

disp(sprintf('<<<<<<<<<<< Slope Filter: Removing %.0f elements with slope greater
than %.1f',numel(find(slope>highslope)),highslope))
disp(sprintf('<<<<<<<<<<\tand %.0f elements with slope less than
%.1f\n',numel(find(slope<lowslope)),lowslope))
yes=find(abs(slope)>highslope | abs(slope) < lowslope);
slope2(yes)=NaN;
aspect2(yes)=NaN;
figure(101)
clf
subplot(1,2,1)
surf( long, lat, slope2);
hold on
plot(Ax,Ay,'r-')
colorbar('SouthOutside')
title('"Filtered by Aspect" Slope Map')
shading interp
view(0 ,90)
axis equal
%%
figure(101)
clf
subplot(1,2,1)
surf(slope2);
colorbar('SouthOutside')
title('"Filtered by Aspect" Slope Map')
shading interp
view(0 ,90)
axis equal
xlim([0 size(slope2,2)])
ylim([0 size(slope2,1)])

%% remove slopes that face the wrong ways
%%% 'Crude' Catch
if dohpf
disp('<<<<<<<<<<< Crude Boundary Filter: Getting rid of boundary effected slopes:')
```

```
   disp(sprintf('<<<<<<<<<<\tWhich Account for %.1f %% of total Identified
Slopes\n',100*sum(   [numel(isfinite(slope2(1:5,1:end)))
numel(isfinite(slope2((end-5):end,1:end)))    numel(isfinite(slope2(1:end,1:5)))
numel(isfinite(slope2(1:end,(end-5):end)))]   )/numel(isfinite(slope2))))
slope2(1:5,1:end) =NaN; slope2((end-5):end,1:end) = NaN; slope2(1:end,1:5) = NaN;
slope2(1:end,(end-5):end) = NaN;
aspect2(1:5,1:end) =NaN; aspect2((end-5):end,1:end) = NaN; aspect2(1:end,1:5) =
NaN; aspect2(1:end,(end-5):end) = NaN;
gradN2(1:5,1:end) =NaN; gradN2((end-5):end,1:end) = NaN; gradN2(1:end,1:5) =
NaN; gradN2(1:end,(end-5):end) = NaN;
gradE2(1:5,1:end) =NaN; gradE2((end-5):end,1:end) = NaN; gradE2(1:end,1:5) =
NaN; gradE2(1:end,(end-5):end) = NaN;
end
slope2 = slope2.*isfinite(bathybu);
aspect2 = aspect2.*isfinite(bathybu);
gradN2 = gradN2.*isfinite(bathybu);
gradE2 = gradE2.*isfinite(bathybu);

%%%
% figure(3);clf;surf((aspect2));view([0 90]);shading interp
% colorbar
%SPREADING DIRECTION IS 87 DEGREES
if exist('slopebu')==0
   % find east facers
   notE = find(aspect2<eastlowaz | aspect2>easthighaz);
   disp(sprintf('<<<<<<<<<<< Azimuth East Filter: Removing %.0f elements which
face not between Azimuth %.1f & %.1f',numel(notE),easthighaz,eastlowaz))
   disp(sprintf('<<<<<<<<<<\tWhich are %.1f & %.1f degrees different from the
spreading direction of %.1f\n',eastaz-eastlowaz,eastaz-easthighaz,eastaz))
   % now for west facers
   notW = find(aspect2<westlowaz | aspect2>westhighaz);
   disp(sprintf('<<<<<<<<<<< Azimuth West Filter: Removing %.0f elements which
face not between Azimuth %.1f & %.1f',numel(notW),westhighaz,westlowaz))
   disp(sprintf('<<<<<<<<<<\tWhich are %.1f & %.1f degrees different from the
spreading direction of %.1f\n',westaz-westlowaz,westaz-westhighaz,westaz))
else
   % find east facers
   notW = find(slope2<0);
   disp(sprintf('<<<<<<<<<<< Azimuth East Filter: Removing %.0f elements which
face West',numel(notW)))
   % now for west facers
   notE = find(slope2>0);
   disp(sprintf('<<<<<<<<<<< Azimuth West Filter: Removing %.0f elements which
face East',numel(notE)))
end
```

```
% allocate the new matrices
slopeE = abs(slope2);
slopeW = abs(slope2);
aspectE=aspect2;
aspectW=aspect2;
% assign values to them
slopeE(notE) = NaN;
slopeW(notW) = NaN;
aspectE(notE)=NaN;
aspectW(notW)=NaN;
%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
fig1 = figure(1);
clf
%Debbie: imagesc(long,lat,slopeE);axis xy; title('East facing slopes')
surf(long,lat,slopeE)
hold on
plot(Ax,Ay,'k*','LineWidth',3)
view([0 90])
shading interp
axis equal
xlim([long(1) long(end)])
ylim([lat(1) lat(end)])
title('East facing slopes')
% xlabel('Longitude (Degrees)')
% ylabel('Latitude (Degrees)')
cb=colorbar('Location','South','Fontsize',10);
cbPos = cb.Position;cbPos(4) = .5*cbPos(4);cb.Position = cbPos;
set(gca,'Fontsize',18,'Fontname','Ubuntu')
hold off
fig1.PaperPositionMode = 'auto';
if print_y_n
print(fig1,sprintf('E_Slopes__%s.png',sprintf('%s_SL%.1f_%.1f_Waz%d_%d_Eaz%
d_%d_%s',...
        HPFstr,...
        lowslope,highslope,...
        westlowaz,westhighaz,...
        eastlowaz,easthighaz,...
        datestr(now,'mm_dd_yyyy'))),'-dpng','-r0')
end

%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
fig2=figure(2);
```

```
clf
%Debbie: imagesc(long,lat,slopeW); axis xy; title ('West facing slopes');
surf(long,lat,slopeW)
hold on
plot(Ax,Ay,'k--','LineWidth',4)
view([0 90])
shading interp
axis equal
xlim([long(1) long(end)])
ylim([lat(1) lat(end)])
title('West facing slopes')
% xlabel('Longitude (Degrees)')
% ylabel('Latitude (Degrees)')
cb=colorbar('Location','South','Fontsize',10);
cbPos = cb.Position;cbPos(4) = .5*cbPos(4);cb.Position = cbPos;
set(gca,'Fontsize',18,'Fontname','Ubuntu')
hold off
% write these to file, if want
% grdwrite2(long,lat, slopeE,'AllEast20_60.grd');
% grdwrite2(long,lat, slopeW, 'AllWest20_60.grd');
fig2.PaperPositionMode = 'auto';
if print_y_n
print(fig2,sprintf('W_Slopes__%s.png',sprintf('%s_SL%.1f_%.1f_Waz%d_%d_Eaz%
d_%d_%s',...
        HPFstr,...
        lowslope,highslope,...
        westlowaz,westhighaz,...
        eastlowaz,easthighaz,...
        datestr(now,'mm_dd_yyyy'))),'-dpng','-r0')
end


%% seperate axis into overlapping parts and create new axis matrix
axis1 = axisbreaker(axis1);
% intitialize these
slopeE_Wside=slopeE;
slopeE_Eside=slopeE;
slopeW_Wside=slopeW;
slopeW_Eside=slopeW;

clear axis_interp
[axis_interp,slopeE_Eside,slopeW_Wside,slopeE_Wside,slopeW_Eside] =
axisinterpolater(axis1,long,lat,slopeE_Eside,slopeW_Wside,slopeE_Wside,slopeW_E
side);

%% plot new axis up
```

```
fig3 = figure(3);
clf
hold on
surf(long,lat,slopeW_Wside)
for m = 1:2:size(axis_interp,2)
   plot(axis_interp(:,m),axis_interp(:,m+1),'-','Color',[.5 .5 .5],'LineWidth',1.3)
end
view([0 90])
shading interp
axis equal
xlim([long(1) long(end)])
ylim([lat(1) lat(end)])
title('West Facing Slopes West of Ridge Axis')
% xlabel('Longitude (Degrees)')
% ylabel('Latitude (Degrees)')
cb=colorbar('Location','South','Fontsize',10);
cbPos = cb.Position;cbPos(4) = .5*cbPos(4);cb.Position = cbPos;
set(gca,'Fontsize',18,'Fontname','Ubuntu')
fig3.PaperPositionMode = 'auto';
if print_y_n
print(fig3,sprintf('W_W_Slopes__%s.png',sprintf('%s_SL%.1f_%.1f_Waz%d_%d_E
az%d_%d_%s',...
          HPFstr,...
          lowslope,highslope,...
          westlowaz,westhighaz,...
          eastlowaz,easthighaz,...
          datestr(now,'mm_dd_yyyy'))),'-dpng','-r0')
end
%% plot up slopes
fig4 = figure(4);
clf
surf(long,lat,slopeE_Eside)
hold on
for m = 1:2:size(axis_interp,2)
   plot(axis_interp(:,m),axis_interp(:,m+1),'-','Color',[.5 .5 .5],'LineWidth',1.3)
end
   view([0 90])
shading interp
axis equal
xlim([long(1) long(end)])
ylim([lat(1) lat(end)])
title('East Facing Slopes East of Ridge Axis')
cb=colorbar('Location','South','Fontsize',10);
cbPos = cb.Position;cbPos(4) = .5*cbPos(4);cb.Position = cbPos;
set(gca,'Fontsize',18,'Fontname','Ubuntu')
fig4.PaperPositionMode = 'auto';
```

```
if print_y_n
print(fig4,sprintf('E_E_Slopes__%s.png',sprintf('%s_SL%.1f_%.1f_Waz%d_%d_Eaz
%d_%d_%s',...
        HPFstr,...
        lowslope,highslope,...
        westlowaz,westhighaz,...
        eastlowaz,easthighaz,...
        datestr(now,'mm_dd_yyyy'))),'-dpng','-r0')
end

clear slopeW_Wcc slopeE_Wcc slopeW_Ecc slopeE_Ecc
slopeW_Wcc = bwconncomp(slopeW_Wside>0,connectivity);
slopeW_Ecc = bwconncomp(slopeW_Eside>0,connectivity);
slopeE_Ecc = bwconncomp(slopeE_Eside>0,connectivity);
slopeE_Wcc = bwconncomp(slopeE_Wside>0,connectivity);
% which properties do you want?
propsofinterest = {  'Centroid','Area','Orientation',...
                'MajorAxisLength','MinorAxisLength',...
                'Eccentricity','Orientation'};
slopeE_Wcc.rps = regionprops(slopeE_Wcc,propsofinterest);
slopeW_Wcc.rps = regionprops(slopeW_Wcc,propsofinterest);
slopeE_Ecc.rps = regionprops(slopeE_Ecc,propsofinterest);
slopeW_Ecc.rps = regionprops(slopeW_Ecc,propsofinterest);
if cutoff>0
   disp('<<<<<<<<<< Using Cutoff >>>>>>>>>>')
   disp(sprintf('\tCutoff is %.0f\n\tCutting:',cutoff))
   disp(sprintf('\t\t%.0f West Facing Slopes, East of
Axis',numel(find([slopeW_Ecc.rps.Area]>cutoff))))
   disp(sprintf('\t\t%.0f East Facing Slopes, West of
Axis',numel(find([slopeE_Ecc.rps.Area]>cutoff))))
   disp(sprintf('\t\t%.0f West Facing Slopes, West of
Axis',numel(find([slopeW_Wcc.rps.Area]>cutoff))))
   disp(sprintf('\t\t%.0f East Facing Slopes, West of
Axis',numel(find([slopeE_Wcc.rps.Area]>cutoff))))
   disp('<<<<<<<<<< <<<<<<<<< >>>>>>>>> >>>>>>>>>>')
elseif cutofflength>0
   disp('<<<<<<<<<< Using CutoffLength >>>>>>>>>>')
   disp(sprintf('\tCutoffLength is %.0f\n\tCutting:',cutofflength))
   disp(sprintf('\t\t%.0f West Facing Slopes, East of
Axis',numel(find([slopeW_Ecc.rps.MajorAxisLength]<cutofflength))))
   disp(sprintf('\t\t%.0f East Facing Slopes, West of
Axis',numel(find([slopeE_Ecc.rps.MajorAxisLength]<cutofflength))))
   disp(sprintf('\t\t%.0f West Facing Slopes, West of
Axis',numel(find([slopeW_Wcc.rps.MajorAxisLength]<cutofflength))))
   disp(sprintf('\t\t%.0f East Facing Slopes, West of
Axis',numel(find([slopeE_Wcc.rps.MajorAxisLength]<cutofflength))))
```

```
   disp('<<<<<<<<<< <<<<<<<<< >>>>>>>>> >>>>>>>>>>')
end
%%
fig6 = figure(6);
clf
hold on
large_slopeW_Wside = slopeW_Wside*NaN;
for n=1:slopeW_Wcc.NumObjects
   if cutoff>0
   if slopeW_Wcc.rps(n).Area>cutoff

large_slopeW_Wside(slopeW_Wcc.PixelIdxList{n})=slopeW_Wside(slopeW_Wcc.
PixelIdxList{n});
   end
   elseif cutofflength>0
   if slopeW_Wcc.rps(n).MajorAxisLength>cutofflength &
slopeW_Wcc.rps(n).MinorAxisLength>cutofflength2

large_slopeW_Wside(slopeW_Wcc.PixelIdxList{n})=slopeW_Wside(slopeW_Wcc.
PixelIdxList{n});
   end
   end
end
surf(long,lat,large_slopeW_Wside)
hold on
for m = 1:2:size(axis_interp,2)
   plot(axis_interp(:,m),axis_interp(:,m+1),'--','Color',[.2 .2 .2],'LineWidth',2)
end
view([0 90])
shading interp
axis equal
xlim([long(1) long(end)])
ylim([lat(1) lat(end)])
title('West Facing Slopes West of Ridge Axis')
cb=colorbar('Location','South','Fontsize',10);
cbPos = cb.Position;cbPos(4) = .5*cbPos(4);cb.Position = cbPos;
set(gca,'Fontsize',18,'Fontname','Ubuntu')
hold off
fig6.PaperPositionMode = 'auto';
if print_y_n
print(fig6,sprintf('W_W_Slopes__%s.png',app_title),'-dpng','-r0')
end
%%
fig7 = figure(7);
clf
hold on
```

```
large_slopeE_Eside = slopeE_Eside*NaN;
for n=1:slopeE_Ecc.NumObjects
   if cutoff>0
   if slopeE_Ecc.rps(n).Area>cutoff

large_slopeE_Eside(slopeE_Ecc.PixelIdxList{n})=slopeE_Eside(slopeE_Ecc.PixelId
xList{n});
   end
   elseif cutofflength>0
   if slopeE_Ecc.rps(n).MajorAxisLength>cutofflength &
slopeE_Ecc.rps(n).MinorAxisLength>cutofflength2

large_slopeE_Eside(slopeE_Ecc.PixelIdxList{n})=slopeE_Eside(slopeE_Ecc.PixelId
xList{n});
   end
   end
end
surf(long,lat,large_slopeE_Eside)
hold on
for m = 1:2:size(axis_interp,2)
   plot(axis_interp(:,m),axis_interp(:,m+1),'-','Color',[.5 .5 .5],'LineWidth',1.3)
end
view([0 90])
shading interp
axis equal
xlim([long(1) long(end)])
ylim([lat(1) lat(end)])
title('East Facing Slopes East of Ridge Axis')
cb=colorbar('Location','South','Fontsize',10);
cbPos = cb.Position;cbPos(4) = .5*cbPos(4);cb.Position = cbPos;
set(gca,'Fontsize',18,'Fontname','Ubuntu')
hold off
fig7.PaperPositionMode = 'auto';
if print_y_n
print(fig7,sprintf('E_E_Slopes__%s.png',app_title),'-dpng','-r0')
end
%%  Get user picks
% first plot up the map
cmin = min(min(bathybu));
cmax = max(max(bathybu));

clear s s1
cmaplim=64;
figure(9)
clf
```

```
s=surf(long,lat,bathybu,'CData',round((-1+cmaplim)*(bathybu-cmin)/(cmax-
cmin))+1,'cdatamapping','direct');
hold on
s1(1)=surf(long,lat,large_slopeE_Eside*0,'CData',large_slopeE_Eside*0+(cmaplim+
1),'cdatamapping','direct','Facealpha',0.5);%,'cdatamapping','direct',
s1(2)=surf(long,lat,large_slopeW_Wside*0,'CData',large_slopeW_Wside*0+(cmapli
m+1),'cdatamapping','direct','Facealpha',0.5);
hold on

set(gcf,'Colormap',[parula(cmaplim);[0 0 0];[0 1 0];[1 0 0]])
view(0,90)
axis equal
axis tight
shading interp
for m = 1:2:size(axis_interp,2)
    plot(axis_interp(:,m),axis_interp(:,m+1),'-r','LineWidth',1.3)
end
lightangle(SD+90,1e-100)
lightangle(SD-90,1e-100)
s.FaceLighting = 'gouraud';
s.AmbientStrength = 0.2;
s.DiffuseStrength = .3;
s.SpecularStrength = .1;
s.SpecularExponent = .1;
s.BackFaceLighting = 'lit'; %'reverselit' | 'unlit' | 'lit'
set(gca,'Fontsize',20,'Fontname','Ubuntu')
title('Bathymetry Overlain by Filtered Slopes')

%% now ask for points

% these OCC points are mine, set to 0 to pick your own
if 1

disp('%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%')
    disp('Using Mark Larson Picks')
    disp('Go to lime 647 and set to "0" to pick your own')
    CCs = [ -45.0262    15.08
           -45.1       14.875
           -44.93      14.843
           -44.9100    14.6650
           -44.8396    13.8339
           -45.0800    13.6980
           -44.9500    13.5150
           -44.9500    13.3200
           -44.9000    13.1040
```

```
          -44.7053    12.8254];
else
   [Cx,Cy] = ginput;
   CCs = [Cx Cy];
end

%% get bounding faults
% because the bounding faults are not that important for manual picks,
% you don't need to always
if input('Use axis as EBF and WBF for now? ( "1" for YES else NO) :\n       ')
   Ebf_interp = axis_interp;
   Wbf_interp = axis_interp;
else
try
   Wbf=load(WestBoundingFaults);
   Ebf=load(EastBoundingFaults);
   disp(sprintf('<<<<<<<<<< Using Bounding Fault data from %s &
%s',WestBoundingFaults,EastBoundingFaults))
   disp('<<<<<<<<<< <<<<<<<<< >>>>>>>>> >>>>>>>>>>')
   fig104=figure(104)
   clf
   imshade(long,lat, bathybu);
   hold on
   plot(Ebf(:,1),Ebf(:,2),'r*')
   plot(Wbf(:,1),Wbf(:,2),'b*')
   Ebf = axisbreaker(Ebf);
   Ebf_interp = interpolateaxis(lat,long,Ebf);
   Wbf = axisbreaker(Wbf);
   Wbf_interp = interpolateaxis(lat,long,Wbf);
catch
   GetBoundingFault
   if input('Want to save these bounding faults?')
      disp('!!!!!!!!! Don"t forget the quotes !!!!!!!!!!')
      WBFfilename = input(sprintf('Name for West Bounding Fault?\n:'));
      save(WBFfilename,'Wbf_interp','-ascii')
      disp('!!!!!!!!! Don"t forget the quotes !!!!!!!!!!')
      EBFfilename = input(sprintf('Name for East Bounding Fault?\n:'));
      save(EBFfilename,'Ebf_interp','-ascii')
      clear EBFfilename WBFfilename
   end
end
end

clear slopes

fig8=figure(8);
```

```
clf
%imshade(long,lat, bathybu);
s=surf(long,lat,bathybu);
% lightangle(100,1e-5)
% lightangle(-80,1e-5)
% light
shading interp
view([0 90])
lightangle(90,1e-2)
lightangle(-60,1e-2)
s.FaceLighting = 'flat';
s.AmbientStrength = 0.3;
s.DiffuseStrength = .6;
s.SpecularStrength = 01;
s.SpecularExponent = 1;
s.BackFaceLighting = 'lit'; %'reverselit' | 'unlit' | 'lit'
hold on
for m = 1:2:size(axis_interp,2)
   plot(axis_interp(:,m),axis_interp(:,m+1),'-m','LineWidth',1.3)
end
shading interp
view([0 90])
axis equal
axis tight
%title(sprintf('Major and Minor Axis of Filtered Slopes \nw/Profile Numbers and
profiles drawn'))
%  xlabel('Longitude (Degrees)')
%  ylabel('Latitude (Degrees)')
title('Profile Locations')
 set(gca,'Fontsize',18,'Fontname','Ubuntu')
%%%
H = gca;
for hs = 1:length(H.XTickLabel)
   dotidx = strfind(H.XTickLabel{hs},'.');
   if length(dotidx) == 1
      H.XTickLabel{hs} = sprintf('%s%c %.0f"',H.XTickLabel{hs}(1:dotidx-
1),char(176),str2num(H.XTickLabel{hs}(dotidx:end))*60);
   else
      H.XTickLabel{hs} = sprintf('%s%c 00"',H.XTickLabel{hs},char(176));
   end
end
for hs = 1:length(H.YTickLabel)
   dotidx = strfind(H.YTickLabel{hs},'.');
   if length(dotidx) == 1
      H.YTickLabel{hs} = sprintf('%s%c %.0f"',H.YTickLabel{hs}(1:dotidx-
1),char(176),str2num(H.YTickLabel{hs}(dotidx:end))*60);
```

```
    else
        H.YTickLabel{hs} = sprintf('%s%c 00"',H.YTickLabel{hs},char(176));
    end
end

%%
% Now Plot major axis, minor axis, profile number and profiles
% on the bathy map
% Reminder, we get data from original map not from any filter
%
ridgedistance = 30;
np=1;
p2m = length(lat)/ll2m([lat(1) lat(end)],[long(1) long(1)]);
% divide by 2 because this becomes half the length of the axis
p2l = abs(long(end) - long(1))/length(long)/2;
numprofiles = 0;
for n=1:size(CCs,1)
    if CCs(n,2) < 14
        SD = 93.4;
    else
        SD = 102.9;
    end
    disp(sprintf('Spreading Direction used: %.1f',SD))
   [val C1] = min(abs(long - CCs(n,1)));
   [val C2] = min(abs(lat - CCs(n,2)));

    slopes.rps(n).Centroid = [C1 C2];

    slopes.rps(n).DistFromAxis = ll2m( [ lat(round(slopeE_Ecc.rps(n).Centroid(2)))...
            axis_interp(round(slopeE_Ecc.rps(n).Centroid(2)),2) ],...
           [ long(round(slopeE_Ecc.rps(n).Centroid(1))) ...
            axis_interp(round(slopeE_Ecc.rps(n).Centroid(2)),1) ])*1e-3 ;


    slopes.rps(n).MajorAxisLength = 150;
    slopes.rps(n).MinorAxisLength = 20;
    slopes.rps(n).Area = 20;

    x = slopes.rps(n).MajorAxisLength*cosd(SD);
    y = slopes.rps(n).MajorAxisLength*sind(SD);
    xs = p2l*x*[-1 1] + long(round(slopes.rps(n).Centroid(1)));
    ys = p2l*y*[1 -1] + lat(round(slopes.rps(n).Centroid(2)));
    slopes.rps(n).MajorAxisTop = [xs(1) ys(1)];
    slopes.rps(n).MajorAxisBot = [xs(2) ys(2)];
    plot(   xs,...
            ys,...
```

```
            '-r','LineWidth',2)
        text(-p2l*x + long(round(slopes.rps(n).Centroid(1))),...
            p2l*y + lat(round(slopes.rps(n).Centroid(2))),...
            sprintf('%d',n),'Color','k','FontSize',15)
        x =
slopes.rps(n).MinorAxisLength*cosd(SD+90);%slopeW_Wcc.rps(n).Orientation+90)
;
        y =
slopes.rps(n).MinorAxisLength*sind(SD+90);%slopeW_Wcc.rps(n).Orientation+90);
        plot(   p2l*x*[-1 1] + long(round(slopes.rps(n).Centroid(1))),...
                p2l*y*[1 -1] + lat(round(slopes.rps(n).Centroid(2))),...
                '-r','LineWidth',2)
    [slopes.profile(n).long slopes.profile(n).lat slopes.profile(n).Axis corner1 corner2 ]
= Centroid_spreadingprofile(bathy,axis_interp,slopes,lat,long,n,SD);
    plot(slopes.profile(n).long, slopes.profile(n).lat,...
            '--','Color',[1 0 0],'LineWidth',1.5);
        slopes.profile(n).corner1 = [ corner1(1,1) corner1(end,1) corner2(end,1)
corner2(1,1) corner1(1,1) ];
        slopes.profile(n).corner2 = [ corner1(1,2) corner1(end,2) corner2(end,2)
corner2(1,2) corner1(1,2) ];
        slopes.profile(n).zs = F(slopes.profile(n).long, slopes.profile(n).lat);
        slopes.gravprof(n).mgals = G(slopes.profile(n).long, slopes.profile(n).lat);
end

%%
% intialize
Profslopes = slopes;

%% Now we make the profile structure,
% this is done manaully with OnePlot_byHand
% or semi automated with other .m files
for n =1:size(CCs,1)
    axis_to_use = whichaxis(axis_interp,long(round(slopes.rps(n).Centroid(1))
),lat(round(slopeW_Wcc.rps(n).Centroid(2))),lat);
    % try disp('Using Already Picked!')
     %  [figi tempprof] =
OnePlot_byHand(1,'orthographic',90,Profslopes,slope,n,n,gravtype,lat,long,bathybu,g
ravy,axis_to_use,Wbf_interp,Ebf_interp,F,G,S,5,0,200);
     %catch
        [figi1, figi2, figi3, figi4, tempprof] =
OnePlot_byHand(1,'orthographic',90,slopes,slope,n,n,gravtype,lat,long,bathybu,gravy
,axis_to_use,Wbf_interp,Ebf_interp,F,G,S,3,0,200);
        disp('Finished with the Profile!')
        for js = length(tempprof.profile(n).pick)
            Profslopes.profile(n).pick(js) = tempprof.profile(n).pick(js);
        end
```

```
    % end
    if print_y_n
        cd proposal' sec4'/
        disp('Saving, please patience')
        disOCC = num2str(CCs(n,2));
        figi1.PaperPositionMode = 'auto'
        figi2.PaperPositionMode = 'auto'
        figi3.PaperPositionMode = 'auto'
        figi4.PaperPositionMode = 'auto'
%         figi1.PaperUnits = 'inches';
%         figi1.PaperSize = [11 8.5];
%         figi1.PaperPosition = [.0 .0 [11 8.5]-0.5];
        filename1 =
sprintf('%d_1_%s_%sN_Nov16_proposal',n,disOCC(1:2),disOCC(4:5));
%%'%s_%sN_%d_DistVSOutRot_Aug17_wShift_wM_wResSqr_wInversion.png',di
sOCC(1:2),disOCC(4:5),n);
        filename2 =
sprintf('%d_2_%s_%sN_Nov16_proposal',n,disOCC(1:2),disOCC(4:5));
%%'%s_%sN_%d_DistVSOutRot_Aug17_wShift_wM_wResSqr_wInversion.png',di
sOCC(1:2),disOCC(4:5),n);
        filename3 =
sprintf('%d_3_%s_%sN_Nov16_proposal',n,disOCC(1:2),disOCC(4:5));
%%'%s_%sN_%d_DistVSOutRot_Aug17_wShift_wM_wResSqr_wInversion.png',di
sOCC(1:2),disOCC(4:5),n);
        filename4 =
sprintf('%d_4_%s_%sN_Nov16_proposal',n,disOCC(1:2),disOCC(4:5));
%%'%s_%sN_%d_DistVSOutRot_Aug17_wShift_wM_wResSqr_wInversion.png',di
sOCC(1:2),disOCC(4:5),n);
%         filenum = num2str(CCs(n,2));
%         filename =
sprintf('%s_%sN_w3Dmisfit_Oct5.png',filenum(1:2),filenum(4:5));
        print(figi1,filename1,'-depsc')
        close(figi1)
        print(figi2,filename2,'-depsc')
        close(figi2)
        print(figi3,filename3,'-depsc')
        close(figi3)
        print(figi4,filename4,'-dpng','-r300')
        close(figi4)
        cd ..
    end
    disp(' *******  Done  ********')
    close all
    pause(1)
end
disp('%%%%%%%%%%%%%%%% All Done %%%%%%%%%%%%%%%%')
```

6.17 OkinoRateCalc.m

```
function RateOut =
OkinoRateCalc(LON,LAT,MODEL,MOVINGPLATE,FIXEDPLATE)
% example RateOut = OkinoRateCalc(-20,-25,'MORVEL','nb','pa')
% INPUTS
% This could be a structure, if you want to change the relative plates used
% LON = longitude
% LAT = latitude
% MODEL = model to use, note that this changes the necessity of the plates
%         MORVEL is most used, i think
% MOVINGPLATE = choose from here:
%             http://ofgs.aori.u-tokyo.ac.jp/~okino/global_plate_geom.jpg
% FIXEDPLATE = not used in absolute.

% MODEL List
% NUVEL-1:(relative motion, Pacific plate fixed)
% NUVEL-1A:(relative motion, Pacific plate fixed)
% Please note that the model parameters for Philippine Sea Plate are based on Seno et
al. (JGR, 1993)
% Please note that the velocity calculated based on NUVEL-1 and NUVEL-1A may
be 4.5% and <2% faster than those measured by space geodetic methods by using
VLBI/SLR (Gordon, Nature 1993)
% NNR-NUVEL-1:(absolute plate motion, no-net rotation)
% NNR-NUVEL-1A:(absolute plate motion, no-net rotation)
% HS3-NUVEL-1A:(absolute plate motion, relative to hotspot frame)
% MORVEL: (new relative motion model for 25 tectonic plates, spreading rates and
fault azimuths are used to determine the motions of 19 plates, and GPS station
velocites and azimuthal data for 6 smaller plates with little or no connection to the
mid-ocean ridges)
% NNR-MORVEL: (absolute plate motion, no-net rotation, for MORVEL 25 plates
and Bird(2003) 's 31 plates)

% the url
okino_rate_calc_url = 'http://ofgs.aori.u-tokyo.ac.jp/~okino/rate_calc_new2012.cgi';
% set up our request form
FormData = [ ...
   {  'model'        ;  MODEL      } ; ...
   {  'movingplate'  ;  MOVINGPLATE } ; ...
   {  'fixed'        ;  FIXEDPLATE  } ; ...
   {  'lon'          ;  num2str(LON)} ; ...
   {  'lat'          ;  num2str(LAT)}];
% urlread is very easy to use!
page = urlread(okino_rate_calc_url,'post',FormData);
```

```
% what we want
peices = {'plate velocity' , 'direction' , 'north component of velocity' , 'east component
of velocity ' };
for n = 1:length(peices)
   idx = strfind(page,peices{n});
   startblock = find(page(idx:end) == '[');
   endblock = find(page(idx:end) == '>');
   val(n) = str2num(page(idx+endblock(1):idx+startblock(1)-2));
end
% construct output structure
RateOut.V_total = val(1);
RateOut.DegFromNorth = val(2);
RateOut.V_north = val(3);
RateOut.V_east = val(4);
```

6.18 plotBestfitModel_onSlope.m

```
% plotBestfitModel_onSlope
fig = figure(fetnum*100000+2);
clf
hold on
%% plot topo

% first assign some constants
minZ = min(Depths);

sp1 = subplot(2,1,1);
hold on
title(sprintf('Feature %g',fet),'Fontsize',15)
VE = 2;
p0=plot(ProPick.profdist,(ProPick.zs- minZ)*VE,'Linewidth',3);

tk2=1;
for tk = 1:length(Distances)
   plot( Distances(tk),...
         (Depths(tk) - minZ)*VE,'o',...
         'MarkerFacecolor',colorN(tk,:),...
         'MarkerEdgecolor',[1 1 1],...
         'MarkerSize',7)
   try
   if round(abs(Distances(tk))) == abs(round(ProPick.fittingdist(tk2)*1e3))
      plot( Distances(tk),...
         (Depths(tk) - minZ)*VE,'o',...
         'MarkerFacecolor',colorN(tk,:),...
         'MarkerEdgecolor',[0 0 0],...
         'MarkerSize',7)
      tk2=tk2+1;
```

```
   end
   catch

   end
end

axis equal
%%

side = sign(Distances(2))*1e3;
if side > 0
   side2 = 1;
else
   side2 = length(Depths);
end
%%% for posterity
% get a distance vector % dis = TES(HEidx,ANidx,IFidx,CTidx).dist(:,teidx); % tdis
= TES(HEidx,ANidx,IFidx,CTidx).topodist(:,teidx);
% get an id for start of footwall
%stID = length(TES(HEidx,ANidx,IFidx,CTidx).topo(:,teidx))-
length(TES(HEidx,ANidx,IFidx,CTidx).slope(:,teidx));
%SlopeBFdist =
linspace(0,TES(HEidx,ANidx,IFidx,CTidx).dist(end,teidx),length(TES(HEidx,ANidx
,IFidx,CTidx).topo)) + offdistX(offdistidx);
%%%

% First plot the weighted model
WeightedModelDistances =
side*(TES(HEidxW,ANidxW,IFidxW,CTidxW).topodist(:,teidxW) +
offdistX(offdistidxW));

[val idx] = min(abs( WeightedModelDistances - ...
                side*min(ProPick.fittingdist)  ));

%%%%%%%%%%%%%%%%%%%% IF FIXED TO FIRST FOOTWALL POINT
%%%%%%%%%%%%%%%%%%%%
% WeightedModelDepths =
(TES(HEidxW,ANidxW,IFidxW,CTidxW).topo(:,teidxW) + ...
%     (diff( [ min(TES(HEidxW,ANidxW,IFidxW,CTidxW).topo(:,teidxW))
min(Depths) ] ) - minZ));
%%%%%%%%%%%%%%%%%%%% IF FIXED TO FIRST FITTED POINT
%%%%%%%%%%%%%%%%%%%%
WeightedModelDepths = TES(HEidxW,ANidxW,IFidxW,CTidxW).topo(:,teidxW) -
...
    TES(HEidxW,ANidxW,IFidxW,CTidxW).topo(idx,teidxW) - minZ -
abs(min(ProPick.fittingtopo));
```

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

pW = plot(   WeightedModelDistances,...
       WeightedModelDepths*VE,...
       '-','Color',[.5 .5 .5 .5],'Linewidth',5);
   %%
if doAll
% get a distances vector for the slope model, we must offset it, and
% multiply by the side (-1 if on West, 1 if on East)
SlopeModelDistances = side*(TES(HEidx,ANidx,IFidx,CTidx).dist(:,teidx)  +
offdistX(offdistidx));
[val idx] = min(abs( SlopeModelDistances - ...
                side*min(ProPick.fittingdist)  ));
% find the elevation to offset out Slope model by
% then construct elevation vector
%%%%%%%%%%%%%%%%%%%% IF FIXED TO FIRST FOOTWALL POINT
%%%%%%%%%%%%%%%%%%%%
% SlopeModelDepths = (  TES(HEidx,ANidx,IFidx,CTidx).topo(:,teidx) + ...
%     (diff([ min(TES(HEidx,ANidx,IFidx,CTidx).topo(:,teidx)) min(Depths) ]) -
minZ));
%%%%%%%%%%%%%%%%%%%% IF FIXED TO FIRST FITTED POINT
%%%%%%%%%%%%%%%%%%%%
SlopeModelDepths = TES(HEidx,ANidx,IFidx,CTidx).topo(:,teidx) - ...
    TES(HEidx,ANidx,IFidx,CTidx).topo(idx,teidx) - minZ -
abs(min(ProPick.fittingtopo));
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%


% do it for Topo model too
TopoModelDistances =
side*(TES(HEidxT,ANidxT,IFidxT,CTidxT).topodist(:,teidxT) +
offdistX(offdistidxT));
[val idx] = min(abs( TopoModelDistances - ...
                side*min(ProPick.fittingdist)  ));
%%%%%%%%%%%%%%%%%%%% IF FIXED TO FIRST FOOTWALL POINT
%%%%%%%%%%%%%%%%%%%%
% TopoModelDepths = (TES(HEidxT,ANidxT,IFidxT,CTidxT).topo(:,teidxT) + ...
%     (diff([min(TES(HEidxT,ANidxT,IFidxT,CTidxT).topo(:,teidxT)) min(Depths) ])
- minZ));
%%%%%%%%%%%%%%%%%%%% IF FIXED TO FIRST FITTED POINT
%%%%%%%%%%%%%%%%%%%%
TopoModelDepths = TES(HEidxT,ANidxT,IFidxT,CTidxT).topo(:,teidxT) - ...
   TES(HEidxT,ANidxT,IFidxT,CTidxT).topo(idx,teidxT) - minZ -
abs(min(ProPick.fittingtopo));
```

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

pS = plot(   SlopeModelDistances,...
      SlopeModelDepths*VE,...
      '-','Color',[1 0 1 .7],'Linewidth',2);

pT = plot(   TopoModelDistances,...
      TopoModelDepths*VE,...
      '-','Color',[0 1 0 .7],'Linewidth',1.5);

l=legend([p0,pW,pS,pT],'Obervation','Best-Fit Weighted Model','Best-Fit Slope
Model','Best-fit Topography
Model','Location','SouthOutside','Orientation','horizontal')
legend BOXOFF
end
%% plot depths to fault root for models

% find axis
[v Axid]=min(abs(ProPick.profdist));
% find some made up error for axis elevation
FaultRootdepth = abs(ProPick.zs(Axid)) - abs(mean(ProPick.zs(Axid-5:Axid+5))) -
6500;
if doAll
[vT idT] = min(TopoModelDepths);

TopoM_FR =  vT - abs(TopoModelDistances(idT))*tand(angles(ANidxT)) ;

[vS idS] = min(SlopeModelDepths);

SlopeM_FR = vS - abs(SlopeModelDistances(idS))*tand(angles(ANidx)) ;

plot([ SlopeModelDistances(idS) 0],[vS SlopeM_FR]*VE,'--','Color',[1 0 1 .7])
plot([ TopoModelDistances(idT) 0],[vT TopoM_FR]*VE,'--','Color',[0 1 0 .7])
% if round(mean([vT TopoM_FR]*VE)*1e-3) == round(mean([vS
SlopeM_FR]*VE)*1e-3)
%    t= text(-1*side,mean([vS SlopeM_FR]*VE)*.75,sprintf('Slope Model z_r: %.1f
km',abs(SlopeM_FR*1e-3)));
% else
%    t= text(-1*side,mean([vS SlopeM_FR]*VE),sprintf('Slope Model z_r: %.1f
km',abs(SlopeM_FR*1e-3)),'HorizontalAlignment','right');
% end
end

[vW idW] = min(WeightedModelDepths);
```

```
WeightedM_FR = vW - abs(WeightedModelDistances(idW))*tand(angles(ANidxW))
;

plot([ WeightedModelDistances(idW) 0],[vW WeightedM_FR]*VE,...
      '--','Color',[.5 .5 .5 .5],'Linewidth',4.5)

% yval = mean([vW WeightedM_FR]*VE);
%
% if doAll
%     if yval == mean([vT TopoM_FR]*VE) | yval == mean([vS SlopeM_FR]*VE)
%         yval = yval/2;
%
%     end
% end

yval = min([WeightedM_FR SlopeM_FR TopoM_FR]*VE);
if side>0
   alignment = 'right';
else
   alignment = 'left'
end
text(-1*side,yval*.25,sprintf('Topo Model %s_r: %.1f
km','\it{z}\rm',abs(TopoM_FR*1e-3)),'HorizontalAlignment',alignment)
text(-1*side,yval*.5,sprintf('Slope Model %s_r: %.1f
km','\it{z}\rm',abs(SlopeM_FR*1e-3)),'HorizontalAlignment',alignment)
text(-1*side,yval*.75,sprintf('Weighted Model %s_r: %.1f
km','\it{z}\rm',abs(WeightedM_FR*1e-3)),'HorizontalAlignment',alignment)

% plot axis
plot([ 0 0 ], [ -1e4 2e3],'--k','Linewidth',1.5 )

%% finally, set xlims & ylims

if doAll
[v MSid]=max(abs(TES(HEidx,ANidx,IFidx,CTidx).topo(:,teidx)));
[v MTid]=max(abs(TES(HEidxT,ANidxT,IFidxT,CTidxT).topo(:,teidxT)));

xlim([  min([ min(ProPick.profdist) TopoModelDistances(MTid)
SlopeModelDistances(MSid)  ]) ...
      max([max(ProPick.profdist) TopoModelDistances(MTid)
SlopeModelDistances(MSid) ]) ])

ylim(VE*[ min([min((Depths(tk) - minZ )) min(TopoM_FR) min(SlopeM_FR)]) ...
   max([max((ProPick.zs - minZ )) max(TopoModelDepths)
max(SlopeModelDepths)]) ...
      ])
```

```
else

[v MWid]=max(abs(TES(HEidxW,ANidxW,IFidxW,CTidxW).topodist(:,teidxW)));

xlim([  min([ min(ProPick.profdist) WeightedModelDistances(MWid)  ]) ...
       max([max(ProPick.profdist) WeightedModelDistances(MWid)  ]) ])

ylim(VE*[ min([min((Depths(tk) - minZ )) min(WeightedM_FR) ]) ...
   max([max((ProPick.zs - minZ )) max(WeightedModelDepths) ]) ...
       ])
end
ylabel(sprintf('Pseudodepths (normalized, and exagerated)
(m*%.0f)',VE),'Fontsize',15)
xlabel('Distance from Axis (km)','Fontsize',15)
h = gca;
h.FontSize=15;
for jk = 1:length(h.XTickLabel)
   tryout{jk} = num2str(h.XTick(jk)*1e-3);
end
h.XTickLabel = tryout';
grid on
box on
l.Position = [0.2505 0.4847 0.5353 0.0150];

%% plot slope
subplot(2,1,2)
% offdist = 0;
%    [te(1).dist, te(1).slope] =
%     CCslopevsTe(60000, offdist,'-k',12,tes);
hold on
colorN = jet(length(Distances));
tk2=1;
for tk = 1:length(Distances)
   plot(  abs(Distances(tk)*1e-3),...
      -(OutwardRotations(tk)),...
      'o',    'MarkerFacecolor',colorN(tk,:),... colors(fid,:),...
            'MarkerEdgecolor',[1 1 1],...
            'MarkerSize',8)
   try
   if abs(round(Distances(tk))) == abs(round(ProPick.fittingdist(tk2)*1e3))
      plot(  abs(Distances(tk)*1e-3),...
         -(OutwardRotations(tk)),...
         'o',    'MarkerFacecolor',colorN(tk,:),... colors(fid,:),...
         'MarkerEdgecolor',[0 0 0 ],...
         'MarkerSize',8)
      tk2=tk2+1;
```

```
   end
   end
end

% plot weighted model slope
plot(
TES(HEidxW,ANidxW,IFidxW,CTidxW).dist(:,teidxW)+offdistX(offdistidxW),...
      TES(HEidxW,ANidxW,IFidxW,CTidxW).slope(:,teidxW),...
      '-','Color',[.5 .5 .5 .5],'Linewidth',5)

if doAll
% plot slope model slope
plot(   TES(HEidx,ANidx,IFidx,CTidx).dist(:,teidx)+offdistX(offdistidx),...
      TES(HEidx,ANidx,IFidx,CTidx).slope(:,teidx),...
      '-','Color',[1 0 1 .7],'Linewidth',1.5)

% plot topo model slope
% get a distance vector
TopoBFdist =
linspace(0,TES(HEidxT,ANidxT,IFidxT,CTidxT).dist(end,teidxT),length(TES(HEid
xT,ANidxT,IFidxT,CTidxT).topo)) + offdistX(offdistidxT);
plot(
TES(HEidxT,ANidxT,IFidxT,CTidxT).topodist(:,teidxT)+offdistX(offdistidxT),...To
poBFdist(stID+1:end),...
      TES(HEidxT,ANidxT,IFidxT,CTidxT).slope(:,teidxT),...
      '-','Color',[0 1 0 .7],'Linewidth',1.5)

% % text(11,-10,sprintf('Slope Fit (magenta):\nHeave = %g km \nAngle = %g^o
\nInfill = %g km\nCrust = %g km\nOffset = %g km\nTe = %g m',...
% %    heaves(HEidx)*1e-3,...
% %    angles(ANidx),...
% %    infill(IFidx)*1e-3,...
% %    crusts(CTidx)*1e-3,...
% %    offdistX(offdistidx),...
% %    tes(teidx)),...
% %    'Fontsize',12 ...
% %    )
% % text(11,-30,sprintf('Topo Fit (green):\nHeave = %g km \nAngle = %g^o \nInfill
= %g km\nCrust = %g km\nOffset = %g km\nTe = %g m',...
% %    heaves(HEidxT)*1e-3,...
% %    angles(ANidxT),...
% %    infill(IFidxT)*1e-3,...
% %    crusts(CTidxT)*1e-3,...
% %    offdistX(offdistidxT),...
% %    tes(teidxT)),...
% %    'Fontsize',12 ...
```

```
% %     )
end


% % text(15,-20,sprintf('Weighted Slope Fit (grey):\nHeave = %g km \nAngle =
%g^o \nInfill = %g km\nCrust = %g km\nOffset = %g km\nTe = %g m',...
% %     heaves(HEidxW)*1e-3,...
% %     angles(ANidxW),...
% %     infill(IFidxW)*1e-3,...
% %     crusts(CTidxW)*1e-3,...
% %     offdistX(offdistidxW),...
% %     tes(teidxW)),...
% %     'Fontsize',15, ...
% %     'Fontweight','bold'...
% %     )

% % text(1,-30,sprintf('Feature: %s\nMeasured:\nHeave = %.1f km\nWeight = %.2f
',...
% %     fet,...
% %     mHeave,...
% %     meanTw),...ProPick.Bfcc.Heave*1e-3),...
% %     'Fontsize',12 ...
% %     )
xlim([0 20])
ylim([-41 10])
xlabel('Distance from Axis (km)','Fontsize',15)
ylabel('Slope (Degrees)','Fontsize',15)

box on
h = gca;
h.FontSize=15;
grid on
if printme
cd mfiles
save2pdf(sprintf('%s_Feature_%g_Bestfit_Slopes.pdf',figstring,fetnum),fig)
cd ..
end
```

6.19 plotErrorSurface.m

```
function [numDF ys modelfitsbysigma] =
plotErrorSurface(R_ChiSqX,minfitidx,tes,heaves,angles,infill,crusts,offdistX,HEidx,
ANidx,IFidx,CTidx,offdistidx,teidx,fet,figstring)
% first make tes in Km
tes=tes*1e-3;
%%
R_ChiSqX = squeeze(R_ChiSqX);
```

```
% next, make a structure for looping
numDF =0;
numHe = length(heaves);
if numHe > 1
   numDF = numDF+1;
   ys(numDF).ylab = 'Heave (km)';
   ys(numDF).ys = heaves;
   ys(numDF).miny = heaves(HEidx);
   ys(numDF).minidx = HEidx;
end
numAn = length(angles);
if numAn > 1
   numDF = numDF+1;
   ys(numDF).ylab = 'Angle (degrees)';
   ys(numDF).ys = angles;
   ys(numDF).miny = angles(ANidx);
   ys(numDF).minidx = ANidx;
end
numIF = length(infill);
if numIF > 1
   numDF = numDF+1;
   ys(numDF).ylab = 'Infill (km)';
   ys(numDF).ys = infill;
   ys(numDF).miny = infill(IFidx);
   ys(numDF).minidx = IFidx;
end
numCr = length(crusts);
if numCr > 1
   numDF = numDF+1;
   ys(numDF).ylab = 'Crust (km)';
   ys(numDF).ys = crusts;
   ys(numDF).miny = crusts(CTidx);
   ys(numDF).minidx = CTidx;
end
numOS = length(offdistX);
if numOS > 1
   numDF = numDF+1;
   ys(numDF).ylab =  'Offset (km)';
   ys(numDF).ys = offdistX;
   ys(numDF).miny = offdistX(offdistidx);
   ys(numDF).minidx = offdistidx;
end
numTe = length(tes);
if numTe > 1
   numDF = numDF+1;
```

```
   ys(numDF).ylab = 'Te (km)';
   ys(numDF).ys = tes;
   ys(numDF).miny = tes(teidx);
   ys(numDF).minidx = teidx;
end

%% Sigma stuff
% do sigma vectors
sigmaMat = [
   1.00   4.00   9.00
   2.30   6.18   11.83
   3.53   8.02   14.16
   4.72   9.72   16.25
   5.89   11.31   18.21
   7.04   12.85   20.06
   8.18   14.34   21.85 ];
foursig=[16.00
19.33
22.06
24.50
26.77
28.91
30.96];
sigmas = [  sigmaMat(numDF,:) ];
sigmalabels = {'\sigma_1' '\sigma_2' '\sigma_3'};
% http://www.reid.ai/2012/09/chi-squared-distribution-table-with.html
%Sigma     1?     1.28          1.64    1.96    2?     2.58       3?     3.29       4?
%CI %      68.3%      80%     90%    95%    95.45%          99%    99.73%
          99.9%  99.99%
%P-value       0.317  0.20    0.10    0.05    0.0455 0.01    0.0027 0.001   0.00006
%chi2(k=1)    1.00    1.64    2.71    3.84    4.00    6.63    9.00    10.83   16.00
%chi2(k=2)    2.30    3.22    4.61    5.99    6.18    9.21    11.83   13.82   19.33
%chi2(k=3)    3.53    4.64    6.25    7.81    8.02    11.34   14.16   16.27   22.06
%chi2(k=4)    4.72    5.99    7.78    9.49    9.72    13.28   16.25   18.47   24.50
%chi2(k=5)    5.89    7.29    9.24    11.07   11.31   15.09   18.21   20.52   26.77
%chi2(k=6)    7.04    8.56    10.64   12.59   12.85   16.81   20.06   22.46   28.91
%chi2(k=7)    8.18    9.80    12.02   14.07   14.34   18.48   21.85   24.32   30.96
%chi2(k=8)    9.30    11.03   13.36   15.51   15.79   20.09   23.57   26.12   32.93
%chi2(k=9)    10.42   12.24   14.68   16.92   17.21   21.67   25.26   27.88   34.85
%chi2(k=10)   11.54   13.44   15.99   18.31   18.61   23.21   26.90   29.59   36.72
disp(sprintf('%g models or %.3f%% are within 1
sigma',numel(find(R_ChiSqX<sigmas(1))),100*numel(find(R_ChiSqX<sigmas(1)))/
numel(R_ChiSqX)))
disp(sprintf('%g models or %.3f%% are within 2
sigma',numel(find(R_ChiSqX<sigmas(2))),100*numel(find(R_ChiSqX<sigmas(2)))/
numel(R_ChiSqX)))
```

```
disp(sprintf('%g models or %.3f%% are within 3
sigma',numel(find(R_ChiSqX<sigmas(3))),100*numel(find(R_ChiSqX<sigmas(3)))/
numel(R_ChiSqX)))



%% now lets histogram the variables,



szRsq = size(R_ChiSqX);
for n = 1:length(sigmas)
   if strcmp('',fet) & strcmp('',figstring)
      m = 1;
   else
      fig=figure(11110*n);clf;
   end
   idxs = find(R_ChiSqX<sigmas(n));
   if numDF == 1
      [ id1 ] = ind2sub(szRsq , idxs);
      modelfitsbysigma(n).ids = id1;
   elseif numDF == 2
      [ id1, id2 ] = ind2sub(szRsq , idxs);
      modelfitsbysigma(n).ids = [id1  id2];
   elseif numDF == 3
      [ id1, id2, id3 ] = ind2sub(szRsq , idxs);
      modelfitsbysigma(n).ids = [id1  id2 id3 ];
   elseif numDF == 4
      [ id1, id2, id3 , id4] = ind2sub(szRsq , idxs);
      modelfitsbysigma(n).ids = [id1  id2 id3  id4];
   elseif numDF == 5
      [ id1, id2, id3, id4, id5 ] = ind2sub(szRsq , idxs);
      modelfitsbysigma(n).ids = [id1  id2 id3  id4  id5];
   elseif numDF == 6
      [ id1, id2, id3, id4, id5 , id6] = ind2sub(szRsq , idxs);
      modelfitsbysigma(n).ids = [id1  id2 id3  id4  id5 id6];
   elseif numDF == 7
      [ id1, id2, id3, id4, id5 , id6 ,id7] = ind2sub(szRsq , idxs);
      modelfitsbysigma(n).ids = [id1  id2 id3  id4  id5 id6 id7];
   end
   if strcmp('',fet) & strcmp('',figstring)
      m = 1;
   else
      hold on
      for m = 1:numDF
         sp = subplot(numDF,1,m);
```

```
        counts = hist(ys(m).ys(modelfitsbysigma(n).ids(:,m)),ys(m).ys);
        brs = bar(ys(m).ys,counts);
        brs.FaceColor = [ .05 .65 .05 ];
        for j = 1:length(counts)
          if counts(j) > 0
            t=text(ys(m).ys(j),counts(j),sprintf('%g',counts(j)),'Fontweight','bold');
            t.VerticalAlignment = 'bottom';
            t.HorizontalAlignment = 'center';
          end
        end
        xlabel(ys(m).ylab)
        ylabel('Counts')
        sp.XTick = ys(m).ys;
        sp.XTickLabel = ys(m).ys;
        if m == 1
          title(sprintf('Feature %s: %s Histogram',fet,sigmalabels{n}))
        end
      end
      % save it
      cd mfiles
      save2pdf(sprintf('%s_Feature_%s_Histogram_%g.pdf',figstring,fet,n),fig)
      cd ..
    end
end


%%
% first do we want a figure?
if strcmp('',fet) & strcmp('',figstring)
    m = 1;
else
    cmap = [ .88 .99 .66;[.88 .99 .66]*.75; [.88 .99 .66]*.5  ];
    for n=1:numDF
    spi=1;
    fig=figure(1111110*n);clf;
    % %[HEidx,ANidx,IFidx,CTidx,offdistidx,teidx]
    %  1  2  3  4  5  6
    %  he an if ct os te
    % heaves % angles % infill % crusts % offdistX  % tes
    nys = ys(n).ys ; miny = ys(n).miny ; ylab = ys(n).ylab;
    for m = 1:(numDF)
      % check that we are not going to plot the same variable against itself
      if m~=n
      clear C h cl cb
      idxs = setdiff(1:numDF,[n m]);
      pR_ChiSqX=permute(R_ChiSqX,[m,n,idxs]);
```

```
    if numDF == 7
    thispR =
pR_ChiSqX(:,:,minfitidx(idxs(1)),minfitidx(idxs(2)),minfitidx(idxs(3)),minfitidx(idxs
(4)),minfitidx(idxs(5)));
    elseif numDF == 6
    thispR =
pR_ChiSqX(:,:,minfitidx(idxs(1)),minfitidx(idxs(2)),minfitidx(idxs(3)),minfitidx(idxs
(4)));
    elseif numDF == 5
    thispR =
pR_ChiSqX(:,:,minfitidx(idxs(1)),minfitidx(idxs(2)),minfitidx(idxs(3)));
    elseif numDF == 4
    thispR = pR_ChiSqX(:,:,minfitidx(idxs(1)),minfitidx(idxs(2)));
    elseif numDF == 3
    thispR = pR_ChiSqX(:,:,minfitidx(idxs(1)));
    end

    subplot(numDF-1,1,spi)
    [C h] = contourf(nys,ys(m).ys, thispR ,...
       [ sigmas] );
    cl = clabel(C);
    for cln = 1:2:length(cl)
       cl(cln).Marker = '.';
    end
    for cln = 2:2:length(cl)
       for sigstr = 1:length(sigmas)
          if strfind(num2str(sigmas(sigstr)),cl(cln).String)
             cl(cln).String = sigmalabels{sigstr};
          end
          cl(cln).HorizontalAlignment ='center';
          cl(cln).FontSize = 18;
       end
    end
    hold on
    plot(miny,ys(m).miny,'+r')
    colormap(cmap);
    view(0, 90)
    if m == 1
       title(sprintf('Feature %s  %s vs %s',fet,ylab,ys(m).ylab))
    else
       title(sprintf(' %s vs %s',ylab,ys(m).ylab))
    end
    ylabel(ys(m).ylab)
    xlabel(ylab)
    spi=spi+1;
    end
```

```
    end

    cd mfiles
    save2pdf(sprintf('%s_Feature_%s_Slope_ErrorSurface_%g.pdf',figstring,fet,n),fig)
    cd ..

    end
end
```

6.20 power2.m

```
function b=power2(x);
% function b=power2(x)
% finds next number power of 2

n=0;
b=0;
while b<x;
n=n+1;
b=2^n;
end
b=2^n;
```

6.21 scattInt.m

```
function [ fctSI, mapbu, newmap, x_map , y_map, long, lat] =
scattInt(map,lat,long,long1,long2,lat1,lat2)

[nul lnsidx]=min(abs(long+long1));
[nul lnbidx]=min(abs(long+long2));
long = long(lnsidx:lnbidx);
[nul ltsidx]=min(abs(lat-lat1));
[nul ltbidx]=min(abs(lat-lat2));
lat = lat(ltsidx:ltbidx);
map = map(ltsidx:ltbidx,lnsidx:lnbidx);
% create a backup
mapbu = map;
[x_map,y_map]=meshgrid(long,lat);

% do some interpolation, so we can filter
[yyy xxx] = find(isfinite(map)==1);
[yyi xxi] = find(isnan(map));
xxx = long(xxx);
yyy = lat(yyy);
xxi = long(xxi);
yyi = lat(yyi);
zzz = map(find(isfinite(map)==1));
%F = griddedInterpolant(x_bathy',y_bathy',map','spline');
```

```
% this interpolation affects the original data IN NO WAY
fctSI = scatteredInterpolant(xxx',yyy',zzz,'natural','nearest');
newmap = map;
newmap(find(isnan(map))) =
fctSI(x_map(find(isnan(map))),y_map(find(isnan(map))));
```

6.22 slopecalc.m

```
function [fyt, ndx, nnx, yt] = slopecalc(heave,te,dx,crustthick,angle,profilelength,ift)

%%%%%%%%% 1. Input parameters for fault geometry
%%% heave/fh : horizontal extension of the fault 6e4 m from Schouten
%%% Te : Effective elastic thickness in m
% dx : minimum spacing, m default 1
% crustthick : crustal thickness, m 1000 default
% angle % fault angle, degrees 60 default
% profilelength=100000;   % profile length, m
%%% fh and te are now input in CCslopevsTe, as are all parameters

%%%%%%%%% 2. call the program that calculates the fault gemoetry, dofault.m
if ift == 0
   [nx,yt,ym,ndx]=dofault(angle,heave,profilelength,dx,crustthick);
else
   [nx,yt,ym,ndx]=dofaultWinfill(angle,heave,profilelength,dx,crustthick,ift);
end

% yt is seafloor topo
% ym is mantle topo
% nx is x array ndx is spacing

%%% 3. To solve the flexure equation we need symmetric topography
% this is *probably* because we need to find the characteristic wavelengths
% using fft
nyt=[yt fliplr(yt)];
nym=[ym fliplr(ym)];
nnx=ndx*[1:length(nym)];

% plot up the initial topo if you want
   noplot=0;
   if(noplot)
   figure(1);clf

   subplot(2,1,1)
   plot(nnx,nyt,'b-',nnx,nym,'g-')
   hold on
   end
```

```
%%%%% 4. Flexure calcuation
w=flex(nyt,nym,ndx,te);    % call the flexure program

if(noplot)
%%%%% Plot the flexural response
plot(nnx,w,'k-')
end%%if(noplot)

% calculate the 'flexed' topographies
fym=nym+w';
fyt=nyt+w'; % subtract the flexural response from the originial topography.

% the result is the curved topography
%
%  our w, is the amount of depression, that the topography experiences
% so we take our initial topography, and add the change (w)
% We get this,
%                    .                  .
% fym ----------------~'`.              ,' `-------------------
%                        `--------------'
%
%
% of course the exact shape changes with the parameters
%
mfyt=max(fyt);
%disp('************************************************')
%disp(['breakaway top ',int2str(mfyt-fyt(end)),' meters'])

if(noplot)
   %%%%% Plot the flexed topography
   plot(nnx,fyt,'r-',nnx,fym,'m-');
   %%fill([nnx,fliplr(nnx)],[fyt,fliplr(fym)],'-g')
   plot([nnx(1),nnx(end)],[fyt(1),fyt(end)],'-k')
   plot([nnx(1),nnx(end)],[0,0],':k')

   %%%%%%%%% 5 . Calculate the tilt
   imax=find([fyt]==max([fyt]));imax=imax(1); % find summit of fault; choose first
index if the two peaks are found
   hfyt=fyt(1:end/2);
   imin=find([hfyt]==min([hfyt]));imin=imin(1); % find base of fault; choose first
index if the two peaks are found
   plot(nnx(imax),nyt(imax),'r.') % plot the point
   plot(nnx(imin),nyt(imin),'r.') % plot the point
   %%axis('equal')
   %%axis([0,lp,-6000,6000])
   grid
```

```
   %legend('Initial Seafloor Topography','Initial Mantle Topography','Flexural
Response','Flexed Seafloor Topography','Flexed Mantle Topography')

   tilt=rad2deg(atan(diff(fyt(imax:imax+1))/ndx)); % calculate the tilt which will be
maximum at this point

   %%%%%% alternatively
   %tilt2=abs(rad2deg(atan(diff(fyt)/ndx)));
   if(0)
   tilt2=abs(rad2deg(atan(diff(real(w))/ndx)));
   else
   tilt2=(rad2deg(atan(diff(real(w))/ndx)));
   end
   %ifau=find(tilt2>an*.3);tilt2(ifau)=tilt2(ifau)*0;
   tilt3=[tilt2(1) tilt2'];
end
if(noplot)
subplot(2,1,2)
if(1)
plot(nnx,fyt,'r-',nnx,fym,'m-');
axis([0,lp,-6000,6000])
grid
elseif(0) % mlarson change, aug 9, don't plot any of this.
plot(nnx,tilt3,'r.');
axis([0,lp,-70,70])
end
end%%if(noplot)
```

6.23 TESmaker.m

```
function TES = TESmaker(heaves,angles,infill,crusts,depths,tes)

% initialize big mat
% % % BigMat.Slope = zeros(29999,...
% % %                     (length(angles)* ...
% % %                         length(infill)* ...
% % %                         length(crusts)* ...
% % %                         length(heaves)* ...
% % %                         length(tes)* ...
% % %                         length(offdistX)) ...
% % %                         );
% % % BigMat.Dist = BigMat.Slope;
% % % BigMat.Topo = zeros(34540,...
% % %                     (length(angles)* ...
% % %                         length(infill)* ...
% % %                         length(crusts)* ...
```

```
% % %                       length(heaves)* ...
% % %                       length(tes)* ...
% % %                       length(offdistX)) ...
% % %                       );
% % initialize models structure
% TES(1:length(heaves),1:length(angles),1:length(infill),1:length(crusts)).topo =0;
% TES(1:length(heaves),1:length(angles),1:length(infill),1:length(crusts)).slope=0;


tic
BMcount = 1;
   for he = 1:length(heaves)
      he
      %[te(he).dist, te(he).slope] = CCslopevsTe(heaves(he), 0,'none');
      for ANid = 1:length(angles)
         ANid
         for IFid=1:length(infill)
            for CTid=1:length(crusts)
               [  TES(he,ANid,IFid,CTid).slope TES(he,ANid,IFid,CTid).topo
TES(he,ANid,IFid,CTid).dist TES(he,ANid,IFid,CTid).topodist] = ...

CCslopesLooper(heaves(he),angles(ANid),crusts(CTid),infill(IFid),depths,tes);
               TES(he,ANid,IFid,CTid).crust = crusts(CTid);
               TES(he,ANid,IFid,CTid).heave = heaves(he);
               TES(he,ANid,IFid,CTid).angle = angles(ANid);
               TES(he,ANid,IFid,CTid).infill = infill(IFid);
% % %                   BigMat.Slope(:, BMcount:length(tes)+BMcount-1) =
TES(he,ANid,IFid,CTid).slope;
% % %                   BigMat.Topo(:, BMcount:length(tes)+BMcount-1) =
TES(he,ANid,IFid,CTid).topo;
% % %                   BigMat.Dist(:, BMcount:length(tes)+BMcount-1) =
TES(he,ANid,IFid,CTid).dist;
% % %                   BMcount = length(tes)+BMcount;
            end
         end
      end
   end
toc
```

6.24 whichaxis.m

```
function axis_to_use = whichaxis(axis,xmidpoint,ymidpoint,long_or_lat)
% first, find which orientation axis is
%if abs(axis(1,1)-axis(end,1)) < abs(axis(1,2)-axis(end,2))
if sum(abs(diff(axis(isfinite(axis(:,1)),1)))) < sum(abs(diff(axis(isfinite(axis(:,2)),2))))
   MA = 1;
   MM = 2;
```

```
else
   MA = 2;
   MM = 1;
end

[nul mid2ix] = min(abs(long_or_lat - ymidpoint));

% do axes stuff
axsc = 1;
ax1 = NaN*zeros(size(axis,1),size(axis,2)/2)';
ax2 = ax1;
for jk = 1:2:size(axis,2)
   ax1(axsc,find(isfinite(axis(:,jk-1+MA)))) = axis(find(isfinite(axis(:,jk-1+MA))),jk-
1+MA);
   ax2(axsc,find(isfinite(axis(:,jk-1+MM)))) = axis(find(isfinite(axis(:,jk-1+MM))),jk-
1+MM);
   ax1midvals(axsc) = ax1(axsc,mid2ix);
   axsc = axsc+1;
end

[nul ax1idx ] = min(abs(xmidpoint - ax1midvals));
axis_to_use(:,1) = ax1(ax1idx,:)';
axis_to_use(:,2) = ax2(ax1idx,:)';
```

7 Bibliography

Behn, M.D., Lin, J., & Zuber, M.T., 2004, Effects of Hydrothermal Cooling and Magma Injection on Mid-Ocean Ridge Temperature Structure, Deformation, and Axial Morphology, in Mid-Ocean Ridges, *in* C.R. German, J. Lin, & L.M. Parson, eds, American Geophysical Union, Washington D.C.. doi: 10.1029/148GM06.

Buck W. R., Martinez F., Steckler M. S., Cochran J. R., 1988, Thermal consequences of lithospheric extension: pure and simple, *Tectonics*, v. 7, n. 2, p. 213 – 234.

Buck, W.R., 1988, Flexural Rotation of Normal Faults, *Tectonics*, v. 7, n. 5, p. 959-973.

Buck, W. R., Lavier, L. L., Poliakov, A. N. B., 2005, Modes of faulting at mid-ocean Ridges, *Nature*: . v. 434, n. 7034, p. 719-723, doi 10.1038/Nature03358.

Byerlee, J.D., 1978, Friction of Rocks, *Pure and Applied Geophysics,* v. 116, p. 615-626.

Cannat, M., Lagabrielle, Y., Bougault, H., Casey, J., deCoutures, N., Dmitriev, L., & Fouquet, Y., 1997, Ultramafic and gabbroic exposures at the Mid-Atlantic Ridge: Geological mapping in the 15°N region, *Tectonophysics*, v. 279, p. 193-213.

Cannat, M., Sauter, D., Mendel, V., Ruellan, E., Okino, K., Escartin, J., Combier, V., & Baala, M., 2006, Modes of seafloor generation at a melt-poor ultraslow-spreading ridge, *Geology*, v. 34, p. 605–608, doi: 10.1130/G22486.1.

Carbotte, S.M., & MacDonald, K., 1992, East Pacific Rise 8^{o}-10^{o}30'N: Evolution of Ridge Segments and Discontinuities from SeaMARC II and Three-Dimensional Magnetic Studies, *Journal of Geophysical Research*, v. 97, n. B5, p. 6959-6982.

Carbotte, S., Small, C., & Donnelly, K., 2004, The influence of ridge migration on the magmatic segmentation of mid-ocean ridges, *Nature*, v. 429, p. 743–746.

Carbotte, S.M., Smith, D.K., Cannat, M., Klein, E.M., 2015, Tectonic and magmatic segmentation of the Global Ocean Ridge System: a synthesis of observations: *in* Wright, T. J., Ayele, A., Ferguson, D. J., Kidane, T. & Vye-Brown, C. eds, Magmatic Rifting and Active Volcanism. *Geological Society, London, Special Publications,* 420, http://dx.doi.org/10.1144/SP420.5.

Christensen, I., Nikolas, 1966, Elasticity of Ultrabasic Rocks, *Journal of Geophysical Research*, v. 71, n. 24, p. 5921-5931.

Collettini, C., Niemeijer, A., Viti, C., & Marone, C., 2009, Fault zone fabric and fault weakness, *Nature*, v. 462, p. 907-911.

Coney, P. J., 1980, Cordilleran metamorphic core complexes: An overview, *Geological Society of American Mem*. v. 153, p. 7-31.

Conrad, C.P. & Lithgow-Bertelloni, C., 2002, How mantle slabs drive plate tectonics, *Science*, v. 298, p. 207-209.

Daly, R. A., 1912, Reconnaissance of the Shuswap Lakes and vicinity: south-central British Columbia, *Geological Survey of Canada Annual Rep*, p. 12.

deMartin, B.J., R.A. Sohn, J.P. Canales, & S. Humphris, 2007, Kinematics and geometry of active detachment faulting beneath the Trans-Atlantic

Geotraverse (TAG) hydrothermal field on the Mid-Atlantic Ridge, *Geology*, v. 35, p. 711–714.

DeMets, C., Gordon, R. G., & Argus, D. F., 2010, Geologically current plate motions, *Geophysical Journal International*, v. 181, p. 1-80.

Dick, H., Lin, J., & Schouten, H., 2003, An ultraslow spreading class of ocean ridge, *Nature*, v. 426, p. 405– 412.

Dick, H. J. B., Thompson, G. & Bryan, W. B., 1981, Low angle faulting and steady-state emplacement of plutonic rocks at ridge-transform intersections: *Eos Transactions of the AGU*, v. 62, n. 406.

Dosso, L., Hanan, B.B., Bougault, H., Schilling, J-G, & Joron, J-L, 1991, Sr-Nd-Pb geochemical morphology between 10° and 17° N on the Mid-Atlantic Ridge: a new MORB isotope signature, *Earth and Planetary Science Letters,* v. 106, p. 29-43.

Escartín, J., Hirth, G., & Evans, B., 1997, Effects of serpentinization on the lithospheric strength and the style of normal faulting at slow-spreading ridges, Earth and Planetary Science Letters, v. 151, p. 181-189.

Escartín, J., D. K. Smith, D.K., Cann, J., Schouten, H., Langmuir, C.H., & Escrig, S., 2008, Central role of detachment faults in accretion of slow spreading oceanic lithosphere: *Nature*, v. 455, p. 790-794, doi:10.1038/nature07333.

Fontaine F., Cannat, M., & Escartín, J., 2008, Hydrothermal circulation at slow-spreading mid-ocean ridges: The role of along-axis variations in axial lithospheric thickness, *Geology*, no. 10; p. 759–762; doi: 10.1130/G24885A.1;

Fujiwara, T., Lin, J., Matsumoto, T., Kelemen, P. B. Tucholke, B. E., & Casey, J. F., 2003, Crustal evolution of the Mid-Atlantic Ridge near the Fifteen-Twenty Fracture Zone in the last 5 Ma, *Geochemistry Geophysics Geosystems*, v. 4, doi: 10.1029/2002GC000364.

Gac, S., Tisseau, C., Dyment, J., & Goslin, J., 2006, Modelling the thermal evolution of slow-spreading ridge segments and their off-axis geophysical signature, *Geophysical Journal International*, v. 164, p. 341-358, doi: 10.1111/j.1365-246X.2005.02844.x.

Gregg, P.M., Hebert, L.B., Montési, L.G.J., & R.F. Katz, 2012, Geodynamic models of melt generation and extraction at mid-ocean ridges, *Oceanography*, v. 25, n. 1, p. 78–88, http://dx.doi.org/10.5670/oceanog.2012.05.

Gudmundsson, Agust, 2011. *Rock Fractures in Geological Processes*. London, UK: Cambridge University Press.

Hayman, N. W., Grindlay, N. R., Perfit, M. R., Mann, P., Leroy, S. & de Lépinay, B. M., 2011, Oceanic core complex development at the ultraslow spreading Mid-Cayman Spreading Center, *Geochemistry, Geophysics, Geosystems*, v. 12, http://dx.doi.org/10.1029/ 2010GC003240.

Jiang, X., Wan, L., Wang, X., Wiang, S., & Hu, B., 2009, Estimation of fracture normal stiffness using a transmissivity-depth correlation, *International Journal of Rock Mechanics & Mining Sciences,* v. 46, p. 51– 58.

Lister, G.S., & Davis, G.A., 1989, The origin of metamorphic core complexes and detachment faults formed during Tertiary continental extension in the northern Colorado River region, U.S.A., *Journal of Structural Geology*, v. 11, n. 1/2, p. 65-94.

Liu, C.Z., Snow, J.E., Hellebrand, E., Brügmann, G., von der Handt, A., Buechl, A. & Hofmann, A.W., 2008, Ancient, highly heterogeneous mantle beneath Gakkel Ridge, *Nature*, v. 452, p. 311-316.

Macdonald, K.C., 1982, Mid-ocean ridges: Fine scale tectonic, volcanic and hydrothermal processes within the plate boundary zone, *Annual Reviews of Earth and Planetary Science*, v. 10, p. 155-190.

Macdonald, K.C., Fox, P.J., Perram, L.J., Eisen, M.F., Haymon, R.M., Miller, S.P., Carbotte, S.M., Cormier, M.-H., & Short, A.N., 1988, A new view of the mid-ocean ridge from the behavior of ridge axis discontinuities, *Nature*, v. 335, p. 217-225.

MacLeod, C.J., Searle, R.C., Murton, B.J., Casey, J.F., Mallows, C., Unsworth, S.C., Achenbach, K.L., & Harris, M., 2009, Life cycle of oceanic core complexes, *Earth and Planetary Science Letters*, v. 287, p. 333–344, doi: 10.1016/j.epsl.2009.08.016.

Maffione, M., Morris, A., & Anderson, M., 2013, Recognizing detachment-mode seafloor spreading in the deep geological past, *Nature, Scientific Reports*, v. 3, p. 2336, doi: 10.1038/srep02336.

Mallows, C., & Searle, R. C., 2012, A geophysical study of oceanic core complexes and surrounding terrain, Mid-Atlantic Ridge at 13°-14°N, *Geochemistry, Geophysics, Geosystems*, v. 13, Q0AG08, doi:10.1029/2012GC004075.

Marrow, C.A., Moore, D.E., & Lockner, D.A., 2000, The effect of mineral bond strength and adsored water on fault gouge frictional strength, *Geophysical Research Letters*, v. 27, n. 6, p. 815-818.

McKenzie, D.P., 1969, Speculations on the consequences and causes of plate motions, *Geophysical Journal of the Royal Astronomical Society*, v. 18, p. 1-32.

Michael, P.J., Langmuir, C. H., Dick, H. J. B., Snow, J. E., Goldsteink, S. L., Graham, D. W. , Lehnertk, K., Kurras, G., Jokatq, W., Muhe, R. &, Edmonds, H. N., 2003, Magmatic and amagmatic seafloor generation at the ultraslow-spreading Gakkel ridge, Arctic Ocean, *Nature*. v. 423. p. 956-961.

Misch, P., 1960, Regional structural reconnaissance in central-northeast Nevada and some adjacent areas: observations and interpretations: *Intermountain Association of Petroleum Geology, 11th Ann. Field Conference, Guidebook*, p. 17-42.

Miranda, E. A., & Dilek, Y., 2010, Oceanic Core Complex Development in Modern and Ancient Oceanic Lithosphere: Gabbro-Localized versus Peridotite-Localized Detachment Models, *Journal of Geology*, v. 118, n. 1.

Montési, L.G.J., Behn, M.D., Hebert, L.B., Lin, J., & Barry, J.L., 2011, Controls on melt migration and extraction at the ultraslow Southwest Indian Ridge 10–16 E, *Journal of Geophysical Research, Solid Earth*, v. 116, B10102, doi:10.1029/2011JB008259.

Morris, G., Pressling, J., Macleod, C.J., Grimes, B., & Searle, R., 2009, Footwall rotation in an oceanic core complex quantified using reoriented Integrated Ocean Drilling Program core samples, *Earth and Planetary Science Letters*, v. 287, p. 217–228.

Mueller, R.D., Roest, W.R., Royer, J.-Y., Gahagan, L.M., & Sclater, J.G., 1997, Digital isochrons of the world's ocean floor, *Journal of Geophysical Research*, v. 102(B2), p. 3211–3214.

Müller, R.D., Sdrolias, M., Gaina, C., & Roest, W.R., 2008, Age, spreading rates and spreading symmetry of the world's ocean crust, *Geochemistry, Geophysics, Geosystems*, v. 9, p. Q04006, doi:10.1029/2007GC001743.

Müller, R. D., Roest, W.R., Royer, J.-Y., 1998, Asymmetric seafloor spreading expresses ridge-plume interactions, *Nature*, 396, 455 – 459.

Ohara, Y., yoshida, T., Kato, Y., & Kasuga, S., 2001, Giant Megamullion in the Parece Vela Backarc Basin, *Marine Geophysical Research*, n. 22, p. 47-61.

Ondréas, H., Cannat, M., Fouquet, Y., & Normand, A., 2012, Geological context and vents morphology of the ultramafic-hosted Ashadze hydrothermal areas (Mid-Atlantic Ridge 13°N): *Geochemistry, Geophysics, Geosystems*, v. 13, Q0AG14, doi:10.1029/2012GC004433.

Perram, L., Cormier, M., & MacDonald, K., 1993, Magnetic and Tectonic Studies of the Dueling Propagating Spreading Centers at 20 40'S on the East Pacific Rise: Evidence for Crustal Rotations, *Journal of Geophysical Research*, v. 98, n. B8, p. 13,835-13,850.

Petersen, S., Kuhn, K., Kuhn, T., Augustin, N., Hékinian, R., Franz, L., Borowski, C., 2009, The geological setting of the ultramafic-hosted Logatchev hydrothermal field (14°45′ N, Mid-Atlantic Ridge) and its influence on massive sulfide formation, *Lithos*, v. 112, is. 1–2, p. 40-56, http://dx.doi.org/10.1016/j.lithos.2009.02.008.

Phipps Morgan, J. & Forsyth, D. W., 1988, Three-dimensional flow and temperature perturbations due to a transform offset: Effects on oceanic crust and upper mantle structure, *Journal of Geophysical Research*, 93, 2955–2966, doi: 10.1029/JB093iB04p02955.

Picazo, S., Cannat, M., Delacour, A., Escartin, J., Roumejon, S., & Silyantev, S., 2012, Deformation associated with the denudation of mantle-derived rocks at the Mid-Atlantic Ridge 13^{o}-15^{o} N: The role of magmatic injections and hydrothermal alteration: *Geochemistry, Geophysics, Geosystems*, v. 13, , Q04G09, doi:10.1029/2012GC004121.

Reston, T.J., & Ranero, C., 2011, The 3 D geometry of detachment faulting at mid ocean ridges: *Geochemistry, Geophysics, Geosystems*, v. 12, , Q0AG05, doi:10.1029/2011GC003666.

Sauter, D., Cannat, M. et al., 2013, Continuous exhumation of mantle-derived rocks at the Southwest Indian Ridge for 11 million years: *Nature Geoscience*, v. 6, p. 314–320, http://dx.doi.org/10.1038/ngeo1771.

Schouten, H., Smith, D. K., Cann, J. R., & Escartin, J., 2010, Tectonic versus magmatic extension in the presence of core complexes at slow-spreading ridges from a visualization of faulted seafloor topography: *Geology*, v. 38, n. 7, p. 615-618, doi: 10.1130G30803

Schubert, Gerald, Donald L. Turcotte, and Peter Olson, 2001, Mantle Convection in the Earth and Planets. 1st ed. *Cambridge*: Cambridge University Press.

Schroeder, T., Cheadle, M. J., Dick, H. J. B., Faul, U., Casey, J. F., & Kelemen, P.B., 2007, Nonvolcanic seafloor spreading and corner-flow rotation

accommodated by extensional faulting at 15_N on the Mid-Atlantic Ridge: A structural synthesis of ODP Leg 209, *Geochemistry, Geophysics, Geosystems*, v. 8, p. Q06015, doi:10.1029/2006GC001567.

Shaw, P. J., 1992, Ridge segmentation, faulting and crustal thickness in the Atlantic Ocean: *Nature*, v.358, n. 6386, p. 490-493.

Shaw, W.T., & Lin, J., 1996, Models of ocean ridge lithospheric deformation: Dependence on crustal thickness, spreading rate, and segmentation, *Journal of Geophysical Research*, v. 101, p. 17,977-17,993.

Smith, D. K., Cann, J. R., & Escartin, J., 2006, Widespread active detachment faulting and core complex formation near 13º N on the Mid-Atlantic Ridge, *Nature*, v. 442, doi:10.1038/nature04950.

Smith, D. K., Escartín, J., Schouten, H., & Cann, J. R., 2008, Fault rotation and core complex formation: Significant processes in seafloor formation at slow-spreading mid-ocean ridges (Mid-Atlantic Ridge, 13°–15°N), Geochemistry, Geophysics, Geosystems, v. 9, p. Q03003, doi:10.1029/2007GC001699.

Spencer, J.E., 1984, Role of tectonic denudation in warping and uplift of low-angle normal faults: *Geology*, v. 12, p. 95-98.

Standish, J. J., H. J. B. Dick, P. J. Michael, W. G. Melson, and T. O'Hearn, 2008, MORB generation beneath the ultraslow spreading Southwest Indian Ridge (9–25°E): Major element chemistry and the importance of process versus source, *Geochemistry, Geophysics, Geosystems*, v. 9, p. Q05004, doi:10.1029/2008GC001959.

Tarduno, J., Bunge, H.-P., Sleep, N., & Hansen, U., 2009, The Bent Hawaiian-Emporer Hotspot Track: Inhereting the Mantle Wind: *Science*, v. 324, n. 5923, p. 50-53, doi:10.1126/science.1161256.

Tucholke, B., Behn, M.D., Buck, W.R., & Lin, J., 2008, Role of melt supply in oceanic detachment faulting and formation of megamullions: *Geology*, v. 36, n. 6, p. 455–458.

Tucholke, B.E., J. Lin, M.C. Kleinrock, M.A. Tivey, T.B. Reed, J. Goff and G. Jaroslow, 1997. Crustal structure and segmentation of the western Mid-Atlantic Ridge flank, 25°30'-27°10'N and 0-29 M.Y., *Journal of Geophysical Research,* v. 102, p. 10,203-10,223.

Turcotte, D. L., & E. R. Oxburgh, 1967, Finite amplitude convection cells and continental drift, J. Fluid Mech., v. 28, p. 29-42.

Turcotte, D.L., & Schubert, G., 2002, Geodynamics, *Cambridge University Press*, 2nd edition. ISBN 0 521 66624.

Wheeler, J. O., 1963, Rogers Pass map-area, British Columbia and Alberta: *Geological Survey of Canada Papers*, p. 62-32.

Wernicke, B., 1985, Uniform-sense normal simple shear of the continental lithosphere*: Canada Journal of Earth Science*, v. 22, p. 108-125.

Wernicke, B. & Axen, G.J., 1988, On the role of isostasy in the evolution of normal fault systems: *Geology*, v. 16, p. 848-851.

Wernicke, B., & Burchfiel, C., 1982, Modes of extensional tectonics, *Journal of Structural Geology*, v. 4, p. 105-115, doi:10.1016/0191-8141(82)90021-9.

Wessel, P., Smith, W. H. F., Scharroo, R., Luis, J.F., &Wobbe, F., 2013, Generic Mapping Tools: Improved version released, EOS Trans. AGU, v. 94, p. 409-410.

White, R. S., T. A. Minshull, M. J. Bickle, and C. J. Robinson, 2001, Melt generation at very slow-spreading oceanic ridges: Constraints from geochemical and geophysical data, *Journal of Petrology*, v. 42(6), p. 1171–1196.

Wilson, T., 1965, A New Class of Fault and their Bearing on Continental Drift: *Nature*, v. 207, n. 4995, p. 343-347, doi:10.1038/207343a0.

Zhou, H-y. & Dick, H. J. B., 2013, Thin crust as evidence for depleted mantle supporting the Marion Rise: *Nature*, v. 494, p. 195–200.