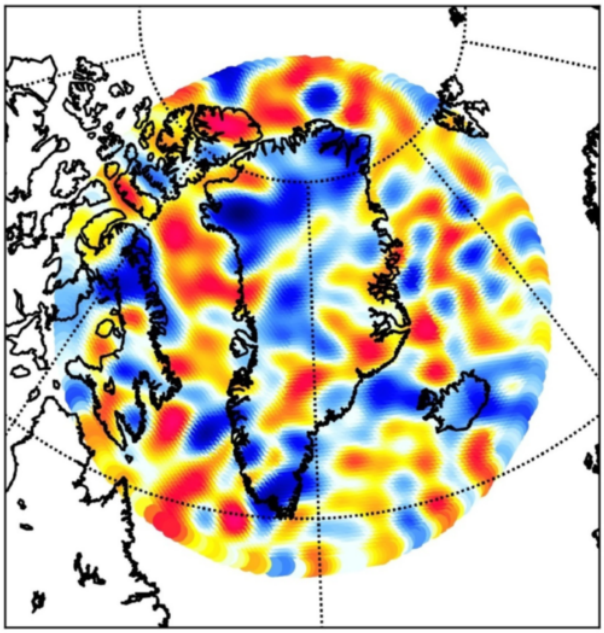
High Resolution Regional Modelling of the Lithospheric Magnetic Field By a Slepian Approach Using Satellite Gradient Data From CHAMP And Swarm

Master's Thesis



Author: Rasmus Ringsborg Joost (s134372)
Supervisor: Chris Finlay

February 2019

DTU Space
National Space Institute

Abstract

The upper parts of the Earth are magnetized, giving rise to a lithospheric magnetic field which is measurable by satellites. This field can be used to study the structure and dynamics of the Earth. Modelling the Earth's lithospheric magnetic field is a challenging task. Especially when utilizing satellite data, due to diminishing of the lithospheric magnetic signature at satellite altitudes. In this thesis, a Slepian approach to regional magnetic field modelling based on linear combinations of globally defined real spherical harmonics is used to produce regional models. This method eliminates the requirement of boundary conditions, simplifying the regional problem greatly. The primary aim of this thesis is to develop a Python toolbox for performing regional modelling of the Earth's lithospheric field with a Slepian approach. The toolbox has been successfully benchmark tested over the Bangui Anomaly and will be employed over four different regions with lithospheric magnetic features of geophysical interest; the Bangui Anomaly at low latitudes with a large amplitude anomaly, Australia which is a thoroughly investigated region at mid-latitude, the Walvis Ridge for an oceanic region and lastly Greenland for a high-latitude analysis. Using gradient data from the *Swarm* and CHAMP satellites, models are computed over the regions of interest using the developed Python toolbox. Results are compared to existing global models, to assess the performance of the Slepian approach to regional modelling. There is a good agreement between the regional models computed here and the existing global models. The high latitude region of Greenland shows improvement over the LCS-1 model which is a state-of-the-art global model derived using satellite data, and it also agrees well with the EMM2015 model which in addition to satellite data utilizes aeromagnetic data. Furthermore, the Slepian approach to regional modelling shows good agreement with the LCS-1 model over the Bangui Anomaly but with more structure, and larger amplitudes. It shows good agreement with EMM2015 over Australia, but edge effects were introduced in this case, suggesting that improvements can be made. The regional model of the Walvis Ridge shows less agreement with the LCS-1 model, but the larger scale features are compatible. Overall, the results obtained using the Slepian approach to regional modelling are very encouraging, particularly when studying high latitude regions that are often poorly represented in global models.

Preface

This master's thesis is submitted as the final project to fulfil the requirements for obtaining the M.Sc. in Earth and Space Physics and Engineering at the Technical University of Denmark. The thesis consists of 35 ECTS points of the master's programme. The work described was initialized August 1st 2018 and finalized February 5th 2019.

I am very grateful for the time my supervisor, Chris Finlay, spent discussing regularization alongside his ability to always have answers to questions or a book to borrow from his extensive library. Furthermore I want to thank M.Sc. students Eigil Yuichi Hyldgaard Lippert and Stefan Krebs for allowing me to use their implementation of the icosahedral grid used in this work. My warmest thanks to my fiancée, Selina, for thoroughly reading my thesis and providing excellent feedback.

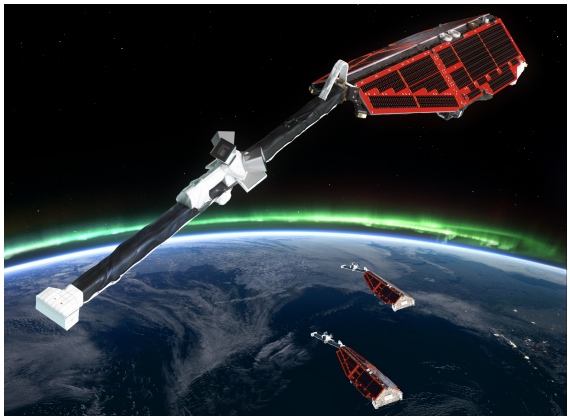
Contents

Chapter 1

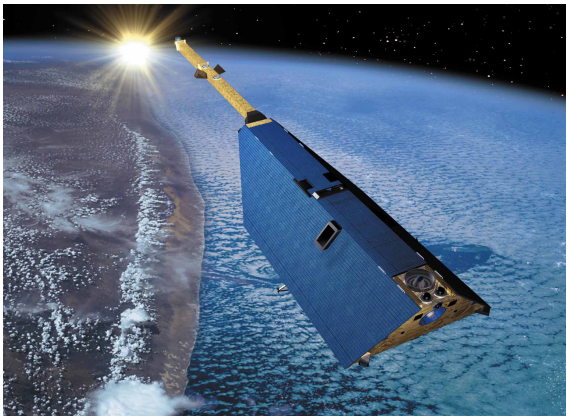
Introduction

Modelling the Earth's lithospheric (crustal) magnetic field using satellite data is a challenging enterprise due to diminishing of the lithospheric magnetic signature at satellite altitudes. Despite the challenge, producing high-resolution models is attractive due to the information contained on crustal geology, such as large-scale tectonic processes, and more regionally the crustal depth, through the Curie boundary.

Currently, only one satellite mission produces relevant data for lithospheric magnetic field modelling, namely the *Swarm* mission (Friis-Christensen et al., 2006) (Figure 1.1a). This on-going mission was launched in 2013. Prior to *Swarm* was the Challenging Mini-satellite Payload (CHAMP, see e.g. Maus (2007)) mission (Figure 1.1b). This mission was launched in 2000 and had its atmospheric re-entry in 2010. Due to its low altitudes towards the end of the mission, the data is of high interest for lithospheric magnetic field modelling.



(a) Artist's impression of the *Swarm* constellation. Source: http://www.esa.int/spaceinimages/Images/2012/02/Swarm_constellation.



(b) Artist's impression of the CHAMP satellite. Source: https://www.dlr.de/rd/en/desktopdefault.aspx/tabid-2440/3586_read-5330/.

Figure 1.1: Artist's impressions of the two satellite missions *Swarm* and CHAMP.

Global magnetic field models are a well-established practice, and the models can be divided into two classes. One class comprehensively models several magnetic sources all at once, e.g. the core

field, lithospheric field and the magnetospheric contributions, see Figure 1.2 for an illustration of sources contributing to the geomagnetic field. This category includes the CHAOS-6 model (Finlay et al., 2016). The second category involves removing unwanted contributions from the data using *a priori* models. The LCS-1 model utilizes CHAOS-6 as an *a priori* model to remove the unwanted contributions from the core field and magnetospheric contributions. Model parametrization is done through an equivalent point source approach (Olsen et al., 2017). With a combination of CHAMP and *Swarm* data this model provides greatly detailed maps of the lithospheric magnetic field. A highly regarded model is the lithospheric magnetic field model (MF7, Maus et al. (2007)), which exclusively utilizes CHAMP data. This model is parametrized by a spherical harmonic expansion of the magnetic potential approach. The EMM2015 (https://www.ngdc.noaa.gov/geomag/EMM/) lithospheric field model combines satellite, aeromagnetic, marine and ground observations. In regions such as Australia where extensive aeromagnetic surveys have been conducted this model is excellent. However in regions of more sparsely distributed data it seems to fall short (Olsen et al., 2017). This model is also parametrized by a spherical harmonic expansion of the magnetic potential.

The models discussed above are all global. Another practice within modelling the geomagnetic field is regional modelling. Producing regional lithospheric magnetic field models utilizing satellite data is challenging, but is practiced in various forms. Thébault et al. (2016) utilize *Swarm* data from the first two years of the mission and approaches the regional model through Revised Spherical Cap Harmonic Analysis (R-SCHA, (Thébault, 2006)). This approach in fact combines 600 regional spherical caps to produce a global model. Thus model parametrization depends on the spherical caps constructed as well as boundary conditions for each of these caps. The parameters can be transformed to spherical harmonic Gauss coefficients (internal source spherical harmonic coefficients), which allows for comparison with other models. A likely challenge with this approach is the applied boundary conditions to the spherical caps.

A new method for regional modelling of the Earth's magnetic field has been developed by Plattner and Simons (2017) and is henceforth referred to as the Slepian approach to regional modelling. It is a potential field approach that uses linear combinations of real, globally defined spherical harmonic functions optimized regionally. The linear combination of functions, alongside its ability to be upwards or downwards continued are called Altitude-Cognizant Gradient Vector Slepian Functions (AC-GVSF). Despite it being a locally optimized basis, the functions are defined globally, eliminating the requirement of boundary conditions, which indeed is an attractive feature. The model parametrization of this approach is based on an expansion of the magnetic potential, with a twist. The localized spherical harmonic coefficients (Slepian coefficients) are approximated by the linear combination of global spherical harmonic functions. Because the Slepian functions are based on real spherical harmonics, the spherical harmonic Gauss coefficients can be retrieved with the Slepian coefficients.

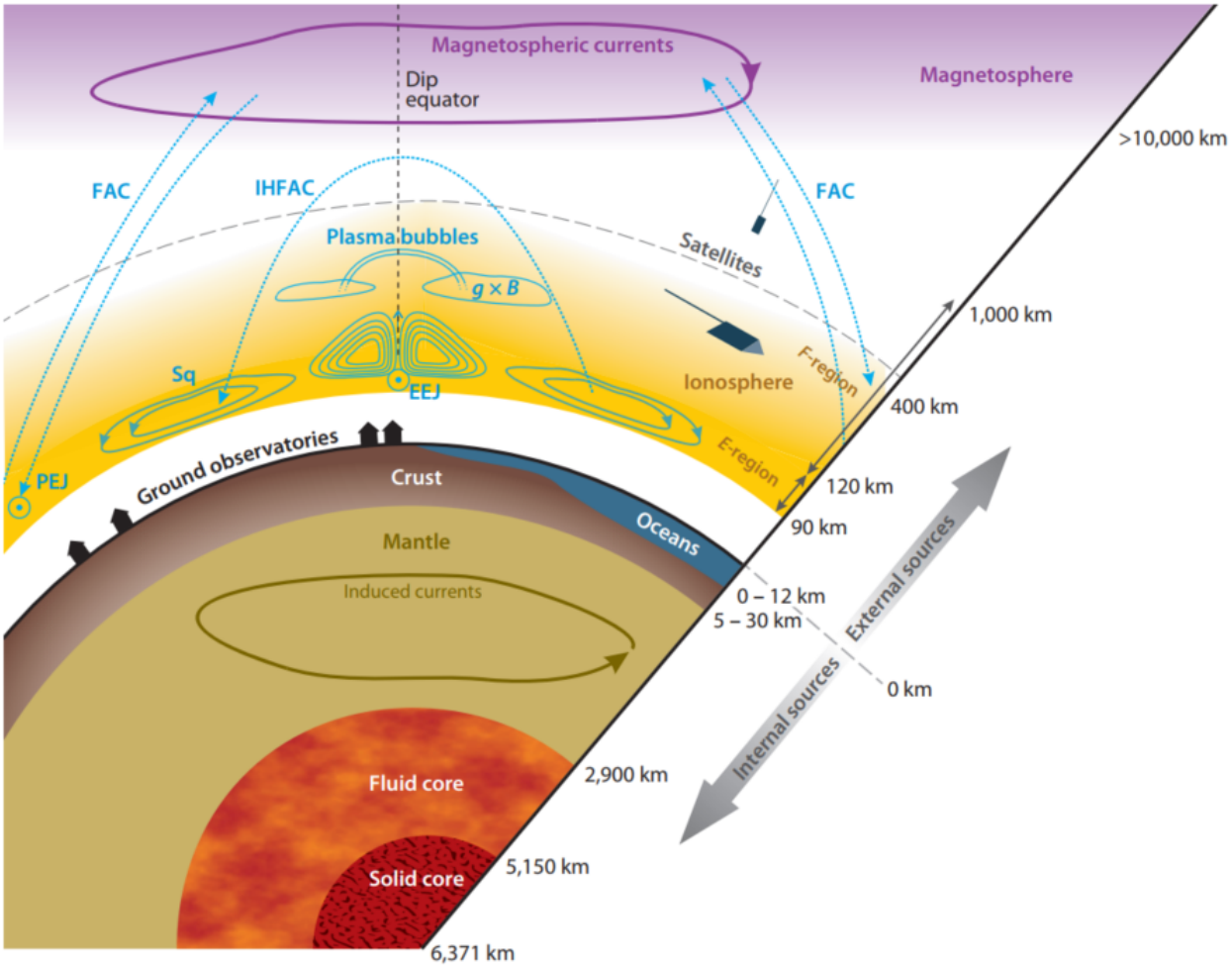


Figure 1.2: Illustration of the sources of the geomagnetic field. Source: DTU course 30740, lecture notes.

This thesis introduces a newly developed Python toolbox for producing high resolution regional models of the Earth's lithospheric magnetic field, with the Slepian approach utilizing satellite data. I initialized the development of the Python toolbox in the synthesis project leading up to this thesis, documented in Rasmus R. Joost (2018). Final development and incorporation of real satellite data has been performed in this master's thesis. The toolbox developed will henceforth be referred to as the Python toolbox. With Python being open source, the Python toolbox can be accessed without the need of a licensed software, e.g. MATLAB. The implementation includes spherical caps as the region of choice due to their attractive symmetric properties that will be discussed in this thesis. Furthermore the data set will be subject to *a priori* models removing unwanted contributions such as the core field and the magnetospheric field. Pushing to produce models of spherical harmonic degree $L = 200$, corresponding to a wavelength of 200 km, makes this one of the more ambitious projects in the field.

The data set containing satellite observations is introduced in Chapter 2, where the CHAMP and *Swarm* satellites are introduced together with data selection criteria and the regions of interest. The theory required to produce regional models through the Slepian approach is introduced in Chapter 3. Localized kernel matrices, AC-GVSF and design matrices are defined followed by an introduction

to the inverse problems to be solved. The Python toolbox developed will be introduced in Chapter 4 where key parts of the code is dissected to grant an overview of the implemented methods. A validation of the toolbox follows the introduction of the toolbox using synthetic data based on the LCS-1 model (Olsen et al., 2017), which is found in Chapter 5. Regional maps of the lithospheric magnetic field evaluated at the Earth's surface, alongside statistics of models that produce these maps, are introduced in Chapter 6 and discussed in detail in Chapter 7. The discussion will furthermore introduce strengths and weaknesses of the Slepian approach to regional modelling and suggestions to its usability. Finally a conclusion is found in Chapter 8 where the findings are reflected upon with respect to the aim of the thesis, stated in the following section.

1.1 Project Milestones

Presented in this section are the updated milestones presented in the project plan. The original project plan is found in Appendix A.

1. Literature study.
2. Optimize the Slepian toolbox built in the synthesis project leading up to this thesis such that the following is achieved:
   (a) Computation optimization. Currently, a design matrix for spherical harmonic degree 50 requires more than 60 minutes of computation time. The optimization lies in determination of associated Legendre functions used in computing the kernel matrix.
   (b) Preparation for real data. The toolbox must be prepared to handle satellite 'gradient' data from *Swarm* satellites *Alpha* and *Charlie* (East-West gradient, across-track), and CHAMP satellite (North-South gradient, along-track).
   (c) Finalize and document toolbox with the aim of a GitHub release.
3. Perform studies of several regions of interest. The study will initially be carried out with spherical harmonic degree 200. Regions include:
   (a) Bangui Anomaly.
   (b) Australia.
   (c) Walvis Ridge and surrounding younger parts of the South Atlantic.
   (d) Greenland.
4. Compare results with other models such as the LCS-1 and EMM2015 models.
5. Document the work.

Chapter 2

Data and Regions of Interest

No matter the modelling task at hand, data is always an essential element. Modelling the Earth's magnetic field can be challenging using only stationary observatories due to poor coverage of the globe. This is true even for regional modelling as the observatories within the region may be too sparsely distributed. The introduction of high quality satellite data provide an excellent opportunity to collect densely distributed data around the globe.

As previously mentioned this thesis makes use of data collected by two missions, the CHAMP mission (Maus, 2007) and the *Swarm* mission (e.g. Friis-Christensen et al. (2006) or Olsen et al. (2016)). The following section briefly introduces these missions and the data set used.

2.1 The CHAMP and *Swarm* Satellites

The CHAMP and *Swarm* missions are both near-polar orbits with high inclinations, providing excellent global coverage. Some orbital information of two missions contributing data to this thesis are summarized in Table 2.1 (Olsen et al., 2017). The *Swarm* missions consists of three identical satellites. Two of these, *Swarm Alpha* and *Swarm Charlie* orbit in near identical near-polar orbits, varying only by approximately 1.4° in longitude which is approximately 155 km at the equator (Olsen et al., 2017). This constellation provides a unique opportunity to observe East-West differences at satellite altitude. The third *Swarm* satellite, *Swarm Bravo*, was launched to a higher altitude, also with a near-polar orbit. Data from this satellite is not considered in this thesis.

The CHAMP satellite had its atmospheric re-entry in September 2010, and its final altitude is derived from the data set, with respect to Earth's mean spherical radius $r_E = 6371.2$ km.

Table 2.1: Summary of the two missions providing data for this thesis. All altitudes listed are with respect to the Earth's mean spherical radius.

	Launch	Atmospheric re-entry	Orbital inclination	Initial altitude	Final altitude
CHAMP	2000	2010	87.2°	454 km	244.24 km
Swarm Alpha	2013	-	87.4°	450 km	-
Swarm Charlie	2013	-	87.4°	450 km	-

Following the orbital information of the two missions is an introduction to the data set used in the thesis.

2.2 The Data Set

Undeniably, measuring the magnetic field with any sort of instrument will include signals from all measurable sources. The internal magnetic field of the Earth includes contributions from e.g. the core magnetic field and the lithospheric magnetic field, with the latter being the contribution to be investigated in this thesis. By utilizing satellite data, another layer of complexity is introduced; contributions from sources considered external at Earth's surface are considered internal at satellite altitude, and must thus also be handled. As mentioned this thesis will use an *a priori* model to remove unwanted contributions. This, alongside other considerations are discussed in the following.

Considering first the satellite altitudes. The amplitude of the lithospheric field is in the hundreds of nanoTeslas (nT) at the Earth's surface, where the core field is in the thousands of nT. Moving towards satellite altitude, the amplitude of the lithospheric field will diminish greatly (Olsen et al., 2017). CHAMP had an altitude of 350 km or below in the last four years of the mission, which means this period is particularly interesting for lithospheric field modelling (Olsen et al., 2017). Thus, data from September 2006 to September 2010 is considered. The two lower *Swarm* satellites orbit in constellation from April 2014 with an approximate altitude of 515 km (derived from the data set) which will be the start of the *Swarm* data set. Unlike Olsen et al. (2017), this thesis makes use of an updated data set with 2018 data included, thus expanding the data set utilizing one additional year of *Swarm* data.

Satellites provide large amounts of data in varying environments. Proper selection of data is crucial in order to e.g. remove unwanted signals from effects of the Sun. The criteria applied to data selection are listed below (extracted from (Olsen et al., 2017))

- Changes in the magnetic signature of the magnetospheric ring current must be less than 3 nT hr^{-1}.
  - And the geomagnetic activity index, described by the Kp index, must satisfy $Kp \leq 3^0$.
- Vector observations are included globally (unlike Olsen et al. (2017) where they are limited to $\pm 55^\circ$ equatorwards with respect to quasi-dipole (QD) latitudes).
  - Poleward of $\pm 55^\circ$ QD latitude, vector data is selected when the electric field at the magnetopause $E_m \leq 0.8$ mV m^{-1} and when the direction of the interplanetary magnetic field is northward (i.e. $B_Z > 0$).
- Only dark data is selected, meaning the sun is at least 10° below the horizon.
  - With the exception of North-South gradients which include sunlit data. The sunlit data is not considered around QD latitudes $\pm 10^\circ$ to avoid Equatorial Electrojet contamination.

The data relevant for magnetic field modelling has now been accounted for. Recall the measurements includes contributions from several sources. Obtaining a data set that consists of the lithospheric field is done by subtracting models of other contributions, such as magnetospheric and core field contributions (Olsen et al., 2017). For the core field up to spherical harmonic degree

$L = 14$ as well as the large-scale magnetospheric field the CHAOS-6_x2 model is used (Finlay et al., 2016). The main reason for noise in the lithospheric magnetic field data set is potential unmodelled large-scale magnetospheric contributions (Olsen et al., 2017).

The data set for the lithospheric magnetic field has been constructed, and the following section introduces satellite gradient data.

2.2.1 Satellite Gradient Data

The arguments for working with gradient data are plenty. For one, gradient data are less affected by large-scale external field contributions (Olsen et al., 2017). This is a reasonable statement considering the scales at which these contributions change versus the scale at which the smaller-scale lithospheric contributions change. Gradient data has proven to be less correlated in time, which makes a larger portion of the available data useful for gradient data (Olsen et al., 2017).

The gradient data produced for the data set used in this thesis are entirely horizontal, i.e. no radial gradients are considered (Olsen et al., 2017). The data set contains no magnetic field data, only gradients, with the field data used to construct the gradient data.

There will be two types of gradients considered for the data set, North-South (NS) and East-West (EW). The gradients are constructed by differencing data points.

North-South
The North-South gradient has contributions from all three satellites. The gradient is approximated by (p. 1464 second column in Olsen et al. (2017))

$$\delta B_{NS} = B(t_j, r_j, \theta_j, \phi_j) - B(t_j + 15\ s, r_j + \delta r, \theta_j + \delta\theta, \phi_j + \delta\phi), \tag{2.1}$$

where j denotes either one of the three satellites and t_j, r_j, θ_j and ϕ_j denotes time, altitude, geographical co-latitude and longitude of an observation, respectively. This states that a subsequent measurement obtained 15 seconds later by the same satellite is used to approximate along-track gradient data. The 15 seconds time delay corresponds to an along-track distance of ≈ 115 km or $\approx 1^\circ$ in latitude (Olsen et al., 2017).

East-West
The constellation of the *Swarm* satellites allow for this EW gradient, which is approximated by (p. 1465 first column, second paragraph in Olsen et al. (2017))

$$\delta B_{EW} = B_A(t_1, r_1, \theta_1, \phi_1) - B_C(t_2, r_2, \theta_2, \phi_2), \tag{2.2}$$

where B_A and B_C are *Swarm Alpha* and *Swarm Charlie* observations, respectively. Initially, a *Swarm Alpha* observation fulfilling the selection criteria is chosen. The *Swarm Charlie* observation with the closest co-latitude value is chosen with one additional constraint that the time difference $\delta t = |t_1 - t_2|$ must not exceed 50 seconds (Olsen et al., 2017). The subtraction can be changed, such that a data point from *Swarm Alpha* is subtracted from the chosen data point from *Swarm Charlie*.

Amount of data
The total amount of data, and how these are distributed between satellites and type of gradient is presented in Table 2.2.

Table 2.2: Number of data points (N_{Total}) distributed between the two gradient types. N_{CHAMP} relates to the CHAMP satellite, N_{SA} to *Swarm Alpha* and N_{SC} to *Swarm Charlie*. δB_{EW} for N_{SA} means $B_A - B_C$ in Equation (2.2) and $B_C - B_A$ for N_{SC}.

	N_{CHAMP}	N_{SA}	N_{SC}	N_{Total}
δB_{NS}	477,020	745,593	743,392	1,966,005
δB_{EW}	-	649,914	712,898	1,362,812
N_{Total}	477,020	1,395,507	1,456,290	3,328,817

The vector gradient data set consists of 3,328,817 observations as indicated by the lower right corner cell of Table 2.2.

The toolbox will be tested at various locations on Earth, these are introduced in the following section.

2.3 Regions of Interest

Since this thesis introduces a tool used for regional modelling of the Earth's lithospheric magnetic field, I have selected some regions with particularly interesting features to investigate.

Bangui Anomaly
The biggest anomaly on Earth which is be located at low latitude. Makes for a good test of low latitude modelling and how regional models capture this large anomaly. The gradient data used over this region is presented in Figure 2.1. The Bangui anomaly is very prominent in this region, with large gradient values over the anomaly itself. It will be interesting to see how the regional approach captures this large anomaly.

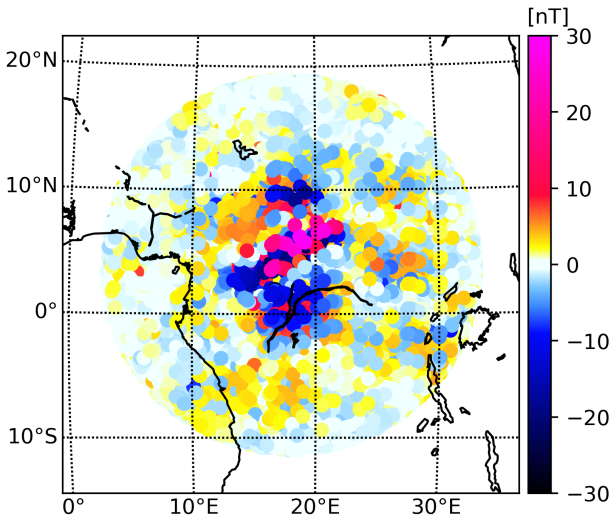


Figure 2.1: Satellite gradient vector data used over the Bangui Anomaly.

Australia

This country is particularly interesting not only because it has many magnetic anomalies but also because it has been subject to extensive aeromagnetic survey, making it one of the more well-studies regions. The gradient data used over this region is presented in Figure 2.2. Australia contains several larger areas of large gradient signal, which could indicate that many features will be visible from modelling this region.

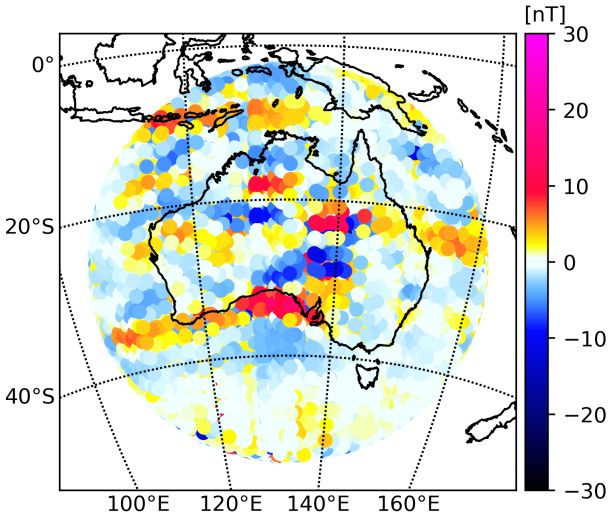


Figure 2.2: Satellite gradient vector data used over Australia.

Walvis Ridge

History of tectonic movement lies within the oceanic crust. These isochrones as well as the larger anomalies in this region makes a great possibility to test oceanic models using the Slepian approach. The gradient data used over this region is presented in Figure 2.3. Clearly, and as expected, the most significant gradient signal is obtained over the Walvis Ridge. There are indications of features in the South South-East region of the spherical cap, and these will be interesting to investigate further.

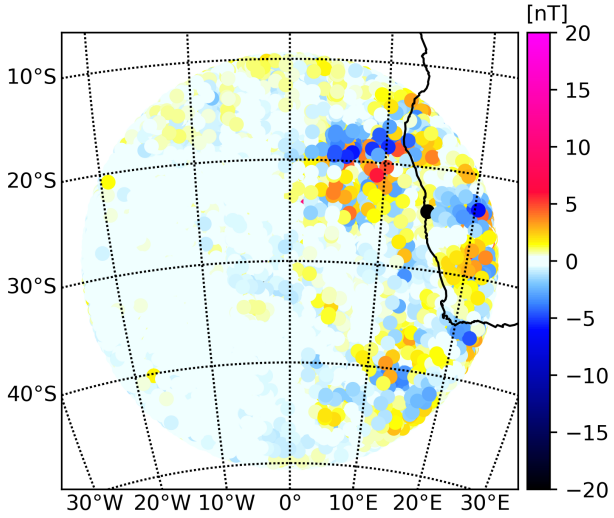


Figure 2.3: Satellite gradient vector data used over the Walvis Ridge.

Greenland

Greenland is a challenging region due to its placement in high latitudes. Models such as LCS-1 shows difficulty modelling this region, and so testing a regional model is how high interest. Greenland thus servers not only as an interesting region, but an excellent test of modelling at high latitudes. The gradient data over this region is presented in Figure 2.4. In this data, cropped to fit over Greenland, there is a North-East trend starting from approximately $(lon, lat) = (60^\circ\ W,\ 61^\circ\ N)$. This is a gradient feature that looks interesting because it is unknown whether it is noise or actual lithospheric magnetic signal. Otherwise nice large areas in the Northern part of Greenland contains signal that could show some interesting features.

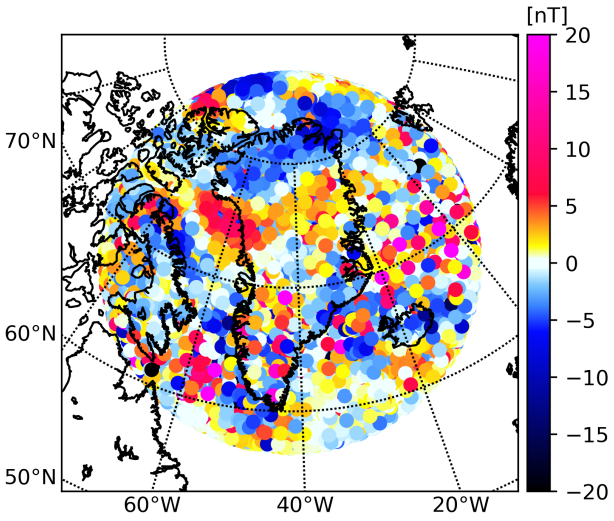


Figure 2.4: Satellite gradient vector data used over Greenland.

This concludes the introduction to the data set used in this thesis. The following chapter introduces

the theory required for constructing regional models of the lithospheric magnetic field using this data set that now includes mostly lithospheric magnetic field signal with the addition of potentially unmodelled large-scale magnetospheric signals.

Chapter 3

Theory of Regional Magnetic Field Modelling

This chapter introduces the necessary theory of regional field modelling using the Slepian approach. First an introduction to the relevant real spherical harmonics theory will be given, followed by how this is used in constructing localized kernel and design matrices and thus regional models.

I note that sections 3.1 through 3.3 follows, with modifications, Rasmus R. Joost (2018) in which I document the initial development of this toolbox.

Unless otherwise stated, notations in the theory will follow that of Plattner and Simons (2017), following is a summary of these (see also Table 3.1 for a visual overview):

> *"In what follows we take pains to distinguish 'light' and 'bold', uncapitalized and capitalized, roman (*g, $\mathbf{G}$*), italicized (*g, G, $\boldsymbol{G}$*), calligraphic (*$\mathcal{G}$, $\boldsymbol{\mathcal{G}}$*) or script (*$\mathscr{G}$*) fonts, depending on whether the quantity of interest is a column vector or a matrix, a scalar value or a scalar function or a vector function, a column vector of scalar functions or of vector functions, or a power spectrum, respectively."* p. 213 Plattner and Simons (2017).

Table 3.1: Notation look-up table.

	Roman		Italicized		Calligraphic		Script	
	Light	Bold	Light	Bold	Light	Bold	Light	Bold
Uncapitalized	Column vector	-	Scalar value	-	-	-	-	-
Capitalized	-	Matrix	Scalar function	Vector function	Column vector of scalar functions	Column vector of vector functions	Power Spectrum	-

3.1 Scalar Potential Fields and Real Spherical Harmonics

Utilizing satellite observations when modelling the Earth's magnetic field, a quantity typically measured is the magnetic field vector $\mathbf{B}$ (Plattner and Simons, 2017). The sources contributing to the magnetic field are divided into two main categories, one related to internal sources and one related to external sources, see Figure 1.2. Gradients of internal source scalar potential field ∇V and external source scalar potential field ∇W are related to the vectorized geomagnetic field $\mathbf{B}$ by

$$\mathbf{B}(r\hat{\boldsymbol{r}}) = \nabla V(r\hat{\boldsymbol{r}}) + \nabla W(r\hat{\boldsymbol{r}}), \tag{3.1}$$

providing V and W satisfies Laplace's equation

$$\nabla^2 \left[V(r\hat{\boldsymbol{r}}) + W(r\hat{\boldsymbol{r}}) \right] = 0, \tag{3.2}$$

which amongst others Blakely (1996) presents a solution for.

The external source scalar potential field W is henceforth omitted from the theory introduced, as this does not originate from the lithospheric magnetic field. Expressing $\nabla = \hat{\boldsymbol{r}}\partial_r + r^{-1}\nabla_1$ and $\nabla_1 = \hat{\boldsymbol{\theta}}\partial_\theta + \hat{\boldsymbol{\phi}}(\sin(\theta))^{-1}\partial_\phi$ in Equation (3.1), the gradient of V expands into

$$\nabla V(r\hat{\boldsymbol{r}}) = \sum_{l=0}^{\infty} \sum_{m=-l}^{l} -r_p^{-1} \left(\frac{r}{r_p} \right)^{-l-2} \mathrm{g}_l^m \Big[\hat{\boldsymbol{r}}(l+1) Y_{lm}(\hat{\boldsymbol{r}}) - \nabla_1 Y_{lm}(\hat{\boldsymbol{r}}) \Big], \tag{3.3}$$

where r_p denotes the reference radius of a planet, r is an arbitrary radius, l are spherical harmonic degrees, and g_l^m are internal source spherical harmonic coefficients at reference radius r_p (Plattner and Simons, 2017).

The internal-field gradient vector spherical harmonics $\boldsymbol{E}_{lm}(\hat{\boldsymbol{r}})$ are given by jointly orthonormalizing the expression in the square bracket of Equation (3.3) (Equation (15) in Plattner and Simons (2017), derived from Freeden and Schreiner (2009) Equations (5.309) and (5.310))

$$\boldsymbol{E}_{lm}(\hat{\boldsymbol{r}}) = \frac{1}{\sqrt{(l+1)(2l+1)}} \big[\hat{\boldsymbol{r}}(l+1) Y_{lm}(\hat{\boldsymbol{r}}) - \nabla_1 Y_{lm}(\hat{\boldsymbol{r}}) \big] \quad \text{for } 0 \leq l \leq L, \tag{3.4}$$

where L is the maximum spherical harmonic degree. The $Y_{lm}(\hat{\boldsymbol{r}})$ term in Equation (3.4) is defined as

$$Y_{lm}(\hat{\boldsymbol{r}}) = Y_{lm}(\theta, \phi) = \begin{cases} \sqrt{2} X_{l|m|}(\theta) \cos(m\phi) & \text{if } -l \leq m < 0 \\ X_{l0}(\theta) & \text{if } m = 0 \\ \sqrt{2} X_{l|m|}(\theta) \sin(m\phi) & \text{if } 0 < m \leq l, \end{cases} \tag{3.5}$$

where θ and ϕ are colatitudes and longitudes, respectively (see Equation (B.72) in Dahlen and Tromp (1998)). The $X_{l|m|}(\theta)$ term introduced in Equation (3.5) is defined as

$$X_{lm}(\theta) = (-1)^m \left(\frac{2l+1}{4\pi} \right)^{1/2} \left[\frac{(l-m)!}{(l+m)!} \right]^{1/2} P_{lm}(\cos(\theta)), \tag{3.6}$$

where $P_{lm}(\cos(\theta))$ are associated Legendre functions and l and m are the angular degrees and orders of the spherical harmonic, respectively (see Equation (B.58) in Dahlen and Tromp (1998)).

Using Equation (3.4) and introducing a spherical harmonic degree dependent continuation operator, the potential is rewritten as

$$\nabla V(r\hat{\mathbf{r}}) = \sum_{l=0}^{\infty} \sum_{m=-l}^{l} A_l(r)\ \mathrm{g}_l^m\ \boldsymbol{E}_{lm}(\hat{\mathbf{r}}), \tag{3.7}$$

where the continuation operator A_l is given by

$$A_l(r) = -r_p^{-1}\sqrt{(l+1)(2l+1)}\left(\frac{r}{r_p}\right)^{-l-2}, \quad \text{for } 0 \leq l \leq L, \tag{3.8}$$

collected in the continuation operator vector $\mathbf{A}(r)$ spanning over all spherical harmonic degrees up to and including L.

The relevant real spherical harmonics have now been accounted for, and the following section will introduce how these are used in regional potential field modelling with the Slepian approach. Furthermore, the following sections introduces silent notation where for example $\mathbf{A}(r_s) = \mathbf{A}$ is the continuation operator at satellite altitude. Thus unless otherwise stated, radially dependent components will be considered at satellite altitude.

3.2 A Slepian Approach to Regional Modelling

The construction of a kernel matrix is crucial in potential field modelling. This matrix provides the mathematical link between satellite observations and model parameters to be estimated. This section introduces how globally defined real spherical harmonics are utilized to define localized kernel matrices and from that obtain AC-GVSF that will be a key element in linking mathematics to observations.

A localized kernel matrix $\mathbf{K}$ is in continuous form defined, like Plattner and Simons (2017) does in their Equation (34), as

$$\mathbf{K} = \mathbf{A}(r_p)\left(\int_R \boldsymbol{\mathcal{E}}_L \cdot \boldsymbol{\mathcal{E}}_L^T\ \mathrm{d}\Omega\right)\mathbf{A}(r_p)^T, \tag{3.9}$$

where $|\cdot|$ denotes the inner products applied to each element pair of vector or matrix vector functions, T denotes the transpose, and $\boldsymbol{\mathcal{E}}_L = (\boldsymbol{E}_{00}, \ldots, \boldsymbol{E}_{lm}, \ldots, \boldsymbol{E}_{LL})^T$ collects $\boldsymbol{E}_{lm}$ for all l where $0 \leq l \leq L$ (note that $\boldsymbol{\mathcal{E}}$ is a bold, capital, and calligraphic E, i.e. a column vector of vector functions, with vector entries given in Equation (3.4)). The integration is over a subregion, R, of the unity sphere, Ω. I note that this localized kernel matrix is independent of data and can be computed prior to any analyses. $\mathbf{K}$ has matrix entries (Equation (13) in Plattner and Simons (2015)) given by

$$K_{lm,l'm'} = A_l(r_p)\left(\int_R \boldsymbol{E}_{lm} \cdot \boldsymbol{E}_{l'm'}\ \mathrm{d}\Omega\right)A_{l'}(r_p), \tag{3.10}$$

and dimensions $(L+1)^2 \times (L+1)^2$ and is a real, symmetric, and positive definite matrix. At high spherical harmonic degree and satellite altitude the localized kernel matrix becomes highly ill-conditioned (Plattner and Simons, 2017). This issue is addressed by a singular value decomposition (SVD, see e.g. Section 3.1 of Aster et al. (2013)) approach. Working with eigenvector decomposition of $\mathbf{K}$ and focusing only on relatively large eigenvalues and their well-determined

eigenvectors the kernel matrix becomes better conditioned (Plattner and Simons, 2017). A truncation parameter J is introduced that satisfies $J \leq (L+1)^2$.

Working with the truncated eigenvector decomposition and considering the properties of $\mathbf{K}$, its eigenvectors are orthogonal, and when orthonormalized the eigenvector decomposition is

$$\mathbf{K}\mathbf{G}_J = \mathbf{G}_J\mathbf{\Lambda}_J, \tag{3.11}$$

where $\mathbf{\Lambda}_J$ is a diagonal matrix with descending real-valued eigenvalues of $\mathbf{K}$ with $1 < \lambda_1 \leq \lambda_2 \leq \ldots \leq \lambda_J$, and columns of the $\mathbf{G}_J$ matrix are sorted eigenvectors of $\mathbf{K}$. Column entries of $\mathbf{G}$ are denoted g_α, and are arranged (Equation (37) in Plattner and Simons (2017))

$$\mathbf{G}_J = (\mathrm{g}_1, \ldots, \mathrm{g}_\alpha, \ldots, \mathrm{g}_J), \tag{3.12}$$

where the remaining eigenvectors $(g_{J+1}, \ldots, g_\alpha, \ldots, g_{(L+1)^2})$ and eigenvalues $(\lambda_{J+1}, \ldots, \lambda_\alpha, \ldots, \lambda_{(L+1)^2})$ have been cut out. Instead of the full dimension of $(L+1)^2 \times (L+1)^2$, the matrix has been reduced to dimension $(L+1)^2 \times J$. The $\mathbf{G}_J$ matrix makes up the Slepian basis. This basis is used to construct the AC-GVSF as described below.

Each g_α of Equation (3.12) contains a set of spherical harmonic coefficients

$$\mathrm{g}_\alpha = (g_{00,\alpha}, \ldots, g_{lm,\alpha}, \ldots, g_{LL,\alpha})^T, \tag{3.13}$$

which can be expanded into the gradient vector spherical harmonic basis and thus defining AC-GVSF by (Equation (41) in Plattner and Simons (2017))

$$\boldsymbol{G}_\alpha(r_p\hat{\boldsymbol{r}}) = \sum_{l=0}^{L}\sum_{m=-l}^{l} A_l(r_p) g_{lm,\alpha} \boldsymbol{E}_{lm}(\hat{\boldsymbol{r}}). \tag{3.14}$$

It is noted, that $\boldsymbol{G}_{\uparrow\alpha}$ is capital, bold and italic, making it a matrix vector function.

Vector functions of Equation (3.14) are collected in a column vector of vector functions to include all AC-GVSF up to and including J (Equation (42) in Plattner and Simons (2017))

$$\boldsymbol{\mathcal{G}}_J = \left(\boldsymbol{G}_1, \ldots, \boldsymbol{G}_\alpha, \ldots, \boldsymbol{G}_J\right)^T = \mathbf{G}_J^T\mathbf{A}(r_p)\boldsymbol{\mathcal{E}}_L, \quad \text{with } 1 \leq J \leq (L+1)^2. \tag{3.15}$$

Equation (3.15) can be upwards continued by applying a radius $r > r_p$ in the continuation operator, which will be useful when constructing the design matrix used in the inverse problem

$$\boldsymbol{\mathcal{G}}_{\uparrow J} = \left(\boldsymbol{G}_{\uparrow 1}, \ldots, \boldsymbol{G}_{\uparrow\alpha}, \ldots, \boldsymbol{G}_{\uparrow J}\right)^T = \mathbf{G}_J^T\mathbf{A}(r)\boldsymbol{\mathcal{E}}_L, \quad \text{with } 1 \leq J \leq (L+1)^2. \tag{3.16}$$

The truncation parameter J is an essential part of constructing a good model. This parameter is a measure of how many globally defined real spherical harmonic functions are combined to construct the localized kernel matrix. It directly affects the amount of AC-GVSF constructed which will impact the inverse problem introduced in the following section. A good choice of J is not straight-forwardly achieved and I use diagnostic tools introduced in Section 5.1.1 and 3.4 to obtain indications of whether the choices made are good or not.

3.3 The Inverse Problem

The purpose of this thesis is to produce regional models of the lithospheric magnetic field. Therefore relating theory to practice is of high relevance. Generally speaking, the inverse problem requires a design matrix which relates model parameters to satellite observations. This matrix is related to the localized kernel matrix introduced in the previous section, and what follows in this section is how the AC-GVSF obtained from the localized kernel matrix are utilized in constructing the design matrix required to solve the inverse problem.

Traditionally in potential field modelling of the lithospheric magnetic field the estimated model parameters are the Gauss coefficients g_l^m (see e.g. Olsen et al. (2017)). These are globally optimized model parameters. This thesis introduces the Slepian model parameters m_J. These are regionally optimized model parameters. The truncation parameter J is noted in the subscript of the Slepian model parameters. For each AC-GVSF used to construct the design matrix, and thus used to describe the regional lithospheric magnetic field, an additional Slepian model parameter must be accounted for. However, for spherical harmonic degree $L = 185$ there are $34,595$ Gauss coefficients to account for. The Slepian approach offers a significant reduction in amount of model parameters which will become evident later in this thesis.

Recall that the localized kernel matrix introduced in Equation (3.9) is in continuous form. Satellite observations are obtained at discrete locations and not continuously. Under the assumption that observation locations are dense within a region, the localized kernel matrix can remain in its continuous form (Plattner and Simons, 2017). If the assumption does not hold, Equation (3.9) needs to be defined discretely. Some discretization is nevertheless necessary, and it is achieved by evaluating the $\boldsymbol{E}_{lm}$ at data locations, and upwards continue to satellite altitudes

$$\left[\mathbf{E}^{sat}_{lm,i}\right]_c = A_l(r_i)\boldsymbol{E}_{lm}(\hat{\mathbf{r}}_i)\cdot\hat{\mathbf{c}}, \tag{3.17}$$

where $\hat{\mathbf{c}}$ denotes indexing for the radial, colatitudinal and longitudinal unit vectors, and i denotes the i'th datum. $[\mathbf{E}^{sat}_{lm,i}]_c$ is assembled into contributions from the three components by

$$\left[\mathbf{E}^{sat}\right]_c = \begin{bmatrix} \left[\mathbf{E}^{sat}_{00,1}\right]_c & \cdots & \left[\mathbf{E}^{sat}_{00,k}\right]_c \\ \vdots & \ddots & \vdots \\ \left[\mathbf{E}^{sat}_{LL,1}\right]_c & \cdots & \left[\mathbf{E}^{sat}_{LL,k}\right]_c \end{bmatrix} \quad \text{into} \quad \mathbf{E}^{sat} = \Big(\left[\mathbf{E}^{sat}\right]_r \ \left[\mathbf{E}^{sat}\right]_\theta \ \left[\mathbf{E}^{sat}\right]_\phi \Big), \tag{3.18}$$

and the dimensions of $\mathbf{E}^{sat}$ is $(L+1)^2 \times 3k$, where k denotes the length of e.g. radial component of the data vector (Equation (84) in Plattner and Simons (2017)).

Through the Slepian approach a design matrix is obtained by multiplying Equation (3.18) with the Slepian basis from Equation (3.12)

$$\mathbf{G}^{sat}_J = \left(\mathbf{G}^T_J \ \mathbf{E}^{sat}\right)^T, \tag{3.19}$$

which is a key ingredient in posing the inverse problem to be solved. A statement of the inverse problem follows, minimizing the L_2-norm of the data misfit, which is the difference between satellite observations and model prediction (Equation (86) in (Plattner and Simons, 2017))

$$\arg \min_{\mathrm{m}_J} \|\mathbf{G}_J^{sat}\ \mathrm{m}_J - \mathrm{d}\|^2, \tag{3.20}$$

where m_J is a vector of Slepian model parameters to be estimated, and d is a vector containing satellite vector observations (see Chapter 2 for an introduction of the data). This is in fact an optimization problem on a regional scale. This translates to the more well-known least squares (LS) problem

$$\left(\left(\mathbf{G}_J^{sat}\right)^T \mathbf{G}_J^{sat}\right) \mathrm{m}_J = \left(\mathbf{G}_J^{sat}\right)^T \mathrm{d}. \tag{3.21}$$

Equation (3.21) defines a $J \times J$ symmetric system with its condition number dependent on the truncation, J (Plattner and Simons, 2017).

The model prediction is a vector with $3k$ entries, where each k set of vector entries represent radial, colatitudinal and longitudinal components, respectively, and is obtained by

$$\mathrm{d}_{pred}^{sat} = \mathbf{G}_J^{sat}\ \mathrm{m}_J. \tag{3.22}$$

Producing model predictions at the Earth's surface requires producing a design matrix at the Earth's surface. This is obtained by evaluating the $\boldsymbol{E}_{lm}$ at a fine coordinate grid with the Earth's mean spherical radius as evaluation radius,

$$\mathbf{G}_J^{r_p} = \left(\mathbf{G}_J^T\ \mathbf{E}^{r_p}\right)^T, \tag{3.23}$$

where $r_p = 6371.2$ km, the mean spherical radius of the Earth, followed by multiplying the model parameters onto this design matrix

$$\mathrm{d}_{pred}^{r_p} = \mathbf{G}_J^{r_p}\ \mathrm{m}_J. \tag{3.24}$$

The internal source spherical harmonic Gauss coefficients can be retrieved using the Slepian approach by

$$\mathrm{g}_l^m = \boldsymbol{\mathcal{G}}_J\ \mathrm{m}_J, \tag{3.25}$$

which will be further discussed in Section 3.4.

With the regionally optimized inverse problem accounted for, the localized gradient inverse problem will be introduced. Since gradient satellite data is the data of choice, and as such the inverse problem must be looked further into.

3.3.1 Localized Gradient Inverse Problem

Minor modifications must be made to the inverse problem, such that it allows for gradient satellite data to be utilized. Because data is differenced along-track or across-track (see Chapter 2 for data introduction), so must design matrices be differenced to construct a localized gradient design matrix. Like Olsen et al. (2017) the gradient design matrices are constructed by differencing design matrices with sets of positions corresponding to those used to obtain the gradient data.

Initially considered is the gradient predictions at satellite altitude

$$\mathrm{d}_1 - \mathrm{d}_2 = \left(\mathbf{G}_{1,J}^{sat} - \mathbf{G}_{2,J}^{sat}\right) \mathrm{m}_J \tag{3.26}$$

or more compactly written

$$\mathrm{d}_{grad} = \mathbf{G}^{sat}_{grad}\ \mathrm{m}_J, \tag{3.27}$$

which compared with Equation (3.22) states that the same J set of model parameters obtained using non-gradient data and design matrices can be obtained through gradient data and design matrices. Equation (3.21) is rewritten to include the gradient data and localized design matrices by

$$\left(\left(\mathbf{G}^{sat}_{grad,J}\right)^T \mathbf{G}^{sat}_{grad,J}\right) \mathrm{m}_J = \mathbf{G}^{sat}_{grad,J}\ \mathrm{d}_{grad}. \tag{3.28}$$

Going forward only localized gradient design matrices alongside vector gradient data will be discussed in relation to solving the inverse problems.

3.3.2 Robust Estimation

Gross outliers and along-track or across-track correlations in satellite gradient observations are expected. Furthermore, the simple LS solution assumes Gaussian distributed noise, which is rarely the case in satellite observations. A method of handling outliers is by introducing data weights with the Iteratively Re-weighted Least Squares (IRLS) method (Constable, 1988).

The IRLS method introduces a new term $\mathbf{W}_h$ in the LS algorithm

$$\left(\left(\mathbf{G}^{sat}_{J}\right)^T \mathbf{W}_h \mathbf{G}^{sat}_{J}\right) \mathrm{m}_J = \left(\mathbf{G}^{sat}_{J}\right)^T \mathbf{W}_h \mathbf{d}, \tag{3.29}$$

where $[\mathbf{W}_h]_{i,i} = w_{h,i}/\sigma_i^2$, where $w_{h,i}$ are huber weights and σ_i^2 are data variances. Huber weighting will be the method of choice in this thesis as also noted in the subscript of $\mathbf{W}_h$.

If $\mathbf{W}_h = \mathbf{I}$ i.e. the data weight matrix is the identity matrix the LS algorithm is obtained.

Huber weights are computed using

$$w_{huber} \propto \begin{cases} c/\mathrm{e} & \mathrm{e} > c \\ 1 & |\mathrm{e}| \leq c \\ -c/\mathrm{e} & \mathrm{e} < -c \end{cases} \tag{3.30}$$

where $c = 1.5$ is a breakpoint constant and e is a data misfit vector (Huber, 1996). Another usual value used is $c = 1.345$ which produces 95% efficiency for Gaussian distributed noise.

Since the process is iterative it is not unlikely that the first iteration is the LS solution. From that the weights are updated with each iteration until some convergence criteria is met. In this thesis the convergence criteria will be based on the two-norm of the model parameters. When a change less than 0.01 per-cent is achieved, the process is terminated and the final weights are thus chosen.

Robust estimation is an excellent method of handling non-Gaussian noise in satellite data. In high resolution modelling the noise distribution is not the only issue to overcome. Regularization will be the method of choice to overcome these challenges presented in the following section.

3.3.3 Regularization

Working with real satellite observations constructing high resolution models requires regularization. This is not only due to complex distributed noise and how it scales at high spherical harmonic degree, causing instability in models, but these problems may also be highly ill-conditioned. Recall Section 3.2 where it is introduced that the localized kernel matrix becomes highly ill-conditioned at high spherical harmonic degree and satellite altitudes. This sufficiently indicates that the simple LS or IRLS methods will not suffice for this thesis.

The regularization problem introduces a cost function to be minimized

$$\Phi = \Phi_{data} + \alpha^2 \Phi_{model}, \tag{3.31}$$

where Φ_{data} is a function related to the L_2-norm of the data misfit (the LS solution) and Φ_{model} is a function related to some L_p-norm of the model (the model regularization term). Expanding the two terms to be minimized and introducing robust estimation yields

$$\Phi(\mathrm{m}_{J,\alpha}) = \left(\mathrm{d} - \mathbf{G}_J^{sat}\mathrm{m}_{J,\alpha}\right)^T \mathbf{W}_h \left(\mathrm{d} - \mathbf{G}_J^{sat}\mathrm{m}_{J,\alpha}\right) + \alpha^2 \|\mathbf{R}\mathrm{m}_{J,\alpha}\|_p, \tag{3.32}$$

where α^2 is a regularization parameter, $\mathbf{R}$ is a regularization matrix and p is an integer determining what norm to minimize, i.e. $p = 1$ means the L_1-norm of the model is minimized. Henceforth this thesis will be concerned with $p = 1$ and $p = 2$ only.

The cost function in fact introduces a trade-off between minimizing the L_2-norm of the data misfit and the model L_p-norm. With a very small α^2 the solution approaches the LS solution, while a very large α^2 will minimize the L_p-norm of the model.

3.3.3.1 L_2-norm Model Regularization

One of the model norm regularizations considered in this thesis is the L_2-norm. This method effectively minimizes $|\mathbf{B}_r|^2$, i.e. the squared radial component of the lithospheric magnetic field over the region of interest (Olsen et al., 2017).

The cost function introduced in Equation (3.32) is adjusted to the $p = 2$ solution

$$\begin{aligned}\Phi(\mathrm{m}_{J,\alpha,L_2}) &= \left(\mathrm{d} - \mathbf{G}_J^{sat}\mathrm{m}_{J,\alpha,L_2}\right)^T \mathbf{W}_h \left(\mathrm{d} - \mathbf{G}_J^{sat}\mathrm{m}_{J,\alpha,L_2}\right) + \alpha^2 \|\mathbf{R}\mathrm{m}_{J,\alpha,L_2}\|_2 \\ &= \left(\mathrm{d} - \mathbf{G}_J^{sat}\mathrm{m}_{J,\alpha,L_2}\right)^T \mathbf{W}_h \left(\mathrm{d} - \mathbf{G}_J^{sat}\mathrm{m}_{J,\alpha,L_2}\right) + \alpha^2 \mathrm{m}_{J,\alpha,L_2}^T \mathbf{R}^T \mathbf{R}\mathrm{m}_{J,\alpha,L_2}.\end{aligned} \tag{3.33}$$

The solution to minimizing Equation (3.33) becomes

$$\left(\left(\mathbf{G}_J^{sat}\right)^T \mathbf{W}_h \mathbf{G}_J^{sat} + \alpha^2 \mathbf{R}^T \mathbf{R}\right)\mathrm{m}_{J,\alpha,L_2} = \mathbf{G}_J^{sat} \mathbf{W}_h \mathrm{d}, \tag{3.34}$$

and solving this for the model parameters yields the algorithm

$$\mathrm{m}_{J,\alpha,L_2} = \left(\left(\mathbf{G}_J^{sat}\right)^T \mathbf{W}_h \mathbf{G}_J^{sat} + \alpha^2 \mathbf{R}^T \mathbf{R}\right)^{-1} \mathbf{G}_J^{sat} \mathbf{W}_h \mathrm{d}. \tag{3.35}$$

The algorithm is solved with the IRLS approach, updating data weights until convergence, for a range of α^2 values. The range of models produced are used to provide insights in choosing a good value for α^2, which is not a trivial task. For each model the model norm $\|\mathrm{m}_{J,\alpha,L_2}\|$ and residual norm $\|\mathrm{d} - \mathbf{G}_J^{sat}\mathrm{m}_{J,\alpha,L_2}\|$ are produced using

$$\|\mathrm{m}_{J,\alpha,L_2}\| = \mathrm{m}_{J,\alpha,L_2}\mathbf{R}^T\mathbf{R}\mathrm{m}_{J,\alpha,L_2} \tag{3.36}$$

$$\|\mathrm{d} - \mathbf{G}_J^{sat}\mathrm{m}_{J,\alpha,L_2}\| = \left(\mathrm{d} - \mathbf{G}_J^{sat}\mathrm{m}_{J,\alpha,L_2}\right)^T \mathbf{W}_h \left(\mathrm{d} - \mathbf{G}_J^{sat}\mathrm{m}_{J,\alpha,L_2}\right), \tag{3.37}$$

which can be used to produce an L-curve (Aster et al., 2013). The L-curve is an excellent tool for visualizing the trade-off between minimizing the data misfit L_2-norm and the model L_p-norm.

Because data uncertainties are known, the discrepancy principle is worth investigating (Aster et al., 2013). The discrepancy principle suggests choosing α^2 that produces the simplest model whose data misfit agrees with the data uncertainty. This is obtained by

$$\left(\mathrm{d} - \mathbf{G}_J^{sat}\mathrm{m}_{J,\alpha,L_2}\right)\left(\mathrm{d} - \mathbf{G}_J^{sat}\mathrm{m}_{J,\alpha,L_2}\right) = N\sigma^2, \tag{3.38}$$

where N is the total amount of data, and σ^2 are data variances. With discrepancy normalized data misfit, the chosen α^2 will be the value that produces data misfit norm equal to one.

3.3.3.2 L_1-norm Model Regularization

The second type of regularization used in this thesis is the L_1-norm model regularization, which effectively minimizes $|\mathbf{B}_r|$ over the region of interest (Olsen et al., 2017). Minimizing the L_1-norm of the model is more complicated than minimizing the L_2-norm of the model. The L_1-norm of the model enforces sparsity in an attempt to fit the data using as few model parameters as possible (Chris Finlay, 2017). This requires implementing model weights, which will be introduced in the following.

The cost function associated with the L_1-norm model regularization is

$$\begin{aligned}\Phi(\mathrm{m}_{J,\alpha,L_1}) &= \left(\mathrm{d} - \mathbf{G}^{sat}\mathrm{m}_{J,\alpha,L_1}\right)^T \mathbf{W}_h \left(\mathrm{d} - \mathbf{G}_J^{sat}\mathrm{m}_{J,\alpha,L_1}\right) + \alpha^2\|\mathbf{R}\mathrm{m}_{J,\alpha,L_1}\|_1 \\ &= \left(\mathrm{d} - \mathbf{G}_J^{sat}\mathrm{m}_{J,\alpha,L_1}\right)^T \mathbf{W}_h \left(\mathrm{d} - \mathbf{G}_J^{sat}\mathrm{m}_{J,\alpha,L_1}\right) + \alpha^2\mathrm{m}_{J,\alpha,L_1}^T\mathbf{R}^T\mathbf{W}_m\mathbf{R}\mathrm{m}_{J,\alpha,L_1},\end{aligned} \tag{3.39}$$

where $\mathbf{W}_m$ is a model weight diagonal matrix. These weights make use of Ekblom's measure as introduced by Farquharson and Oldenburg (1998). Diagonal matrix entries are for the L_1-norm model regularization defined as

$$[W_m]_{i,i} = \frac{1}{\left((\mathbf{R}\mathrm{m}_{J,\alpha,L_1})^2 + \epsilon^2\right)^{1/2}}, \tag{3.40}$$

where $\mathbf{R}\mathrm{m}_{J,\alpha,L_1}$ is a model prediction of the radial component of the magnetic field at Earth's surface for a given set of model parameters $\mathrm{m}_{J,\alpha,L_1}$ and ϵ is Ekblom's measure. $\epsilon = 10^{-6}$ nT is a value selected to ensure numerical stability for very small values of the predicted field.

The solution to minimizing Equation (3.39) is

$$\left(\left(\mathbf{G}_J^{sat}\right)^T \mathbf{W}_h \mathbf{G}_J^{sat} + \alpha^2 \mathbf{R}^T \mathbf{W}_m \mathbf{R} \right) \mathrm{m}_{J,\alpha,L_1} = \mathbf{G}_J^{sat} \mathbf{W}_h \mathrm{d}, \tag{3.41}$$

and solving this for the model parameters yields the algorithm

$$\mathrm{m}_{J,\alpha,L_1} = \left(\left(\mathbf{G}_J^{sat}\right)^T \mathbf{W}_h \mathbf{G}_J^{sat} + \alpha^2 \mathbf{R}^T \mathbf{W}_m \mathbf{R} \right)^{-1} \mathbf{G}_J^{sat} \mathbf{W}_h \mathrm{d}, \tag{3.42}$$

which is a non-linear system of equations due to $\mathbf{W}_m$ being dependent on the model parameters (Chris Finlay, 2017). The L_1-norm model regularization is more complicated as a consequence of this non-linearity. The approach for solving Equation (3.42) follows that of the IRLS where now both data weights $\mathbf{W}_h$ and model weights $\mathbf{W}_m$ are updated with each iteration.

L-curves and the discrepancy principle are used for this method and model norm $\|\mathrm{m}_{J,\alpha,L_1}\|$ and data misfit norm $\|\mathrm{d} - \mathbf{G}_J^{sat}\mathrm{m}_{J,\alpha,L_1}\|$ are computed by

$$\|\mathrm{m}_{J,\alpha,L_1}\| = \mathrm{m}_{J,\alpha,L_1} \mathbf{R}^T \mathbf{W}_m \mathbf{R} \mathrm{m}_{J,\alpha,L_1} \tag{3.43}$$

$$\|\mathrm{d} - \mathbf{G}_J^{sat}\mathrm{m}_{J,\alpha,L_1}\| = \left(\mathrm{d} - \mathbf{G}_J^{sat}\mathrm{m}_{J,\alpha,L_1}\right)^T \mathbf{W}_h \left(\mathrm{d} - \mathbf{G}_J^{sat}\mathrm{m}_{J,\alpha,L_1}\right). \tag{3.44}$$

The discrepancy principle will also be worth investigating for the L_1-norm model regularized solution. It is much alike Equation (3.38) with only model parameters changed

$$\left(\mathrm{d} - \mathbf{G}_J^{sat}\mathrm{m}_{J,\alpha,L_1}\right) \left(\mathrm{d} - \mathbf{G}_J^{sat}\mathrm{m}_{J,\alpha,L_1}\right) = N\sigma^2. \tag{3.45}$$

This concludes the two regularization methods utilized in this thesis. What follows is a brief explanation to a subject only mentioned in both L_1-norm and L_2-norm model regularization: the regularization matrix.

3.3.3.3 Regularization Matrix

In geomagnetism problems the regularization matrix is typically a prediction of the radial component of the Earth's magnetic field at mean spherical radius on a regular grid used as a-priori knowledge to the model (Olsen et al., 2017). The grid must be regular to avoid convergence issues near the poles. This thesis uses the icosahedral approach to constructing a regular grid as introduced in e.g. Baumgardner and Frederickson (1985).

An potential issue that must be accounted for is aliasing. Aliasing in spherical harmonic analysis is where higher spherical harmonic degrees (i.e. smaller length-scales) can not be distinguished from lower spherical harmonic degrees (i.e. larger length-scales). When aliasing occurs, the power of certain spherical harmonic degrees may be misinterpreted as power from other spherical harmonic degrees. A coarse grid will likely cause aliasing while a fine grid will more easily distinguish various length-scales (Gubbins, 2004).

The number of points needed to avoid aliasing at spherical harmonic degree $L = 200$ on a global grid can be found in Table 2 in Kother (2017) who states that at least $51,000$ points are required.

3.4 The Geomagnetic Power Spectrum

Mauersberger (Mauersberger, 1956) and Lowes (Lowes, 1966) demonstrated how the variance of the total magnetic field can be divided into sums of individual contributions (Maus, 2008). This variance is called the Mauersberger-Lowes spectrum and is defined as

$$R_l = (l+1)\left(\frac{r_e}{r}\right)^{2l+4} \sum_{m=-l}^{l} \left(g_l^m\right)^2, \tag{3.46}$$

where $r_e = 6371.2$ km is the mean spherical radius of the Earth and g_l^m are internal source Gauss coefficients.

The Mauersberger-Lowes spectrum is globally defined on the spherical approximation of the Earth. Furthermore, the spherical harmonic Gauss coefficients are required to compute this spectrum.

The power spectra computed by Equations (3.46) and (3.51) are not comparable. The localized internal source Gauss coefficients retrieved via Equation (3.25) can be applied in Equation (3.46) for an approximated global power spectrum. Because the area over which the localized Gauss coefficients are computed is not global a correction must be introduced. The fraction of the spherical cap surface area with respect to Earth's spherical surface area is computed by determining first

$$h = r_e\left(1 - \cos(\theta)\right) \tag{3.47}$$

$$r = \sqrt{2r_e h - h^2}, \tag{3.48}$$

which combines to the spherical cap surface area

$$s_{cap} = \pi\left(r^2 + h^2\right). \tag{3.49}$$

The spherical surface area of the Earth is

$$s_{earth} = 4\pi r_e^2, \tag{3.50}$$

and the ratio of Equations (3.50) and (3.49) respectively provides the scaling required to approximate the localized internal source Gauss coefficient power over the entire globe. This diagnostic tool provides an excellent opportunity of investigating whether the model parameters are stable, and how the power of a certain region is distributed across the spherical harmonic degrees considered.

Locally, the power spectrum of each AC-GVSF is defined by (Equation (38) in Plattner and Simons (2017))

$$\mathscr{G}_\alpha(l) = \frac{1}{2l+1} \sum_{m=-l}^{l} g_{lm,\alpha}^2, \tag{3.51}$$

which provides an excellent opportunity of investigating the power distribution of the AC-GVSF which serves as a diagnostic tool for choosing the Slepian truncation parameter J.

Chapter 4

Implementation as a Python Toolbox

This chapter introduces the Python toolbox developed, which is a toolbox built for producing regional models of magnetic fields with a Slepian approach. Implementations of key parts of the theory is discussed and presented in the following, where computation of Legendre functions, kernel matrices, design matrices and the inverse problem and data selection within a region are considered. The fundamentals of the Python toolbox builds on an existing MATLAB toolbox developed by Alain Plattner (2017).

First an introduction to the flow of the Python toolbox is presented in Figure 4.2. Explanations of the flow chart icons is found in Figure 4.1. When computing regional models using this Python toolbox, an initialization script is first run. This contains information of location of the spherical cap (region of interest), spherical harmonic degree of the analysis, the Slepian truncation parameter J and regularization parameters. Next satellite data within the chosen region is extracted from the data set described in Chapter 2.

The Python toolbox regularly saves large matrices to avoid unnecessary re-computation. Therefore,the next step is a `try`-statement, which initially searches for existing kernel matrix, design matrix and eigenvalues. If this fails, it will go on to another `try`-statement, finding kernel matrix and eigenvalues alone. If the first `try` statement is successful the Python toolbox will move directly to the inversions. If it the second `try`-statement succeeds a design matrix will be computed based on the loaded kernel matrix. If this `try`-statement fails, both kernel matrix, design matrix and eigenvalues will be computed.

Whether design matrix and kernel matrix are computed or loaded, the next step is choosing the type of inversion. These choices are specified in the initialization script, and it is possible to choose to perform LS, L_1-norm and L_2-norm regularization individually, in pairs or all at once. Prior to inversions the regularization matrix is computed (not used for LS).

When model parameters have been obtained, prediction maps are created and plotted, visualizing the results. Furthermore diagnostics such as power spectra can be computed for the model parameters and data misfit.

Figure 4.1: Flow chart icons.

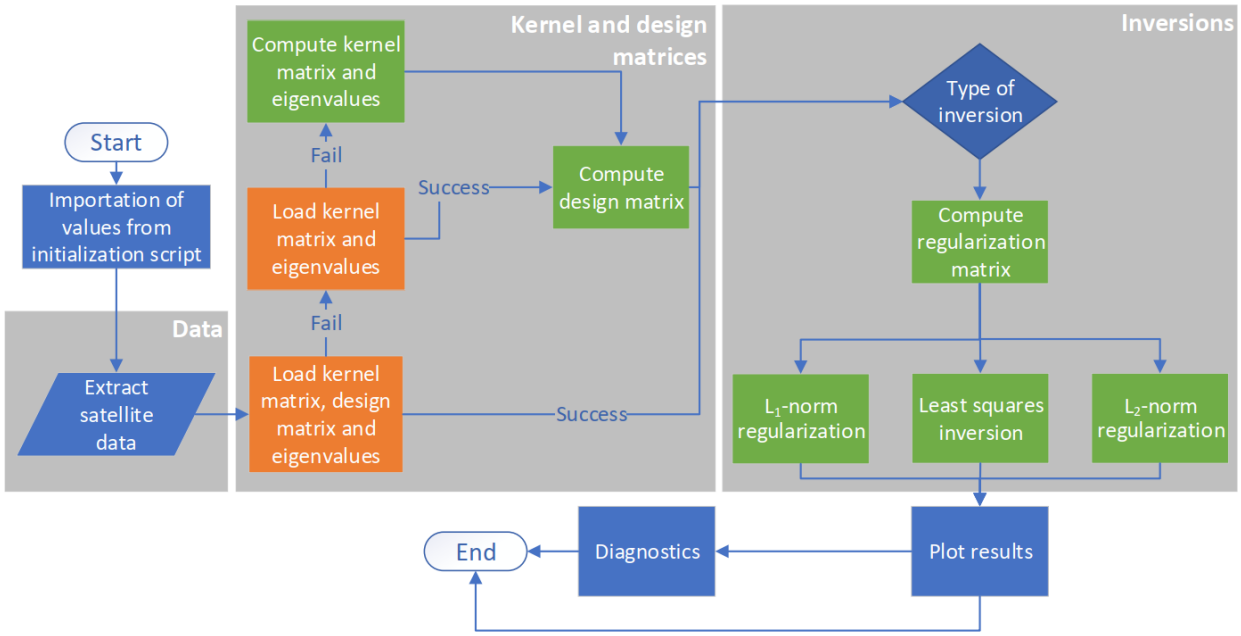


Figure 4.2: Python toolbox flow chart. Orange colored icons indicates a `try` statement, green icons computations and blue icons miscellaneous.

The flow chart shown in Figure 4.2 contains the general overview, with many black boxes performing various tasks. In the following sections the development of the key elements of the Python toolbox will be described.

4.1 Computing Legendre Functions

In spherical harmonic analysis a key aspect of constructing a kernel matrix is the computation of Legendre functions, as the Legendre functions are associated with the X_{lm} of Equation (3.6).

Initially another toolbox was utilized to calculate the Legendre functions, called `GMT_tools` written by Nils Olsen[1] (see Appendix F). The `GMT_tools` toolbox includes a function called `get_Pnm()` that outputs Legendre functions and derivatives with respect to co-latitude for all spherical harmonic degrees and orders up to `nmax` in a 3D tensor. The function uses the recursion scheme from Langel (1987) in their Equation (27) and Table 2.

The implementation of the recursion scheme is shown in Listing 4.1. The `GMT_tools` toolbox resulted was used in the first iteration of the Python toolbox, for all computations of Legendre

[1] Professor of Geophysics, Division of Geomagnetism, DTU Space. http://www.space.dtu.dk/english/service/phonebook/person?id=38306&cpid=45460&tab=1.

functions. One of the milestones in this project, was the optimization of the Python toolbox in order to produce high resolution models. During development, as documented in Rasmus R. Joost (2018), I had issues with computation times being extraordinarily high even for a low spherical harmonic case.

I located the issue being in the construction of the kernel matrix, and more specifically when computing Legendre functions for the radial spherical harmonic functions, later introduced in Section 4.2. Prior to the newly implemented solution, these were computed using the `get_Pnm()` function from the `GMT_tools` toolbox. The computation time of a kernel matrix of spherical harmonic degree $L = 50$ was one hour. Computing a kernel matrix of degree $L = 200$ was never attempted with this solution. To bring down the computation time, a rewritten version of the function `get_Pnm()` called `get_Pm()` was implemented, and can also be found in the `GMT_tools` toolbox (Appendix F). This implementation was necessary because the input co-latitudes changes between iterations, so one comprehensive computation using `get_Pnm()` will not suffice. It computes only Legendre functions for a given spherical harmonic order m up to a given degree L_{max} satisfying $L_{min} > m$, also with respect to input co-latitudes. No co-latitudinal derivatives are computed. This simplifies the output to a 2D array and reduces the computation time of a spherical harmonic degree $L = 50$ kernel matrix to two minutes, 30 times faster than the first iteration. A kernel matrix of spherical harmonic degree $L = 200$ takes between 13-18 hours to compute, depending on the size of the cap. This serves as a good indication that high resolution modelling would be a tedious task without the implemented optimization. The new implementation is introduced in Listing 4.2

Listing 4.1: `get_Pnm()` from Appendix F.

```
import numpy as np
def get_Pnm(nmax, theta):
    """
    Calculation of associated Legendre functions P(n,m) (Schmidt
        normalized)
    and its derivative dP(n,m) vrt. theta.

    Input: theta[:] co-latitude (in rad)
           nmax  maximum spherical harmonic degree
    Output: Pnm    ndarray PD with Legendre functions

    P(n,m) ==> Pnm(n,m) and dP(n,m) ==> Pnm(m,n+1)
    """

    costh = np.cos(theta)
    sinth = np.sqrt(1-costh**2)

    Pnm = np.zeros((nmax+1, nmax+2, len(theta)))
    Pnm[0][0] = 1
    Pnm[1][1] = sinth

    rootn = np.sqrt(np.arange(0, 2*nmax**2+1))

#     Recursion relations after Langel "The Main Field" (1987),
# eq. (27) and Table 2 (p. 256)
```

```
25      for m in np.arange(0, nmax):
26  #           Pnm_tmp = np.sqrt(m+m+1)*Pnm[m][m]
27              Pnm_tmp = rootn[m+m+1]*Pnm[m][m]
28              Pnm[m+1][m] = costh*Pnm_tmp
29              if m > 0:
30                  Pnm[m+1][m+1] = sinth*Pnm_tmp/rootn[m+m+2]
31              for n in np.arange(m+2, nmax+1):
32                  d = n*n - m*m
33                  e = n + n - 1
34                  Pnm[n][m] = (e*costh*Pnm[n-1][m]-\
35                               rootn[d-e]*Pnm[n-2][m])/rootn[d]
36
37  #     dP(n,m) = Pnm(m,n+1) is the derivative of P(n,m) vrt. theta
38      Pnm[0][2] = -Pnm[1][1]
39      Pnm[1][2] =  Pnm[1][0]
40      for n in np.arange(2, nmax+1):
41              l = n + 1
42              Pnm[0][l] = -np.sqrt(.5*(n*n+n))*Pnm[n][1]
43              Pnm[1][l] = .5*(np.sqrt(2.*(n*n+n))*Pnm[n][0]-\
44                              np.sqrt((n*n+n-2.))*Pnm[n][2])
45
46              for m in np.arange(2, n):
47                  Pnm[m][l] = .5*(np.sqrt((n+m)*(n-m+1.))*Pnm[n][m-1] -\
48                              np.sqrt((n+m+1.)*(n-m))*Pnm[n][m+1])
49
50              Pnm[n][l] = .5*np.sqrt(2.*n)*Pnm[n][n-1]
51
52      return Pnm
```

Listing 4.2: `get_Pm()` from Appendix F. Input of the function includes the maximum spherical harmonic degree `nmax`, the spherical harmonic order `order` and an array of co-latitudes `theta`.

```
1   import numpy as np
2   def get_Pm(nmax, order, theta):
3       """
4       Calculation of associated Legendre functions P(n,m) (Schmidt
            normalized)
5
6       Input: theta[:] co-latitude (in rad)
7              order spherical harmonic order
8              nmax  maximum spherical harmonic degree
9       Output: Pm    ndarray with Legendre functions
10
11      P(n,m) ==> Pnm(n,m) and dP(n,m) ==> Pnm(m,n+1)
12      """
13      if type(n) != int:
14          nmax=max(n)
15          if nmax==0:
```

```
16                nmax+=1
17        else:
18            nmax=n
19        costh = np.cos(theta)
20        sinth = np.sqrt(1-costh**2)
21
22        Pm = np.zeros((nmax+1,len(theta)))
23
24        rootn = np.sqrt(np.arange(0, 2*nmax**2+1))
25
26    #     Recursion
27        for m in np.arange(0, nmax):
28            if m==0:
29                Pmdiag=np.ones(len(theta))
30            elif m==1:
31                Pmdiag=sinth
32            Pm_tmp = rootn[m+m+1]*Pmdiag
33            if m == order:
34                Pm[m,:] = Pmdiag
35                Pm[m+1,:] = costh*Pm_tmp-
36                for L in np.arange(m+2, nmax+1):
37                    d = L*L - m*m
38                    e = L+L - 1
39                    Pm[L,:] = (e*costh*Pm[L-1,:]-rootn[d-e]*Pm[L-2,:])/
                          rootn[d]
40                break
41            if m > 0:
42                Pmdiag = sinth*Pm_tmp/rootn[m+m+2]
43
44        return Pm
```

4.2 Computing the Kernel Matrix in Regions of Rotational Symmetry

By now, it has been established that properly constructing a kernel matrix is crucial to produce reliable models. What has not been discussed is how this is implemented in practice, which will be introduced in this section. First a brief note on why regions of rotational symmetry are attractive followed by the implementation.

Recall that the region of interest can be any arbitrarily defined region. These kernel matrices are however computationally costly because of the dimensions of $(L+1)^2 \times (L+1)^2$ (Plattner and Simons, 2017). The eigenvector matrix, $\mathbf{G}_J$ from Equation (3.12) can be truncated with some arbitrary Slepian truncation parameter J, reducing the dimensions of the analysis going forward to $(L+1)^2 \times J$. This only applies for the eigenvector matrix, thus the kernel matrix must be computed to its full dimension. However, when considering a region of rotational symmetry, like a spherical cap, the kernel matrix can be re-ordered into block-diagonal form with block sizes of

$(L+1)\times(L+1)$, significantly reducing the complexity of the problem (Plattner and Simons, 2017).

The re-ordering of the kernel matrix $\mathbf{K}$ in the case of a spherical cap is done by first considering the internal-field gradient vector spherical harmonics $\boldsymbol{E}_{lm}$ of Equation (3.4), which can be described by a linear combination of radial and tangential spherical harmonics, $\boldsymbol{P}_{lm}$ and $\boldsymbol{B}_{lm}$, defined by Dahlen and Tromp (1998) as

$$\boldsymbol{P}_{lm}(\hat{\boldsymbol{r}}) = \hat{\boldsymbol{r}}Y_{lm}(\hat{\boldsymbol{r}}) \quad \text{and} \quad \boldsymbol{B}_{lm}(\hat{\boldsymbol{r}}) = \frac{\nabla_1 Y_{lm}}{\sqrt{l(l+1)}}, \quad \text{for} \quad 1 \leq l \leq L \ \text{ and } \ -l \leq m \leq l, \tag{4.1}$$

where l is a spherical harmonic degree, L is the maximum spherical harmonic degree and m is the spherical harmonic order. In the case of $l = m = 0$ it applies that $\boldsymbol{P}_{00}(\hat{\boldsymbol{r}}) = \hat{\boldsymbol{r}}$ (Plattner and Simons, 2017). Pointwise orthogonality between $\boldsymbol{P}_{lm}$ and $\boldsymbol{B}_{lm}$ means that the regional integral over products of $\boldsymbol{E}_{lm}$, required for constructing the kernel matrix, can be described with linear combinations of integral products of $\boldsymbol{P}_{lm}$ and $\boldsymbol{B}_{lm}$, using the same spherical harmonic degrees and orders as the $\boldsymbol{E}_{lm}$ (Plattner and Simons, 2017)

$$\int_R \boldsymbol{E}_{lm}\cdot\boldsymbol{E}_{l'm'}\,\mathrm{d}\Omega = \sqrt{\frac{(l+1)(l'+1)}{(2l+1)(2l'+1)}}\int_R \boldsymbol{P}_{lm}\cdot\boldsymbol{P}_{l'm'}\,\mathrm{d}\Omega + \sqrt{\frac{ll'}{(2l+1)(2l'+1)}}\int_R \boldsymbol{B}_{lm}\cdot\boldsymbol{B}_{l'm'}\,\mathrm{d}\Omega. \tag{4.2}$$

The integral products of $\boldsymbol{P}_{lm}$ and $\boldsymbol{B}_{lm}$ are derived in (Plattner and Simons, 2014) and defined as

$$\int_R \boldsymbol{P}_{lm}\cdot\boldsymbol{P}_{l'm'}\,\mathrm{d}\Omega = 2\pi\delta_{mm'}\int_0^\Theta X_{lm}X_{l'm'}\sin(\theta)\,\mathrm{d}\theta \tag{4.3}$$

$$\int_R \boldsymbol{B}_{lm}\cdot\boldsymbol{B}_{l'm'}\,\mathrm{d}\Omega = \frac{2\pi\delta_{mm'}\int_0^\theta [X'_{lm}X'_{l'm'} + m^2(\sin(\theta))^{-2}X_{lm}X_{l'm'}]\sin(\theta)\mathrm{d}\theta}{\sqrt{l(l+1)l'(l'+1)}}, \tag{4.4}$$

where Θ denotes half the opening angle of the spherical cap, X'_{lm} denotes the X_{lm} differentiated with respect to co-latitude θ and $\delta_{mm'}$ ensures the integral is zero when $m \neq m'$. Recall the kernel matrix is defined by $\mathbf{K} = \mathbf{A}(r_p)\left(\int_R \boldsymbol{\mathcal{E}}_L\cdot\boldsymbol{\mathcal{E}}_L^T\,\mathrm{d}\Omega\right)\mathbf{A}(r_p)^T$ with $\boldsymbol{\mathcal{E}}_L = (\boldsymbol{E}_{00},\ldots,\boldsymbol{E}_{lm},\ldots,\boldsymbol{E}_{LL})^T$ collecting all required $\boldsymbol{E}_{lm}$. This regional integral over products of $\boldsymbol{E}_{lm}$ have now been theoretically accounted for through radial and tangential spherical harmonics. The continuation operator $\mathbf{A}(r_p)$ acts as a normalization to the kernel matrix, since the $\boldsymbol{E}_{lm}$ are defined on the unity sphere. Applying e.g. the Earth's mean spherical radius means the kernel matrix is defined at that radius.

I approach the implementation of the integral over products numerically using a Gauss-Legendre quadrature, introduced in Section 8.5 in Hildebrand (1987). I compute absicasses, weights and number of points used in the integration using the function `glquadinit()` from Appendix D. The number of points used in the integration is dependent on spherical harmonic degree and is computed by $N = (l+1)/2$ where N is the number of integration points and l is a spherical harmonic degree. The absicasses computed from the Gauss-Legendre quadrature are used as input co-latitudes when computing Legendre functions for the construction of the kernel matrix, which can be seen in Appendix C.1.

Due to the code length I refer to Appendix C.1 for an overview of how the radial and tangential spherical harmonics are implemented and how they are combined to construct kernel matrices.

After computing the kernel matrix, it must be rotated to fit the center coordinates of the spherical cap. The placement of the kernel matrix is initially constructed at the North Pole with the opening angle of the spherical cap defining its size. Rotating the spherical cap to an arbitrary location is performed on the eigenvector matrix $\mathbf{G}_J$ from Equation (3.12). Rotating to an arbitrary location is achieved by applying a rotation matrix based on Section C.8 in Dahlen and Tromp (1998), where appropriate rotation angles are used. The implementations for kernel matrix rotation are found in Appendix E.

4.3 Computing the Design Matrix

Constructing the design matrix requires an eigenvector matrix. This is obtained from eigenvector decomposition of the kernel matrix, where the computation of the kernel matrix is described in the previous section. Implementing the design matrix requires computation of the $\boldsymbol{E}_{lm}$. This is achieved by computing the Legendre functions alongside their derivatives with respect to co-latitude individually as described in Section 4.1. Next the X_{lm} (Equation (3.6)) is computed allowing construction of the $\boldsymbol{E}_{lm}$.

The design matrix is constructed in a `for`-loop, looping over all spherical harmonic degrees, up to and including L. This is seen in the `for` loop beginning in line 50 of Appendix C.2. Since the kernel matrix is defined at the Earth's surface, the design matrix must be upwards continued which is performed by applying a radial factor, computed within the aforementioned `for`-loop.

From the design matrix model parameters are obtained in the inverse problem.

4.4 The Inverse Problem

After the Python toolbox has been provided with both kernel and design matrix, the next step is inversion. It is capable of performing three inversions; LS, L_1-norm and L_2-norm regularization, where the two latter requires a computed regularization matrix. In the following sections, the implementation of the two latter inversions and the regularization matrix will be described.

4.4.1 The Robust Solution

Robust estimation is an iterative process. The implementation will be as such. I use a `while` loop, updating data weights with each iteration until a convergence criteria is met. Recall the convergence criteria is a change in model parameter two-norm of less than 0.01 percent. The first iteration will, if no prior weights are given, be the LS solution. I compute initial weights based the data misfit of an LCS-1 prediction and satellite data over a region of interest. This implementation is presented in Listing 4.3. The satellite gradient vector observations are provided, and the LCS-1 full model predictions are computed. Using the data uncertainties from the data set, the residuals are computed and huber weights of Equation (3.30) are determined.

Listing 4.3: Initial weights used in the robust estimation. `dsat` is a vector satellite gradient observations, `rsat` is a vector of satellite altitudes, `phi` and `theta` are satellite longitudes and co-latitudes, respectively, and `sigma_rtp` are data uncertainties for the three vector components.

```
1 def initweight(dsat,rsat,phi,theta,sigma_rtp):
2     from lib.synth.synthfunctions import gausscoef,synth_gradient
3     n = len(phi[0])
```

```
4       c = 1.5
5       title = "matr/LCS1dat_r%i_L%i_n%i_cen_%i_%i.npy"%\
6                                                   (rcap,Lsynth,n,cen[0],cen
                                                        [1])
7       try:
8           dLCS1 = np.load(title)
9       except FileNotFoundError:
10          mLCS1 = gausscoef(185,0)
11          dLCS1 = synth_gradient(rsat,phi,theta,185,mLCS1)
12          np.save(title,dLCS1)
13
14      resi = (dsat-dLCS1)/sigma_rtp
15      weights_out = np.minimum(abs(c/resi),1)
16      return weights_out
```

The conceptual implementation of the IRLS method is introduced in Listing 4.4. Using an initial set of J model parameters set to one, an initial convergence criteria of `conver = 1`, a breakpoint constant `c = 1.5` and the design matrix `Gup`, the computation of weights begin. The robust solution alone is not used because this project aims at high resolution models which will require regularization. An issue with the `while` loop implementation is that highly unstable models will change a lot from iteration to iteration, meaning the convergence criteria is never met. This issue is addressed by gradually decreasing the amount of Slepian model parameters which leads to convergence.

Listing 4.4: IRLS conceptual implementation. `Wh` is obtained using the code from Listing 4.3, `Cd` are data variances, `conver` is the convergence criteria and `sigma_rtp` are data uncertainties.

```
1  import numpy as np
2
3  Wh = initweights(dsat,rsat,phi,theta,sigma_rtp)
4  Cd = sigma_rtp**2
5  conver = 1
6  c = 1.5
7  model = np.ones(J)
8  while conver <1e-2:
9       m_prev = model
10      model = np.linalg.solve(Gup*(Wh/Cd)@Gup.T, Gup*(Wh/Cd)@dsat)
11      e = (dsat - Gup.T@model)/sigma_rtp
12      Wh = np.minimum(c/abs(e),1)
13
14      conver = np.linalg.norm(m_prev-model)/np.linalg.norm(model)
```

4.4.2 Constructing the Regularization Matrix

A key aspect in regularization is the regularization matrix. This acts as *a priori* knowledge to the model parameters and serves to minimize the L_p-norm of the model parameters. The Python toolbox implementation includes evaluating a regular grid on the planetary surface. In the case of the Earth, its mean spherical radius is used. The regular grid used in this work is an icosahedral

grid. I only minimize the radial component of the magnetic field (Olsen et al., 2017), and thus only this part is provided as output in Listing 4.5.

Listing 4.5: Regularization matrix implementation in the Python toolbox. The function requires as input the $\mathbf{G}_J$ matrix and a refinement degree to determine amount of points are in the icosahedral grid.

```
def regmatrix(kerneleigvec,dlen):
    Lrtitle = "matr/Lr-r%i-L%i-J%i-refdeg%i-cen-%i-%i.npy"%\
                                                (rcap,Lmax,J,ref_deg,cen[0],cen
                                                    [1])
    try:
        Lr = np.load(Lrtitle)
        print("Regularization matrix file found.",end="\n")
    except FileNotFoundError:
        print("No regularization matrix file found. Computing one.",end
            ="\n")
        try:
            regx,regy = np.load("matr/icosahedral_ref%i.npy"%(ref_deg))
            print("Located global icosahedral grid of refinement degree
                : %i"\
                    %(ref_deg),end="\n")
        except FileNotFoundError:
            regx,regy,n_points = regulargrid(ref_deg)
            np.save("matr/icosahedral_ref%i.npy"%(ref_deg),(regx,regy))

        trim = regulargrid_cropped(regx,regy,rcap,cen,True)
        regX,regY = regx[trim],regy[trim]
        regZ = np.ones(regX.shape)*a
        L = raft.designeval(kerneleigvec[:,:J],regX*rad,regY*rad,regZ,a
            ,1)
        Lr = L[:,:dlen]
        np.save(Lrtitle,Lr)
        print("LtL regularization matrix computed.",end="\n")
    return Lr
```

4.4.3 L_2-norm Model Regularization

The more simple version of L_p-norm model regularization is the L_2-norm. In a robust scheme, the varying parameter is the data weights only. As discussed in Section 4.4.1, I have implemented a method of obtaining data weights based on LCS-1 predictions. Furthermore, the L_2-norm regularization is implemented with an option to choose these initial weights as the chosen data weights without further iterative updating. Equation (3.34) is solved only varying the regularization parameter α^2 with each iteration. Listing 4.6 presents the implementation for the L_2-norm model regularization where also model and residual norms of Equations (3.36) and (3.37) are computed for the L-curve. Furthermore it includes the option to produce prediction maps for the range of α^2 values. Residual norms, model norms and weights are saved with an appropriate title in the `matr` sub-folder of the toolbox.

Listing 4.6: L_2-norm model regularization scheme. `dsat` are satellite observations, `dataerror` are data uncertainties both obtained from the data set. `designsat` is the design matrix, `designz` is a design matrix at Earth's mean spherical surface to produce prediction maps. `RtR` is the product of the regularization matrix transposed with itself. `Cd` are data variances, **Wh** are data weights computed from Listing 4.3 and `G` is the eigenvector matrix.

```
def L2Lcurve(dsat,dataerror,designsat,designz,RtR,Wh,Cd,G):
    try:
        resi_norm = np.load(resnormtitle)
        model_norm = np.load(modnormtitle)
    except FileNotFoundError:
        resi_norm = np.zeros(n)
        model_norm = np.zeros(n)

        for i in range(n): # Loop over number of alpha squared
            if regrobust: # Compute data weights
                m_alpha = np.ones(len(designsat))
                conver,j = 1,0
                while conver > 1e-2:
                    j+=1
                    m_prev = m_alpha.copy()
                    m_alpha = np.linalg.solve((designsat*(Wh/Cd))\
                                @designsat.T+alpha_sq_all[i]*RtR,\
                                (designsat*(Wh/Cd))@dsat)
                    e = (dsat-designsat.T@m_alpha)/dataerror
                    Wh = np.minimum(c/abs(e),1)
                    conver = np.linalg.norm(m_prev-m_alpha)/\
                                np.linalg.norm(m_alpha)

            else: # Fixed data weights
                m_alpha = np.linalg.solve((designsat*(Wh/Cd))\
                            @designsat.T+alpha_sq_all[i]*RtR,\
                            (designsat*(Wh/Cd))@dsat)
            dmod = designsat.T@m_alpha
            model_norm[i] = m_alpha.T@RtR@m_alpha
            resi_norm[i] = ((dsat-dmod)*(Wh/Cd))@(dsat-dmod)
            if mapmalp:
                np.save(malpL2sav,m_alpha)
                dmodz = designz.T@m_alpha
                malphamapsL2(dmodz,m_alpha,alpha_sq_all[i],G,i)
    np.save(resnormtitle,resi_norm)
    np.save(modnormtitle,model_norm)
    np.save(alphasqtitle,alpha_sq_all)

    return resi_norm,model_norm
```

This method ensures that model parameters, model prediction maps and Mauersberger-Lowes power spectra are saved in each iteration if wanted, which provides useful diagnostic tools alongside the L-curve for choosing the regularization parameter α^2.

4.4.4 L_1-norm Model Regularization

The more complex type of L_p-norm model regularization is the L_1-norm regularization. This implementation is built around the robust scheme as for the L_2-norm model regularization. With the introduction of model weights, the varying parameters are now data and model weights. They must meet the convergence criteria of a change in model parameter two-norm of less than 0.01 percent. This implementation also features the ability to choose data weights based on an LCS-1 prediction, which can be chosen as the final weights or updated iteratively. The implementation resembles that of the L_2-norm much with the exception of the newly incorporated data weights, and Listing 4.7 presents the code.

Listing 4.7: L_1-norm model regularization scheme. `dsat` are satellite observations, `dataerror` are data uncertainties both obtained from the data set. `designsat` is the design matrix, `designz` is a design matrix at Earth's mean spherical surface to produce prediction maps. `R` is the regularization matrix. `Cd` are data variances, **Wh** are data weights computed from Listing 4.3 and `G` is the eigenvector matrix.

```
1  def L1Lcurve(dsat,dataerror,designsat,designz,R,Wh,Cd,G):
2      try:
3          resi_norm = np.load(resnormtitle)
4          model_norm = np.load(modnormtitle)
5      except FileNotFoundError:
6          resi_norm = np.zeros(n)
7          model_norm = np.zeros(n)
8
9          for i in range(n):
10             if regrobust: # compute data weights and model weights
11                 m_alpha = np.ones(len(designsat))
12                 Wm = np.ones(len(R.T))
13                 conver,j = 1,0
14                 while conver > 1e-2:
15                     j+=1
16                     m_prev = m_alpha.copy()
17                     m_alpha = np.linalg.solve((designsat*(Wh/Cd))\
18                                 @designsat.T+alpha_sq_all[i]*((R*Wm)@R.
                                     T),\
19                                 (designsat*(Wh/Cd))@dsat)
20                     e = (dsat-designsat.T@m_alpha)/dataerror
21                     Wh = np.minimum(c/abs(e),1)
22                     Wm = 1/np.sqrt((R.T@m_alpha)**2+epsilon**2)
23                     conver = np.linalg.norm(m_prev-m_alpha)/\
24                                 np.linalg.norm(m_alpha)
25             else: # Fixed data weights, compute model weights
26                 m_alpha = np.ones(len(designsat))
27                 Wm = np.ones(len(R.T))
28                 conver,j = 1,0
29                 while conver > 1e-2:
30                     j+=1
31                     m_prev = m_alpha.copy()
32                     m_alpha = np.linalg.solve((designsat*(Wh/Cd))\
```

```
                                        @designsat.T+alpha_sq_all[i]*((R*Wm)@R.
                                            T),\
                                        (designsat*(Wh/Cd))@dsat)
                    Wm = 1/np.sqrt((R.T@m_alpha)**2+epsilon**2)
                    conver = np.linalg.norm(m_prev-m_alpha)/\
                                        np.linalg.norm(m_alpha)
            dmod = designsat.T@m_alpha
            model_norm[i] = m_alpha.T@((Lr*Wm)@R.T)@m_alpha
            resi_norm[i] = ((dsat-dmod)*(Wh/Cd))@(dsat-dmod)
            if mapmalp:
                np.save(malpL1sav,m_alpha)
                dmodz = designz.T@m_alpha
                malphamapsL1(dmodz,m_alpha,alpha_sq_all[i],G,i)
    np.save(resnormtitle,resi_norm)
    np.save(modnormtitle,model_norm)
    np.save(alphasqtitle,alpha_sq_all)

    return resi_norm,model_norm
```

As the L_2-norm, the L_1-norm model regularization provides the same possibilities of producing diagnostic tools such as Mauersberger-Lowes power spectra and model prediction maps.

4.5 Data Sorting

Sorting the data regionally must be carried out with a spherical setting in mind. To correctly sort data only within a desired region, I compute the cosine of the solid angles with respect to the center coordinates of the spherical cap using

$$\cos(\mu_{ik}) = \cos(\theta_i)\cos(\theta_k) + \sin(\theta_i)\sin(\theta_k)\cos(\phi_i - \phi_k), \tag{4.5}$$

with subscript k denoting the spherical cap and i denoting data. θ is co-latitude and ϕ is longitude (Kother et al., 2015). The solid angles are compared the the opening angle of the spherical cap, and only data with solid angles smaller than or equal to the opening angle are selected. This is implemented using two lines of code shown in Listing 4.8, where I obtain the actual solid angles by taking the `arccos` of the right-hand side of Equation (4.5).

Listing 4.8: Solid angle computation for data trimming. `theta` and `phi` are data co-latitudes and longitudes and `cen` is a tuple of spherical cap center coordinates. First coordinate is the longitude, and the second is the co-latitude. `radius` is the opening angle of the spherical cap.

```
sangle = arccos(cos(theta)*cos(cen[1])+\
         sin(theta)*sin(cen[1])*cos(phi-cen[0]))
trim = np.less_equal(sangle,radius)
```

This concludes the implementation of key aspects of the Python toolbox. The following chapter will introduce two synthetic test cases constructed as a proof of concept for the Python toolbox.

Chapter 5

Synthetic Test Cases

Prior to this work, the Slepian approach to regional modelling has only been used with either non-gradient satellite data or synthetic data. Furthermore, previous models generated are no higher than spherical harmonic degree $L = 130$ (e.g. Plattner and Simons (2015), Plattner and Simons (2017)). Before deriving models using real gradient satellite data, I tested an early iteration of the Python toolbox with synthetic cases as presented in this chapter.

The main goals of the test cases are:

- Proof of concept test.
  - Test whether the Python toolbox works with synthetic gradient data.
- High spherical harmonic degree test.
  - Python toolbox validation for scientific purposes.
- L_1-norm and L_2-norm model regularized solutions.
  - Show that regularization works with the Slepian approach.

I constructed two test cases based on the above goals. Common to both cases is that synthetic gradient data is acquired using the LCS-1 model parameters (Olsen et al., 2017). This lithospheric magnetic field model goes up to spherical harmonic degree $L = 185$. Knowing the model parameters, the synthetic gradient data is given by the forward problem

$$\mathrm{d}_{grad}^{synth} = \mathrm{d}_1^{synth} - \mathrm{d}_2^{synth} = (\mathbf{G}_1 - \mathbf{G}_2)\,\mathrm{m}_{LCS-1}, \tag{5.1}$$

where $\mathrm{d}_{grad}^{synth}$ is the synthetic gradient data, d_i^{synth} are synthetic data vectors that will construct either across-track or along-track gradient data, $\mathbf{G}_i$ are design matrices and m_{LCS-1} are model parameters from the LCS-1 model. For both synthetic test cases I use the full LCS-1 model. I obtain the design matrices $\mathbf{G}_i$ using the `design_SHA()` function from `GMT_tools` toolbox (see Appendix F).

The test cases are located at the site of the Bangui Anomaly and will consist of a spherical cap with opening angle of 15°. I extract location, altitude and data uncertainty information from the real data set to best imitate a real case scenario (see Sections 2.2.1 and 4.5 for an introduction to data selection and sorting). I include information with a 15° radius from the center coordinates $(lon, lat) = (20^\circ\ E, 4^\circ\ N)$. The locations alongside altitudes are used as input to `design_SHA()` to

obtain $\mathbf{G}_i$ of Equation (5.1). Now knowing the model parameters and $\mathbf{G}_i$ I compute the synthetic data.

I add noise to the synthetic data of the second test case based on known data uncertainties available in the data set. The implementation uses the `numpy.random.normal()` function with zero mean and known standard deviations for each datum.

5.1 Test Case One

This first test is constructed as a proof of concept for the Python toolbox, i.e. to investigate whether or not the Python toolbox works as intended. The test is of spherical harmonic degree 185 and utilizes noise-free synthetic data.

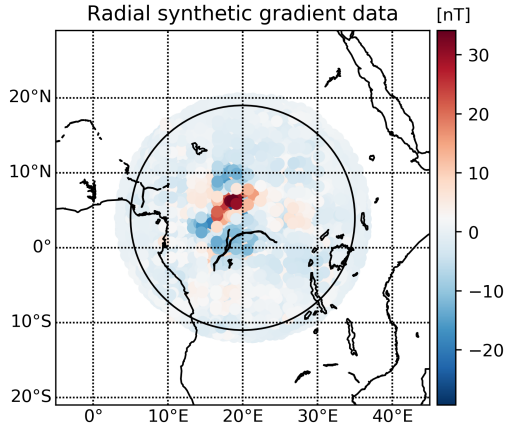
I construct synthetic data at satellite altitude using actual satellite positions

$$\mathrm{d}_{grad}^{synth,sat} = \mathrm{d}_1^{synth,sat} - \mathrm{d}_2^{synth,sat} = \left(\mathbf{G}_1^{sat} - \mathbf{G}_2^{sat}\right) \mathrm{m}_{L=185,LCS-1}, \tag{5.2}$$

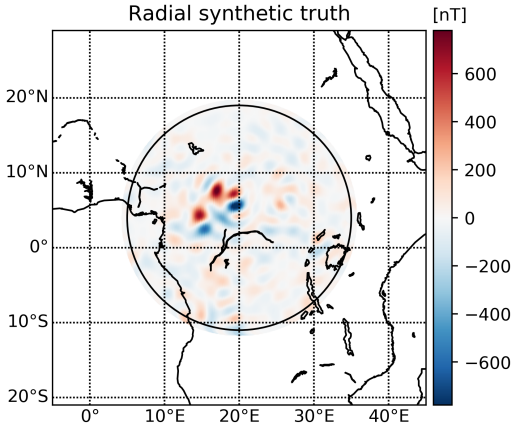
where the *sat* superscript means actual satellite positions and altitudes. This data will be used in the inversion to obtain Least Squares Slepian model parameters. I construct a synthetic truth at the Earth's surface using

$$\mathrm{d}^{synth,truth} = \mathbf{G}^{truth} \mathrm{m}_{L=185,LCS-1}, \tag{5.3}$$

where $\mathrm{d}^{synth,truth}$ is the LCS-1 model evaluated on a 0.1° spacing coordinate grid at the Earth's mean spherical surface, $a = 6371.2$ km. Equations (5.2) and (5.3) provide the data presented in Figures 5.1a and 5.1b, respectively. In summary, Figure 5.1a contains the synthetic satellite data which must be used to model the synthetic truth shown in Figure 5.1b.



(a) Radial component of $\mathrm{d}_{grad}^{synth,sat}$ projected onto one set of coordinates. The black circle indicates 15° radius, the cut-off radius for selecting data.



(b) Radial component of the synthetic truth at Earth's surface. The black circle indicates 15° radius, the cut-off radius for selecting data.

Figure 5.1: Radial components of the synthetic data with $L = 185$.

5.1.1 Regional Model of Test Case One

I construct a localized kernel matrix of the spherical cap with an opening angle of 15° over the Bangui anomaly using $L = 185$ and $J_{max} = 2000$. I ensure to construct a kernel matrix with more Slepian functions than I anticipate will be necessary, as redundant functions can be negated by truncating to include only $J < J_{max}$ functions.

Investigating eigenvalues of the kernel matrix as a function of number of Slepian functions, I approximate the best Slepian truncation parameter, J. Figure 5.2 presents this plot, where eigenvalues have diminished greatly within the J_{max} included Slepian functions. It is thus within reason to further truncate to include only $J = 1850$ Slepian functions. I use this plot as a diagnostic tool for estimation of when eigenvalues become too small, and thus selection of the truncation parameter J is obtained from this plot.

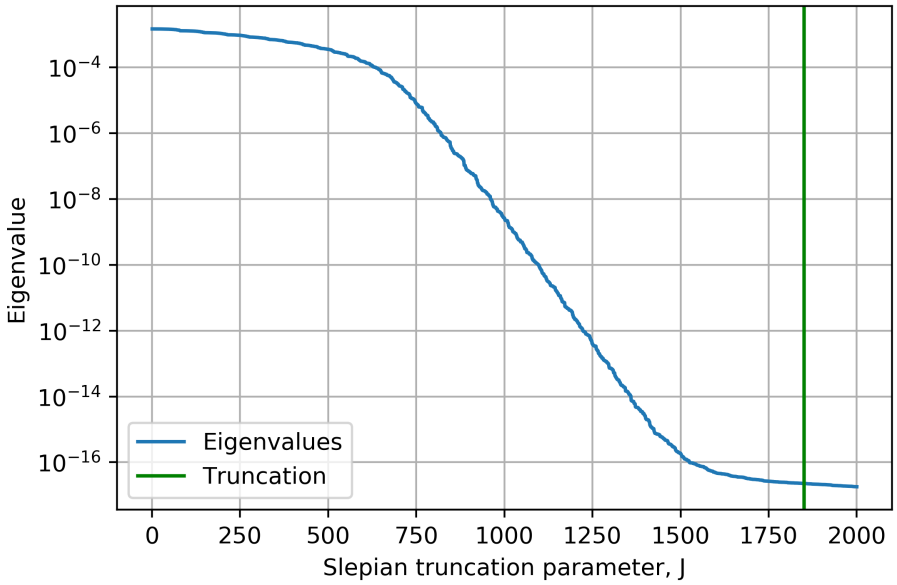
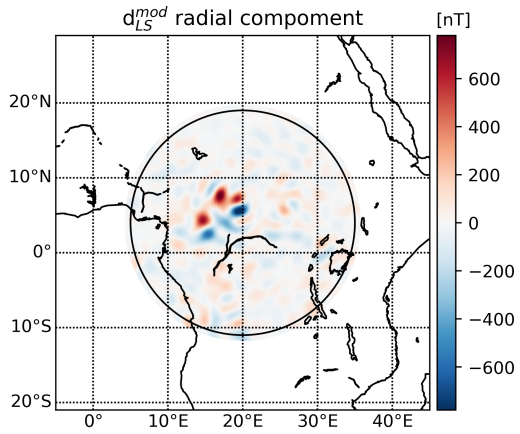


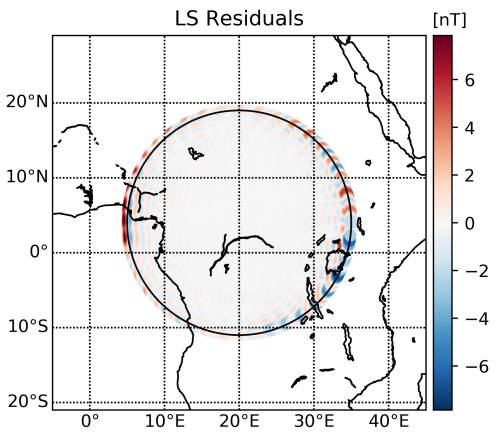
Figure 5.2: Localized kernel matrix eigenvalues as a function of amount of Slepian functions with the estimation of a good Slepian truncation at $J = 1850$ indicated by the green vertical line.

I construct and obtain localized design matrices evaluated at actual satellite altitudes and solve the inverse problem stated in Equation (3.28) to obtain the Least Squares Slepian model parameters. The resulting map is presented in Figures 5.3a and 5.3b.

The resulting model is in high agreement with the synthetic truth varying only near the edges of the spherical cap by a maximum of ±6 nT. A good result is expected due to noiseless data, however I did not expect a simple LS solution to perform this well at high spherical harmonic degrees. This could indicate issues with this early iteration of the Python toolbox. I urge caution using this synthetic test case for validation of the Python toolbox.



(a) Radial component of the modelled Least Squares data.



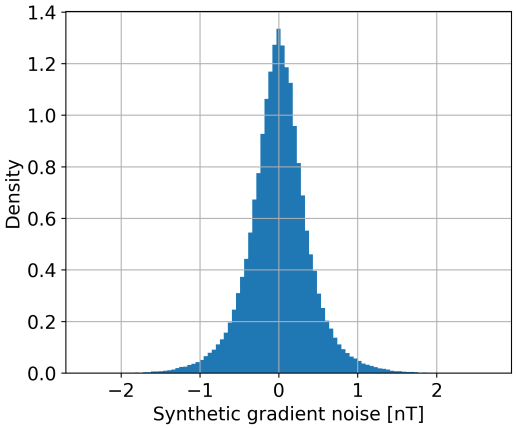
(b) Radial component of the difference between d_{LS}^{mod} and synthetic truth at Earth's surface.

Figure 5.3: Least Squares solution to the $L = 185$ synthetic case without added noise.

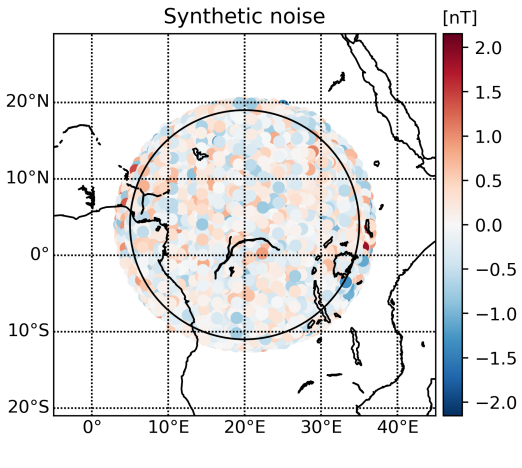
5.2 Test Case Two

The second test case is a high spherical harmonic degree performance case that will use $L_{max} = 200$. I now distinguish between L_{max} that will describe the spherical harmonic degree of the model and L_{synth} that describes the spherical harmonic degree of the synthetic data.

Synthetic gradient data and a synthetic truth are the same as Figures 5.1 and 5.1b, but noise has been added to the synthetic satellite data. In Figures 5.4a and 5.4b the noise distribution added to the data is shown.



(a) Histogram of noise from a zero-mean normal distribution with known standard deviations.



(b) Scatter plot of the synthetic noise. The black circle indicates the 15° radius of the spherical cap.

Figure 5.4: Synthetic noise histogram and scatter plot.

5.2.1 Regional Model of Test Case Two

I construct a regional model for the $L = 200$ case using $J_{max} = 3,000$. I investigate the localized kernel matrix eigenvalues as a function of amount of Slepian functions and choose $J = 1,500$. Figure 5.5 presents eigenvalues as a function of Slepian functions.

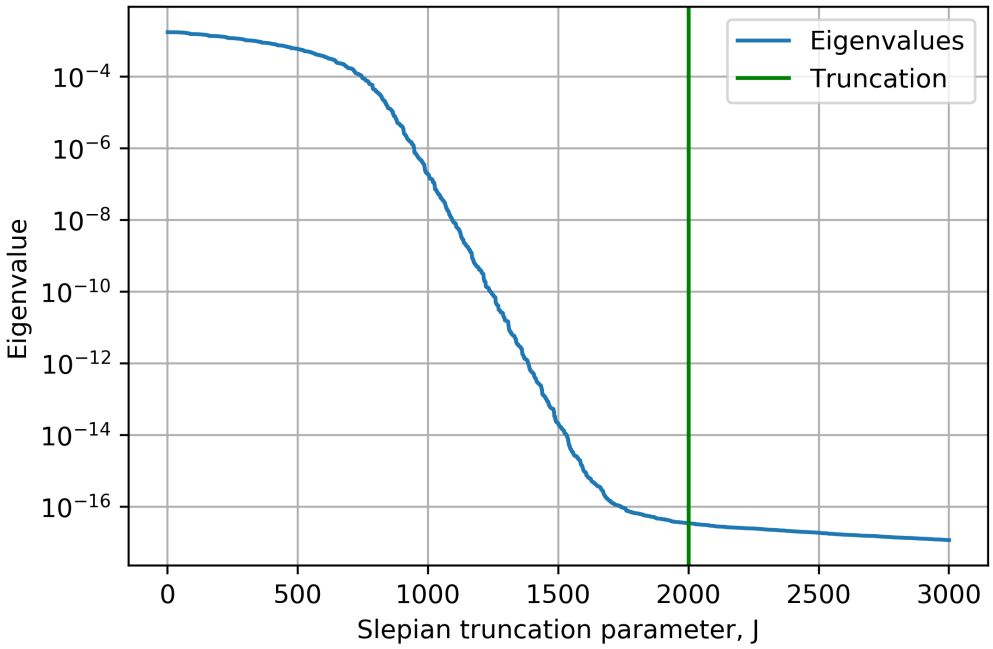


Figure 5.5: Localized kernel matrix eigenvalues plotted against amount of Slepian Functions for the $L = 200$ case.

I investigate only the L_2-norm model regularization for this test because this is the more simple type of regularization.

Because this test is performed with an early iteration of the Python toolbox, the icosahedral grid was not yet implemented. Therefore in constructing the regularization matrix I do not use an equal area grid. Because the test site is close to the Equator, the missing equal area effects of the icosahedral grid will not impact the results greatly. The grid is defined as a spherical cap with a radius of $r = rcap + 10°$, where $rcap$ is the opening angle of the spherical cap to be investigated, such that the entirety of the area is more than covered.

Investigating the L_2-norm model regularization I iterate over varying values of the regularization parameter α^2. For each iteration I solve the inverse problem stated in Equation (3.34), and compute model and residual norms according to Equations (3.36) and (3.37). I plot the L-curve resulting from the model and residual norms and use the discrepancy principle to estimate the best value for α^2. By applying the discrepancy principle, I normalize the data misfit with respect to the number of data (see e.g. Equation (3.38)). I choose the value of α^2 that produces a data misfit of 1. Investigating Figure 5.6 it becomes clear that the regularization parameter that produces the best data misfit with respect to the discrepancy principle is $\alpha^2 = 7 \cdot 10^{-5}$.

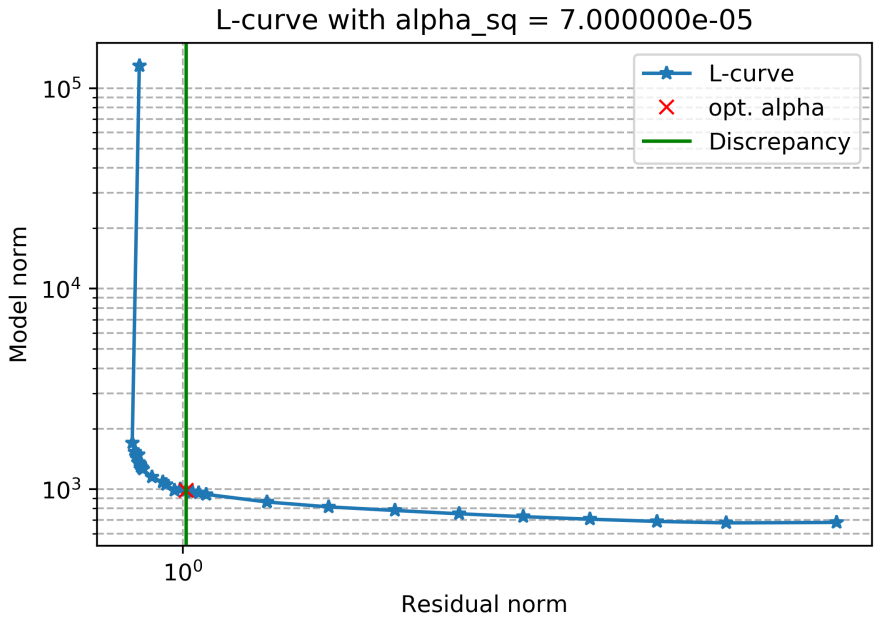
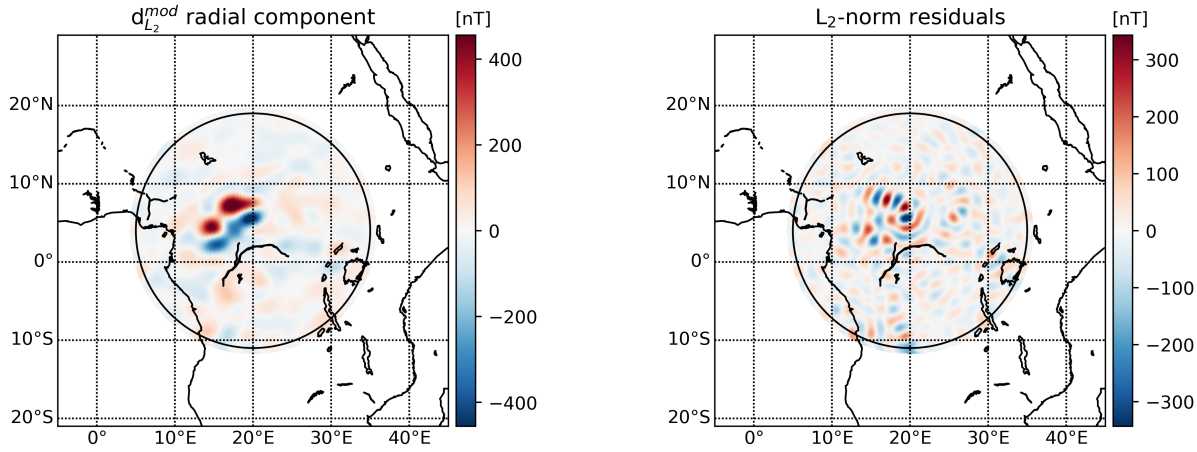


Figure 5.6: L-curve for the $L_{max} = 200$ case. X-axis is computed according to Equation (3.37) and y-axis according to Equation (3.36) and both have been normalized with respect to the discrepancy principle (see Equation (3.38)).

I re-compute the problem stated in Equation (3.34), this time applying the chosen value of α^2 and I obtain the model prediction and associated difference from the synthetic truth, presented in Figure 5.7.



(a) Radial component of the L_2-norm regularized prediction.

(b) Difference between radial components of the synthetic truth (Figure 5.1b) and the L_2-norm regularized prediction.

Figure 5.7: L_2-norm regularized solution for the $L_{max} = 200$ case. Black circles indicate the radius of the spherical cap (15°).

The L_2-norm regularized solution produced for the $L_{200} = 200$ case is indeed as one would expect. The biggest difference is observed over the Bangui anomaly, while the entirety of the field has been dampened. This test successfully achieves a reasonable L_2-norm regularized model over the Bangui anomaly at high spherical harmonic degrees with noised synthetic data.

5.3 Synthetic Test Case Summary

Two synthetic tests were carried out to investigate the performance of the Slepian toolbox.

Test case one investigated whether or not the toolbox works as expected using gradient satellite data. It was found to successfully reproduce the synthetic truth of LCS-1, up to spherical harmonic degree $L = 185$. However due to this test case being carried out with an early iteration of the Python toolbox, I urge caution with the conclusion that a LS solution of high spherical harmonic degree using noise-free data produces reliable results.

Test case two investigates a high spherical harmonic degree case of $L_{max} = 200$ using noised synthetic data. I performed L_2-norm model regularization and by investigating an L-curve I determined the α^2 that best suits the discrepancy principle to be $\alpha^2 = 7 \cdot 10^{-5}$. This regularization produces decent results, with the overall amplitude of the model prediction dampened as one would expect.

Despite the test cases being carried out early in the process, I will leave it to the real results for concluding whether or not the Python toolbox has been successfully implemented. But test case two does suggest the toolbox works as intended with regularized solutions.

Chapter 6

Results

In this chapter the results obtained using the Python toolbox will be presented. Plots of model prediction maps for the regions of interest (described in Section 2.3), L-curves, power spectra and data misfit histograms are presented individually for each region followed by a table of model and data misfit statistics.

Data misfit, e, is defined as the difference between satellite observations, d, and model predictions at satellite altitude, d^{pred},

$$\mathrm{e} = \mathrm{d} - \mathrm{d}^{pred}, \tag{6.1}$$

and will be presented in histograms that shows the density of each residual bin. This density is obtained using the `hist()` function from the `Matplotlib` library[1] and the normalization is done such that the integral of the probability density function over the range is one.

The root-mean-square error, RMSE, will be computed for data misfit using

$$\mathrm{RMSE} = \sqrt{\frac{\sum_{i=1}^{N} \left(\mathrm{e}_i^2\right)}{N}}, \tag{6.2}$$

where N is the length of the the data misfit vector. The RMSE diagnostic is useful for determining whether or not the data has been sufficiently fitted. An RMSE close to zero means the data has been fitted well. Furthermore, model statistics will include the model norms derived from regularization

$$\|\mathrm{m}_{J,\alpha,L_2}\| = \mathrm{m}_{J,\alpha,L_2}^T \mathrm{R}^T \mathrm{R}\mathrm{m}_{J,\alpha,L_2} \tag{6.3}$$

$$\|\mathrm{m}_{J,\alpha,L_1}\| = \mathrm{m}_{J,\alpha,L_1}^T \mathrm{R}^T \mathrm{W}_m \mathrm{R}\mathrm{m}_{J,\alpha,L_1}, \tag{6.4}$$

which provides a diagnostic tool for determining whether or not models are sensible. Table 6.1 presents the settings used for all models produced in the thesis. These values can be inserted into the initialization script of the Python toolbox to produce similar models.

[1] https://matplotlib.org/api/_as_gen/matplotlib.pyplot.hist.html

Table 6.1: Analysis information used for the four regional models produced. The **spherical cap** category includes center coordinates for the spherical cap alongside the opening angle of the cap. The **analysis** category includes spherical harmonic degree (L) of the analysis and amount of Slepian functions included (J). The **inversion** category includes values used for the regularization parameter α^2 for both L_1-norm and L_2-norm regularization.

	Bangui	Australia	Walvis Ridge	Greenland
Spherical cap				
Longitude	18° E	133° E	0°	42° W
Latitude	4° N	28° S	30° S	72° N
Opening angle	15°	25°	20°	15°
Kernel				
L	200	200	200	200
J	1500	2500	1700	1250
Inversion				
L_1-norm α^2	3.70×10^{-4} nT^{-1}	7.3×10^{-4} nT^{-1}	1×10^{-3} nT^{-1}	2.8×10^{-4} nT^{-1}
L_2-norm α^2	8.0×10^{-6} nT^{-2}	2.8×10^{-5} nT^{-2}	2.8×10^{-5} nT^{-2}	1×10^{-5} nT^{-2}

6.1 Bangui Anomaly

The Bangui Anomaly is one of the largest known lithospheric anomaly on Earth. This region provides an excellent opportunity to benchmark test the Python toolbox over a strong anomaly at low latitudes.

The L-curves related to the choice of regularization parameter α^2 is presented in Figures 6.1 and 6.2. These have both been normalized with respect to the discrepancy principle. Clearly, the discrepancy principle can not be utilized for this area, seeing as the misfit norms are all larger than 1 nT. This may either have to do with the difficulty of assigning uncertainties to data, making them unreliable, or with the fact that the gradients over the anomaly are difficult to model, causing an increase in the misfit norm. The L_1-norm regularized model has a regularization parameter close to the knee of the L-curve, which suggests a good trade-off between minimizing model norm and misfit norm. This parameter is not chosen solely because it was near the knee, but also from visual inspection of prediction maps. The L_2-norm regularized model has a lower regularization parameter compared to the L_1-norm, thus minimizing the model norm less than the misfit norm. This value was chosen based on visual inspection of prediction maps since the knee of the L-curve did not produce sensible results.

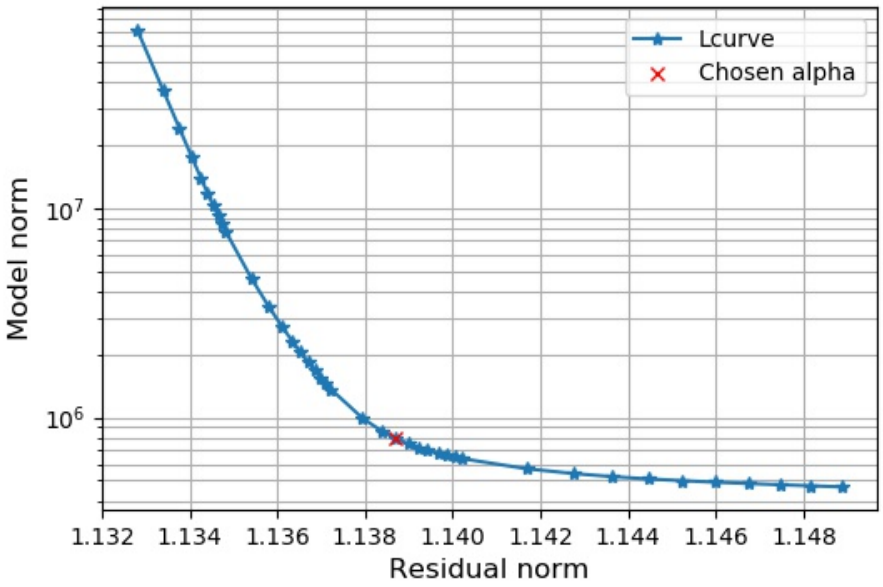


Figure 6.1: L_1-norm regularized model and residual norms for the Bangui Anomaly as computed by Equations (3.43) and (3.44), respectively, with residual norm normalized with respect to the discrepancy principle.

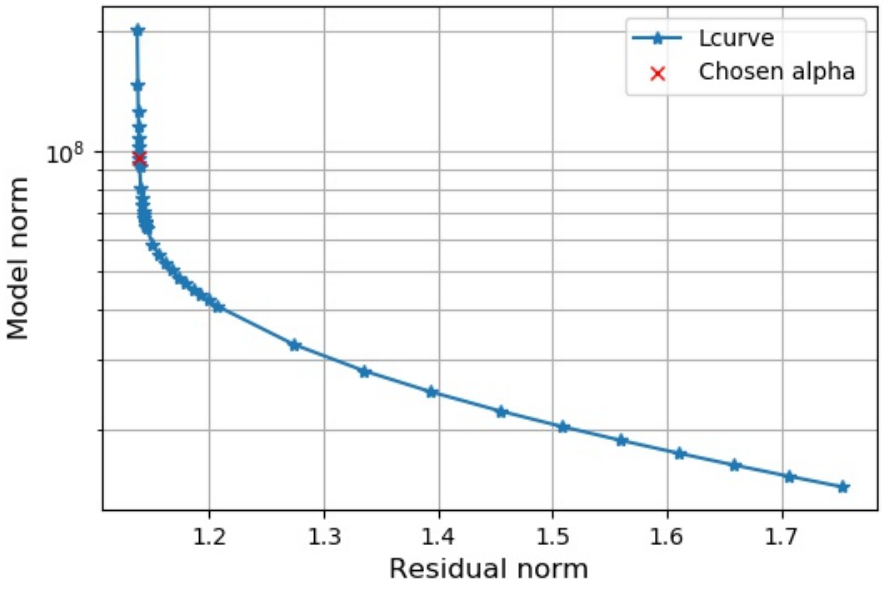


Figure 6.2: L_2-norm regularized model and residual norms for the Bangui Anomaly as computed by Equations (3.36) and (3.37), respectively, with residual norms normalized with respect to the discrepancy principle.

Figures 6.3 and 6.4 present the L_1-norm and L_2-norm regularized regional model prediction maps of the Bangui Anomaly, respectively. Both predictions captures the strong anomaly around coordinate $(lon, lat) = (18^\circ\ E,\ 5^\circ\ N)$ as well as smaller scale features in the surrounding area. Note that the colorscale shows larger span compared to the satellite data from Figure 2.1, since the satellite measurements are at satellite altitude, and the model predictions are at the Earth's surface. The prediction maps will be subject to discussion when compared to the LCS-1 model in Section 7.1.1.

The data misfit for the L_1-norm and L_2-norm regularized predictions are presented in Figures 6.5 and 6.6, and have the same mean values $\bar{e}_{L_1} = -0.009$ nT and $\bar{e}_{L_2} = -0.009$ nT. This serves as an indication that the data has been satisfactorily fitted in this region. The residual distributions of both model predictions are very similar and they appear to be Laplace-distributed. There are some outliers present, and it is believed that these are located over the anomaly itself, due to the difficulty of modelling rapid large changes. This hypothesis was investigated using a scatter plot of the radial component data misfits shown in Figure 6.7. From this it is evident that the residuals have a tendency of bundling up in the area of the anomaly.

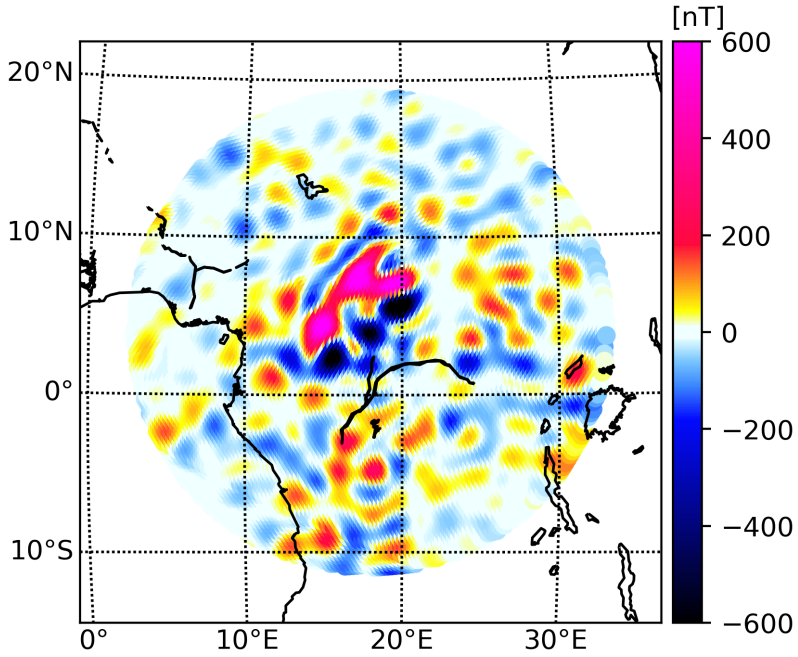


Figure 6.3: L_1-norm regional model prediction of the radial field at the Earth's mean spherical radius $a = 6371.2$ km for the Bangui Anomaly.

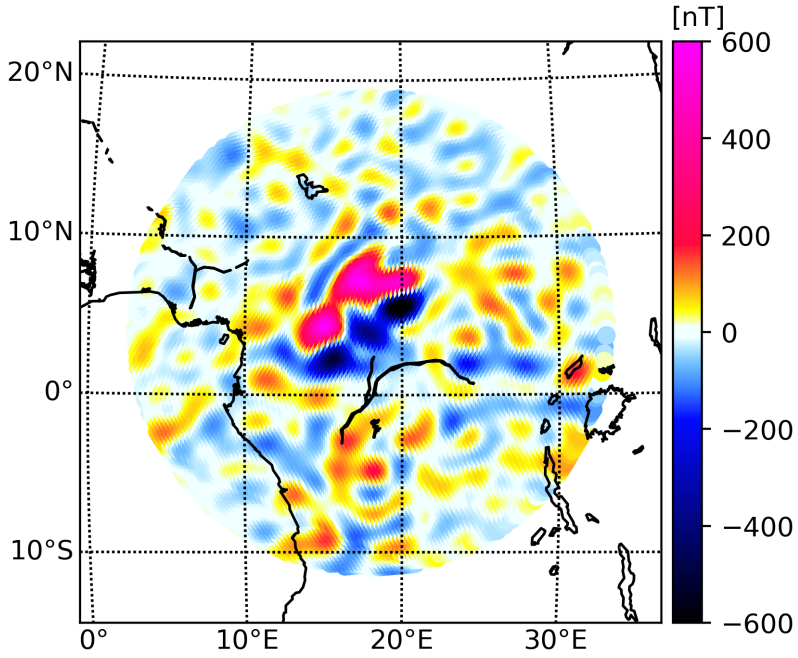


Figure 6.4: L_2-norm regional model prediction of the radial field at the Earth's mean spherical radius $a = 6371.2$ km for the Bangui Anomaly.

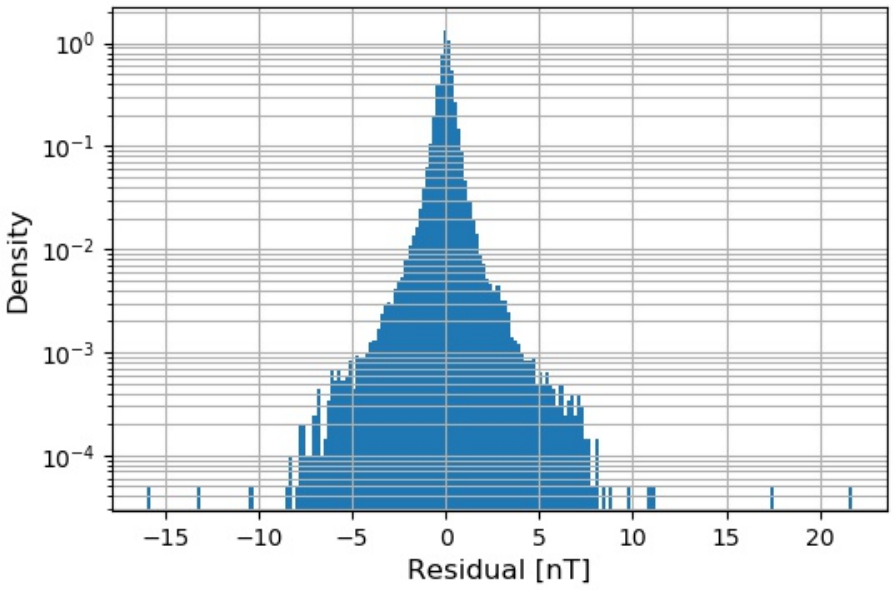


Figure 6.5: Histogram showing the data misfit distribution of the L_1-norm regularized prediction over the Bangui Anomaly.

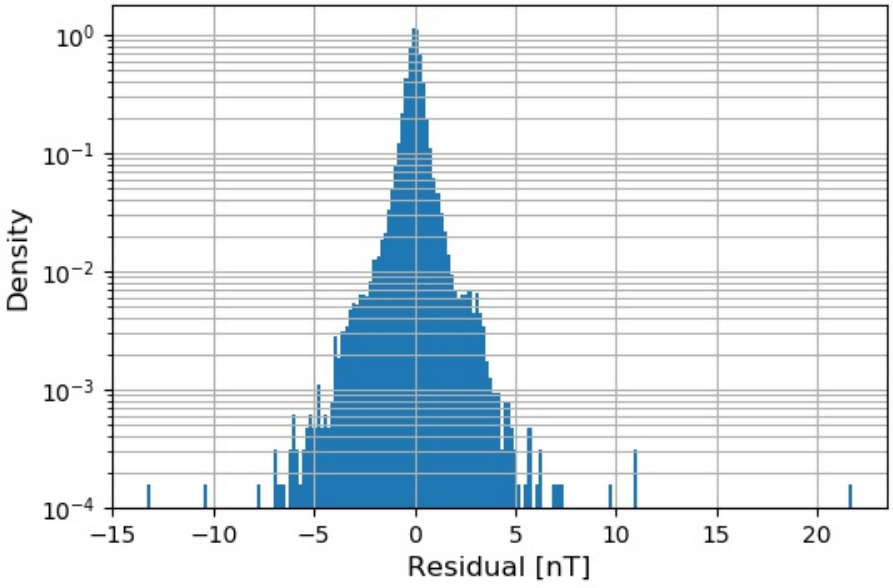


Figure 6.6: Histogram showing the data misfit distribution of the L_2-norm regularized prediction over the Bangui Anomaly.

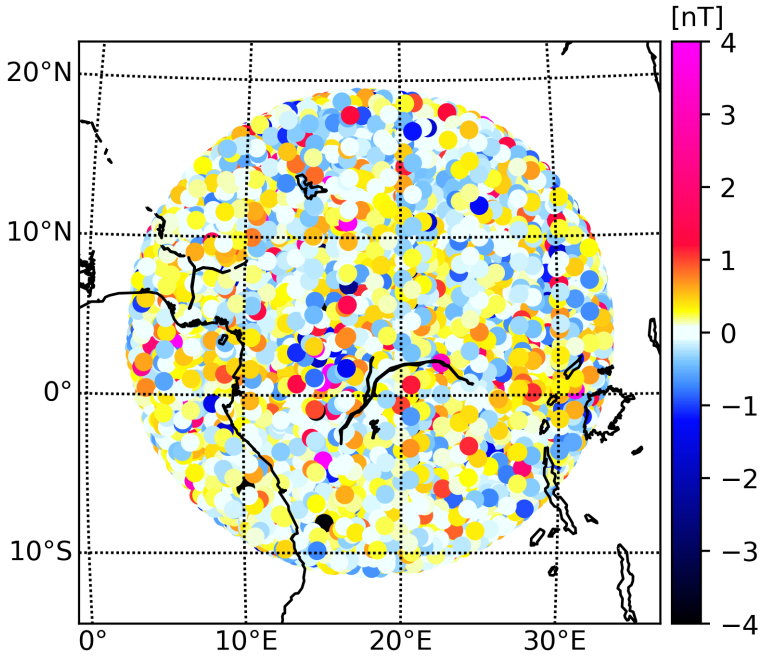


Figure 6.7: Scatter plot of the L_1-norm prediction radial component data misfit for the Bangui Anomaly.

Localized internal source spherical harmonic Gauss coefficients are computed using Equation (3.25) for both the L_1-norm and L_2-norm regularized Slepian model parameters. With the correction introduced in Equation (3.49) these Gauss coefficients are approximated globally instead of regionally allowing for the Mauersberger-Lowes power spectra to be computed for the two sets of model parameters using Equation (3.46). In Figure 6.8 the Mauersberger-Lowes power spectra is shown. These are used as a diagnostic tool for the order of magnitude of the power of the model parameters. The order of magnitude of the spectra is 10^1 nT^2, with the exception of high spherical harmonic degrees for the L_2-norm. This suggests that the models are reasonable. The two spectra shows the same tendencies and the spike seen at high spherical harmonic degree is likely due to numerical instability.

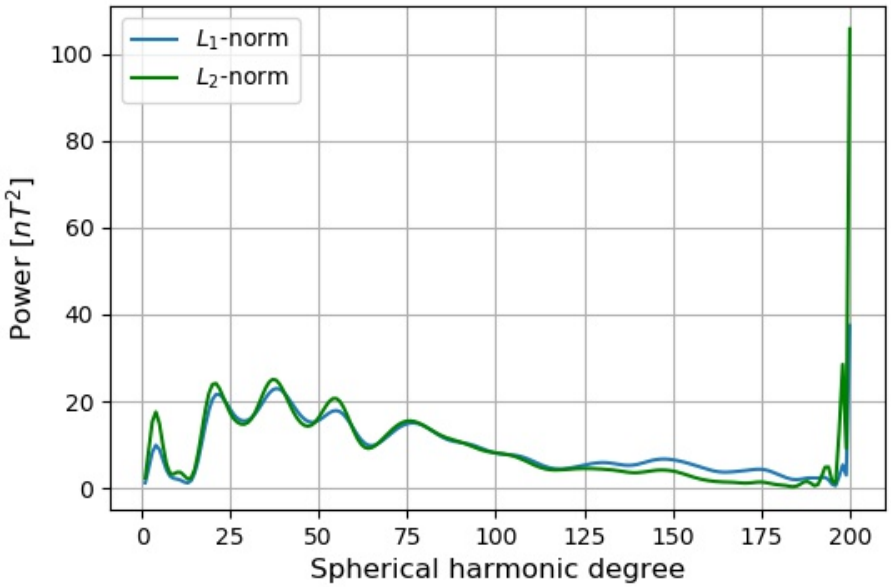


Figure 6.8: Mauersberger-Lowes power spectrum over the Bangui Anomaly.

6.2 Australia

Where the Bangui Anomaly allowed the Python toolbox to be tested at low latitudes, Australia provides an opportunity to test this regional modelling approach with its many, complex anomalies throughout the country at mid latitudes. Australia has been subject to extensive aeromagnetic surveys making it an interesting region for regional modelling.

The L-curves produced for both regularization methods are presented in Figures 6.9 and 6.10 and are both normalized with respect to the discrepancy principle. Both the L_1-norm and L_2-norm regularized models have regularization parameters reasonably close to the knee of the L-curves. Like for the Bangui Anomaly, the L_2-norm has a smaller regularization parameter than the L_1-norm. The discrepancy principle is not valid for this region either, seeing as the residual norms are almost twice as large as they should be. Again, this is no surprise given the difficulty of assigning data uncertainties.

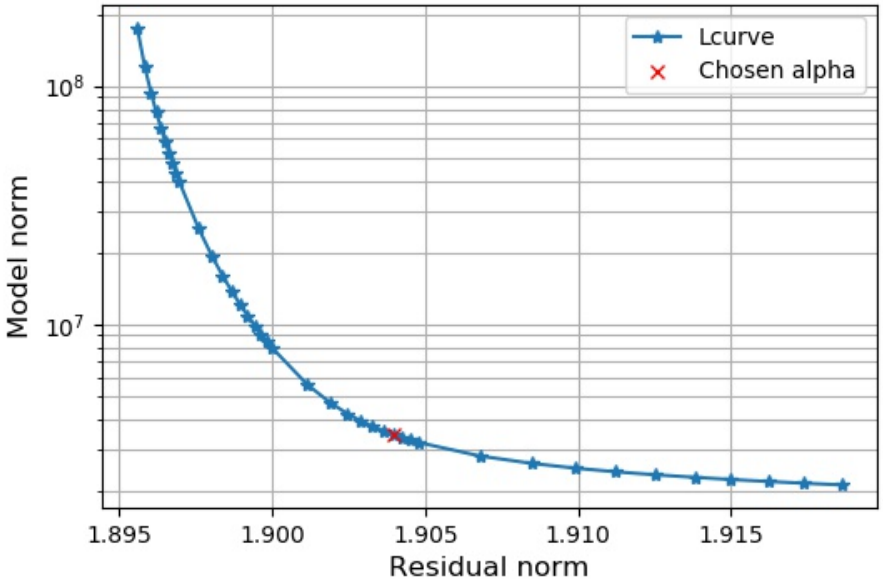


Figure 6.9: L_1-norm regularized model and residual norms for Australia as computed by Equations (3.43) and (3.44), respectively, with residual norm normalized with respect to the discrepancy principle.

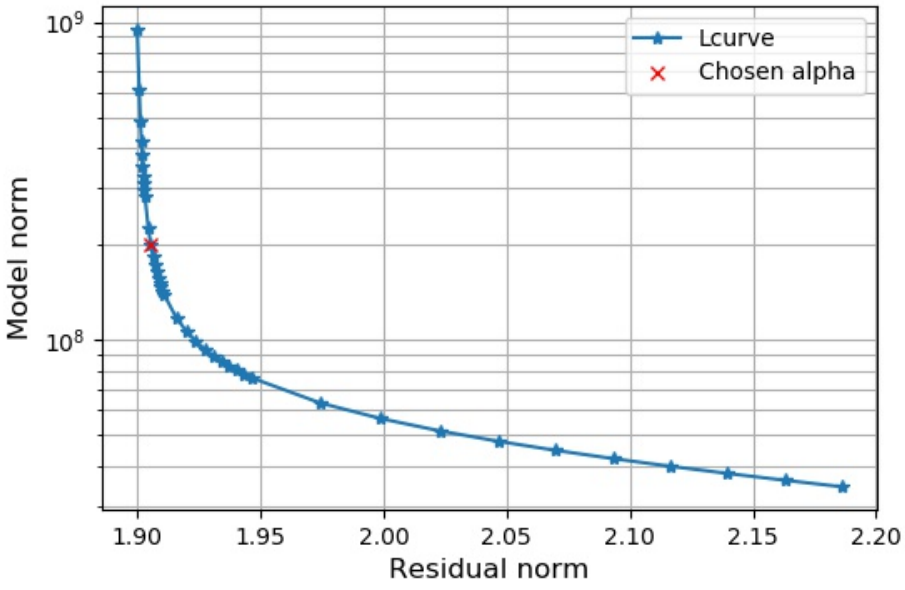


Figure 6.10: L_2-norm regularized model and residual norms for Australia as computed by Equations (3.36) and (3.37), respectively, with residual norms normalized with respect to the discrepancy principle.

The lithospheric magnetic field maps produced from L_1-norm and L_2-norm regularization using the regularization parameters from Table 6.1 are presented in Figures 6.11 and 6.12. The L_1-norm prediction introduces some edge effects along the West North-West edges, but the anomalies related to Australia seem to be reasonably captured by the model. The South-Central anomaly around $(lon, lat) = (138^\circ\ E,\ 35^\circ\ S)$ of Australia alongside smaller-scale features across the country are looking promising. The L_2-norm prediction does not have the same edge effects as the L_1-norm prediction does. The structures are, as expected, not as prominent in the L_2-norm prediction, but features such as the South-Central anomaly and many smaller scale features do seem promising with the L_2-norm prediction as well. Comparison to the EMM2015 model truncated to spherical harmonic degree 185 is done in Section 7.1.2.

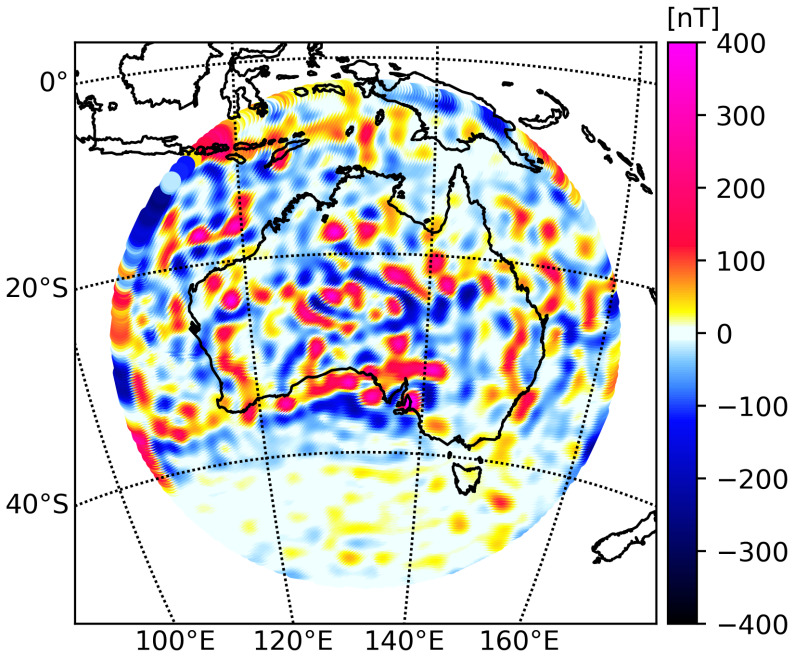


Figure 6.11: L_1-norm regional model prediction of the radial field at the Earth's mean spherical radius for Australia.

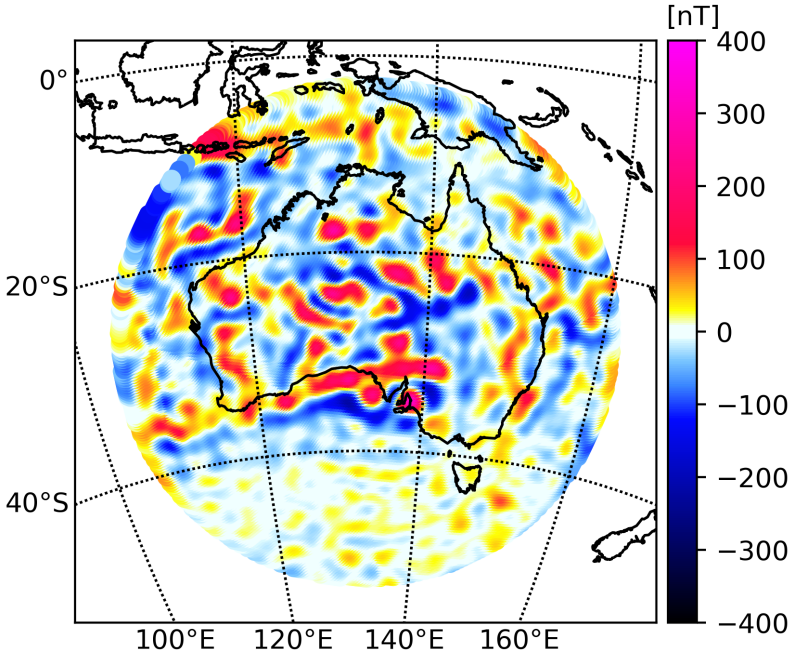


Figure 6.12: L_2-norm regional model prediction of the radial field at the Earth's mean spherical radius for Australia.

Residual histograms of the data misfit are produced, and presented in Figures 6.13 and 6.14. Both histograms are centred near zero both with a mean of 0.035 nT. Large outliers are seen in both histograms, both with a maximum of 7467.2 nT and a mimumum of −6086.3 nT. This could indicate that the edge effect seen in the L_1-norm prediction map are not the cause of these outliers, as they might not otherwise be present in the data misfit histogram for the L_2-norm prediction. I have determined the outliers to be due to the co-latitudinal and longitudinal components, with the main contributor being the longitudinal component. The curiosity of the large outliers is investigated in Figures 6.15a and 6.15b where residuals are presented in scatter plots. Both scatter plots clearly indicate that the largest outliers are located in the southern area of the spherical cap. Furthermore, along-track features are evident.

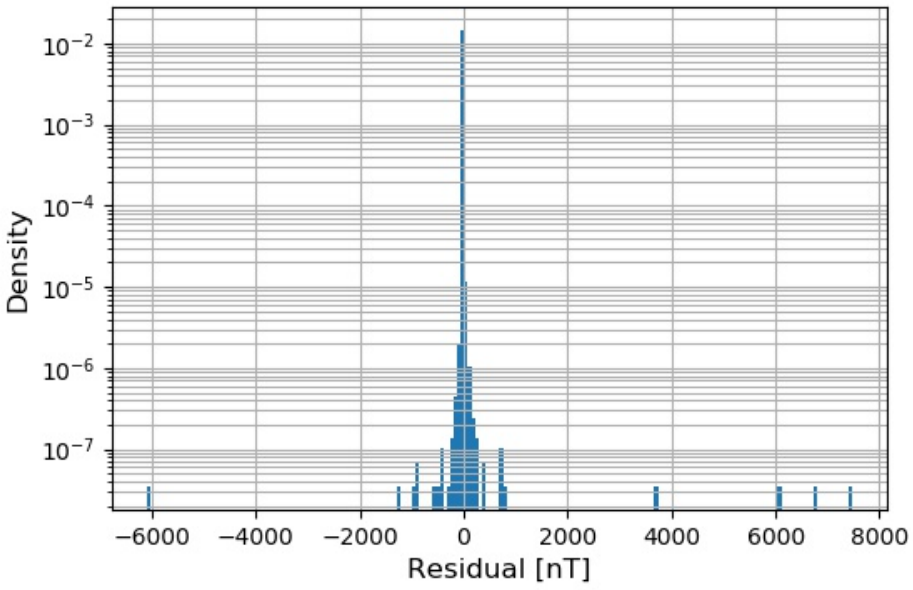


Figure 6.13: Histogram showing the data misfit distribution of the L_1-norm regularized prediction over Australia.

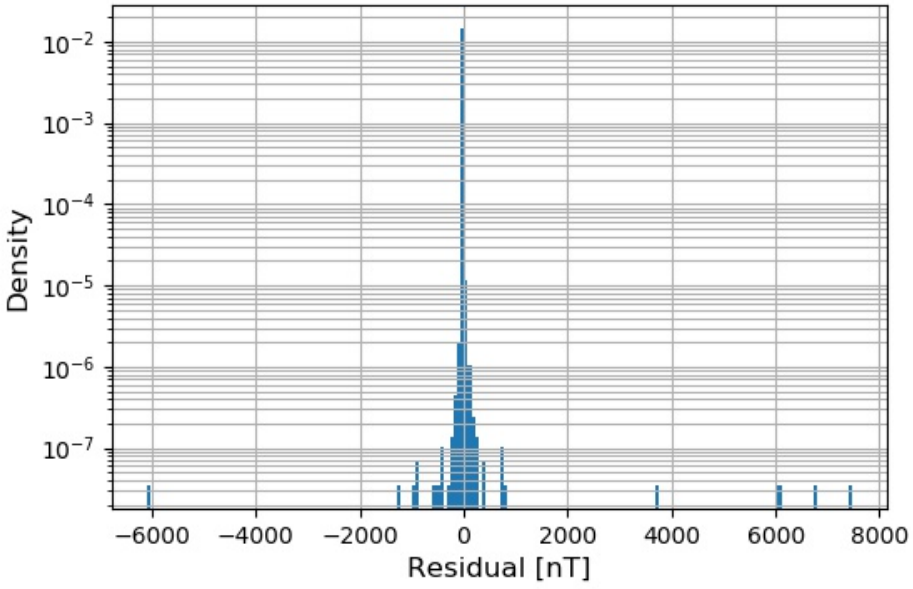
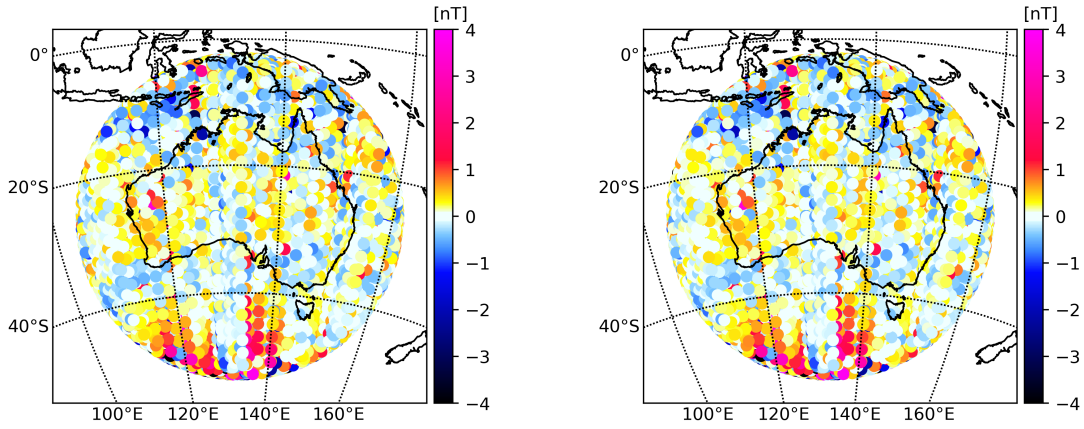


Figure 6.14: Histogram showing the data misfit distribution of the L_2-norm regularized prediction over Australia.



(a) L_1-norm. **(b)** L_2-norm.

Figure 6.15: Scatter plots of the L_1-norm and L_2-norm prediction longitudinal component data misfit for Australia.

A Mauersberger-Lowes power spectrum is computed for these models, presented in Figure 6.16. The two spectra shows the same tendencies, suggesting good agreement with varying amplitude. Similar to the Bangui Anomaly, a large spike at high spherical harmonic degree exist.

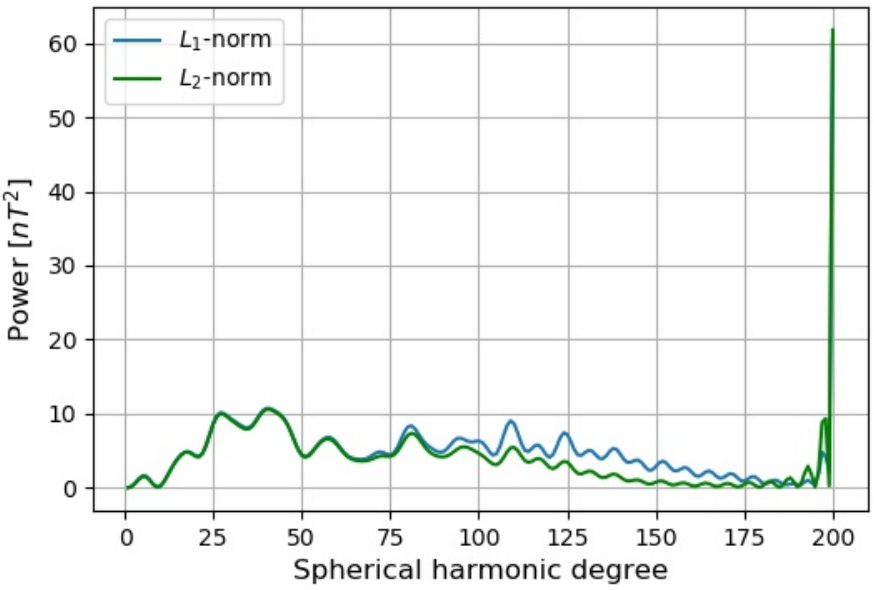


Figure 6.16: Mauersberger-Lowes power spectrum over Australia.

6.3 Walvis Ridge

The Walvis Ridge region provides a mostly oceanic region which cater the interesting opportunity of investigating how well this regional modelling approach detects the South Atlantic isochrones, as well as other potentially interesting features.

The L-curves produced for both types of regularization are presented in Figures 6.17 and 6.18, and are normalized with respect to the discrepancy principle. Residual norms for both types of regularization crosses 1 nT, and the regularization parameters that produces the residual norm closest to one, should be a sensible choice. These regularization parameters, however, produce highly regularized prediction maps, therefore smaller values for the regularization parameters are chosen. This serves as an example that the Walvis Ridge data uncertainties are not fully to be trusted when selecting regularization parameters using the discrepancy principle. The final chosen α^2 values do not lie far from the knee of the L-curve and are marked in the Figures. These are also considered a sensible trade-off between minimization of model norm and residual norm.

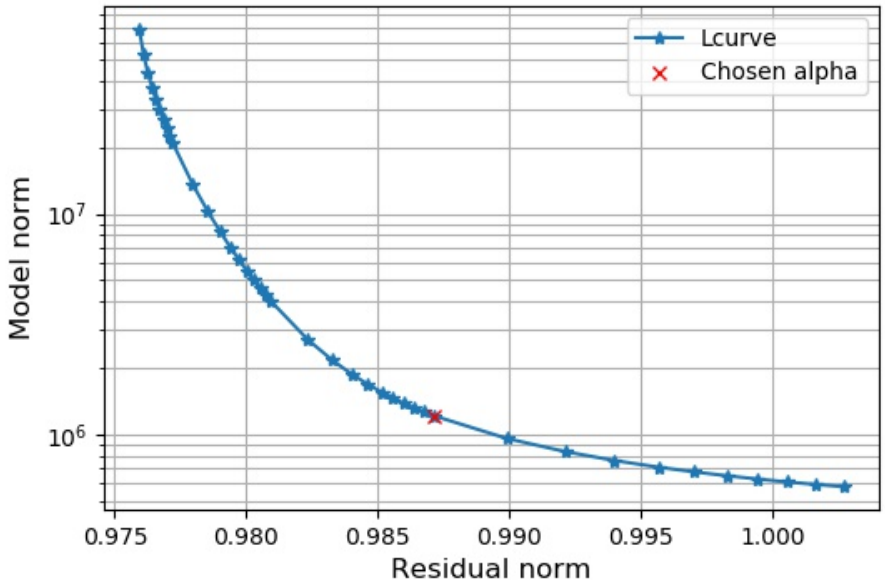


Figure 6.17: L_1-norm regularized model and residual norms for the Walvis Ridge as computed by Equations (3.43) and (3.44), respectively, with residual norm normalized with respect to the discrepancy principle.

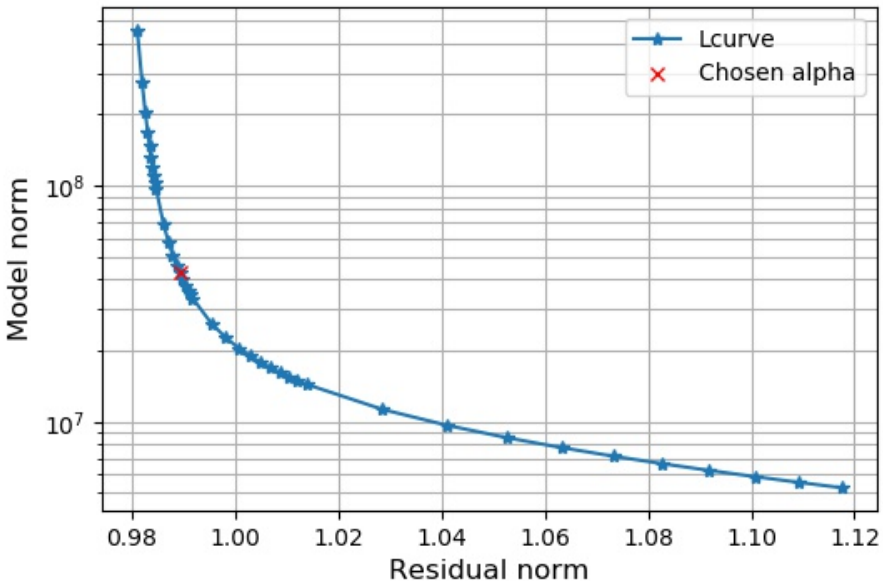


Figure 6.18: L_2-norm regularized model and residual norms for the Walvis Ridge as computed by Equations (3.36) and (3.37), respectively, with residual norms normalized with respect to the discrepancy principle.

The L_1-norm and L_2-norm regularized prediction maps are presented in Figures 6.19 and 6.20. Both models seem to capture the more prominent features off the Western coast of Africa. The South Atlantic isochrones do seem to be captured by the regional models as well, as there are several North-South trends in the Western longitudes. Comparison with the LCS-1 model is done in Section 7.1.3, where this will be further investigated.

Data misfit histograms are presented in Figures 6.21 and 6.22. These are closely centred around zero with the exception of a few large outliers, suggesting that the majority of data has been fitted sufficiently.

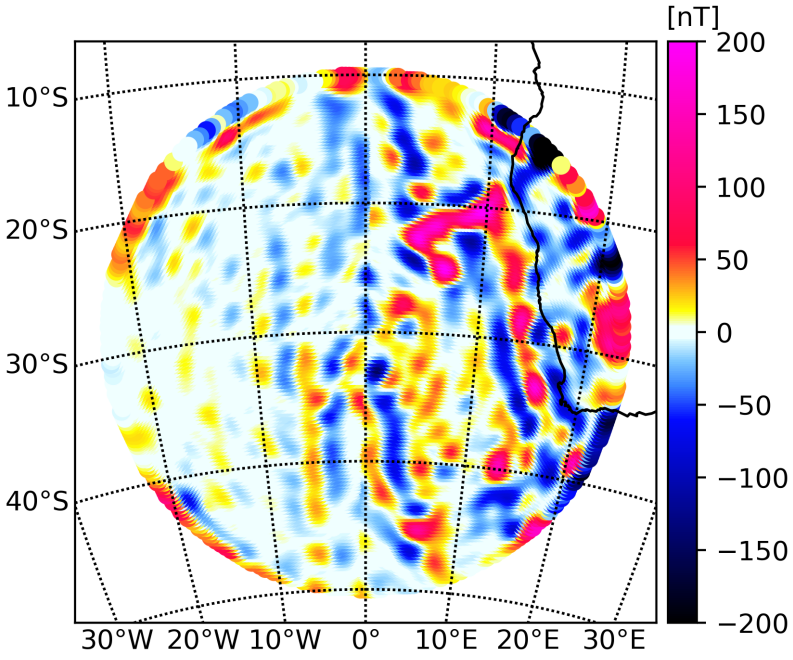


Figure 6.19: L_1-norm regional model prediction of the radial field at the Earth's mean spherical radius for the Walvis Ridge.

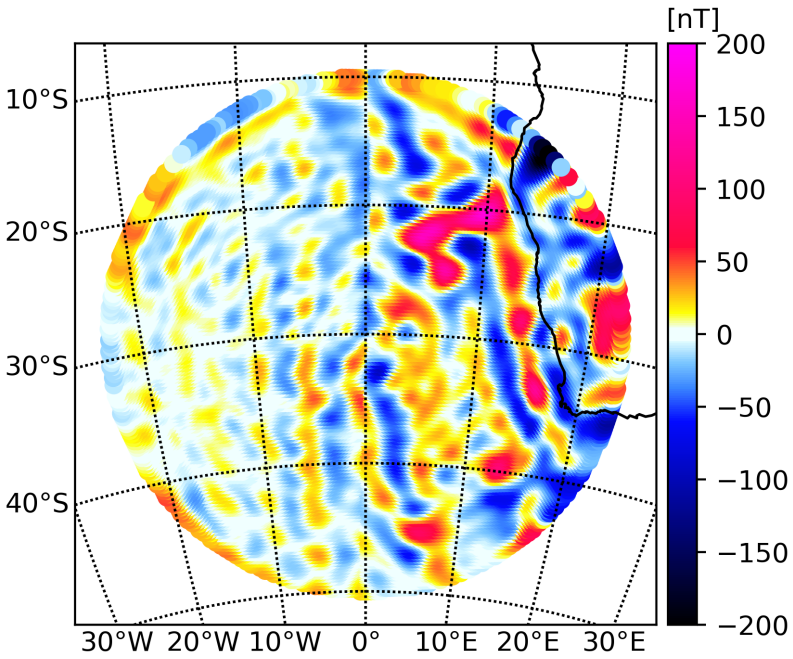


Figure 6.20: L_2-norm regional model prediction of the radial field at the Earth's mean spherical radius for the Walvis Ridge.

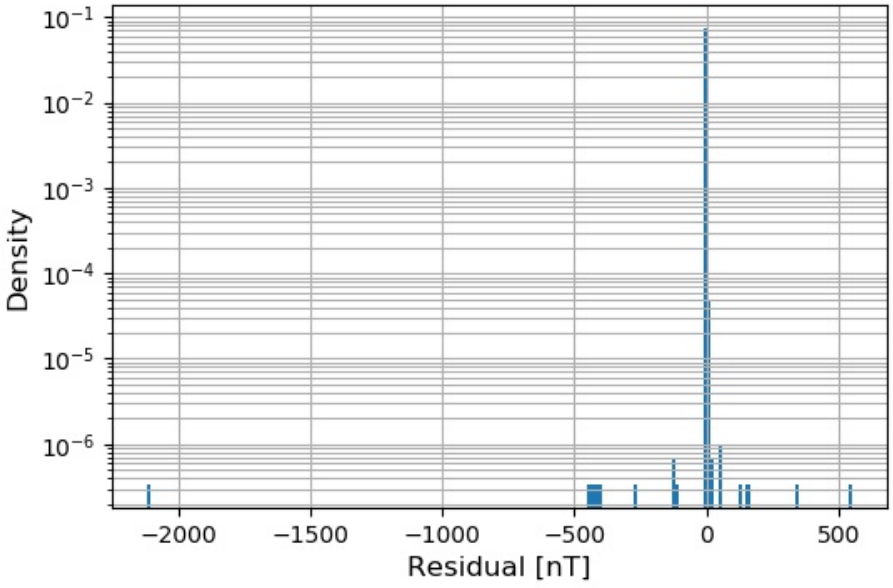


Figure 6.21: Histogram showing the data misfit distribution of the L_1-norm regularized prediction over the Walvis Ridge.

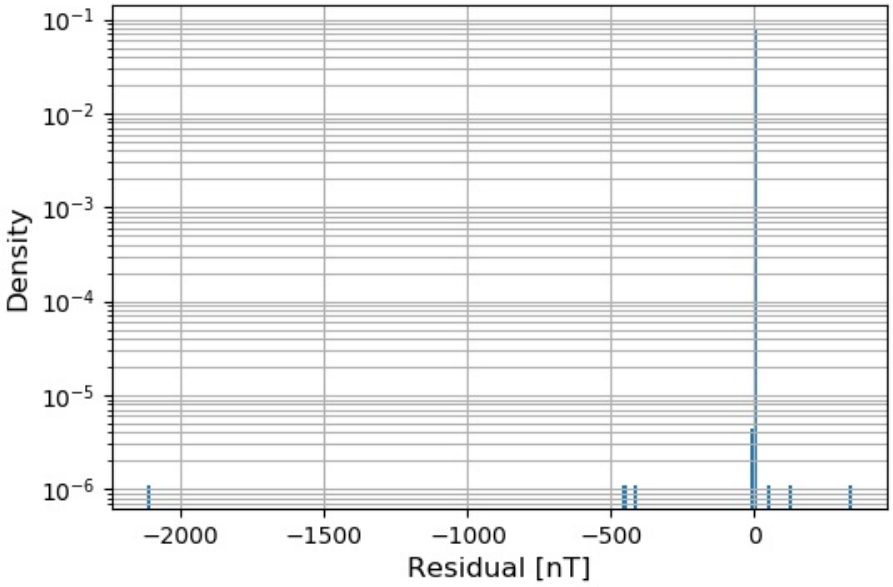


Figure 6.22: Histogram showing the data misfit distribution of the L_2-norm regularized prediction over the Walvis Ridge.

Finally a Mauersberger-Lowes power spectrum for each regularization method is computed and presented in Figure 6.23. Again agreement between the two models is evident, and indeed the order of magnitude of the power is within a reasonable range. Also in this case a large spike appears at high spherical harmonic degree.

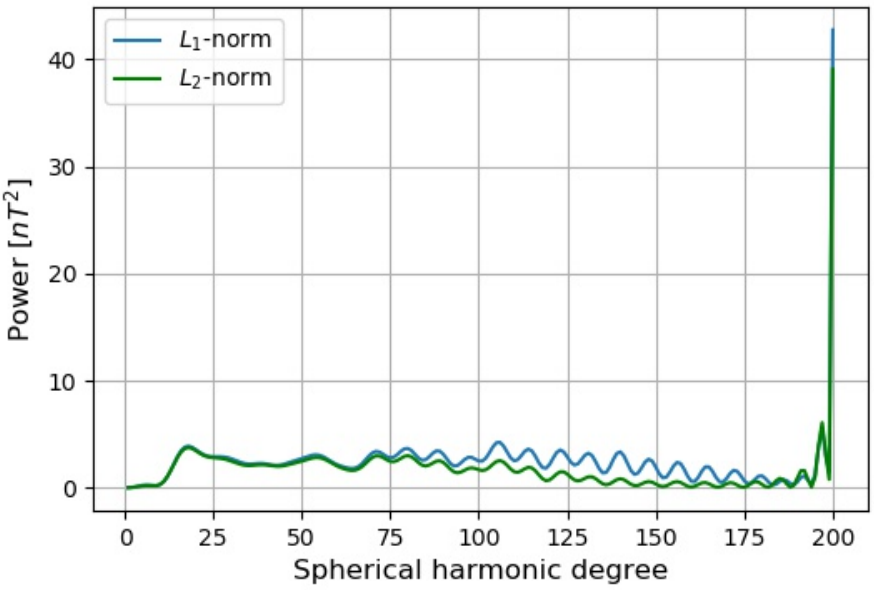


Figure 6.23: Mauersberger-Lowes power spectrum over the Walvis Ridge.

6.4 Greenland

The most challenging test of the regional modelling tool are at high latitude regions. Greenland is the case of such a region. At this high latitude, the noise contributions from the co-latitudinal and longitudinal components are too large and complex, therefore the regional models are constructed using only radial data. The L-curves produced for both regularization methods are presented in Figures 6.24 and 6.25. Both has been normalized with respect to the discrepancy principle. The L_1-norm model has a regularization parameter close to the knee of the L-curve, while the L_2-norm model is much less regularized resulting in a larger model norm. For this high latitude region, the discrepancy principle is not applicable, because the residual norms are all larger than one.

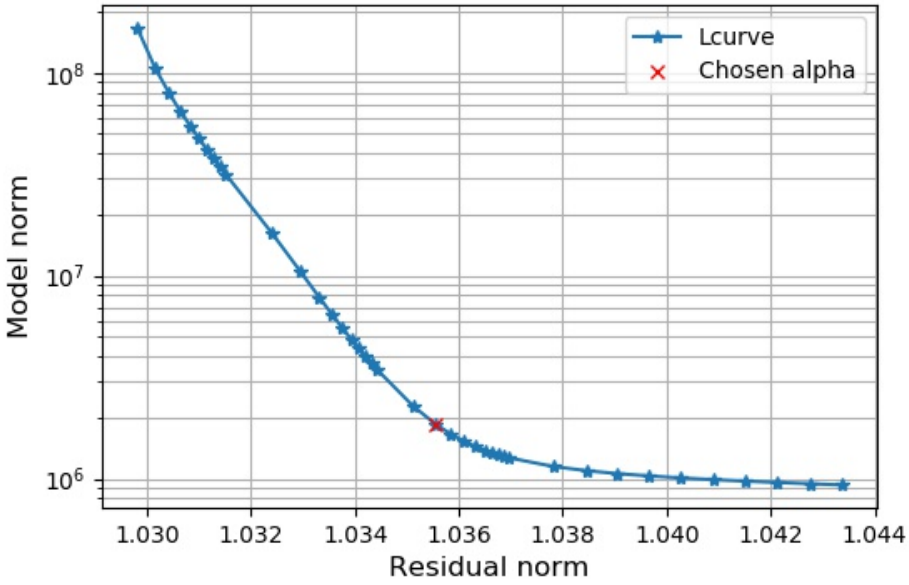


Figure 6.24: L_1-norm regularized model and residual norms for Greenland as computed by Equations (3.43) and (3.44), respectively, with residual norms normalized with respect to the discrepancy principle.

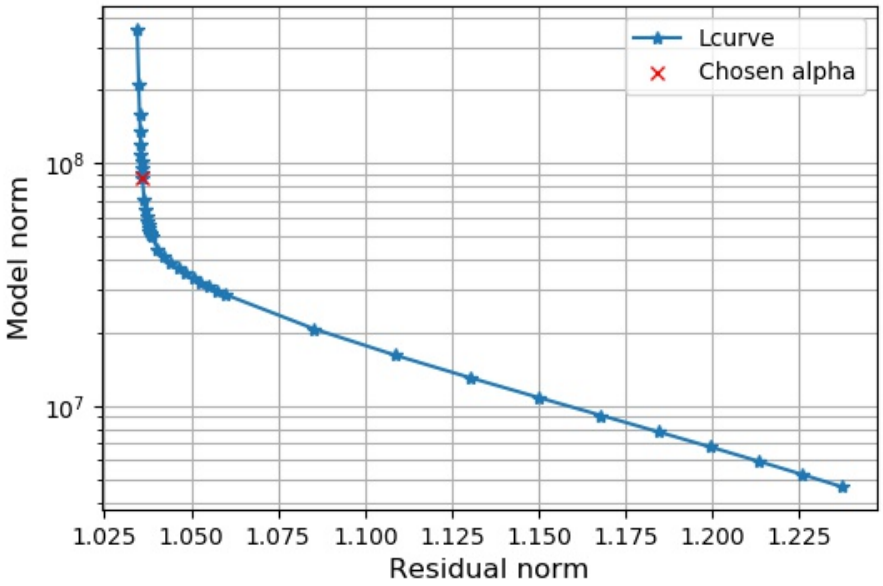


Figure 6.25: L_2-norm regularized model and residual norms for Greenland as computed by Equations (3.36) and (3.37), respectively, with residual norms normalized with respect to the discrepancy principle.

The prediction maps for both regularization methods are presented in Figures 6.26 and 6.27. Many structures around the entire country are captured with these regional models. There is one peculiar structure, in South-West Greenland around $(lon, lat) = (63^\circ\ N,\ 50^\circ\ W)$, stretching into the ocean westwards, appear in both the L_1-norm and L_2-norm regularized model predictions. This feature will definitely be subject to discussion when compared to the EMM2015 model truncated to spherical harmonic degree $L = 185$ in Section 7.1.4. The L_1-norm and L_2-norm predictions produces similar results, with the L_1-norm seemingly producing more well-defined structures for the anomalies. The detailed structures over Greenland suggests that this regional approach produces reasonable results, but comparison to known models must be conducted before any conclusions can be made.

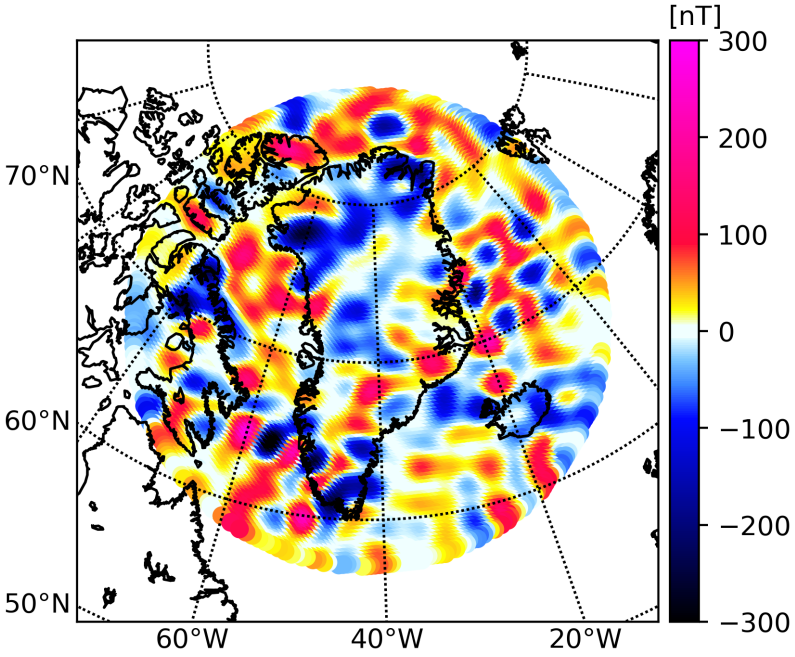


Figure 6.26: L_1-norm regional model prediction of the radial field at the Earth's mean spherical radius for Greenland.

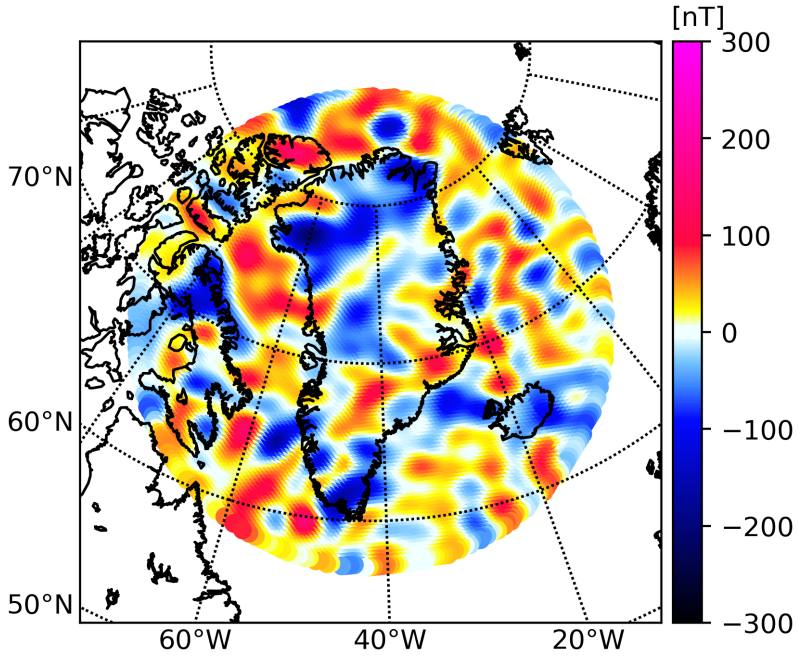


Figure 6.27: L_2-norm regional model prediction of the radial field at the Earth's mean spherical radius for Greenland.

The data misfit only include the radial contributions, and are presented for both regularization methods in Figures 6.28 and 6.29. Data misfit means for the L_1-norm and L_2-norm regularized models are very similar at -0.0084 nT and -0.0085 nT, respectively, with few outliers present.

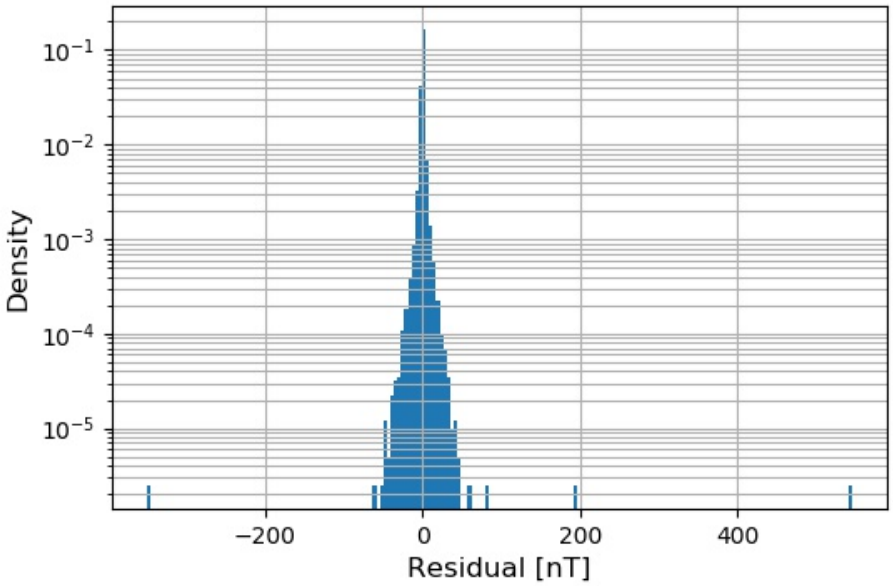


Figure 6.28: Histogram showing the data misfit distribution of the L_1-norm regularized prediction over Greenland.

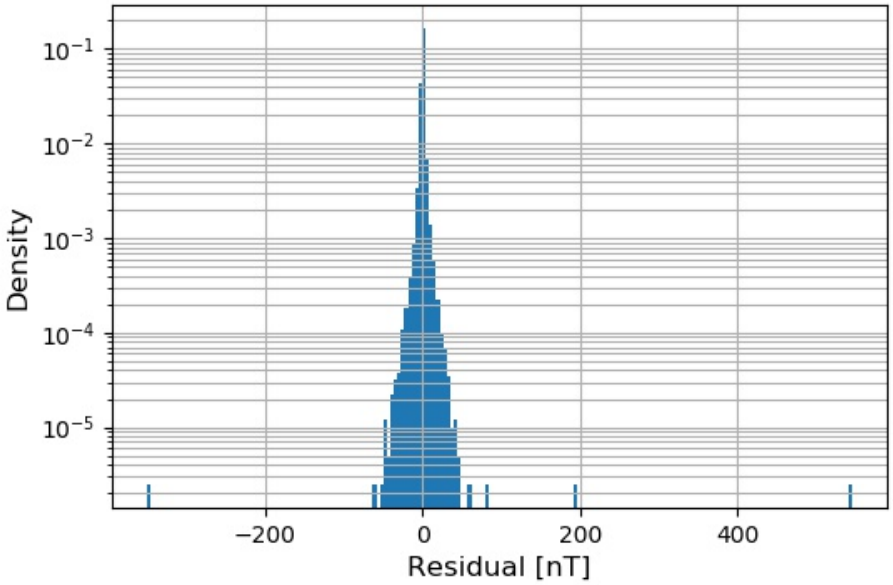


Figure 6.29: Histogram showing the data misfit distribution of the L_2-norm regularized prediction over Greenland.

A Mauersberger-Lowes power spectrum is computed for each regularization method and presented in Figure 6.30. There is an overall good agreement, and the power spectrum is at a reasonable order of magnitude, suggesting that the model parameters are reasonable. Again a large peak is visible at high spherical harmonic degrees.

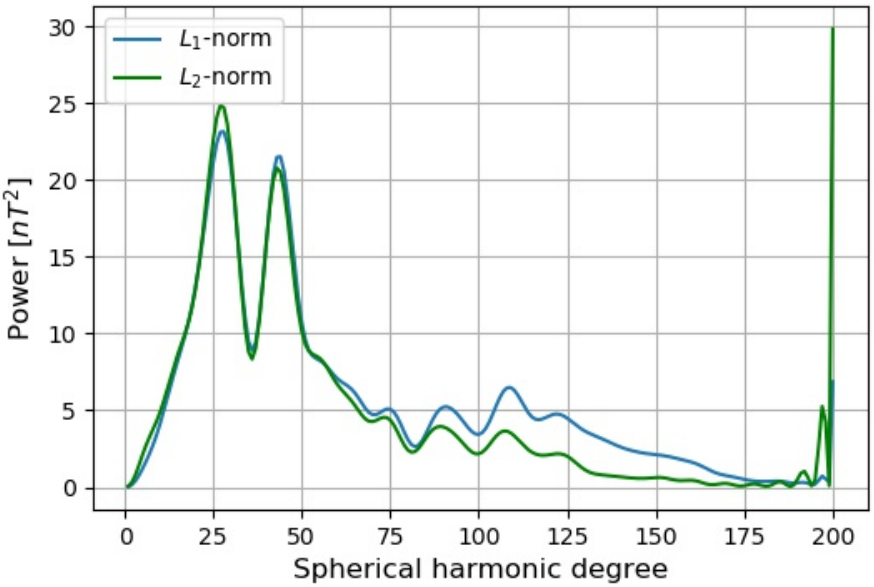


Figure 6.30: Mauersberger-Lowes power spectrum over Greenland.

6.5 Model Statistics

Table 6.2 presents the relevant model norm and RMSE statistics for the final models presented in Sections 6.1 through 6.4. The RMSE values of the targeted regions vary. It seems reasonable and expected when investigating the data misfit histograms for each region. E.g. Bangui has no large outliers and its larger misfit values are close to symmetric around zero. Australia introduces some very large outliers that are not symmetric around zero, which could suggest an RMSE larger than zero which is also observed. I believe, despite the very large outliers present, the data has been overall sufficiently fitted with these regional models, based on RMSE values and data mistfit histograms. Model norms for both regularization types, for all regions, are within a reasonable range, and there is thus no reason to suspect that the model parameters are unrealistic.

Table 6.2: Model and misfit statistics of the regional models produced in this thesis. The model norms $\|\mathrm{m}_{J,\alpha}\|$ are computed using Equations (6.4) and (6.3), and the RMSE are computed using Equation (6.2) for each of the two regularized solutions per region.

	Bangui		Australia		Walvis Ridge		Greenland	
	L_1-norm	L_2-norm	L_1-norm	L_2-norm	L_1-norm	L_2-norm	L_1-norm	L_2-norm
$\|\mathrm{m}_{J,\alpha}\|$	21.87 nT	2631.94 nT^2	24.11 nT	1386.64 nT^2	16.26 nT	569.31 nT^2	20.46 nT	964.78 nT^2
RMSE	0.660 nT	0.660 nT	21.48 nT	21.48 nT	5.089 nT	5.089 nT	3.61 nT	3.61 nT

Chapter 7

Discussion

In this chapter I will discuss the results of the regional models produced using the Python toolbox. Because other lithospheric magnetic field models exist, such as LCS-1 and EMM2015, I will investigate how the newly produced regional models differ from existing global models. I will discuss the Slepian truncation parameter J with respect to the regional model constructed over the Bangui Anomaly. I will reflect on the uses of the Python toolbox as *a priori* modelling to aeromagnetic data. Furthermore, a discussion of the impact on the regional models as the *Swarm* satellites descends. Lastly I will reflect on advantages and limitations of regional modelling with a Slepian approach.

7.1 Model Comparison

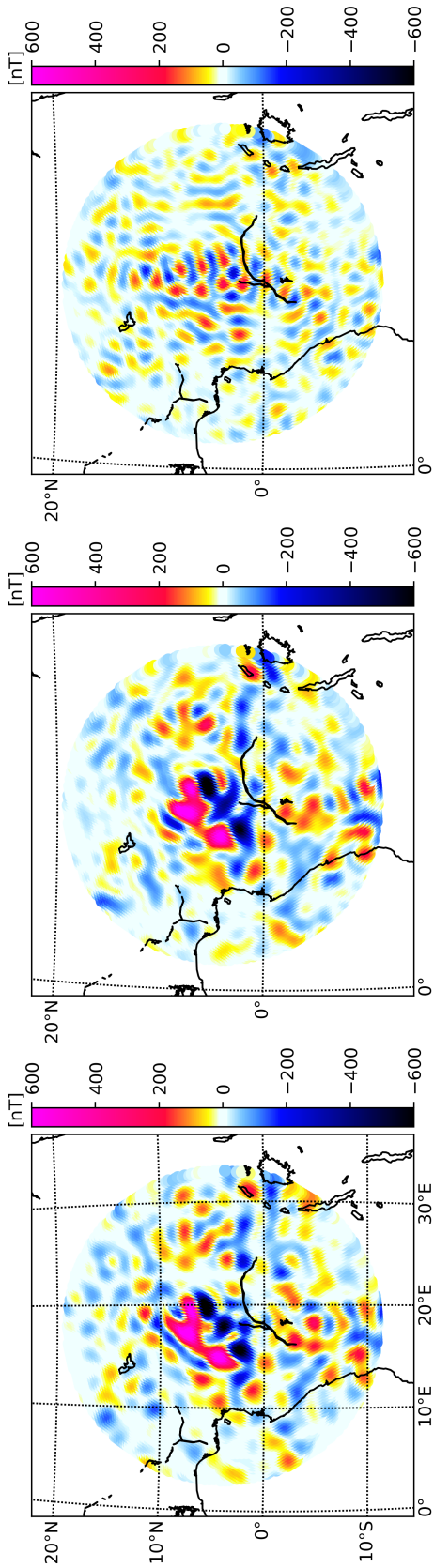
In this section I will discuss the Slepian approach to regional modelling with respect to the global models LCS-1 and EMM2015 truncated to spherical harmonic degree $L = 185$. I obtain the LCS-1 prediction by solving the forward problem $\mathrm{d}^{mod} = \mathbf{G}\mathrm{m}_{LCS-1}$, where $\mathbf{G}$ is the design matrix constructed with the same grid used for the regional model and m_{LCS-1} are the LCS-1 model parameters. The same procedure is used for obtaining EMM2015 data, where m_{LCS-1} is replaced with $\mathrm{m}_{EMM2015}$.

7.1.1 The Bangui Anomaly

The L_1-norm regularized model prediction of the Bangui Anomaly presented in Section 6.1 is compared to the LCS-1 model visually in Figure 7.1. The L_2-norm prediction comparison is found in Appendix B.1.

The regional prediction, Figure 7.1a, nicely reproduces the expected anomalies found in the LCS-1 prediction, Figure 7.1b. The anomaly itself seem to show more detailed structure, particularly in the Northern and Central region in the regional model. The smaller features surrounding the anomaly are largely captured equally for the regional and LCS-1 predictions. The regional prediction may even show a little less coherent structures, and is more affected by the L_1-norm regularization. The regional prediction has maximum and minimum of 895.5 nT and -1018.6 nT, respectively, against the LCS-1 prediction with 821.9 nT and -970.3 nT as maximum and minimum. This indicates that the regional model produces larger amplitudes compared to a global model, which is expected due the regional optimization. Figure 7.1c shows a difference map of the radial component of the LCS-1 model and the L_1-norm regularized regional model. The amplitudes of the differences are highest over the anomaly which is a result of the regional model producing larger amplitudes over the anomaly. The differences in the surrounding regions suggest good agreement with the

LCS-1 model. Considering this is a first attempt at constructing a regional model over the Bangui Anomaly using the Python toolbox, I think the approach shows excellent promise at low latitudes.



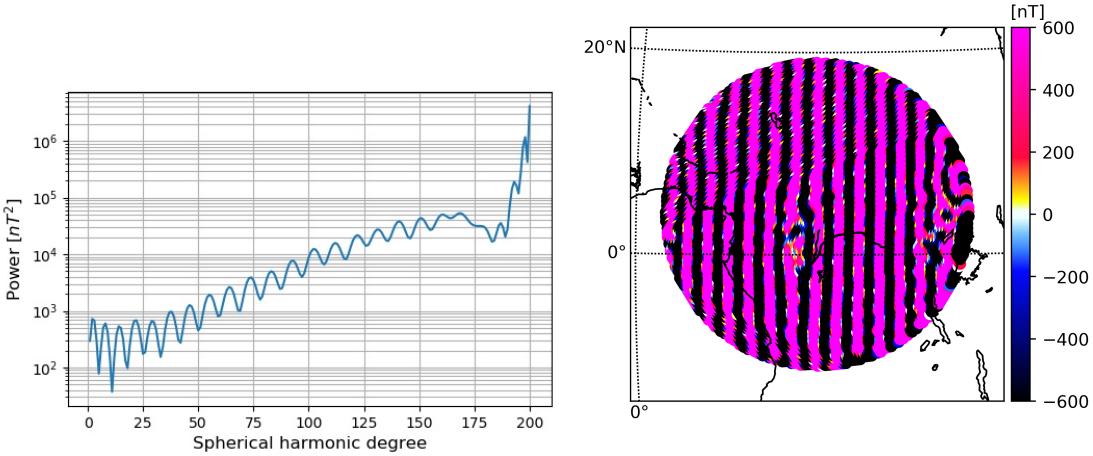
(a) L_1-norm regional model prediction of the radial field at the Earth's mean spherical radius.

(b) LCS-1 prediction of the radial field at the Earth's mean spherical radius.

(c) Difference map of the radial component of the LCS-1 model and the L_1-norm regularized regional model.

Figure 7.1: Radial components of the regional L_1-norm prediction and the LCS-1 prediction over the Bangui Anomaly together with a difference map between the to predictions.

The Mauersberger-Lowes power spectra shown in Figure 6.8 introduced peculiar behaviour at large spherical harmonic degrees where the power sharply increased. I urge caution interpreting upon this result, as the Gauss coefficients are not directly obtained, but rather computed via the Slepian model parameters and the kernel matrix eigenvectors. This tendency where a sharp peak appears at large spherical harmonic degrees are likely due to numerical instabilities. I use the Mauersberger-Lowes power spectra mainly to investigate if the regularization and model parameters are reasonable. An unreasonable regularization causes the Gauss coefficients obtained from the Slepian model parameters to either 'explode' or be suppressed which is visible in such a power spectrum. Figure 7.2 presents an example of the former, where the regularization parameter for an L_1-norm regularized solution is $\alpha^2 = 1 \times 10^{-6}$ nT^{-1}, which will result in a largely under-regularized model. The model produced in this example is clearly not sensible.



(a) Mauersberger-Lowes power spectrum.

(b) Prediction map at Earth's mean spherical surface. Radial component.

Figure 7.2: Example of an under-regularized L_1-norm regularized solution.

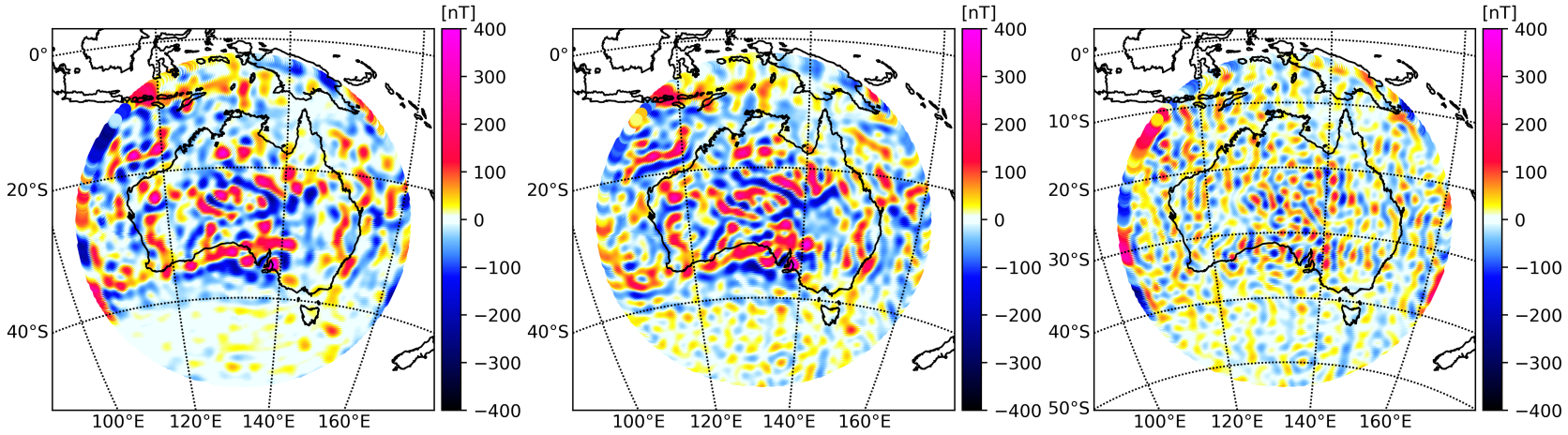
7.1.2 Australia

I compare the L_1-norm predictions of the regional model over Australia, with predictions from the EMM2015 model in Figure 7.3. Comparisons of both regularization types with the LCS-1 model and the L_2-norm regularized solution with the EMM2015 model are found in Appendix B.2.

The regional model, Figure 7.3a, of Australia overall captures the same anomalies as the EMM2015 model, Figure 7.3b, does. The South Central anomaly is well presented in the regional model, and the smaller structures in East Australia seem to be more coherent compared to the EMM2015 prediction. Predictions of West Australia seem quite similar for the two models, with the exception of some probably better defined smaller features in the EMM2015 prediction. The maximum and minimum values of the regional prediction are 528.77 nT and −285.99 nT, respectively, while for the EMM2015 prediction they are 506.16 nT and −364.10 nT, respectively. For the EMM2015 prediction, the minimum value is located within the South Central anomaly, and is from the radial component of the magnetic field prediction. The regional prediction minimum is located at 138° E and 15° S which is in the Northern part of Australia, from the co-latitudinal component of the magnetic field prediction. Thus, the two models predict the location of the minimum values very differently. In the case of the Australia region, the regional model do not show larger amplitudes as for the Bangui Anomaly, only the positive value shows larger amplitude. I suspect the estimation of the data uncertainties used in the inversion may have caused the large data misfit from Figure 6.13. This will impact the regional model and may explain why the regional model does not produce larger amplitudes in predictions.

From the difference map in Figure 7.3c it is clear that edge effects are present in the regional model. But it also becomes clear that removing the outer five degrees of the spherical cap prediction will provide very reasonable results when compared to EMM2015. The difference map shows largest amplitudes over larger anomalies such as the South Central Anomaly. Due to the lower amplitudes observed otherwise, I believe there is good agreement between the two models.

The regional model of Australia perhaps lack some details in the anomaly structures compared to EMM2015. However, the regional model presented in this thesis is a first attempt at producing such a regional model and improvements can be made in areas such as data selection, data uncertainties and tuning of the regularization parameter.



(a) L_1-norm regional model prediction of the radial field at the Earth's mean spherical radius.

(b) EMM2015 prediction of the radial field at the Earth's mean spherical radius.

(c) Difference map of the radial component of the EMM2015 model and the L_1-norm regularized regional model.

Figure 7.3: Radial components of the regional L_1-norm prediction and the EMM2015 prediction over Australia together with a difference map between the to predictions.

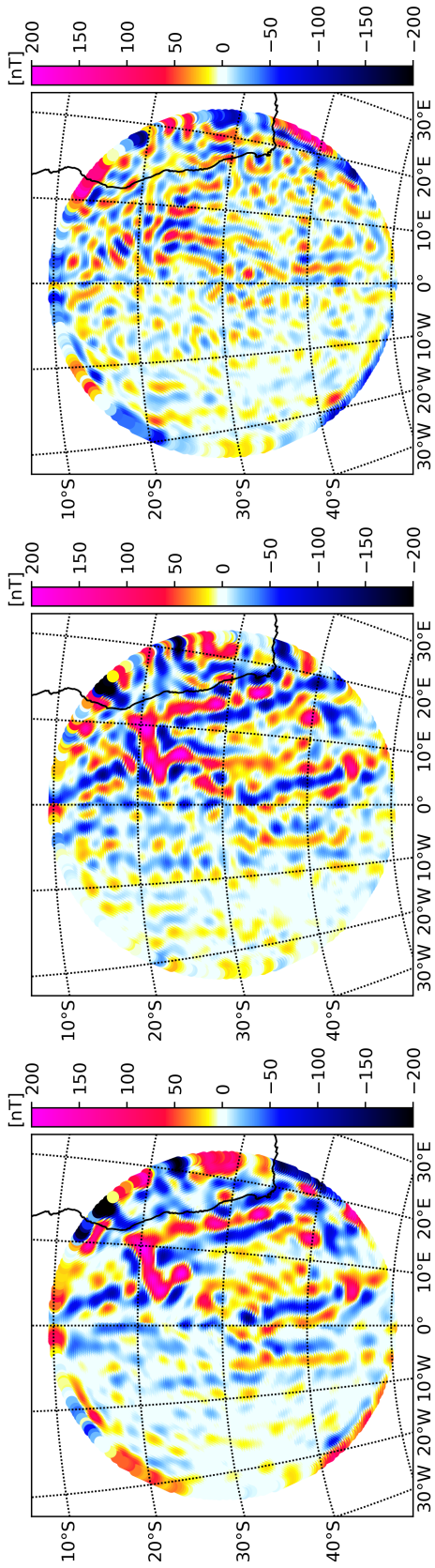
7.1.3 Walvis Ridge

The L_1-norm regularized regional model is compared to the LCS-1 model in Figure 7.4. The L_2-norm regional model comparison with LCS-1 is found in Appendix B.3

The regional model, Figure 7.4a, of the Walvis ridge also captures the overall anomalies compared to the LCS-1 model, Figure 7.4b. Maximum and minimum for the regional model are 230.38 nT and −420.63 nT, respectively, and for LCS-1 they are 236.39 nT and −349.05 nT. The regional model thus only captured larger values in the negative domain. I believe this is likely due to edge effects of the L_1-norm regularized model. The South Atlantic isochrones are perhaps not as well defined in the regional model, as in the LCS-1 model. Furthermore, the regional model also shows extensive edge effects as opposed to the LCS-1 model. The North-South trending feature described in Olsen et al. (2017), located between latitudes 30° S and 45° S and along approximately 5° E, seem to be captured in the regional model. This further suggests that this feature is real.

The difference map, Figure 7.4c has largest amplitudes over the Walvis Ridge Anomaly, which is expected. It suggests overall good agreement between the two models.

I believe the regional model has much room for improvement, however this is a first approach to a mainly oceanic inversion. Despite the discrepancies, I believe the regional model shows promise in this region.



(a) L_1-norm regional model prediction of the radial field at the Earth's mean spherical radius.

(b) LCS-1 prediction of the radial field at the Earth's mean spherical radius.

(c) Difference map of the radial component of the LCS-1 model and the L_1-norm regularized regional model.

Figure 7.4: Radial components of the regional L_1-norm prediction and the LCS-1 prediction over the Walvis Ridge together with a difference map between the to predictions.

7.1.4 Greenland

In this section I will discuss the regional prediction maps produced for Greenland compared with LCS-1 and EMM2015. This is a particularly interesting region due its high latitudes. The LCS-1 comparison with the L_1-norm regularized model is presented in Figure 7.5 and the EMM2015 comparison with the L_1-norm regularized model is presented in Figure 7.6. Both L_2-norm regularized model comparisons are found in Appendix B.4.

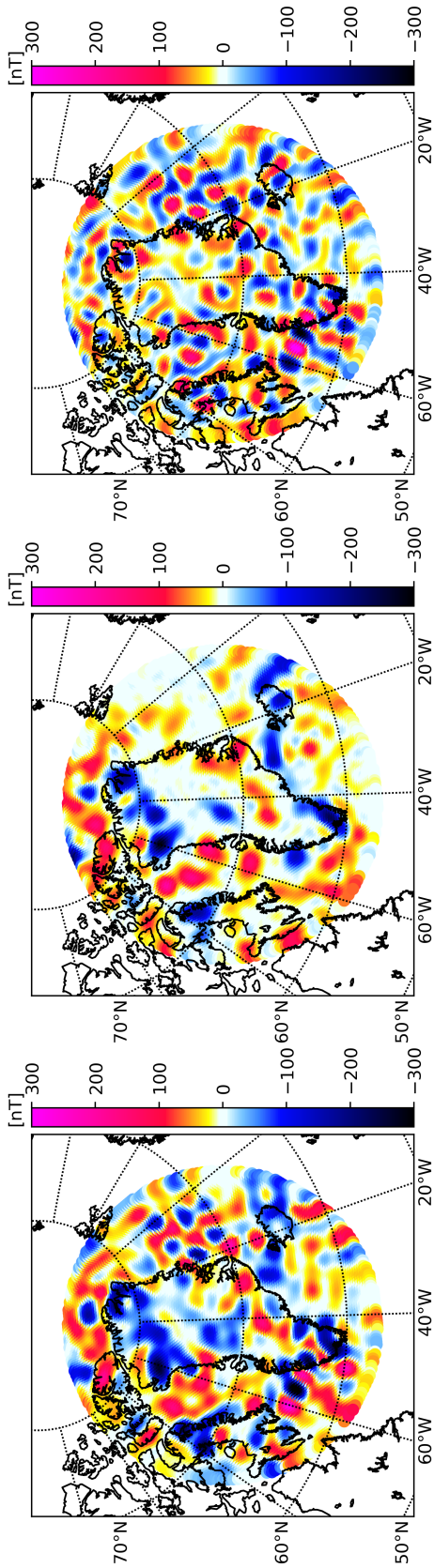
Over Greenland, studies of the Curie boundary using models of the lithospheric magnetic field is being carried out, e.g. in Martos et al. (2018). I believe that using regional models with aeromagnetic data combined with satellite data will provide excellent models for improving on these types of studies. The following results has been modelled only using satellite data.

7.1.4.1 Comparison With the LCS-1 Model

The regional prediction over the Greenland region, Figure 7.5a performs significantly better than the LCS-1 prediction of the same region, Figure 7.5b. The LCS-1 prediction is highly affected by the L_1-norm regularization. This comes to show as many small scale features are damped. Inspecting the regional model prediction, much more structures, large as well as small, is seen. The large scale features predicted by the regional model are also predicted by the LCS-1 model. This yields good agreement of large scale structures between the two models, but nothing can be concluded for the smaller scale features. The South-Western feature around $(lon, lat) = (63^\circ\ N,\ 50^\circ\ W)$ expanding westwards into the ocean is not visible in the LCS-1 prediction as it is for the regional model prediction. Thus nothing can be concluded about this peculiar feature. The regional model have maximum and minimum of 311.42 nT and -400.27 nT, respectively, and the LCS-1 model they are 198.73 nT and -288.17 nT which shows the regional model producing much larger amplitudes. The difference map of the radial component of the LCS-1 model and the L_1-norm regularized regional model, Figure 7.5c, is difficult to interpret upon, due to the obvious difference in small scale features of the two predictions.

7.1.4.2 Comparison With the EMM2015 Model

The regional prediction over the Greenland region, Figure 7.6a performs comparably to the EMM2015 prediction of the same region, Figure 7.6b. The EMM2015 prediction includes aeromagnetic data, which enables the model to predict more fine scale features, compared to the LCS-1 model. There is now some agreement to be seen with the smaller scale features of the EMM2015 model compared to the regional model. Agreement is also found in the large scale features, including the South-Western feature around $(lon, lat) = (63^\circ\ N,\ 50^\circ\ W)$ expanding westwards into the ocean. This suggests that the feature might be real. Generally, the smaller scale features over Greenland are predicted with smaller amplitude in the regional model compared to the EMM2015 model, with the exception of South-East Greenland. The regional model have maximum and minimum of 311.42 nT and -400.27 nT, respectively, and the EMM2015 model they are 195.66 nT and $-316,85$ nT. It is highly encouraging to see the regional model producing larger amplitudes than the EMM2015 model. The difference map of the radial component of the EMM2015 model and the L_1-norm regularized regional model, Figure 7.6c, suggests agreement between the two models. The degree of the agreement is up to debate, but I believe the regional model produces sensible results. The South-Western feature around $(lon, lat) = (63^\circ\ N,\ 50^\circ\ W)$ is evident in the difference map, which might suggest amplitudes are predicted differently between the two models.

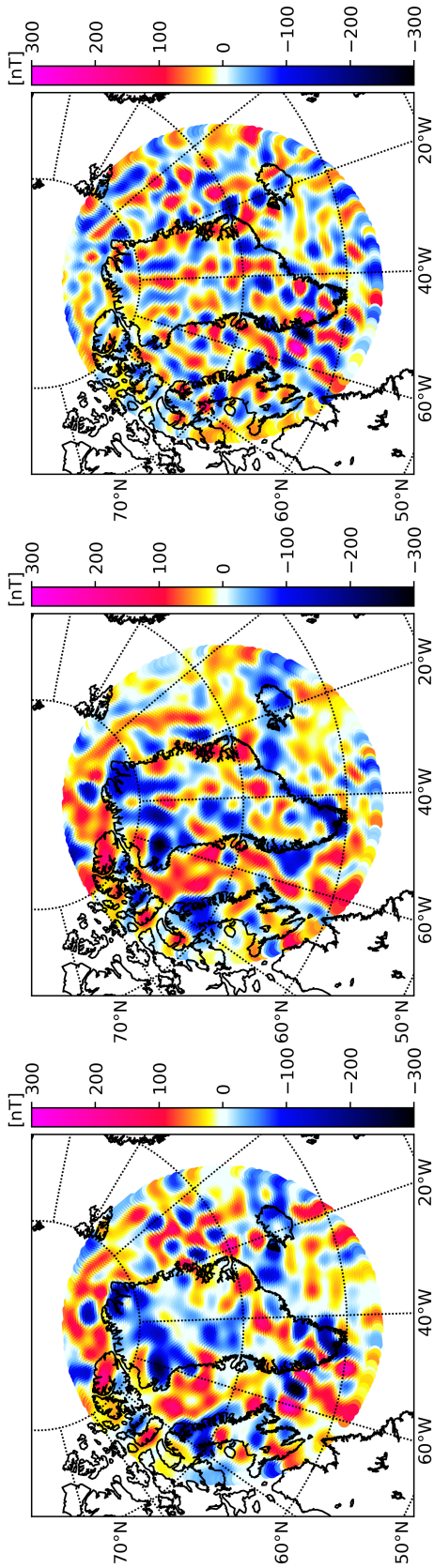


(a) L_1-norm regional model prediction of the radial field at the Earth's mean spherical radius.

(b) LCS-1 prediction of the radial field at the Earth's mean spherical radius.

(c) Difference map of the radial component of the LCS-1 model and the L_1-norm regularized regional model.

Figure 7.5: Radial components of the regional L_1-norm prediction and the LCS-1 prediction over Greenland together with a difference map between the to predictions.



(a) L_1-norm regional model prediction of the radial field at the Earth's mean spherical radius.

(b) EMM2015 prediction of the radial field at the Earth's mean spherical radius.

(c) Difference map of the radial component of the EMM2015 model and the L_1-norm regularized regional model.

Figure 7.6: Radial components of the regional L_1-norm prediction and the EMM2015 prediction over Greenland together with a difference map between the to predictions.

7.2 Influence of the Slepian Truncation Parameter

Up until this point it has been discussed that the Slepian truncation parameter J plays an important role in constructing sensible models. No real proof as been provided to back up this claim. In this section I will discuss the AC-GVSF power spectra for the Bangui Anomaly region, and how these are used as a diagnostic tool to determine a good Slepian truncation parameter.

The power spectra of the AC-GVSF are computed using Equation (3.51) and there are $J = 1500$ of them to investigate. Luckily they do not change rapidly, so I can investigate out-takes of the power spectra for this region. The power spectra of the AC-GVSF is an important diagnostic tool to investigate power distribution with respect to spherical harmonic degrees. If there are spherical harmonic degrees where the AC-GVSF power spectra do not deposit power, this indicates that the choice of J is bad. Consider the nine out-takes of the 1500 AC-GVSF power spectra in Figure 7.7. These suggest that power is in fact distributed through all spherical harmonic degrees apart from the very lowest that are related to the core field. This indicates a good choce in J. If J is chosen too small, power will mainly be distributed at lower spherical harmonic degrees, indicating that J must be increased. If J is chosen too large, instabilities can be introduced, due to some functions having little to no power in the spherical harmonic degrees that are investigated.

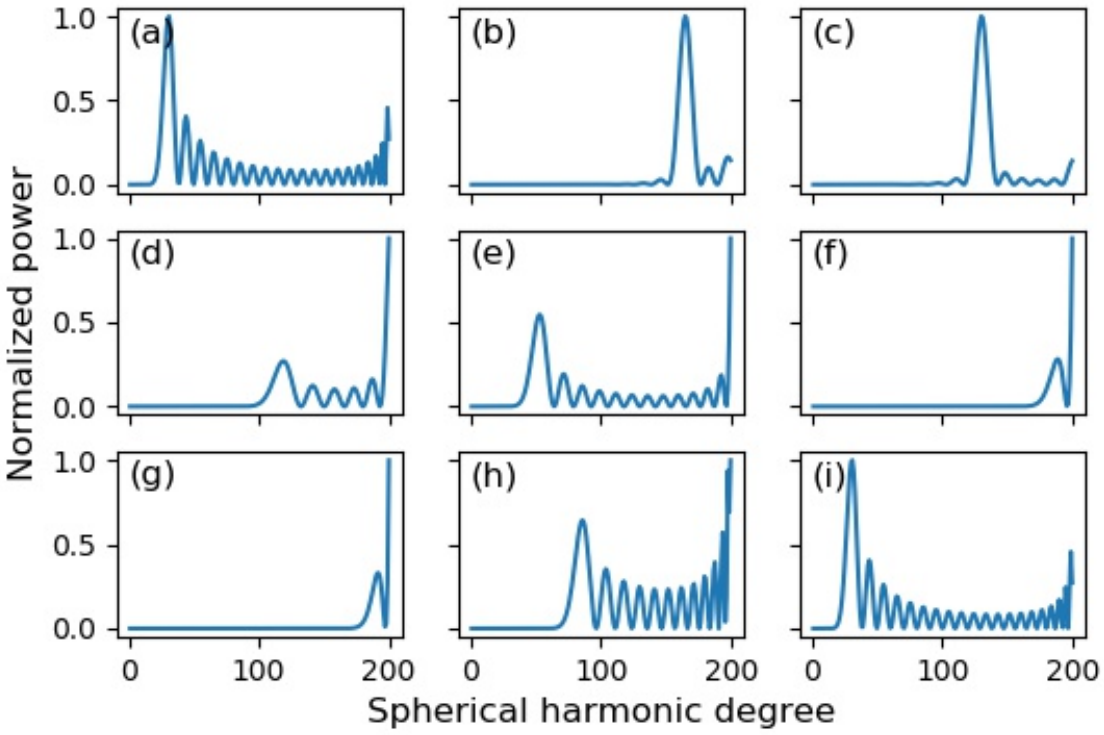


Figure 7.7: Normalized power of AC-GVSF with respect to spherical harmonic degrees. The normalization is with respect to the highest value in a given power spectrum, such that power varies from 0-1. Panel **a)** $J = 1$, **b)** $J = 200$, **c)** $J = 400$, **d)** $J = 600$, **e)** $J = 800$, **f)** $J = 1000$, **g)** $J = 1200$, **g)** $J = 1400$ and **h)** $J = 1500$.

7.3 Correction of Aeromagnetic Data

Aeromagnetic surveys may be used to investigate short wavelength magnetic signals of the lithospheric field. This is due to the relatively low observation altitudes of aeroplanes compared to satellites.

In order to properly investigate the short wavelength signals of the lithospheric field, the aeromagnetic data must be processed to exclude the effects of larger scale features, such as the core field and larger scale lithospheric contributions.

The Earth's magnetic core field, spherical harmonic degrees 1-15, can be filtered out using the CHAOS-6 model as *a priori* (Finlay et al., 2016). This thesis provides an excellent method for filtering out high spherical harmonic degrees related to the larger scale features of the lithospheric field. Regional effects are potentially better accounted for using the Slepian approach rather than a global model (e.g. LCS-1 (Olsen et al., 2017), MF7 (Maus et al., 2007) and EMM2015 etc.) because the optimization problem is performed locally. I suggest for low latitude regions to use the Slepian approach up to spherical harmonic degree $L = 175$. For higher latitudes, it is more difficult to provide suggestions at this stage. I will note that this regional approach looks very promising over Greenland, and I will not be surprised to see reliable high resolution models when a few improvements have been incorporated. For the moment, my suggestion is using this regional approach up to a maximum of spherical harmonic degree $L = 150$ for the higher latitudes.

I believe, because this regional approach suggest more details in some anomalies and smaller-scale features, it is going to be a good addition to the existing modelling tools.

7.4 As *Swarm* Descends

Comparing the Slepian approach to regional modelling with global models, particularly such as MF-6 (Maus et al., 2007) and MF-7, the aspect of what data is used must be considered. The MF-models uses CHAMP data exclusively, while LCS-1 incorporates not only *Swarm* data but also utilizes gradient data as introduced in Chapter 2.

This Slepian approach to regional modelling is based on an updated LCS-1 data set. For the time being the *Swarm* satellites are at relatively high altitudes compared to CHAMP as also seen in Figure 2 in Olsen et al. (2017). As the *Swarm* satellites descend it is expected to see an increase in power of higher spherical harmonic degrees, which is visualized in Figure 1 in Olsen et al. (2017).

Updating regional models with the years to come will definitely show improvements in the higher spherical harmonic degrees. Certainly, *Swarm* provides high quality data at the lower spherical harmonic degrees up to around degree $L = 120$. The remaining spherical harmonic degrees are currently likely to be explained at best using CHAMP data.

7.5 Regional Modelling Using a Slepian Approach

I dedicate this section to the pros and cons of regional modelling using the Slepian approach using the Python toolbox developed in this thesis. First a look at the advantages of using this regional modelling approach, followed by possible limitations.

7.5.1 Advantages

This thesis presents a newly developed Python toolbox that uses real, *globally* defined, spherical harmonics to perform *local* analyses of the Earth's lithospheric magnetic field. During comparison with e.g. LCS-1 predictions it became clear that this newly developed toolbox is capable of producing high resolution models that probably suggests more detailed structures to known anomalies.

The regional models are particularly interesting at higher latitudes, where global models such as LCS-1 seem to be limited. I have produced maps over Greenland where a much greater level of structure is seen. In lower latitudes this regional approach seem to produce results quite close to the LCS-1 model, with the exception of possibly introducing finer details.

What is definitely the case, is that interdisciplinary scientific studies will benefit greatly from better knowing e.g. the lithospheric magnetic field. I believe this regional approach offers a good opportunity to produce better estimates of the lithospheric magnetic field, particularly at high latitudes. Studies of the Curie boundary over Greenland could be an investigation that will benefit from regional modelling.

7.5.2 Limitations

All methods are bound by limitations. So is the Slepian approach to regional modelling. One of the more difficult aspects is determining the truncation parameter J. The diagnostic tools introduced in this thesis are power spectra of AC-GVSF and eigenvalue plots with respect to amount of AC-GVSF. These are highly subjective approaches. People may interpret what well-determined eigenvalues are differently, and thus construct the truncated eigenvector matrix differently. The power spectra approach also requires the user to investigate many figures. Even if this is done, there is no guarantee that this yields the 'correct' result.

This thesis introduced the limitation of edge-effects in regional predictions of Australia. I think most regional models are prone to issues like this. What is also true is that one may remove an outer ring of the spherical cap to circumvent this effect. I think this shows great promise, as computing models e.g. with a 20° opening angle instead of e.g. a 15° opening angle is not so much more computationally demanding.

There are methods of estimating the optimum number of AC-GVSF to use in a model, introduced in Plattner and Simons (2017). This incorporates a bias, variance and mean-square error of a large set of models, and is thus very comprehensive. I have found that estimating a number of J from an eigenvalue plot and adjusting the model from this estimate is an efficient way of producing promising, high resolution regional models of the Earth's lithospheric magnetic field.

7.6 A Note on the Change in Project Plan

The project plan was revised during the project period. The original project plan is shown in Appendix A, while the revised plan is shown in Section 1.1. The regions of interest were changed because we believed models at high latitude were of high importance, to show the capabilities of the Python toolbox at various latitudes. Furthermore, the LCS-1 model did not perform well over Greenland, so it was worthwhile investigating this region with a different approach. The reduction

in regions of interest was due to redundancy in regions at certain latitudes, as well as type of region.

Chapter 8

Conclusion

This work has introduced the Slepian approach to regional modelling implemented as a Python toolbox for computing regional models of the lithospheric magnetic field. I construct a localized kernel matrix using globally defined spherical harmonic functions linearly combined for regional optimization. Furthermore, I obtain AC-GVSF from the kernel matrix to construct localized design matrices used in potential field modelling to obtain Slepian model parameters.

The Slepian approach to regional modelling was benchmark tested over the Bangui Anomaly which suggested that the toolbox is capable of reproducing a synthetic test case reliably.

Utilizing data from the CHAMP and *Swarm* missions with selection criteria described in Olsen et al. (2017), I employed the Python toolbox to construct several models over regions with geophysical interest on the Earth. These regions include: a large amplitude anomaly at low latitudes with the Bangui Anomaly, a mid-latitude thoroughly investigated region with Australia, an oceanic region with the Walvis Ridge, and a high-latitude region of high scientific interest with Greenland.

I implemented L_1-norm and L_2-norm regularization to handle instabilities at high spherical harmonic degrees and complex noise in data. This was successfully implemented and regional models were constructed for all regions of interest. I consulted diagnostic tools such as Mauersberger-Lowes power spectra and the power spectra of AC-GVSF to determine the proper Slepian truncation parameter, J, to use.

The resulting models showed great potential for the Python toolbox when predicting the lithospheric field at high latitudes. It showed great agreement with the EMM2015 model, even though the EMM2015 model utilizes aeromagnetic data (lower altitude data) where the regional model utilizes satellite data only. This introduces great opportunities for modelling the lithospheric field at higher latitudes using this Python toolbox, if aeromagnetic data is utilized in toolbox as well. This is a strength of the Python toolbox, that aeromagnetic data is applicable in addition to satellite data. Furthermore, as the *Swarm* satellites descend, finer scale features will be better resolved using satellite data.

A common issue with regional models is the edge effect, which was also present in this analysis e.g. over Australia. With the Slepian approach to regional modelling this problem can be circumvented by expanding the spherical cap. One may remove unwanted contributions by cropping out the newly added expansion of the resulting lithospheric magnetic field map to obtain only well-resolved

maps. If one constructs a sufficiently large spherical cap, the target of interest will not be affected by the cropping.

The regional model of the Bangui Anomaly did show larger amplitudes than the LCS-1 model as well as suggesting a more detailed structure. I believe improvements can be made to the model, but the current state of the Python toolbox can definitely be considered a proof of concept.

Over Australia, the Slepian approach to regional modelling performs well, but there are room for improvement, as edge-effects were seen. However, compared to the EMM2015 model the regional approach shows great promise.

The oceanic region over the Walvis Ridge did not show improvement over the LCS-1 model, suggesting further work in this area is required.

Greenland provided an excellent benchmark for this Python toolbox as it is a high latitude region because the poor representation of these regions in global models. The Slepian approach to regional modelling produced excellent models using only radial components of gradient satellite data. Compared to the LCS-1 model which is in fact a state-of-the-art global model derived from satellite gradient data, this regional approach performed better. The EMM2015 model utilizes satellite and aeromagnetic data where possible. The regional model agrees very well with this global model.

On a final note, while this has been a first look into the possibilities of the Slepian approach to regional modelling implemented as a Python toolbox, and I believe this method shows great promise and has a bright future in providing the scientific community with high-resolution regional models.

8.1 Outlook

Due to the time limitation of this thesis there are many changes to the developed toolbox that have not been finished. These include

- Optimize the synthetic model scripts.
- Incorporate use of various types of data.
- Optimize user-friendly aspects of the toolbox, such as more options to construct more specific models.

The results of this thesis are very encouraging, and I think further investigation with different data selection criteria and perhaps mixing aeromagnetic data with satellite data when possible would provide even more promising details of the lithospheric magnetic field.

The Slepian approach is not limited to modelling the Earth's magnetic field alone. Because it is a potential field approach, one can easily construct models over other planetary bodies or moons. E.g. the Martian surface has been investigated in Plattner and Simons (2015) using satellite data. The Moon is an example of an interesting place for lithospheric field modelling, which the Slepian approach to regional modelling can be applied to.

References

Alain Plattner. 2017. Slepian. https://github.com/Slepian/Slepian/wiki.

Richard C. Aster, Brian Borchers, and Cllifford H. Thurber. 2013. *Parameter Estimation and Inverse Problems*, 2 edition. Elsevier.

John R Baumgardner and Paul O Frederickson. 1985. Icosahedral discretization of the two-sphere. *SIAM Journal on Numerical Analysis*, 22(6):1107–1115.

Richard J Blakely. 1996. *Potential theory in gravity and magnetic applications*. Cambridge university press.

Chris Finlay. 2017. Lecture notes: 30760 Inverse Problems.

C. G. Constable. 1988. Parameter estimation in non-Gaussian noise. *Geophysical Journal*, 94:131–142.

F. A. Dahlen and J. Tromp. 1998. *Theoretical Global Seismology*. Princeton University Press.

Colin G Farquharson and Douglas W Oldenburg. 1998. Non-linear inversion using general measures of data misfit and model structure. *Geophysical Journal International*, 134(1):213–227.

Christopher C Finlay, Nils Olsen, Stavros Kotsiaros, Nicolas Gillet, and Lars Tøffner-Clausen. 2016. Recent geomagnetic secular variation from swarm and ground observatories as estimated in the chaos-6 geomagnetic field model. *Earth, Planets and Space*, 68(1):112.

W. Freeden and M. Schreiner. 2009. *Spherical Functions of Mathematical Geosciences: A Scalar, Vectorial, and Tensorial Setup*. Springer.

Eigil Friis-Christensen, H Lühr, and Gauthier Hulot. 2006. Swarm: A constellation to study the earth's magnetic field. *Earth, planets and space*, 58(4):351–358.

David Gubbins. 2004. *Time Series Analysis and Inverse Theory for Geophysicists*. Cambridge University Press.

Francis Begnaud Hildebrand. 1987. *Introduction to numerical analysis*. Courier Corporation.

Peter J Huber. 1996. *Robust statistical procedures*, volume 68. Siam.

Livia Kother. 2017. *Lithospheric field modelling based on equivalent point sources*. PhD dissertation, Technical University of Denmark.

Livia Kother, Magnus D Hammer, Christopher C Finlay, and Nils Olsen. 2015. An equivalent source method for modelling the global lithospheric magnetic field. *Geophysical Journal International*, 203(1):553–566.

R. A. Langel. 1987. The main field. In J. A. Jacobs, editor, *Geomagnetism*, volume 1, pages 249–512. Academic Press, London.

FJ Lowes. 1966. Mean-square values on sphere of spherical harmonic vector fields. *Journal of Geophysical Research*, 71(8):2179–2179.

Yasmina M Martos, Tom A Jordan, Manuel Catalan, Thomas M Jordan, Jonathan L Bamber, and David G Vaughan. 2018. Geothermal heat flux reveals the iceland hotspot track underneath greenland. *Geophysical Research Letters*, 45(16):8214–8222.

P Mauersberger. 1956. Das mittel der energiedichte des geomagnetischen hauptfeldes an der erdoberflache und seine saulare anderung. *Gerlands Beitr. Geophys.*, 65:207–215.

S Maus, Y Fan, C Manoj, M Rother, J Rauberg, C Stolle, and H Luhr. 2007. Sixth generation lithospheric magnetic field model, mf6, from champ satellite magnetic measurements. In *AGU Fall Meeting Abstracts*.

Stefan Maus. 2007. Champ. In *Encyclopedia of geomagnetism and paleomagnetism*, pages 59–60. Springer.

Stefan Maus. 2008. The geomagnetic power spectrum. *Geophysical Journal International*, 174(1):135–142.

Nils Olsen, Dhananjay Ravat, Christopher C. Finlay, and Livia K. Kother. 2017. LCS-1: a high-resolution global model of the lithospheric magnetic field derived from champ and *Swarm* satellite observations. *Geophysical Journal International*, 211:1461–1477.

Nils Olsen, Claudia Stolle, Rune Floberghagen, Gauthier Hulot, and Alexey Kuvshinov. 2016. Special issue "swarm science results after 2 years in space". *Earth, Planets and Space*, 68(1):172.

Alain Plattner and Frederik J. Simons. 2014. Spatiospectral concentration of vector fields on a sphere. *Appl. Comput. Harmon. Anal.*, 36:1–22.

Alain Plattner and Frederik J. Simons. 2015. High-resolution local magnetic field models for the Martian South Pole from Mars Global Surveyor data. *J. Geophys. Res. Planets*, 120:1543–1566.

Alain Plattner and Frederik J. Simons. 2017. Internal and external potential-field estimation from regional vector data at varying satellite altitude. *Geophysical Journal International*, 211:207–238.

Rasmus R. Joost. 2018. Development and Validation of a Slepian Function Approach to Regional Magnetic Feld Modelling. Synthesis Project.

Erwan Thébault. 2006. Global lithospheric magnetic field modelling by successive regional analysis. *Earth, planets and space*, 58(4):485–495.

Erwan Thébault, Pierre Vigneron, Benoit Langlais, and Gauthier Hulot. 2016. A swarm lithospheric magnetic field model to sh degree 80. *Earth, Planets and Space*, 68(1):126.

Appendix A

Original Project Plan

Thesis Project Plan

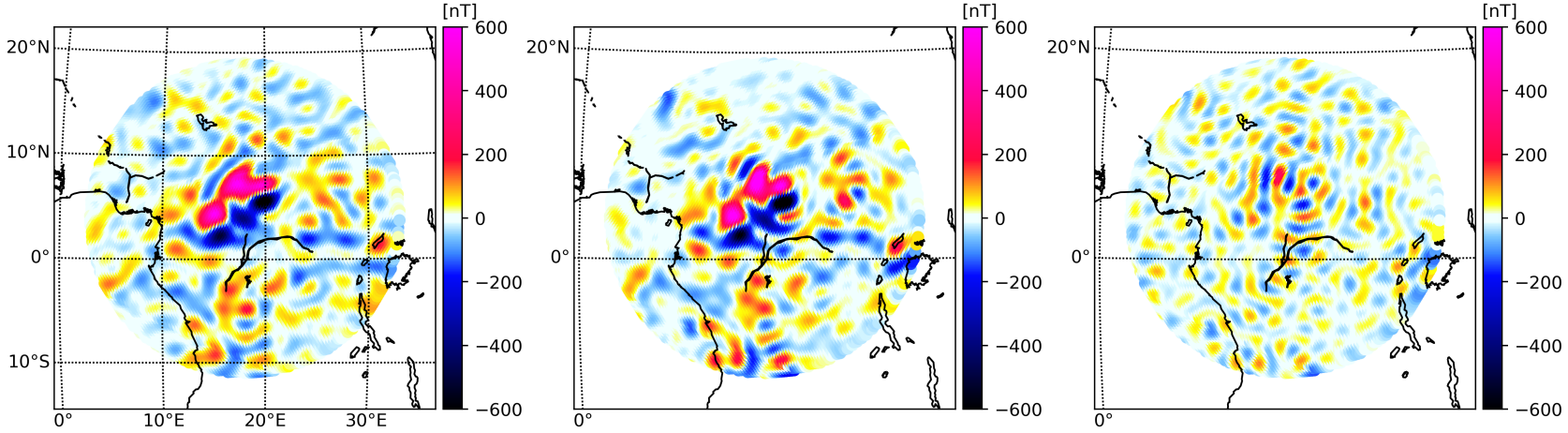
This document serves as the project plan of the master's thesis of Rasmus Joost with supervisor Chris Finlay. Major milestones will be presented in the following list, and sub-goals will be nested within each milestone. The milestones presented reflect the learning objectives for the master thesis found at http://sdb.dtu.dk/2018/35/703#Master's_thesis. These are initial milestones and are thus subject to change as the project evolves.

1. Literature study. Find relevant articles outside those provided by the supervisor.
2. Optimize the Slepian toolbox built in the synthesis project leading up to this thesis such that the following is achieved:
   (a) Computation optimization. Currently, a design matrix for spherical harmonic degree 50 requires more than 60 minutes of computation time. The optimization lies in determination of associated Legendre functions used in computing the kernel matrix.
   (b) Preparation for real data. The toolbox must be prepared to handle satellite 'gradient' data from *Swarm* satellites *Alpha* and *Charlie* (East-West gradient, across-track), and CHAMP satellite (North-South gradient, along-track).
   (c) Finalize and document toolbox with the aim of a GitHub release.
3. Perform studies of several regions of interest. The study will initially be carried out with spherical harmonic degree 200. Regions include:
   (a) Bay of Bengal.
   (b) East African Rift.
   (c) Bangui Anomaly (central Africa).
   (d) Walvis Ridge and surrounding younger parts of the South Atlantic.
   (e) Australia.
4. Compare results with other models such as the LCS-1 model.
5. Write up thesis.

Appendix B

Results

B.1 Bangui Anomaly



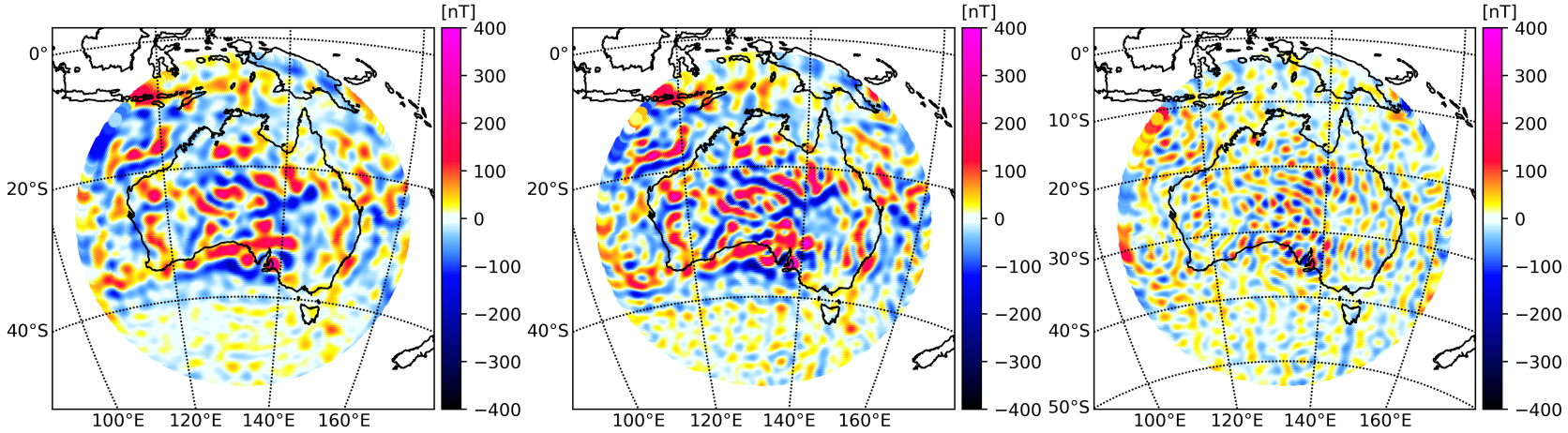
(a) L_2-norm regional model prediction of the radial field at the Earth's mean spherical radius.

(b) LCS-1 prediction of the radial field at the Earth's mean spherical radius.

(c) Difference map of the radial component of the LCS-1 model and the L_2-norm regularized regional model.

Figure B.1: Radial components of the regional L_2-norm prediction and the LCS-1 prediction over the Bangui anomaly together with a difference map between the to predictions.

B.2 Australia

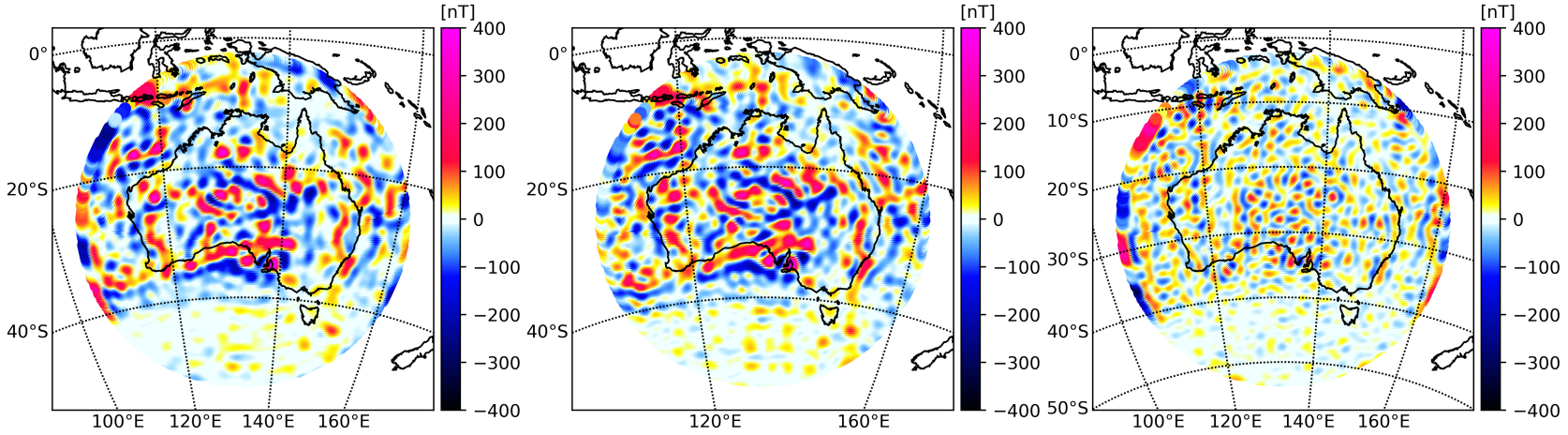


(a) L_2-norm regional model prediction of the radial field at the Earth's mean spherical radius.

(b) EMM2015 prediction of the radial field at the Earth's mean spherical radius.

(c) Difference map of the radial component of the EMM2015 model and the L_2-norm regularized regional model.

Figure B.2: Radial components of the regional L_2-norm prediction and the EMM2015 prediction over Australia together with a difference map between the to predictions.

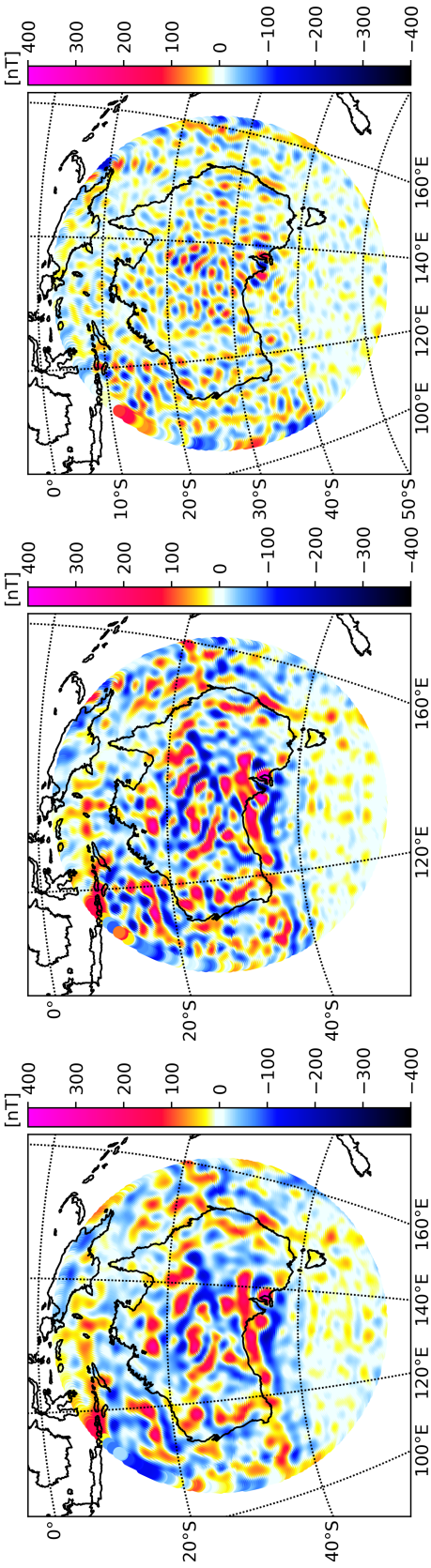


(a) L_1-norm regional model prediction of the radial field at the Earth's mean spherical radius.

(b) LCS-1 prediction of the radial field at the Earth's mean spherical radius.

(c) Difference map of the radial component of the LCS-1 model and the L_1-norm regularized regional model.

Figure B.3: Radial components of the regional L_1-norm prediction and the LCS-1 prediction over Australia together with a difference map between the to predictions.



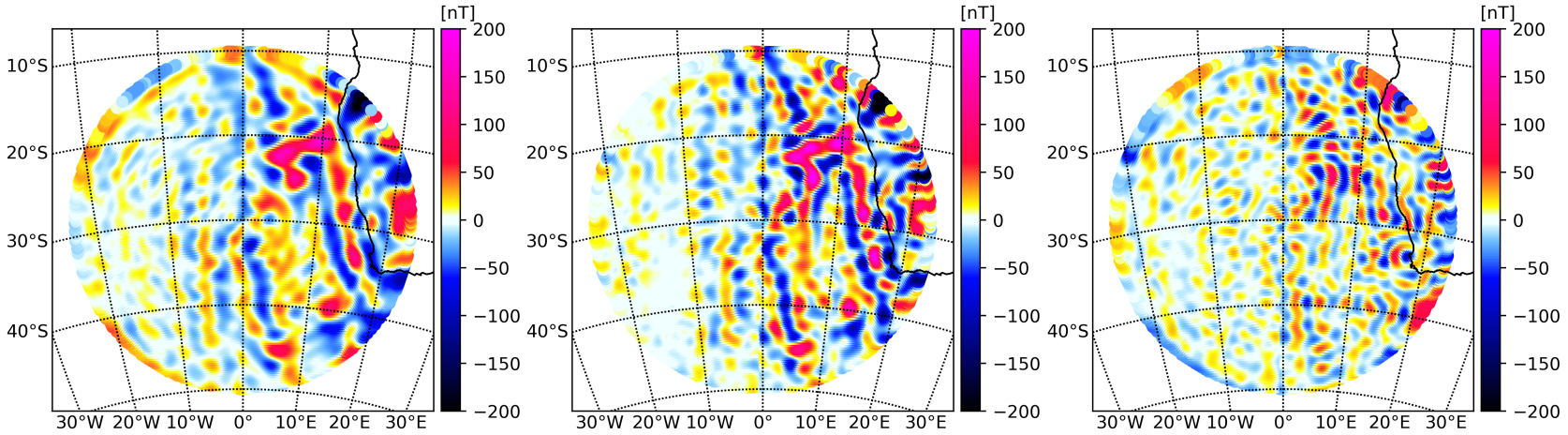
(a) L_2-norm regional model prediction of the radial field at the Earth's mean spherical radius.

(b) LCS-1 prediction of the radial field at the Earth's mean spherical radius.

(c) Difference map of the radial component of the LCS-1 model and the L_2-norm regularized regional model.

Figure B.4: Radial components of the regional L_2-norm prediction and the LCS-1 prediction over Australia together with a difference map between the to predictions.

B.3 Walvis Ridge



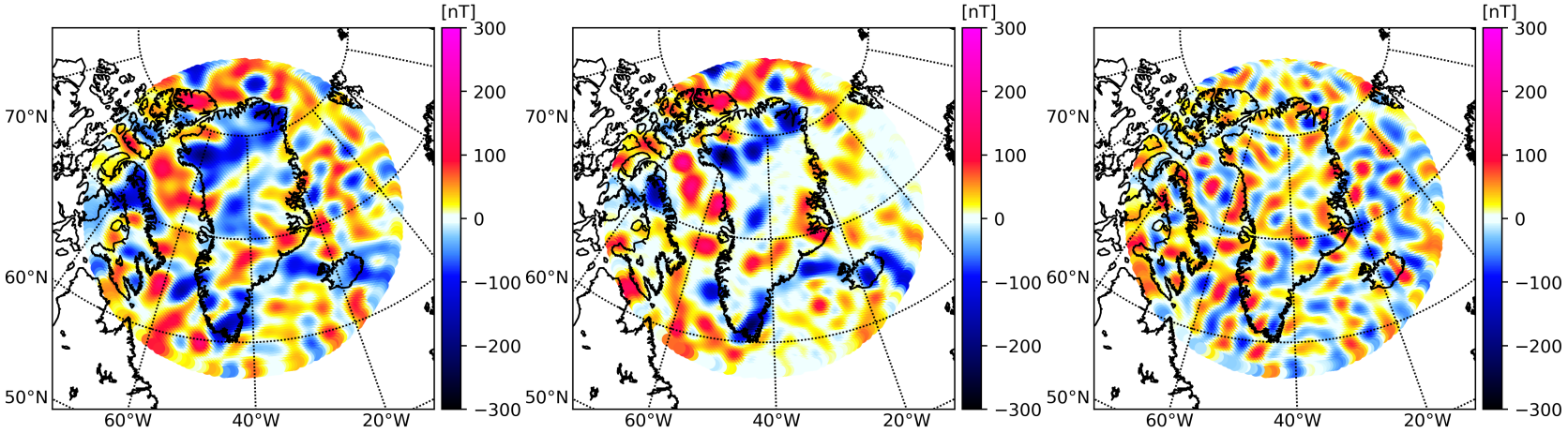
(a) L_2-norm regional model prediction of the radial field at the Earth's mean spherical radius.

(b) LCS-1 prediction of the radial field at the Earth's mean spherical radius.

(c) Difference map of the radial component of the LCS-1 model and the L_2-norm regularized regional model.

Figure B.5: Radial components of the regional L_2-norm prediction and the LCS-1 prediction over the Walvis Ridge together with a difference map between the to predictions.

B.4 Greenland

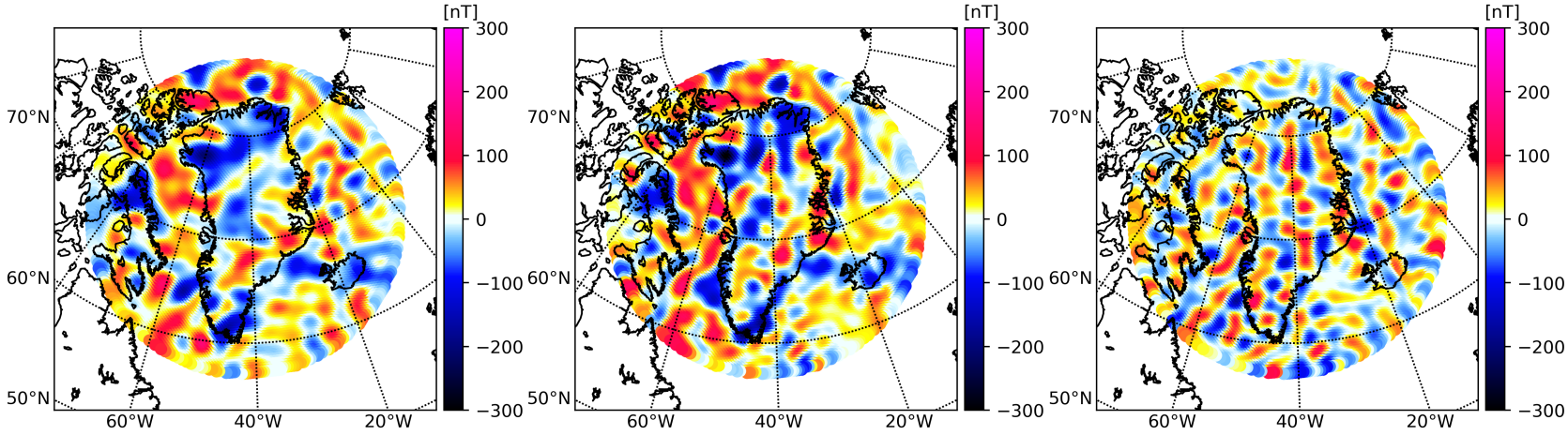


(a) L_2-norm regional model prediction of the radial field at the Earth's mean spherical radius.

(b) LCS-1 prediction of the radial field at the Earth's mean spherical radius.

(c) Difference map of the radial component of the LCS-1 model and the L_2-norm regularized regional model.

Figure B.6: Radial components of the regional L_2-norm prediction and the LCS-1 prediction over Greenland together with a difference map between the to predictions.



(a) L_2-norm regional model prediction of the radial field at the Earth's mean spherical radius.

(b) EMM2015 prediction of the radial field at the Earth's mean spherical radius.

(c) Difference map of the radial component of the EMM2015 model and the L_2-norm regularized regional model.

Figure B.7: Radial components of the regional L_2-norm prediction and the EMM2015 prediction over the Walvis Ridge together with a difference map between the to predictions.

Appendix C

Computation of Kernel and Design Matrices

C.1 Kernel Matrix

Listing C.1: Code implementation for computing the kernel matrix.

```
1  import time
2  import numpy as np
3  from lib.tools import legendre as leg
4  from lib.tools import GMT_tools as gmt
5  from lib.tools import toolfunctions as toft
6
7  def kernelK(rcap, r, rref, Lmax, m):
8      ''' K = kernelK(rcap,r,rref,Lmax,m)
9
10     Computes a localized kernel matrix for a Polar cap of radius rcap.
11     Possibilities of placing the kernel at various altitudes are
           implemented
12     with r and rref, where rref is a reference radius, e.g. Earth's
           radius.
13     The kernel matrix is constructed for a particular harmonic order, m
           .
14
15     The kernel matrix can be aribitrarily placed at other coordinates.
16     This is done in rotateglm.
17
18     Computations are performed with respect to Plattner and Simons 2017
19     Appendix A1.
20
21     INPUT:
22         rcap        Opening angle of spherical cap in radians.
23         r           Continuation radius, e.g. satellite radius.
24         rref        Reference radius, e.g. planetary radius -
25                     if r=rref, kernel is placed at rref.
```

```
26          Lmax          Spherical harmonic degree of the kernel to be
                constructed.
27          m             Spherical harmonic order, 0 >=m >= L.
28
29      OUTPUT:
30          K             Localized kernel for the Polar cap.'''
31      t = time.time() # Keep track of computation time
32      ## Obtain Blm kernel
33      # Determine amount of nodes used in Gauss Legendre quadrature
34      glnodes = max(200,2*Lmax + 1)
35
36      # Assign zero as North Pole
37      npole = 0
38      x0 = np.cos(rcap) # Opening angle of the spherical cap
39      x1 = np.cos(npole)
40
41      # Consider absolute value of m only
42      m = np.abs(m)
43
44      ## Compute B kernel in accordance with Plattner & Simons 2017 Eq.
            A4
45      if Lmax == 0: # L = 0 is purely radial, so we consider only L > 0
46          Blm = []
47      else:
48          Lmin = max(1, m) # Minimum L is at least 1. Maximum is already
                given!
49          length = Lmax - Lmin + 1
50
51          # Preallocate space
52          Blm = np.zeros((length, length))
53
54          # Avoid aliasing by checking the nyqvist degree
55          if glnodes < (Lmax + 1) and glnodes != 0:
56              raise Warning("Sample finer to avoid aliasing.")
57
58          # Gauss Legendre quadrature integration interval
59          intv = np.array((x0,x1))
60
61          x, w, N = leg.glquadinit(glnodes,intv)
62          #t1 = time.time()
63          # Rewrite absicasses to colattitudes (in rad)
64          xacos = np.arccos(x)
65          Xlm = np.zeros((N, length))  # Preallocate space for Xlm and
                dXlm
66          dXlm = np.zeros((N, length))
67          for i in np.arange(length): # Compute the Xlm and dXlm by Pnm
                and dPnm
68              l = Lmin + i
```

```
69              Pnm = gmt.get_Pnm(l, xacos) # Obtain evaluated Legendre
                    functions
70              dPnm = Pnm[m][l+1] # ... and their derivatives
71
72              Xlm[:, i] = (Pnm[l][m] * np.sqrt(2*l + 1))/(np.sqrt(2 - (m
                    == 0)))
73              dXlm[:, i] = (dPnm * np.sqrt(2*l + 1))/(np.sqrt(2 - (m ==
                    0)))
74          # Calculate Gauss Legendre products, i.e. apply Xlm and dXlm in
                Eq. A4
75          # of Plattner & Simons 2017
76          Bprodint = np.zeros((N,int((length**2 + length)/2))) #
                Preallocate
77          idx = 0 # Indexing
78          for i in np.arange(length): # i will index l in Xlm and dXlm
79              for j in np.arange(i,length): # j will index l' in Xlm and
                    dXlm
80                  l = Lmin + i
81                  lprime = Lmin + j
82                  Bprodint[:,idx] = 2 * np.pi / np.sqrt(l*(l + 1)* \
83                                    lprime*(lprime + 1)) * \
84                                    (dXlm[:,i]*dXlm[:,j] + m / \
85                                    np.sin(np.arccos(x))*Xlm[:,i]*\
86                                    m / np.sin(np.arccos(x))*Xlm[:,j])
87                  idx += 1
88          Bw = Bprodint.T @ w # Apply weights
89          idx = 0 # Transform Blm to upper triangle matrix
90          for i in np.arange(length+1):
91              for j in np.arange(i,length):
92                  Blm[i,j] = Bw[idx]
93                  idx += 1
94          # Fill the lower triangle of matrix
95          Blm = Blm + Blm.T - np.diag(np.diag(Blm))
96          Blm = Blm/4/np.pi # Scale Blm
97          #print("Blm constructed for m = %i, it took %f seconds"\
98          #       %(m,time.time()-t1))
99          if m == 0: # Increase size of Blm if m = 0, to fit Plm
100             Bzero = np.zeros((Lmax + 1,Lmax + 1))
101             Bzero[1:,1:] = Blm.copy()
102             Blm = Bzero.copy()
103     ## Obtain Plm kernel
104     # Preallocate space
105     Plm = np.empty((Lmax + 1 - max(m,0), Lmax + 1 - max(m,0)))
106     Lmin = max(m,0) # Ensure L >= m at all times
107     intv = np.array((np.cos(rcap),1)) # Integration interval
108     #t2 = time.time()
109     Nglprev=0
110     for L1 in np.arange(Lmin, Lmax + 1):
```

```
111         for L2 in np.arange(L1, Lmax + 1):
112             # Obtain initiating values for Gauss Legendre quadrature
113             glnodes=max(L1+L2, 200*(m*m != 0))
114             xgl, wgl, Ngl = leg.glquadinit(glnodes, intv = None)
115             # Rescale xgl, this can be achieved in glquadinit, but
                    weights must
116             # remain unscaled to ensure wgl.sum() = 2
117             xgl = intv[0] + (xgl + 1)/2*(np.diff(intv))
118             xacos = np.arccos(xgl) # Transform absicasses to colatitude
119             if Nglprev != Ngl:
120                 Pnm = gmt.get_Pm(Lmax,m,xacos)
121             integrand = Pnm[L1] * Pnm[L2]
122             # Finalize computation and obtain the Gauss Legendre
                    quadrature
123             glquad = (wgl @ integrand)*np.diff(intv)/2
124             # Insert Plm entries
125             Plm[L1 - Lmin, L2 - Lmin] = glquad * np.sqrt(2*L1 + 1) * \
126                                         np.sqrt(2*L2 + 1) / \
127                                         (4*np.pi)*np.pi*(1+(m==0))
128             # Plm is symmetric
129             Plm[L2 - Lmin, L1 - Lmin] = Plm[L1 - Lmin, L2 - Lmin]
130             Nglprev=Ngl
131     #print("Plm constructed for m = %i, it took %f seconds"%(m,time.
            time()-t2))
132     bigl = np.arange(Lmin,Lmax+1)
133
134     # Scaling factors for combining Blm and Plm to make Elm integraded
            product
135     facPmat = np.diag(np.sqrt((bigl + 1) / (2*bigl + 1)))
136     facBmat = np.diag(-np.sqrt(bigl / (2*bigl + 1)))
137
138     K = facPmat @ Plm @ facPmat.T + facBmat @ Blm @ facBmat.T
139
140     ## Continue kernel to desired level
141     bigl = np.arange(Lmin,Lmax + 1)
142     # Calculate BKB' = (B(BK)')'
143     K = kernelcontinue(K, Lmax, r, rref, bigl) # BK
144     K = K.T # (BK)'
145     K = kernelcontinue(K, Lmax, r, rref, bigl) # B(BK)'
146     K = K.T # (B(BK)')'
147     K = (K + K.T)/2 # Avoid numerical dissymmetrification
148     print("K constructed for m = %i/%i in %f seconds"
149           %(m, Lmax, time.time() - t))
150
151     # Output
152     return K
```

C.2 Design matrix

Listing C.2: Code implementation for computing design matrices.

```
def designeval(G, phi, theta, radi, rref, onorout=1):
    ''' Geval = designeval(G, phi, theta, radi, rref, onorout)

    Evaluates the coefficients computed in glmalpha or rotateglm (i.e. G
    matrix), at data locations. This builds design matrices at actual
    satellite altitudes to extract model parameters, and at a planetary
    surface to produce a model.

    INPUT:
        G           Matrix of AC-GVSF's obtained by glmalpha or
                    rotateglm.
        phi         Data longitudes.
        theta       Data colatitudes.
        radi        Data radii, e.g. satellite radii, or an array of
                    planetary
                    radii with the same size as phi and theta.
        rref        Reference radius, e.g. planetary radius.
        onorout     Coefficient format, addmon or addmout.
                    1: addmout (default)
                    0: addmon.

    OUTPUT:
        Geval       Matrix of Slepian basis evaluated at data locations
                    .
                    Dimensions are J x 3k, where k=len(phi). '''
    # Determine maximum spherical harmonic degree
    Lmax = int(np.sqrt(len(G)) - 1)
    # Length of data
    dlen = len(theta)
    # Transform input coefficients
    if not onorout:
        _,_,_,_,_,_,_,_,_,rinm = toft.addmon(Lmax)
        G = G[rinm,:]

    # Preallocate space
    Geval = np.zeros((G.shape[1],3*dlen))

    # Phase shift longitude
    phi = phi + np.pi
    divsinvals = 1/np.sin(theta)
    # L=0 is done separately, as for L=0, Elm = Ylm
    L = 0
    m = 0
```

```
41      # Obtain Xlm for L, m = 0, 0
42      Xlm = np.sqrt(2*L+1)/np.sqrt(4*np.pi)*gmt.get_Pnm(Lmax,theta)[L,m
            ,:]
43      # Continuation of Xlm
44      Xlm *= ( (-L-1)/rref*(radi/rref)**(-L-2) )
45      # Longitudinal phase. For L = 0, this is simply an array of ones
46      P = np.ones(phi.shape)
47      Geval[:,:dlen] = G[L**2:(L+1)**2,:].T * np.r_[Xlm*P]
48
49      # Iterate over the remaining L's
50      for L in np.arange(1,Lmax+1):
51          m = np.arange(-L,L+1)
52          Pnm = gmt.get_Pnm(L,theta) # Legendre functions
53          dPnm = Pnm[abs(m), L+1] # ... and their derivatives
54          # Compute Xlm and dXlm
55          Xlm = np.sqrt(2*L + 1) / np.sqrt(4*np.pi) * \
56                Pnm[L,abs(m),:] * np.c_[(-1)**abs(m)/np.sqrt(2-(m==0))]
57          dXlm = np.sqrt(2*L + 1) / np.sqrt(4*np.pi) * \
58                 dPnm * np.c_[(-1)**abs(m)/ np.sqrt(2 - (m==0))]
59          # Longitudinal phase
60          P = np.cos(np.c_[m]*np.r_[phi] - \
61                     np.pi/2*np.c_[m>0]*np.ones((1,len(phi)))) \
62                     * np.c_[np.sqrt(2 - (m==0))]
63          dP = -np.sin(np.c_[m]*np.r_[phi] - \
64                       np.pi/2*np.c_[m>0]*np.ones((1,len(phi)))) \
65                       * np.c_[np.sqrt(2-(m==0))*m]
66
67          # Radial factor
68          Rfac = np.tile((1/rref*(radi/rref)**(-L-2)),(len(m),1))
69
70          # Now perform a renomalization for E
71          # Erad = (-L-2)*X*P
72          # Etheta = dX*P
73          # Ephi = 1/sin(theta)*X*dP
74          #
75          # Radial factor is included in all the cases
76
77          # Sum over all degrees
78          # Radial component
79          Geval[:,:dlen] = Geval[:,:dlen] + \
80                           G[L**2:(L+1)**2,:].T @ ((-L-1)*Rfac*Xlm*P)
81          # Colatitudinal component
82          Geval[:,dlen:2*dlen] = Geval[:,dlen:2*dlen] + \
83                                 G[L**2:(L+1)**2,:].T @ (Rfac*dXlm*P)
84          # Longitudinal component
85          Geval[:,2*dlen:] = Geval[:,2*dlen:] + \
86                             G[L**2:(L+1)**2,:].T @ (Rfac* \
87                             np.tile(divsinvals,(len(m),1))*Xlm*dP)
```

```
88
89      # Output
90      return Geval
```

Appendix D

Gauss-Legendre Quadrature

```
# -*- coding: utf-8 -*-
"""
Legendre toolbox for the Python Slepian tool.
The toolbox is written by Rasmus R. Joost,
based largely on the work of Alain Plattner
and Frederik Simons in the Slepian toolbox
for Matlab: https://github.com/Slepian/Slepian/wiki
"""
import numpy as np
from lib.tools import GMT_tools as gmt

def roots(Lmax):
    ''' r,Jac = roots(Lmax)

    Computes Legendre polynomial of degree Lmax roots. The tri-diagonal
    Jacobian matrix is constructed and eigenvalues (which are the roots)
    are extracted.

    INPUT:
        Lmax        Legendre polynomial degree.

    OUTPUT:
        r           roots.
        Jac         Jacobian matrix. '''
    # Indice array
    n = np.arange(1,Lmax)
    # subdiagonal array
    d = n / np.sqrt(4 * n**2 - 1)
    # Jacobian matrix, shift subdiagonals +1 and -1 w.r.t diagonal
    Jac = np.diag(d,k=1) + np.diag(d,k=-1)
    # Obtain eigenvalues of Jacobian matrix
    r,_ = np.linalg.eigh(Jac,UPLO='U')

    # Output
```

```
35      return r, Jac
36
37  def glquadinit(l,intv):
38      ''' x,w,N = glquadinit(l,intv)
39
40      For a polynomial degree l, the absicasses x, weights w, and points used
41      in integration, N, are computed.
42      The integration interval, intv, must be defined between [-1,1].
43      It can either be two values or a 2D array of varying intervals.
44
45      INPUT:
46          l       Polynomial degree, e.g. Legendre polynomial degree.
47          intv    Integration interval.
48
49      OUTPUT:
50          x       Absicasses of Legendre-Gauss quadrature.
51          w       Weights of Legendre-Gauss quadrature.
52          N       Points used in Legendre-Gauss quadrature.'''
53      N = int(np.ceil((l + 1) / 2)) # Determine amount of legendre functions
54      # Obtain absicasses for Gaussian quadrature
55      x, _ = roots(N)
56
57      if N > 1: # Obtain Legendre functions, avoid issues at N=0
58          Pm = gmt.get_Pm(N-1,0,np.arccos(x))
59      else:
60          Pm = gmt.get_Pm(N+(N==0),0,np.arccos(x))
61      # Get weights using legendre polynomial of degree N-1
62      Pl = Pm[N-1,:]
63      Pldiff = -N * Pl / ( x**2 - 1)
64      # Weights
65      w = 2 / (1 - x**2) / Pldiff**2
66
67      if intv is not None:
68          if np.size(intv) == 2: # Interval are two scalars
69              a = intv[0]
70              b = intv[1]
71              # Rescale absicasses
72              x = a + (x + 1)/2*(b - a)
73              # Rescale weights
74              w = w * (b - a)/2
75          else: # Intervals are arrays.
76              a = intv[0,:]
77              b = intv[1,:]
78
79              xtmp, wtmp = x, w
80              x = np.zeros((len(xtmp),len(a)))
81              w = np.zeros((len(wtmp),len(b)))
82
```

```
83              for idx in np.arange(len(a)):
84                  x[:,idx] = a[idx] + (xtmp + 1)/2*(b[idx] - a[idx])
85                  w[:,idx] = wtmp * (b[idx] - a[idx])/2
86
87      # Output
88      return x, w, N
```

Appendix E

Rotating the Kernel Matrix

```
1  def rotglmalpha(lmcosi,alpha,beta,gamma,rcsh):
2      ''' lmcosirot = plm2rot(lmcosi,alpha,beta,gamma,rcsh)
3
4      Rotates spherical harmonic coefficients on the unit sphere using
5      Euler angles in an active rotation convention (Dahlen and Tromp
           1998 pages
6      920-924, the inverse of that). The active rotations are achieved by
7      a series of passive rotations. These are related by flipping the
           signs
8      of Equations of Appendix C.8 in Dahlen and Tromp (1998).
9      Yields coefficients of rotated new field in the non-rotated
           coordinate
10     system.
11
12     Euler angles used must be in degrees, and alpha (0<360), beta
           (0<180),
13     and gamma (0<360).
14
15     INPUT:
16         lmcosi       [l m cos sin] matrix with order m>=0.
17         alpha,       Euler angles in degrees. Rotations are over
18         beta,        alpha around z, increasing from y to x, then
19         gamma        beta around old y, increasing from x to z, then
20                      gamma around old z, increasing from y to x.
21
22         rcsh         1: coefficients belong to real harmonics,
23                      0: coefficients belong to complex harmonics.
24
25
26     OUTPUT:
27         lmcosirot    [l m cosrot sinrot], i.e. matrix with rotated
               cosine
28                      and sine coefficients.'''
29     # Preallocate space
```

```
30      Crot = np.zeros(len(lmcosi[:,0]))
31      Srot = np.zeros(len(lmcosi[:,0]))
32
33      if alpha == 0 and beta == 0 and gamma == 0: # No rotation if angles
            are zero
34          lmcosirot = lmcosi.copy()
35          return lmcosirot
36      else:
37          # Convert rotation coordinates to radians
38          alpha = alpha*np.pi/180
39          beta = beta*np.pi/180
40          gamma = gamma*np.pi/180
41
42          # Find maximum spherical harmonic degree
43          Lmax = int(max(lmcosi[:,0]))
44          lmcosirot = lmcosi.copy()
45
46          # Obtain Wigner-D rotation matrix for spherical harmonics
47          D,_ = dlmb(Lmax)
48
49          if rcsh == 1:
50              lmcosi = r2c(lmcosi)
51
52          # Passive rotation over alpha - pi/2
53          C, S = rotcof(lmcosi[:,2],lmcosi[:,3], alpha-np.pi/2)
54
55          # Loop over all degrees
56          Cbeta = np.cos(np.arange(0,Lmax+1)*beta)
57          Sbeta = np.sin(np.arange(0,Lmax+1)*beta)
58          for l in np.arange(0,Lmax+1):
59              # Constuct alternating matrices with twos and zeros
60              i,j = np.meshgrid(np.arange(0,l+1),np.arange(0,l+1))
61              Czeros = np.mod(i+j+np.mod(l+1,2),2)*2
62              Czeros[:,0] = 1
63              Szeros = np.mod(i+j+np.mod(l,2),2)*2
64              Szeros[:,0] = 1
65
66              # Passive rotation over -pi/2
67              Ccos,_,_ = shcos(C,l)
68              Ssin,_,_ = shsin(S,l)
69
70              Crotneg = D[l].T * Czeros @ Ccos
71              Srotneg = D[l].T * Szeros @ Ssin
72
73              # Passive azimuthal rotation over beta
74
75              Crotbeta = Crotneg * Cbeta[0:l+1] + Srotneg * Sbeta[0:l+1]
76              Srotbeta = Srotneg * Cbeta[0:l+1] - Crotneg * Sbeta[0:l+1]
```

```
77
78              # Passive rotation over pi/2
79              Crotpos = np.dot(D[l] * Czeros, Crotbeta)
80              Srotpos = np.dot(D[l] * Szeros, Srotbeta)
81
82              # Fill d
83              lo = int(addmup(l-1,drk=None))
84              hi = int(addmup(l,drk=None))
85              Crot[lo:hi] = Crotpos.copy()
86              Srot[lo:hi] = Srotpos.copy()
87
88          # Passive azimuthal rotation over gamma + pi/2
89          Crotgam, Srotgam = rotcof(Crot,Srot,gamma+np.pi/2)
90          lmcosirot[:,2] = Crotgam.copy()
91          lmcosirot[:,3] = Srotgam.copy()
92
93          if rcsh == 1:
94              lmcosirot = c2r(lmcosirot)
95
96          # Output
97          return lmcosirot
98
99  def dlmb(L):
100     ''' D, d = dlmb(L)
101
102     Compute matrix elements for spherical harmonic polar rotation
            around
103     the y-axis over 90 degrees. The rotation is split into two parts,
104     both containing a constant -pi/2 rotation - others are azimuthal.
105
106     Based on a code by T. Guy Masters.
107
108     INPUT:
109         L       Spherical harmonic degree.
110
111     OUTPUT:
112         D       Wigner D-matrix for m>=0. '''
113
114     # Preallocate space, determine output size
115     d = np.zeros(np.sum((np.arange(L+1) + 1)**2))
116     d[0] = 1
117
118     if L >= 1:
119         d[1] = 0
120         d[2] = 1/np.sqrt(2)
121         d[3] = -1/np.sqrt(2)
122         d[4] = 1/2
123     # Prepare loop over all degrees
```

```
124     idx = 4
125     f1 = 1/2
126     for l in np.arange(2,L+1):
127         lp = l+1
128         kdx = idx+lp
129         fl = l + lp
130 
131         f = np.sqrt(np.arange(1,l+1) * (fl - np.arange(1,l+1)))
132         f1 = f1*(2*l-1)/(2*l)
133 
134         d[kdx] = -np.sqrt(f1)
135         d[kdx-1] = 0
136 
137         for i in np.arange(2,l+1):
138             j = kdx-i
139             d[j] = -f[i-2]*d[j+2]/f[i-1]
140 
141         # Positive N (bottom triangle)
142         f2 = f1
143         g1 = l
144         g2 = lp
145 
146         for N in np.arange(1,l+1):
147             kdx = kdx + lp
148             en2 = N+N
149             g1 += 1
150             g2 -= 1
151             f2 = f2*g2/g1
152             d[kdx] = -np.sqrt(f2)
153             d[kdx-1] = d[kdx]*en2/f[0]
154 
155             for i in np.arange(2,l-N+1):
156                 j = kdx - i
157                 d[j] = (en2*d[j+1] - f[i-2]*d[j+2])/f[i-1]
158 
159         # Upper triangle
160         for i in np.arange(1,l+1):
161             for j in np.arange(i,l+1):
162                 d[idx+j*lp+i] = d[idx+i*lp+j-l]
163         # Fix signs
164         isn = 1+np.mod(l,2)
165         for i in np.arange(0,l+1):
166             kdx = idx + i*lp
167             for j in np.arange(isn,lp+1,2):
168                 d[kdx+j] = -d[kdx+j]
169 
170         idx = idx + lp*lp
171     d = d.T
```

```
172     # Rearrange d into rotation matrices. One entry per spherical
            harmonic degree
173     D = []
174     cst = 0
175     for l in np.arange(1,L+2):
176         if l == 1:
177             D.append(d[cst:cst+l**2])
178         else:
179             D.append(d[cst:cst+l**2].reshape(l,l))
180         cst = cst + l**2
181
182     # Output
183     return D, d
184
185 def r2c(lmcosi):
186     ''' lmrc = r2c(lmcosi)
187
188     Transforms real harmonic coefficients to complex harmonic
            coefficients.
189
190     INPUT:
191         lmcosi      [l m cos sin] matrix with real harmonic
                coefficients.
192
193     OUTPUT:
194         lmrc        [l m cos sin] matrix with complex harmonic
                coefficients. '''
195     _,_,mz,_,_,_,_,_,_ = addmon(int(max(lmcosi[:,0])))
196     lmrc = lmcosi.copy()
197
198     # Divide by sqrt(2)
199     lmrc[:,2:] = lmcosi[:,2:]/np.sqrt(2)
200     # Except mz entries, i.e. m = 0
201     lmrc[mz,2:] = lmcosi[mz,2:]
202     # Output
203     return lmrc
204
205 def c2r(lmrc):
206     ''' lmcosi = c2r(lmrc)
207
208     Transforms complex harmonic coefficients to real harmonic
            coefficients.
209
210     INPUT:
211         lmrc        [l m cos sin] matrix with complex harmonic
                coefficients.
212
213     OUTPUT:
```

```
214         lmcosi      [l m cos sin] matrix with real harmonic
                coefficients.'''
215     _,_,mz,_,_,_,_,_,_,_ = addmon(int(max(lmrc[:,0])))
216     lmcosi = lmrc.copy()
217     # Multiply everything with sqrt(2)
218     lmcosi[:,2:] = lmrc[:,2:]*np.sqrt(2)
219     # Except mz entries, i.e. m = 0
220     lmcosi[mz,2:] = lmrc[mz,2:]
221
222     # Output
223     return lmcosi
224
225 def rotcof(cos,sin,rot):
226     ''' Cosrot,Sinrot = rotcof(cos,sin,rot)
227     Rotates real harmonic coefficients over some angle using Equation
228     (C.245 in Dahlen and Tromp (1998)).
229
230     INPUT:
231         cos         Cosine coefficients.
232         sin         Sine coefficients.
233         rot         Rotation angle [radians].
234     OUTPUT:
235         Cosrot      Rotated cosine coefficients.
236         Sinrot      Rotated sine coefficients. '''
237     # Determine maximum spherical harmonic degree
238     L = int(addmup(np.size(cos),0))
239
240     Cosrot = cos.copy()
241     Sinrot = sin.copy()
242
243
244     Cangl = np.cos(np.arange(0,L+1)*rot)
245     #Cangl[abs(Cangl) < 1e-10] = 0
246     Sangl = np.sin(np.arange(0,L+1)*rot)
247     #Sangl[abs(Sangl) < 1e-10] = 0
248
249     for l in np.arange(0,L+1):
250         # Extract coefficients
251         Ccos, lo, up = shcos(cos,l)
252         Ssin, _, _ = shsin(sin,l)
253
254         # Rotate and collect
255         Cosrot[lo:up+1] = Ccos*Cangl[:l+1] + Ssin*Sangl[:l+1]
256         Sinrot[lo:up+1] = Ssin*Cangl[:l+1] - Ccos*Sangl[:l+1]
257
258     # Output
259     return Cosrot, Sinrot
260
```

```
261 def shcos(lmcosi,L):
262     ''' Ccos, lo, up = shcos(lmcosi, L)
263 
264     Find cosine coefficients belonging to degree L from a 1D or 2D
        array, i.e.
265     inputting either an array with cosine coefficients or an lmcosi
        matrix
266     of [l m cos sin] columns.
267 
268     INPUT:
269         lmcosi      as 2D array with [l m cos sin] columns.
270         lmcosi      as array with cos entries from 2D array.
271         L           spherical harmonic degree.
272 
273     OUTPUT:
274         CCos        Cosine coefficients belonging to degree L.
275         lo, up      Upper and lower indices of coeffients from input.
                ''' 
276     # Determine input dimension and extract minimum degree
277     if lmcosi.ndim == 1:
278         Lmin = 0
279     else:
280         Lmin = min(lmcosi[:,0]).astype(int)
281 
282     # Cosine coefficient indices belonging to degree L
283     lo = int(addmup(L-1,1) - addmup(Lmin-1,1))
284     up = int(addmup(L,1) - addmup(Lmin-1,1) - 1)
285 
286     # Obtain coefficients
287     if lmcosi.ndim == 1:
288         Ccos = lmcosi[lo:up+1].copy()
289     else:
290         Ccos = lmcosi[lo:up+1,2].copy()
291 
292     # Output
293     return Ccos, lo, up
294 
295 def shsin(lmcosi,L):
296     ''' Ssin, lo, up = shsin(lmcosi,L)
297 
298     Find sine coefficients belonging to degree L from a 1D or 2D array,
        i.e.
299     inputting either an array with cosine coefficients or an lmcosi
        matrix
300     of [l m cos sin] columns.
301 
302     INPUT:
303         lmcosi      as 2D array with [l m cos sin] columns.
```

```
304         lmcosi      as array with cos entries from 2D array.
305         L           spherical harmonic degree.
306     OUTPUT:
307         Ssin        Sine coefficients belonging to degree L.
308         lo,up       Upper and lower indices of coeffients from input.
                ''',
309     # Determine input dimension and extract minimum degree
310     if lmcosi.ndim == 1:
311         Lmin = 0
312     else:
313         Lmin = min(lmcosi[:,0]).astype(int)
314
315     # Sine coefficient indices belonging to degree L
316     lo = int(addmup(L-1,1) - addmup(Lmin-1,1))
317     up = int(addmup(L,1) - addmup(Lmin-1,1) - 1)
318
319     # Obtain coefficients
320     if lmcosi.ndim == 1:
321         Ssin = lmcosi[lo:up+1].copy()
322     else:
323         Ssin = lmcosi[lo:up+1,3].copy()
324
325     # Output
326     return Ssin, lo, up
```

Appendix F

GMT_tools

```
1  import numpy as np
2
3  def get_Pnm(nmax, theta):
4      """
5      Calculation of associated Legendre functions P(n,m) (Schmidt
           normalized)
6      and its derivative dP(n,m) vrt. theta.
7
8      Input: theta[:] co-latitude (in rad)
9             nmax  maximum spherical harmonic degree
10     Output: Pnm     ndarray PD with Legendre functions
11
12     P(n,m) ==> Pnm(n,m) and dP(n,m) ==> Pnm(m,n+1)
13     """
14
15     costh = np.cos(theta)
16     sinth = np.sqrt(1-costh**2)
17
18     Pnm = np.zeros((nmax+1, nmax+2, len(theta)))
19     Pnm[0][0] = 1
20     Pnm[1][1] = sinth
21
22     rootn = np.sqrt(np.arange(0, 2*nmax**2+1))
23
24 #      Recursion relations after Langel "The Main Field" (1987),
25 # eq. (27) and Table 2 (p. 256)
26     for m in np.arange(0, nmax):
27 #          Pnm_tmp = np.sqrt(m+m+1)*Pnm[m][m]
28         Pnm_tmp = rootn[m+m+1]*Pnm[m][m]
29         Pnm[m+1][m] = costh*Pnm_tmp
30         if m > 0:
31             Pnm[m+1][m+1] = sinth*Pnm_tmp/rootn[m+m+2]
32         for n in np.arange(m+2, nmax+1):
33             d = n*n - m*m
```

```
34                  e = n + n - 1
35                  Pnm[n][m] = (e*costh*Pnm[n-1][m]-\
36                                  rootn[d-e]*Pnm[n-2][m])/rootn[d]
37
38  #     dP(n,m) = Pnm(m,n+1) is the derivative of P(n,m) vrt. theta
39      Pnm[0][2] = -Pnm[1][1]
40      Pnm[1][2] =  Pnm[1][0]
41      for n in np.arange(2, nmax+1):
42              l = n + 1
43              Pnm[0][l] = -np.sqrt(.5*(n*n+n))*Pnm[n][1]
44              Pnm[1][l] = .5*(np.sqrt(2.*(n*n+n))*Pnm[n][0]-\
45                              np.sqrt((n*n+n-2.))*Pnm[n][2])
46
47              for m in np.arange(2, n):
48                  Pnm[m][l] = .5*(np.sqrt((n+m)*(n-m+1.))*Pnm[n][m-1] -\
49                              np.sqrt((n+m+1.)*(n-m))*Pnm[n][m+1])
50
51              Pnm[n][l] = .5*np.sqrt(2.*n)*Pnm[n][n-1]
52
53      return Pnm
54
55  def get_Pm(n, order, theta):
56      """
57      Calculation of associated Legendre functions P(n,m) (Schmidt
            normalized)
58
59      Input: theta[:] co-latitude (in rad)
60             nmax  maximum spherical harmonic degree
61      Output: Pnm     ndarray PD with Legendre functions
62
63      P(n,m) ==> Pnm(n,m) and dP(n,m) ==> Pnm(m,n+1)
64      """
65      if type(n) != int:
66          nmax=max(n)
67          if nmax==0:
68              nmax+=1
69      else:
70          nmax=n
71      costh = np.cos(theta)
72      sinth = np.sqrt(1-costh**2)
73
74      Pm = np.zeros((nmax+1,len(theta)))
75
76      rootn = np.sqrt(np.arange(0, 2*nmax**2+1))
77
78  #       Recursion
79      for m in np.arange(0, nmax):
80          if m==0:
```

```
81              Pmdiag=np.ones(len(theta))
82          elif m==1:
83              Pmdiag=sinth
84          Pm_tmp = rootn[m+m+1]*Pmdiag
85          if m == order:
86              Pm[m,:] = Pmdiag
87              Pm[m+1,:] = costh*Pm_tmp
88              for L in np.arange(m+2, nmax+1):
89                  d = L*L - m*m
90                  e = L+L - 1
91                  Pm[L,:] = (e*costh*Pm[L-1,:]-rootn[d-e]*Pm[L-2,:])/
                        rootn[d]
92              break
93          if m > 0:
94              Pmdiag = sinth*Pm_tmp/rootn[m+m+2]
95
96      return Pm
97
98  def design_SHA(r, theta, phi, nmax):
99
100     """
101     Created on Fri Feb  2 09:18:42 2018
102
103     @author: nilos
104     A_r, A_theta, A_phi = design_SHA(r, theta, phi, N)
105
106      Calculates design matrices A_i that connects the vector
107      of (Schmidt-normalized) spherical harmonic expansion coefficients,
108      x = (g_1^0; g_1^1; h_1^1; g_2^0; g_2^1; h_2^1; ... g_N^N; h_N^N)
109      and the magnetic component B_i, where "i" is "r", "theta" or "phi
             ":
110          B_i = A_i*x
111          Input: r[:]       radius vector (in units of the reference
                 radius a)
112          theta[:]  colatitude     (in radians)
113          phi[:]    longitude      (in radians)
114          N         maximum degree/order
115
116      A_r, A_theta, A_phi = design_SHA(r, theta, phi, N, i_e_flag)
117      with i_e_flag = 'int' for internal sources (g_n^m and h_n^m)
118                      'ext' for external sources (q_n^m and s_n^m)
119      """
120
121     cml = np.zeros((nmax+1, len(theta))) # cos(m*phi)
122     sml = np.zeros((nmax+1, len(theta))) # sin(m*phi)
123     a_r = np.zeros((nmax+1, len(theta)))
124     cml[0]= 1
125     for m in np.arange(1, nmax+1):
```

```
126             cml[m]=np.cos(m*phi)
127             sml[m]=np.sin(m*phi)
128         for n in np.arange(1, nmax+1):
129             a_r[n]=r**(-(n+2))
130 
131         Pnm = get_Pnm(nmax, theta)
132         sinth = Pnm[1][1]
133 
134 # construct A_r, A_theta, A_phi
135         A_r =     np.zeros((nmax*(nmax+2), len(theta)))
136         A_theta = np.zeros((nmax*(nmax+2), len(theta)))
137         A_phi =   np.zeros((nmax*(nmax+2), len(theta)))
138 
139         l = 0
140         for n in np.arange(1, nmax+1):
141             for m in np.arange(0, n+1):
142                 A_r[l] =     (n+1.)*Pnm[n][m]    *cml[m] * a_r[n]
143                 A_theta[l] =       -Pnm[m][n+1] *cml[m] * a_r[n]
144                 A_phi[l] =        m*Pnm[n][m]    *sml[m] * a_r[n] / sinth
145                 l=l+1
146                 if m > 0:
147                     A_r[l] =     (n+1.)*Pnm[n][m]    * sml[m] * a_r[n]
148                     A_theta[l] =       -Pnm[m][n+1] * sml[m] * a_r[n]
149                     A_phi[l] =       -m*Pnm[n][m]    * cml[m] * a_r[n] /
                            sinth
150                     l=l+1
151         return A_r.transpose(), A_theta.transpose(), A_phi.transpose()
152 
153 def synth_grid(gh, r, theta, phi):
154         """
155         Created on Fri Feb  2 09:18:42 2018
156 
157         @author: nilos
158         B_r, B_theta, B_phi = synth_grid(gh, r, theta, phi)
159 
160         """
161         n_coeff = len(gh)
162         nmax = int(np.sqrt(n_coeff+1)-1)
163         N_theta   = len(theta)
164         N_phi     = len(phi)
165         n = np.arange(0, nmax+1)
166 
167         r_n       = r**(-(n+2))
168 
169         cos_sin_m = np.ones((n_coeff, N_phi))
170         sin_cos_m = np.zeros((n_coeff, N_phi))
171         T_r     = np.zeros((n_coeff, N_theta))
172         T_theta = np.zeros((n_coeff, N_theta))
```

```
173     T_phi   = np.zeros((n_coeff, N_theta))
174
175     Pnm = get_Pnm(nmax, theta)
176     sinth = Pnm[1][1]
177
178     k=0
179     for n in np.arange(1, nmax+1):
180         T_r[k]     = (n+1.)*r_n[n]*Pnm[n][0]
181         T_theta[k] = -r_n[n]*Pnm[0][n+1]
182         k = k+1
183         for m in np.arange(1, n+1):
184             T_r[k]     = (n+1)*r_n[n]*Pnm[n][m]
185             T_theta[k] = -r_n[n]*Pnm[m][n+1]
186             T_phi[k]   = m*r_n[n]*Pnm[n][m] / sinth
187             cos_sin_m[k] = np.cos(m*phi)
188             sin_cos_m[k+1] = cos_sin_m[k]
189             T_r[k+1]     = T_r[k]
190             T_theta[k+1] = T_theta[k]
191             T_phi[k+1]   = -T_phi[k]
192             cos_sin_m[k+1] = np.sin(m*phi)
193             sin_cos_m[k] = cos_sin_m[k+1]
194             k = k+2
195
196     tmp = cos_sin_m*gh[:, np.newaxis]
197     B_r = np.matmul(T_r.transpose(), tmp)
198     B_theta = np.matmul(T_theta.transpose(), tmp)
199     B_phi = np.matmul(T_phi.transpose(), sin_cos_m*gh[:, np.newaxis])
200
201     return B_r, B_theta, B_phi
202
203 def read_shc(shc_fn,cols='all'):
204   """
205   Read values of Gauss coefficients (g,h) from column(s) in file.
206
207   File should be ascii file obeying the SHC format.
208
209   Parameters
210   ----------
211   shc_fn : str
212     Path of input SHC ascii file
213   cols : list_like
214     List of columns to read from file. This should correspond to the
215     columns the different times values coefficients will be
216     read from. In a standard SHC file the first two columns (0 and 1)
217     correspond to the degree (l) and order (m) of the harmonic and
218     should not be included in `cols`. As such the default value
219     ``cols='all'`` corresponds to ``cols=range(2,2+N_times)``, where
220     ``N_times`` is the number of time snapshots in the file.
```

```
221
222     Returns
223     -------
224     Tuple
225       Tuple with following values at given indices:
226
227         0. numpy.ndarray of gaussian coefficients with such that
228             ``myarray[0]`` gives all coefficients at the first time point
                ,
229             given that there are multiple time snapshots. Otherwise
230             ``array[0]`` will only contain the first coefficient.
231         1. spline order `k` as an integer used to reconstruct model from
232             time snapshots.
233         2. number of columns as an integer.
234         3. time of the temporal snapshots (in fractional years in the
235             standard SHC format) as 1D `numpy.ndarray`.
236
237     Notes
238     -----
239     Missing data values marked as NaN are currently not handled.
240
241     """
242
243     with open(shc_fn) as f:
244       headerlen=0
245       header_fin=False
246       for line in f:
247         if header_fin: # finished reading header
248           times=np.empty(N_times)
249           c=0
250           for t in line.split():
251             times[c]=float(t)
252             c+=1
253           break
254         else:
255           if line.startswith('#'):
256             headerlen+=1
257           else:
258             header_fin=True
259             N_min,N_max,N_times,spline_order,N_step = \
260               (int(v) for v in line.split()[:5])
261     cols=range(2,2+N_times)
262 #   gh=np.loadtxt(shc_fn,skiprows=headerlen+2)
263     gh=np.loadtxt(shc_fn,skiprows=headerlen+2,usecols=cols,unpack=True)
264 #   if len(gh.shape)==1:
265 #     gh=np.expand_dims(gh,0)
266
267     # currently not passing on N_min,N_max,N_step
```

```
268   return gh,spline_order,times
269
270 def mauersberger_lowes_spec(gh,r=1):
271     """ The Mauersberger-Lowes spatial powerspectrum"""
272     ratio=1/r
273     N = int(np.sqrt(gh.size+1)-1) # maximum spherical harmonic degree
274     R_l=np.empty(N)
275     gh_idx=0
276     for l in range(1,N+1):
277         gh_idx_n=gh_idx+2*l+1
278         g_sq=np.sum(gh[gh_idx:gh_idx_n]**2)
279         R_l[l-1] = (l+1)*ratio**(2*l+4)*g_sq
280         gh_idx=gh_idx_n
281     return R_l
282
283
284 def degree_correlation(gh1,gh2,lmax=-1,lmin=1):
285     """Correlation per spherical harmonic degree between two models 1
        and 2"""
286     if lmax<1:
287         lmax1,lmin1=get_l_maxmin(len(gh1))
288         lmax2,lmin2=get_l_maxmin(len(gh2))
289         lmax = min(lmax1,lmax2)
290         lmin = min(lmin1,lmin2)
291     c12=np.empty(lmax+1-lmin)
292     i=0
293     for l in range(lmin,lmax+1):
294         #m=0
295         g12 = gh1[i]*gh2[i]
296         g11 = gh1[i]**2
297         g22 = gh2[i]**2
298         i+=1
299         for m in range(1,l+1):
300             g12 += gh1[i]*gh2[i]
301             g11 += gh1[i]**2
302             g22 += gh2[i]**2
303             i += 2
304         c12[l-lmin] = g12/np.sqrt(g11*g22)
305     return c12
```

DTU Space
Division of Geomagnetism
Technical University of Denmark

Elektrovej, Building 356
DK - 2800 Kongens Lyngby
Tlf. (+45) 45 25 95 00
Fax (+45) 45 25 97 01