Emmanuel Nahayo

A study of geomagnetic field time variations over southern Africa using a regional harmonic spline core magnetic field model derived from CHAMP satellite and ground magnetic observations

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

Johannesburg, October 2019

School of Geosciences, University of the Witwatersrand

Declaration

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

4^{th} day of October 2019

Acknowledgements

Undertaking this PhD has been a truly life-changing experience for me and it would not have been possible to do without the support and guidance that I received from many people.

I would like to first say a very big thank you to my supervisors Prof. Susan J. Webb and Dr. Pieter B. Kotzé for all the support, guidance and encouragement they gave me. Without their guidance and constant feedback this PhD would not have been achievable.

Many thanks also to Dr. Monika Korte who has read my manuscript from its early stages for valuable comments and suggestions for its improvement.

A very special thank you to Dr. Vincent Lesur and Dr. Ingo Wardinski for their dedication in explaining to me the modelling technique that was used in my PhD studies.

I gratefully acknowledge the financial support that was provided by the South African National Space Agency (SANSA) for funding my studies.

The support of the CHAMP mission by the German Aerospace Center (DLR) and the Federal Ministry of Education and Research are gratefully acknowledged.

The operation of southern African ground magnetic observatories by the South African National Space Agency (SANSA), supported by the Department of Science and Technology (DST), is acknowledged for high quality ground-based data.

Many thanks to the Centre for High Performance Computing (CHPC) for its financial support to attend its two workshops in 2016 to improve my computer programming skills.

Dedication

I dedicate this thesis to my whole family, my mother who taught me to trust God and believe in hard work and that so much could be done with little, my father who encouraged me to believe in myself and to always persevere to overcome tough challenges, my wife Denyse and my two daughters Joyce and Hirwa who have been my inspiration through the whole journey of my studies.

Abstract

The geomagnetic field spatial variation in Southern Africa is characterised by a high horizontal gradient, and its study requires high spatial resolution data. Regional field models are more suitable than global models to study small scale features of the geomagnetic field variation. They use more dense data, hence a good data coverage allows a detailed description of the geomagnetic variation in the area of investigation. In Southern Africa, the ground recording stations are limited (4 magnetic observatories and 38 geomagnetic repeat stations) and satellite data are needed for studies where high spatial and temporal resolution data are required. The southern African region borders the South Atlantic Anomaly (SAA) where the strength of the magnetic field is approximately 30% weaker compared to other regions of the same latitudes. In an attempt of supporting geomagnetic field model data users and investigating the evolution of the South Atlantic Anomaly, a study to develop and derive a southern African regional field model combining CHAMP satellite data with ground-based data showed that the regional model over Southern Africa can be improved by combining both satellite and ground-based data. The internal Earth processes generating the main field can be studied by modelling the core field using inversion of geomagnetic field components measured above the surface of the Earth, and they assist in monitoring the long-term variation of the geomagnetic field on time scales of 1 year and more, therefore giving valuable information on the evolution of the SAA in southern Africa.

The study of the geomagnetic field time variation in the southern Africa region was carried out applying the harmonic splines technique on CHAMP satellite and ground data that have been recorded between 2001 and 2010. A Southern Africa Regional Model (SARM) was derived and evaluated using global models, such as International Geomagnetic Reference Field (IGRF-11) and GFZ Reference Internal

Magnetic Model (GRIMM), by calculating the difference between SARM and global models. The results of this study suggest that the 200 km gridded SARM model developed from only satellite data is shown to match monthly averaged ground data to within 1.5%, 1.6% and 0.7% for X, Y and Z, respectively, suggesting the temporal variations are valid where ground data are not available. In addition, these results also confirm the earlier findings of the 2007 magnetic jerk and rapid secular variation fluctuations of 2003 and 2004 on the eastern edge of the SAA which extends 1000 km into South Africa.

CHAMP satellite and relatively densely spaced (300 km) ground-based data measured over southern African region between 2005 and 2010 are incorporated in a new regional, harmonic spline based, core field model. Our new SACFM-2 model is compared to our previously developed SARM model, a regional model based only on satellite data, and the global CHAOS-6 model developed by Finlay et al. (2016). The results agree well with the CHAOS-6 model (within 0.4%) in the vertical (Z) component and total field (F) that are used to investigate the evolution of the South Atlantic Anomaly in the region. The computed maps of main field of the Z component and total intensity F show a steady decrease of the field between 2005.5 and 2010.5, reaching an average of 40 nT and 50 nT per year, respectively, in the southwest of South Africa, indicating evolution of the South Atlantic Anomaly under Southern Africa and an increase of the geographical area of this feature.

In addition, a southern African lithospheric magnetic model (SALMM) over Southern Africa was developed based on the Spherical Cap Harmonic Analysis method using CHAMP satellite measurements at an altitude between 332 km and 365 km, and between 2007.0 and 2009.0 epochs. This lithospheric magnetic model SALMM was compared to a regional model developed by Vervelidou et al. (2018) and the global model MF7. Our lithospheric magnetic model SALMM was interpreted in terms of regional (100s km) geological features and long-wavelength geological magnetic anomalies.

Contents

List of Figures

List of Tables

List of abbreviations

CHAMP	CHAllenging Minisatellite Payload
SARM	Southern African Regional Model
SACFM	Southern African Core Field Model
SALMM	Southern African Lithospheric Magnetic Model
CM4	Comprehensive Model phase 4
Magsat	Magnetic satellite
IGRF	International Geomagnetic Reference Field
GRIMM	GFZ Reference Internal Magnetic Model
POMME	POtsdam Magnetic Model of the Earth
SCHA	Spherical Cap Harmonic Analysis
Dst	Disturbance Storm Time
INTERMAGNET	International Real-time Magnetic Observatory Network
SAA	South Atlantic Anomaly

Chapter 1

Introduction

1.1 Thesis Objective

The objective of this thesis was to study the geomagnetic field time variation over southern Africa between 2001 and 2010 by developing a regional magnetic field model. The focus of this study was to shed light on small scale features of the time variation of the geomagnetic field like rapid secular variation fluctuations and identify the geomagnetic jerks that occurred in this region. The regional magnetic field model was to be used to determine whether the South Atlantic Anomaly (SAA) is expanding in southern Africa or not and determine the magnitude of the drop in field strength. The lithospheric magnetic field model developed during this study was employed to identify the location of anomalous magnetic variations that can be tied to lithospheric geological features in southern Africa.

1.2 Organisation of the Thesis

The background of the thesis is presented in Chapter 2. This chapter presents the summary of the literature of the geomagnetic field modelling. It explains different methods that are used in global and regional field modelling. There is a detailed explanation of the technique of harmonic splines that has been used in this study of the time-variation of geomagnetic field over southern Africa.

Geomagnetic field modelling requires measurements as input data. CHAMP satellite and ground-based data are presented in Chapter 3. This chapter highlights the data selection process using geomagnetic indices and data processing for the preparation of the model input data.

The study of secular variation over southern Africa, applying the harmonic splines technique to CHAMP satellite data recorded between 2001 and 2010, is presented in Chapter 4. A regional model, Southern African Regional Model (SARM), is developed and evaluated using global models. SARM is used to shed light on small scale temporal and spatial features of the time-variation geomagnetic field in this region.

In an attempt to improve the SARM model, the investigation of combining both satellite and ground-based data to develop a regional model over southern Africa was conducted. This is presented in chapter 5 and suggests the areas of improvement. The developed regional field model, Southern African Core Field Model (SACFM-2), was subsequently improved and used to study the evolution of South Atlantic Anomaly over southern Africa (Chapter 6).

In Chapter 6, the new model SACFM-2 is compared to the regional SARM model and the global CHAOS-6 model. In this chapter, I present the investigation of the SAA using the Z component and total intensity F. The SACFM-2 computed maps of the main field of the Z component and total field F show a steady decrease of the field over the years during the study period 2005-2010.

In Chapter 7, Spherical cap harmonic analysis technique is applied on CHAMP satellite data recorded between 2007.0 and 2009.0 epochs to develop a lithospheric magnetic model SALMM in X, Y and Z field components at satellite altitude. This model SALMM is evaluated with the help of a recent developed regional high resolution lithospheric field model over southern Africa by Vervelidou et al. (2018) and the global model MF7. The prominent long-wavelength lithospheric field anomalies in the region are identified to confirm the previous work. Closing remarks are presented in Chapter 8.

1.3 Contribution of the Thesis

This study has shown for the first time that an accurate southern African regional model can be derived using harmonic splines applied to satellite data only. This harmonic spline regional model, SARM, was able to identify the 2007 geomagnetic jerk as well as the 2003 and 2004 rapid secular variation fluctuations over southern Africa. The regional model, using densely data, is an improvement to any global model by describing the small scale features of the time-variation of the geomagnetic field.

This study has shown that the regional model over southern Africa can be improved combining both satellite and ground-based data. The resulting model SACFM-2, with a better performance in the main field of the Z component than the CHAOS-6 model, was used to study the evolution of South Atlantic Anomaly in the region. It has been shown that during the selected period of study 2005-2010, there is a steady decrease of the field indicating the evolution of the South Atlantic Anomaly over southern Africa and suggesting an increase of the area of this feature. Between 2005.5 and 2010.5, the decrease of the field in magnitude ranges between 0 to 200 nT and 0 to 250 nT in the Z component and total intensity F, respectively, with faster decrease in the south-west corner of the considered area in the study, reaching 0.9%.

The developed regional lithospheric field model SALMM at satellite altitude (350 km) was used to identify long-wavelength anomalies (1250 km) in the southern African region to confirm the previous work of other crustal field modellers. The use of only satellite vector data was proven to be sufficient to model successfully a long wavelength lithospheric magnetic field over southern Africa in X, Y and Z components at satellite altitude.

This thesis has contributed three research articles to peer-reviewed journals.

Chapter 2

Geomagnetic field modelling

2.1 Earth's magnetic field

The magnetic field measured above the surface of Earth can be closely approximated by the field of a large bar magnet placed at the center of the Earth, with its north end oriented toward the south magnetic pole (Fig. 2.1). The imaginary bar magnet axis is tilted at an angle of about 11 degrees from the spin axis of the Earth. In actual fact the shape of the geomagnetic field is more complex than a bar magnet. The magnetic field is not completely dipolar and it changes with both location and time.

The Earth is composed of layers, from outside to inside: a thin outer crust, a silicate mantle, an outer core and an inner core. Both pressure and temperature increase with depth within the Earth where the temperature is about 25°C/km heading towards the center. The temperature at the core mantle boundary (approximately at 2890 km beneath the Earth's surface) is roughly 4800°C, hot enough for the outer core to exist in a molten liquid state. Because of the increased pressure, the inner core is a solid due to the increased pressure and it is composed mainly of nickel and iron, with a small portion of lighter elements. The outer core is in constant motion due to both the Earth's rotation and convection. The upward motion of the light elements drive the convection as the heavier elements freeze onto the inner core.

The accepted hypothesis in the scientific community for generating a geomagnetic

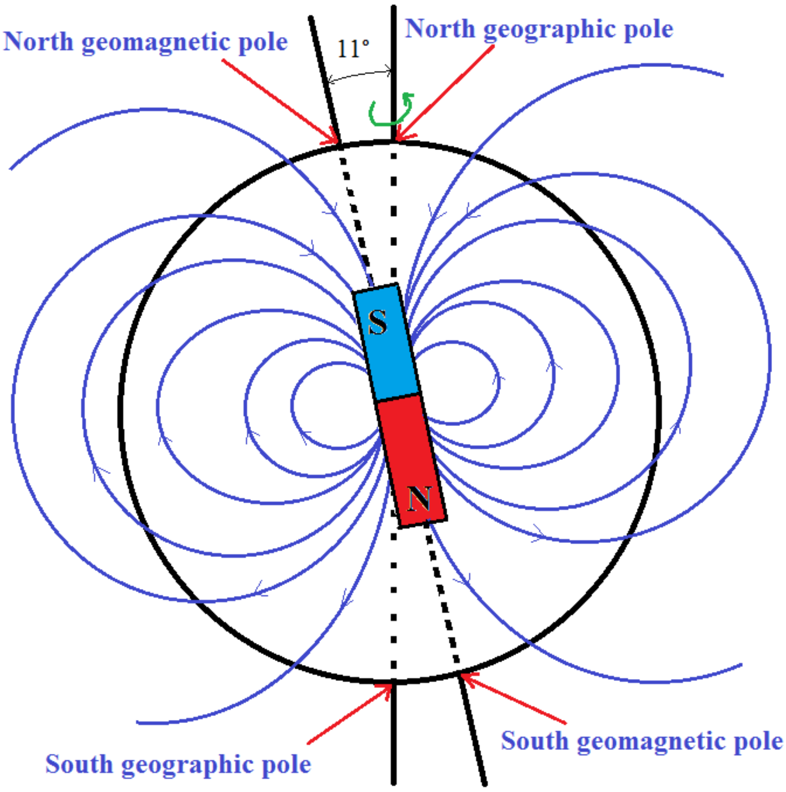

Figure 2.1: The illustration of the imaginary bar magnet placed at the center of the Earth.

field is that the liquid outer core of the Earth maintains an electric current as a self-excited dynamo (Campbell, 2003). From Maxwell's equations, the electric and changing magnetic fields are closely linked and can affect each other. Faraday's law of induction states that the electromotive force induced in a circuit is directly proportional to the time rate of change of magnetic flux through the circuit,

$$\nabla \times \boldsymbol{E} = -\frac{\partial \boldsymbol{B}}{\partial t} \tag{2.1}$$

where $\boldsymbol{E}$ and $\boldsymbol{B}$ are electric field and magnetic field vectors respectively.

According to the principle of electrical motors, the motion of an electrical conductor through a magnetic field will cause electrons to flow thus generating an electrical current.

The generation of such a magnetic field is explained by a dynamo process due to the movement of the fluid (molten iron) outer core of the Earth (Fig. 2.2). Due to the heat flow from the inner core, the Earth's outer core is in a state of turbulent convection. The motion of the electrically conducting iron in the Earth's outer core in the presence of the Earth's magnetic field induces electric currents. The magnetic field is generated out of these new currents, and as the effect of this internal feedback, the process is self-sustaining so long as there is energy source adequate to maintain convection (Hollerbach, 1996).

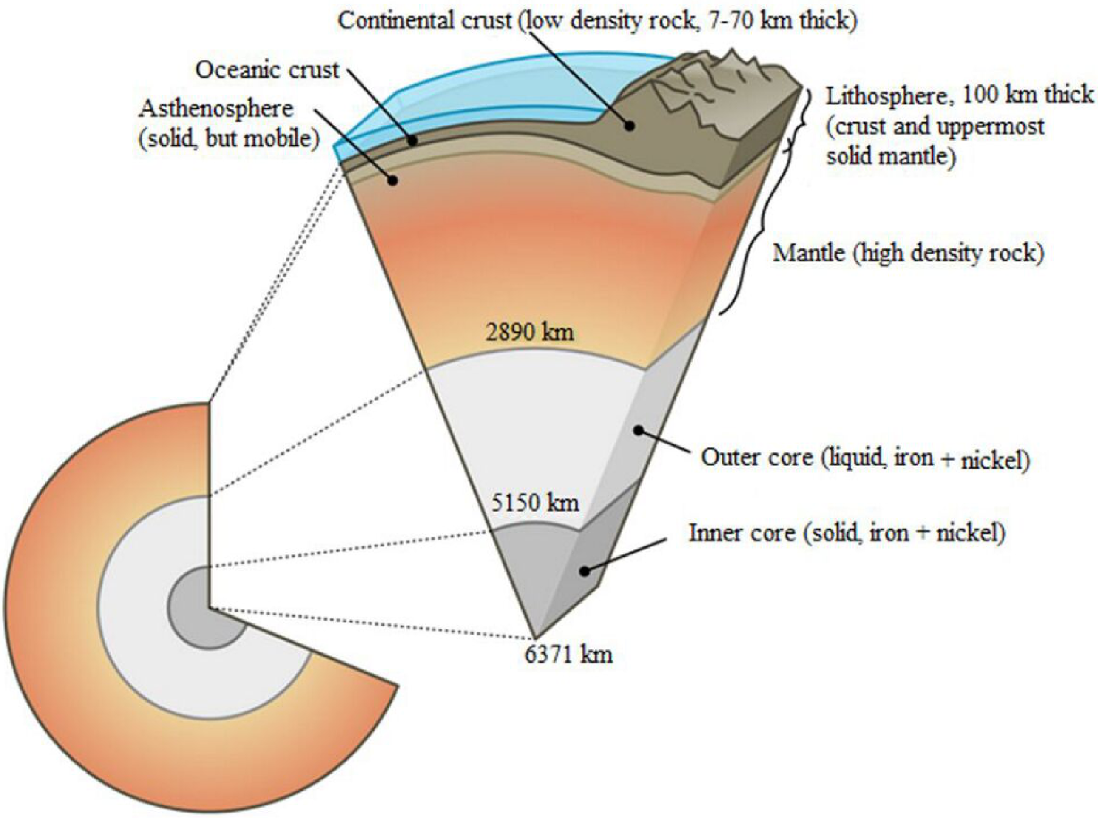

Figure 2.2: The layers of Earth with their physical properties and chemical composition.

The majority of the magnetic field measured at the Earth's surface has its origin in electrical currents flowing in the liquid outer core. This field is known as the *main* or *core field* (Hollerbach, 1996). It exhibits long-term changes which take place at unpredictable and irregular rates. Additionally to this dominant contribution of the Earth's magnetic field, the lithospheric field arising from magnetised rocks and thus contain information about the magnetic field at the time of their solidification, and from electromagnetically induced currents in the conducting planetary interior (Saur et al., 2010) that produce induced fields in Earth's electrically conducting ocean, lithosphere, and mantle which arise from primarily time varying fields from within the core and external to the Earth. Another significant contribution is that of external sources which originate in the ionosphere and magnetosphere (Pulkkinen et al., 2003).

The study of Earth's magnetic field allows scientists to understand the dynamics of the core, and in paleomagnetism they can shed light on plate tectonics and Earth's evolution. There are various applications in the field of geomagnetism like navigation using compasses and geophysical exploration in the search for ore deposits. The geomagnetic field is the shield of the Earth's atmosphere against the cosmic particles and the solar wind consisting of solar particles emitted frequently by the Sun which can harm life (radiation hazards on life) or destroy human technology on Earth (e.g. electrical power grids) and in space in the Earth's atmosphere (e.g. communication satellites).

2.2 Geomagnetic field components

A small needle dipole magnet that is freely balanced will dip following the local inclination angle in the direction of the local horizontal magnetic field component in a north-south direction. This needle dipole magnet has two ends, a south pole and a north pole. The north pole of the magnet is attracted by the Earth's dipole field toward the north arctic region, because opposite ends of magnets attract each other this region of the Earth should be called south pole. This convention was ignored for the Earth to avoid confusion so that geographic and geomagnetic pole names agree.

The Earth's magnetic field is a vector quantity; it has a strength and a direction at each point in space. For its description, three quantities are needed and they are commonly given as one of the three following options:

- Three orthogonal strength components (X, Y and Z)
- Two angles and the total field strength (D, I and F)
- One angle and two strength components (D, H, and Z)

Figure 2.3 shows the representation of these 7 elements in a diagram.

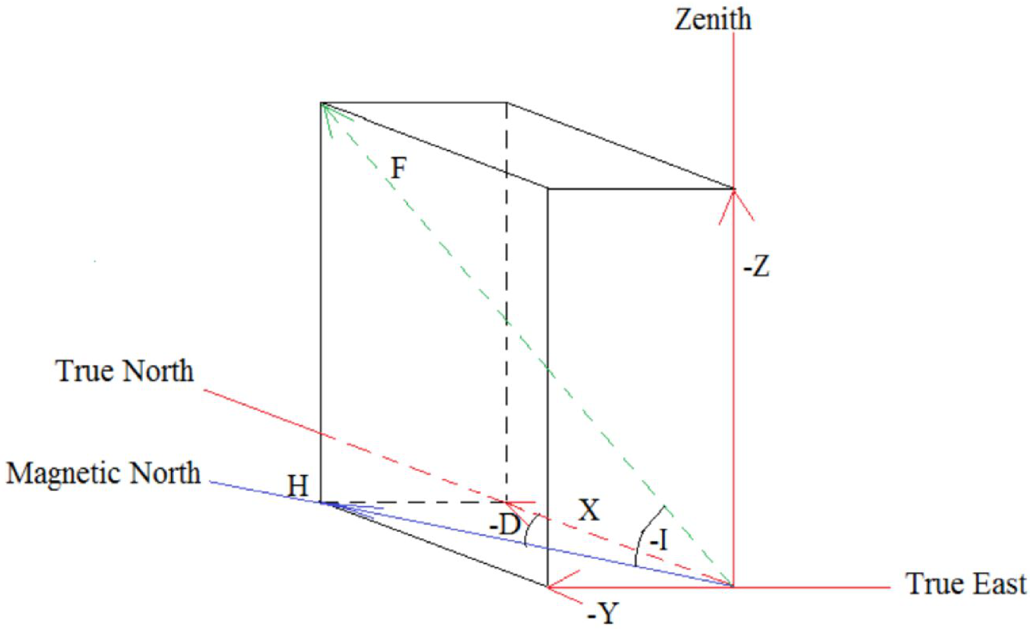

Figure 2.3: The representation of the geomagnetic field components in Southern Africa (southern hemisphere). The field component X is positive northward, and the Y and Z components are negative westward and upward, respectively, in the local magnetic coordinate system. The upward orientation of the Vertical component, which results in both Z and I having negative values, shows the southern hemisphere magnetic field orientation. In Southern Africa, the Y component westward orientation results in both Y and D having negative values.

The seven elements are related through the following expressions:

$$D = arctan(\frac{Y}{X})$$

$$I = arctan(\frac{Z}{H})$$

$$H = \sqrt{X^2 + Y^2}$$

$$X = Hcos(D)$$

Table 2.1: Magnetic components

Component	Description
F	The total intensity of magnetic field vector
H	Horizontal component
Z	Vertical component (by convention Z is negative upward)
X	North component (X is positive northward)
Y	East component (Y is negative westward)
D	Magnetic declination (angle between geographic north and magnetic north and it is negative westward of geographic north)
I	Magnetic inclination (angle measured from the horizontal plane to the magnetic field vector and it is negative upward)

$$Y = Hsin(D)$$

$$F = \sqrt{X^2 + Y^2 + Z^2}$$

In the Earth's spherical coordinates, θ is the angle measured from the geographic North pole along a great circle of longitude, ϕ is eastward angle along latitude line measured from a reference longitude, and the radial direction, r, measured from the center of the Earth. On the surface of the Earth, where field components X, Y and Z correspond to $-\theta$, ϕ, and -r directions, the field $\vec{B}$ in spherical coordinates becomes $B_\theta = -X$, $B_\phi = Y$ and $B_r = -Z$ (Campbell, 2003).

The SI unit of magnetic induction B is Tesla (T). The intensity of Earth's magnetic field is very small, and in geomagnetism the unit used for magnetic induction B is nanotesla (nT) (1nT = 10^{-9}T).

2.3 Internal and external parts of magnetic field

The observed magnetic field at or near the surface of Earth is the superposition of contributions from external sources (electric currents flowing in the ionosphere and magnetosphere) and internal sources (fluid core, magnetisation of rocks in the crust and induced currents in the Earth's interior by the time variations of external fields). The internal part is mainly the field generated in the outer core known as main field or core field. Its strength ranges from 20,000 nT at equator to 70,000 nT at the poles and it typically represents more than 98% of the total main field. In addition, there is a small contribution from the lithospheric magnetic field which consists of induced magnetisation from the main field and permanent remanent magnetisation.

The sun emits continuously a stream of charged particles consisting mainly of electrons and protons with a high speed (200-1000 km/s) known as solar wind. The solar wind carries with it the solar magnetic field which fills up the interplanetary space and its interaction with the Earth's magnetic field results in the penetration of some particles in the upper atmosphere of the Earth. These charged particles become a source of electric fields which generate measured external magnetic field on the Earth's surface. Therefore, the external magnetic field is generated from magnetic sources outside the Earth, and is mainly produced by tidal motions in the ionosphere and interactions of the Earth's magnetosphere with the solar wind and the field aligned currents (FACs) coupling Earth's magnetosphere and ionosphere with the stream of charged particles released from the upper atmosphere of the Sun. Figure 2.4 illustrates the interaction of solar wind with the Earth's magnetic field and Figure 2.5 shows spatial locations of the magnetic field sources.

2.3.1 Separation of internal and external magnetic field contributions

The magnetic field measured at or above the Earth's surface is the superposition of contributions from various sources in the Earth's environment (ionospheric and magnetosheric currents) and its interior (molten liquid outer core, magnetisation of rocks in the crust and induced currents in the Earth) (Olsen et al., 2010). Geomagnetic field variations show a regular variation with a fundamental period of 24 hours

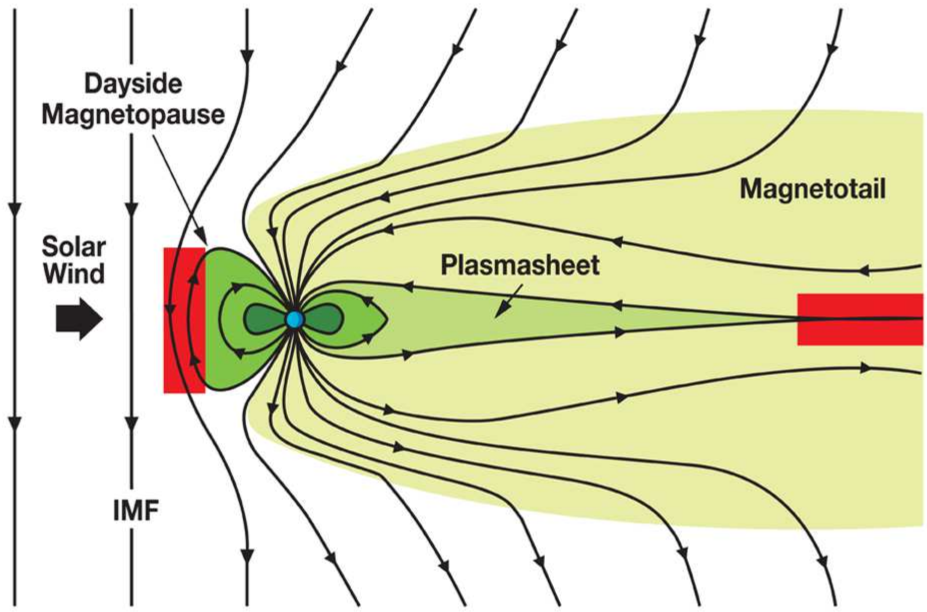

Figure 2.4: Schematic representation of the Earth's magnetosphere and its characteristic regions.

(solar quiet variations, Sq) and this is particularly large at mid to low latitudes and depends on latitude, local time, season, and solar cycle. It is caused primarily by electrical currents in the upper atmosphere (Macmillan and Droujinina, 2007). The ring current dominating at mid-latitudes is the main source of the external field, and during geomagnetic storms the decrease of the horizontal component can reach hundreds of nT (more than 300 nT) (Cid et al., 2014). Figures 2.6 and 2.7 show plots of data recorded at three magnetic observatories (HER, HBK and TSU) on a very quiet and a very disturbed day, respectively.

Some indices that are derived from Earth's magnetic observatories are used to characterise the external field activity (Mayaud, 1980). Two planetary indices Kp and Dst are mostly used to select the quietest solar activity data (see details in Chapter 3). For core and lithospheric field modelling in the mid and low latitudes, the selection of satellite data for the model considering only data recorded during a certain local night time interval and the use of the Dst index during selection play a major role in excluding the data records with unacceptable high external field contributions.

The first attempt to separate internal and external magnetic field contributions

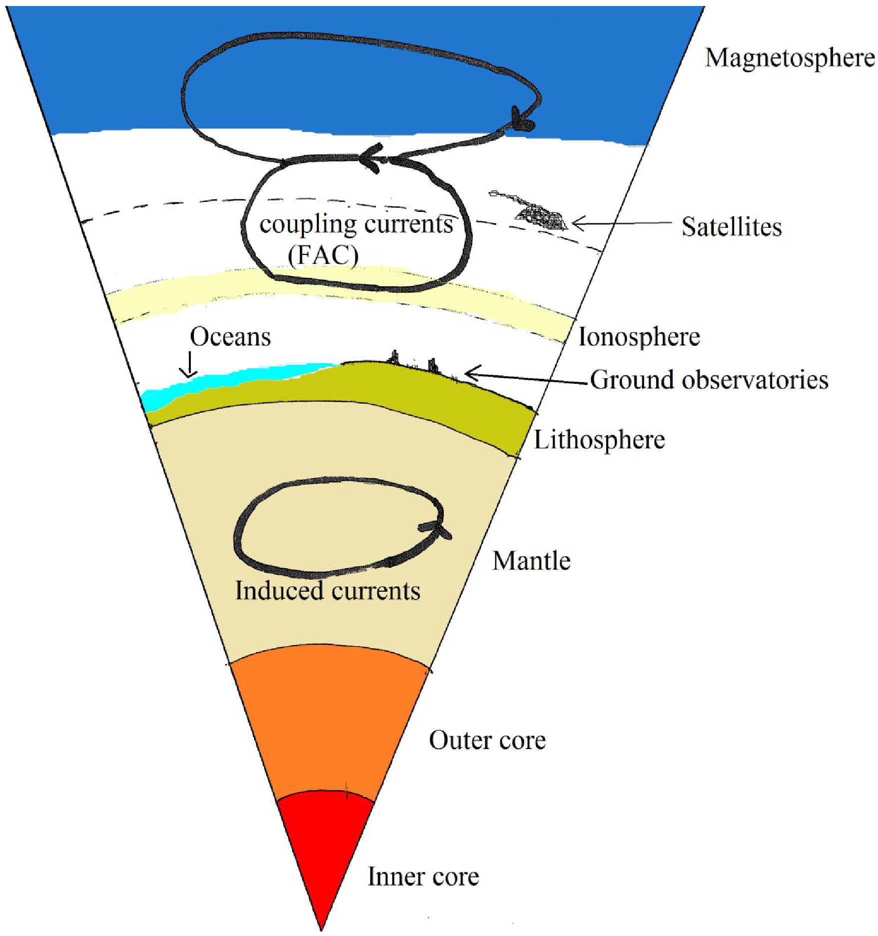

Figure 2.5: Schematic representation of magnetic field sources with their spatial location with altitude.

was conducted by Gauss (1839). He developed and applied spherical harmonic analysis method to observations of the Earth's magnetic field (see details in Section 2.5). This separation of measured Earth's magnetic field into external and internal parts is performed with respect to the altitude of the observations, not with respect to the Earth's surface. Therefore, magnetic field sources between Earth's surface and satellite altitude are seen as internal sources by satellites. The use of spherical harmonic analysis on measured satellite data would include all internal magnetic fields with sources below the satellite altitude, therefore considering also the field contributions from the ionospheric currents.

Sabaka et al. (2004) approached the separation of internal and external field contributions developing a Compressive Model (CM4) which describes the various components of the near-Earth magnetic field, by co-estimating internal, ionospheric and

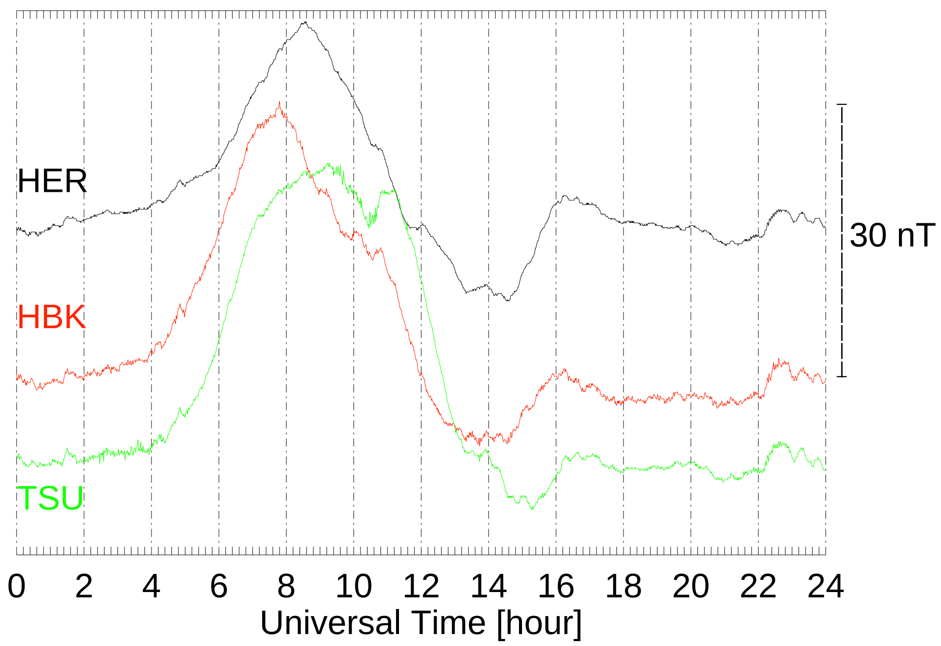

Figure 2.6: The plot of a solar-quiet magnetic variation day of 1-min average data of the H component recorded on 19th July 2008 at three magnetic observatories in the southern African region, Hermanus (HER), Hartebeesthoek (HBK) and Tsumeb (TSU). it is clear that the magnetic variations are high during the day time between 4:00 and 16:00 UT.

magnetospheric fields together with their secondary and Earth-induced counterparts in one all-inclusive inversion process. With the CM4 model, it is possible to perform a joint analysis of ground-based and satellite data and be able to separate Earth's internal sources, ionospheric sources (between Earth's surface and satellite altitude) and magnetospheric sources (above low orbiting satellite altitude). Figure 2.8 shows the observed and CM4 predicted monthly mean values curves of the X component at Hermanus magnetic observatory between 2000 and 2010. The CM4 model was used in this study to remove ionospheric and magnetospheric field contributions from model input data.

2.4 Secular variation

The Earth's magnetic field is not constant, it varies significantly on all temporal and spatial scales. The study of past geomagnetic fields with the help of magnetized rocks, that retain geomagnetic information during their formation, has shown the

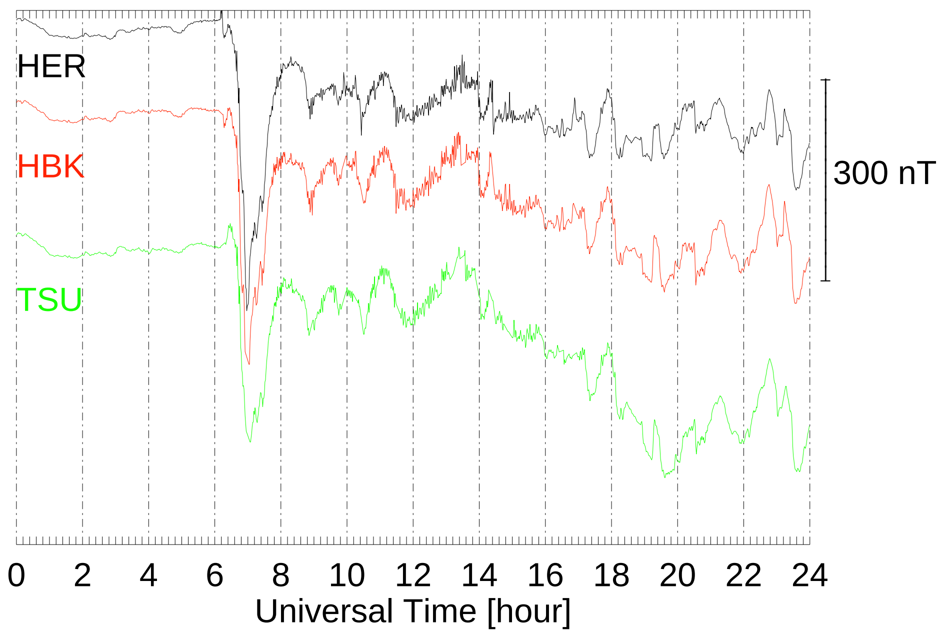

Figure 2.7: The plot of a very disturbed magnetic variation day of 1-min average data of the H component recorded on 29th October 2003 at three magnetic observatories in the southern African region, Hermanus (HER), Hartebeesthoek (HBK) and Tsumeb (TSU). It is shown that between 6:00 and 8:00 UT the change of magnetic variation reached around 400 nT in the H component.

occurrences of the switch of the polarity of the dipole known as a field reversal. The magnetic north pole becomes south pole and vice versa. The field reversals take several thousand years to complete and occur at irregular intervals. The last field reversal occurred 780 000 years ago (Valet and Plenier, 2008).

The main field changes over periods varying from years to decades which occur at irregular and unpredictable rates. This phenomenon is known as secular variation, while sudden changes in the linear secular variation trend are known as secular variation impulses or geomagnetic jerks (Courtillot et al., 1978; Mandea et al., 2010). At the Earth's surface, a geomagnetic jerk in the field components is identified in the first time derivative of the geomagnetic field as a change in curvature at the time of event; the corresponding secular variation has a V-shaped feature (Courtillot et al., 1978). On the other hand, rapid geomagnetic variations (short term changes of magnetic field) are driven by external currents in the ionosphere and magnetosphere, and internal telluric currents induced in the Earth (Pulkkinen et al., 2003).

Some parts of the southern African region have exhibited a rapid decrease of the

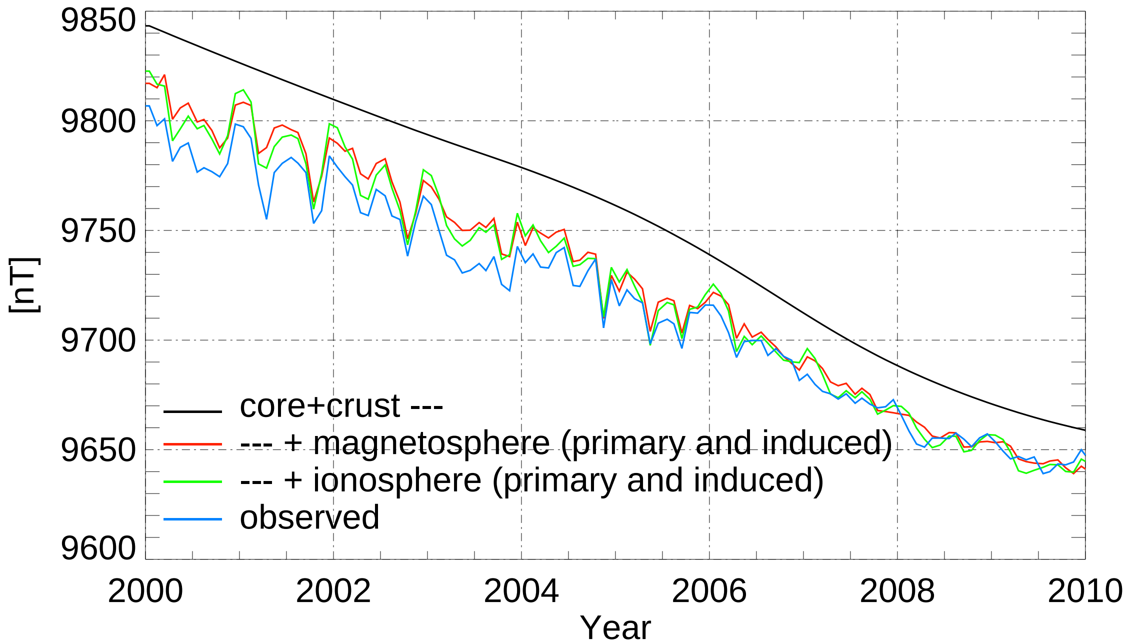

Figure 2.8: Observed and predicted monthly mean values of the X component at Hermanus magnetic observatory. The core and crustal fields are predicted from CHAOS-6 and MF7, respectively. The external field contributions (magnetospheric and ionospheric magnetic fields) are predicted from CM4.

magnetic field intensity during the last few decades, the plots of the magnitude values of X, Y and Z field components between 1941 and 2016 show long-term Earth's magnetic variations at three magnetic observatories, HER, HBK and TSU (Figs. 2.9 - 2.11). The study of the long term variation of the geomagnetic field in this region, influenced strongly by the SAA where the field is already diminished, can contribute towards a better understanding of morphology changes of this magnetic anomaly. Some geophysicists suggest that the observed magnetic field changes are associated with the growth of a patch of reverse magnetic flux at the core-mantle boundary beneath southern Africa and south Atlantic ocean (e.g. Wardinski and Holme, 2006).

2.5 Theory and global field modelling

Geomagnetic field modelling was introduced by Gauss in the 19th century using a spherical harmonic representation technique to achieve a spatial separation of contributions to the magnetic field. He used this representation to generate a least

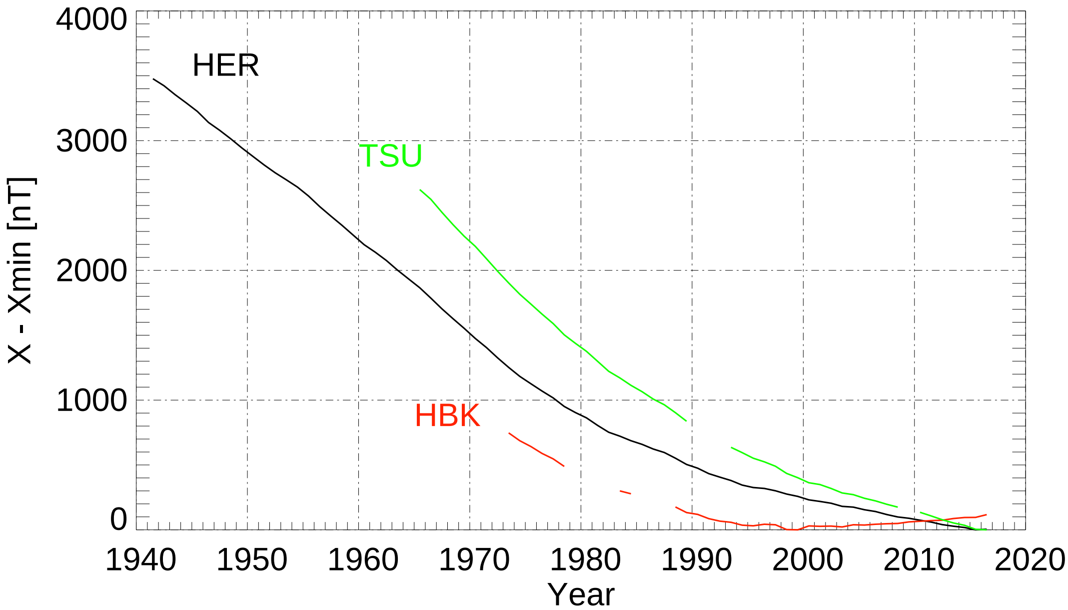

Figure 2.9: The plot showing the long-term decreases in the X component of the Earth's magnetic field over southern Africa between 1941 and 2016. The minimum values of X (Xmin) subtracted from X at HER, HBK and TSU are 9557 nT, 12263 nT and 14037 nT, respectively.

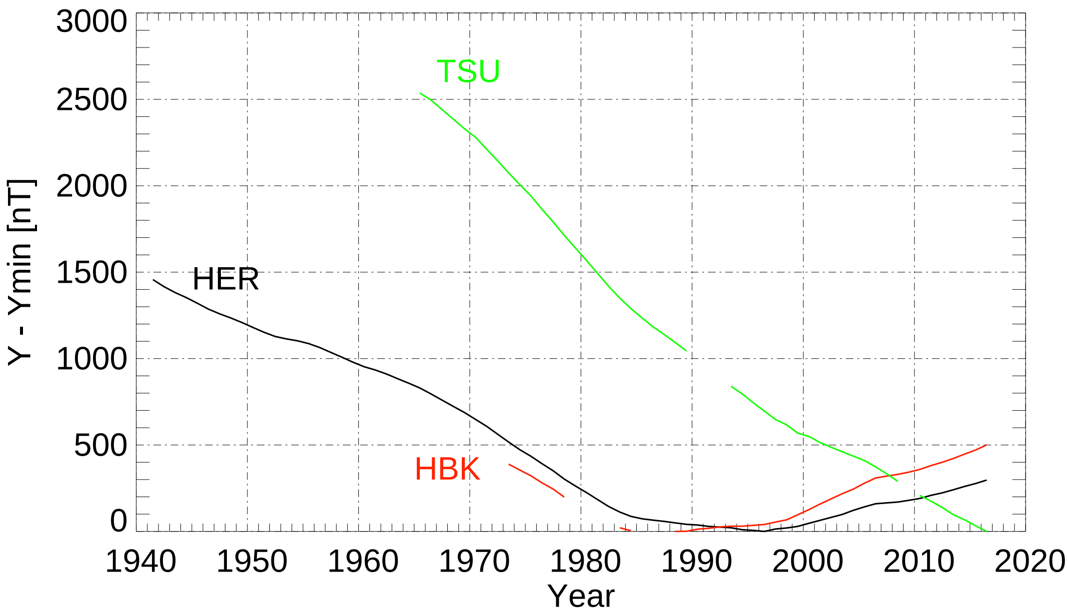

Figure 2.10: The plot showing the long-term decreases in the Y component of the Earth's magnetic field over southern Africa between 1941 and 2016. The minimum values of Y (Ymin) subtracted from Y at HER, HBK and TSU are 4308 nT, 3530 nT and 2227 nT, respectively.

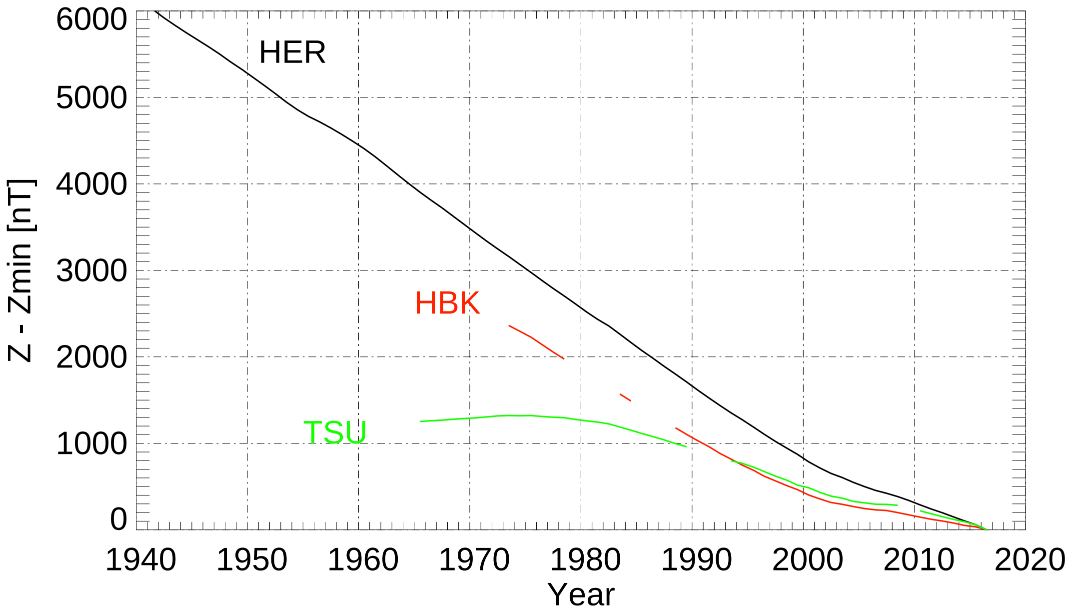

Figure 2.11: The plot showing the long-term decreases in the Z component of the Earth's magnetic field over southern Africa between 1941 and 2016. The minimum values of Z (Zmin) subtracted from Z at HER, HBK and TSU are 23233 nT, 25118 nT and 25751 nT, respectively.

squares fit to magnetic observatory data and demonstrated that the largest part of the geomagnetic field is of internal origin (Garland, 1979).

If we assume that across the boundary between the Earth and its atmosphere, where electric currents and field variations are negligible, then according to Maxwell's equations the curl of **B** equals to zero.

$$\nabla \times \mathbf{B} = \mathbf{i}\left(\frac{\partial B_z}{\partial y} - \frac{\partial B_y}{\partial z}\right) + \mathbf{j}\left(\frac{\partial B_x}{\partial z} - \frac{\partial B_z}{\partial x}\right) + \mathbf{k}\left(\frac{\partial B_y}{\partial x} - \frac{\partial B_x}{\partial y}\right) = 0, \qquad (2.2)$$

where $\mathbf{i}, \mathbf{j}$ and $\mathbf{k}$ are the three orthogonal directions. This equation requires that the magnetic field **B** can be obtained from the negative gradient of the scalar magnetic potential V.

$$\mathbf{B} = -\left[\mathbf{i}\frac{\partial V}{\partial x} + \mathbf{j}\frac{\partial V}{\partial y} + \mathbf{k}\frac{\partial V}{\partial z}\right] = -\nabla V. \qquad (2.3)$$

The other important Maxwell equation shows that the divergence of the field **B** is zero.

$$\nabla \cdot \mathbf{B} = \left[\frac{\partial B_x}{\partial x} + \frac{\partial B_y}{\partial y} + \frac{\partial B_z}{\partial z}\right] = 0. \qquad (2.4)$$

From equations (2.3) and (2.4) the Laplace's equation can be written as follows:

$$\nabla \cdot \nabla V = \nabla^2 V = 0. \tag{2.5}$$

The Laplace's equation in spherical coordinates becomes

$$\frac{\partial}{\partial r}\left(r^2 \frac{\partial V}{\partial r}\right) + \frac{1}{sin\theta}\frac{\partial}{\partial \theta}\left(sin\theta \frac{\partial V}{\partial \theta}\right) + \frac{1}{sin^2\theta}\frac{\partial^2 V}{\partial \phi^2} = 0, \tag{2.6}$$

where r, θ and ϕ are geocentric coordinates of radius (distance measured from the center of Earth), colatitude (polar angle measured from North pole) and longitude (azimuthal angle measured from the Greenwich meridian), respectively (Campbell, 2003). The equation is solved by separation of variables. The magnetic potential $V(r, \theta, \phi)$ is considered as a product of a radial and angular contributions: $V(r, \theta, \phi) = R(r) \cdot S(\theta, \phi)$. The angular function $S(\theta, \phi)$ can again be separated into a polar contribution depending on latitude and azimuthal contribution depending on longitude $S(\theta, \phi) = T(\theta) \cdot L(\phi)$.

The German mathematician Carl Frederich Gauss in his treatise *Allegemeine Theorie des Erdmagnitismus* of 1838 was the first to represent the potential of the field as a sum of spherical harmonics (Garland, 1979). When internal and external sources are considered, V is expressed as

$$V = a \sum_{l=1}^{\infty} \left[\left(\frac{r}{a}\right)^l S_l^e + \left(\frac{a}{r}\right)^{l+1} S_l^i \right], \tag{2.7}$$

where a is the Earth's radius and S_l^e and S_l^i are external and internal source terms of the potential function V.

The general solution to Laplace's equation yields a spherical harmonic representation for the geomagnetic field that provides a formal separation between internal and external sources, and within each source decomposes the field according to spherical harmonic degree l and order m.

$$V = V^i + V^e = a\sum_{l=1}^{Li}\sum_{m=0}^{l}\left(g_l^m cos(m\phi) + h_l^m sin(m\phi)\right)\left(\frac{a}{r}\right)^{l+1} P_l^m(cos\theta)$$
$$+ a\sum_{l=1}^{Le}\sum_{m=0}^{l}\left(q_l^m cos(m\phi) + s_l^m sin(m\phi)\right)\left(\frac{r}{a}\right)^{l} P_l^m(cos\theta), \quad (2.8)$$

where a is a reference radius (normally the mean radius of the Earth which is 6371.2 km), Li is the maximum degree of the internal potential Gauss coefficients g_l^m and h_l^m, and Le is that of the external potential Gauss coefficients q_l^m and s_l^m, $P_l^m(cos\theta)$ are associated Schmidt semi-normalised Legendre functions
$\int_{\phi=0}^{2\pi}\int_{0=0}^{\pi}\left(P_l^m(cos\theta)cos(m\phi)\right)^2 sin\theta d\theta d\phi = \frac{4\pi}{2l+1}$, of degree l and order m.

The field components can be derived from the potential function V. It was shown in equation 2.3 that the magnetic field is the gradient of the scalar magnetic potential, and the spherical gradient operator is

$$\nabla = \hat{r}\frac{\partial}{\partial r} + \hat{\theta}\frac{1}{r}\frac{\partial}{\partial\theta} + \hat{\phi}\frac{1}{rsin\theta}\frac{\partial}{\partial\phi}. \quad (2.9)$$

The unit vectors $\hat{r}$ points outwards from the center of Earth, $\hat{\theta}$ southwards and $\hat{\phi}$ eastwards. The field components in geocentric spherical coordinates can be derived from the scalar magnetic potential V:

$$B_\theta = -\frac{1}{r}\frac{\partial V}{\partial\theta}, \quad (2.10)$$

$$B_\phi = -\frac{1}{rsin(\theta)}\frac{\partial V}{\partial\phi}, \quad (2.11)$$

$$B_r = -\frac{\partial V}{\partial r}, \quad (2.12)$$

where X, Y and Z correspond to $-B_\theta$, B_ϕ and $-B_r$, respectively. Figure 2.12 shows the relationship between cartesian and spherical coordinates.

Each term within the spherical harmonic expansion has its spatial contribution to the magnetic field depending on its degree l and order m. The lower degrees

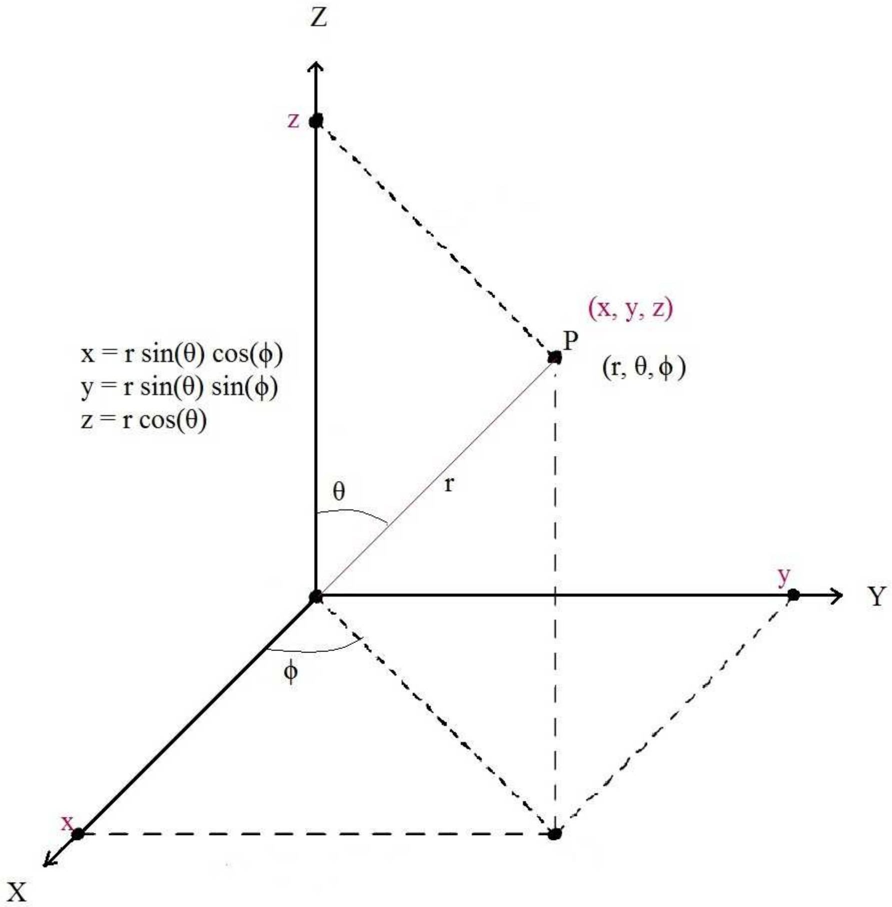

Figure 2.12: Relationship between cartesian (x, y, z) and spherical coordinates (r, θ, ϕ).

correspond to the large spatial scales, $l = 1$ is the largest spatial scale dipole part of the field (e.g. Constable and Constable, 2004). A spherical harmonic degree determines a spatial wavelength, an approximate length scale of degree l is given by $\Pi a/l$ where a is the radius of the Earth.

2.5.1 Geomagnetic power spectrum

In magnetic field modelling, the geomagnetic power spectrum that is the mean square value of the magnetic field over the surface of the Earth plays an important role (Lowes, 1974). It can be used for the comparison of field models derived from different data sets, for separating core and lithospheric field, and for identifying noise and systematic errors (Maus, 2008).

The power spectrum K_l at degree l of the magnetic field $\mathbf{B}_l$ is defined as the scalar product $\mathbf{B}_l \cdot \mathbf{B}_l$ averaged over the surface of the sphere with radius a.

$$K_l = \frac{1}{4\pi a^2} \int_0^{2\pi} \int_0^{\pi} \mathbf{B}_l \cdot \mathbf{B}_l \; a^2 sin\theta d\theta d\phi. \tag{2.13}$$

For internal sources only, the contribution of magnetic field of a particular degree l is given as follows:

$$\mathbf{B}_l = -\nabla V^i = -\nabla \left[a \left(\frac{a}{r} \right)^{l+1} \sum_{m=o}^{l} \left(g_l^m cosm\phi + h_l^m sinm\phi \right) P_l^m(cos\theta) \right]. \tag{2.14}$$

From equations 2.13 and 2.14, the power spectrum for a particular degree l is given by

$$K_l = (l+1) \sum_{m=0}^{l} \left[(g_l^m)^2 + (h_l^m)^2 \right]. \tag{2.15}$$

Figure 2.13 shows the approximate separation of three main contributions of the measured geomagnetic internal field related to the degree l by Cain et al. (1974), core field dominating the spectrum up to degree $l = 14 - 15$, crustal field and noise. According to the interpretation of the spatial magnetic power spectrum derived from MAGSAT by Langel and Estes (1982), the core field dominates the spectrum for degrees $l \leq 12$ and the crustal field dominates the spectrum for $l \geq 16$. Furthermore, Maus (2008) using his model NGDC-720, presents the magnetic power spectrum illustrating the substantial contributions of internal field by lower spherical harmonic degrees and their corresponding wavelengths.

2.5.2 Downward continuation

The study of core dynamics of the geodynamo requires to know the characteristics of the magnetic field close to its source, for instance at the core mantle boundary (CMB). The field prediction from a geomagnetic field model can be downward continued through its radial dependence, enabling the field components to be mapped at the CMB.

From equation 2.14 we deduce that

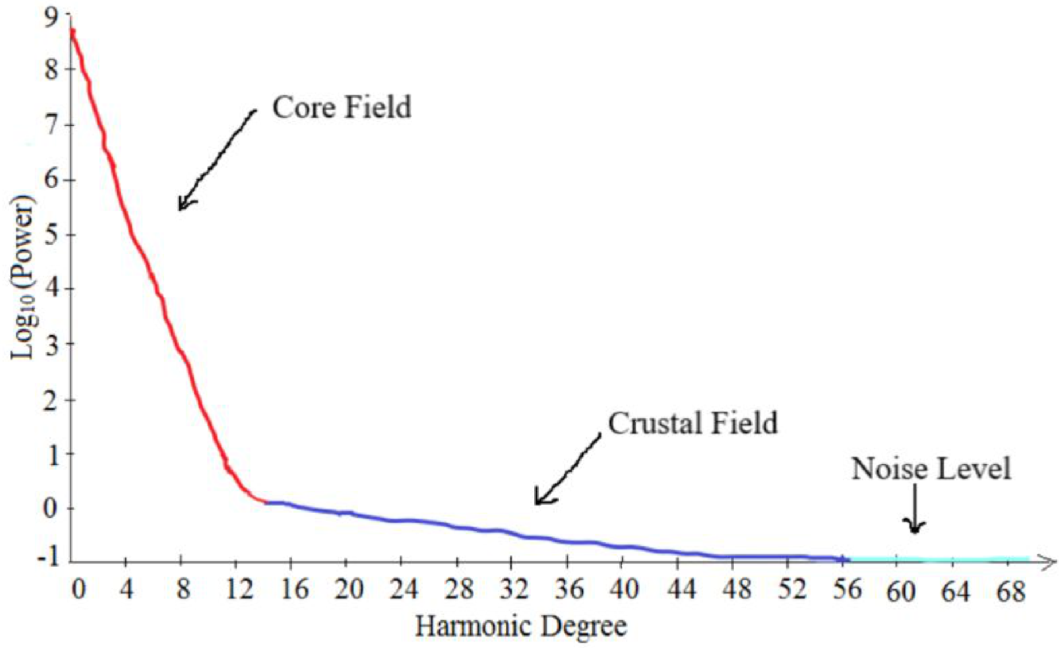

Figure 2.13: Sketch diagram of normalised power in spherical harmonics of internal geomagnetic field referring to computed geomagnetic power spectrum by Cain et al. (1974). The core field dominates the spectrum for lower degrees $l \leq 14$ and the crustal field dominates the spectrum for $l \geq 16$.

$$\mathbf{B} = -\nabla V_l(r) = \mathbf{B}_l(a)(l+1)\left(\frac{a}{r}\right)^{l+2}, \tag{2.16}$$

where $V_l(r) = V_l(a)\ (a/r)^{l+1}$.

In this case, the geomagnetic power spectrum is given by

$$K_l(r) = K_l(a)\left(\frac{a}{r}\right)^{2l+4}. \tag{2.17}$$

with $K_l(a)$ the power spectrum at Earth's surface (r = a).

The use of power spectrum at different surfaces of spheres of radius r has shown that the higher degree components of the field are more amplified than the lower degree in the process of downward continuation (Loper and Lay, 1995). The study of the field at the CMB requires care in this process of downward continuation, as it can lead to artificial amplification of short wavelengths from crustal fields. This is the reason why it is not advisable to investigate the core field at the CMB with $l > 14$, the high harmonic degree components are sharply amplified upon downward continuation and the core field at the CMB will be contaminated with the crustal field (Loper and Lay, 1995).

2.5.3 Geomagnetic secular variation modelling using splines

The description of spherical harmonics with constant Gauss coefficients ignores the time variation of the field and gives static fields at a particular time. But, as the field changes with time, most of the recent models use time-dependent Gauss coefficients. Temporal variations in the internal field are modeled by expanding the Gauss coefficients g_l^m and h_l^m in a Taylor series in time about some epoch t_0,

$$g_l^m(t) = g_l^m(t_o) + \dot{g}_l^m(t - t_0) + \ddot{g}_l^m \left(\frac{t - t_0}{2!}\right)^2 + \cdots . \quad (2.18)$$

Most of models include the first two terms on the right hand side, but the third time derivative term is included particularly for studies of geomagnetic jerks. This representation is only adequate for limited periods of years depending on the behavior of the secular variation, the use of the spline representation yield better results for any study time interval (Langel et al., 1986).

A spline is a function defined by piecewise polynomials that meet at nodal points called knots. Any spline function of given degree can be expressed as a linear combination of basis splines (or B-splines) of that degree. According to De Boor (1972) and Cox (1972), if $t_i, i = 1, \cdots, m$ are a set of strictly increasing real numbers and $s(t)$ is a polynomial of degree n-1 or less in each of the following intervals

$t \leq t_1;\ t_{i-1} \leq t < t_i,\ i = 1, \cdots, m;\ t_m < t;$

and if $s(t)$ and its derivatives up to order n-2 are continuous, then $s(t)$ is a spline function of order n or degree n-1 with knots at $\{t_i\}$. The B-spline of degree n-1 with knots at $t_{i-n}, t_{i-n+1}, \cdots, t_i$ is defined by

$$M_{ni}(t) = \sum_{k=i-n}^{i} \frac{(t_k - t)^{n-1}}{W'_{ni}(t_k)}, \quad (2.19)$$

where

$$t_+^{n-1} = \begin{cases} 0 \; for \; t < 0 \\ t^{n-1} \; for \; t \geq 0. \end{cases}$$

$$W'_{ni} = \frac{d}{dt}[W_{ni}] = \frac{d}{dt}\left[(t - t_{i-n})(t - t_{i-n+1}) \cdots (t - t_i)\right]. \tag{2.20}$$

M_{ni} is zero for $t \geq t_i$ and for $t \leq t_{i-n}$ and is strictly positive for $t_{i-n} < t < t_i$. B-splines have different names according to degree n; linear B-splines for n = 2, quadratic for n = 3, cubic for n = 4, quartic for n = 5, quintic for n = 6, etc. To fit a set of data over the time range t_l to t_h, it is necessary to specify the desired knots, $\{t_i, i = 1, \cdots, m\}$ that have to be defined in the interval (t_l, t_h) and 2n additional knots have to be added such that

$t_{-n+1} \leq t_{-n+2} \leq \cdots \leq t_0 \leq t_1$ and $t_h \leq t_{m+1} \leq \cdots \leq t_{m+n-1} \leq t_{m+n}$.

In general, B-spline of order n over a time interval $[t_l, t_h]$ is given by

$$s(t) = \sum_{i=1}^{m+n} c_i M_{ni}(t), \tag{2.21}$$

where c_i are coefficients determined by a least square fit of $s(t)$ to the data.

The expansion of Gauss coefficients using B-splines of order n to introduce time dependency in the geomagnetic field model is given by

$$g_l^m(t) = \sum_{i=1}^{m+n} a_{l,i}^m M_{ni}(t), \tag{2.22}$$

where $a_{l,i}^m$ are coefficients determined by a least square fit of $g_l^m(t)$ to the Gauss coefficients.

The temporal description of geomagnetic field using B-splines has some advantages. The B-splines can easily map localised features like temporal discontinuities, as the basis functions are non-zero only on discrete adjoining intervals (Geese et al., 2010).

2.5.4 Examples of global field models

International Geomagnetic Reference Field (IGRF)

The IGRF is a series of mathematical models of the Earth's main field and its annual rate of change (secular variation). The IGRF model is made available once every five years by the members of IAGA working group V-MOD. The latest IGRF-12 model describes the Earth's main magnetic field between epochs 1900.0 and 2015.0 with a forecast to 2020.0 (Thébault et al., 2015). The IGRF is developed using magnetic field data from satellites and from geomagnetic observatories and surveys around the world. The IGRF models the geomagnetic field $\vec{B}\,(r,\phi,\theta,t)$ as a gradient of a magnetic scalar potential $V(r,\phi,\theta,t)$.

$\vec{B}\,(r,\phi,\theta,t) = -\nabla V(r,\phi,\theta,t)$.

The magnetic scalar potential model consists of the Gauss coefficients which define a spherical harmonic expansion of V (Thébault et al., 2015).

$V(r,\phi,\theta,t) = a\sum\limits_{l=1}^{L}\sum\limits_{m=0}^{l}\left(g_l^m(t)cos\, m\phi + h_l^m(t)sin\, m\phi\right)P_l^m(cos\theta)$,

where r is radial distance from Earth's center, L is the maximum degree of the expansion, ϕ is East longitude, θ is colatitude, a is the Earth's radius, g_l^m and h_l^m are Gauss coefficients, and $P_l^m(cos\theta)$ are the Schmidt normalised associated Legendre functions of degree l and order m. The Gauss coefficients are assumed to vary linearly over the time interval specified by the model.

CHAOS model

The CHAOS model series is derived primarily from magnetic satellite data (scalar data from Ørsted, CHAMP, SAC-C and Swarm and vector data from Ørsted, CHAMP and Swarm satellites), and monthly means of 160 ground observatories. It estimates the internal geomagnetic field at the Earth's surface with high resolution in space and time. It describes the Earth's magnetic field between epochs 1999.0 and 2016.5 (Finlay et al., 2016b).

GFZ Reference Internal Magnetic Model (GRIMM)

The latest GRIMM model (GRIMM-3) is derived from CHAMP satellite magnetic vector data and observatory hourly mean vector data, spanning the period 2001 to 2010. The time dependency was represented by order six B-splines with knots one year apart spanning 2000-2011 (Lesur et al., 2010).

Potsdam Magnetic Model of the Earth (POMME)

POMME model series describes both the internal and the external magnetic fields, taking into account the variability of space weather. It represents the core, crust and external fields (Maus et al., 2006). The temporal variations of the internal field are produced by a piece-wise linear representation of the Gaussian spherical harmonic coefficients of the magnetic potential. The latest POMME model series, Pomme-10, was derived from CHAMP satellite vector magnetic measurements between July 2000 and September 2010, Oersted satellite total field measurements between January 2010 and June 2014 and Swarm satellite vector magnetic measurements between December 2013 and November 2015.

Lithospheric Magnetic Field Model (MF6)

The sixth-generation lithospheric magnetic filed model (MF6) was built using CHAMP satellite data recorded between 2004 and 2007. MF6 was estimated to a spherical harmonic degree 120, corresponding to 333 km wavelength resolution (Maus, 2008). The latest lithospheric magnetic filed model MF7 was produced using CHAMP measurements from May 2007 to April 2010 and estimated to a harmonic degree 133, corresponding to 300 km wavelength resolution.

High-resolution lithospheric magnetic field model (LCS-1)

Olsen et al. (2017) have recently used both CHAMP and Swarm satellite observations to develop a high-resolution global lithospheric magnetic field model (LCS-1). LCS-1 is based on four years (September 2006 - September 2010) of magnetic observations taken by CHAMP satellite at altitudes lower than 350 km, and magnetic observations taken by two lower Swarm satellite Alpha and Charlie between April 2014 and December 2016. The model is determined entirely from magnetic gradient data, north-south and east-west gradient data. It is the first satellite-derived global lithospheric model with degrees above n = 133, and its maximum degree is n = 185 (Olsen et al., 2017).

2.6 Regional field modelling

The regional geomagnetic field modelling plays a major role when there is a need to represent or describe the Earth's magnetic field over a restricted area with fixed boundaries on the Earth's surface, given data over this area. Regional models that are usually based on denser data than the global models are more accurate and representative over their regions than the global models and they can be made available easily and timely (Haines, 1990).

Technically, regional models are able to represent wavelengths shorter than those in the global models. They are recommended in studying the small scale features of the geomagnetic field variation.

Different methods used in the regional geomagnetic field modelling that include surface polynomials, rectangular harmonic analysis, spherical cap harmonic analysis and harmonic splines.

2.6.1 Surface Polynomials

The use of surface polynomials was the first analytical method to produce regional models. Polynomials are functions only of colatitude θ and longitude ϕ, and not of radial distance r. For example a geomagnetic field component C (e.g. horizontal field component H) can be represented by the following surface polynomial

$$\begin{aligned} C(\theta,\phi) = c_0 + c_1(\theta-\theta_0) + c_2(\phi-\phi_0) + c_3(\theta-\theta_0)^2 \\ + c_4(\theta-\theta_0)(\phi-\phi_0) + c_5(\phi-\phi_0)^2 + \cdots , \end{aligned} \tag{2.23}$$

where $c_0, c_1, c_2, c_3, c_4, c_5, \cdots$ are polynomial coefficients, (θ_0, ϕ_0) is the centre of the area of interest and θ and ϕ are colatitude and longitude, respectively.

The polynomial degree is usually chosen to be very low (3 or less) to depict the long minimum wavelengths in the models, but this may or may not be desirable depending on the application (Haines, 1990). Without any altitude dependence, the polynomial modelling technique requires that data be reduced to the same altitude (Nahayo and Kotzé, 2012).

2.6.2 Rectangular Harmonic Analysis

The Rectangular harmonic Analysis (RHA) technique is applied over a small portion (less than 2,000,000 km^2) of the Earth's surface that can be considered as a two-dimensional surface. Alldredge (1981) argues that the limitation of the good resolution of the geomagnetic field by spherical harmonic analysis can be overcome by using rectangular harmonic analysis in successively smaller areas so that the data are more fully utilized. In the region of interest, data can be collected at different close altitudes and converted to x, y and z coordinates. The solution of Laplace's equation in x, y and z coordinates gives the potential expression in the form of a double Fourier series of sines and cosines multiplied by an exponential of z (Haines, 1990):

$$\begin{aligned} V = Ax + By + Cz + \sum_{m=1}^{M_0}\{a_0^m cos(mx) + b_0^m sin(mx)\}exp\{-k_x mz\} \\ + \sum_{n=1}^{N_0}\{a_n^0 cos(ny) + c_n^0 sin(ny)\}exp\{-k_y nz\} \\ + \sum_{m=1}^{M}\sum_{n=1}^{N}\{a_n^m cos(mx)cos(ny) + b_n^m sin(mx)cos(ny) \\ + c_n^m cos(mx)sin(ny) + d_n^m sin(mx)sin(ny)\}exp\{-\sqrt{(k_x m)^2 + (k_y n)^2 z}\}, \end{aligned} \quad (2.24)$$

where A, B, C, a_0^m, b_0^m,a_n^0, c_n^0, a_n^m, b_n^m,c_n^m and d_n^m are coefficients,

the negative gradient of which gives the magnetic field $\boldsymbol{B}$:

$$\boldsymbol{B} = -\boldsymbol{\nabla} V. \quad (2.25)$$

The x, y, and z are ordinary cartesian, or rectangular coordinates with z vertically upward. The origin of the coordinate system is usually taken at the center of the region of data to be modelled.

Haines (1990) stated that one of the limitations of this technique is that the terms Ax, By and Cz do not tend to zero as z tends to infinity, as a potential should if it is due to internal sources. Furthermore, it is not possible to incorporate the measurements from different altitudes because the exponential does not provide the natural decrease of a Newtonian potential field (Thébault, 2003).

2.6.3 Spherical Cap Harmonic Analysis

The method of Spherical Cap Harmonic Analysis (SCHA) is the analogue of the method of ordinary spherical harmonic analysis, being applied to a spherical cap rather than to the whole sphere (Haines, 1985). The potential for internal sources is expressed as:

$$V(r,\theta,\phi) = a\sum_{k=0}^{K}\sum_{m=0}^{k}(\frac{a}{r})^{n_k(m)+1}P^m_{n_k(m)}(cos\theta)[g^m_k cos(m\phi) + h^m_k sin(m\phi)], \qquad (2.26)$$

where r, θ and ϕ represent the geocentric spherical coordinates; radius, colatitude, and longitude respectively; a is the reference radius; $P^m_{n_k(m)}$ $(cos\theta)$ is the associated Legendre function, with integral order m and real degree $n_k(m)$; K is the ordering index; g^m_k and h^m_k are the spherical cap coefficients. An improvement in the technique (revised spherical cap harmonic analysis) by Thébault (2003) has the ability to incorporate the measurements collected between the ground altitude and the satellite altitude while still complying with Maxwell's equations (Thébault et al., 2004, 2006).

2.6.4 Wavelet Analysis

The method of wavelet analysis requires generally a small number of coefficients to represent functions and large data-sets accurately. This method was introduced by Holschneider et al. (2003) to deal with some challenges found in other geomagnetic field modelling methods like SHA and SCHA where there are limitations linked to the precision in observations, their distributions and to the use of truncated spherical harmonic expansions. This method can be used for global as well as regional field models.

The harmonic splines method, that is used in developing global and regional models, is presented in detail in the next section as it was used in the development of the geomagnetic core field model in this study.

2.7 Harmonic Splines

The harmonic splines were first introduced by Shure et al. (1982) for global magnetic field modelling, but it is also suitable for regional modelling. Harmonic splines were first used by Shure et al. (1985) for global magnetic field modelling using Magsat data. Shure et al. (1985) compared the traditional spherical harmonic modelling technique with the harmonic spline technique; they both rely on expansions in terms of harmonic functions outside the source region. The standard method uses the familiar spherical harmonics as its basis as developed by Gauss in 1838 (Garland, 1979). The geomagnetic potential V can be written as

$$V(r,\theta,\phi) = a \sum_{l=1}^{L} \left(\frac{a}{r}\right)^{l+1} \sum_{m=-l}^{l} \beta_l^m Y_l^m(\theta,\phi), \tag{2.27}$$

where r is radial distance from the Earth's center, θ colatitude, ϕ longitude, a radius to which the model is referred ($r \leq a$), L is the maximum degree of expansion of internal sources, Y_l^m fully normalized complex spherical harmonics (Shure et al., 1985), and β_l^m complex coefficients describing the specific model. The magnetic field $\mathbf{B}$ can be expressed as the gradient of a scalar potential, with $\mathbf{B} = -\nabla V$. The vector components of the field measured are linearly related to the model, as presented by the $\{\beta_l^m\}$. Core (main field) models are then constructed by performing a least square fit for the $\{\beta_l^m\}$ complex coefficients and choosing some degree L as the cut-off between core sources and other signals or noise. In contrast to the traditional technique, for harmonic spline models, instead of determining the model which minimizes the mean square residual, we seek the one which is least rough or, equivalently, maximally smooth. Harmonic splines, in close analogy to cubic splines on an interval (Ahlberg et al., 1967), are the functions resulting from the requirement that we fit the data with the smoothest possible model and then interpolate using a finite linear combination of known functions. The harmonic spline models naturally possess a spherical harmonic development, but the expansion contains an infinite number of terms, which is not the case for the traditional spherical harmonic technique.

The use of harmonic splines to model magnetic data requires solving a square system of equations with dimension equal to the number of data points (Shure et al., 1982). Geese et al. (2010) states that the harmonic spline functions satisfy Laplace's equations, therefore allowing a potential field approach putting together the individual components in a physical meaningful way. Lesur (2006) and Geese et al. (2010) describe a representation of the model that is used in this study, highlighting localised constraints connected with local basis functions.

Considering ψ as the integral of the square of the second derivative of the radial component of the magnetic field B_r over a sphere Ω defined by a radius a

$$\psi = \int_{\Omega} \left|\nabla_h^2 B_r\right|^2 d\Omega, \tag{2.28}$$

where the horizontal derivative $\nabla_h = \nabla - \partial_r/r$ affects only the angular dependencies. The second derivative corresponds to the curvature of a function, then ψ can be considered as a measure of smoothness. If ψ is minimised we get a radial magnetic field that varies only sparsely on the reference sphere.

The radial field component B_r can be written as

$$B_r = \sum_{l,m}^{L} (l+1)\, g_l^m \left(\frac{a}{r}\right)^{l+2} Y_l^m(\theta, \phi), \tag{2.29}$$

where $\sum_{l,m}^{L}$ denotes the double sum over l and m, and the g_l^m are the Gauss coefficients. The orthogonality relation of the spherical harmonics Y_l^m over the unit sphere Ω can be written as

$$\int_{\Omega} Y_l^m Y_{l'}^{m'} d\Omega = \frac{4\pi}{2l+1} \delta_{mm'} \delta_{ll'}, \tag{2.30}$$

where $\delta u'$ is the Kronecker delta that equals 1 if $l = l'$ and zero else. We express the horizontal derivative of a spherical harmonic as

$$\nabla_h^2 Y_l^m = -l(l+1) Y_l^m. \tag{2.31}$$

The integral in Equation 3.27 reduces to

$$\psi = \sum_{l,m}^{L} (l+1)^4\, l^2 \frac{4\pi}{2l+1} (g_l^m)^2. \tag{2.32}$$

The expression of this integral as scalar product is

$$< \mathbf{a}, \mathbf{b} >= \sum_{l,m}^{L} 4\pi \frac{l^2(l+1)^4}{2l+1} a_l^m b_l^m. \tag{2.33}$$

Considering a dataset of N measurements of the radial component of the magnetic field at given points (θ_i, ϕ_i, r_i), $i = 1, 2, 3, \cdots, N$; we can write each of these values as scalar product in the form of Equation 2.33.

$$\begin{aligned} B_r(\theta_i, \phi_i, r_i) &= \sum_{l,m}^{L} (l+1) \left(\frac{a}{r_i}\right)^{l+2} Y_l^m(\theta, \phi) g_l^m \\ &= \sum_{l,m}^{L} 4\pi \frac{(l+1)^4 l^2}{2l+1} k_{li}^m g_l^m \\ &= < \mathbf{k_i}, \mathbf{g} > \end{aligned} \tag{2.34}$$

where $\mathbf{k_i} = [K_{li}^m]_{\{l,m\}}$ and $\mathbf{g} = [g_l^m]_{\{l,m\}}$. The vector components k_{li}^m are defined as

$$k_{li}^m = \frac{2l+1}{4\pi(l+1)^3 l^2} \left(\frac{a}{r_i}\right)^{l+2} Y_l^m(\theta_i, \phi_i). \tag{2.35}$$

The vectors $\mathbf{g}$ can be written as a linear combination of the vectors $\mathbf{k_i}$

$$\mathbf{g} = \sum_{j=1}^{N} \alpha_j^r \mathbf{k_j}. \tag{2.36}$$

Equations 2.34 and 2.36 are used to write the following linear system

$$[B_r(\theta_i, \phi_i, r_i)]_{\{i\}} = \mathbf{\Gamma} \cdot \alpha^{\mathbf{r}}, \tag{2.37}$$

where $\alpha^{\mathbf{r}} = [\alpha_j^r]_{\{j\}}$ and the matrix $\mathbf{\Gamma}$ is defined as

$$\Gamma_{ij} = \sum_{l,m}^{L} 4\pi \frac{(l+1)^4 l^2}{2l+1} k_{li}^m k_{lj}^m. \tag{2.38}$$

The vector $\alpha^{\mathbf{r}}$ is known by solving the linear system in Equation 2.37, and it is used to compute magnetic field values at any point (θ, ϕ, r).

$$
\begin{aligned}
B(\theta,\phi,r) =& -\nabla a \sum_{l,m}^{L} g_l^m \left(\frac{a}{r}\right)^{l+1} Y_l^m(\theta,\phi) \\
=& -\nabla a \sum_{l,m}^{L} \left\{ \sum_j k_{lj}^m \alpha_j^r \right\} (\frac{a}{r})^{l+1} Y_l^m(\theta,\phi) \\
=& -\nabla \sum_j \alpha_l^r \left\{ a \sum_{,m}^{L} \frac{2l+1}{4\pi l^2 (l+1)^3} \times \left(\frac{a}{r_j}\right)^{l+2} Y_l^m(\theta_j,\phi_j) \left(\frac{a}{r}\right)^{l+1} Y_l^m(\theta,\phi) \right\}.
\end{aligned}
\tag{2.39}
$$

In a compact form, Equation 2.39 can be written as

$$
B(\theta,\phi,r) = -\sum_j \alpha_j^r \nabla F_j^{L_r}(\theta,\phi,r), \tag{2.40}
$$

where the function $F_j^{L_r}(\theta,\phi,r)$ is defined as

$$
F_j^{Lr} = a \sum_{l,m}^{L} f_l(l+1) \left[\left(\frac{a}{r_j}\right)^{l+2} Y_l^m(\theta_j,\phi_j) \right] \left(\frac{a}{r}\right)^{l+1} Y_l^m(\theta,\phi), \tag{2.41}
$$

where $\sum_{l,m}^{L}$ denotes the double sum $\sum_{l=1}^{L}\sum_{m=-l}^{l}$, θ, ϕ and r are the geocentric colatitude, longitude and radius, a is the Earth's reference radius (6371.2 km), $Y_l^m(\theta,\phi)$ are the usual Schmidt semi-normalised spherical harmonic functions of degree l and order m, L is the maximum degree of expansion of internal sources and $f_l = \frac{2l+1}{4\Pi(l+1)^4 l^2}$. The parameter f_l makes the resulting model to minimise the functional ψ defined in Equation 2.28 and imposes smooth characteristics on the derived field model (Geese et al., 2010).

The functions F_j^{Lr} are determined at the measurement positions (θ_j, ϕ_j, r_j) with $j = 1, \cdots, N$; and they allow an exact fit to the radial magnetic field data as long as the data contain the spherical harmonics degree lower or equal to L. The difference of this method and the one of harmonic spline presented in Shure et al. (1982) lies in the use of finite series to define F_j^{Lr}. The choice of a maximum degree L (i.e. working with truncated series) allows the results to be comparable to global models based on spherical harmonics.

The functions for the horizontal components of the magnetic field are derived similarly.

$$F_j^{L\theta} = a \sum_{l,m}^{L} f_l \left[\left(\frac{a}{r_j} \right)^{l+2} \partial_\theta Y_l^m(\theta_j, \phi_j) \right] \left(\frac{a}{r} \right)^{l+1} Y_l^m(\theta, \phi), \tag{2.42}$$

$$F_j^{L\phi} = a \sum_{l,m}^{L} f_l \left[\left(\frac{a}{r_j} \right)^{l+2} \frac{\partial \phi}{sin\theta_j} Y_l^m(\theta_j, \phi_j) \right] \left(\frac{a}{r} \right)^{l+1} Y_l^m(\theta, \phi), \tag{2.43}$$

Using all three base functions ($F_j^{Lr}(\theta, \phi, r)$, $F_j^{L\theta}(\theta, \phi, r)$ and $F_j^{L\phi}(\theta, \phi, r)$), the magnetic field **B** can be written as

$$B(\theta, \phi, r) = -\bigtriangledown \left\{ \sum_j \alpha_j^r F_j^{Lr}(\theta, \phi, r) + \sum_j \alpha_j^\theta F_j^{L\theta}(\theta, \phi, r) + \sum_j \alpha_j^\phi F_j^{L\phi}(\theta, \phi, r) \right\}, \tag{2.44}$$

In a compact form, the Earth's magnetic field is given by the following equation

$$B(\theta, \phi, r) = -\sum \alpha_j \nabla F_j^L (\theta, \phi, r), \tag{2.45}$$

where α_j are the coefficients of the model and $\nabla F_j^L(\theta, \phi, r)$ are harmonic spline functions.

Considering the above magnetic field representation, and provided that the functions are defined at the observation locations, it is feasible to fit three-component data as a potential field, as long as the data contain spherical harmonic degrees lower than or equal to L. Equation 2.44 is time independent, it defines a linear system of equations where α_j^r, α_j^θ and α_j^ϕ are unknowns, the data are magnetic field measurements and the matrix consists of combinations of the functions F_j^{Lr}, $F_j^{L\theta}$ and $F_j^{L\phi}$. This linear system is square and can be solved directly. For time representation, we expand each of the coefficients α_j^r, α_j^θ and α_j^ϕ on a basis of B-splines, i.e piecewise polynomials between spline knots:

$$\alpha_j^r = \sum_{k=1}^{n_{knots}} \beta_{k_j}^r b_k(t) \tag{2.46}$$

and α_j^θ and α_j^ϕ are expanded similarly.

The maximum degree of expansion of internal sources L was chosen to be 12 for the core field to decrease the contributions of the lithospheric field. This degree is small enough to give out mostly the field contributions from the core with the minimum wavelength of 4000 km. A trade-off between the misfit (Table 2.2) and temporal resolution was considered. For the temporal representation, order six B-splines with spline knots spaced at 1 year intervals are used. A linear system is built from equations 2.44 and 2.46 and we solve for the unknown model parameters $\beta_{k_j}^r$, $\beta_{k_j}^\theta$ and $\beta_{k_j}^\phi$ as defined in equation 2.46.

Table 2.2: The RMS values of the misfit errors between model and satellite data for different L values when order six B-splines are used as temporal basis functions (Nahayo et al., 2015).

L	X (nT)	Y (nT)	Z (nT)
11	12.64	6.91	6.53
12	12.63	6.90	6.54
13	12.60	6.89	6.50
14	12.60	6.88	6.50

Chapter 3

Data selection and processing

In this study, satellite and ground-based data are used to develop regional core and lithospheric magnetic field models. At satellite altitude, the internal sources include contributions from the core, the lithosphere and the ionosphere; and external sources consist of magnetospheric contributions. The use of appropriate data selection criteria makes possible to isolate most of external sources to properly model the static and time varying internal Earth's magnetic field. The selection of data was performed by choosing a time interval restricting data selection to only night times as during day times data are contaminated by the solar quiet day variation. This regular variation with a fundamental period of 24 hours is specially large at mid to low latitudes and depends on latitude, local time, season and solar cycle. It is caused primarily by electrical currents in the ionosphere and magnetosphere (Macmillan and Droujinina, 2007).

3.1 Geomagnetic indices

Geomagnetic indices measure magnetic activity that occurs, usually, over periods of time of less than a few hours (e.g. K-index is computed for a 3-h interval) recorded by magnetometers at ground-based observatories (Mayaud, 1980). The indices measure the geomagnetic variations that have their origin in the Earth's ionosphere and magnetosphere. Geomagnetic indices may be classified into three main categories looking at the region from which the data records come that are used in their derivation. These include the equatorial Dst index for low-latitude, Kp (including Kn, Ks, Km and their corresponding a-type indices) for mid-latitude, and AE (including the related indices AU and AL) for auroral zone disturbances (Allen, 1982).

3.1.1 K and Kp indices

A 3-hour K integer index was first introduced by (Bartels et al., 1939). The regular variation (S_R) is removed from horizontal component (H) observatory data and the range of the remaining data occuring over 3-hour interval is measured. This is then converted to a quasi logarithmic K integer from 0 to 9 according to a scale specific to each observatory. Kp measures planetary-scale magnetic activity and it is derived from 13 sub-auroral observatories (Menvielle and Berthelier, 1991).

3.1.2 Auroral electrojet index AE

In 1966, Davis and Sugiura (1966) introduced the AE index as a measure of the global auroral electrojet activity. Electrojets are the currents that flow within the low ionosphere and are linked to the remote current sources in the magnetosphere through field-aligned currents. Given the directions of these electrojets (eastward and westward), the H component is the most appropriate for monitoring their temporal variations (Mayaud, 1980). Taking the average quiet time level in the H component as a reference, at the auroral magnetic observatories the deviations ΔH from this reference level are directly related to the magnitude of the eastward and westward electrojets for $\Delta H > 0$ and $\Delta H < 0$, respectively (Mayaud, 1980). The AU and AL indices monitor respectively, the eastward and westward electrojets.

3.1.3 Disturbance Storm-Time (Dst) index

The studies in geomagnetic storms have shown that at equatorial and mid-latitudes, there is a decrease in H component during magnetic storm that can closely be represented by a uniform magnetic field parallel to the geomagnetic dipole axis and directed towards south (Moos, 1910). The initial phase of the storm is marked by the sudden increase in H for a few hours which is known as the sudden storm commencement (ssc). Then a large decrease begins in H that indicates the development of the main phase of the storm. The severity of the disturbance is represented by the magnitude of the decrease in H and this is often explained as an enhancement of a westward magnetospheric ring current, whose Dst index, that monitors the disturbance field, is derived continuously as a function of Universal Time (Sugiura, 1964). The Dst index is an index of magnetic activity derived from a network of four mid-latitude geomagnetic observatories: Hermanus (34.4°S, 19.2°E), Kakioka (36.2°N, 140.2°E), San Juan (15.6°N, 87.2°W) and Honolulu (21.3°N, 157.8°W) (fig. 3.1).

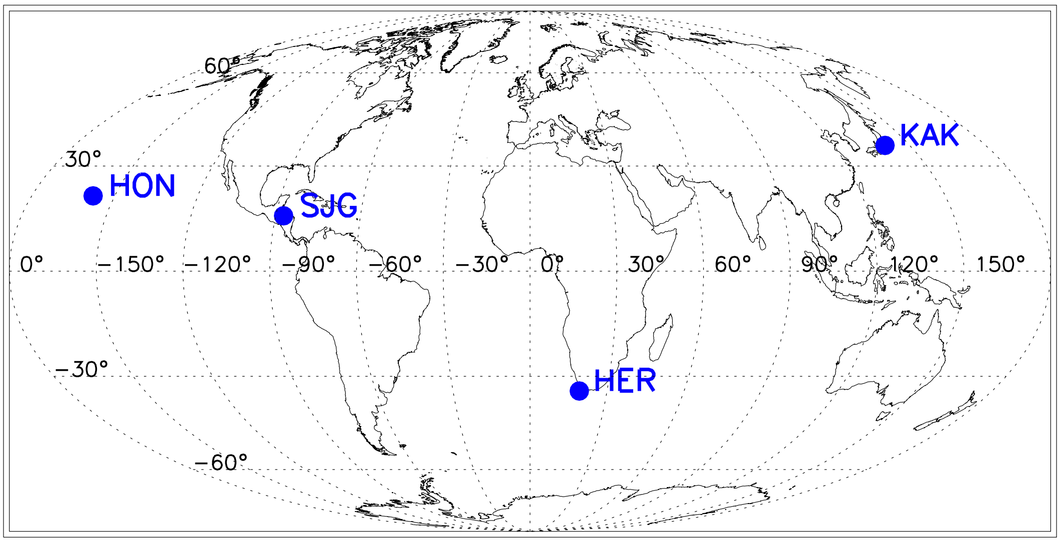

Figure 3.1: The Dst network observatories: Hermanus (HER), Kakioka (KAK), San Juan (SJG) and Honolulu (HON).

The external field has various sources with the domination of the ring current at mid-latitudes. The Dst index, which is a proxy of the strength of the ring current, was then solely used in quiet data selection (Saba et al., 1997) and selecting the night side data using an appropriate night time interval. Dst index has sometimes positive variations mainly caused by the compression of the magnetosphere from the increases of the solar wind pressure where the magnetopause current dominates the ring current, showing magnetic activities. A compromise was reached between getting quiet time data and a good data coverage before setting the data selection criterion of Dst values to be between -20 and 0 nT.

3.2 CHAMP satellite data

3.2.1 CHAMP satellite mission

The CHAMP(CHAllenging Mini-satellite Payload) mission was proposed by the German researcher Dr. Christoph Reigber in 1994 at GeoForschungsZentrum Postdam (GFZ), Germany. The German small satellite CHAMP was launched on July 15, 2000 into a circular, near polar 454 km altitude orbit. Its mission was to map the Earth's gravitational geopotential by the analysis of observed orbit perturbations, the magnetic geopotential via on-board magnetometry, and perform atmosphere profiling by GPS radio occultation measurements (Reigber et al., 2002).

It was designed to have an initial altitude of 454 km decaying to about 300 km over the mission's lifetime of 5 years. But, the CHAMP satellite continued to record data until 2010. The low altitude orbit supports the spatial resolution of the geopotential field whereas the long mission duration helps to recover temporal field variations (Reigber et al., 2002).

The satellite consisted of a trapezoidal body of 0.6 m height, 4 m length and 1.6 m/0.4 m bottom/top width and 4 m long boom extending in the flight direction, and the mass was 522 kg including 32 kg of payload mass (Lühr, 2001).

3.2.2 Vector satellite data

The data selection was performed on CHAMP vector magnetic data of 1 Hz sampling interval recorded between approximately 300 km and 380 km of altitude. The quiet night data recorded between 2001 and 2010 were selected over the southern Africa region. Only quiet time data corresponding to a Dst index between -20 and 0 nT measured during local night times between 19:00 - 06:00 were considered. The model input data were obtained by creating data bins of $0.2^\circ \times 0.2^\circ$ on a grid of chosen degrees (less or equal to 2°) of latitude and longitude. The averaging process of data recorded at the same epoch (within 1 day) made it possible to obtain data values at the same geographic location at different epochs and different altitudes (the data center is regarded as a geomagnetic repeat station only with the difference that the data might be measured at different altitudes between 300 and 380 km).

it was found that the altitudes of the values in the same bin recorded at the same epoch are very close (the average scatter value less than 10 m) and the altitude of the averaged value was calculated by averaging altitudes of individual values in the bin recorded at the same epoch.

3.3 Ground data

The hourly mean data derived from 1-minute data for four continuous recording magnetic observatories Hermanus (HER), Tsumeb (TSU), Hartebeesthoek (HBK) and Keetmanshop (KMH) (also available at http://www.intermagnet.org) were selected using the same criteria for satellite data selection. The monthly mean values were computed by calculating the average of the obtained hourly values to get the input data for the model. The data collected during field survey over southern African region at 38 geomagnetic repeat stations were also used as input data (also available at http://www.geomag.bgs.ac.uk/
data_service/data/). The collection of magnetic data at the repeat stations is conducted by running a LEMI-008 fluxgate vector magnetometer over night and taking absolute observations in the evening and morning using the standard absolute equipment of a Declination/Inclination-fluxgate magnetometer (http://www.bartington.com/) and an Overhauser magnetometer (http://www.gemsys.ca/). The computed baseline values are applied to 10 night hour variations between 20:00 - 06:00 LT to obtain the main field values at this geomagnetic repeat station for this particular day when the survey is conducted (Korte et al., 2007).

Chapter 4

Secular variation over southern Africa between 2001 and 2010

The contents of this chapter are the preprint of a paper that has been published in the Journal of Atmospheric and Solar-Terrestrial Physics (2015, volume 123, pages 13-21, https://doi.org/10.1016/j.jastp.2014.12.002). I conceived this study in discussion with my supervisor Pieter B Kotzé and I acknowledge valuable contributions from Susan J Webb in its final stage. Work for drafting this manuscript was completed by me, with the remaining authors contributing to improve the manuscript.

A southern Africa harmonic spline core field model derived from CHAMP satellite data

E. Nahayo[1,3], P. B. Kotzé[1] and H. McCreadie[2]

[1]South African National Space Agency (SANSA) Space Science, PO Box 32, Hermanus, South Africa

[2] University of KwaZulu-Natal, Private Bag X54001, Durban, South Africa

[3]University of the Witwatersrand, PO Box 2050, Johannesberg, South Africa

Tel: +27 28 312 1196, Fax: +27 28 312 2039, Email address: enahayo@sansa.org.za

4.1 Abstract

The monitoring of the Earth's magnetic field time variation requires a continuous recording of geomagnetic data with a good spatial coverage over the area of study. In southern Africa, ground recording stations are limited and the use of satellite data is needed for the studies where high spatial resolution data is required. We show the fast time variation of the geomagnetic field in the southern Africa region by deriving an harmonic spline model from CHAMP satellite measurements recorded between 2001 and 2010. The derived core field model, the Southern Africa Regional Model (SARM), is compared with the global model GRIMM-2 and the ground based data recorded at Hermanus magnetic observatory (HER) in South Africa and Tsumeb magnetic observatory (TSU) in Namibia where the focus is mainly on the long term variation of the geomagnetic field. The results of this study suggest that the regional model derived from the satellite data alone can be used to study the small scale features of the time variation of the geomagnetic field where ground data is not available. In addition, these results also support the earlier findings of the occurrence of a 2007 magnetic jerk and rapid secular variation fluctuations of 2003 and 2004 in the region.

4.2 Introduction

The CHAMP satellite mission provided very high quality vector measurements of the Earth's magnetic field which has enabled many studies about its external and internal sources (Reigber et al., 2005). The majority of the magnetic field measured at the Earth's surface is due to a geodynamo mechanism operating in the liquid, metallic, outer core. It is generated in the fluid outer core by the self-exciting dynamo process. Electrical currents flowing in the slowly moving molten iron core generate this magnetic field known as the main or core field (Olsen and Mandea, 2007). It exhibits long-term changes which occur at irregular and unpredictable rates. In addition to this dominant contribution of the Earth's magnetic field, the lithospheric field arising from the magnetisation carried by rocks in the crust and upper mantle must also be considered. Another significant contribution is that of external sources which originate in the ionosphere and magnetosphere. In this study, the focus is put on the core field. The understanding of its time variations can shed light on the processes that drive them inside the liquid outer core.

The Earth's main magnetic field is not constant but changes non-uniformly with time. This phenomenon is known as secular variation, while sudden changes in the linear secular variation trend are known as geomagnetic jerks or secular variation impulses (Mandea et al., 2010). Torsional oscillations in the fluid outer core have been used to explain the occurrence of geomagnetic jerks (Bloxham et al., 2002) on a decadal timescale. No direct information, however is available for timescales of less than a couple of years (Mandea and Olsen, 2009). It has been shown that changes in the magnetic field occurring over only few months, as well as the fluid flow at the top of the core that is believed to generate them, can be resolved(Olsen and Mandea, 2008). These changes involving rapid variations over timescale of few months have been called rapid secular variation fluctuations, or, in short, rapid fluctuations by Mandea and Olsen (2009). Chulliat et al. (2010), using the annual differences of monthly means of the Y and Z components from 1997 to 2009 at five INTERMAGNET magnetic observatories (Ascension, Kourou, MBour, Tamanrasset and Tsumeb), showed the occurrence of a 2007 jerk in Y and Z components. Kotzé (2011) used ground data collected at Hermanus observatory (HER) to show a 2007 jerk in X, Y and Z. The polynomial modelling of CHAMP satellite data measured between 2001 and 2005 over southern Africa identified rapid secular variation fluctuations during 2003 and 2004 in all components X, Y and Z (Nahayo and Kotzé, 2012)

The considered area of study, southern Africa, is located in a region where the rapid decrease of magnetic field intensity is observed at the Earth's surface stretching across southern Africa and the south Atlantic Ocean (Lesur et al., 2008). This coincides with a region known as the South Atlantic Anomaly where the field is already very weak by approximately 30% compared to other locations at similar latitudes. In this region, the ground recording stations are limited and the use of high quality satellite data is needed for studies where spatial and temporal resolution data is required.

An attempt to study the time variation of the geomagnetic field in this region using the harmonic splines technique was done previously using only ground-based data (Geese et al., 2009). In this paper, the results of the use of harmonic splines technique on CHAMP satellite data recorded between 2001 and 2010 were compared with ground-based data recorded in the same period at Hermanus and Tsumeb magnetic observatories and the global GRIMM-2 model (Lesur et al., 2010).

4.3 Method

Our aim is to derive a core field regional model over southern Africa using harmonic splines. The harmonic splines were first introduced by Shure et al. (1982) for global magnetic field modelling, but it is also suitable for regional modelling. Modelling magnetic data by harmonic splines requires solving a square system of equations with dimension equal to the number of data points (Shure et al., 1982). The harmonic spline functions satisfy Laplace's equations, therefore allowing a potential field approach combining the individual components in a physical meaningful way (Geese et al., 2011). The following description of the model starts with the spatial aspect and ends with the introduction of a time dependency in the model.

According to Lesur (2006) and Geese et al. (2009), the Earth's magnetic field can be written as:

$$B(\vartheta,\varphi,r) = -\bigtriangledown \left\{ \sum_j \alpha_j^r F_j^{Lr}(\vartheta,\varphi,r) + \sum_j \alpha_j^\vartheta F_j^{L\vartheta}(\vartheta,\varphi,r) + \sum_j \alpha_j^\varphi F_j^{L\varphi}(\vartheta,\varphi,r) \right\}, \tag{4.1}$$

where the functions $F_j^{Lr}(\vartheta,\varphi,r)$, $F_j^{L\vartheta}(\vartheta,\varphi,r)$ and $F_j^{L\varphi}(\vartheta,\varphi,r)$ are defined as follows:

$$F_j^{Lr} = a \sum_{l,m}^{L} f_l (l+1) \left[\left(\frac{a}{r_j}\right)^{l+2} Y_l^m(\vartheta_j,\varphi_j) \right] \left(\frac{a}{r}\right)^{l+1} Y_l^m(\vartheta,\varphi), \tag{4.2}$$

$$F_j^{L\vartheta} = a \sum_{l,m}^{L} f_l \left[\left(\frac{a}{r_j}\right)^{l+2} \partial_\vartheta Y_l^m(\vartheta_j,\varphi_j) \right] \left(\frac{a}{r}\right)^{l+1} Y_l^m(\vartheta,\varphi), \tag{4.3}$$

$$F_j^{L\varphi} = a \sum_{l,m}^{L} f_l \left[\left(\frac{a}{r_j}\right)^{l+2} \frac{\partial\varphi}{sin\vartheta_j} Y_l^m(\vartheta_j,\varphi_j) \right] \left(\frac{a}{r}\right)^{l+1} Y_l^m(\vartheta,\varphi), \tag{4.4}$$

where $\sum_{l,m}^{L}$ denotes the double sum $\sum_{l=1}^{L}\sum_{m=-l}^{l}$, ϑ, φ and r are the geocentric colatitude, longitude and radius, a is the Earth's reference radius (6371.2 km), $Y_l^m(\vartheta, \varphi)$ are the usual Schmidt semi-normalised spherical harmonic functions of degree l and order m, L is the maximum degree of expansion of internal sources and $f_l = \frac{2l+1}{4\Pi(l+1)^4 l^2}$. The parameter f_l imposes smooth characteristics on the derived field model.

With this magnetic field representation, and given that the functions are defined at the observation locations, it is possible to fit three-component data as a potential field, as long as the data contain spherical harmonic degrees lower than or equal to L. Equation (1) is time independent, it defines a linear system of equations where α_j^r, α_j^ϑ and α_j^φ are unknowns, the data are magnetic field measurements and the matrix consists of combinations of the functions F_j^{Lr}, $F_j^{L\vartheta}$ and $F_j^{L\varphi}$. This linear system is square and can be solved directly. For time representation, we expand each of the coefficients α_j^r, α_j^ϑ and α_j^φ on a basis of B-splines, i.e piecewise polynomials between spline knots:

$$\alpha_j^r = \sum_{k=1}^{n_{knots}} \beta_{k_j}^r b_k(t) \tag{4.5}$$

and α_j^ϑ and α_j^φ are expanded similarly.

The maximum degree of expansion of internal sources L was set to 12 for the core field to minimise the contributions of the lithospheric field. This degree is small enough to reveal mostly the field contributions from the core with the minimum wavelength of 4000 km. A trade-off between the misfit (Table 4.1) and temporal resolution was considered. For the time representation, order six B-splines with 21 spline knots spaced at 1 year intervals between 2001.0 and 2011.0 are used. A linear system is built from equations (1) and (5) and we solve for the unknown model parameters $\beta_{k_j}^r$, $\beta_{k_j}^\vartheta$ and $\beta_{k_j}^\varphi$ as defined in eq. (5).

Table 4.1: The RMS values of the misfit errors between SARM and satellite data for different L values when order six B-splines are used as temporal basis functions.

L	X (nT)	Y (nT)	Z (nT)
11	12.64	6.91	6.53
12	12.63	6.90	6.54
13	12.60	6.89	6.50
14	12.60	6.88	6.50

4.4 Data selection and processing

4.4.1 Satellite data

The data selection was performed on CHAMP satellite data that was recorded between roughly 300 km and 480 km of altitude. The quiet night data recorded between 2001 and 2010 were selected over the southern Africa region covering the area between 10°S and 38°S in latitude and 10°E and 38°E in longitude. Only quiet time data corresponding to a Dst index between -20 and 0 nT measured during local night times between 19:00 - 06:00 were considered. The choice of the time interval for data selection was done not to use satellite data that were contaminated by the solar quiet day variation. This regular variation with a fundamental period of 24 hours is particularly large at mid to low latitudes and is dependent on local time, latitude, season and solar cycle. It is caused mainly by electrical currents in the upper atmosphere (e.g. Macmillan and Droujinina, 2007). The external field has various sources with the ring current dominating at mid-latitudes. The Dst index, which is a proxy of the strength of the ring current, was therefore exclusively used in data selection. Figure 1 shows the plot of an example of data variations for a quiet day and a disturbed day chosen randomly between 2001 and 2010 with the corresponding estimated Dst indices varying between -20 and -5 nT and between -184 and 29 nT, respectively. Positive variations in Dst index are mostly caused by the compression of the magnetosphere from solar wind pressure increases where the magnetopause current dominates the ring current, indicating magnetic activities. A trade off was made between getting quiet time data and a good data coverage before setting the data selection criterion of Dst values to be between -20 and 0 nT.

The model input data were obtained by creating 225 data bins of 0.2°×0.2° on a

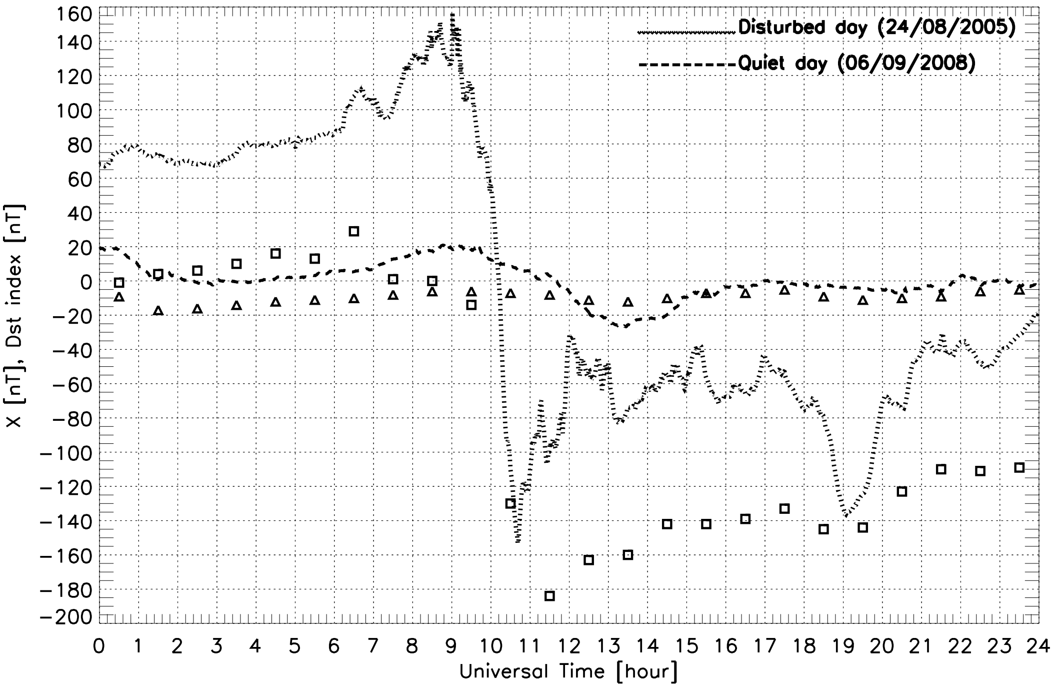

Figure 4.1: The illustration of data selection using Dst indices by plotting data variations of disturbed and quiet days (chosen randomly between 2001 and 2010) with corresponding Dst indices represented by square and triangle shapes for disturbed and quiet days, respectively. These 2 days were recorded at Hermanus magnetic observatory on the 24^{th} August 2005 and 6^{th} September 2008.

grid of 2° of latitude and 2° of longitude and additional 49 data bins on a grid of 4° of latitude and 4° of longitude. The middle point of the bin was considered to be a data center, 274 data centers were created (Fig. 4.2). The data values recorded in the same bin at the same epoch (within 4 days, e.g. 2005.34) were averaged to get a single data value.The size of a data bin was chosen to be small to avoid a big scatter between the data values recorded at the same epoch (average scatter values between data values recorded at the same epoch in the same data bin are 15.7 nT, 6.1 nT and 6.9 nT for X, Y and Z, respectively). The same epoch time interval (within 4 days) was chosen to avoid averaging data values recorded at different altitudes that are very far apart, the CHAMP satellite altitude was drifting during its time span (average scatter of altitudes of measured data in the same data bin at the same epoch is 0.05 km). Besides this, in the southern Africa region the secular variation for some components can reach 60 nT per year (5 nT per month on average) and the same epoch considered for several days can affect the averaged data quality. This averaging process made possible to obtain data at the same geographic location at different epochs (the data center is considered as a geomagnetic repeat station only with the difference that the data might be measured at different altitudes between 330 and 480 km). In this process, it was possible to accumulate a total of 3692 input data points corresponding to 11076 different input data values of X, Y and Z from 274 data centers for the period of 2001 - 2010 (Fig. 4.2). Figure 3 shows that all data bins in each year do not have data, either there are no satellite tracks passing over the data bins or the measured data do not meet the data selection criteria.

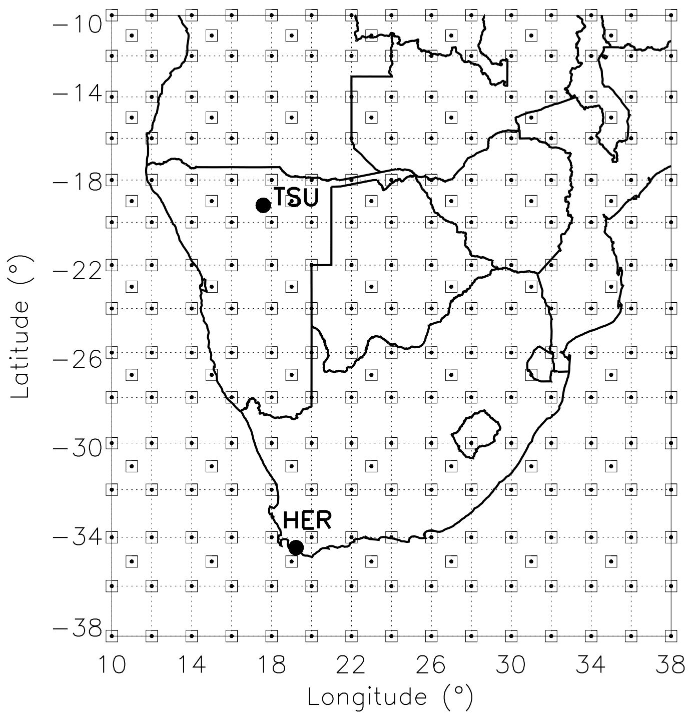

Figure 4.2: The data bins (0.2°×0.2°) used for generating input data (represented by square shapes), the middle of each data bin represents a data center. The 2 black big dots represent two permanent observatories Hermanus (HER) in South Africa and Tsumeb (TSU) in Namibia.

4.4.2 Ground data

The hourly mean data for two permanent magnetic observatories Hermanus and Tsumeb (also available at http://www.intermagnet.org) were selected using the same criteria of satellite data selection. To get monthly values, the average of hourly values in one month was performed. For Tsumeb data were missing for the whole

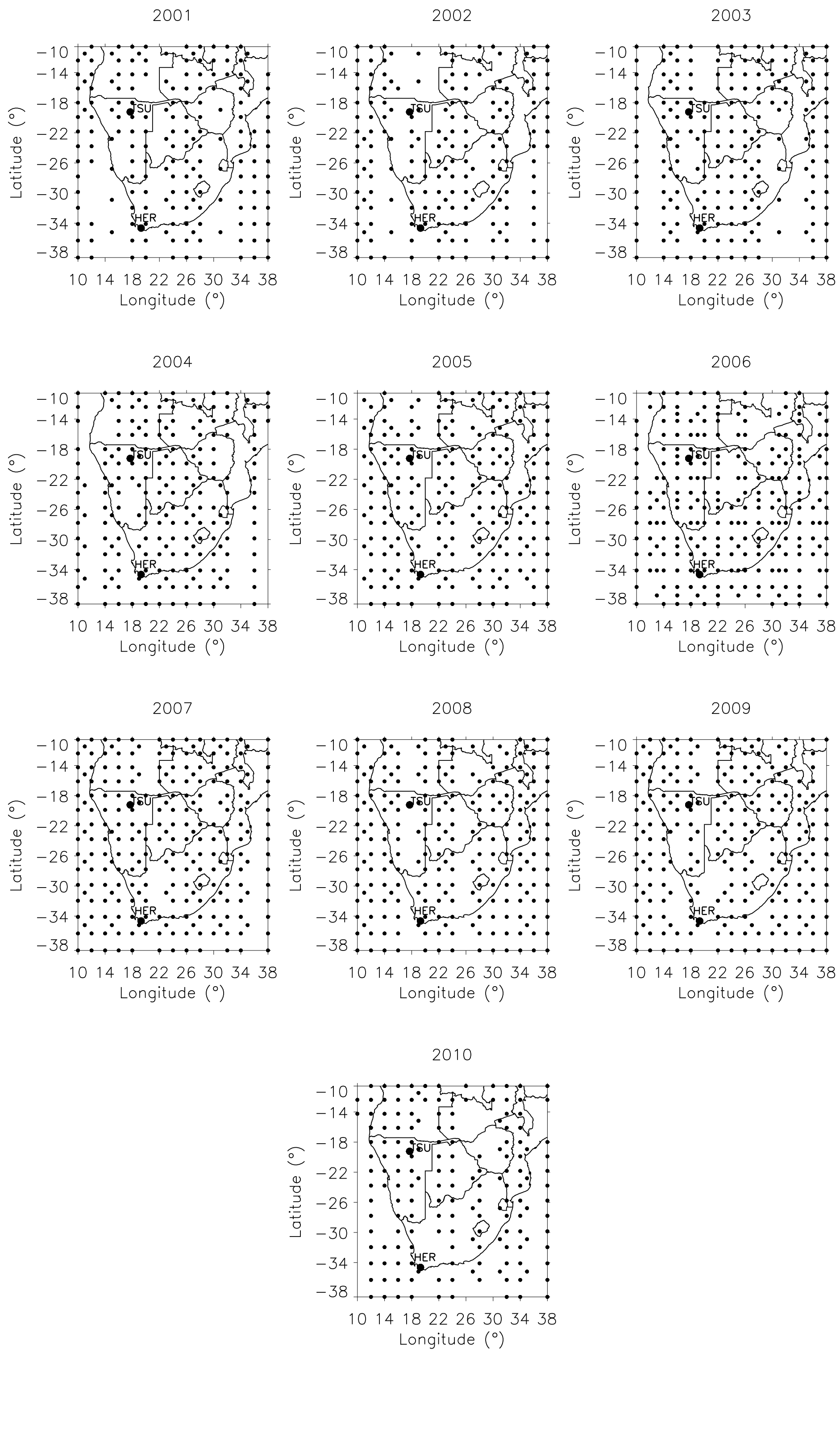

Figure 4.3: The data coverage per year between 2001 and 2010. It should be noted that one data center can have more than one data points at different epochs in the same year.

year of 2009 and it was decided to use only data recorded between 2001 and 2008. Furthermore, spline interpolation was performed to fill in the gaps of missing data for January and February in 2004, March and April in 2005, January, February and March in 2006 and March and April in 2008.

4.4.3 GRIMM-2 model

The GRIMM-2 model is derived from CHAMP satellite magnetic vector data and observatory hourly mean vector data, spanning the period 2001.0 to 2009.5. The data selection criteria for both SARM model and GRIMM-2 is given in Table 4.2. For GRIMM-2 the field in the Earth's core is modelled with the Earth's core reference radius of 3485 km while the maximum spherical harmonic degree is set to 16. The time dependency was represented by order six B-splines with knots one year apart spanning 2000-2011 (Lesur et al., 2010).

Table 4.2: The comparison between data selection criteria for SARM and GRIMM-2 models.

	SARM	GRIMM-2
Data source	CHAMP satellite data only	CHAMP satellite data and observatory hourly mean vector data
Local time	between 19:00 and 06:00	between 23:00 and 05:00
Indices	Dst index between -20 and 0 nT	norm of the Vector Magnetic Disturbances (VMD) less than 20 nT and norm of its time derivative less than 100 nT/day
Data sampling	274 data centers	20 sec minimum between sampling points

4.5 Data modelling, data analysis and results

The SARM model is based harmonic spline fitting of vector satellite data recorded between 2001 and 2010. The root mean square (rms) values of the difference between observed and model values were 12.6 nT, 6.9 nT and 6.5 nT for X, Y and Z components, respectively.

Secular variation values for each of X, Y and Z components in the analysis of derived data values from the 2 models (SARM and GRIMM-2) and ground data at 2 permanent observatories (HER and TSU) were calculated from monthly values as follows: $dB/dt = \dot{B(t)} = [B(t+6)-B(t-6)]/1year, (B = X, Y or Z)$, where the unit of t is month. In order to remove the annual variation, caused by magnetospheric and ionospheric currents and their Earth-induced counterparts, a 12-month running mean was applied to $\dot{B}(t)$ (B = X,Y or Z) (Olsen and Mandea, 2007) for SARM and ground data. It was noticed that the SARM model performance is of high standard inside the time interval [2001.5 : 2010.5], and the data analysis was done based on this time interval. Table 4.3 presents the difference between SARM and GRIMM-2 secular variation models between 2002 and 2009. The big differences in 2002 may be partially attributed to a bad data coverage in this year.

Table 4.3: RMS values of the differences between SARM and GRIMM-2 secular variation models between 2002 and 2009 at 400 km altitude. These differences were computed using 225 data points over the modelled area.

Component	2002.5	2003.5	2004.5	2005.5	2006.5	2007.5	2008.5	2009.5
dX/dt (nT/yr)	13.0	7.7	8.1	4.5	6.5	2.1	3.8	2.5
dY/dt (nT/yr)	12.6	6.1	6.3	5.6	3.4	2.5	1.9	3.6
dZ/dt (nT/yr)	6.3	6.1	6.9	3.9	6.1	3.2	4.1	4.3

The presented results of the main field and secular variation of X, Y and Z components in Fig. 4.5 and Fig. 4.6 respectively were obtained by predicting the geomagnetic field at 225 points at 400 km altitude. Table 4.4 shows the rms values of the differences between main field and secular variation values predicted by 2 models (SARM and GRIMM-2) at 225 points at 400 km altitude for the selected epoch 2006.5. The comparative evaluation between SARM and GRIMM-2 global models was performed, by comparing the two models over the entire region for a selected epoch at 2006.5 at two positions above HER (-34.3°, 19.2°, 400 km) and TSU (-19.2° ,17.6° , 400 km) observatories. In this case the focus was on secular variation to establish whether SARM is able to identify some geomagnetic jerks or rapid secular variation fluctuations in the region (Fig. 4.7). The comparison of SARM and GRIMM secular variation models with the help of measured satellite data was conducted. The choice of points over southern Africa was performed looking at the data centers having data in January and December during the same year, 17 points were selected (Fig. 4.4). The satellite data was first reduced to the same altitude (400 km) using the IGRF 11 model (Finlay et al., 2010). For a data point value D at a given geodetic coordinate (ϑ, φ, S), the data point value at a new altitude N is given as follows:

$$D_{(\vartheta,\varphi,N)} = D_{(\vartheta,\varphi,S)} + (IGRF11_{(\vartheta,\varphi,N)} - IGRF11_{(\vartheta,\varphi,S)}) \tag{4.6}$$

where S is the satellite altitude. The secular variation values at these points were calculated by getting the difference between data recorded in January and December (Table 4.5). Table 4.6 shows the RMS values of the differences between the SARM and GRIMM-2 secular variation models and secular variation values calculated from CHAMP satellite data at 17 points and at 400 km altitude.

Table 4.4: The RMS values of the differences between the SARM and GRIMM-2 models for epoch 2006.5 at 400 km altitude.

Field component	SARM - GRIMM-2 Main field (nT)	SARM - GRIMM-2 Secular variation (nT/yr)
X	25.7	6.5
Y	9.2	3.4
Z	8.5	6.1

4.6 Discussion and conclusions

The SARM model shows a general agreement in mapping the temporal and spatial characteristics of magnetic features with GRIMM-2 model, which was derived by fitting a vector data consisting of CHAMP satellite and observatory data. Table 4.4 presents the rms values of differences between these two models for epoch 2006.5 at 400 km altitude. For the main field models, the rms values are 25.7 nT, 9.2 nT and 8.5 nT for the X, Y and Z components, respectively. The rms values for the secular variation models are 6.5 nT, 3.4 nT and 6.1 nT for the X, Y and Z components, respectively. The regional model presents some edge effects that are noticeable in the maps of the differences of main and secular variation models of

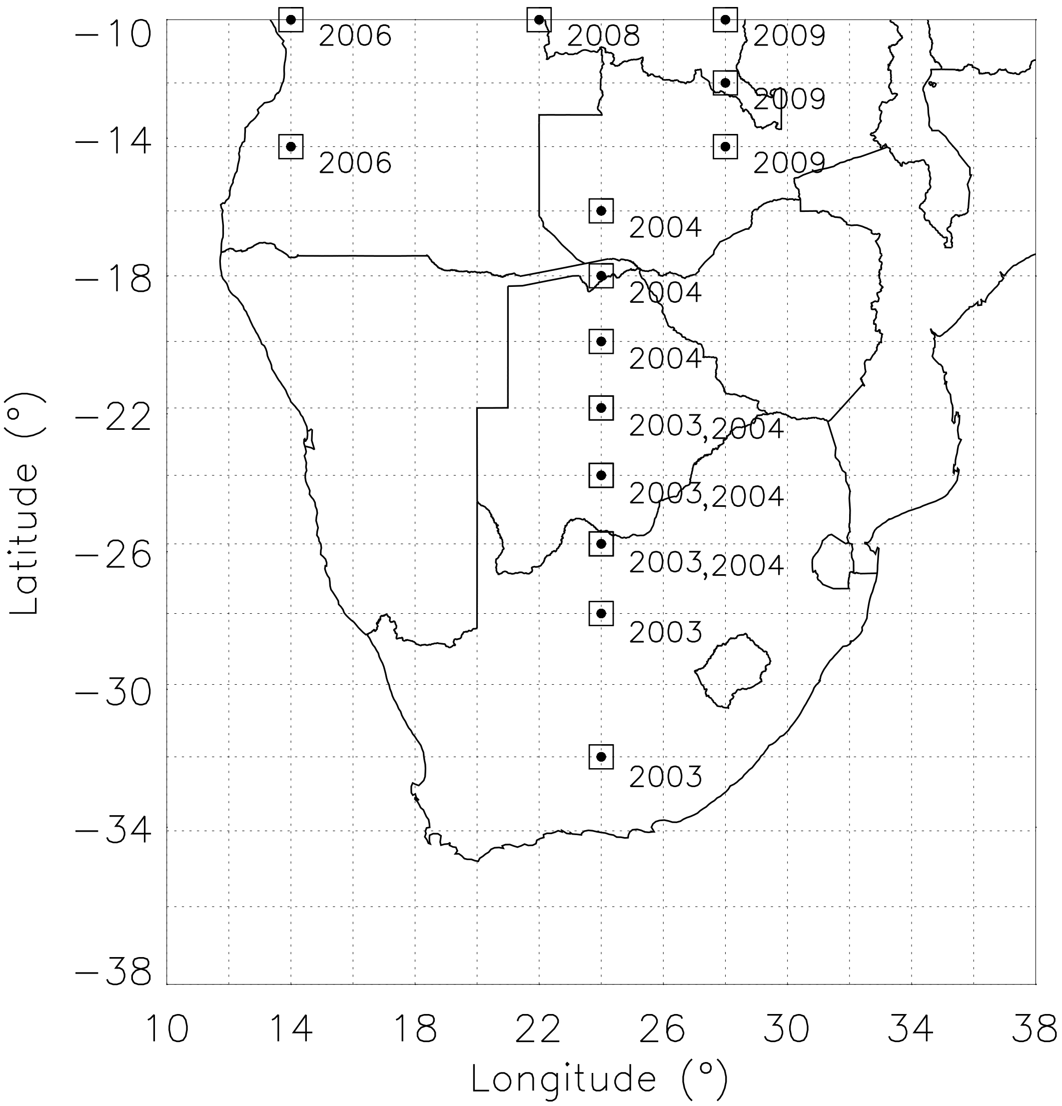

Figure 4.4: The data points used for the comparative evaluation between SARM and GRIMM-2 models. The choice of the points was based on the availability of satellite data in January and December during the same year at data centers, and the secular variation at these points were calculated by getting the difference between the data recorded in these two months. This was possible after reducing data to the same altitude (400km) with the help of IGRF model.

GRIMM-2 and SARM (Fig. 4.5 and Fig. 4.6). This is not surprising as regional models are characterised by this at and around the extreme edges of the area under investigation. The comparative evaluation of SARM and GRIMM-2 models with the help of satellite data at 17 selected points is shown in Tables 4.5 and 4.6. The rms values of the differences between secular variation values derived from SARM and computed from satellite data at 400 km are 13.3 nT/yr, 8.2 nT/yr and 7.7 nT/yr for X, Y and Z components, respectively. On the other hand, the rms values of the

Table 4.5: The values of SARM and GRIMM-2 secular variation models and secular variation values calculated from measured CHAMP satellite data at 18 selected points at 400 km altitude. $\dot{X}$,$\dot{Y}$ and $\dot{Z}$ are in nT/yr.

			Satellite data			SARM			GRIMM-2		
Colat.	Long.	Epoch	$\dot{X}$	$\dot{Y}$	$\dot{Z}$	$\dot{X}$	$\dot{Y}$	$\dot{Z}$	$\dot{X}$	$\dot{Y}$	$\dot{Z}$
112.100	24.100	2003.5	-12.3	-14.7	33.3	-7.3	-1.8	24.1	-3.6	-1.8	23.9
114.100	24.100	2003.5	0.5	-16.5	24.5	-8.0	-6.3	24.8	-4.2	-5.8	24.5
116.100	24.100	2003.5	9.4	-15.4	25.5	-8.4	-10.9	24.9	-4.7	-9.6	24.8
118.100	24.100	2003.5	8.24	-18.4	31.9	-8.5	-15.4	24.5	-5.4	-13.1	25.0
122.100	24.100	2003.5	0.25	-27.6	17.8	-8.6	-23.0	21.5	-7.1	-19.4	25.2
106.100	24.100	2004.5	-7.3	18.0	21.9	2.3	15.4	20.6	-2.5	10.4	17.5
108.100	24.100	2004.5	-7.0	13.0	10.2	2.8	10.6	21.2	-3.3	6.2	18.4
110.100	24.100	2004.5	22.1	15.2	19.6	2.3	5.9	21.6	-4.1	2.1	19.1
112.100	24.100	2004.5	15.3	8.4	18.2	1.2	1.1	22.3	-4.8	-2.0	19.8
114.100	24.100	2004.5	-3.4	-2.7	17.0	-0.2	-3.6	23.5	-5.6	-6.1	20.3
116.100	24.100	2004.5	-16.7	-13.4	18.6	-1.6	-8.1	25.2	-6.4	-9.9	20.7
100.100	14.100	2006.5	-31.9	42.0	-41.4	-9.6	45.9	-26.1	-15.1	46.8	-27.7
104.100	14.100	2006.5	-37.8	29.0	-21.4	-16.9	40.4	-19.7	-21.6	42.8	20.7
101.100	22.100	2008.5	-0.71	51.9	27.6	-1.0	39.9	11.1	-3.4	43.0	7.3
101.100	28.100	2009.5	5.2	35.9	24.3	7.4	23.8	30.6	8.7	27.1	28.3
102.100	28.100	2009.5	2.0	31.7	22.9	7.7	20.9	31.3	9.2	18.8	26.2
104.100	28.100	2009.5	-7.4	26.9	33.7	8.3	17.8	31.2	9.2	18.8	26.2

Table 4.6: The RMS values of the differences between the SARM and GRIMM-2 secular variation models and secular variation values calculated from CHAMP satellite data at 17 points and at 400 km altitude (Table 4.5).

Field component	SARM - Satellite data	GRIMM-2 - Satellite data
dX/dt (nT/yr)	13.3	12.5
dY/dt (nT/yr)	8.2	8.8
dZ/dt (nT/yr)	7.7	12.7

differences between secular variation values derived from GRIMM-2 and computed from satellite data at 400 km are 12.5 nT/yr, 8.8 nT/yr and 12.7 nT/yr for X, Y and Z components, respectively. Two models show a similar performance in X and Y components and SARM performs better in the Z component.

In an attempt to identify geomagnetic jerks and rapid secular variation fluctuations at HER and TSU observatories between 2001 and 2010, the SARM model shows a general agreement with ground data at HER and TSU in all components (Fig. 4.7). The occurrence of 2007 geomagnetic jerk is seen in the X, Y and Z

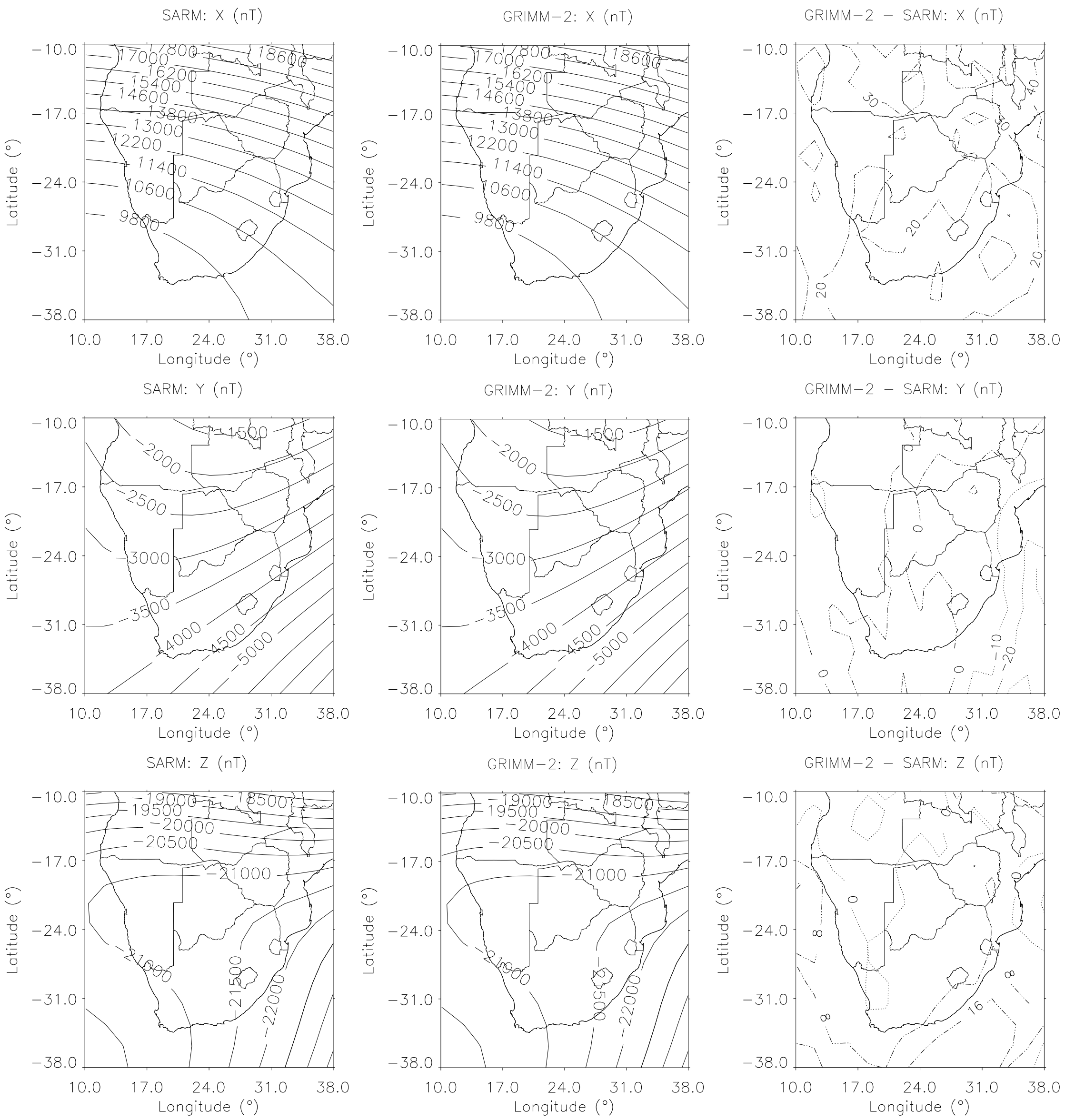

Figure 4.5: Comparison of the main field of X, Y and Z components derived from two models (SARM and GRIMM-2) for 2006.5 at 400 km of altitude. Left column is SARM, GRIMM-2 in the middle and right column is the difference between 2 models. Top row is the X component, Y in the middle and bottom row is the Z component.

components at HER and TSU. During data processing it was noticed that there is a big scatter (especially in the X component) in averaged measurements in the same bin between −10° and −20° latitude. This would require to reduce the size of data bins in the area to improve the model. The occurrence of the jerk is shown by both models and ground data at slightly different times around 2007.0. The

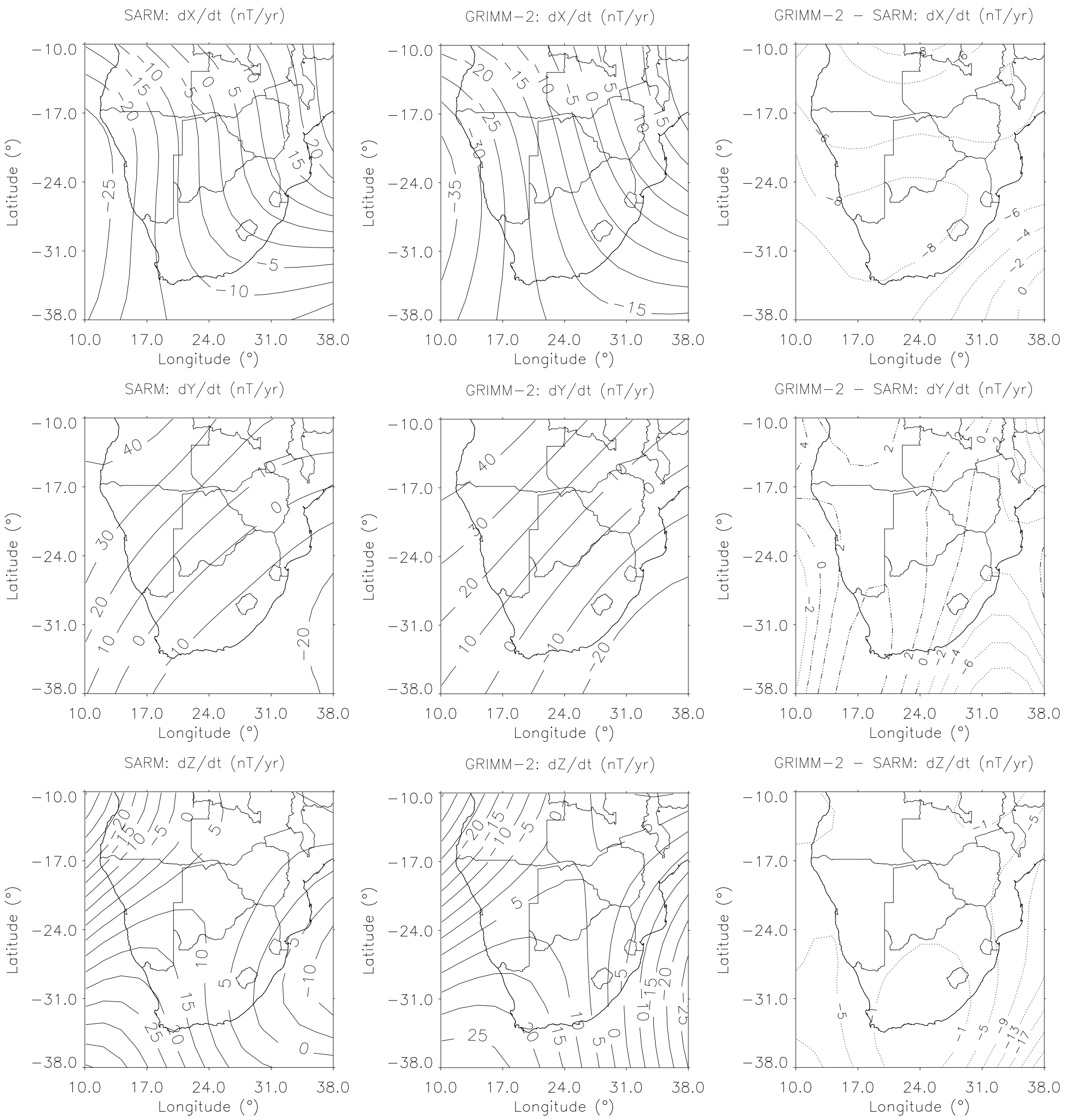

Figure 4.6: Comparison of the secular variation of X, Y and Z components derived from two models (SARM and GRIMM-2) for 2006.5 at 400 km of altitude. Left column is SARM, GRIMM-2 in the middle and right column is the difference between 2 models. Top row is the X component, Y in the middle and bottom row is the Z component.

2003 and 2004 rapid secular variation fluctuations are clearly shown in the X, and Z components at HER and TSU by the SARM model and ground data. However, the GRIMM-2 model does not show these small scale features of the time variation of geomagnetic field in the region.

Thanks to the dedicated and intensified magnetic field measurements by the

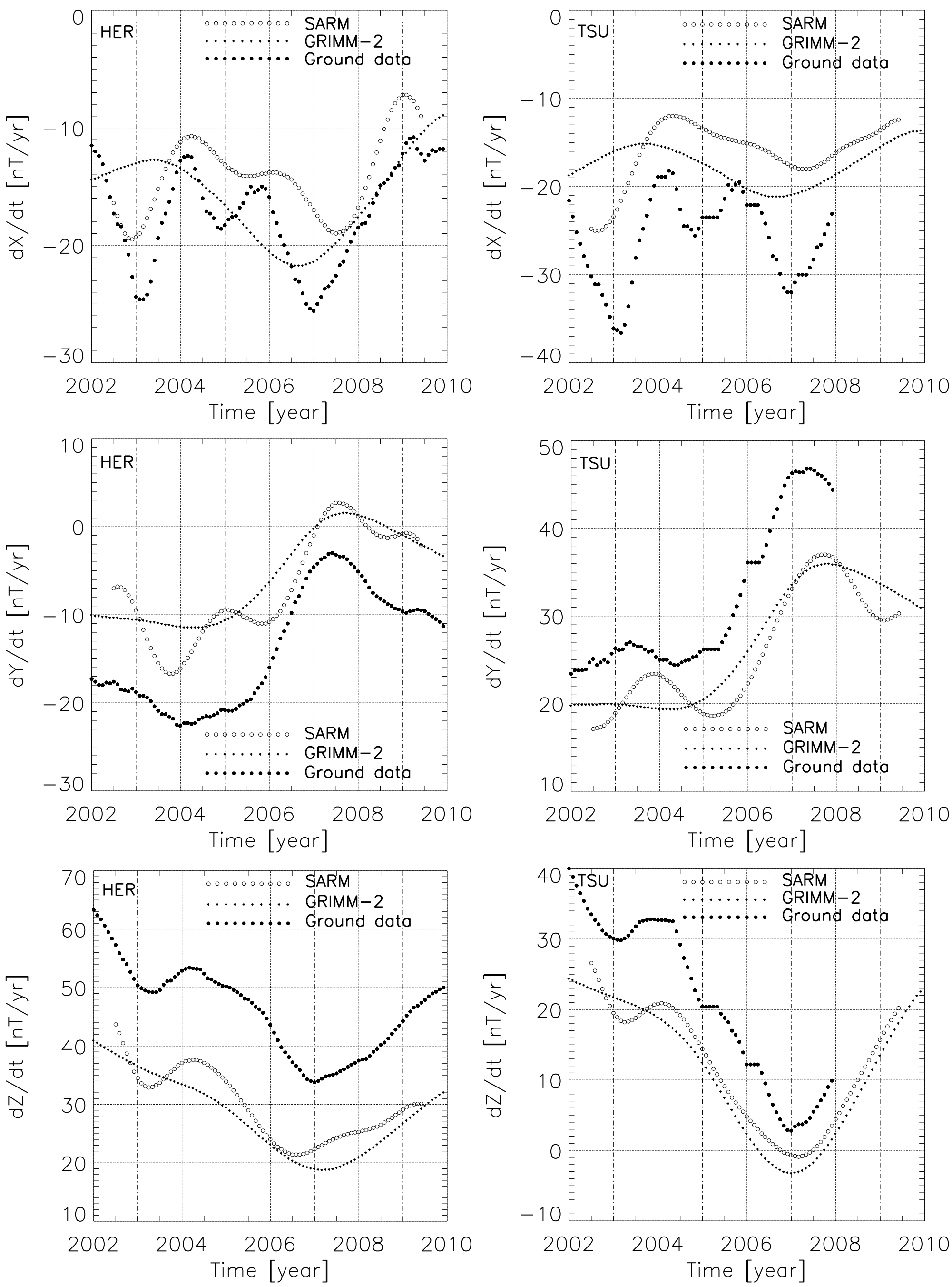

Figure 4.7: Secular variation for X, Y and Z derived from two models (SARM and GRIMM-2) at 400 km above HER and TSU observatories and ground data at these observatories between 2002.0 and 2010.0. The ground data is used to perform a comparative evaluation of two models about the occurrence of the 2007 geomagnetic jerk and 2003 and 2004 rapid secular variation fluctuations in southern Africa. The left and right columns represent the temporal variations of geomagnetic field at HER and TSU, respectively.

CHAMP satellite, we are now able to map the field with high accuracy. The two modelling approaches presented in our study both benefit from this dense data set leading to accurate models for the main field as well as secular variation investigations. Both approaches feature its own particular advantages and limitations in respect to their usability, physical meaning, and inter- and extrapolation characteristics. Nonetheless, both methods revealed similar results within the area of observation, revealing highly non-linear time-varying characteristics of the geomagnetic field in southern Africa. Since the first magnetic field measurements of the Earth's field started in the nineteenth century, a steady decrease of the Earth's magnetic dipole has been observed. The change in the core field strength is not evenly distributed across the globe, with the most rapid decrease observed across the South Atlantic region. Southern Africa therefore provides an ideal opportunity to study these geomagnetic field changes as witnessed by our present investigation using CHAMP satellite data.

In conclusion, the SARM model derived from only CHAMP satellite data was compared with the global model GRIMM-2 and ground data. The regional model was able to identify the 2007 geomagnetic jerk as well as the 2003 and 2004 rapid secular variation fluctuations at two magnetic observatories (HER and TSU). However, there is a need to improve the model by removing more external field noise from CHAMP satellite data and increasing the number of data centers to have a good data coverage with the help of the CM4 model. The differences between the regional model (SARM) and the global model (GRIMM-2) may be partially attributed to the different data selection criteria and the different datasets used in developing the models (Finlay et al., 2010). In building GRIMM-2 model, both ground and satellite data were used whereas only satellite data was used for SARM.

4.7 Acknowledgements

The support of the CHAMP mission by the German Aerospace Center (DLR) and the Federal Ministry of Education and Research are gratefully acknowledged. We thank Vincent Lesur and Anne Geese for the harmonic spline technique source code and two anonymous reviewers for their constructive comments.

Chapter 5

The use of both satellite and ground-based data to develop a regional magnetic field model over southern Africa

The contents of this chapter are the preprint of an extended abstract that has been published in the proceedings of the 14th SAGA Biennial Technical Meeting and Exhibition (2015, pages 342-345). I conceived this study in discussion with my supervisors. Work for drafting this manuscript was completed by me, with the remaining authors contributing to improve the manuscript.

An investigation in the use of satellite and ground-based data to develop a harmonic spline core magnetic field model over southern Africa

E. Nahayo[1,2], P. B. Kotzé[1] and S. J. Webb[2]

[1]South African National Space Agency (SANSA) Space Science, South Africa, enahayo@sansa.org.za

[2] University of the Witwatersrand, South Africa, susan.webb@wits.ac.za

Biography

Mr Emmanuel Nahayo is a Scientific Data Analyst at South African National Space Agency (SANSA) Space Science in Hermanus, South Africa. He is doing a PhD part time in Geophysics in the School of Geosciences at University of the Witwatersrand in South Africa.

Dr Pieter B. Kotzé is a Senior Research Fellow at SANSA Space Science.

Dr Susan J. Webb is a Senior Lecturer at University of the Witwatersrand in the School of Geosciences.

5.1 Summary

The study of the time variations of the Earth's magnetic field requires continuous recording of geomagnetic data with good temporal and spatial coverage over the area of study. In southern Africa, ground recording stations are limited in number and the use of both satellite and ground data are needed for studies where high temporal and spatial resolution data are required. An investigation into the combination of both satellite and ground-based data to develop a southern Africa core field magnetic model (SACFM-2) is conducted using a harmonic splines technique on data recorded by the CHAMP satellite between 2005 and 2010, ground-based data collected at 35 geomagnetic repeat stations in southern Africa and observatory data recorded at four INTERMAGNET observatories (Hermanus, Hartebeesthoek, Tsumeb and Keetmanshoop) during the same time period. The global GFZ Reference Internal Magnetic Model (GRIMM-3) is used in the comparative evaluation of the more localized southern Africa model. The results suggest that this approach of combining satellite and ground-based data to develop a core field model over southern Africa is possible. The occurrence of 2007 geomagnetic jerk is shown by both models (SACFM-2 and GRIMM-3) in the Y and Z components of the magnetic field. The SACFM-2 model does not show the geomagnetic jerk in the X component due to the noise.

textitKey words: magnetic data, regional field modelling, geomagnetic secular variation.

5.2 Introduction

The Earth's main magnetic field generated in the core is not constant but changes non-uniformly with time. This temporal change is known as secular variation. Sudden changes in the linear secular variation trend are known as geomagnetic jerks or secular variation impulses (Mandea et al., 2010).

The considered area of study, southern Africa, is located in a region where a rapid decrease of magnetic field intensity is observed at the Earth's surface stretching across southern Africa and the South Atlantic Ocean (Lesur et al., 2008). This coincides with a region known as the South Atlantic Magnetic Anomaly where the field is already very weak, approximately 30% diminished compared to other locations at similar latitudes. The study of the long term variation of the geomagnetic field over southern Africa can contribute to monitoring the evolution of the South Atlantic Magnetic Anomaly and in studying processes in the Earth's core. De Santis et al. (2012) suggest that global warming may even be partly controlled by deep Earth processes triggering geomagnetic phenomena on a century time scale such as the South Atlantic Magnetic Anomaly. The internal Earth processes generating the Earth's main magnetic field can be studied by modelling the core field using the technique of inversion on measured magnetic field data and can assist to monitor the long-term variation of the geomagnetic field.

An investigation into methods of combining satellite and ground-based data is conducted over the period between 2005 and 2010 in southern Africa. The choice of this interval is motivated by the availability of CHAMP satellite data and ground-based data collected at 35 southern Africa geomagnetic repeat stations and four magnetic observatories, Hermanus (HER), Hartebeesthoek (HBK), Tsumeb (TSU) and Keetmanshoop (KMH). The southern Africa core field model is developed using harmonic splines and is evaluated with the help of the global model GRIMM-3 (Lesur et al., 2011) together with independent ground data at 3 geomagnetic repeat stations (Windhoek (WIN) in Namibia, Maun (MAU) in Botswana and Garies (GAR) in

South Africa).

The results show a good agreement with the global model GRIMM-3 where both models show the occurrence of 2007 geomagnetic jerk in the Y and Z component (Kotzé, 2011). However, there is still significant noise obscuring the X component after noise removal from the CHAMP satellite data. This is to be expected as the X component is the most influenced by external magnetic fields.

5.3 Methods and Results

5.3.1 Data source and selection

The data selection was performed on CHAMP satellite data that was recorded between roughly 300 km and 400 km of altitude. The quiet night data recorded between 2005 and 2010 were selected over the southern Africa region covering the area between 16°S and 36°S in latitude and 10°E and 34°E in longitude. Only quiet time data corresponding to a Dst index between -20 and 0 nT measured during local night times between 19:00 - 06:00 were considered (Figure 5.1).

The hourly mean data for four permanent magnetic observatories HER, TSU, HBK and KMH (also available at http://www.intermagnet.org) were selected using the same criteria of satellite data selection. The monthly values were computed by calculating the average of the obtained hourly values to get the input data for the model. The field survey data collected at 35 geomagnetic repeat stations were also used as input data (also available at http://www.geomag.bgs.ac.uk/data_service/data/surveydata.html). Three geomagnetic stations, WIN, GAR and MAU were not included to contribute input data (not to influence the performance of the model at these points) and were only used in the evaluation of the developed model using independent ground data.

The global magnetic field, comprehensive model (CM4), was used to remove the contributions of ionospheric and magnetospheric currents from all data (Sabaka et al., 2004). Atmospheric tidal winds due to differential solar heating or due to gravitational lunar force move the ionospheric plasma against the geomagnetic field lines, thus generating electric fields. These electric currents result in an additional

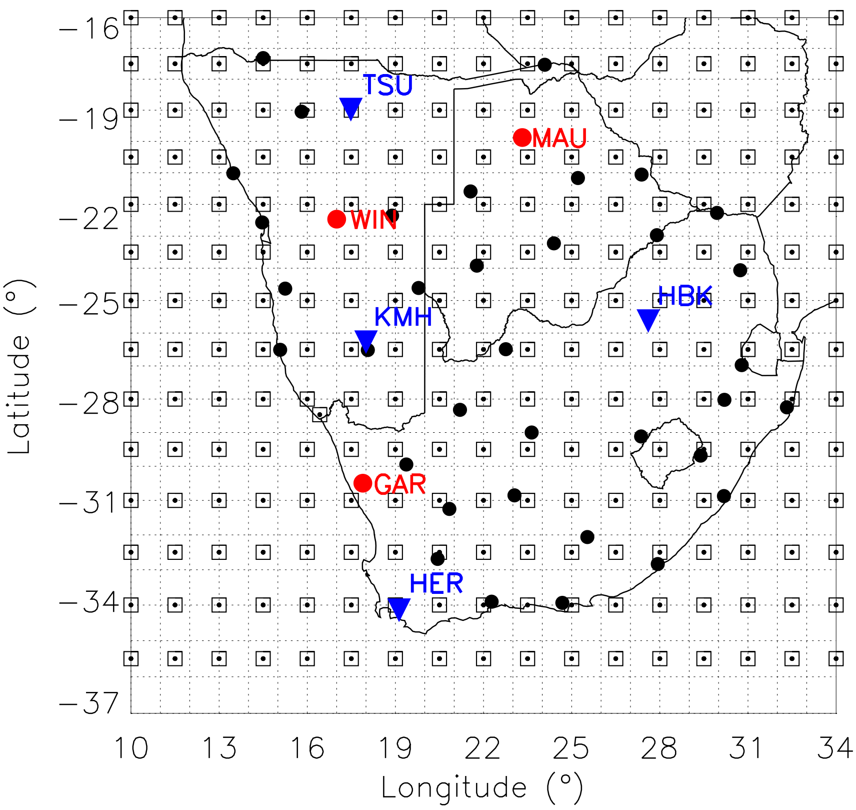

Figure 5.1: The map showing the data bins (0.2° x 0.2°, square shapes on the map) on a grid of 1.5° of latitude and 1.5° of longitude. The input data are the averages of recorded values at the same epoch in the same bin (small dots inside the bins are data centers). The big black dots are geomagnetic stations and three red big dots are the geomagnetic repeat stations (WIN, GAR and MAU) used in the developed model evaluation. The four blue triangles are the permanent observatories HER, HBK, TSU and KMH.

magnetic field observed during magnetospheric quiet conditions. Additional electric currents are generated by the varying magnetospheric convection electric field. They also contribute to the measured magnetic field.

5.3.2 Harmonic splines technique

The method of using harmonic splines for geomagnetic field modelling was first introduced by Shure et al. (1982) for global magnetic modelling, but it is also suitable for regional modelling. The harmonic spline functions satisfy Laplace's equation. Thus, they allow a potential field approach combining the individual components of the magnetic field in a physically meaningful way (Geese et al., 2011).

According to Lesur (2006) and Geese et al. (2009), for each data location defined by its geocentric coordinates(ϑ, φ, r), the Earth's magnetic field can be written as:

$$B(\vartheta,\varphi,r) = -\bigtriangledown \left\{ \sum_j \alpha_j^r F_j^{Lr}(\vartheta,\varphi,r) + \sum_j \alpha_j^\vartheta F_j^{L\vartheta}(\vartheta,\varphi,r) + \sum_j \alpha_j^\varphi F_j^{L\varphi}(\vartheta,\varphi,r) \right\}, \tag{5.1}$$

where the functions $F_j^{Lr}(\vartheta,\varphi,r)$, $F_j^{L\vartheta}(\vartheta,\varphi,r)$ and $F_j^{L\varphi}(\vartheta,\varphi,r)$ are defined as follows:

$$F_j^{Lr} = a\sum_{l,m}^{L} f_l(l+1)\left[\left(\frac{a}{r_j}\right)^{l+2} Y_l^m(\vartheta_j,\varphi_j)\right]\left(\frac{a}{r}\right)^{l+1} Y_l^m(\vartheta,\varphi), \tag{5.2}$$

$$F_j^{L\vartheta} = a\sum_{l,m}^{L} f_l\left[\left(\frac{a}{r_j}\right)^{l+2} \partial_\vartheta Y_l^m(\vartheta_j,\varphi_j)\right]\left(\frac{a}{r}\right)^{l+1} Y_l^m(\vartheta,\varphi), \tag{5.3}$$

$$F_j^{L\varphi} = a\sum_{l,m}^{L} f_l\left[\left(\frac{a}{r_j}\right)^{l+2} \frac{\partial\varphi}{sin\vartheta_j} Y_l^m(\vartheta_j,\varphi_j)\right]\left(\frac{a}{r}\right)^{l+1} Y_l^m(\vartheta,\varphi), \tag{5.4}$$

where $\sum_{l,m}^{L}$ denotes the double sum $\sum_{l=1}^{L}\sum_{m=-l}^{l}$, ϑ, φ and r are the geocentric colatitude, longitude and radius, a is the Earth's reference radius (6371.2 km), $Y_l^m(\vartheta,\varphi)$ are the usual Schmidt semi-normalised spherical harmonic functions of degree l and order m, L is the maximum degree of expansion of internal sources and $f_l = \frac{2l+1}{4\Pi(l+1)^4 l^2}$. The parameter f_l imposes smooth characteristics on the derived field model. Equation (1) is time independent, it defines a linear system of equations where α_j^r, α_j^ϑ and α_j^φ are unknowns. This linear system is square and can be solved directly. For time representation, we expand each of the coefficients α_j^r, α_j^ϑ and α_j^φ on a basis of B-splines. The maximum degree of expansion of internal sources was set to 12. This degree is small enough to reveal mostly the field contributions from the core. For the time representation, order six B-splines with spline knots spaced at 1 year intervals between 2005.0 and 2011.0 are used.

5.3.3 Modelling results

Using 564 data centers with 4570 data points (13710 different data values of X, Y and Z) recorded between 2005 and 2010, the root-mean square (rms) values of the differences between input data and SACFM-2 model data values are 11.2 nT, 12.4 nT and 4.7 nT for X, Y and Z respectively. The comparative evaluation of

SACFM-2 using the global model GRIMM-3 was performed focussing only on the secular variation models.

Figure 5.2 shows the comparison between two secular variation models (SACFM-2 and GRIMM-3) at the three selected reference points (MAU, WIN and GAR) for X, Y and Z field components. Table 5.1 shows the comparison of the secular variation models of SACFM-2 and GRIMM-3 at ground level using values computed at 564 data centers over the whole area of study. The comparative evaluation of SACFM-2 using GRIMM-3 and ground data is presented in Figure 5.3 and Table 5.2.

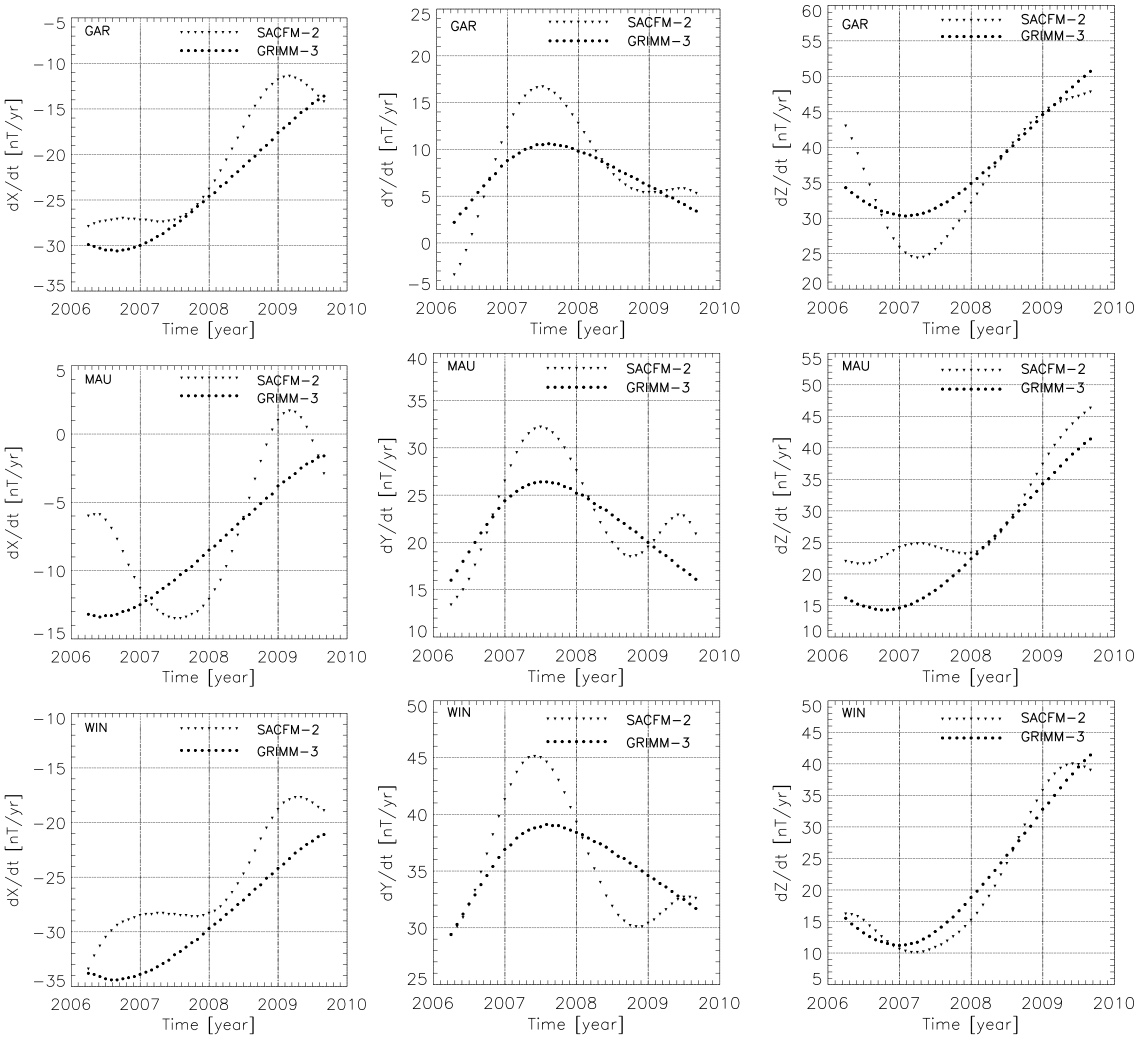

Figure 5.2: Comparison of secular variation models of SACFM-2 and GRIMM-3 at ground level at GAR, MAUN and WIN geomagnetic repeat stations. The first column is the X component, Y in the middle and Z on the left. The first raw is GAR station, MAU in the middle and WIN at the bottom. The occurrence of the 2007 geomagnetic jerk is visible in the Y and Z components and in both models.

Table 5.1: The RMS values of the differences between the SACFM-2 and GRIMM-3 secular variation models at ground level at the beginning of each year from 2006 to 2010 computed at 279 data centers. The RMS values unit is nT/yr.

Field component	2006.0	2007.0	2008.0	2009.0	2010.0
dX/dt (nT/yr)	8.4	6.7	6.8	7.1	10.6
dY/dt (nT/yr)	8.2	12.1	4.1	4.9	6.1
dZ/dt (nT/yr)	10.2	6.8	5.1	4.9	15.1

Table 5.2: The RMS values of the differences between the SACFM-2 and GRIMM-3 secular variation models and ground data at 3 reference points (GAR, MAU and WIN).

Field component	SACFM-2 minus Ground data	GRIMM-3 minus Ground data
dH/dt (nT/yr)	6.7	6.1
dD/dt (min/yr)	1.9	1.7
dZ/dt (nT/yr)	7.4	3.8

5.4 Conclusions

These preliminary results show that combining ground-based and satellite data to develop a continuous core field model over southern Africa using harmonic splines technique yields promising results. Table 1 compares two models over the whole area of study at the beginning of each year from 2006 to 2010. The agreement in performance of these two models is weak close to the beginning and the end of the time interval (2006.0 and 2010.0). These differences can be partially attributed to different data sets used by these models due to different criteria of data selection. Table 5.2 and Figure 5.3 show close performance of SACFM-2 and GRIMM-3 in all components. GRIMM-3 performs better in the Z component. Figure 5.2 shows a similar performance of two models in Y and Z components as two models identify the 2007 geomagnetic jerk. The noise in X component does not allow the model to show the occurrence of this geomagnetic jerk at WIN and GAR geomagnetic repeat stations (Figure 5.3). Further investigations will be conducted on how to improve this regional model, especially reducing noise by restricting the considered local time interval in data selection and increasing the number of data points for the model. Using an accurate regional core field model over southern Africa would make

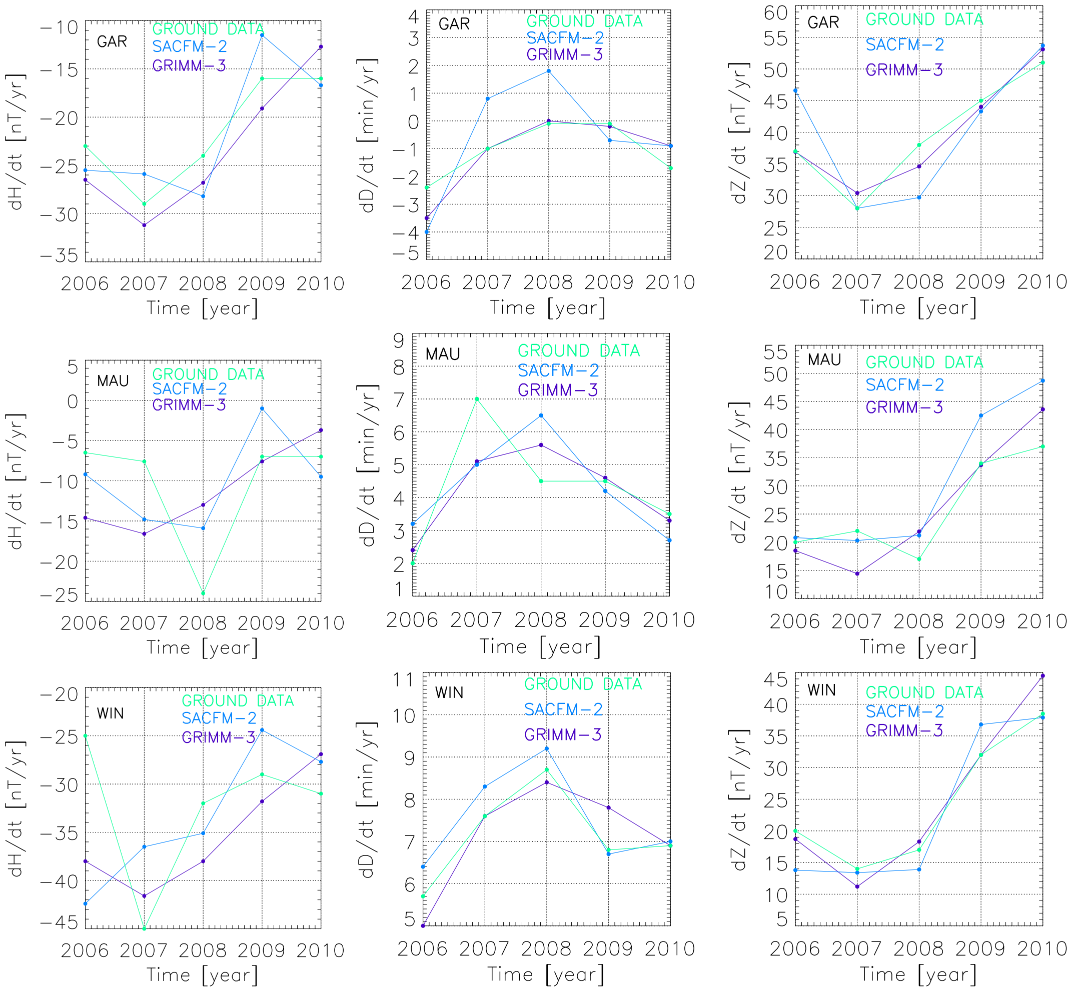

Figure 5.3: Comparative evaluation of SACFM-2 with independent ground data at GAR, MAU and WIN geomagnetic repeat stations. The first column is the H component, D in the middle and Z on the left. The first raw is GAR station, MAU in the middle and WIN at the bottom.

possible the monitoring of the evolution of the South Atlantic Magnetic Anomaly in this region contributing to the long term space weather forecasting in the region.

5.5 Acknowledgments

The support of the CHAMP mission by the German Aerospace Center (DLR) and the Federal Ministry of Education and Research and the work of GFZ German Research Centre for Geosciences to make data available are gratefully acknowledged. Thanks to Vincent Lesur and Anne Geese for the harmonic spline technique source code. This work was funded by the South African National Space Agency (SANSA)

Space Science.

Chapter 6

Evolution of South Atlantic Anomaly over southern Africa

The contents of this chapter are the preprint of a paper that has been published in the journal of Earth, Planets and Space (2018, https://doi.org/10.1186/s40623-018-0796-6). I conceived this study in discussion with my supervisors. Work for this manuscript and write-up was completed by me, with the remaining authors contributing to improve the manuscript.

A harmonic spline magnetic main field model for southern Africa combining ground and satellite data to describe the evolution of the South Atlantic Anomaly in this region between 2005 and 2010

Emmanuel Nahayo[1,3], Pieter B Kotzé[1], Monika Korte[2] and Susan J Webb[3]

[1]South African National Space Agency (SANSA) Space Science, PO Box 32, Hermanus, South Africa, email: enahayo@sansa.org.za and pkotze@sansa.org.za

[2]GFZ German Research Centre for Geosciences,Telegrafenberg, 14473 Potsdam, Germany, email: monika@gfz-potsdam.de

[3]University of the Witwatersrand, PO Box 2050, Johannesberg, South Africa, email: susan.webb@wits.ac.za

6.1 Abstract

CHAMP satellite and relatively densely spaced ground-based data measured over southern Africa between 2005 and 2010 are combined in a new regional, harmonic spline based, core field model. This new SACFM-2 model is compared to the regional SARM model, which is based only on satellite data, and the global CHAOS-6 model. The results agree well in the vertical (Z) component, with somewhat larger differences in the horizontal components. The Z component and total intensity F are used to investigate the evolution of the South Atlantic Anomaly in this region. The computed maps of main field of the Z component and total intensity F show a steady decrease of the field over the years during the study period, indicating the evolution of the South Atlantic Anomaly over southern Africa and suggesting an increase of the area of this feature.

Key words: Harmonic splines, Geomagnetic data, Secular variation, South Atlantic Anomaly

6.2 Introduction

The largest part of the magnetic field measured at the Earth's surface can be ascribed to a self-exciting geo-dynamo process that takes place inside the outer core. This outer core is fluid and consists mostly of iron and nickel that are good electrical conductors. The main field is generated by electrical currents flowing in the outer core (e.g. Hollerbach, 1996). It undergoes long term changes with decrease and increase of the field at irregular and unpredictable rates. This phenomenon of a long term time variation of geomagnetic field is also known as secular variation, while abrupt changes in the secular variation trend are referred to as geomagnetic jerks (Courtillot et al., 1978). There are additional sources of the Earth's magnetic field. The lithospheric field results from the rocks magnetisation in the crust and upper mantle. Electrical currents in the ionosphere and magnetosphere, resulting from interactions between solar wind and the near-Earth space environment, add a significant contribution of external sources.

Monitoring the field evolution by direct observations has revealed that the magnetic dipole of the Earth oriented along the Earth's rotational axis has decreased in the last 175 years and it has become now 9% weaker than it was in 1840 (e.g. Finlay et al., 2016a). The decrease of the field is attributed to the changes taking place in the molten iron outer core. In a recent study, (Finlay et al., 2016a) found that the majority of the dipole decay is caused by meridional flux advection since 1840 with magnetic diffusion making a small contribution.

The chosen area of study, southern Africa, lies in a region where magnetic field intensity has decreased rapidly and this region covers the area from southern Africa over the South Atlantic Ocean to South America. (e.g. Badhwar, 1997). Southern Africa borders a region known as the South Atlantic Anomaly (SAA) where the field has already weakened by approximately 30% compared to other regions located at similar latitudes. The study of the long term variation of the geomagnetic field over southern Africa can contribute to monitoring the evolution of the South Atlantic Anomaly (SAA) (Fig. 6.1). The SAA is essential in explaining how the dipole decay rate may be controlled by a planetary-scale gyre in the molten outer core (Finlay et al., 2016a). Some geophysicists see this anomaly as an indicator of forthcoming geomagnetic transition, such as a dipole excursion or reversal (Pavón-Carrasco and DeSantis, 2015). The weakening in field intensity can have a negative effect on the magnetic field shielding of the Earth against high energetic particles and cosmic rays that can penetrate into the magnetophere and atmosphere, creating space weather hazards for technology such as, e.g., communication satellites (e.g. Heirtzler et al., 2002).

An investigation into a combination of both satellite and ground-based data to develop a Southern Africa Core Field Magnetic Model (SACFM-2) is conducted using a harmonic spline technique based on data recorded by the CHAMP satellite between 2005 and 2010, ground-based data collected at 38 geomagnetic repeat stations in southern Africa and observatory data recorded at four INTERMAGNET

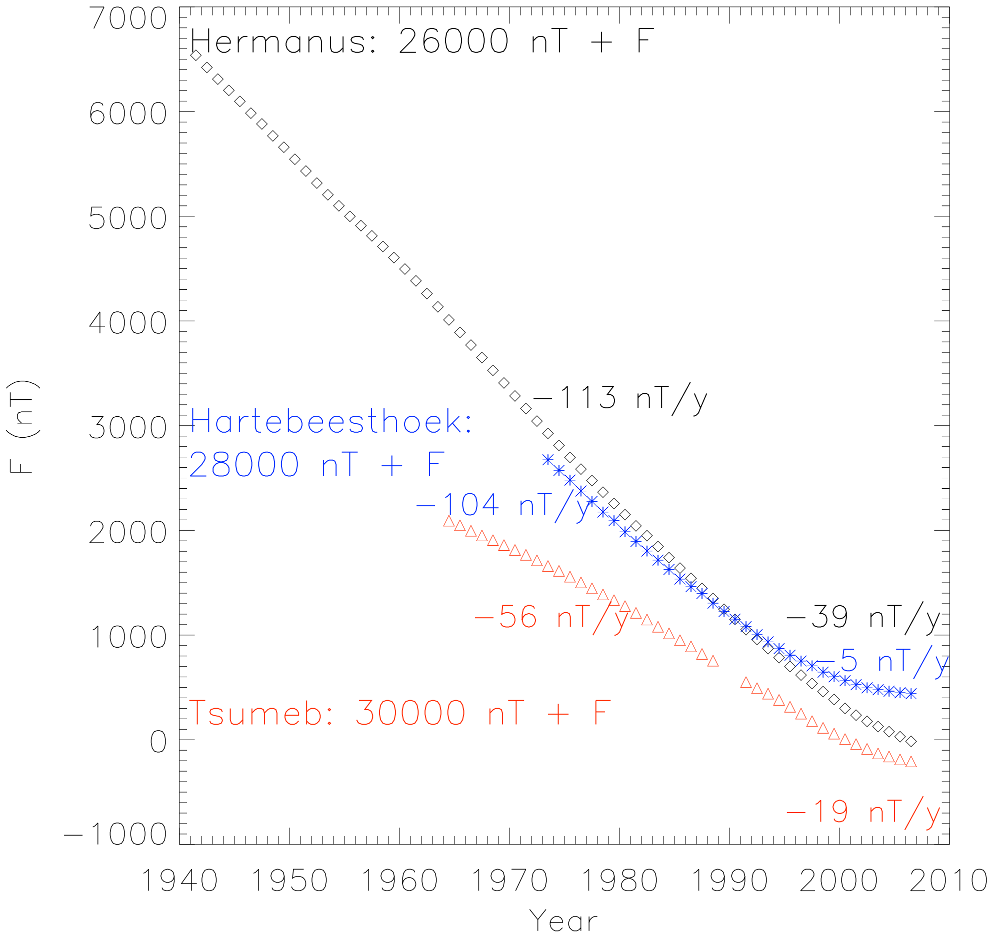

Figure 6.1: Evolution of total field intensity from 1940 to 2010 at Hermanus (HER), Hartebeesthook (HBK) and Tsumeb (TSU) magnetic observatories in southern Africa. There is a decrease of 20% of total field intensity at Hermanus between 1940 and 2010.

observatories (Hermanus, Hartebeesthoek, Tsumeb and Keetmanshoop) during the same time period. Regional field studies of the long term time variation of the geomagnetic field in southern Africa using the harmonic spline technique were conducted before using only either ground-based data (Geese et al., 2009) or satellite data (Nahayo et al., 2015). The method is summarized in section 6.3 and the utilized data are described in section 6.4. The new model and its fit to the different data types is presented and compared to previous models in section 6.5 with the help of the selected points over the area of study in Table 6.1, and further discussed in section 6.6. Finally, the evolution of the SAA in southern Africa between 2005 and 2010 is discussed based on the new model, before concluding with an outlook to future monitoring of the magnetic field and its evolution in the region (Section 6.7).

Table 6.1: Geocentric coordinates of the points that are used in the evaluation of the SACFM-2 model at ground level.

Station	Colatitude (°)	Longitude (°)	Geocentric altitude (km)
Hermanus (HER)	124.246	19.225	6371.226
Hartebeesthoek (HBK)	115.732	27.707	6372.755
Tsumeb (TSU)	109.083	17.584	6372.473
Keetmanshop (KMH)	116.377	18.100	6372.263
Maun (MAU)	109.978	23.420	6372.107
South West Point (SWP)	123.000	14.000	6372.000

6.3 Method

The aim of this paper is to use a regional main field model, derived from CHAMP satellite and ground-based magnetic data, to investigate the evolution of the SAA over southern Africa. This regional model is derived using harmonic splines, following the approach recently applied by Geese et al. (2009, 2011) to southern African ground data. The harmonic splines were first introduced by Shure (Shure et al., 1982) for global magnetic field modelling using Magsat data, but they are also appropriate for regional modelling. The harmonic spline functions meet the Laplace's equations requirements, they allow a potential field approach combining the individual magnetic field components in a physical significant way (Geese et al., 2011). The technique allows the successful integration of both satellite and ground-based observations at different altitudes above the Earth's surface. We summarize the method starting with the spatial aspect and introducing at the end a time dependency in the model.

According to Lesur (2006) and Geese et al. (2009), the Earth's magnetic field can be written as:

$$B(\vartheta,\varphi,r) = -\nabla\left\{\sum_j \alpha_j^r F_j^{Lr}(\vartheta,\varphi,r) + \sum_j \alpha_j^{\vartheta} F_j^{L\vartheta}(\vartheta,\varphi,r) + \sum_j \alpha_j^{\varphi} F_j^{L\varphi}(\vartheta,\varphi,r)\right\}, \tag{6.1}$$

where the functions $F_j^{Lr}(\vartheta,\varphi,r)$, $F_j^{L\vartheta}(\vartheta,\varphi,r)$ and $F_j^{L\varphi}(\vartheta,\varphi,r)$ are defined as follows:

$$F_j^{Lr} = a\sum_{l,m}^{L} f_l(l+1)\left[\left(\frac{a}{r_j}\right)^{l+2} Y_l^m(\vartheta_j,\varphi_j)\right]\left(\frac{a}{r}\right)^{l+1} Y_l^m(\vartheta,\varphi), \tag{6.2}$$

$$F_j^{L\vartheta} = a\sum_{l,m}^{L} f_l\left[\left(\frac{a}{r_j}\right)^{l+2} \partial_\vartheta Y_l^m(\vartheta_j,\varphi_j)\right]\left(\frac{a}{r}\right)^{l+1} Y_l^m(\vartheta,\varphi), \tag{6.3}$$

$$F_j^{L\varphi} = a\sum_{l,m}^{L} f_l\left[\left(\frac{a}{r_j}\right)^{l+2} \frac{\partial\varphi}{sin\vartheta_j} Y_l^m(\vartheta_j,\varphi_j)\right]\left(\frac{a}{r}\right)^{l+1} Y_l^m(\vartheta,\varphi), \tag{6.4}$$

where $\sum_{l,m}^{L}$ denotes the double sum $\sum_{l=1}^{L}\sum_{m=-l}^{l}$, ϑ, φ and r are the geocentric colatitude, longitude and radius, a is the Earth's reference radius (6371.2 km), $Y_l^m(\vartheta,\varphi)$ are the usual Schmidt semi-normalised spherical harmonic functions of degree l and order m, L is the maximum degree of expansion of internal sources and $f_l = \frac{2l+1}{4\Pi(l+1)^4 l^2}$. The parameter f_l defines the shape of the localized functions.

Considering the above description of the magnetic field representation, and provided that the functions are defined at the observation locations, the fitting of three-component data as a potential field is possible, so long as the data contain spherical harmonic degrees less than or equal to L. Equation (1) is not time dependent and it determines a linear system of equations where α_j^r, α_j^ϑ and α_j^φ are unknowns, the data are magnetic field measurements and the matrix is expressed by combinations of the functions F_j^{Lr}, $F_j^{L\vartheta}$ and $F_j^{L\varphi}$. The solution of this linear system can be found directly. For time representation, each of the coefficients α_j^r, α_j^ϑ and α_j^φ is expanded on a basis of B-splines, i.e piecewise polynomials between spline knots:

$$\alpha_j^r = \sum_{k=1}^{n_{knots}} \beta_{k_j}^r b_k(t) \tag{6.5}$$

and α_j^ϑ and α_j^φ are expanded in a similar way.

No regularization was applied, i.e. the spatial smoothness of the model is determined by degree L and the temporal variability by the number of splines. To suppress the contributions of the lithospheric field to the derived core field model, the maximum degree of expansion of internal sources L was set to 12. The degree 12, with the minimum wavelength of 4000 km, is small enough to retain mostly the field contributions from the core. For the time representation, we use order six B-splines with 17 spline knots spaced at 1 year intervals between 2005.0 and 2011.0. A linear system is built from equations (1) and (5) and using a least squares method, we solve for the unknown model parameters $\beta^r_{k_j}$, $\beta^\vartheta_{k_j}$ and $\beta^\varphi_{k_j}$ as defined in eq. (5).

6.4 Data selection and processing

6.4.1 Vector satellite data

Data selection for magnetically quiet times are commonly done in order to eliminate external field influences as best possible from core field models when using satellite data (e.g. Lesur et al., 2011). We performed data selection on CHAMP vector magnetic data of 1 Hz sampling interval recorded between approximately 300 km and 380 km of altitude. Quiet night data recorded between 2005 and 2010 were selected over the southern African region covering the area between 16°S and 36°S in latitude and 10°E and 34°E in longitude. Only quiet time data corresponding to a Dst index between -20 and 0 nT measured during local night times between 20:00 - 06:00 were considered. The time interval was chosen to eliminate the influence of the solar quiet day variation. The external field has various sources with the ring current dominating at mid-latitudes. The Dst index, which is a proxy for the strength of the ring current (e.g. Saba et al., 1997), was therefore exclusively used in data selection. A trade-off was made between getting quiet time data and a good data coverage for setting the data selection criterion of Dst values to be between -20 and 0 nT (Nahayo et al., 2015).

The model input data were obtained by creating 525 data bins of 0.2°×0.2° on a grid of 1° of latitude and 1° of longitude. Additionally 120 data bins on a grid of 2° of latitude and 2° of longitude and offset by 0.5° were used to increase the number of data points. The middle of each bin is named a data center in the following. Six hundred and forty five (645) satellite data centers were created (fig. 6.2). We averaged data recorded at the same epoch (within 1 day) in each bin, which provided values at the same geographic location at different epochs and different altitudes. A data center thus can be considered similar to a geomagnetic repeat station, with the differences that the data might be measured at different altitudes between 300 and 380 km and that repeat intervals are not annual). The altitudes of the values in the same bin recorded at the same epoch were found to be very close (with and average scatter value of less than 10 m) so that no altitude correction was necessary before averaging. The altitude of the averaged value was taken as the average of altitudes of individual values in the bin recorded at the same epoch. Considering all used bins, the average scatter values between data values in the same bin recorded at the same epoch are 14.5 nT, 5.9 nT and 3.8 nT for X, Y and Z components, respectively.

6.4.2 Ground data

The hourly mean values derived from 1-minute data from the four continuous recording magnetic observatories Hermanus (HER), Tsumeb (TSU), Hartebeesthoek (HBK) and Keetmanshop (KMH) (also available at http://www.intermagnet.org) were selected using the same criteria as for satellite data selection. Monthly mean values were computed as averages of the obtained hourly values to get the input data for the model.

The field survey data collected at 38 geomagnetic repeat stations from South Africa, Namibia and Botswana were also used as input data (also available at http://www.geomag.bgs.ac.uk/data_service/data/surveydata.shtml). These data were collected by running a LEMI-008 fluxgate vector magnetometer over night and taking absolute observations in the evening and morning using the standard absolute equipment of a DI-fluxgate magnetometer (http://www.bartington.com/ mag-01h-declinometer-inclinometer-system.html) and an Overhauser magnetometer (http://www.gemsys.ca/technology/ground/). The absolute observations provide baseline values for the variometer values, which are averaged over the night hours between 20:00 - 06:00 LT to obtain the main field values at this geomagnetic repeat station for this particular night when the survey is conducted (Korte et al., 2007). The repeat stations thus provide approximately annually spaced values for quiet night times. The location of all ground stations is indicated in Fig. 6.2.

6.4.3 Model input data

The data used to produce the model fall in three categories, i.e., the monthly mean values from observatory data, the spot values from geomagnetic repeat stations and the averaged values from satellite data centers (Table 6.2). The selection of weights for the different data categories (see Table 6.2) was chosen assuming that the external field (noise) has cancelled out better in the ground data, in particular the time-averaged observatory data, than the satellite data and taking into account the different numbers of data with strong dominance of the satellite data. Remaining external field contributions to data were further removed using the comprehensive model phase 4 (CM4)(Sabaka et al., 2004; Onovughe, 2016). The computed CM4 external field values, primary and induced magnetospheric and ionospheric contributions, were removed from all data before the averaging process. The crustal field was removed from both satellite and ground-based data before the averaging process using the Potsdam Magnetic Model of the Earth (POMME-9) (Maus et al., 2006). This model includes in its representation the field contribution from crustal magnetization up to spherical harmonic degree and order 133 (300 km resolution) and the maximum expansion degree for the core field, that was not removed, was set to 15.

Table 6.2: The SACFM-2 model input data in 3 categories, satellite (SAT), observatory (OBS) and geomagnetic repeat station (GRS) data.

	SAT	OBS	GRS
Number of data centers	645	4	35
Number of data points	6150	290	210
Weight of data	25	100	50

6.5 Results

The SACFM-2 model is based on harmonic spline fitting of vector satellite and ground-based data recorded between January 2005 and December 2010. The root mean square (rms) values of the difference between input data and model values are 10.7 nT, 6.3 nT and 4.7 nT for the X, Y and Z components, respectively. Spline-based models often suffer from end effects and we therefore consider SACFM-2 to be reliable from 2005.5 and 2010.5.

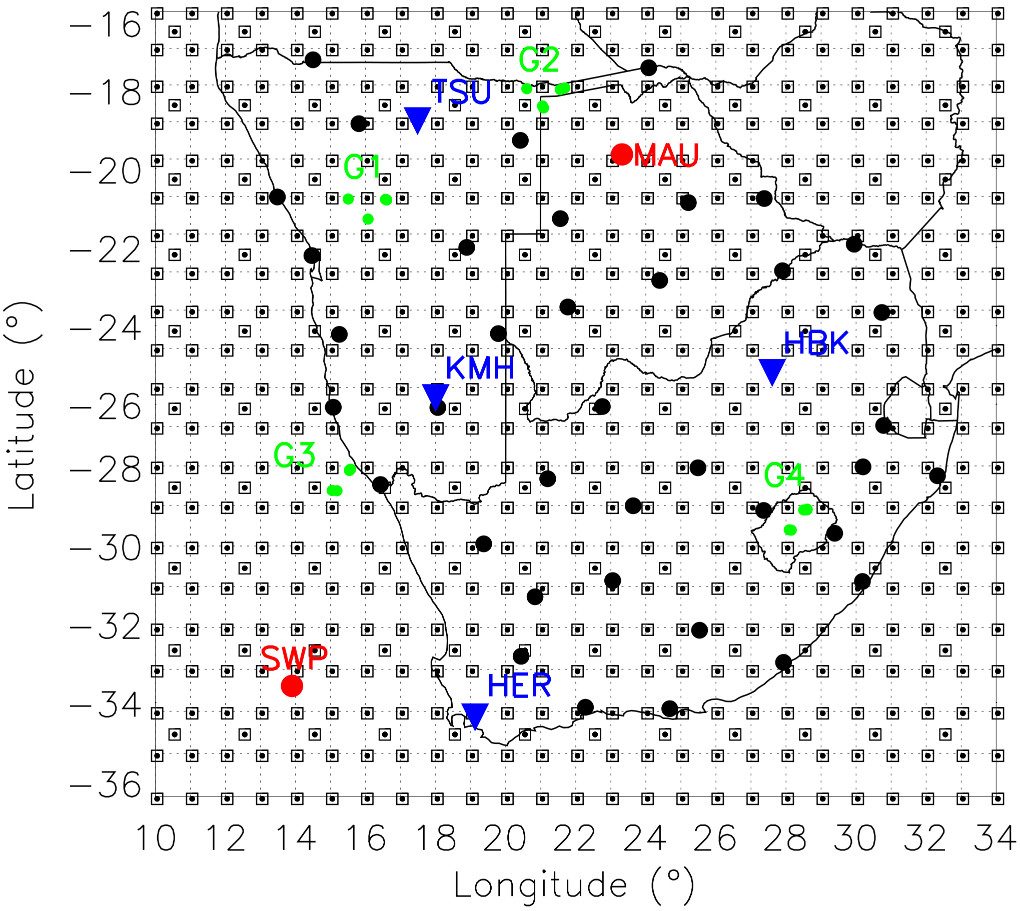

Figure 6.2: The data bins (0.2°×0.2°) used for satellite input data (represented by square shapes), the middle of each data bin represents a data center (small dots). The input data are the averages of recorded values within one day in the same bin at different altitudes between 300 and 380 km. The big black dots are geomagnetic repeat stations. The two red big dots are locations at the ground level used in the evaluation of the new SACFM-2 model. The four groups of green dots are the selected points for independent data at satellite altitude (G1, G2, G3 and G4) used in the evaluation of the SACFM-2 model. The four blue triangles are the permanent observatories HER, HBK, TSU and KMH.

A comparative evaluation of SACFM-2 and CHAOS-6 (Finlay et al., 2016b) at 0.8 km altitude (average altitude for ground-based data) and for all three components X, Y and Z over the whole area of study is presented in Tables 6.3 and 6.4 for main field and secular variation, respectively. CHAOS-6 is a global spherical harmonic model spanning the time interval 1999.0 to 2016.5. For the time of interest here it is based on monthly means from 160 ground observatories and CHAMP and Oersted satellite quiet time vector and scalar data. Differences in secular variation between the two models are generally small on average except towards the end of the time interval in X and Z. There is a systematic, temporally nearly constant difference between the two models of the same order of magnitude in all three components. Figure 6.3 shows comparative maps of all three field components from the two models for epoch 2008.0. Figure 6.4 presents the time evolution of the main field of X, Y and Z components for both models SACFM-2 and CHAOS-6 at four magnetic observatories in the region (HER, HBK, TSU and KMH).

Three groups of input data, the satellite data, the observatory data and the repeat station data were used to evaluate the SACFM-2 model in comparison to the regional model SARM (Nahayo et al., 2015) and global model CHAOS-6. The independent data at satellite level were also used to compare the performance of both SACFM-2 and CHAOS-6 models (Tables 6.5 and 6.6). SARM is built by the same modelling technique but based on satellite data only, and the spatial data coverage is less dense (data centers are on a grid of $2^\circ \times 2^\circ$ of latitude and longitude) with the maximum degree of expansion of internal sources set to 12. The same maximum degree 12 was used in getting synthesized data from CHAOS-6. SACFM-2, Quiet time external field correction and lithospheric field elimination had not been performed on the SARM data set. The results of rms values for this comparison are given in Tables 6.7 and 6.8. The rms fit to the input data of SACFM-2 is very close, in general below 12 nT, in all components and for all three data types. For the satellite data, the misfit of SARM is two to three times larger, and the misfit of CHAOS-6 is clearly much larger, in particular for X and Z. For the observatory data the fit of SACFM-2 is much better than that of SARM or CHAOS-6 in all components (less than 7 nT rms misfit in SACFM-2 compared to 43 to 168 nT in the other two models). CHAOS-6 fits this data type clearly closer than SARM in X and Z, with some apparent time dependence, and the fits to Y are of the same order for both these models. The misfit to the repeat station data are of similar magnitude for SARM and CHAOS-6, slightly lower for X in the latter, and notably lower in all components for SACFM-2.

In Figures 6.5 to 6.7 the secular variation from data, SACFM-2 and CHAOS-6 is shown for a couple of widely distributed locations. The secular variation values for each component were calculated from monthly observatory and model values as follows: $dB/dt = \dot{B}(t) = [B(t+6) - B(t-6)]/1year, (B = X, Y or Z)$, where the unit of t is months. SACFM-2 shows notably higher temporal variability than CHAOS-6, most pronounced in the X component. Figures 6.8 and 6.9 show the evolution of the Z component and total intensity F over the time interval 2005.5 to 2010.5 as predicted by the SACFM-2 model, indicating a weakening of the field strength over most of the studied area during that time.

Table 6.3: RMS values of the differences between SACFM-2 and CHAOS-6 main field models between 2005 and 2010 at 0.8 km altitude. These differences were computed using 684 data points over the modelled area.

Component	2005.5	2006.5	2007.5	2008.5	2009.5	2010.5
Δ X (nT)	92.6	89.9	93.1	92.0	88.0	88.4
Δ Y (nT)	91.3	89.0	83.7	82.9	82.9	80.8
Δ Z (nT)	87.5	88.3	89.1	86.9	85.3	80.6

Table 6.4: RMS values of the differences between SACFM-2 and CHAOS-6 secular variation models between 2005 and 2010 at 0.8 km altitude. These differences were computed using 684 data points over the modelled area. The secular variation values were computed by getting the change of the field between two consecutive epochs of one year interval (e.g. epochs 2006.0 and 2007.0 were used to get secular variation values at 2006.5 epoch).

Component	2006.5	2007.5	2008.5	2009.5
dX/dt (nT/yr)	3.7	4.3	4.2	10.1
dY/dt (nT/yr)	6.6	3.8	3.2	3.7
dZ/dt (nT/yr)	5.8	4.1	4.3	8.0

6.6 Discussion

The comparative evaluation of SACFM-2 using regional model SARM based only on satellite data and global model CHAOS-6 firstly shows a strong dependence of the different models on their input data (Tables 6.7 and 6.8). This is not surprising

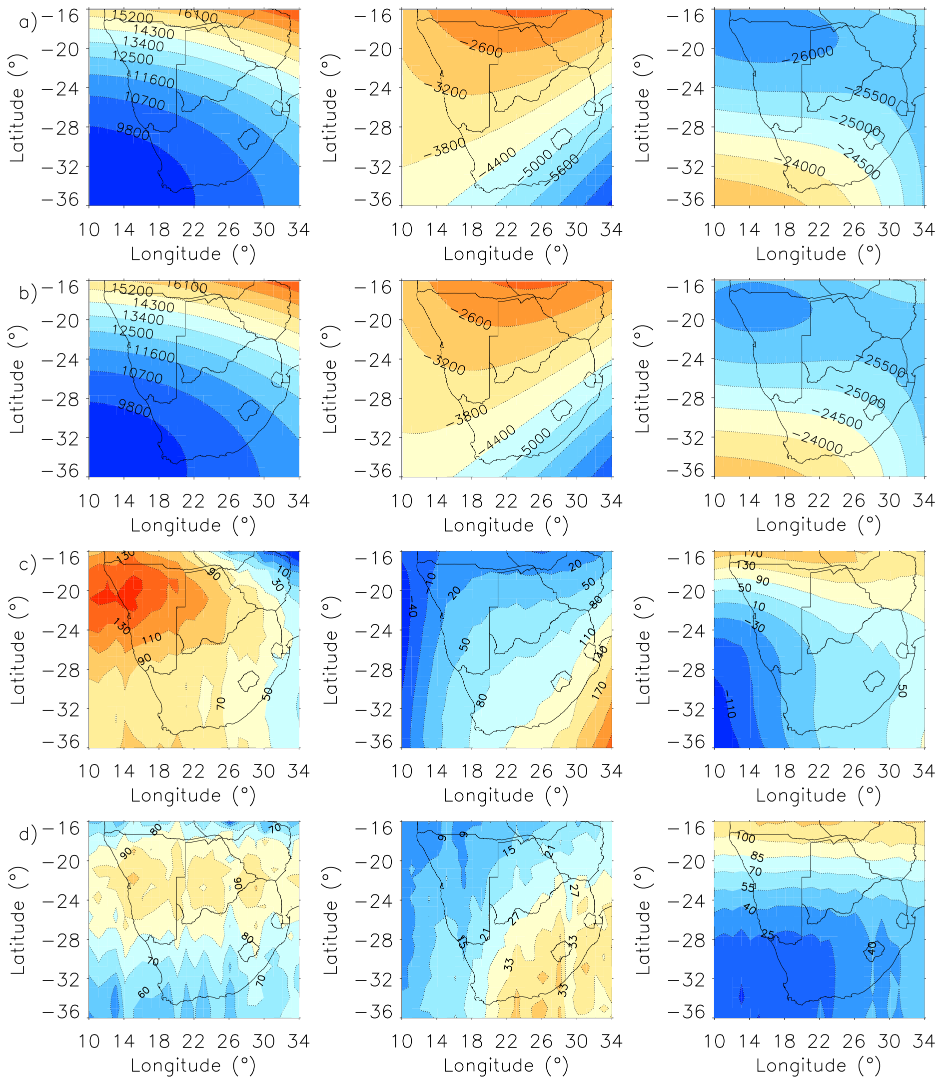

Figure 6.3: Comparison of models SACFM-2 (a) and CHAOS-6 (b) at epoch 2008.0, their differences are shown in the third row (c) and last row (d) at 0.8 km and 350 km altitudes, respectively. The three components X, Y and Z are plotted in the first, second and third columns, respectively. The unit is nanotesla.

and has to be kept in mind when using and interpreting models. The fact that the satellite data with external and lithospheric field elimination used here are fit less well by SARM than SACFM-2, which used satellite data without these corrections, indicates that SARM contains more external field leakage and/or lithospheric field influence and clearly is superseded by the new SACFM-2. External field signals in general tend to be stronger than lithospheric field signatures at satellite altitude, so the influence of the latter might be negligible, though. This is supported by

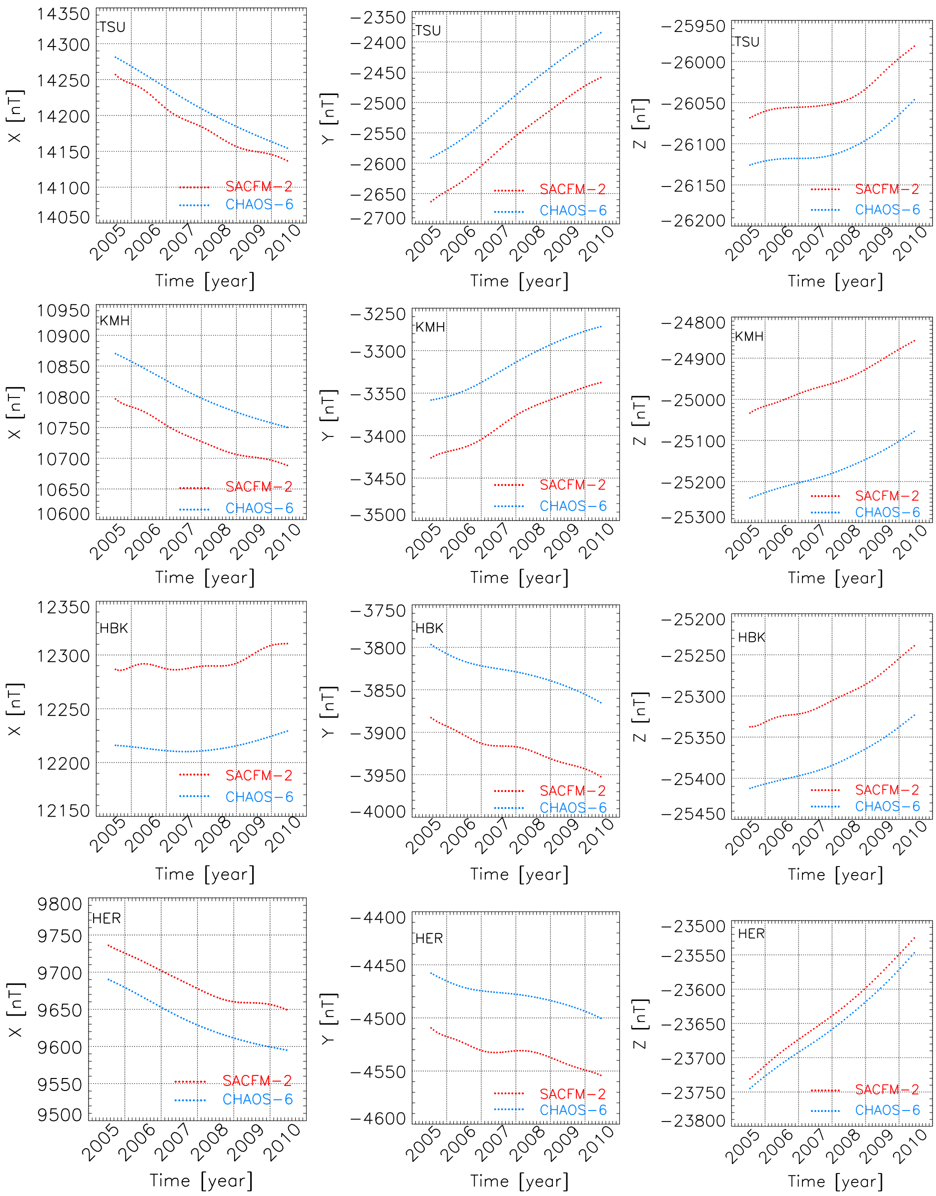

Figure 6.4: Time evolution of the main field of X, Y and Z components for SACFM-2 and CHAOS-6 between 2005 and 2010 at four magnetic observatories (TSU, KMH, HBK and HER). Synthesized main field data are calculated at the center of each month between 2005.5 and 2010.5. The three columns from left to right are for X, Y and Z components, respectively. The four rows from top to bottom are for TSU, KMH, HBK and HER observatories, respectively.

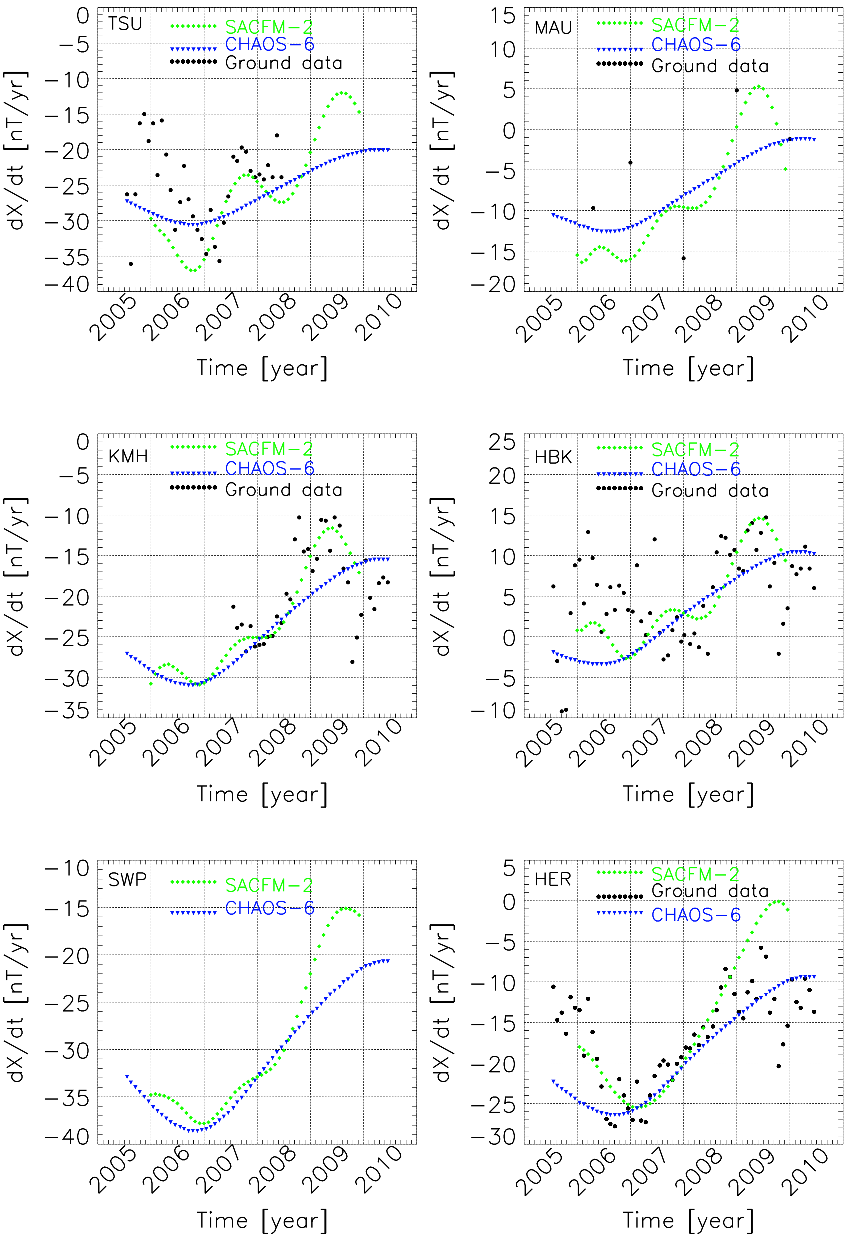

Figure 6.5: Secular variation of the X component at the four observatories (data in black) and two additional reference points between 2005 and 2010 as given by SACFM-2 (green) and CHAOS-6 (blue) models.

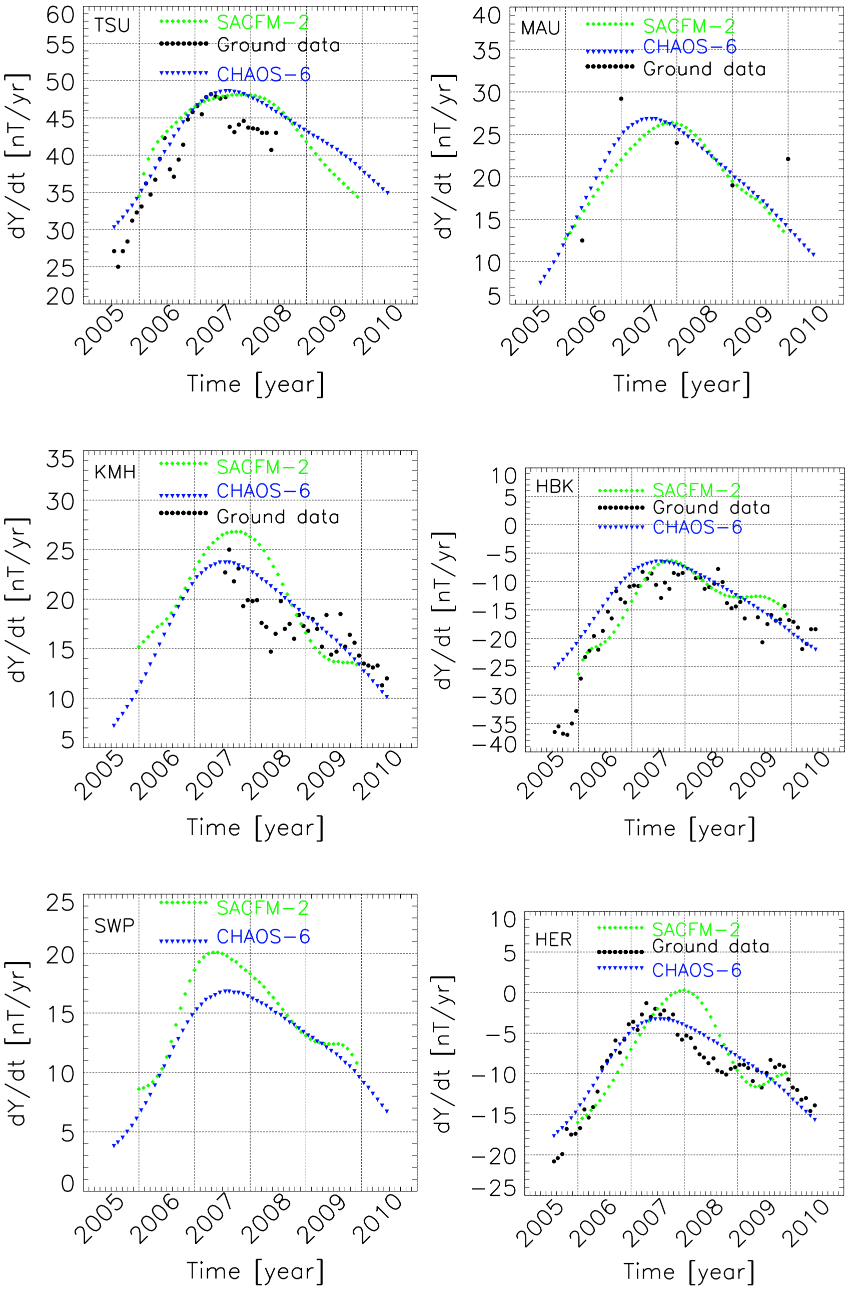

Figure 6.6: Secular variation of the Y component at the four observatories (data in black) and two additional reference points between 2005 and 2010 as given by models SACFM-2 (green) and CHAOS-6 (blue).

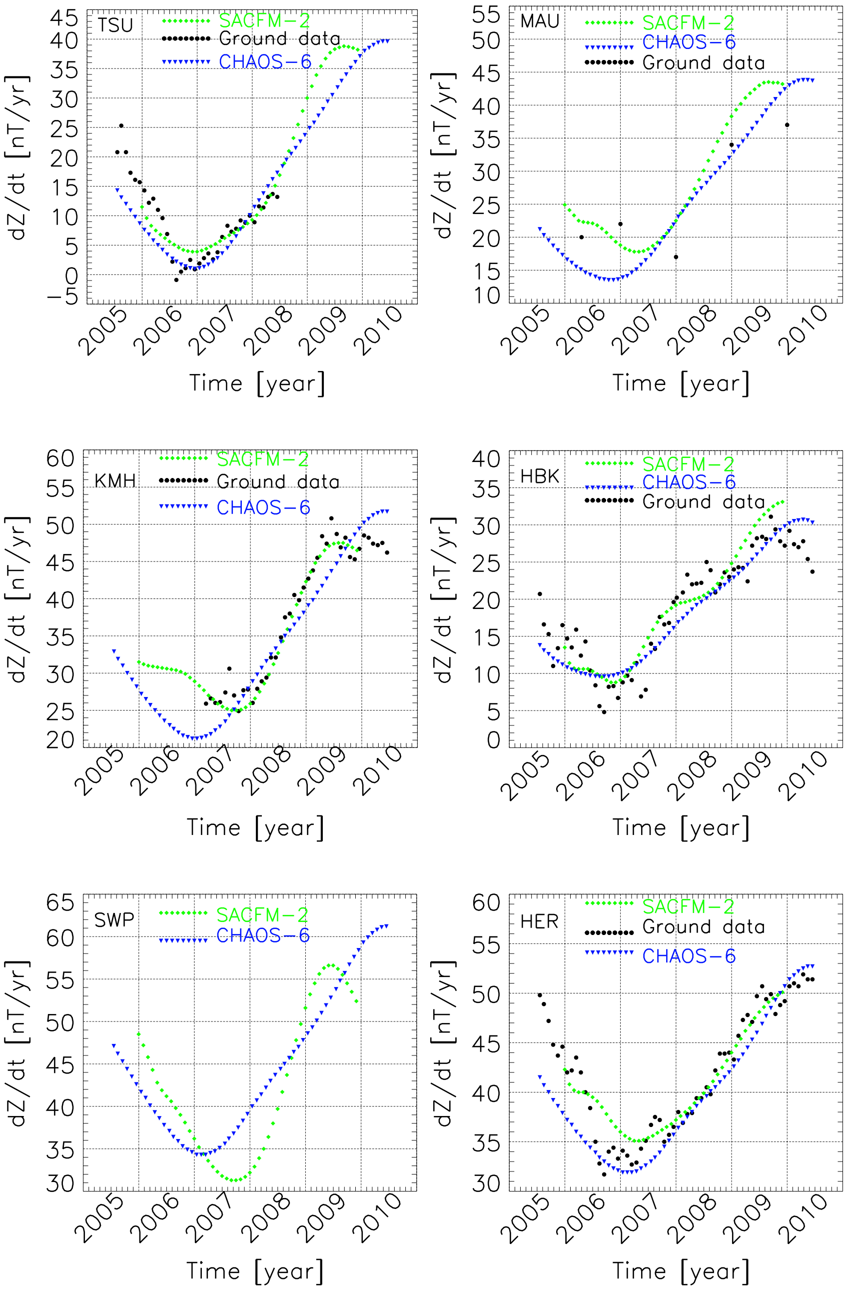

Figure 6.7: Secular variation of the Z component at the four observatories (data in black) and two additional reference points between 2005 and 2010 as given by models SACFM-2 (green) and CHAOS-6 (blue).

Table 6.5: The presentation of satellite data and computed data from the SACFM-2 and CHAOS-6 main field models for the Z component at 4 reference group points (G1, G2, G3 and G4) at selected epochs between 2005 and 2010. The Z component values are in nT.

	Col. (°)	Lon. (°)	Rad. (km)	Epoch	Satellite data	SACFM-2	CHAOS-6
	111.6037	16.0703	6730.2202	2005.943	-21602	-21605	-21531
	111.0930	16.6017	6736.3521	2006.825	-21525	-21532	-21453
G1	111.0943	16.5997	6726.6172	2007.452	-21631	-21640	-21562
	111.0807	16.5683	6717.5015	2008.126	-21727	-21738	-21661
	111.0707	15.5057	6700.3765	2009.501	-21888	-21895	-21824
	108.0810	21.6857	6737.7280	2005.559	-21255	-21256	-21155
	108.5707	21.0513	6741.6235	2006.353	-21277	-21278	-21184
G2	108.1033	20.6020	6719.1484	2007.984	-21471	-21472	-21378
	108.1257	21.5560	6707.0132	2008.787	-21598	-21601	-21497
	108.6230	21.0797	6699.3726	2009.545	-21736	-21737	-21644
	118.6240	15.1830	6731.6548	2005.926	-21297	-21296	-21271
	118.6183	15.0463	6737.7383	2006.778	-21222	-21225	-21201
G3	118.0837	15.5720	6729.0352	2007.334	-21352	-21351	-21327
	118.1008	15.5420	6706.0439	2008.877	-21527	-21530	-21505
	118.1150	15.5390	6701.1177	2009.501	-21547	-21552	-21532
	119.5990	28.0930	6728.9092	2005.995	-21891	-21901	-21850
	119.6070	28.1510	6743.4180	2006.329	-21771	-21774	-21737
G4	119.1018	28.5980	6729.5313	2007.312	-21924	-21942	-21888
	119.1053	28.5060	6712.1265	2008.497	-22056	-22067	-22017
	119.0860	28.6050	6693.4595	2009.874	-22199	-22205	-22165

Table 6.6: The RMS values of the differences between the SACFM-2 and CHAOS-6 main field models and measured satellite data at 4 reference group points (G1, G2, G3 and G4) for selected epochs in Table 5.

Component	SACFM-2 - Satellite data	CHAOS-6 - Satellite data
ΔX (nT)	5.7	85.7
ΔY (nT)	10.0	21.4
ΔZ (nT)	6.9	62.6

the observation that SARM shows the least fit to all ground data among the three models.

Interpreting the differences between the regional SACFM-2 and the global CHAOS-6 model (Tables 6.6, 6.7, 6.8 and Fig. 6.3) is less straightforward. Except for basis functions, details of quiet time data selection and data processing two further differences between these two models are that CHAOS-6 uses less ground information as it does not include repeat station data, and that CHAOS-6 should better minimize large-scale magnetospheric field leakage by taking this contribution into account separately from the core field in its parameterization. The main field description

Table 6.7: The RMS values of the differences between satellite or observatory input data and the three models, SARM, SACFM-2 and CHAOS-6 for the period 2005-2010. In 2005 and 2010, only the input satellite or observatory data recorded after 2005.5 and before 2010.5 were used, respectively.

	Component	2005	2006	2007	2008	2009	2010
	ΔX (nT)	28.1	27.0	24.0	22.3	24.6	29.4
SAT - SARM	ΔY (nT)	16.0	16.2	17.0	16.2	15.9	16.9
	ΔZ (nT)	16.2	14.0	14.6	13.1	13.3	16.1
	ΔX (nT)	10.2	10.7	10.0	11.1	11.7	12.3
SAT - SACFM-2	ΔY (nT)	6.9	5.2	5.6	7.1	6.9	8.1
	ΔZ (nT)	4.8	3.6	4.7	4.8	5.1	5.2
	ΔX (nT)	83.6	80.1	81.5	81.8	77.2	76.1
SAT - CHAOS-6	ΔY (nT)	23.4	24.2	23.6	24.2	27.9	28.7
	ΔZ (nT)	66.0	61.9	61.8	62.7	62.9	62.3
	ΔX (nT)	150.0	134.4	123.8	119.7	117.8	122.4
OBS - SARM	ΔY (nT)	61.0	55.8	57.9	54.9	43.1	54.1
	ΔZ (nT)	122.9	146.4	167.1	165.7	164.7	167.5
	ΔX (nT)	4.5	6.3	3.5	4.2	4.5	6.6
OBS- SACFM-2	ΔY (nT)	2.6	1.9	2.5	6.7	2.4	2.9
	ΔZ (nT)	2.1	2.1	1.9	2.6	2.3	2.3
	ΔX (nT)	52.8	57.7	61.8	61.5	66.6	49.8
OBS - CHAOS-6	ΔY (nT)	69.6	69.6	70.0	68.0	70.6	59.4
	ΔZ (nT)	58.7	98.3	121.5	126.0	141.9	105.0

Table 6.8: The RMS values of the differences between geomagnetic repeat stations input data (GRS) and the three models, SARM, SACFM-2 and CHAOS-6 for the period 2005-2009. The year 2010 is not included because the ground vector survey was carried out in the second half of the year, outside the interval where SACFM-2 is considered to be reliable.

	Component	2005	2006	2007	2008	2009
	ΔX (nT)	156.3	166.9	172.4	174.3	167.6
GRS - SARM	ΔY (nT)	124.2	119.3	121.8	130.0	120.3
	ΔZ (nT)	216.3	193.5	190.4	194.0	169.1
	ΔX (nT)	10.0	9.1	6.9	10.7	7.8
GRS - SACFM-2	ΔY (nT)	7.8	4.9	6.4	7.4	6.9
	ΔZ (nT)	9.2	4.8	5.4	6.9	8.1
	ΔX (nT)	87.6	100.1	102.6	107.1	101.5
GRS - CHAOS-6	ΔY (nT)	124.7	121.7	117.6	128.9	119.7
	ΔZ (nT)	206.4	185.3	178.9	191.0	159.7

of SACFM-2 might be influenced by local lithospheric field residuals at the ground stations that have not fully been eliminated in the data processing (fig. 6.4). This would not influence the secular variation of SACFM-2.

The observed model differences are mostly large-scale in all components (Fig. 6.3), yet do not show the geometry of ring-current dominated magnetospheric field, which could be assumed to leak more strongly into SACFM-2 than CHAOS-6. On the other hand, the scale of the differences argues against local lithospheric field influence in SACFM-2, although it seems surprising that spherical harmonic degree up to 133 as used for the lithospheric field elimination from the data should be enough to remove local crustal fields to residuals of less than 10 nT in ground data (Tables 6.7 and 6.8). Future expansions of the modelling method to co-estimate crustal biases for the ground stations (e.g. Geese et al., 2011; Talarn et al., 2017) might shed light on the influence of the lithospheric field on the regional model. Trends of increasing residuals to observatory data in X and Z from 2005 to 2008 in CHAOS-6 not seen in SACFM-2 might indicate a more accurate secular variation description by the regional model (Table 6.7). Figures 6.5 and 6.7 suggest that this might result from a slight lack of temporal variability in CHAOS-6 in the southern African region, where secular variation and its gradients are strong in global comparison. However,

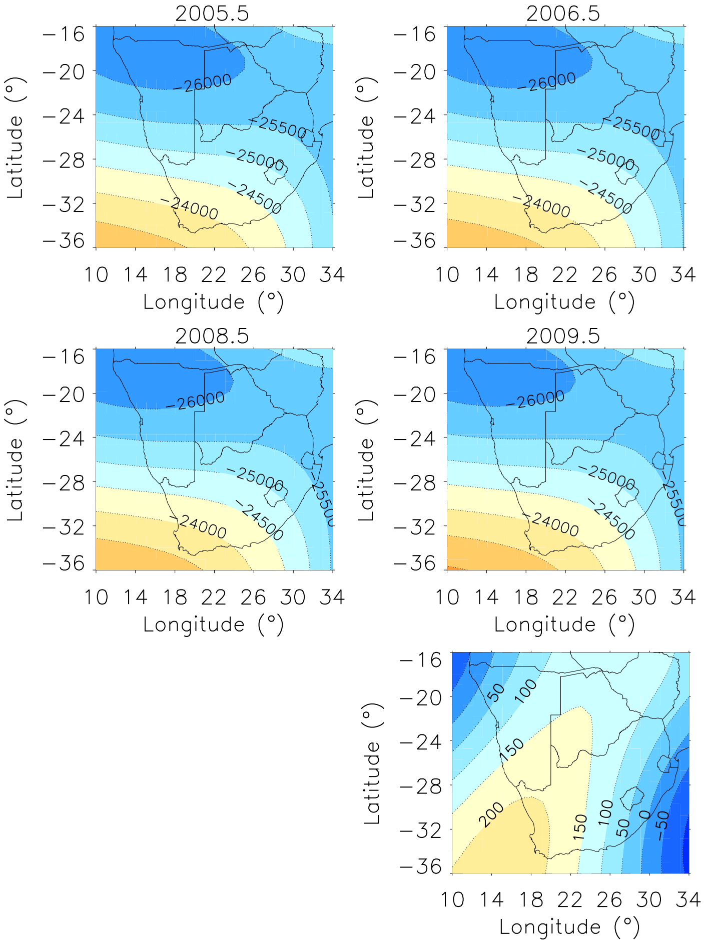

Figure 6.8: Maps of the Z main field model derived from CHAMP satellite and ground data at 6 selected epochs between 2005 and 2010 (first two rows) and the change of the field between 2005.5 and 2010.5 epochs (last row) at 0.8 km altitude.

a similar trend is only seen in the X component for the repeat station data (Table 6.8).

The comparative evaluation of the SACFM-2 using independent data at 4 groups of reference points at satellite altitude suggest that the regional model SACFM-2 performs better in all three components compared to CHAOS-6 (Table 6.6). The noise in X and Y might be the source of the deterioration of the SACFM-2 model at ground level in these components. The noise signal also amplifies with downward continuation.The inclusion of more ground data seems to have improved the model at the Earth's surface in many regions compared to downward predictions from a model from satellite data only, and helps to track the rapid secular variation observed in the southern African region.

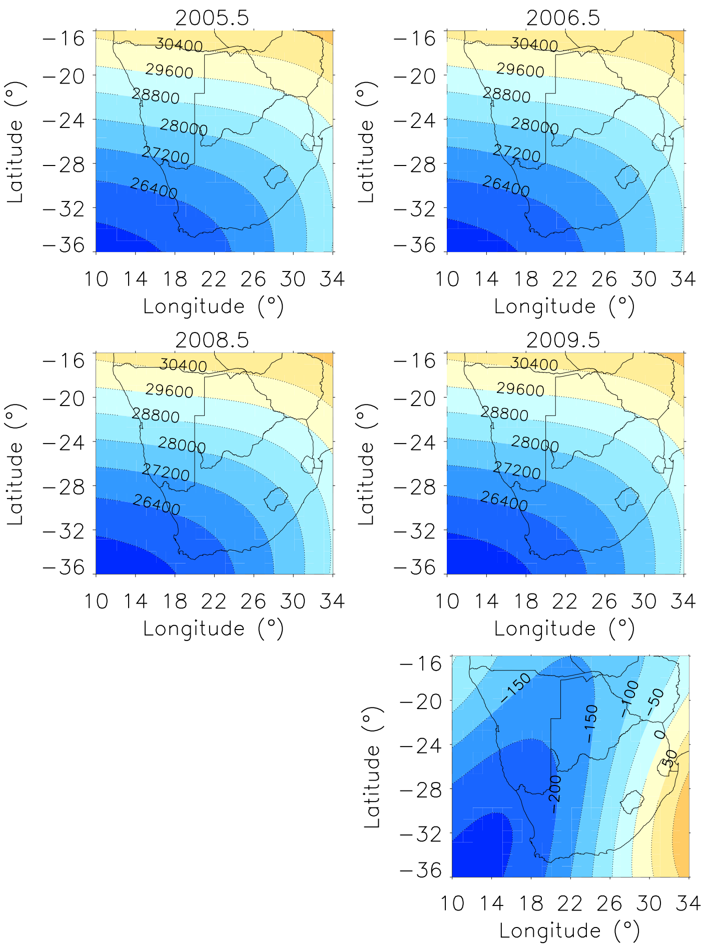

Figure 6.9: Maps of the F main field model derived from CHAMP satellite and ground data at 6 selected epochs between 2005 and 2010 (first two rows) and the change of the field between 2005.5 and 2010.5 epochs (last row) at 0.8 km altitude.

The vertical field and total intensity maps in Figs. 6.8 and 6.9, respectively, show a continuing decrease of field strength over southern Africa from 2005.5 to 2010.5, contributing to a deepening of the SAA. The decrease of the field in magnitude ranges between 0 to 200 nT and 0 to 250 nT in the Z component and total intensity F, respectively, with faster decrease in the south-west corner of the studied area. However, there is a very small area in the south-east corner that shows the increase in the field strength. The secular variation curves in Fig. 6.7 confirm that the decrease ranges between 0 and 55 nT/yr, with faster changes in the south (HER, SWP) than the north (TSU, MAU). Strong secular variation gradients manifest themselves in the observed patterns. In the northern part of the region, the rate of change in the Z component decreases from east to west (see MAU and TSU), while it decreases from west to east in the southern part (see KMH and HBK or SWP and HER). The rate of change from west to east increases in the X component (Fig. 6.5) and does not show a clear trend in the Y component (Fig. 6.6). The magnitude of the Y component is decreasing in the northern part and south western part of the region (see TSU, MAU and SWP) and increasing in the south eastern part (see HER and HBK). Both the global and the regional model (CHAOS-6 and SACFM-2) identify the 2007 geomagnetic jerk in Y and Z components (Fig. 6.6 and 6.7). There is also an indication of another geomagnetic jerk during 2009 of opposite sign to the 2007 jerk that is indicated by some of the SACFM-2 Z component curves (SWP, KMH, TSU and MAU). This confirms results by Chulliat et al. (2015) and Kotzé and Korte (2016), who used only ground-based field survey and observatory data.

6.7 Conclusions

We have presented a new regional magnetic field model for the southern African region, named SACFM-2, which for the first time incorporates both ground and satellite data. The ground data coverage is improved compared to global models by using data from 38 repeat stations distributed through South Africa, Namibia and Botswana. The use of both CHAMP satellite and ground-based data has shown the possibility to develop accurate regional models that can be used to characterise the variation of the geomagnetic field over a small area. A comparison of SACFM-2 to the global CHAOS-6 model showed similar results within the area of observation, revealing highly non-linear time-varying characteristics of the geomagnetic field in southern Africa between 2005 and 2010.

SACFM-2 supersedes the previous SARM regional model, which was based only on satellite data without elimination of external and lithospheric field influences during data processing. From comparisons we conclude that SARM suffered more strongly than SACFM-2 from external field leakage, while the global CHAOS-6 model might slightly underestimate secular variation in this area.

Like other models, SACFM-2 shows a steady decrease of the Earth's magnetic field over southern Africa, thus confirming the deepening of the SAA in this region.

The comparative evaluation of these two models with the help of measured satellite and ground-based data shows that the regional model SACFM-2 correlates better with observations than CHAOS-6 for the Z main field applications. The SACFM-2 fits all component data better at satellite level (Table 6.8), but noise apparently gets amplified by the downward continuation in X and Y in regions not well-constrained by ground data.

Given the high magnetic field horizontal gradient in Southern Africa, an increase in the number of data centers for better coverage might improve the model further. Moreover, an improved approach for data selection and noise removal is required to improve the horizontal components X and Y for downward continuation, and an extension of the modelling method to co-estimate crustal biases of the ground data might avoid lithospheric field leakage into the model. A careful monitoring of the SAA evolution is envisaged using high quality geomagnetic field observations by European Space Agency Swarm satellite constellation (Olsen et al., 2014), providing denser data coverage than a single satellite mission and an additional third observation level. Future models will expand our current knowledge about the magnetic anomaly region, its geographic expansion towards inside the African continent and how fast the field intensity decreases.

6.8 Acknowledgements

The support of the CHAMP mission by the German Aerospace Center (DLR) and the Federal Ministry of Education and Research are gratefully acknowledged. South African National Space Agency (SANSA) is acknowledged for high quality ground-based data. We thank Vincent Lesur and Anne Geese for the harmonic spline technique source code used to develop SACFM-2 model. We also thank Christopher Finlay for making CHAOS-6 model source code available. We acknowledge constructive comments from two anonymous reviewers.

Chapter 7

Lithospheric magnetic field modelling over southern Africa at satellite altitude

The contents of this chapter are the preprint of the paper that has been published in the South African Journal of Geology (2019, https://doi.org/10.25131/sajg.122.0012). Few changes following the comments and suggestions of one of this thesis Examiners were added to improve the manuscript. I conceived this study in discussion with my supervisors. Work for this manuscript and write-up was completed by me, with the remaining authors contributing to improve the manuscript.

Application of Spherical Cap Harmonic Analysis on CHAMP satellite data to develop a lithospheric magnetic field model over southern Africa at satellite altitude

Emmanuel Nahayo[1,2], Pieter B Kotzé[1] and Susan J Webb[2]

[1]South African National Space Agency (SANSA) Space Science, PO Box 32, Hermanus, South Africa, email: enahayo@sansa.org.za and pkotze@sansa.org.za

[2]University of the Witwatersrand, Wits, 2050, Johannesberg, South Africa, email: susan.webb@wits.ac.za

7.1 Abstract

We apply a Spherical Cap Harmonic Analysis technique on CHAMP satellite data recorded over southern Africa between 2007.0 and 2009.0 epochs, and develop a Southern African Lithospheric Magnetic Model (SALMM) at satellite altitude. The comparative evaluation of the SALMM with the global model MF7 shows a good agreement in the Y and Z field components that are not much contaminated by external field contributions. We use the Z lithospheric field map to confirm the prominent long-wavelength anomalies over the southern African region and its surrounding ocean areas, discussing the underlying geological and tectonic structures of the identified crustal anomalies.

7.2 Introduction

The lithospheric magnetic field is generated by magnetised rocks and depends on their mineralogy, temperature, chemical alteration, deformation, age and geomagnetic main field orientation (Dunlop and Özdemir, 2007). It is produced by the magnetisation carried by rocks in the crust and upper mantle. Most magnetic minerals on Earth are titano-magnetite and titano-hematite for which the Curie temperatures are 580° and 670° respectively for pure iron phases (Thébault et al., 2010). Materials have different structures of intrinsic magnetic moments that depend on temperature; the Curie temperature is the critical point at which material's intrinsic magnetic moments change direction. Permanent magnetism is caused by the fixed alignment of magnetic moments and induced magnetism is created when disordered magnetic moments are forced to align in an applied magnetic field. The Curie temperatures are met at depths of close to 30 km in continental regions and 6-7 km in the oceanic regions (Thébault et al., 2010). The rocks that are deeper, in the region where temperature is above the Curie point, are non-ferromagnetic.

The crustal field is referred to as the anomaly field. Crustal fields in a particular location are actually in excess or in deficit of the main field; this depends on the change (increase or decrease) in magnitude of the Earth's main field with respect to the expected value for that location. Then, terrestrial magnetic anomalies are positive or negative (Langlais et al., 2010). At some places on Earth they can be very intense at their source, up to 200 000 nT (Jankowski and Sucksdorff, 1996), and decrease rapidly as the distance to the source increases, depending on the spatial scale of the anomalies. At satellite altitudes (e.g. 400 km), crustal fields range between ± 20 nT (Mandea and Purucker, 2005). The Bangui magnetic anomaly (BMA) in central Africa is one of the most prominent magnetic anomalies in the world (Hemant and Maus, 2005). The origin of the BMA has been a debate among geoscientists. Regan and Marsh (1982) supported the source for this anomaly to be geological in origin, but Girdler et al. (1992) proposed a non geological origin of the anomaly, possibly an iron meteorite, as the source of the Bangui anomaly. Hemant and Maus (2005) state that the lower crust in some regions of the world can have significantly different magnetic properties than the corresponding upper part of the crust. Therefore, a mafic lower crust which is basically gabbroic in composition can explain the source for the observed strong anomaly over Bangui. The BMA has an amplitude of 28 nT at Magsat altitudes (around 400 km) and covers an area of 700 000 km^2 (Girdler et al., 1992). Another very strong magnetic anomaly in the world, reaching 27 nT at Magsat altitudes (Taylor and Frawley, 1987), is the Kursk magnetic anomaly (KMA), a territory rich in iron ores located in southwest Russia (Figure 7.1). The rocks found at the KMA are ferrugineous quartzites composed of alternating thin layers of quartz and iron oxides, and their origin is thought to be volcanogenic-sedimentary or chemogenic-sedimentary (Zhabin and Sirotin, 2009).

The lithospheric field results from the superposition of induced and remanent magnetisations. The magnetic properties of the Earth's sub-surface rocks play a major role in revealing the subsurface geology structure and its formation. The maps of magnetic field anomalies present information on different ranges of spatial wavelengths depending on which data (ground, marine, airborne and satellite data) are used. The ground, marine and airborne data contain information on short

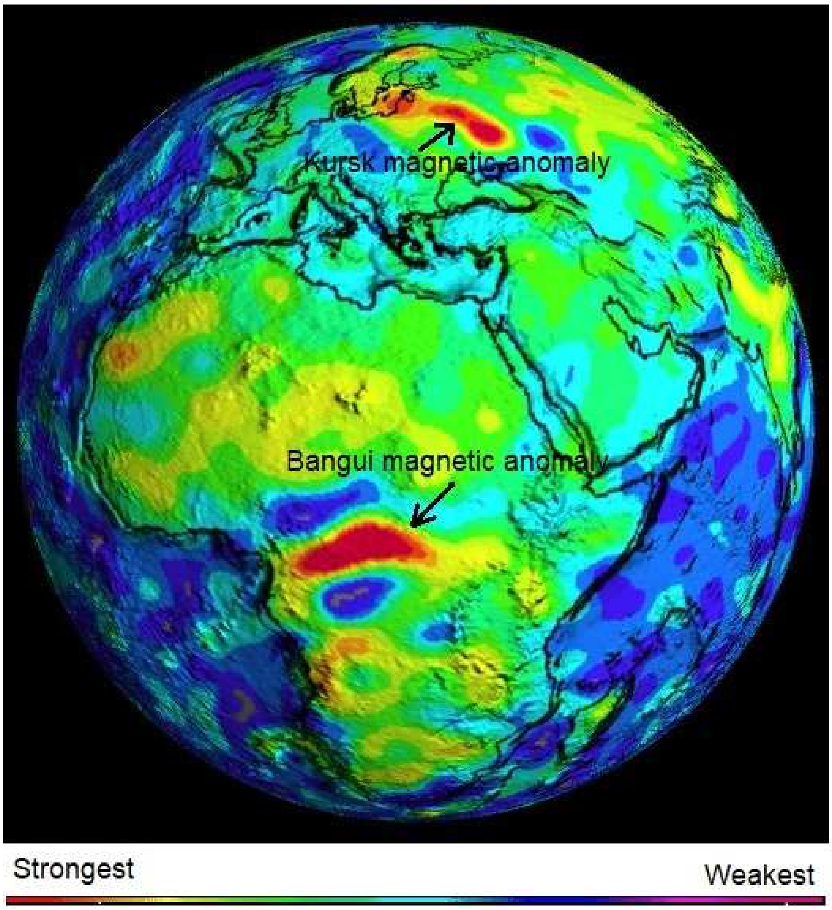

Figure 7.1: The map showing the Kursk magnetic anomaly, the high intensity (red) anomaly in the north reaching 27 nT (Taylor and Frawley, 1987), and the Bangui magnetic anomaly, the high intensity (red) in central Africa reaching 28 nT (Girdler et al., 1992) at 400 km altitude. The map shows the results of a model of the magnetisation of the Earth's crust, based on measurements of the magnetic field made by NASA satellites Magsat, OGO-2, OGO-4 and OGO-6 (Purucker, 2013).

spatial wavelengths and are used to reveal magnetic anomalies originating from a source close to the Earth's surface (less than 500 meters). The satellite data help in revealing magnetic anomalies with long spatial wavelengths (greater than 300 km), magnetic anomalies at greater depths (more than 500 meters beneath the Earth's surface) and regional geological structures (Korte and Mandea, 2016).

The Earth's magnetic field anomalies play an important role in revealing information necessary to study crustal structures. The maps of magnetic anomalies are helpful in understanding geological conditions and tectonic structures. A high resolution map of magnetic anomalies of the Earth can be used in some modern topics in geosciences, covering both fundamental and applied research, such as studying the structural and petrophysical characterisation of the crust, its rheology and geodynamic evolution, or even hydrocarbon exploration (Gabriel et al., 2011).

The CHAMP satellite data selection is summarised in the section "Data and existing models" and the method is described in the section "Method". The new lithospheric model, Southern African Lithospheric Magnetic Model (SALMM) and its fit to input model data is presented and compared to the global model MF7 in Figure 7.7 in the section "Results", and further discussed in the section "Discussion". Finally, the prominent long-wavelength anomalies over southern Africa and its surrounding ocean areas are confirmed and discussed, focussing on the underlying geological and tectonic features, before concluding with suggestions of improving the Spherical Cap Harmonic Analysis (SCHA) regional lithospheric field model in future.

7.3 Data and existing models

Satellite measurements are used to map efficiently the long-wavelength crustal fields that are of interest for geologic and tectonic studies at a regional scale (Maus et al., 2007). This paper presents the study of crustal magnetic anomalies in southern Africa at satellite altitudes using low-Earth orbit CHAMP satellite data recorded between 2007.0 and 2009.0 epochs. The focus is only on long spatial wavelength anomalies over southern Africa.

Previously, Antoine (1990) developed a lithospheric magnetic field model over southern Africa using a 6-month long recorded scalar data set by the Magsat satellite mission (Figure 7.2) and showed a strong correspondence between the southern Africa Magsat anomalies and tectonic provinces. Kotzé (2002) modelled and analysed Ørsted total field data (recorded around 700 km altitude) to produce a lithospheric field map over southern Africa, and identified the prominent long-wavelength anomalies in the region, including the Walvis Ridge and Agulhas anomalies.

CHAMP satellite magnetic observations with greatly improved measurement accuracies and spatial coverage (Maus et al., 2002) are used to develop a lithospheric magnetic field model over southern Africa. A recent study of the lithospheric mag-

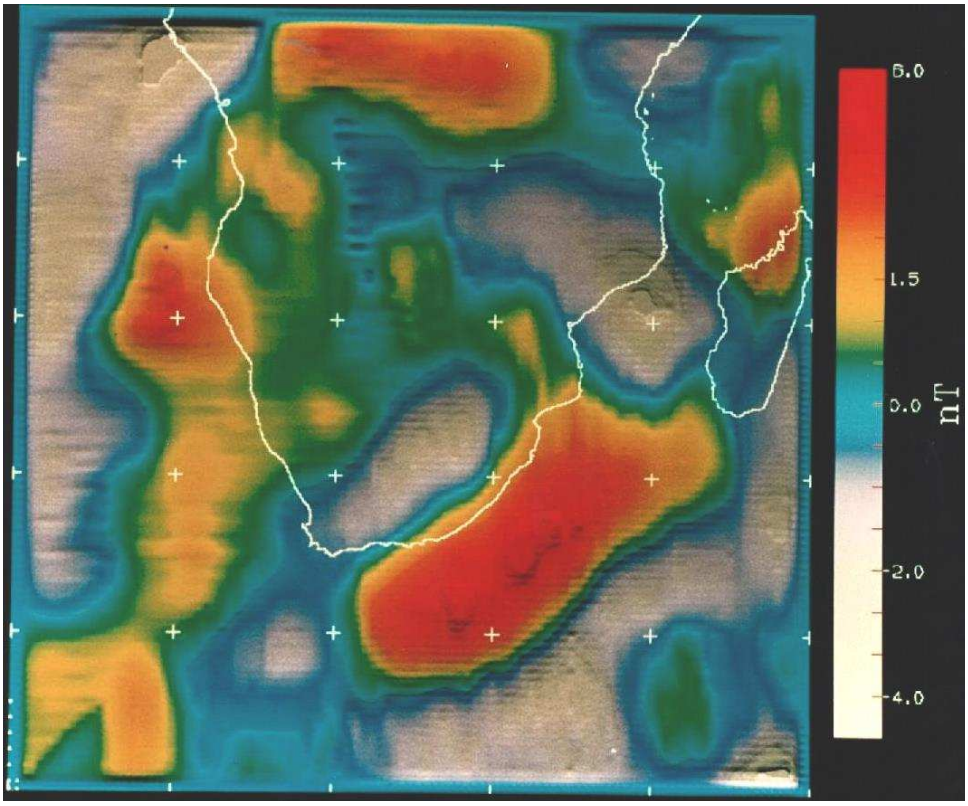

Figure 7.2: The total field anomaly map over southern Africa at 400 km altitude (Antoine, 1990).

netic field over southern Africa was conducted by Vervelidou et al. (2018) using data from CHAMP, Swarm, World Digital Magnetic Anomaly Map (WDMAM) and ground magnetic data. This high-resolution lithospheric magnetic field model, with the maximum spherical harmonic degree of 340, was developed applying the Revised-Spherical Cap Harmonic Analysis (R-SCHA) on data distributed at different altitudes between 0 and 480 km. The summary of the results of this study are shown in Figure 7.3 and they are used for a comparative evaluation of the new model.

The measured magnetic field above the Earth's surface is the superposition of fields of various origins including internal and external sources. The effects of the external currents from ionosphere and magnetosphere are minimised by selecting only quiet and night data values with the help of Disturbance storm time (Dst) magnetic activity index. The Dst was found to be a suitable index as it describes the ring current, which is the dominating external field source at mid-latitudes.

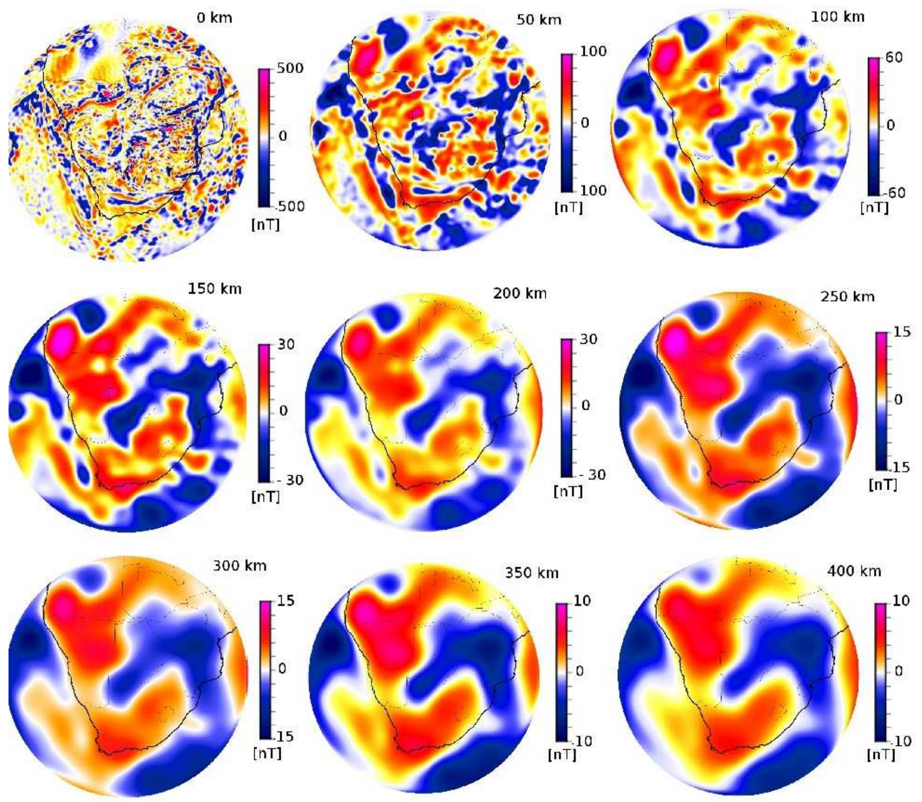

Figure 7.3: The Z component of lithospheric field model plotted at different altitudes above the Earth's mean reference radius; Figure 5 by Vervelidou et al. (2018).

Then, the remaining task becomes to separate the core and the crustal fields. The internal geomagnetic field comprises the main field (core field), the lithospheric field, and the contributions of the fields induced by time-varying external sources. The core field dominates at long wavelengths (greater than 2500 m) whereas the shorter wavelengths (less than 2500 m) are dominated by the lithospheric field (Maus et al., 2006). According to the interpretation of the spatial magnetic power spectrum derived from Magsat by Langel and Estes (1982), the core field dominates the spectrum from harmonic expansion degrees less or equal to 12 and the crustal field dominates the spectrum for degrees greater or equal to 16. In this study, the lithospheric field residual data are computed by subtracting from the data the core field contribution obtained from the GRIMM-3 core field model (Lesur et al., 2011) where the degree of expansion of internal sources is set to 15. After modelling residual data using Spherical Cap Harmonic Analysis (SCHA), the results are evaluated with the help of existing lithospheric models comprising local models and the global model MF7. The lithospheric magnetic field model MF7 was developed using CHAMP satellite data recorded between May 2007 and April 2010, and estimated to a harmonic degree 133, corresponding to 300 km wavelength resolution (Maus, 2010).

Data selection was performed on CHAMP vector magnetic data of 1 Hz sampling interval recorded between approximately 332 km and 365 km of altitude (Figure 7.4). Quiet night data recorded between 2007.0 and 2009.0 epochs were selected over the southern African region covering the area between 15° S and 40° S in latitude and 10° E and 38° E in longitude. Positive Dst values were not considered in the choice of the range of Dst index values for solar quiet data selection. Positive variations in Dst index are mostly caused by magnetospheric compressions resulting from interplanetary shocks that often occur in the initial phase of magnetic storms (Karinen and Mursula, 2005). Only quiet time data corresponding to a Dst index between -20 and 0 nT measured during local night times between 22:00 and 05:00 LT were considered. Remaining external field contributions to data were further removed using the comprehensive model phase 4 (CM4)(Sabaka et al., 2004; Onovughe, 2016). The computed CM4 external field values comprise primary and induced magnetospheric and ionospheric contributions. The contamination from external field contributions is particularly large on the X component, due to the geometry of auroral electrojets and equatorial electrojet (Chulliat et al., 2010). The X component is more affected than other field components by the external field contributions and the techniques of reducing noise like night quiet data selection and the use of the CM4 model are not very efficient to remove all the noise (Nahayo et al., 2018). Therefore, further data cleaning was performed to reduce noise in data for the X component by dropping all records with the X component absolute value greater than 15 nT, which resulted in 16% of total records being dropped. The model input data were selected by retaining data recorded on a grid of $0.3° \times 0.3°$ within distance of 0.1° of latitude and longitude from the grid points. The total number of data points (61670 triplets of vector data) are plotted in Figure 7.5. This approach of selecting data on the grid was performed to get evenly distributed data over the study area. The residual data were calculated by subtracting from the data the internal field contribution using the global model GRIMM-3 (Lesur et al., 2011), with a maximum degree of expansion of spherical harmonic functions set to 15.

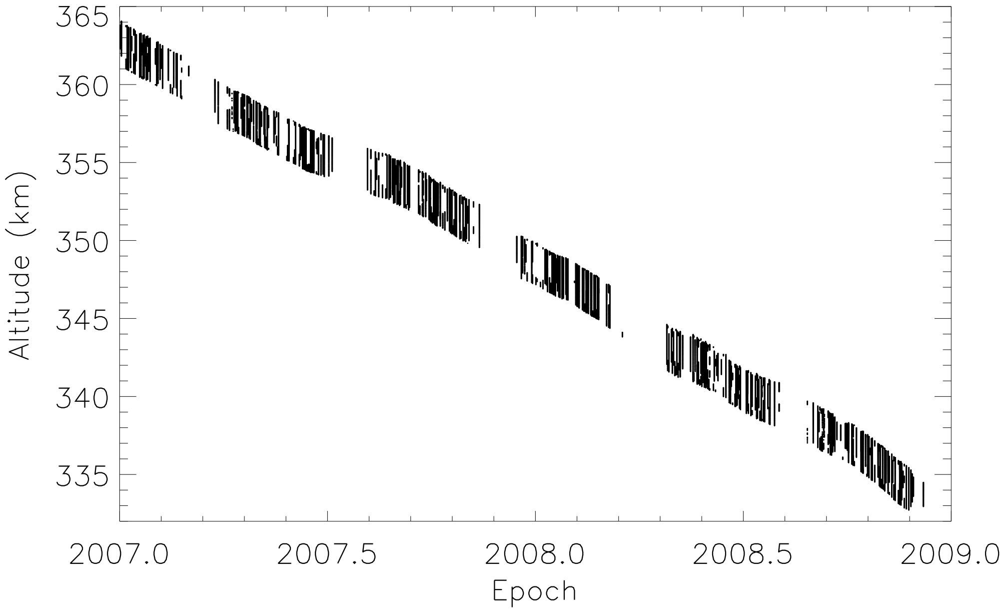

Figure 7.4: The flight of CHAMP satellite between 2007.0 and 2009.0 epochs using only the model input data.

7.4 Method

SCHA is a regional modelling technique founded on appropriate functions which are solutions of Laplace's equation over a restricted, cap-like region of the Earth. This mathematical technique was developed by Haines (1985) to overcome the non-orthogonality problem in the case of global spherical harmonic models when applied to restricted areas. SCHA is used to model a potential field and its spatial derivatives on a regional scale.

The solution to Laplace's equation over a spherical cap for internal sources (Haines, 1988) is given in the equation below:

$$V^i(r,\theta,\phi) = a \sum_{k=0}^{KINT} \sum_{m=0}^{k} \left(\frac{a}{r}\right)^{n_k(m)+1} P^m_{n_k(m)}(cos\theta)[g_k^{m,i} cos(m\phi) + h_k^{m,i} sin(m\phi)] \quad (7.1)$$

where r, θ, ϕ are the geocentric spherical coordinates radius, colatitude, and longitude; a is the reference radius; and $P^m_{n_k(m)}(cos\theta)$ is the associated Legendre function of the first kind with integral order m and real degree $n_k(m)$. The parameter k is an ordering index. $KINT$ is the maximum index for internal sources. The

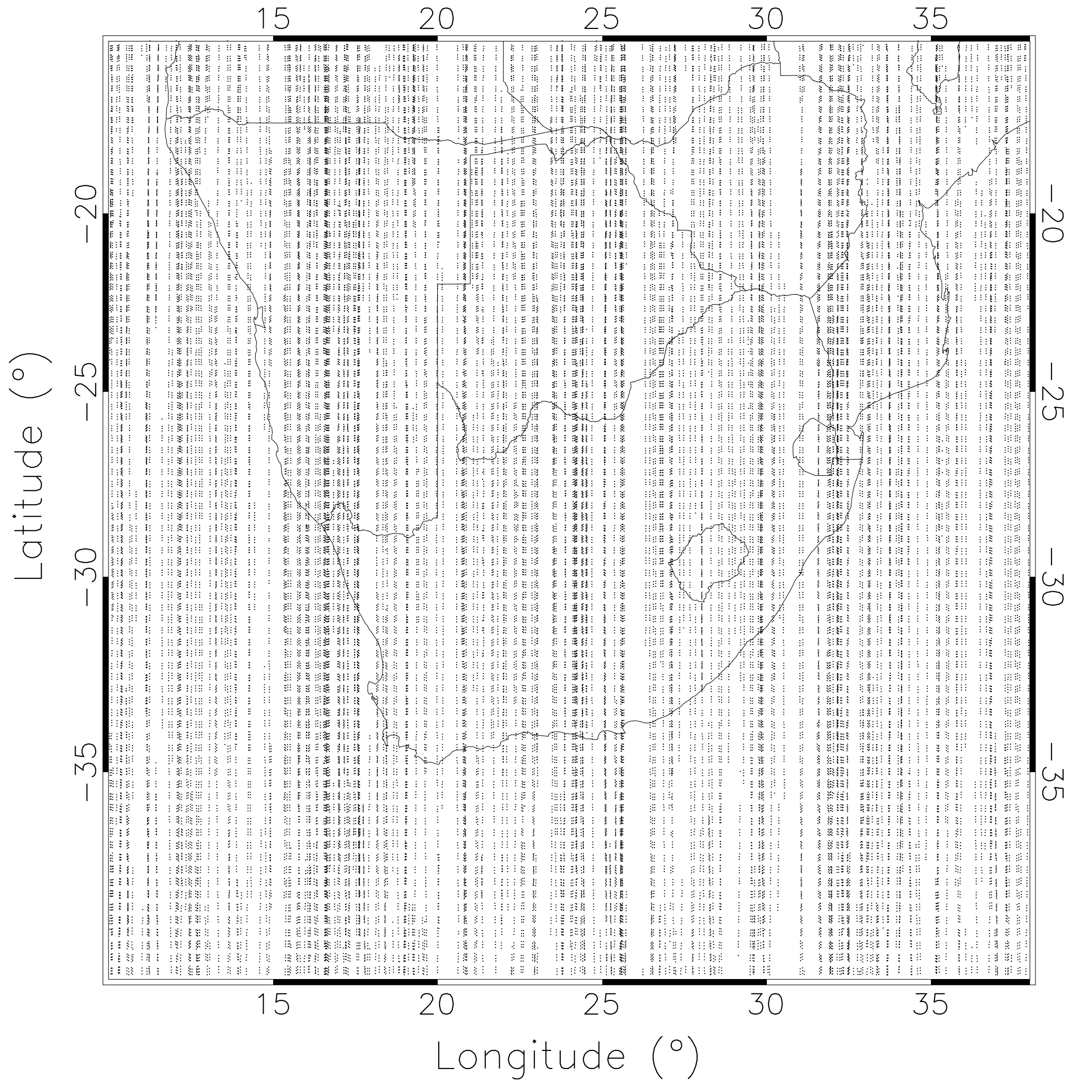

Figure 7.5: The plot of data points selected for the SCHA model. The presented data points are between 332 and 365 km altitude. The selection of data was on a grid of $0.3° \times 0.3°$ of latitude and longitude. All data points within $0.1°$ from the grid points were selected. Data records with absolute values of the X component greater than 15 nT were dropped.

constants $g_k^{m,i}$ and $h_k^{m,i}$ are spherical cap harmonic coefficients for internal sources.

Haines (1988) states that a solution of the above equation is termed a potential field, and the gradient of a scalar potential field is a vector field whose curl and divergence are both identically zero. The solution of Laplace's equation then provides a method of modelling such a vector field not only on a surface but also throughout space (Figure 7.6).

If the half-angle of the spherical cap is denoted by θ_0, then $n_k(m)$ are determined

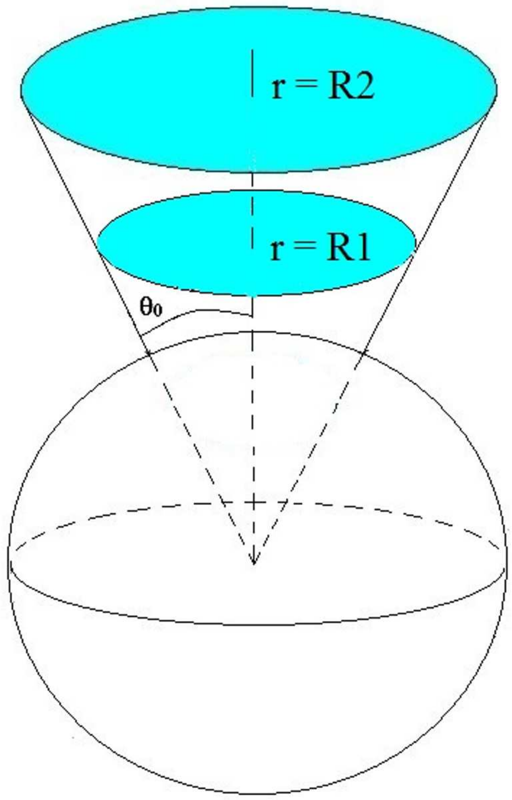

Figure 7.6: Spherical cap of half-angle θ_0. Data are distributed between surfaces r = R1 (lowest satellite altitude) and r = R2 (highest satellite altitude).

as the roots of the equation, for given m:

$$\frac{dP^m_{n_k(m)}(cos\theta_0)}{d\theta} = 0,\ k - m = \text{even} \tag{7.2}$$

and additionally, if differentiability with respect to θ is required:

$$P^m_{n_k(m)}(cos\theta_0) = 0,\ k - m = \text{odd}. \tag{7.3}$$

If the expansion in Equation 7.1 is truncated at k = KINT, the number of model coefficients is $(KINT + 1)^2$ and the coefficients are determined using the method of least squares.

The residual data were converted from a geocentric coordinate system to a new pole at 27.5°S and 24.0°E, which is the centre of the spherical cap. The SCHA technique has some limitations: Torta and De Santis (1996) showed that, when internal and external coefficients are used separately, they are not able to accurately describe the respective contributions. The solution was the use of an artificial cap that is larger than the area covered by the available data. In this study, the selected cap of a half-angle of 18° was considered to have a larger cap than the studied area to avoid the shortcomings of the SCHA technique. SCHA has shown other problems of modelling efficiently multilevel data sets because its coefficients are altitude dependent, and its model is barely extrapolated to other altitudes when the spherical caps are small (Thébault and Gaya-Piqué, 2008). The altitude range of considered CHAMP satellite data is too small (33 km) to significantly affect much the performance of SCHA and the aim of this study is to model the lithospheric magnetic field at satellite altitude.

The selection of the suitable ordering index for internal sources KINT and half-angle θ_0 for the selected data set is presented in Table 7.1. KINT was set to 6 for a stable SCHA model with a half-angle of 18°, and the lithospheric magnetic field wavelength represented in the model is 1250 km. The combination of KINT = 6 and $\theta_0 = 17$ yielded the same root mean square (rms) values rounded at 1 decimal point, but $\theta_0 = 18°$ was chosen to minimise the inherent edge effects of the SCHA regional model (Table 7.1).

7.5 Results

The developed model is based on SCHA fitting of lithospheric residual vector data. The rms values of the difference between input data and model values are 6.3 nT, 4.2 nT and 3.4 nT for the X, Y and Z components, respectively. The lithospheric model data were computed at 350 km altitude. Figure 7.7 shows the maps of X, Y and Z field components at 350 km altitude for the comparative evaluation of the results with the help of the global model MF7.

7.6 Discussion

The results of the modelling of the residual lithospheric field data of three magnetic field components are shown in Figure 7.7. The observed misfit is due to

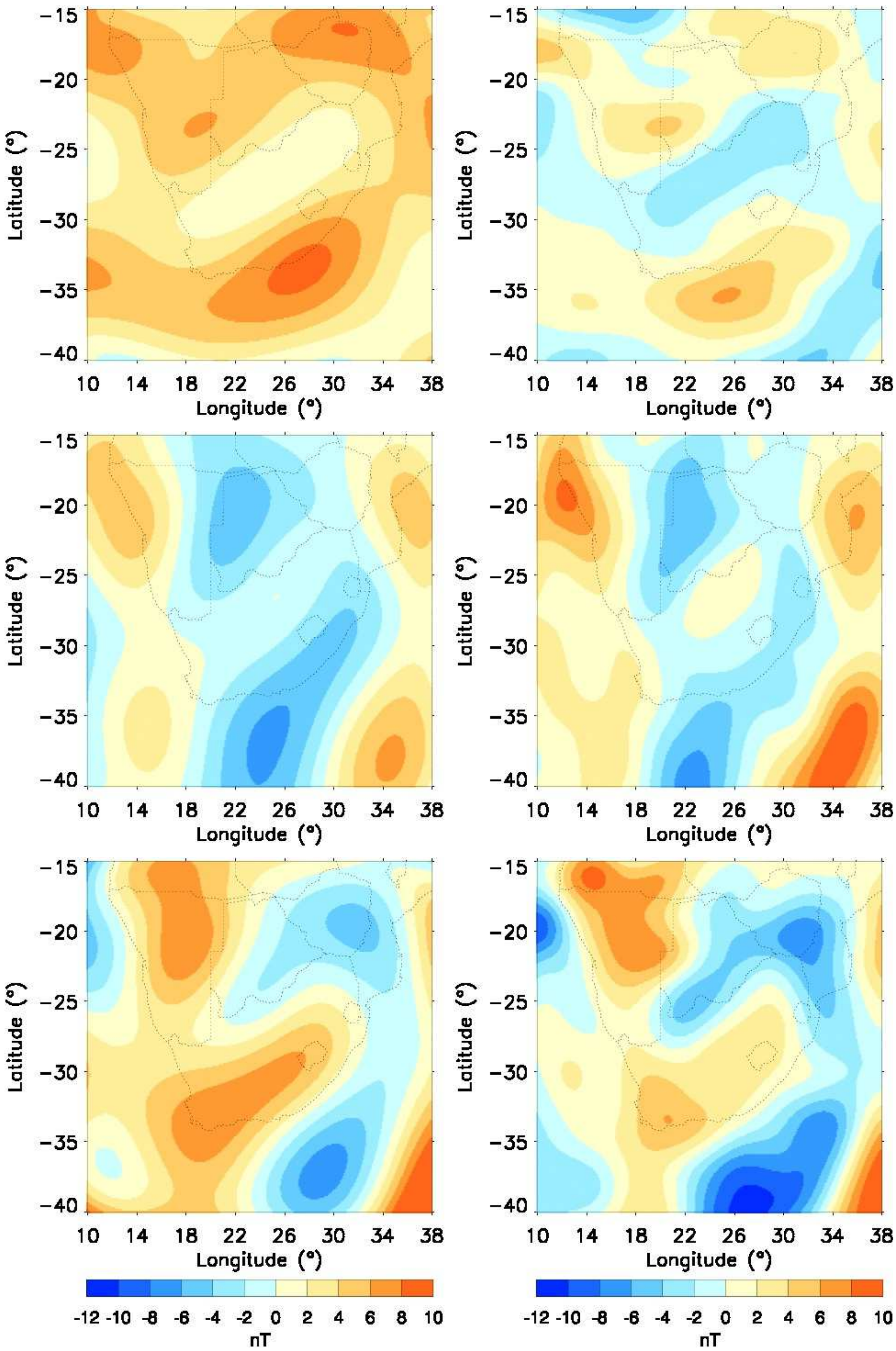

Figure 7.7: Comparison of developed regional model SALMM (left) and global model MF7 (right). The top, middle and bottom rows are the X, Y and Z field components, respectively. The anomaly maps are produced at 350 km altitude.

Table 7.1: The selection of ordering index KINT and the half-angle θ_0. The rms values for X, Y and Z field components of differences between model input data and computed SCHA model data are calculated for each ordering index KINT from KINT= 4 to 7 and for each half-angle θ_0 from $\theta_0 = 15$ to 19. The rms values are in nT.

KINT		$\theta_0 = 15$	$\theta_0 = 16$	$\theta_0 = 17$	$\theta_0 = 18$	$\theta_0 = 19$
	X	6.5	6.5	6.7	6.4	6.4
4	Y	4.6	4.5	4.5	4.5	4.5
	Z	4.0	3.9	3.8	3.8	3.8
	X	6.5	6.4	6.4	6.4	6.4
5	Y	4.5	4.4	4.3	4.3	4.3
	Z	3.7	3.7	3.6	3.6	3.6
	X	6.5	6.4	6.3	6.3	6.4
6	Y	4.4	4.2	4.2	4.2	4.3
	Z	3.9	3.7	3.4	3.4	3.6
	X	16.3	90.9	7.4	6.7	12.7
7	Y	16.8	58.3	4.7	4.5	11.6
	Z	35.7	94.1	5.1	4.2	13.4

external field contributions that were not fully removed from the data. All three components broadly reproduce the features that are in the MF7 model maps (Figure 7.7), but the X component map presents mostly positive anomaly in contrast to what is in the MF7 model. The Y and Z components from regional and global models agree well with minor differences, where some parts of the map would be negative on one side and positive on the other side. The results are supported also by the recently developed regional lithospheric field model by Vervelidou et al. (2018) in the Z component, and Figure 7.3 shows the Z component map at 350 km with similar features. This new regional lithospheric field model over southern Africa is an improvement over previous regional models focusing only on long wavelength field anomalies (Antoine, 1990; Kotzé, 2002). With improved data quality of CHAMP satellite measurements, it was possible to shed light on long wavelength field anomalies over southern Africa in X, Y and Z components at a satellite altitude of 350 km.

In this discussion, the reader should be aware that, if scalar total field data are used in studying field anomalies in this region, the mentioned anomalies are for the total field F and they happen to be opposite in sign to the anomalies in the Z component, that are going to be discussed here. The prominent long-wavelength field anomalies over southern Africa are easily identifiable in the Z lithospheric field map. With the help of Figure 7.8, presenting the Z map together with the outline of large-scale tectonic features, some magnetic field anomalies are identified and discussed. These magnetic field anomalies comprise the oceanic negative anomalies in the Z component bordering the southern Africa. These anomalies are the Agulhas and Walvis ridge anomalies. The Agulhas anomaly is located in the southwest Indian Ocean where the Agulhas plateau rises 2.5 km above the surrounding ocean floor (Gohl and Uenzelmann-Neben, 2001), and the Walvis ridge anomaly is just off the Namibian coast where the Walvis rise meets the African continent (Figure 7.8). The Agulhas plateau consists of overthickened oceanic crust and was formed as part of a Large Igneous Province (LIP) between 100 and 94 ($\pm$5) Ma (Parsiegla et al., 2008). LIPs result in massive volcanic eruptions that occur during a relatively short time (1-5 Ma) and form an anomalously thick oceanic crust (Bryan and Ernst, 2008). The Agulhas plateau has an average crustal thickness of 20 km, and has some layers of volcanic flows on the southern part (Gohl and Uenzelmann-Neben, 2001). Douglas (1977) says that most of the large total field magnetic anomaly is generated by the basement topography, and magnetic models that incorporate the basement relief suggest that the basement material is basalt. The magnetic lineation pattern in the Agulhas Basin suggests that a tectonic plate (Malvinas plate) existed during Campanian to Maestrichtian times (Labrecque and Hayes, 1979). The Agulhas anomaly is interpreted to represent the remnant scar of the process that led to fragmentation of Gondwana (e.g. Antoine and Moyes, 1992). The eastern Walvis ridge, close to the Namibian coast, is composed mostly of basalts and consists of thick oceanic crust of Icelandic type and sea-mounts buried under thick sediments, and its formation is thought to be the result of continental break-up (Fromm et al., 2017). Fullerton et al. (1989) say that the total field magnetic anomaly contrast between the Walvis Ridge and adjacent Angola and Cape Basins is too large to be explained by induced magnetization of even alkali basaltic crust. Despite that, most of the Walvis Ridge and portions and adjacent basins lie within the Cretaceous

Quiet Zone where prolonged normal polarity may have produced enough volume of material to cause a measurable magnetic anomaly contribution at satellite elevation (Fullerton et al., 1989).

The sources of the continental magnetic anomalies primarily comprise rock types formed early in the geological history of the Earth; most of the anomalies lie over the geological regions Precambrian in age (Hemant, 2003). The continental strong negative anomaly in the Z component is observed over the Zimbabwe craton and the southern Mozambique belt. The Zimbabwe craton comprises the Neoarchaean granites and greenstones (Jelsma and Dirks, 2002). The 30-40 km thick continental crust went through a major accretionary event between 2.68 and 2.60 Ga with associated late volcanism, regional deformation and the development of syntectonic sedimentary successions in foreland-type basins (Jelsma and Dirks, 2002). Ranganai et al. (2016) state that the Zimbabwe craton magnetic anomaly is closely associated with basement structures and bedrock lithology. The southern Mozambique belt is dominated by Rodinian-Grenvillian-age juvenile magmatism and crustal genesis, with a strong metamorphic overprint during the Pan-African (Grantham et al., 2003).

The continental strong positive anomalies in the Z component include the southern cape anomaly covering parts of Cape fold and Namaqua-Natal belts and Kaapvaal craton. The Cape Fold Belt (CFB) is formed in the late Palaeozoic age (e.g. Shone and Booth, 2005). The Palaeozoic Cape Supergroup dominates CFB outcrops and consists predominantly of quartzites, sandstones and shales (Blewett, 2016). Depositional ages are estimated at between 510 and 350 Ma, based on detrital zircons and paleontology (Miller et al., 2016). The Kaapvaal craton formed and stabilised between 3.7 and 2.6 Ga, and it is a mixture of early Archean (3.0-3.5 Ga) granite-greenstone terranes and older igneous rocks with average crustal thickness of 38 km (Nguuri et al., 2001). The paleointensity study of the Proterozoic-Archean volcanic rocks from Kaapvaal craton has indicated the presence of single-domain and pseudo-single-domain (SD-PSD) magnetite grains as the main magnetic mineral (Shcherbakova et al., 2014). In the top part of Namibia from the edge of the Kalahari basin, which extends eastwards into Botswana (Antoine, 1990), there is a strong positive anomaly in the Z component that covers parts of Damara belt and

Congo craton. The Damara belt in Namibia resulted from the Neoproterozoic convergence of three Palaeoprotozoic plates, the Rio de la Plata (currently located in Brazil and Uruguay), Congo and Kalahari cratons (e.g. Lehmann et al., 2015; Gray et al., 2008). This took place during convergence and amalgamation of a number of continental fragments to form the Gondwana Supercontinent (Meert, 2003). The major geological composition of the Pan-African Damara orogen are the Archaean-Proterozoic basements inliers, carbonates, turbidites and the foreland basin deposits (Gray et al., 2008). The southern part of the Congo craton, adjacent to Damara belt, is characterised by very thick (depth of 250 km) and resistive lithosphere, and the interpretation of the resistive upper crustal features suggests that they resulted from the emplacement of igneous intrusions during Pan-African magmatism ($\sim$500 Ma) (Khoza et al., 2013). It comprises the Angolan shield that consists of granite-gneisses and metasediments overlain by Paleoproterozoic cover (Begg et al., 2009).

The discussion of Z magnetic field anomalies with the help of large-scale tectonic features does not give clear explanations why some cratons or belts with the same crustal entities have different anomalies. The Zimbabwe craton shows a negative anomaly while the Kaapvaal craton has a positive anomaly. It is also strange to notice different crustal entities like Congo craton and Damara belt have the same magnetic field anomaly. A further study on the interpretation of the existence of a magnetic field anomaly signal in certain areas over southern Africa, using the underlying geological and tectonic features, is required to shed light on these observations.

7.7 Conclusion

The use of the regional model technique, SCHA, and CHAMP satellite vector data to develop a long-wavelength lithospheric field model over southern Africa was successful and confirmed the results of previous studies in the region. The use of only satellite vector data was proven to be sufficient to model successfully a long wavelength lithospheric field over southern Africa in X, Y and Z components at satellite altitude. However, the model can be improved by carefully selecting very quiet data to avoid any data contamination from external fields. The use of accurate models to remove the external field contributions from data can also substantially

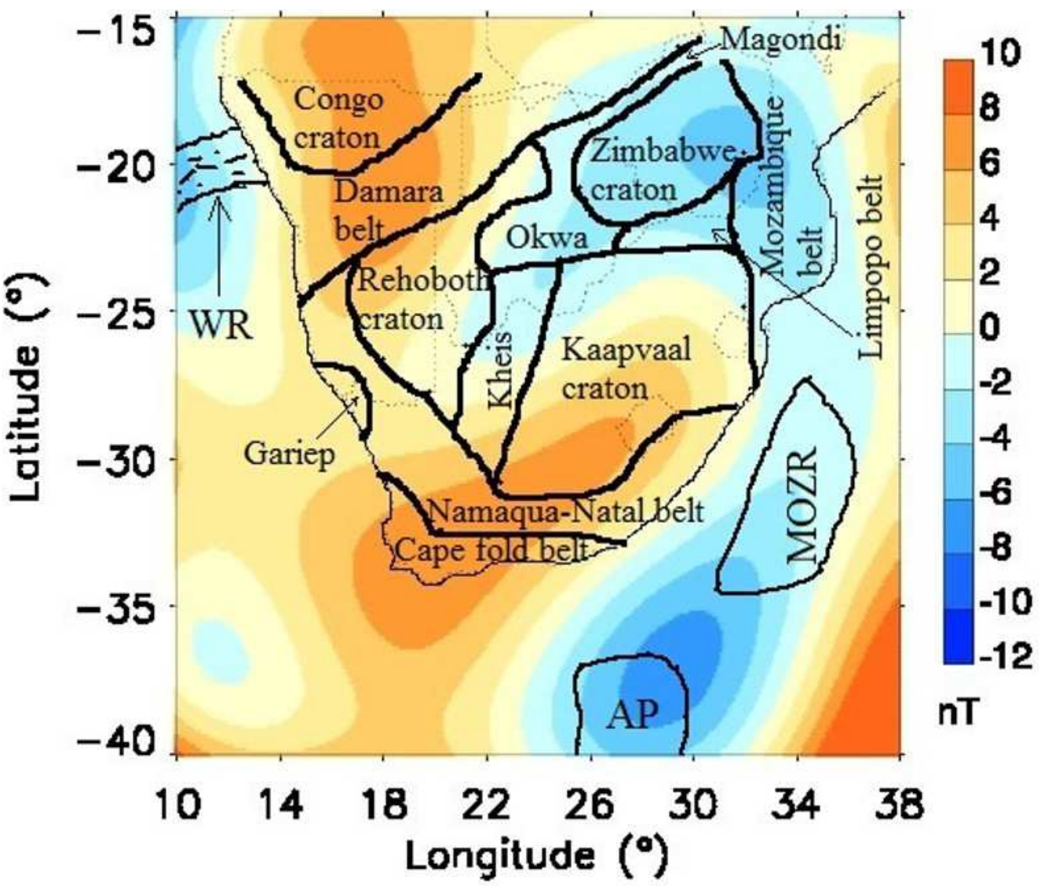

Figure 7.8: Presentation of the Z component map of the SCHA lithospheric magnetic field model over southern Africa together with the outline of large-scale tectonic features (Korte and Mandea, 2016). Some submerged topography features of ocean surrounding the southern Africa region are drawn on the map, the Walvis Ridge (WR) in the southern Atlantic ocean joins Africa at the northwest border of Namibia, the Agulhas Plateau (AP) located in the south western Indian ocean, south of South Africa, and the Mozambique Ridge (MOZR) in the southern Indian ocean, east of South Africa.

improve the model. The improved Swarm magnetic data can be used to produce a more accurate lithospheric field model. The prominent long-wavelength lithospheric field anomalies identified by previous studies were confirmed and discussed, and these include the Walvis Ridge and Agulhas oceanic field anomalies. The interpretation of magnetic anomalies with the help of large-scale tectonic features shows that, further investigation on the link between the crustal entities and the observed magnetic anomaly signal is required to clarify some shortcomings encountered in the discussion of the results.

7.8 Acknowledgements

The support of the CHAMP mission by the German Aerospace Center (DLR) and the Federal Ministry of Education and Research are gratefully acknowledged.

Chapter 8

Concluding Remarks

8.1 Conclusions

This study of geomagnetic main field changes over southern Africa, using CHAMP and ground-based data, has substantially improved our fundamental knowledge on the non-uniform characteristics decrease of the Earth's magnetic dipole, across this region in close proximity of the South Atlantic Anomaly. Results obtained indicated that the most rapid decrease of the core field is occurring across the South Atlantic region. The new developed regional field models SARM and SACFM-2 describe the geomagnetic temporal and spatial variations with detailed information, revealing highly non-linear time-varying characteristics of the geomagnetic field over southern Africa. The occurrence of the 2007 geomagnetic jerk and as well as the 2003 and 2004 rapid secular variation fluctuations in the region were confirmed. A steady decrease of the Earth's magnetic field over southern Africa, reaching 0.9% in total intensity in some areas (south-western region) between 2005.5 and 2010.5, has confirmed the deepening of the SAA in this region.

In this study, the use of the regional model technique, SCHA, and CHAMP satellite data to develop a long-wavelength lithospheric field model over southern Africa was successful. The prominent long-wavelength lithospheric field anomalies identified by previous studies, including the Agulhas anomaly, were confirmed and discussed.

8.2 Future studies

Regional field models are an excellent way to represent the small scale features of the time variation of geomagnetic field. Accurate regional field models, to study the steady decrease of the Earth's magnetic dipole and to carefully monitor the SAA evolution in the southern African region, can be developed using high quality geomagnetic field observations by European Space Agency Swarm satellite constellation. This Swarm mission provides denser data coverage than a single satellite mission, and has east-west gradients from two side-by-side satellites and another satellite at a higher altitude which help resolve some of the time varying fields not constrained by a single satellite mission. These models will increase our current fundamental knowledge about the temporal and spatial magnetic field variation over southern Africa and particularly South Atlantic Anomaly region, its geographic expansion and the rate of change of the geomagnetic field in this region.

This work is not comprehensive and there is always room for further improvements. One of these is to improve the lithospheric magnetic field model by performing carefully satellite data selection that reduces the contribution of external magnetic fields from magnetospheric and ionospheric currents. One method of doing this is to use different magnetic activity indices like Disturbance storm time (Dst) and planetary k-index (Kp), and considering very quiet geomagnetic conditions for satellite data selection. Ravat et al. (2003) state that the effectiveness of the comprehensive magnetic field model CM depends on the spatial and temporal matching of the main and external field variations that it tries to approximate, and this should be taken into account by data processors. It is encouraged to use the CM model to remove main and external fields from scalar and vector data. More detailed modelling could incorporate the latest comprehensive magnetic model CM5 (Sabaka et al., 2015) which should yield a more accurate lithospheric magnetic field model, as it is an improved model over CM4. The use of higher quality data with low noise by three satellites from Swarm mission, with the Revised Spherical Cap Harmonic Analysis (R-SCHA) improved model technique that incorporates observations from multiple altitudes, is likely to yield a better magnetic model with a better spatial resolution (Thébault et al., 2006), although it is unlikely to be comparable with airborne data for lithospheric studies (with wavelengths on the order of 80 km).

The interpretation of observed long wavelength magnetic anomaly signals developed from these spherical harmonic expansions of satellite data needs to account for the associated geological and tectonic features, and should correlate with known geological features. This interpretation can be improved by a better understanding of the geological processes which constitute the source of these magnetic anomalies. Ravat et al. (1992) presented an interpretation of the long-wavelength magnetic anomalies near the present continental margins of Africa and South America, and explain how the magnetic anomaly signal around the Walvis Ridge is not the same. The north eastern most portion of the Walvis Ridge is associated with total field positive anomaly and its south western portion is not. The described approach in their work, using the evolution of observed geological features, can be used as a guide to shed light on some unanswered questions regarding the interpretation of positive and negative magnetic anomaly signals observed over Southern Africa. The observed magnetic anomaly signal at satellite altitude results from different geological features (Ravat et al., 1993), and a proper interpretation looking at different sources of the observed anomaly is recommended.

References

Ahlberg, J. H., Nelson, E. N., and Walsh, J. L. (1967). *The theory of splines and their applications*, chapter 2, pages 9–10. New York: Academic Press.

Alldredge, L. R. (1981). Rectangular Harmonic Analysis Applied to the Geomagnetic Field. *Journal of Geophysical Research*, 86(B4):3021–3026. https://doi.org/10.1029/JB086iB04p03021.

Allen, J. H. (1982). Some common used magnetic activity indices: Their derivation, meaning and use. In *1982 NOAA Proceedings of the Workshop on Satellite Drag*. NOAA/EDIS/NGSDC, World Data Center-A for STP, D-63, 325 Broadway, Boulder, Colorado 80303, USA.

Antoine, L. A. (1990). *Processing and Interpretation of Near-earth Satellite Magnetic Data*. PhD thesis, University of the Witwatersrand, Johannesburg, South Africa.

Antoine, L. A. G. and Moyes, A. B. (1992). The Agulhas Magsat anomaly: Implications for continental break-up of Gondwana. *Tectonophysics*, 212:33–44. https://doi.org/10.1016/0040-1951(92)90138-V.

Badhwar, G. (1997). Drift rate of the South Atlantic Anomaly. *Journal of Geophysical Research*, 102(A2):2343–2349. https://doi.org/10.1029/96JA03494.

Bartels, J., Heck, N. H., and Johnson, H. F. (1939). The three-hour-range index measuring geomagnetic activity. *Journal of Geophysiscal Research*, 44:411–418. https://doi.org/10.1029/TE044i004p00411.

Begg, G. C., Griffin, W. L., Natapov, L. M., O'Reilly, S. Y., Grand, S. P., O'Neill, C., Hronsky, J. M. A., Djomani, Y. P., Swain, C., Deen, T., and

Bowden, P. (2009). The lithospheric architecture of Africa: Seismic tomography, mantle petrology, and tectonic evolution . *Geosphere*, 5:23–50. https://doi.org/10.1130/GES00178.1.

Blewett, S. C. J. and Phillips, D. (2016). *An Overview of Cape Fold Belt Geochronology: Implications for Sediment Provenance and the Timing of Orogenesis*, pages 45–55. ISBN: 978-3-319-40858-3. https://doi.org/10.1007/978-3-319-40859-0_5.

Bloxham, J., Zatman, S., and Dumberry, M. (2002). The origin of Geomagnetic Jerks. *Nature*, 420:65–68. https://doi.org/10.1038/nature01134.

Bryan, S. E. and Ernst, R. E. (2008). Revised definition of large igneous provinces (LIPs). *Earth-Science Reviews*, 86:175–202. https://doi.org/10.1016/j.earscirev.2007.08.008.

Cain, J. C., Davis, W. M., and Regan, R. D. (1974). A $n = 22$ model of geomagnetic field (abstract). *Eos, Transactions American Geophysical Union*, 56:1108.

Campbell, W. H. (2003). *The Earth's main filed. In Introduction to Geomagnetic Fields*, pages 1–66. Cambridge University Press. doi:10.1017/CB09781139165136.002.

Chulliat, A., Acken, P., and Maus, S. (2015). Fast equatorial waves propagating at the top of the Earth's core. *Geophysical Research Letters*, 42:3321–3329. https://doi.org/10.1002/2015GL064067.

Chulliat, A., Thébault, E., and Hulot, G. (2010). Core field acceleration pulse as a common cause of the 2003 and 2007 geomagnetic jerks. *Geophysical Research Letters*, 37(L07301). https://doi.org/10.1029/2009GL042019.

Cid, C., Palacios, J., Saiz, E., Guerrero, A., and Cerrato, Y. (2014). On extreme geomagnetic storms. *Journal of Space Weather and Space Climate*, 4:A28. https://doi.org/10.1051/swsc/2014026.

Constable, C. G. and Constable, S. C. (2004). *Satellite magnetic field measurements: Applications in studying the deep Earth*. Sparks RSJ and Hawkesworth CT (eds.) The state of the planet: Frontiers and challenges in Geophysics, Geophysical monograph 150, pp. 147-159. Washington, DC: American Geophysical Union.

Courtillot, V., Ducruix, J., and LeMouël, J. L. (1978). Sur une accélération récente de la variation séculaire du champ magnétique terrestre. *Comptes Rendus De l'Académie Des Sciences*, 287:1095–1098.

Cox, M. G. (1972). The numerical evaluation of B-splines. *Journal of the Institute of Mathematics and its Applications*, 10:134–149. https://doi.org/10.1093/imamat/10.2.134.

Davis, T. N. and Sugiura, M. (1966). Auroral electrojet activity index AE and its universal time variations. *Journal of Geophysical Research*, 71:785–801.

De Boor, C. (1972). On calculating with B-splines. *Approximation theory*, 6:50–62. https://doi.org/10.1016/0021-9045(72)90080-9.

De Santis, A., Qamili, E., Spada, G., and Gasperini, P. (2012). Geomagnetic South Atlantic Anomaly and global sea level rise: A direct connection? *Journal of Atmospheric and Solar-Terrestrial Physics*, 74:129–135. https://doi.org/10.1016/j.jastp.2011.10.015.

Douglas, M. B. (1977). Agulhas Plateau off southern Africa: A geophysical study. *GSA Bulletin*, 88(6):749–763.

Dunlop, D. and Özdemir, O. (2007). *Magnetisations in rocks and minerals, chap 8. In: Kono M (ed) Treatise on geophysics*, volume 5. Elsevier, Amsterdam.

Finlay, C. C., Aubert, J., and Gillet, N. (2016a). Gyre-driven decay of the Earth's magnetic dipole. *Nature Communications*, 7:10422. https://doi.org/10.1038/ncomms10422.

Finlay, C. C., Maus, S., Beggan, C. D., Bondar, T. N., Chambodut, A., Chernova, T. A., Chulliat, A., Golovkov, V. P., Hamilton, B., Hamoudi, M., Holme, R., Hulot, G., Kuang, W., Langlais, B., Lesur, V., Lowes, F. J., Lühr, H., Macmillan, S., Mandea, M., McLean, S., Manoj, C., Menvielle, M., Michaelis, I., Olsen, N., Rauberg, J., Rother, M., Sabaka, T. J., Tangborn, A., T∅ffner-Clausen, L., Thébault, E., Thomson, A. W. P., Wardinski, I., Wei, Z., and Zvereva, T. I. (2010). International Geomagnetic Reference Field. *Geophysical Journal International*, 183:1216–1230. https://doi.org/10.1111/j.1365-246X.2010.04804.x.

Finlay, C. C., Olsen, N., Kotsiaros, S., Gillet, N., and T∅ffner-Clausen, L. (2016b). Recent geomagnetic secular variation from Swarm and ground observatories as estimated in the CHAOS-6 geomagnetic field model. *Earth, Planets and Space*, 68:112. https://doi.org/10.1186/s40623-016-0486-1.

Fromm, T., Jokat, W., Ryberg, T., Behrmann, J. H., Haberland, C., and Weber, M. (2017). The onset of Walvis Ridge: Plume influence at the continental margin. *Tectonophysics*, 716:90–107. https://doi.org/10.1016/j.tecto.2017.03.011.

Fullerton, L. G., Frey, H. V., Roark, J. H., and Thomas, H. H. (1989). Evidence for remanent contribution in magsat data from the cretaceous quiet zone in the south atlantic. *Geophysical Research Letters*, 16(10):1085–1088. https://doi.org/10.1029/GL016i010p01085.

Gabriel, G., Vogel, D., Scheibe, R., Lindner, H., R., P., Wonik, T., and Krawczyk, C. (2011). Anomalies of the Earth's total magnetic field in Germany – the first complete homogenous data set reveals new opportunities for multiscale geoscientific studies. *Geophyiscal Journal International*, 184:1113–1118. https://doi.org/10.1111/j.1365-246X.2010.04924.x.

Garland, G. D. (1979). The contributions of Carl Friedrich Gauss to Geomagnetism. *Historia Mathematica*, 6:5–29. https://doi.org/10.1016/0315-0860(79)90100-9.

Gauss, C. F. (1839). Allgemeine Theorie des Erdmagnetismus. Resultate aus den Beobachtungen des Magnetischen Vereins im Jahre 1838. *(Göttinger Magnetischer Verein, Leipzig)*, pages 1–52.

Geese, A., Glabmeier, K., and Mandea, M. (2010). Earth's magnetic field: Observation and modelling from global to regional scales. Scientific technical report STR 11/03, German Centre of Geosciences, GFZ. https://doi.org/10.2312/GFZ.b103-11036.

Geese, A., Hayn, M., Mandea, M., and Lesur, V. (2009). Regional Modelling of the Southern African Geomagnetic Field using Harmonic Splines. *Geophysical Journal International*, 200:1–16. https://doi.org/10.1111/j.1365-246X.2010.04575.x.

Geese, A., Korte, M., Kotzé, P. B., and Lesur, V. (2011). Southern African geomagnetic secular variation from 2005 to 2009. *South African Journal of Geology*, 114:515–524. https://doi.org/10.2113/gssajg.114.3-4.515.

Girdler, R. W., Taylor, P. T., and Frawley, J. J. (1992). A possible impact origin for the Bangui magnetic anomaly (central Africa). *Tectonophysics*, 212:45–58. https://doi.org/10.1016/0040-1951(92)90139-W.

Gohl, K. and Uenzelmann-Neben, G. (2001). The crustal role of the Agulhas Plateau, southwest Indian Ocean: evidence from seismic profiling. *Geophysical Journal International*, 144:632–646. https://doi.org/10.1046/j.1365-246x.2001.01368.x.

Grantham, G. H., Maboko, M., and Eglington, B. M. (2003). A review of the evolution of the Mozambique Belt and implications for the amalgamation and dispersal of Rodinia and Gondwana. *Geological Society, London, Special Publications*, 206:401–425. https://doi.org/10.1144/GSL.SP.2003.206.01.19.

Gray, D. R., Foster, D. A., Meert, J. G., Goscombe, B. D., Armstrong, R., Trouw, R. A. J., and Passchier, C. W. (2008). A Damara orogen perspective on the assembly of southwestern Gondwana. *Geological Society, London, Special Publications*, 294:257–278. https://doi.org/10.1144/SP294.14.

Haines, G. V. (1985). Spherical Cap Harmonic Analysis. *Journal of Geophysical Research*, 90:2593–2598. https://doi.org/10.1029/JB090iB03p02583.

Haines, G. V. (1988). Computer programs for spherical cap harmonic analysis of potential and general fields. *Computers and Geosciences*, 14:413–447. https://doi.org/10.1016/0098-3004(88)90027-1.

Haines, G. V. (1990). Regional Magnetic Field Modelling: A Review. *Journal of Geomagnetism and Geoelectricity*, 42:1001–1018.

Heirtzler, J. R., Allen, J. H., and Wilkinson, D. C. (2002). Ever-present South Atlantic Anomaly damages spacecraft. *Eos, Transactions, American Geophysical Union*, 83(15):165–172. https://doi.org/10.1029/2002EO000105.

Hemant, K. (2003). *modelling and Interpretation of Global Lithospheric magnetic Anomalies*. PhD thesis, Free University of Berlin, Berlin, Germany.

Hemant, K. and Maus, S. (2005). Geological modelling of the CHAMP magnetic anomaly maps using a geographical information system technique. *Journal of Geophysical Research*, 110:B12103. https://doi.org/10.1029/2005JB003837.

Hollerbach, R. (1996). On the theory of the geodynamo. *Physics of the Earth and Planetary Interiors*, 98:163–183. https://doi.org/10.1016/S0031-9201(96)03185-8.

Holschneider, M., Chambodut, A., and Mandea, M. (2003). From global to regional analysis of the magnetic field on the sphere using wavelet frames. *Physics of the Earth and Planetary Interiors*, 135:107–124. https://doi.org/10.1016/S0031-9201(02)00210-8.

Jankowski, J. and Sucksdorff, C. (1996). *Guide for Magnetic Measurements and Observatory Practice*, page 235. International Association of Geomagnetism and Aeronomy, Warsaw.

Jelsma, H. A. and Dirks, P. H. G. M. (2002). Neoarchaean tectonic evolution of the Zimbabwe Craton. *Geological Society, London, Special Publications*, 199:183–211. https://doi.org/10.1144/GSL.SP.2002.199.01.10.

Karinen, A. and Mursula, K. (2005). A new reconstruction of the Dst index for 1932-2002. *Annales Geophysicae*, 23:475–485. https://doi.org/10.5194/angeo-23-475-2005.

Khoza, T. D., Jones, A. G., Muller, M. R., Evans, R. L., Miensopust, M. P., and Webb, S. J. (2013). Lithospheric structure of an Archean craton and adjacent mobile belt revealed from 2-D and 3-D inversion of magnetotelluric data: Example from southern Congo craton in northern Namibia. *Journal of Geophysical Research: Solid Earth*, 118:4378–4397. https://doi.org/10.1002/jgrb.50258.

Korte, M. and Mandea, M. (2016). Geopotential field anomalies and regional tectonic features - two case studies: southern Africa and Germany. *Solid Earth*, 7:751–768. https://doi.org/10.5194/se-7-751-2016.

Korte, M., Mandea, M., Kotzé, P. B., Nahayo, E., and Pretorius, B. (2007). Improved observations at the southern African geomagnetic repeat station network. *South African Journal of Geology*, 110:175–186. https://doi.org/10.2113/gssajg.110.2-3.175.

Kotzé, P. B. (2002). Modelling and analysis of Ørsted total field data over Southern Africa. *Geophysical Research Letters*, 29(15). https://doi.org/10.1029/2001GL013868.

Kotzé, P. B. (2011). Signature of 2007 geomagnetic jerk at the Hermanus magnetic observatory, South Africa. *South African Journal of Geology*, 114:207–210. https://doi.org/10.2113/gssajg.114.2.207.

Kotzé, P. B. and Korte, M. (2016). Morphology of the Southern African geomagnetic field derived from observatory and repeat station survey observations: 2005-2014. *Earth, Planets and Space*, 68:23. https://doi.org/10.1186/s40623-016-0403-7.

Labrecque, J. L. and Hayes, D. E. (1979). Seafloor spreading history of the Agulhas Basin. *Earth and Planetary Science Letters*, 45(2):411–428. https://doi.org/10.1016/0012-821X(79)90140-7.

Langel, R. A. and Estes, R. H. (1982). A geomagnetic field spectrum. *Geophysical Research Letters*, 9:250–253. https://doi.org/10.1029/GL009i004p00250.

Langel, R. A., Kerridge, D. J., Barraclough, D. R., and Malin, S. R. C. (1986). A Spline Representation. *Journal of Geomagnetism and Geoelectricity*, 38:573–597. https://doi.org/10.5636/jgg.38.573.

Langlais, B., Lesur, V., Purucker, M., Connerney, J., and Mandea, M. (2010). Crustal Magnetic Fields of Terrestrial Planets. *Space Science Reviews*, 152: 223. https://doi.org/10.1007/s11214-009-9557-y.

Lehmann, J., Saalmann, K., Naydenov, K. V. Milani, L., Belyanin, G. A., Zwingmann, H., Charlesworth, G., and Kinnaird, J. A. (2015). Structural and geochronological constraints on the PanAfrican tectonic evolution of the northern Damara Belt, Namibia. *Tectonics*, 35:103–135. https://doi.org/10.1002/2015TC003899.

Lesur, V. (2006). Introducing localised constraints in global geomagnetic field modelling. *Earth, Planets and Space*, 58:477–483. https://doi.org/https://doi.org/10.1186/BF03351943.

Lesur, V., Wardinski, I., and Hamoudi, M. (2011). Third version of the GFZ Reference Internal Magnetic Model: GRIMM-3: General Assembly European Geosciences Union (Vienna, 2011). *Geophysical Research Abstracts*, 13, EGU2011-6154.

Lesur, V., Wardinski, I., Hamoudi, M., and Rother, M. (2010). The second generation of the GFZ Reference Internal Magnetic Model: GRIMM-2. *Earth, Planets and Space*, 62:765–773. https://doi.org/10.5047/eps.2010.07.

Lesur, V., Wardinski, I., Rother, M., and Mandea, M. (2008). GRIMM: the GFZ Reference Internal Magnetic Model based on vector satellite and observatory data. *Geophysical Journal International*, 173:382–394. https://doi.org/10.1111/j.1365-246X.2008.03724.x.

Loper, D. E. and Lay, T. (1995). The core-mantle boundary region. *Journal of Geophysical Research*, 100(B4):6397–6420. https://doi.org/10.1029/94JB02048.

Lowes, F. J. (1974). Spatial power spectrum of the main geomagnetic field and extrapolation to the core. *Geophysical Journal International*, 36(3):717–730. https://doi.org/10.1111/j.1365-246X.1974.tb00622.x.

Lühr, H. (2001). Announcement of Opportunity for CHAMP. Technical report, GeoForschungsZentrum, Potsdam, Germany.

Macmillan, S. and Droujinina, A. (2007). Long-term trends in geomagnetic daily variation. *Earth, Planets and Space*, 59(3):791–795. https://doi.org/10.1186/BF03352699.

Mandea, M., Holme, R., Pais, A., Pinheiro, K., Jackson, A., and Verbanac, G. (2010). Geomagnetic Jerks: Rapid Core Field Variations and Core Dynamics. *Space Science Reviews*, 155:147–175. https://doi.org/10.1007/s11214-010-9663-x.

Mandea, M. and Olsen, N. (2009). Geomagnetic and Archeomagnetic jerks: Where Do We Stand? *Eos, Transactions American Geophysical Union*, 90(24):208–208. https://doi.org/10.1029/2009EO240004.

Mandea, M. and Purucker, M. (2005). Observing, modeling, and interpreting magnetic fields of the solid Earth. *Surveys in Geophysics*, 26. https://doi.org/10.1007/s10712-005-3857-x.

Maus, S. (2008). The geomagnetic power spectrum. *Geophysical Journal International*, 174:135–142. https://doi.org/10.1111/j.1365-246X.2008.03820.x.

Maus, S. (2010). Magnetic field model MF7. Available at: http://www.geomag.us/models/MF7.html (Accessed June 1, 2018).

Maus, S., Lühr, H., Rother, M., Hemant, K., Balasis, G., Ritter, P., and Stolle, C. (2007). Fifth-generation lithospheric magnetic field model from CHAMP measurements. *Geochemistry Geophysics Geosystems*, 8(5). https://doi.org/10.1029/2006GC001521.

Maus, S., Rother, M., Holme, R. T., Lühr, H., Olsen, N., and Haak, V. (2002). First scalar magnetic anomaly map from CHAMP satellite data indicates weak lithospheric field. *Geophysical Research Letters*, 29(14). https://doi.org/10.1029/2001GL013685.

Maus, S., Rother, M., Stolle, C., Mai, W., Choi, S., Lühr, H., Cooke, D., and Roth, C. (2006). Third generation of the Potsdam Magnetic Model of the Earth (POMME). *Geochemistry, Geophysics, Geosystems*, 7, Q07008. https://doi.org/10.1029/2006GC001269.

Mayaud, P. N. (1980). *Derivation, Meaning, and Use of Geomagnetic Indices*. Geophysical monograph 22. American Geophysical Union, Washington, D. C.

Meert, J. G. (2003). A synopsis of events related to the assembly of eastern Gondwana. *Tectonophysics*, 362:1–40. https://doi.org/10.1016/S0040-1951(02)00629-7.

Menvielle, M. and Berthelier, A. (1991). The K-derived planetary indices description and availability. *Reviews of Geophysics*, 29:415–432. https://doi.org/10.1029/91RG00994.

Miller, W., Armstrong, R., and and de Wit, M. (2016). Geology and U/Pb geochronology of the Gamtoos Complex and lower paleozoic Table Mountain Group, Cape Fold Belt, Eastern Cape, South Africa. *South African Journal of Geology*, 119(1):147–170. https://doi.org/10.2113/gssajg.119.1.147.

Moos, N. F. F. (1910). *Colaba magnetic data, 1846 to 1905, Part 2: The phenomenon and its discussion.* Government Central Press, Bombay, India.

Nahayo, E. and Kotzé, P. B. (2012). Polynomial modelling of CHAMP satellite data to investigate rapid secular variation fluctuations over southern Africa during 2003 and 2004. *Earth, Planets and Space*, 64:525–603. https://doi.org/10.5047/eps.2012.03.001.

Nahayo, E., Kotzé, P. B., Korte, M., and Webb, S. J. (2018). A harmonic spline magnetic main field model for Southern Africa combining ground and satellite data to describe the evolution of the South Atlantic Anomaly in this region between 2005 and 2010. *Earth, Planets and Space*, 70:30. https://doi.org/10.1186/s40623-018-0796-6.

Nahayo, E., Kotzé, P. B., and McCreadie, H. (2015). A southern Africa harmonic spline core field model derived from CHAMP satellite data. *Journal of Atmospheric and Solar-Terrestrial Physics*, 123:13–21. https://doi.org/10.1016/j.jastp.2014.12.002.

Nguuri, T. K., Gore, J., James, D. E., Webb, S. J., and Wright, C. (2001). Crustal structure beneath southern Africa and its implications for the formation and evolution of the Kaapvaal and Zimbabwe cratons. *Geophysical Research Letters*, 28(13):2501–2504. https://doi.org/10.1029/2000GL012587.

Olsen, N., Glassmeier, K. H., and Jia, X. (2010). Separation of the Magnetic Field into External and Internal Parts. *Space Science Reviews.*, 152:135–153.

Olsen, N., Hulot, G., Lesur, V., Finlay, C. C., Beggan, C., Chulliat, A., Sabaka, T. J., Floberghagen, R., Friis-Christensen, E., Haagmans, R., Lühr, H., T∅ffnerClausen, L., and Vigneron, P. (2014). The Swarm Initial Field Model for the 2014 geomagnetic field. *Geophysical Research Letters*, 42. https://doi.org/10.1002/2014GL062659.

Olsen, N. and Mandea, M. (2007). Investigation of a secular variation impulse using satellite data: The 2003 geomagnetic jerk. *Earth and Planetary Science Letters*, 255:94–105. https://doi.org/10.1016/j.epsl.2006.12.008.

Olsen, N. and Mandea, M. (2008). Rapidly changing flows in the Earth's core. *Nature Geoscience*, 1:390–394. https://doi.org/10.1038/ngeo203.

Olsen, N., Ravat, D., Finlay, C. C., and Kother, L. K. (2017). LCS-1: a high-resolution global model of the lithospheric magnetic field derived from CHAMP and Swarm satellite observations. *Geophysical Journal International*, 211(3):1461–1477. https://doi.org/10.1093/gji/ggx381.

Onovughe, E. (2016). Can the comprehensive model phase 4 (CM4) predict the geomagnetic diurnal field for days away from quiet time? *Annales Geophysicae*, 34:887–900. https://doi.org/10.5194/angeo-34-887-2016.

Parsiegla, N., Gohl, K., and Uenzelmann-Neben, G. (2008). The Agulhas Plateau: structure and evolution of a Large Igneous Province. *Geophysical Journal International*, 174:336–350. https://doi.org/10.1111/j.1365-246X.2008.03808.x.

Pavón-Carrasco, F. and DeSantis, A. (2015). The South Atlantic Anomaly: The Key for a Possible Geomagnetic Reversal. *Frontiers of Earth Science*, 4:40. https://doi.org/10.3389/feart.2016.00040.

Pulkkinen, A., Amm, O., Viljanen, A., and working group, B. (2003). Separation of the geomagnetic variation field on the ground into external and internal parts using the spherical elementary current system method. *Earth, Planets and Space*, 55:117–129. https://doi.org/10.1186/BF03351739.

Purucker, M. (2013). Magnetisation of the Earth's crust. Available at: https://denali.gsfc.nasa.gov/research/crustal_mag/prep/ (Accessed: August 7, 2018).

Ranganai, R. T., Whaler, K., and and Ebinger, C. J. (2016). Aeromagnetic Interpretation in the south-central Zimbabwe Craton: (Re-appraisal of) Crustal Structure and Tectonic Implications. *International Journal of Earth Sciences*, 105(8):2175–2201. https://doi.org/10.1007/s00531-015-1279-7.

Ravat, D., Hildenbrand, T. G., and Roest, W. (2003). New way of processing near-surface magnetic data: The utility of comprehensive magnetic field model. *The Leading Edge*, 22(8):784–785. https://doi.org/10.1190/1.1605082.

Ravat, D. N., Hinze, W. J., and Taylor, P. T. (1993). European tectonic features observed by Magsat. *Tectonophysics*, 220:157–173. https://doi.org/0040-1951(93)90229-D.

Ravat, D. N., Hinze, W. J., and von Frese, R. R. B. (1992). Analysis of Magsat magnetic contrasts across Africa and South America. In: R. R. B. von Frese and P. T. Taylor (Editors), Lithospheric Analysis of Magnetic and Related Geophysical Anomalies. *Tectonophysics*, 212:59–76. https://doi.org/10.1016/0040-1951(92)90140-2.

Regan, R. D. and Marsh, B. D. (1982). The Bangui magnetic anomaly: Its geological origin. *Journal of Geophysical Research*, 87:1107–1120. https://doi.org/10.1029/JB087iB02p01107.

Reigber, C., Lühr, H., and Schwintzer, P. (2002). CHAMP mission status. *Advances in Space Research*, 30(2):129–134. https://doi.org/10.1016/S0273-1177(02)00276-4.

Saba, M. M. F., Gonzalez, W. D., and Clúe de Gonzalez, A. L. (1997). Relationships between the AE, ap and Dst indices near solar minimum (1974) and solar maximum (1979). *Annales Geophysicae*, 15:1265–1270. https://doi.org/10.1007/s00585-997-1265-x.

Sabaka, T. J., Olsen, N., and Purucker, M. (2004). Extending comprehensive models of the Earth's magnetic field with Ørsted and CHAMP data. *Geophysical Journal International*, 159:521–547.

Sabaka, T. J., Olsen, N., Tyler, R. H., and Kuvshinov, A. (2015). CM5, a pre-Swarm comprehensive geomagnetic field model derived from over 12 yr of CHAMP, Ørsted, SAC-C and observatory data. *Geophysical Journal International*, 200:1596–1626. https://doi.org/10.1093/gji/ggu493.

Saur, J., Neubauer, F., and Glassmeier, K. (2010). Induced Magnetic Fields in Solar System Bodies. *Space Science Reviews*, 152:391–421. https://doi.org/10.1007/s11214-009-9581-y.

Shcherbakova, V. V., Shcherbakov, V. P., Zhidkov, G. V., and and Lubnina, N. V. (2014). Palaeointensity determinations on rocks from Palaeoproterozoic dykes from the Kaapvaal Craton (South Africa). *Geophysical Journal International*, 197(3):1371–1381. https://doi.org/10.1093/gji/ggu098.

Shone, R. W. and Booth, P. W. K. (2005). The Cape Basin, South Africa: A review. *Journal of African Earth Sciences*, 43 (1-3):196–210. https://doi.org/10.1016/j.jafrearsci.2005.07.013.

Shure, L., Parker, R. L., and Backus, G. E. (1982). Harmonic splines for geomagnetic modelling. *Physics of the Earth and Planetary Interiors*, 28:215–229.

Shure, L., Parker, R. L., and Langel, R. A. (1985). A preliminary Harmonic spline Model from Magsat Data. *Journal of Geophysical Research*, 90:11505–11512.

Sugiura, M. (1964). Hourly values of equatorial Dst for the IGY. *Annals of InternationalGeophysical Year*, 35:945–948.

Talarn, A., Pavón-Carrasco, F. J., Torta, J. M., and Catalán, M. (2017). Evaluation of using R-SCHA to simultaneously model main field and secular variation multilevel geomagnetic data for the North Atlantic. *Physics of the Earth and Planetary Interiors*, 263:55–68. https://doi.org/10.1016/j.pepi.2016.11.008.

Taylor, P. T. and Frawley, J. J. (1987). Magsat anomaly data over the Kursk region, U.S.S.R. *Physics of the Earth and Planetary Interiors*, 45(3):255–265. https://doi.org/10.1016/0031-9201(87)90014-8.

Thébault, E. (2003). *Modélisation régionale du champ magnétique terrestre*. PhD thesis, Université Louis Pasteur, Strasbourg I, France.

Thébault, E., Finlay, C. C., and Toh, H. (2015). Special issue "International Geomagnetic Reference Field—the twelfth generation". *Earth, Planets and Space*, 67:158. https://doi.org/10.1186/s40623-015-0313-0.

Thébault, E. and Gaya-Piqué, L. (2008). Applied comparisons between SCHA and R-SCHA regional modeling techniques. *Geochemistry, Geophysics, Geosystems*, 9. https://doi.org/10.1029/2008GC001953.

Thébault, E., Purucker, M., Whaler, K. A., Langlais, B., and Sabaka, T. J. (2010). The Magnetic Field of the Earth's Lithosphere. *Space Science Reviews*, 155:95–127. https://doi.org/10.1007/s11214-010-9667-6.

Thébault, E., Schott, J. J., and Mandea, M. (2006). Revised spherical cap harmonic analysis (R-SCHA): Validation and properties. *Journal of Geophysical Research*, 111(B01102). https://doi.org/10.1029/2005JB003836.

Thébault, E., Schott, J. J., Mandea, M., and Hoffbeck, J. P. (2004). A new proposal for spherical cap harmonic modelling. *Geophysical Journal International*, 159:83–103. https://doi.org/10.1111/j.1365-246X.2004.02361.x.

Torta, J. M. and De Santis, A. (1996). On the derivation of Earth's conductivity structure by means of spherical cap harmonic analysis. *Geophysical Journal International*, 127:441–451. https://doi.org/10.1111/j.1365-246X.1996.tb04732.x.

Valet, J. P. and Plenier, G. (2008). Simulations of time-varying non-dipole field during geomagnetic reversals and excursions. *Physics of the Earth and Planetary Interiors*, 169:178–193. https://doi.org/10.1016/j.pepi.2008.07.031.

Vervelidou, F., Thébault, E., and Korte, M. (2018). A high resolution lithospheric magnetic field model over southern Africa based on a joint inversion of CHAMP, Swarm, WDMAM and ground magnetic field data. *Solid Earth*, 9:897–910. https://doi.org/10.5194/se-9-897-2018.

Wardinski, I. and Holme, R. (2006). A time-dependent model of the Earth's magnetic field and its secular variation for the period 1980 to 2000. *Journal of Geophysical Research*, 111, B12101. https://doi.org/10.1029/2006JB004401.

Zhabin, A. V. and Sirotin, V. I. (2009). Orgin of Ferrugineous Quartzites of the Kursk Magnetic Anomaly. *Doklady Earth Sciences*, 427:737–739. https://doi.org/10.1134/S1028334X09050079.

Appendix A: Publications and conferences

A.1 Publications

Nahayo, E., Kotzé, P. B., Cilliers, P. J., and Lotz, S. (2019). Observations from SANSA's geomagnetic network during the Saint Patrick's Day storm of 17-18 March 2015. *South African Journal of Science*, 115 (1/2), https://doi.org/10.17159/sajs.2019/5204.

Nahayo, E., Kotzé, P. B., and Webb, S. J. (2019). Application of Spherical Cap Harmonic Analysis on CHAMP satellite data to develop a lithospheric magnetic field model over southern Africa at satellite altitude. *South African Journal of Geology*, 122.2:163-172, https://doi.org/10.25131/sajg.122.0012.

Nahayo, E., Kotzé, P. B., Korte, M., and Webb, S. J. (2018). A harmonic spline magnetic main field model for southern Africa combining ground and satellite data to describe the evolution of the South Atlantic Anomaly in this region between 2005 and 2010. *Earth Planets Space*, 70:30, https://doi.org/10.1186/s40623-018-0796-6.

Nahayo, E., Kotzé, P. B., and McCreadie, H. (2015). A southern Africa harmonic spline core field model derived from CHAMP satellite data. *Journal of Atmospheric and Solar-Terrestrial Physics*, 123:13-21, https://doi.org/10.1016/j.jastp.2014.12.002.

Nahayo, E., Kotzé, P. B., and Webb, S. J. (2015). An investigation in the use of satellite and ground-based data to develop a harmonic spline core magnetic field model over southern Africa (Extended Abstract). *Proceedings of the 14th South African Geophysical Association (SAGA) Biennial Conference and Exhibition, Drakensberg, South Africa, September 2015*, pp. 342-345.

Nahayo, E., Kotzé, P. B. and McCreadie, H. (2014). Study of the time variation of geomagnetic field over southern Africa applying harmonic splines technique on CHAMP satellite data. *Proceedings of South African Institute of Physics Conference (SAIP), Richards Bay, South Africa, July 2013*, pp. 422-427, ISBN: 978-0-620-62819-8.

A.2 Conferences

Nahayo, E., Kotzé, P. B., and Webb, S. J. (2017). Use of harmonic splines to investigate the evolution of the South Atlantic Anomaly over southern Africa between 2005 and 2010. *Poster presentation, Joint IAPSO-IAMAS-IAGA Assembly, Cape Town, South Africa, 27 August - 1 September 2017.*

Nahayo, E., Kotzé, P. B., and Cilliers, P. J. (2016). Overview of the South African National Space Agency geomagnetic observations network in space weather applications. *Oral presentation at XVIIth IAGA Workshop on Geomagnetic Observatory Instruments, Data Acquisition and Processing, Dourbes, Belgium, 4 - 10 September 2016.*

Nahayo, E., Kotzé, P. B., and Webb, S. J. (2015). An investigation in the use of satellite and ground-based data to develop a harmonic spline core magnetic field model over southern Africa. *Poster presentation, December 2015 Graduate Virtual Poster Showcase, AGU Showcase, Washington, DC.*

Nahayo, E., Kotzé, P. B., and Webb, S. J. (2015). A study into the geomagnetic field time variation over southern Africa applying harmonic splines to CHAMP satellite and ground observations. *Oral presentation, African Workshop on Space Science*

Research, SANSA Space Science, Hermanus, South Africa, 4 - 8 May 2015.

Appendix B: IDL scripts for data selection and processing

B.1 Read CHAMP data from Common Data Format (CDF) files

This subroutine was made available to me by Dr. Peter Sutcliffe (former SANSA researcher) and I have made some changes to suit my needs. It was used to get CHAMP data recorded in Southern Africa,reading CHAMP data files in CDF format. This task was required for chapters 4, 5, 6 and 7.

```
;**********************************************************************
PRO GET_CHAMP, DATE, SC, NB, TIME, GcLat, GcLon, GcRad, Qual, B1, B2,$
 B3, Alpha, Beta

;Procedure to read CHAMP [FGM(NEC)] vector magnetic field data from
;a CDF file
;Input:
;  DATE  : Date for which to read data [Integer Array: YYYY,MM,DD]
;  SC    : Data coordinate system ['FGM' or 'NEC']
;Output:
;  NB    : Number of data values
;  TIME  : Time (in UTC seconds for date) associated with data values
;  in CDF file
;  GcLat : Geocentric Latitude [degrees]
;  GcLon : Geocentric Longitude [degrees]
;  GcRad : Geocentric Radius [km]
;  Qual  : Data quality identifier
;   If SC=FGM then:
```

```
;  B1    : Component 1 of vector magetic field in FGM system
;  B2    : Component 2 of vector magetic field in FGM system
;  B3    : Component 3 of vector magetic field in FGM system
;   If SC=NEC then:
;  B1    : Northward component of vector magetic field in NEC system
;  B2    : Eastward component of vector magetic field in NEC system
;  B3    : Center (downward) component of vector magetic field in
;  NEC system
;  Alpha : Angle between B1 and resultant B12 of B1 and B2
;  (in the horizontal plane if NEC)
;  Beta  : Angle between B12 and resultant of B12 and B3
;  (in the vertical plane plane if NEC)
;  NOTE: The FGM or NEC field is used for computing Alpha and Beta
; depending on SC
;
; Author: Dr. Peter Sutcliffe
; Modified by: E. Nahayo
;*********************************************************************
;Generate filename
mm='mm'
dd='dd'
yyyymmdd='yyyymmdd'
yr=2010
path='C:\Research\PHD\DATA\2010\'
FOR DOY=1,365 DO BEGIN
    ;call a subroutine 'yymmdd.pro' to generate yyyymmdd for the file
    yymmdd,yr,DOY,mm,dd,yyyymmdd
    FILENAME='ch-me-2-fgm-nec_'+yyyymmdd+'.cdf'
    PATH_FILENAME =path+'\ch-me-2-fgm-nec_'+yyyymmdd+'.cdf'
    RESULT=FILE_TEST(PATH_FILENAME)
    IF RESULT NE 0 THEN BEGIN
        OPENW, lun, $
         path+'\CHAMPSCAN\2010\scanchamp'+yyyymmdd+'.dat',/get_lun
        ;Open CDF file
        cdfid = CDF_OPEN(PATH_FILENAME)
        CDF_CONTROL, cdfid, GET_VAR_INFO=var_info, $
         /ZVARIABLE, VARIABLE=5;6
        IF var_info.maxrec EQ -1 THEN BEGIN
            PRINT, 'No records written in CDF file: ',FILENAME
```

```
    STOP
ENDIF ELSE NB=var_info.maxrec + 1
;PADVAL=var_info.padvalue
PRINT,'************************************************************'
PRINT, 'CDF file:  ', FILENAME, '  contains  ', NB, '  data values'

;Get date and time information and generate time array
;(TIME[NB] is an array holding the times, expressed in
     ;seconds UTC for DATE, of data values)
CDF_VARGET, cdfid, 'EPOCH', EPOCH, rec_count=NB
TIME = FLTARR(NB)
TIME_GAP = 0
FOR IB=0L,NB-1 DO BEGIN
    CDF_EPOCH, EPOCH[IB], YEAR,MONTH,MDAY,HOUR,MINUTE,SEC,MILLI,$
    /BREAKDOWN_EPOCH
    TIME[IB] = HOUR*3600. + MINUTE*60. + SEC + MILLI/1000.
    IF IB EQ 0 THEN PRINT,'FGM data start: ',$
    YEAR,MONTH,MDAY,HOUR,MINUTE,SEC
    IF IB EQ NB-1 THEN PRINT,'FGM data end: ',$
     YEAR,MONTH,MDAY,HOUR,MINUTE,SEC
    IF IB NE 0 THEN BEGIN
      IF TIME[IB]-TIME[IB-1] GT 1.1 THEN BEGIN
         TIME_GAP = TIME_GAP + 1
         PRINT,'TIME GAP end: ',HOUR,MINUTE,SEC,'    Gap= ', $
   TIME[IB]-TIME[IB-1]
      ENDIF
    ENDIF
ENDFOR

;Read geocentric coordinate positions associated with magnetic
  ;field values
CDF_VARGET, cdfid, 'GEO_LAT', GcLat, rec_count=NB
CDF_VARGET, cdfid, 'GEO_LON', GcLon, rec_count=NB
CDF_VARGET, cdfid, 'GEO_ALT', GcAlt, rec_count=NB
CDF_ATTGET, cdfid, 'REFRADIUS', 3, Earth_Rad, /ZVARIABLE
GcRad = Earth_Rad + GcAlt

;Read data quality identifier
CDF_VARGET, cdfid, 'QUALITY', QUAL2, rec_count=NB
```

```
QUAL = BYTARR(NB)
FOR I = 0L, NB-1 DO QUAL[I]=QUAL2[1,I]

;Read magnetic field values
;BVEC = FLTARR(3,NB)
CDF_VARGET, cdfid, 'VEC', BVEC, rec_count=NB
  ;          Variable 5 (vector magnetic field)
B1 = FLTARR(NB)
FOR I = 0L, NB-1 DO B1[I]=BVEC[0,I]
B2 = FLTARR(NB)
FOR I = 0L, NB-1 DO B2[I]=BVEC[1,I]
B3 = FLTARR(NB)
FOR I = 0L, NB-1 DO B3[I]=BVEC[2,I]
BTOT = SQRT(B1^2 + B2^2 + B3^2)

;Compute angles Decl and Incl from field components
    ;Decl : Angle between B1 and resultant B12 of B1 and B2
        (in the horizontal plane if NEC)
    ;Incl  : Angle between B12 and resultant of B12 and B3
            (in the vertical plane plane if NEC)
    ;NOTE: The FGM or NEC field is used for computing Alpha
                and Beta depending on SC
     Decl =180/!pi * ATAN(B2,B1)
     B12 = SQRT(B1^2 + B2^2)
     Incl = 180/!pi * ATAN(B3,B12)
     PRINTF, lun,FORMAT='(A4,1A3,1A4,2A3,10A10)','year','mm','day',$
    'hr','mn','GdLat', 'GdLon', 'GdAlt(km)','X(nT)','Y(nT)',$
    'Z(nT)','F(nT)','H(nT)','D()','I()'
     for i=0L,NB-1 do begin
        IF (Gclat[i] LE -10) AND (GcLat[i] GE -40) THEN BEGIN
             IF (GcLon[i] LE 40) AND (GcLon[i] GE 10) THEN BEGIN
               hrt=time(i)/3600;hour
               hh=fix(hrt);hour
               mn=(hrt-hh)*60;min
               PRINTF, lun,FORMAT='(i4,3(1x,i2.2),1x,f6.3,3(1x,F8.3),$
         5(1x,F10.3),2(1x,F7.3))',yr,mm,dd,hh,mn,GcLat[i],GcLon[i],$
           GcAlt[i],B1[i],B2[i],B3[i],BTOT[i],B12(i),Decl(i),Incl(i)
             ENDIF
         ENDIF
```

```
        endfor
        free_lun,lun
        CDF_CLOSE, CDFID
    ENDIF
ENDFOR
END


;==============================================================================
pro yymmdd,yy,juday,mm,dd,yyyymmdd
;***************************************
;return date and part of the filename
;input: year and julian day
;   yy: year
;   juday: Julian day
;output:
; mm: month
; dd: day
;  yyyymmdd:date
;
;Author: E. Nahayo
***************************************
mth=[31,28,31,30,31,30,31,31,30,31,30,31]
;leap year case
;--------------
if yy mod 4 eq 0 then mth(1)=29
;julian day

if juday le 31 then begin
    mm=1
    dd=juday
endif else begin
   cnt=1
   dd=juday
   while dd gt mth(cnt-1) do begin
     dd=dd-mth(cnt-1)
     cnt=cnt+1
   endwhile
   mm=cnt
```

```
endelse
;get date yyyymmdd
;-----------------
p1='yyyy'
p2='mm'
p3='dd'
p1=string(yy,format='(i4)')
p2=string(mm,format='(i2.2)');month
p3=string(dd,format='(i2.2)');month
yyyymmdd=p1+p2+p3
return
end
```

B.2 Select night and quiet data values

I wrote the following IDL subroutines to select night and quiet data from CHAMP satellite data, the required task for chapters 4, 5, 6 and 7.

```
;***************************************************************************
pro nightvalues, inputfile, outputfile
;Program to read 1 day data c:...\data\champscan\scanchampyyyymmdd.dat
;and retrieve the records of night time values.
;These records are written in another file:
;c:...\data\night_quiet\nightvyyyymmdd.dat
;Input:
;  Year
;  inputfile: Scanned CHAMP satellite data for a selected region

;Output:
;  outputfile: File to hold only night values with the following elements:
;   GcLat : Geocentric Latitude [degrees]
;   GcLon : Geocentric Longitude [degrees]
;   GcAlt : Altitude [km]
;   X    : North component
;   Y    : East component
;   Z    : Vertical component
;   H    : Horizontal component
;   Decl : Declination
```

```
;   Incl : Inclination
;
;Author: E. Nahayo
;**************************************************************************
;Generate filename
mm='mm'
dd='dd'
yyyymmdd='yyyymmdd'
year=2005
yrs=string(year,format='(i4)')
FOR DOY=1,365 DO BEGIN
    yymmdd,yr,DOY,mm,dd,yyyymmdd
    inputfile='c:\research\data\champscan\'+yrs+'\scanchamp'+yyyymmdd+'.dat'
    outputfile='c:\research\data\night_quiet\'+yrs+'\nightv'+yyyymmdd+'.dat'

;check the existence of the file
;--------------------------------
    result=file_test(inputfile)
if result ne 0 then begin
        openr,lun,inputfile,/get_lun
        openw,lunr,outputfile,/get_lun
        firstline='firstline'

;read the header from input file and write it in the output file
;----------------------------------------------------------------
        readf,lun,firstline
        printf,lunr,firstline

;select night values
;-------------------
        while ~(eof(lun)) do begin
            readf,lun,format='(i4,3(1x,i2.2),1x,f6.3,3(1x,F8.3),$
                  5(1x,F10.3),2(1x,F7.3))', yr,mm,dd,hh,mn,GcLat,GcLon,GcAlt,$
                               X,Y,Z,F,H,Decl,Incl
            if (hh ge 0 and hh le 3)or (hh ge 17 and hh le 23) then begin
                   printf,lunr,format='(i4,3(1x,i2.2),1x,f6.3,3(1x,F8.3),$
                   5(1x,F10.3),2(1x,F7.3))', yr,mm,dd,hh,mn,GcLat,GcLon,GcAlt,$
                                   X,Y,Z,F,H,Decl,Incl
            endif
```

```
        endwhile
    endif else begin
      goto, jump
    endelse
    free_lun,lun
    free_lun,lunr
 jump:
ENDFOR
end

%-----------------------------------------------------------------------------
pro quiet_time_dst_values
;Program to select the dst indices in a certain range of values
;
;Author: E. Nahayo
;==============================================================
fname='c:\research\data\Dst\dst2001_2010.dat'
line=''
year=2005
yrs=string(year,format='(i4)')
outname='c:\research\data\Dst\Dst'+yrs+'_xtr.txt'
openw,lunw,outname,/get_lun

;print the header of the file
;----------------------------
printf,lunw,'year: '+yrs
printf,lunw,format='(a3,a3,12(a4))','Mth','Day','0','1','2','3',$
        '17','18','19','20','21','22','23'

openr,lun,fname,/get_lun

;variables to hold data
;dy0 to dy23 to hold Dst values for 24 hours
;-------------------------------------------
yr=0 & mm=0& dy=0 & dy0=0 & dy1=0 & dy2=0
dy3=0 & dy4=0 & dy17=0 & dy18=0 & dy19=0 & dy20=0
dy21=0 & dy22=0 & dy23=0
line=''
cnt=0
```

```
while ~(eof(lun)) do begin

   ;read values into variables
  ;---------------------------
readf,lun,format='(a120)',line

yr=fix(strmid(line,3,2))
mm=fix(strmid(line,5,2))
dy=fix(strmid(line,8,2))
dy0=fix(strmid(line,20,4))
dy1=fix(strmid(line,24,4))
dy2=fix(strmid(line,28,4))
dy3=fix(strmid(line,32,4))
dy17=fix(strmid(line,88,4))
dy18=fix(strmid(line,92,4))
dy19=fix(strmid(line,96,4))
dy20=fix(strmid(line,100,4))
dy21=fix(strmid(line,104,4))
dy22=fix(strmid(line,108,4))
dy23=fix(strmid(line,112,4))

       yrflg=2000+yr

;hours with Dst values outside the preferred range (Dst between -20 and 0 nT)
;are signed a value of 100 (a flag to recognise them for future computation).
;----------------------------------------------------------------------------

       if yrflg eq year then begin
if dy0 lt -20 or dy0 gt 0 then dy0=100
if dy1 lt -20 or dy1 gt 0 then dy1=100
if dy2 lt -20 or dy2 gt 0 then dy2=100
if dy3 lt -20 or dy3 gt 0 then dy3=100
if dy17 lt -20 or dy17 gt 0 then dy17=100
if dy18 lt -20 or dy18 gt 0 then dy18=100
      if dy19 lt -20 or dy19 gt 0 then dy19=100
      if dy20 lt -20 or dy20 gt 0 then dy20=100
      if dy21 lt -20 or dy21 gt 0 then dy21=100
      if dy22 lt -20 or dy22 gt 0 then dy22=100
      if dy23 lt -20 or dy23 gt 0 then dy23=100
```

```
      printf,lunw,format='(i2,2x,i2,11(i4))',mm,dy,dy0,dy1,dy2,$
     dy3,dy17,dy18,dy19,dy20,dy21,dy22,dy23
      cnt++
      endif
 endwhile
 free_lun,lun
 free_lun,lunw
 end

%------------------------------------------------------------------
;*****************************************************************
pro quiet_night_data, dstfile,outfile
;program to select quiet night data using dst indices
;observatory hourly data/satellite data selection
;input:
; dstfile: File of Dst indices
;output:
; outfile: File of quiet night data
;
;Author: E. Nahayo
;*****************************************************************
obs='her'
for i=2005, 2010 do begin
  years=string(i,format='(i4)')

;file names for Dst indices and output of quiet night data
;---------------------------------------------------------
  dstfile='c:\research\data\dst\dst'+years+'_xtr.txt'
  outfile='c:\research\data\qn'+obs+years+'_night.dat'

;array to contain selected hours
;-------------------------------
  hr_arr=[1,2,3,4,17,18,19,20,21,22,23,24]

  openr,lund,dstfile,/get_lun
  openw,lunw,outfile,/get_lun
  skip_lun,lund,2,/lines

;read dst values into str_arr
```

```
;----------------------------
  while ~eof(lund) do begin
    str_arr=strarr(12)
h1='' & h2=h1 & h3=h1 &h4=h1 & h17=h1 & h18=h1 & h19=h1
h20=h1 &h21=h1 &h22=h1 &h23=h1 & h24=h1
readf,lund,format='(i3,i3,12a4)',mth,day,h1,h2,h3,h4,h17,$
h18,h19,h20,h21,h22,h23,h24
      str_arr(0)=h1
      str_arr(1)=h2
      str_arr(2)=h3
      str_arr(3)=h4
      str_arr(4)=h17
      str_arr(5)=h18
      str_arr(6)=h19
      str_arr(7)=h20
      str_arr(8)=h21
      str_arr(9)=h22
      str_arr(10)=h23
      str_arr(11)=h24

;read observatory hourly data values and select quiet night values
;-----------------------------------------------------------------
      openr,lun2,'c:\research\data\n'+obs+years+'_night.dat',/get_lun
      while ~eof(lun2) do begin
         readf,lun2,format='(i5,3(i3),f8.1,f8.1,f9.1)',year,mth1,day1,hr,x,y,z
         if mth eq mth1 and day eq day1 then begin
           for k=0,11 do begin
                 dstv1=str_arr(k)
                 dstvs=STRCOMPRESS(dstv1,/REMOVE_ALL)
                 hrk=hr_arr(k)

;save/print quiet night data in the output file
;-----------------------------------------------
               if dstvs ne '100' then begin
                      if hr eq hrk then begin
                         printf,lunw,format='(i5,3(i3),f8.1,f8.1,f9.1)',$
                         year,mth1,day1,hr,x,y,z
                       endif
                 endif
```

```
              endfor
          endif
       endwhile
     free_lun,lun2
   endwhile
   free_lun,lund,lunw
  endfor
end
```

B.3 Preparation of harmonic spline model input data

I wrote the following IDL subroutines to prepare the input data for harmonic spline model, the required task for chapters 4, 5 and 6.

```
;************************************************************************
pro splines_input
;program to read 1 day data c:\research\data\night_quiet\.dat
;and create input data files to be processed to get harmonic spline model
;input data.
;
;Author: E. Nahayo
;************************************************************************
;Generate filename
mm='mm'
dd='dd'
yyyymmdd='yyyymmdd'
mth=[31,28,31,30,31,30,31,31,30,31,30,31]
for yy=2001,2010 do begin
ydays=365.0
  if yy mod 4 eq 0 then begin
 mth(1)=29
 ydays=366.0
endif
years=string(yy,format='(i4)')
outfile='c:\research\data\spline_input_data\'+years+'_20-4UT.txt'
openw,lunw,outfile,/get_lun
```

```
FOR DOY=1,ydays DO BEGIN
  yymmdd,yy,DOY,mm,dd,yyyymmdd
  inputfile='c:\research\night_quiet_data\'+years+$
   '\night_quiet_c'+yyyymmdd+'.txt'
  result=file_test(inputfile);check the existence of the file
  if result ne 0 then begin
      openr,lun,inputfile,/get_lun
      firstline='firstline'
      readf,lun,firstline
      while ~(eof(lun)) do begin
          readf,lun,FORMAT='(1x,i4,2(1x,i2.2),2x,i2,3(f9.3),$
        f8.3,5f11.3,2(f8.3))',yr,mm,dd,hh,mn,GdLat,GdLon,$
               GdAlt,x,y,z,f,h,d,Incl

           ;restricting data to 20-04 UT time interval
           ;----------------------------------------
          if hh ne 17 and hh ne 18 and hh ne 19 and hh ne 5 $
          and hh ne 6 then begin
        ;convert latitude in colatitude
           clt=90-gdlat
           lon=gdlon
            ;convert geodetic altitude in geocentric altitude
           alt=6371+gdalt

            ;calculate epoch
            ;---------------
           ndays=0.0
            for i=0,mm-1 do begin
            ndays= ndays+mth(i)
           endfor
           ndays=ndays+dd-mth(mm-1)
           epoch= yr+ndays/ydays

    ;print data
    ;-----------
         printf,lunw,format='(f10.4,f10.4, f10.4,f10.3,f10.2,$
          f10.2,f10.2,i7,a4,i4)',clt,lon,alt,epoch,x,y,z,mm,'PNT','5'
       endif
      endwhile
```

```
        free_lun,lun
    endif
ENDFOR
free_lun,lunw
endfor
end

**********************************************************************
pro data_bins
;This program creates the data bins of circles of 0.1 deg radius
;on a 1 deg x 1 deg of Latitude and Longitude grid.
;
;Author: E. Nahayo
;*********************************************************************
for yy=2005,2010 do begin
   years=string(yy,format='(i4)')
 outfile='c:\research\data\data_bins_data\'+years+'bins01_1.txt'
 inputfile='c:\research\data\spline_input_data\'+years+'.txt'

;statistical analysis file
 ;-------------------------
 statfile='c:\research\data\data_bins_data\'+years+'stat01_1.txt'
  openw,lunw,outfile,/get_lun
  openw,lunw1,statfile,/get_lun
  cnt=1
  stat_tot=0
  for cl=100,128,1 do begin
   for lo =10,38,1 do begin
        statcnt=0
     openr,lun,inputfile,/get_lun
      pnt='ass'
          while ~(eof(lun)) do begin
             readf,lun,format='(f10.4,f10.4, f10.4,f10.3,$
                 f10.2,f10.2,f10.2,i7,a4,i4)',$
                             clt,lon,alt,epoch,x,y,z,mm,pnt,nb
            if (abs (cl-clt) le 0.1)and (abs(lo-lon)le 0.1) then begin
                  cnts='c'+strcompress(string(cnt,format='(i)'),/remove_all)
                    printf,lunw,format='(f10.4,f10.4, f10.4,f10.3,$
                    f10.2,f10.2,f10.2,i7,a4,a5)',$
```

```
                        clt,lon,alt,epoch,x,y,z,cnt,cnts,'5'
                       statcnt++
             endif
          endwhile
          cnts='c'+strcompress(string(cnt,format='(i)'),/remove_all)
          printf,lunw1,format='(a5,i3)',cnts,statcnt
          stat_tot=stat_tot+statcnt
          free_lun,lun
          cnt++
        endfor
      endfor
      free_lun,lunw
      printf,lunw1,format='(i5)',stat_tot
      free_lun,lunw1
endfor
end
%---------------------------------------------------------------------------
;***************************************************************************
pro group1epoch
;This program reads the satellite data grouped in bins and average data of
;the same epoch (normally data of the same epoch recorded in the
;same bin have very close altitude values).
;The value standing for the bin at a particular epoch is obtained.
;
;Author: E. Nahayo
;***************************************************************************
year=2010.0
yrs=string(year,format='(i4)')
outfile='c:\research\data\same_epoch_data\'+yrs+'_1epoch01_1.txt'
statfile='c:\research\data\same_epoch_data\'+yrs+'_stat01_1.txt'
inputfile='c:\research\data\data_bins_data\'+yrs+'cent01_1.txt'
openw,lunw,outfile,/get_lun
openw,lunw1,statfile,/get_lun
openr,lun,inputfile,/get_lun
pnt='ass'
cnt=0
epch=epch+.252;starting epoch

;initialise required variables to receive data
```

```
;---------------------------------------------
sumx=0.0 & sumy=0.0 & sumz=0.0 & sumlat=0.0 &  sumlon=0.0
sumalt=0.0 & centercnt=1L
temp_arrx=[0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0]
temp_arry=temp_arrx & temp_arrz=temp_arrx & temp_arra=temp_arrx
avx=0.0 & avy=0.0 & avz=0.0 & ava=0.0 & avcnt=0
while ~(eof(lun)) do begin
  readf,lun,format='(f10.4,f10.4, f10.4,f10.3,f10.2,f10.2,f10.2,i7,a4,i5)',$
  clt,lon,alt,epoch,x,y,z,mm,pnt,nb
   tclt=clt
   tlon=lon
   talt=alt
   tx=x
   ty=y
   tz=z
   if epch eq epoch then begin
       sumx=sumx+x
       sumy=sumy+y
       sumz=sumz+z
       sumlat=sumlat+clt
       sumlon=sumlon+lon
       sumalt=sumalt+alt
       temp_arrx(cnt)=x
       temp_arry(cnt)=y
       temp_arrz(cnt)=z
       temp_arra(cnt)=alt
       cnt=cnt+1
   endif else begin
          clt=sumlat/cnt
          lon=sumlon/cnt
          alt=sumalt/cnt
          x=sumx/cnt
          y=sumy/cnt
          z=sumz/cnt
        cnts='c'+strcompress(string(centercnt,format='(i)'),/remove_all)
         printf,lunw,format='(f10.4,f10.4, f10.4,f10.3,f10.2,f10.2,$
             f10.2,i6,a7,a2)',clt,lon,alt,epch,x,y,z,centercnt,cnts,'5'

         ;check if there are some elements inside the arrays
```

```
;---------------------------------------------------
nelts=where(temp_arrx ne 0.0)
if nelts(0) ne -1 and n_elements(nelts)ge 2 then begin
  temp_arrx=temp_arrx(nelts)
  nelts=where(temp_arry ne 0.0)
  temp_arry=temp_arry(nelts)
  nelts=where(temp_arrz ne 0.0)
  temp_arrz=temp_arrz(nelts)
  nelts=where(temp_arra ne 0.0)
  temp_arra=temp_arra(nelts)
  printf,lunw1,clt,lon,alt,epch,stddev(temp_arrx),stddev(temp_arry),$
         stddev(temp_arrz),stddev(temp_arra),$
               format='(7f8.2,f8.3)'
    avx=avx+stddev(temp_arrx)
    avy=avy+stddev(temp_arry)
    avz=avz+stddev(temp_arrz)
    ava=ava+stddev(temp_arra)
    avcnt++
 endif

;initialise variables for the next round
          ;---------------------------------------
  temp_arrx=[0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0]
 temp_arry=temp_arrx & temp_arrz=temp_arrx & temp_arra=temp_arrx
sumx=0.0 & sumy=0.0 & sumz=0.0 & sumlat=0.0 & sumlon=0.0
sumalt=0.0 & cnt=0
  centercnt=centercnt+1
epch=epoch
sumx=sumx+tx
 sumy=sumy+ty
 sumz=sumz+tz
 sumlat=sumlat+tclt
 sumlon=sumlon+tlon
 sumalt=sumalt+talt
 cnt=cnt+1
  endelse
endwhile
printf,lunw1,avx/avcnt,avy/avcnt,avz/avcnt,ava/avcnt,format='(3f8.2,f8.3)'
free_lun,lun
```

```
free_lun,lunw
free_lun,lunw1
end
%---------------------------------------------------------------------------
;****************************************************************************
pro group1center
;Program to read averaged data of the same epoch and group them according
;to their geographic coordinates (center of their respective bins)
;This is the final step to get harmonic spline model input data.
;
;Author: E. Nahayo
;****************************************************************************
outfile='c:\research\data\same_center_data\model_input_data_2001_2010_01_1.txt'
openw,lunw,outfile,/get_lun
pnt='sss'
altsum=0.0
altcnt=0
centercnt=1L
flag=0
for cl=100,128,1 do begin
   for lo=10,38,1 do begin
     inputfile='c:\research\data\same_epoch_data\2001_2010epoch01_1.txt'
     openr,lun,inputfile,/get_lun
     while ~(eof(lun)) do begin
       readf,lun,format='(f10.4,f10.4, f10.4,f10.3,f10.2,f10.2,f10.2,i6,a7,a2)',$
       clt,lon,alt,epoch,x,y,z,mm,pnt,nb
            if (abs (cl-clt) le 0.1)and (abs(lo-lon)le 0.1) then begin
              cnts='c'+strcompress(string(centercnt,format='(i4.4)'),/remove_all)
               printf,lunw,format='(f10.4,f10.4, f10.4,f10.3,f10.2,f10.2,$
                   f10.2,i5,a6,a4)',cl,lo,alt,epoch,x,y,z,centercnt,cnts,'25'
               flag=1
               altsum=altsum+alt
               altcnt=altcnt+1
            endif
     endwhile
     if flag eq 1 then begin
       alt=altsum/altcnt
       altsum=0.0
       altcnt=0.0
```

```
        centercnt=centercnt+1
        flag=0
    endif
   free_lun,lun
  endfor
endfor
free_lun,lunw
end
```

Appendix C: FORTRAN programs for the development of a harmonic spline model

The program mkfield was originally written by Dr. Vincent Lesur (former researcher at GFZ German Research Centre for Geosciences). It was used by Dr. Anne Hemshorn before being made available to me. It was used for chapters 4, 5 and 6.

```
c*******************************************************************************
c Program mkfield
c Vincent Lesur 06/05/2008
c     modified many times
c            Anne Hemshorn 30/10/2008
c     modified by E Nahayo 10/04/2014
c         itmax > 0       find parameters to fit the data
c         itmax = 0       do the forward problem only
c         itmax < 0       compute eigen values and vectors of normal equations
c       input:
c               path          where to write output files
c               itmax         Maximum number of itterations
c               npmax         number max of data point with correlated errors
c               npm           Maximum total number of data points
c               nd            space dimension
c               npt           Total Number of data
c               ppos          data point position in ndD
c               nb            Number or base function to use
c               BC            First estimation of Base function coefficients
c               dl(3)         damping factor & max condition number acceptable
c               FM            misfit function (like l2_norm.f)
```

```
c               BS            Base subroutine to use
c               fstd          std Function
c               fdamp         damping Subroutine
c               nt(*)         data type
c               cov(*)        covariance matrix in SLAP Column format
c               icov/jcov     Integer vector describing cov format
c               stdt          STD Target
c               BC(*)         Base function coefficients
c               xyzf(*)       results of forward modelling
c               stdt          estimated STD
c               GG            eigen vectors of the last normal matrix
c               BB            eigen values of the last normal matrix
c
c******************************************************************************
        use hsplines_sa
        implicit none
        integer nbm,npm,ncm,nd,npmax
        parameter (nbm=10000000,npm=100000,ncm=3*npm,nd=6,npmax=3)
        integer nt(npm),ijcov(ncm,2)
        integer nb,i,j,k,npt,nc,itmax,idrv
        real*8  bc(nbm),ddat(nd+1,npm),cov(ncm),xyzf(npm)
        real*8  stdt,std,dl(3),var(nbm),radius
        real*8, allocatable :: gg(:),bb(:) !EM NAH
        integer nr,inr,rnr,crust
        integer st,ss,lw,step
        real*8 dy,dltgd,dln,alt,xx,sx,yy,sy,zz,sz,djd,dltgc,rad,cd,sd
        real*8 xgc,ygc,zgc,dd,ds,dt,norm
        character nom2*100,nom3*100,cc*5
        character*5 station(2000)
        real*8 l2_norm,l2_std
        real*8 l1_norm,l1_std
        external l2_norm,l2_std
        external l1_norm,l1_std
        external damp_rien
        allocate(gg(1:nbm*200),bb(1:nbm))
        lw=12
        open(10,file='DAT/reps_obs_11.dat',iostat=i)
        open(12,file="Results/stations_11.out")
c===============================================================!EM NAH
```

```
c a new file of stations with the same altitude
open(122,file="Results/stations_12.out")
c================================================================
        npt=0
        rnr=0
!EM NAH: read input data from repos_obs_11.dat and write
!stn name, col, lon, rad and record number into stations_11.out file
        do while (npt+3.le.npm)
            read(10,*,iostat=i)dltgc,dln,rad,dy,xgc,ygc,zgc,nr,cc
            if(i.ne.0) EXIT
            if(dy.ge.2005.0d0.and.dy.le.2011.d0) then
            if(nr.ne.inr) then
               write(*,*)nr,cc,dltgc,dln
               rnr=rnr+1
               cntrg(1,rnr)=dltgc
               cntrg(2,rnr)=dln
               cntrg(3,rnr)=rad
               station(rnr)=cc
                write(12,'(a6,1x,2f10.4,f9.3,i5)')station(rnr),
     >            dltgc,dln,rad,rnr
c====================================================================!EM NAH
c make a fixed altitude !EM NAH: write into stations_12.out with a fixed radius.
radius=6721.2
       write(122,'(a6,1x,2f10.4,f9.3,i5)')station(rnr),
     >            dltgc,dln,radius,rnr
c====================================================================
            end if
!EM NAH: tranform a date given in decimal year in Julian days from 00:00
            call dy2mjd(dy,djd)
            if(nr.eq.1)
     >          write(*,'(7f10.2,i5)')dltgc,dln,rad,dy,xgc,ygc,zgc,rnr
!EM NAH: if long is negative add 360
            if(dln.lt.0.0d0)dln=360.d0+dln
              crust=0
c !EM NAH: Build the data array ddat(number of dimensions+1,
c        max number of data points)
c    building ddat
            npt=npt+1
            ddat(1,npt)=dltgc
```

```
            ddat(2,npt)=dln
            ddat(3,npt)=rad
            ddat(4,npt)=djd
            ddat(5,npt)=dble(rnr)
            ddat(nd+1,npt)=xgc
            nt(npt)=1+crust
c
            npt=npt+1
            ddat(1,npt)=dltgc
            ddat(2,npt)=dln
            ddat(3,npt)=rad
            ddat(4,npt)=djd
            ddat(5,npt)=dble(rnr)
            ddat(nd+1,npt)=ygc
            nt(npt)=2+crust
c
            npt=npt+1
            ddat(1,npt)=dltgc
            ddat(2,npt)=dln
            ddat(3,npt)=rad
            ddat(4,npt)=djd
            ddat(5,npt)=dble(rnr)
            ddat(nd+1,npt)=zgc
            nt(npt)=3+crust !nt(npm - max number of data points) is data type
            inr=nr
            end if
        enddo
        nbcg=rnr
        close(10)
        close(12)
c Dr Ingo comments
c=================================================================================
c     idrv determines the degree of temporal derivative of the temporal constraint
c     idrv = 2 for cubic B-splines
c     idrv = 3 for order 6 B-Splines
c     currently idrv = 0
c     BUT THIS WILL MINIMISE THE ZEROTH TIME DERIVATIVE WHICH IS NOT MEANINGFUL
c==================================================================================
        idrv=0
```

```
!EM NAH: Initialise variables
!nb :number of coefficients
!bc: First estimation of Base function coefficients
!lw: maximum degree of expansion of spherical harmonic functions
!idrv=0: degree of temporal derivative of the temporal constraint
        call init_hsplines_sa(nb,bc,0,lw)
c       call init_hsplines_sa(nb,bc,idrv,lw)
        write(*,*)"nb  ",nb
c  Defining covariance
        nc=npt !EM NAH: npt is the total number of data points
        call dy2mjd(2007.0d0,djd)
        do i=1,nc
          ijcov(i,1)=i
          ijcov(i,2)=i
          if (ddat(5,i).eq.  4.0d0 .or.
     >        ddat(5,i).eq. 14.0d0 .or.
     >        ddat(5,i).eq. 16.0d0 .or.
     >        ddat(5,i).eq. 32.0d0 ) then
            cov(i)=1.d0
          else
            if(ddat(4,i).lt.djd) then
              cov(i)=0.2d0
            else
             cov(i)=0.5d0
            endif
          endif
        enddo
        call DS2Y(npt,nc,ijcov(1,1),ijcov(1,2),cov,0)
c       Calling main sub_mm_3.outine for L2 inversion
        itmax=1
        stdt=0.0d0
        ds=1.d0
        dt=1.d0
        do i=1,ss
          ds=ds*0.1d0
        end do
        do i=1,st
         dt=dt*0.1d0
        end do
```

```
        dl(1)=0.0d0!damp in time, dt
        dl(2)=0.0d0!damp in space, ds
        dl(3)=1.0d12
        write(*,*)'starting opt_d_l3'
        write(*,*)'The reference year will be: ',ryg
        write(*,*)'number of data point :',npt
        write(*,*)'number of parameters :',nb
        write(*,*)'number of centres :',nbcg
        write(*,*)'maximum degree :',lwg
        write(*,*)'damping st & ss :',dl(1),dl(2)
        nom2='./Results/' !EM NAH a directory to receive output data files
c    call to opt_d_l3()
c    ==================
c       Computes parameters BC from a set of NPT data points.
c       The "base function" subroutine used is BS
c       The normal equations are built by adding the contributions of groups
c       of uncorrelated data points. The Data covarience matrix, input in a
c       SLAP column format, is used to determine groups of uncorrelated data
c       The inversion process may be non-linear => First estimation of the
c       solution must be given.
        call opt_d_l3(nom2,itmax,npmax,npm,nd,npt,ddat,nb,bc,dl,l2_norm
     >          ,sub_hsplines_sa,l2_std,smooth_st
     >          ,nt,cov,ijcov(1,1),ijcov(1,2),stdt,xyzf
     >          ,BB,GG)
        write(*,'(A)')' '
        write(*,'(A,e15.7)')'The L2 STD is: ',stdt
        write(*,'(A)')' '
        std=stdt*npt/(npt-nb)
        var(1:nb)=std
        call formalstd(nb,dl(3),BB,GG,std,var)
      write(nom2,'(a)')
     >      'Results/eigenval_11_c_nodp.out'
        call displayeigv(nom2,nb,BB)
c  Saving updated base coefficients
        write(nom2,'(a)')
     >      'Results/model_11_c_nodp.out'
        call write_model5(nom2,nb,ryg,bc,var,stdt)
c  Writing fit to data per component
        do i=1,npt
```

```
          djd=ddat(4,i)
          call mjd2dy(djd,dy)
          ddat(4,i)=dy
        end do
        write(nom3,'(a)')
     >       'Results/X_11_c_nodp.out'
        call write_comp(nom3,npt,npm,nd,nt,1,ddat,xyzf)
        write(nom3,'(a)')
     >       'Results/Y_11_c_nodp.out'
        call write_comp(nom3,npt,npm,nd,nt,2,ddat,xyzf)
        write(nom3,'(a)')
     >       'Results/Z_11_c_nodp.out'
        call write_comp(nom3,npt,npm,nd,nt,3,ddat,xyzf)
        close(10)
        close(12)
        stop
        end
c=================================================================
c*****************************************************************
c     module hsplines_sa
c                       V. Lesur 05/05/2008
c     modified for recent data
c                       A.Geese 01/09/2009
c                       E. Nahayo 05/08/2014
c  This module set the variable and define a system of representation
c  close to the harmonic splines over Europe
c
c  Number of parameters as organised in BE:
c  nbhs: number of harmonic splines including their time dependence
c
c    Global data:
c       ryg   : real*8 : reference year
c    irg : integer : number of repeat stations
c       idrvg: integer : derivative order to use
c
c       nbcg: integer : number of centres
c       lwg : integer : maximum degree in local functions
c       cutag: real*8 : max angle above which the function gradient=0
c       cntrg: real*8 : array (3,nbcg) of centre positions (colat;long;radius)
```

```
c       alphg: real*8 : array (lwg+1) spectral values for wavelets
c
c       rag  : real*8 : Harmonic splines reference radius
c
c       isog: integer : spline order
c       nkg : integer : number of time knots for yearly splines
c       tkg : real*8  : array of spot model positions
c
c    Subroutines:
c      init_hsplines_sal2.f    initialise the global variables
c      sub_hsplines_sal2.f     model subroutine
c*********************************************************************
        module hsplines_sa
        implicit none
        integer lwmaxg,nbcmaxg,nbsmaxg
        parameter (lwmaxg=250,nbcmaxg=10000,nbsmaxg=100)
        integer nbcg,lwg, irg,idrvg
        real*8  rag,ryg,alphg(lwmaxg+1),cutag,cntrg(3,nbcmaxg)
        integer isog,nkg
        real*8 tkg(nbsmaxg)
        contains
c*********************************************************************************
c    subroutine init_hsplines_sa
c       V. Lesur  24/04/2007
c  Set initial values for global data and define the model vector bc(*)
c        Anne     20/08/2008
c   Changed function centres definition:
c   not  read in from specific file but calculated according to sampling theorem
c   where all centres are thrown away that are not in the ROI+margin
c       Anne     20/11/2008
c   Increased radius of integration
c   Decreased radius of center positions in order to obtain
c   localized kernels
c       Anne     10/02/2008
c   Changed base functions: Using r, phi & theta
c       Anne     01/09/2009
c   Set new knot positions for recent data
c       E.Nahayo   05/08/2014
c   Set new knot positions for satellite data
```

```
c  output:
c       nb    : integer  : number of parameters in the model
c       bc(nb): real*8   : array of initial parameters
c*****************************************************************************
        subroutine init_hsplines_sa(nb,bc,idrv,lw)
        implicit none
        integer nb,i,j,lw,idrv
        real*8 bc(*),dl,dd,dln,clt,alt,cltgc,rad
        real*8 knot,ys,ye
c  Reference year
        call dy2mjd(2008.0d0,ryg)
c  Default derivative
        idrvg=idrv
c  Max degree & cut angle
        lwg=12
        lw=lwg
        cutag=180.d0
c  Harmonic spline reference radius
      rag=6371.2d0
c  Define the localize function  (alphg(1) correspond to l=0)
        alphg(1)=0.0d0
c       fl=(2*l+1)/((l+1)**4)*l**2)
        do i=2,lwg+1
          dl=dble(i-1)
          dd=(2.d0*dl+1.d0)
          dd=dd/(dl+1.d0)**4
          alphg(i)=-dd/dl**2
        enddo
        open(11,file='Results/alphas.out')
    do i=1,lwg+1
            write(11,*)i,alphg(i)
        enddo
        close(11)
        open(11,file='Results/centres.out')
           do i=1,nbcg
            write(11,*)i,cntrg(1,i),cntrg(2,i),cntrg(3,i)
          enddo
       close(11)
c  Repeat stations offsets number equals number of center positions
```

```
      irg=nbcg
c  Defining splines
      isog=6
      nkg=17
      call dy2mjd(2005.0d0,tkg(1))
      call dy2mjd(2005.0d0,tkg(2))
      call dy2mjd(2005.0d0,tkg(3))
      call dy2mjd(2005.0d0,tkg(4))
      call dy2mjd(2005.0d0,tkg(5))
      call dy2mjd(2005.0d0,tkg(6))
c
      call dy2mjd(2006.0d0,tkg(7))
      call dy2mjd(2007.0d0,tkg(8))
      call dy2mjd(2008.0d0,tkg(9))
      call dy2mjd(2009.0d0,tkg(10))
      call dy2mjd(2010.0d0,tkg(11))
c
      call dy2mjd(2011.0d0,tkg(12))
      call dy2mjd(2011.0d0,tkg(13))
      call dy2mjd(2011.0d0,tkg(14))
      call dy2mjd(2011.0d0,tkg(15))
      call dy2mjd(2011.0d0,tkg(16))
      call dy2mjd(2011.0d0,tkg(17))

      do i=1,nkg
        tkg(i)=(tkg(i)-ryg)/365.25d0
      enddo
      nb=nbcg*(nkg-isog)
      nb=nb+nbcg*(nkg-isog)
      nb=nb+nbcg*(nkg-isog)
      nb=nb+3*irg
      bc(1:nb)=0.d0
      return
      end subroutine init_hsplines_sa
c*********************************************************************
c    subroutine sub_hsplines_sal2
c       V. Lesur  22/01/2009
c*********************************************************************
      recursive subroutine sub_hsplines_sa(iof,nub,nb,bc,bp,be)
```

```
      implicit none
      character iof
      integer nub,nb
      integer i,j,nuhx,nuhy,nuhz,nur
      real*8 bp(*),be(*),bc(*),dw(4)
      real*8, allocatable :: dwx(:),dwy(:),dwz(:)
c  Set values
      be(1:nb)=0.0d0
      if(nub.le.100)then !why the maximum limit is 100?
c  defining pointer for model
      nuhx=1
      nuhy=nuhx+nbcg*(nkg-isog)
      nuhz=nuhy+nbcg*(nkg-isog)
      nur=nuhz+nbcg*(nkg-isog)
c  wwrt calculate be
      allocate(dwx(1:nb),dwy(1:nb),dwz(1:nb))
      dwx(1:nb)=0.0d0
      dwy(1:nb)=0.0d0
      dwz(1:nb)=0.0d0
      call hspln_row('X','n',bp,dwx(nuhx),dwy(nuhx),dwz(nuhx))
      call hspln_row('Y','n',bp,dwx(nuhy),dwy(nuhy),dwz(nuhy))
      call hspln_row('Z','n',bp,dwx(nuhz),dwy(nuhz),dwz(nuhz))
      if(nub.lt.10)
     > call sub_ofst(bp,dwx(nur),dwy(nur),dwz(nur))
      i=mod(nub,10)
      if(i.eq.1) be(1:nb)=dwx(1:nb)
      if(i.eq.2) be(1:nb)=dwy(1:nb)
      if(i.eq.3) be(1:nb)=dwz(1:nb)
      if(i.eq.4) then
         dw(1:4)=0.0d0
         do i=1,nb
           dw(1)=dw(1)+dwx(i)*bc(i)
           dw(2)=dw(2)+dwy(i)*bc(i)
           dw(3)=dw(3)+dwz(i)*bc(i)
         enddo
         dw(4)=dsqrt(dw(1)**2+dw(2)**2+dw(3)**2)
         dw(1:3)=dw(1:3)/dw(4)
         be(1:nb)=dw(1)*dwx(1:nb)+dw(2)*dwy(1:nb)+dw(3)*dwz(1:nb)
      endif
```

```
        deallocate(dwx,dwy,dwz)
        if(iof.eq.'f')be(1:nb)=be(1:nb)*bc(1:nb)
        else
c        damping equations
          if(nub.eq.101)then
            i=int(bp(1))
            be(i:i)=1.d0
          endif
          if(iof.eq.'f')be(1:nb)=0.0d0
        endif
        return
        end subroutine  sub_hsplines_sa
c***********************************************************************
c       subroutine hspln_row.f      V. Lesur 07/03/2007
c       Compute a row of the matrix of condition
c***********************************************************************
        subroutine hspln_row(dir,soc,bp,dx,dy,dz)
        implicit none
        integer i,nb
        real*8 bp(*),dx(*),dy(*),dz(*)
        character soc,dir
        integer ik,nbt
        real*8 dt
c Number of parameters for harmonic splines
        nb=nbcg*(nkg-isog)
c Find the spline interval
        dt=(bp(4)-ryg)/365.25d0
        ik=2
        do while (dt.ge.tkg(ik).and.ik.lt.nkg)
          ik=ik+1
        enddo
        ik=ik-1
c Calculate dx,dy,dz vectors
        call XYZlocal_XYZtd(dir,isog,nkg,tkg,ik,ryg,idrvg
     >                           ,1,nbcg,lwg,alphg,cntrg,rag
     >                           ,cutag,bp,dx,dy,dz)
c Rotate in Cartesian and then in X,Y,Z,SM
        if(soc.eq.'y')then
        do i=1,nb
```

```
          dx(i)=-dx(i)
          dz(i)=-dz(i)
          call SPH_GEO(bp(1),bp(2),dx(i),dy(i),dz(i))
          call GEO_SMm(bp(4),dx(i),dy(i),dz(i))
        enddo
        endif
        return
        end subroutine hspln_row
c*********************************************************************
c       subroutine sub_ofst.f      Anne 18/05/2008
c       for the repstat offsets
c   yon - ofsets in SM system of coordinates
cccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
        subroutine sub_ofst(bp,dx,dy,dz)
        implicit none
        integer i,nb,ir
        real*8 bp(*),dx(*),dy(*),dz(*)
c Number of parameters for lithosphere field
        nb=3*irg
        dx(1:nb)=0.0d0
        dy(1:nb)=0.0d0
        dz(1:nb)=0.0d0
        ir=(int(bp(5))-1)*3
c Calculate dx,dy,dz vectors
        dx(ir+1)=1.0d-1
        dy(ir+2)=1.0d-1
        dz(ir+3)=1.0d-1
        return
        end subroutine sub_ofst
c**********************************************************************
c       subroutine smooth_time.f      V. Lesur 23/08/2007
c       build the damping matrix to minimise the integration in
c       space and time of the "idrv" time derivative of the radial
c       component of the magnetic field  squared. The integration is
c       at the reference radius
c***********************************************************************
        subroutine smooth_time2(dl,nb,gg,bb,bc)
        implicit none
        integer nb
```

```
      real*8 dl(*),gg(nb,*),bb(*),bc(*)
      integer nspl,idrv,il,is,it,iu,iv,i,j,k,l,m
      real*8 ra,dd,dthi,dphi,dri,dthj,dphj,drj,dij(3,3)
      real*8 pposi(3),pposj(3)
      real*8, allocatable :: damp(:),dlf(:),dbi(:,:)
      real*8 dcosd,dsind
      nspl=nkg-isog
      allocate (damp(nspl*nspl),dlf(lwg+1),dbi(3*nspl,3))
      ra=rag
c   Compute time damping sub-matrix
      idrv=2
      call damp_spline(isog,nkg,tkg,idrv,damp)
c   For each centre pairs
      do i=1,nbcg
        pposi(1)=cntrg(1,i)
        pposi(2)=cntrg(2,i)
        pposi(3)=cntrg(3,i)
        dbi=0.0d0
        do j=1,nbcg
          pposj(1)=cntrg(1,j)
          pposj(2)=cntrg(2,j)
          pposj(3)=cntrg(3,j)
          dij=0
          call  Br2(pposi,pposj,alphg,rag,lwg,dij(1,1))
          call  Bt2(pposi,pposj,alphg,rag,lwg,dij(2,2))
          call  Bp2(pposi,pposj,alphg,rag,lwg,dij(3,3))
          call  BrBt(pposi,pposj,alphg,rag,lwg,dij(1,2))
          call  BrBp(pposi,pposj,alphg,rag,lwg,dij(1,3))
          call  BtBp(pposi,pposj,alphg,rag,lwg,dij(2,3))
          dij(2,1)=dij(1,2)
          dij(3,1)=dij(1,3)
          dij(3,2)=dij(2,3)
c   build damping
          l=(i-1)*nspl
          m=(j-1)*nspl
          do is=1,nspl
            do it=1,nspl
              do iu=1,3
                do iv=1,3
```

```
                   k=(it-1)*nspl+is
                   dd=dl(1)*dij(iu,iv)*damp(k)
                   gg(l+is+(iu-1)*nbcg*nspl,m+it+(iv-1)*nbcg*nspl)=
     >             gg(l+is+(iu-1)*nbcg*nspl,m+it+(iv-1)*nbcg*nspl)+dd
                   dbi(is,iu)=dbi(is,iu)+dd*bc(m+is+iu)
                  enddo
                enddo
              enddo
            enddo
          enddo
          l=(i-1)*nspl
          do is=1,nspl
             do iu=1,3
              bb(l+is+(iu-1)*nbcg*nspl)=
     >        bb(l+is+(iu-1)*nbcg*nspl)-dbi(is,iu)
             enddo
          enddo
        enddo
        deallocate (damp,dlf,dbi)
        return
        end subroutine smooth_time2
c*********************************************************************
c       subroutine smooth_space.f      V. Lesur 23/08/2007
c       build the damping matrix to minimise the second space
c       derivative at the CMB of the hsplines
c*********************************************************************
        subroutine smooth_space(dl,nb,gg,bb,bc)
        implicit none
        integer nb
        real*8 dl(*),gg(nb,*),bb(*),bc(*)
        integer nspl,idrv,il,is,it,iu,iv,i,j,k,l,m
        real*8 ra,dd,dthi,dphi,dri,dthj,dphj,drj,dij(3,3)
        real*8 pposi(3),pposj(3)
        real*8, allocatable :: damp(:),dlf(:),dbi(:,:)
       real*8 dcosd,dsind
        nspl=nkg-isog
        allocate (damp(nspl*nspl),dlf(lwg+1),dbi(3*nspl,3))
        ra=rag
c   Compute time damping sub-matrix
```

```
        idrv=0
        call damp_spline(isog,nkg,tkg,idrv,damp)
c   For each centre pairs
        do i=1,nbcg
          pposi(1)=cntrg(1,i)
          pposi(2)=cntrg(2,i)
          pposi(3)=cntrg(3,i)
          dbi=0.0d0
          do j=1,nbcg
            pposj(1)=cntrg(1,j)
            pposj(2)=cntrg(2,j)
            pposj(3)=cntrg(3,j)
            dij=0
            call  Br2(pposi,pposj,alphg,rag,lwg,dij(1,1))
            call  Bt2(pposi,pposj,alphg,rag,lwg,dij(2,2))
            call  Bp2(pposi,pposj,alphg,rag,lwg,dij(3,3))
            call  BrBt(pposi,pposj,alphg,rag,lwg,dij(1,2))
            call  BrBp(pposi,pposj,alphg,rag,lwg,dij(1,3))
            call  BtBp(pposi,pposj,alphg,rag,lwg,dij(2,3))
            dij(2,1)=dij(1,2)
            dij(3,1)=dij(1,3)
            dij(3,2)=dij(2,3)
c   build damping
            l=(i-1)*nspl
            m=(j-1)*nspl
            do is=1,nspl
              do it=1,nspl
                do iu=1,3
                  do iv=1,3
                   k=(it-1)*nspl+is
                   dd=dl(1)*dij(iu,iv)*damp(k)
                   gg(l+is+(iu-1)*nbcg*nspl,m+it+(iv-1)*nbcg*nspl)=
     >             gg(l+is+(iu-1)*nbcg*nspl,m+it+(iv-1)*nbcg*nspl)+dd
                   dbi(is,iu)=dbi(is,iu)+dd*bc(m+is+iu)
                  enddo
                enddo
              enddo
            enddo
          enddo
```

```
          l=(i-1)*nspl
          do is=1,nspl
             do iu=1,3
              bb(l+is+(iu-1)*nbcg*nspl)=
     >        bb(l+is+(iu-1)*nbcg*nspl)-dbi(is,iu)
             enddo
          enddo
        enddo
        deallocate (damp,dlf,dbi)
        return
        end subroutine smooth_space
c*********************************************************************
c        subroutine smooth_st       V.Lesur 14/05/2005
c       Smooth in Space and time
c*********************************************************************
        subroutine smooth_st(dl,nb,gg,bb,bc)
        implicit none
        integer nb
        real*8 dl(*),gg(nb,*),bb(*),bc(*)
        real*8 dw(2)
c    First the time
        if(dl(1).ne.0.0d0)then
          call smooth_time2(dl,nb,gg,bb,bc)
        endif
c    Second in space
        if(dl(2).ne.0.0d0)then
          dw(1)=dl(2)
          dw(2)=0.0d0
          call smooth_space(dw,nb,gg,bb,bc)
        endif
        return
        end subroutine smooth_st
c********************************************************************
        end module hsplines_sa
c********************************************************************

c===============================================================================
c*******************************************************************************
c subroutine opt_d_l3
```

```
c Vincent Lesur  13/01/2005
c vbfl: modified 24/08/05 for:
c         1- use the transpose of ppos and remove ddat from calling parameters
c vbfl: modified 14/03/07 for:
c         1- preconditioner
c       Should be used with F90
c       calling recurcive subroutines
c       non-parallel optimiser using non-gradient method
c       Computes parameters BC from a set of NPT data points.
c       The "base function" subroutine used is BS
c       The norm used for minimisation correspond to the FM misfit function
c       The STD is computed using fstd function and
c       the damping is done using the Fdamp subroutine
c       All these subroutines & functions are user provided and shoud be
c        declared external in the calling program.
c       The normal equations are built by adding the contributions of groups
c       of uncorrelated data points. The Data covarience matrix, input in a
c       SLAP column format, is used to determine groups of uncorrelated data
c       The inversion process may be non-linear => First estimation of the
c       solution must be given.
c       CALLED: cpt_dat_vals, cptstd, cptnp, cptnormeq
c         &     fdamp (external), eig_solve2, eigen_vv
c       depending on itmax:
c         itmax > 0       find parameters to fit the data
c         itmax = 0       do the forward problem only
c         itmax < 0       compute eigen values and vectors of normal equations
c       input:
c               path          where to write output files
c               itmax         Maximum number of itterations
c               npmax         number max of data point with correlated errors
c               npm           Maximum total number of data points
c               nd            space dimension
c               npt           Total Number of data
c               ppos          data point position in ndD
c               nb            Number or base function to use
c               BC            First estimation of Base function coefficients
c               dl(3)         damping factor & max condition number acceptable
c               FM            misfit function (like l2_norm.f)
c               BS            Base subroutine to use
```

```
c               fstd          std Function
c               fdamp         damping Subroutine
c               nt(*)         data type
c               cov(*)        covariance matrix in SLAP Column format
c               icov/jcov     Integer vector describing cov format
c               stdt          STD Target
c       output:
c               BC(*)         Base function coefficients
c               xyzf(*)       results of forward modelling
c               stdt          estimated STD
c               GG            eigen vectors of the last normal matrix
c               BB            eigen values of the last normal matrix
c*******************************************************************************
        subroutine opt_d_l3(path,itmax,npmax,npm,nd,npt,ppos,nb,bc,dl
     > ,FM,BS,fstd,FDAMP,nt,cov,icov,jcov,stdt,xyzf,BB,GG)
        implicit none
        integer itmax,nd,npt,nb,nt(*),icov(*),jcov(*),i,j,ip,npmax,npm
        real*8  ppos(*),bc(*),cov(*),stdt,dl(*),xyzf(*)
        real*8  BB(*),GG(*)
        real*8, allocatable :: dw(:),dsol(:),eigv(:),ddat(:)
        character*2 yon_iterate,yon_build,yon_solve
        character path*100
        real*8 FM,FSTD
        external FM,BS,FSTD,FDAMP
        integer it
        real*8 std
        character cit*2
        real*4 t,tarray(2),dtime
        std=0.0d0
        it=0
        yon_iterate='ok'
        if(itmax.gt.0)then
          yon_build='in'
          yon_solve='in'
        elseif(itmax.eq.0)then
          yon_build='no'
          yon_solve='no'
        else
          yon_build='ok'
```

```
          yon_solve='no'
        endif
        do while (yon_iterate.eq.'ok')
          it=it+1
c  All: set up
          allocate (ddat(1:npt))
          do i=1,npt
            ddat(i)=ppos(i*(nd+1))
          enddo
c  All: Forward modelling & std estimate
          t=dtime(tarray)
          call cpt_dat_vals(nd,npm,npt,nt,ppos,nb,BC,BS,xyzf)
          call cptstd(npmax,npt,icov,jcov,cov,ddat,xyzf,fstd,std)
          t=dtime(tarray)
          write(*,'(A,e9.2,A,e9.2)')'Forward modelling: user time'
     >        ,tarray(1),' system time',tarray(2)
          write(*,*)
          write(*,*)'std =',std,'  & it   =',it-1
          write(*,*)
c  All: What to do next ?
          if(std.gt.stdt.and.yon_build.eq.'in') yon_build='ok'
c  All: build normal equations if std > stdt or itmax < 0
          if (yon_build.eq.'ok') then
c  MP: start clock
            t=dtime(tarray)
c  SP: build their parts of normal equations
            ip=1
    write(*,'(A)') 'Here'
            call cptnormeq(npmax,npm,npt,ip,nd
     >       ,ppos,nb,FM,BS,bc,icov,jcov,cov,ddat,nt,xyzf,GG,BB)
        write(*,'(A)') 'Here'
c  MP: stop clock
            t=dtime(tarray)
            write(*,'(A,e9.2,A,e9.2)')'Normal equations: user time'
     >          ,tarray(1),' system time',tarray(2)
            write(*,*)
c  MP: Damp the normal equations
            if (it<2) call fdamp(dl,nb,GG,BB,bc)
c  All: What to do next ?
```

```
          if(yon_solve.eq.'in')yon_solve='ok'
        endif
        deallocate(ddat)
c  MP: Now solve the normal equations: it's a MP only
        if (yon_solve.eq.'ok') then
c  MP: Solve
          allocate(dsol(1:nb),eigv(1:nb),dw(1:nb))
          t=dtime(tarray)
          call precond('d',nb,GG,BB,dsol,dw)
          call eig_solve2(nb,dl(3),GG,eigv,BB,dsol)
          call precond('i',nb,GG,BB,dsol,dw)
          t=dtime(tarray)
          write(*,'(A,e9.2,A,e9.2)')'solving system: user time'
     >          ,tarray(1),' system time',tarray(2)
c  MP: Update parameters
          do i=1,nb
            BC(i)=BC(i)+dsol(i)
            BB(i)=eigv(i)
          enddo
          deallocate(dsol,eigv,dw)
c  MP: and save
          cit(2:2)=char(48+it-int(it/10.)*10)
          cit(1:1)=char(48+int(it/10.))
          i=0
          do while (path(i+1:i+1).ne.' ')
            i=i+1
          enddo
          open(10,file=path(1:i)//'model_it'//cit//'.out')
            write(10,*)'# model at iteration : ',cit
            do i=1,nb
              write(10,*)i,BC(i)
            enddo
          close(10)
        endif
c  All: What to do next iteration ?
        if(it.lt.itmax) then
          if(yon_solve.eq.'ok')then
            yon_iterate='ok'
            yon_build='in'
```

```
            yon_solve='in'
          else
            yon_iterate='no'
          endif
        else
          if(yon_solve.eq.'ok')then
            yon_iterate='ok'
            yon_build='no'
            yon_solve='no'
          else
            yon_iterate='no'
          endif
        endif
c  All: End of itterative process
      enddo
c  MP: SVD of GG with eigenvalues in BB and QUIT
      if (itmax.lt.0) then
        allocate(dw(1:nb))
        call eigenvv(nb,GG,BB,dw)
        deallocate(dw)
      endif
      stdt=std
      return
      end

%**************************************************
%Subroutines
%**************************************************
cccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
c   subroutine AtB
c V. Lesur 20/02/2002
c    modified 23/02/2004: GOTO removed
c    modified 09/05/2007: IMPLICIT NONE
c       Left Multiplication of the transpose of a nlXnc1 A1 matrix with
c        the nlXnc2 A2 matrix  NC1.LE.NL
c       input:
c       nl Number of lines
c       nc1/2   Number of column of A1/2
c       A1/2    Matrices
```

```
c       dw      work array dim min nc1
c       output:
c       A2             transpose(A1)*A2
cccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
        subroutine AtB(nl,nc1,nc2,A1,A2,dw)
        implicit none
        integer nl,nc1,nc2,ic,il,k,k1,k2
        real*8 A1(*),A2(*),dw(*)
        if(nc1.le.nl) then
        do ic=1,nc2
          do il=1,nc1
            dw(il)=0.0d0
            do k=1,nl
              k1=k+(il-1)*nl
              k2=k+(ic-1)*nl
              dw(il)=dw(il)+A1(k1)*A2(k2)
            enddo
          enddo
          do il=1,nc1
            k2=il+(ic-1)*nc1
            A2(k2)=dw(il)
          enddo
        enddo
        else
          write(*,*)'Error in AtB'
          stop
        endif
        return
        end
cccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
c Subroutine Bp2
c Anne Hemshorn 26/02/2009
c   Computes for a center pair (i,j)
c   sum_l^L f_l^2 4pi(l+1)^4/(2l+1) *(C/r_i)^(l+2) *(C/rj)^(l+2)
c              *(d_phi Ylm(i)* d_phi Ylm(j))
c   squared radial field for damping
c   colat,long,radius given in degree,degree,km
c       input:
c         lwg      INTEGER     Maximum SH degree
```

```
c         pposi(3) REAL*8      array of positions in space
c         pposj(3)             ppos(1)=colat (deg)
c                              ppos(2)=long  (deg)
c                              ppos(3)=radius (km)
c         alph(*)  REAL*8      array of alpha values
c         ra       REAL*8      reference radius
c       output:
c         yy    REAL*8
cccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
subroutine Bp2(pposi,pposj,alph,ra,lw,yy)
      implicit none
      real*8 pposi(3),pposj(3),alph(*),ra,st
      real*8 yy,fac(lw+1),dri,drj,dsind,dl
      real*8, allocatable :: biy(:),bjy(:)
      integer lw,i,j,k,l
      external dsind
      i=lw*(lw+2)
      allocate(biy(1:i),bjy(1:i))
      dri=ra/pposi(3)
      drj=ra/pposj(3)
      call Ysph_bi0(lw,pposi(1),biy)
      call Ysph_bi0(lw,pposj(1),bjy)
      do i=1,lw+1
      dl=dble(i-1)
      fac(i)=alph(i)**2
      fac(i)=fac(i)*dble((dl+1)**2)/(dble(2*dl+1))
      fac(i)=fac(i)*drj**(i+2)*dri**(i+2)
      fac(i)=fac(i)/(dsind(pposi(1))*dsind(pposj(1)))
      end do
      yy=0.0d0
      do i=1,lw
         k=i*i
         yy=yy+fac(i+1)*biy(k)*bjy(k)
         do j=1,i
           k=i*i+2*(j-1)+1
           yy=yy+fac(i+1)*biy(k)*bjy(k)
           yy=yy+fac(i+1)*biy(k+1)*bjy(k+1)
         enddo
      enddo
```

```
      end subroutine Bp2
cccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
c Subroutine Br2
c Anne Hemshorn 26/02/2009
c   Computes for a center pair (i,j)
c   sum_l^L f_l^2 4pi(l+1)^4/(2l+1) *(C/r_i)^(l+2) *(C/rj)^(l+2)*(Pl0(cos(m)))
c   squared radial field for damping
c   colat,long,radius given in degree,degree,km
c       input:
c         lwg      INTEGER     Maximum SH degree
c         pposi(3) REAL*8      array of positions in space
c         pposj(3)             ppos(1)=colat (deg)
c                              ppos(2)=long  (deg)
c                              ppos(3)=radius (km)
c         alph(*)  REAL*8      array of alpha values
c         ra       REAL*8      reference radius
c       output:
c         zz    REAL*8
cccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
c   Compute F radial derivative space integration
      subroutine Br2(pposi,pposj,alph,ra,lw,zz)
      real*8 pposi(3),pposj(3),alph(*),ra,zz
      real*8 dd
      real*8, allocatable :: dlf(:)
      integer lw,il
      allocate (dlf(lw+1))
      dthi=pposi(1)
      dphi=pposi(2)
      dri=ra/pposi(3)
      dthj=pposj(1)
      dphj=pposj(2)
      drj=ra/pposj(3)
c     compute angle between the two positions
      dd=dsind(dthi)*dsind(dthj)*dcosd(dphi-dphj)
      dd=dcosd(dthi)*dcosd(dthj)+dd
      call mk_lf_0(lwg,dd,dlf)
      do il=1,lw
         zz=dble((il+1)**4)/dble(2*il+1)
         zz=zz*alph(il+1)**2
```

```
         zz=zz*dlf(il+1)
         zz=zz*dri**(il+2)
         zz=zz*drj**(il+2)
      enddo
      end subroutine Br2
cccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
c Subroutine BrBp2
c Anne Hemshorn 26/02/2009
c   Computes for a center pair (i,j)
c   sum_l^L f_l^2 4pi(l+1)^3/((2l+1)*sin(theta_j)) *(C/r_i)^(l+2) *(C/rj)^(l+2)
c              *( Ylm(i)* d_phi Ylm(j))
c   squared radial field for damping
c   colat,long,radius given in degree,degree,km
c       input:
c         lwg      INTEGER     Maximum SH degree
c         pposi(3) REAL*8      array of positions in space
c         pposj(3)             ppos(1)=colat (deg)
c                              ppos(2)=long  (deg)
c                              ppos(3)=radius (km)
c         alph(*)  REAL*8      array of alpha values
c         ra       REAL*8      reference radius
c       output:
c         zy    REAL*8
cccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
subroutine BrBp(pposi,pposj,alph,ra,lw,zy)
      implicit none
      real*8 pposi(3),pposj(3),alph(*),ra,zy
      real*8 fac(lw+1),dl,dri,drj,dsind
      real*8, allocatable :: biz(:),bjy(:)
      integer lw,il,i,j,k,l
      external dsind
      i=lw*(lw+2)
      allocate(biz(1:i),bjy(1:i))
      dri=ra/pposi(3)
      drj=ra/pposj(3)
      call Zsph_bi0(lw,pposi(1),biz)
      call Ysph_bi0(lw,pposj(1),bjy)
      do i=1,lw+1
      dl=dble(i-1)
```

```
      fac(i)=alph(i)
      fac(i)=fac(i)*dble((dl+1)**3)/(dble(2*dl+1))
      fac(i)=fac(i)*drj**(i+2)*dri**(i+2)
      fac(i)=fac(i)/(dsind(pposi(1)))
      end do
      zy=0.0d0
      do i=1,lw
            k=i*i
            zy=zy+fac(i+1)*biz(k)*bjy(k)
            do j=1,i
              k=i*i+2*(j-1)+1
              zy=zy+fac(i+1)*biz(k)*bjy(k)
              zy=zy+fac(i+1)*biz(k+1)*bjy(k+1)
            enddo
      enddo
      end subroutine BrBp
cccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
c Subroutine BrBt
c Anne Hemshorn 26/02/2009
c   Computes for a center pair (i,j)
c   sum_l^L f_l^2 4pi(l+1)^3/(2l+1) *(C/r_i)^(l+2) *(C/rj)^(l+2)
c              *(Ylm(i)* d_theta Ylm(j))
c   squared radial field for damping
c   colat,long,radius given in degree,degree,km
c       input:
c         lwg      INTEGER     Maximum SH degree
c         pposi(3) REAL*8      array of positions in space
c         pposj(3)             ppos(1)=colat (deg)
c                              ppos(2)=long  (deg)
c                              ppos(3)=radius (km)
c         alph(*)  REAL*8      array of alpha values
c         ra       REAL*8      reference radius
c       output:
c         xz    REAL*8
cccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
subroutine BrBt(pposi,pposj,alph,ra,lw,xz)
      implicit none
      real*8 pposi(3),pposj(3),alph(*),ra,xz
      real*8 fac(lw+1),dl,dri,drj
```

```
      real*8, allocatable :: bix(:),bjz(:)
      integer lw,il,i,j,k,l
      i=lw*(lw+2)
      allocate(bix(1:i),bjz(1:i))
      dri=ra/pposi(3)
      drj=ra/pposj(3)
      call Zsph_bi0(lw,pposj(1),bjz)
      call Xsph_bi0(lw,pposi(1),bix)
      do i=1,lw+1
      dl=dble(i-1)
      fac(i)=alph(i)**2
      fac(i)=fac(i)*dble((dl+1)**3)/(dble(2*dl+1))
      fac(i)=fac(i)*drj**(i+2)*dri**(i+2)
      end do
      xz=0.0d0
      do i=1,lw
         k=i*i
         xz=xz+fac(i+1)*bix(k)*bjz(k)
         do j=1,i
           k=i*i+2*(j-1)+1
           xz=xz+fac(i+1)*bix(k)*bjz(k)
           xz=xz+fac(i+1)*bix(k+1)*bjz(k+1)
         enddo
      enddo
      end subroutine BrBt
ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
c Subroutine Bt2
c Anne Hemshorn 26/02/2009
c   Computes for a center pair (i,j)
c   sum_l^L f_l^2 4pi(l+1)^4/(2l+1) *(C/r_i)^(l+2) *(C/rj)^(l+2)
c              *(d_theta Ylm(i)* d_theta Ylm(j))
c   squared radial field for damping
c   colat,long,radius given in degree,degree,km
c       input:
c         lwg      INTEGER     Maximum SH degree
c         pposi(3) REAL*8      array of positions in space
c         pposj(3)             ppos(1)=colat (deg)
c                              ppos(2)=long  (deg)
c                              ppos(3)=radius (km)
```

```
c          alph(*)  REAL*8      array of alpha values
c          ra       REAL*8      reference radius
c        output:
c          xx    REAL*8
ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
  subroutine Bt2(pposi,pposj,alph,ra,lw,xx)
      implicit none
      real*8 pposi(3),pposj(3),alph(*),ra,xx
      real*8 fac(lw+1),dl,dri,drj
      real*8, allocatable :: bix(:),bjx(:)
      integer lw,il,i,j,k,l
      i=lw*(lw+2)
      allocate(bix(1:i),bjx(1:i))
      dri=ra/pposi(3)
      drj=ra/pposj(3)
      call Xsph_bi0(lw,pposi(1),bix)
      call Xsph_bi0(lw,pposj(1),bjx)
      do i=1,lw+1
      dl=dble(i-1)
      fac(i)=alph(i)**2
      fac(i)=fac(i)*dble((dl+1)**2)/(dble(2*dl+1))
      fac(i)=fac(i)*drj**(i+2)*dri**(i+2)
      end do
      xx=0.0d0
      do i=1,lw
         k=i*i
         xx=xx+fac(i+1)*bix(k)*bjx(k)
         do j=1,i
           k=i*i+2*(j-1)+1
           xx=xx+fac(i+1)*bix(k)*bjx(k)
           xx=xx+fac(i+1)*bix(k+1)*bjx(k+1)
         enddo
      enddo
      end subroutine Bt2
ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
c Subroutine Bt2
c Anne Hemshorn 26/02/2009
c   Computes for a center pair (i,j)
c   sum_l^L f_l^2 4pi(l+1)^4/(2l+1) *(C/r_i)^(l+2) *(C/rj)^(l+2)
```

```
c              *(d_theta Ylm(i)* d_theta Ylm(j))
c   squared radial field for damping
c   colat,long,radius given in degree,degree,km
c       input:
c         lwg      INTEGER     Maximum SH degree
c         pposi(3) REAL*8      array of positions in space
c         pposj(3)             ppos(1)=colat (deg)
c                              ppos(2)=long  (deg)
c                              ppos(3)=radius (km)
c         alph(*)  REAL*8      array of alpha values
c         ra       REAL*8      reference radius
c       output:
c         xz    REAL*8
cccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
  subroutine BtBp(pposi,pposj,alph,ra,lw,xz)
      implicit none
      real*8 pposi(3),pposj(3),alph(*),ra,xz
      real*8 fac(lw+1),dl,dri,drj,dsind
      real*8, allocatable :: bix(:),bjx(:)
      integer lw,il,i,j,k,l
      external dsind
      i=lw*(lw+2)
      allocate(bix(1:i),bjx(1:i))
      dri=ra/pposi(3)
      drj=ra/pposj(3)
      call Xsph_bi0(lw,pposj(1),bjx)
      do i=1,lw+1
      dl=dble(i-1)
      fac(i)=alph(i)**2
      fac(i)=fac(i)*dble((dl+1)**3)/(dble(2*dl+1))
      fac(i)=fac(i)*drj**(i+2)*dri**(i+2)
      fac(i)=fac(i)/(dsind(pposi(1))*dsind(pposj(1)))
      end do
      xz=0.0d0
      do i=1,lw
         k=i*i
         xz=xz+fac(i+1)*bix(k)*bix(k)*bjx(k)*bjx(k)
         do j=1,i
           k=i*i+2*(j-1)+1
```

```
          xz=xz+fac(i+1)*bix(k)*bix(k)*bjx(k)*bjx(k)
          xz=xz+fac(i+1)*bix(k+1)*bix(k+1)*bjx(k+1)*bjx(k+1)
        enddo
      enddo
      end subroutine BtBp
cccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
        subroutine build_dat(sdat,tk)
        implicit none
        integer nb,npt,nl,i,j,k,l,nloc,lw,test,idrv
        real*8  sdat(*),tk(*),nt(8000)
        real*8  dyr,rad,xx,yy,zz,dd,djd,cltgc,dln,dt,ds
        real*8  inte(360,360),Br1,Br2,norm,deltax,deltay
        character nom2*100,cc*4
        real*8 l2_norm,l2_std
        real*8 l1_norm,l1_std,dsind
        cltgc= 0.d0
        dln=   0.d0
        rad= 6371.2d0
        npt=0
        do i = 1,175,10
        deltay=i*1.0d0
        do j = 1,355,10
        deltax=j*1.0d0
        do k = 6,15
          tk(k)=tkg(k)*365.25d0+ryg
          call mjd2dy(tk(k),dyr)
          do l = 1,4
             npt = npt + 1
             sdat(1,npt) = cltgc + deltay     !colat (deg)
             sdat(2,npt) = dln + deltax        !long (deg)
             sdat(3,npt) = rad         !radius (km)
c             call dy2mjd(dyr,djd)
             sdat(4,npt) =  djd        !date decimal year
             sdat(5,npt) = 2.0d0       !HER
             nt(npt) = l+10
          enddo
        enddo
        enddo
        enddo
```

```
        end subroutine build_dat
cccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
c Program cal_norm
c               based on forward by
c Vincent Lesur 06/05/2008
c     modified many times
c            Anne Hemshorn 30/03/2009
cccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
        subroutine calc_norm(nom2,norm)
        use hsplines_sa_x
        implicit none
        integer nbm,npm,nd
        parameter (nbm=12000,npm=600000,nd=6)
        integer nt(npm),st,ss
        integer nb,npt,nl,i,j,k,l,nloc,lw,test,idrv
        real*8  bc(nbm),sdat(nd+1,npm),xyzf(npm),tk(25)
        real*8  dyr,rad,pow(nbm),xx,yy,zz,dd,djd,cltgc,dln,dt,ds
        real*8  inte(360,360),Br1,Br2,norm,deltax,deltay
        character nom2*100,cc*4
        real*8 l2_norm,l2_std
        real*8 l1_norm,l1_std,dsind
        external l2_norm,l2_std
        external l1_norm,l1_std
        external damp_rien
        external dsind
        open(11,file='Results/centres.out')
        do i=1,nbcmaxg
          read(11,*,iostat=test)j,cntrg(1,j),cntrg(2,j),cntrg(3,j)
          if(test.ne.0) EXIT
        enddo
        close(11)
        nbcg=i-1
        ds=1.d0
        dt=1.d0
        do i=1,ss
          ds=ds*0.1d0
        end do
        do i=1,st
          dt=dt*0.1d0
```

```
end do
idrv=2
call init_hsplines_sa_x(nb,bc,idrv)
open(10,file=nom2,status='old',iostat=test)
if(test.ne.0) THEN
write(*,*) 'ERROR in calc_norm: NO INFILE ', nom2
else
read(10,*)cc
read(10,*)cc
read(10,*)cc
read(10,*)dd
do i=1,nb
   read(10,*)cc,j,k,bc(i),dd
enddo
close(10)
cltgc= 0.d0
dln=   0.d0
rad= 6371.2d0
npt=0
do i = 1,175,10
deltay=i*1.0d0
do j = 1,355,10
deltax=j*1.0d0
do k = 6,15
  tk(k)=tkg(k)*365.25d0+ryg
  call mjd2dy(tk(k),dyr)
  do l = 1,4
     npt = npt + 1
     sdat(1,npt) = cltgc + deltay     !colat (deg)
     sdat(2,npt) = dln + deltax        !long (deg)
     sdat(3,npt) = rad         !radius (km)
     call dy2mjd(dyr,djd)
     sdat(4,npt) =  djd        !date decimal year
     sdat(5,npt) = 2.0d0       !HER
     nt(npt) = l+10
  enddo
enddo
enddo
enddo
```

```
      write(*,*)npt
      call cpt_dat_vals(nd,npm,npt,nt,sdat,nb,
     >                     bc,sub_hsplines_sa_x,xyzf)
      open(24,file='Results/forward_norm')
      nloc = 1
      do k=6,15
      do i = 1,175,10
        do j = 1,355,10
          write (24,'(3f14.4,4e15.7)')
     >    90.d0-sdat(1,nloc),sdat(2,nloc),sdat(4,nloc)
     >    ,xyzf(nloc),xyzf(nloc+1),xyzf(nloc+2)
     >    ,xyzf(nloc+3)
           nloc = nloc + 4
        enddo
      enddo
      enddo
      close(24)
      nloc = 1
      do i = 1,175,10
      do j = 1,355,10
      inte(i,j)=0
      do k = 6,14
          Br1=xyzf(nloc+2)
          Br2=xyzf(nloc+6)
          inte(i,j)=inte(i,j)+(Br2**2+Br1**2+Br1*Br2)*2.5d0
          nloc = nloc + 4
      enddo
      enddo
      enddo
      norm=0
      do i=1,175,10
      do j=1,355,10
       norm=norm+inte(i,j)*dsind(cltgc+deltay)
      end do
      end do
      end if
      end subroutine calc_norm
cccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
```

```
c subroutine concoct_B
c V. Lesur 30/01/2001
c       From a Vector B= A1'.W1'.R.W1.b1  (nl1 elements),
c       the matrices A2 (nl2 rows and nl1 columns), W2 (diagonal nl2Xnl2)
c       and the vector b2 (nl2 elements)
c       "concocte" the Vector  B=A1'.W1'.R.W1.b1 +A2'.W2'.R.W2.b2
c       where R are a misfit matrices (diagonal)
c             W are diagonal weigth matrices
c             A are matrices built from the equations of condition
c             b are vectors of the right-hand side of equations of conditions
c   Input:
c BB Vector ofthe right-hand side of normal equations (nl1)
c       A2 Matrice of equations of condition (nl2Xnl1)
c       W2s     Inverse of Diagonal covariance matrix (W2 matrix squared) (nl2)
c       B2      Vector of right-hand side of equations of conditions (nl2)
c       FM      Misfit Function
c    (user provided, external in calling program)
c       MV      Misfit to data vector (nl2)
c       nl1/2   number of lignes
c   output:
c BB      Updated Vector ofthe right-hand side of normal equations (nl1)
cccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
subroutine concoct_B(FM,nl1,nl2,MV,W2s,A2,B2,BB)
        implicit none
        integer nl1,nl2,il,j
        real*8 MV(*),W2s(*),A2(nl2,*),B2(*),BB(*),dw
        real*8 FM
        external FM
        do j=1,nl2
          dw=FM(j,MV)*W2s(j)*B2(j)
          do il=1,nl1
            BB(il)=BB(il)+A2(j,il)*dw
          enddo
        enddo
        return
        end
cccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
c subroutine concoct_G
c V. Lesur 30/01/2001
```

```
c       From a Symmetric Matrix G= A1'.W1'.R.W1.A1 of nl1 lignes and columns
c       and the matrices A2 (nl2 lines and nl1 columns), W2 (diagonal nl2Xnl2)
c       "concocte" the matrix  G=A1'.W1'.R.W1.A1 +A2'.W2'.R.W2.A2
c       where R are a misfit matrices
c             W are diagonal weigth matrices
c             A are matrices built from the equations of condition
c   Input:
c GG Matrix of normal equations (nl1Xnl1)
c       A2 Matrice of equations of condition (nl2Xnl1)
c       W2s     Inverse of Diagonal covariance martix (W2 matrix squared) (nl2)
c       FM      Misfit Function
c    (user provided, external in calling program)
c       MV      Misfit to data vector (nl2)
c       nl1/2   number of lignes
c   output:
c GG      Updated matrix of normal equations (nl1Xnl1)
ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
subroutine concoct_G(FM,nl1,nl2,MV,W2s,A2,GG)
        implicit none
        integer nl1,nl2,il,ic,j
        real*8 MV(*),W2s(*),A2(nl2,*),GG(nl1,*),dw1,dw2
        real*8 FM
        external FM
        do j=1,nl2
          dw1=FM(j,MV)*W2s(j)
          do ic=1,nl1
            dw2=dw1*A2(j,ic)
            do il=ic,nl1
                GG(il,ic)=GG(il,ic)+A2(j,il)*dw2
             enddo
          enddo
        enddo
        do ic=2,nl1
          do il=1,ic-1
            GG(il,ic)=GG(ic,il)
          enddo
        enddo
        return
        end
```

```
ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
c       subroutine cpt_dat
c               V. Lesur 09/02/2005
c       recusive subroutine, should be compiled using F90
c       previously fmxyz.f.
c       Forward modelling for a single data value
c       Output also one row of the condition matrix
c  Input:
c       nub             base number
c       nb              Number of bases
c       BS              Base subroutine to use (real*8 external function)
c       BC Base coeff such that data= SUM{i} BC(i)*BS(i)
c       BP              Base parameters for spatial and temporal positions
c  Output:
c DATA DATA value = SUM{i} BC(i)*BS(i)
c       BV              Vector of Base elements
ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
recursive subroutine cpt_dat(nub,nb,BS,BC,BP,DATA,BV)
        implicit none
        integer :: nub,nb,ib
        real*8 :: BC(*),BP(*),DATA,BV(*)
        external BS
        call BS('f',nub,nb,BC,BP,BV)
        data=0.0d0
        do ib=1,nb
          data=data+BV(ib)
        enddo
        return
        end
ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
c subroutine cpt_dat_val
c V. Lesur 09/02/2005
c       recusive subroutine, should be compiled using F90
c       derived from fmxyz.f.
c Forward modelling for a single data value
c  Input:
c       nub             base number
c       nb              Number of bases
c       BS              Base subroutine to use (real*8 external function)
```

```
c       BC         Base coeff such that data= SUM{i} BC(i)*BS(i)
c       BP              Base parameters for spatial and temporal positions
c  Output:
c DATA DATA value = SUM{i} BC(i)*BS(i)
ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
recursive subroutine cpt_dat_val(nub,nb,BS,BC,BP,DATA)
        implicit none
        integer :: nub,nb,ib
        real*8 :: BC(*),BP(*),DATA
        real*8, allocatable :: BV(:)
        external BS
        allocate(BV(1:nb))
        call BS('f',nub,nb,BC,BP,BV)
        data=0.0d0
        do ib=1,nb
          data=data+BV(ib)
        enddo
        deallocate(BV)
        return
        end
ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
c subroutine cpt_dat_vals
c Vincent Lesur 09/02/2005
c       calling recusive subroutine, should be compiled using F90
c       previously called mkxyzf.
c       for a given set of basis coefficients BC and basis functions
c       calculate the mainfield X,Y,Z or F components (depending on nt value)
c       for a set of data point in a nd dimmensional space
c       Called: cpt_dat
c       input:
c         nd         Space dimension
c         npm            maximum number of data points
c         np             number of data points
c   ppos(nd+1,*) point position in ndD
c         nt(*)          basis number for each point
c         nb Number of base functions
c         BC             base coefficients
c         BS             Base Subroutine to use
c output:
```

```
c         XYZF(*) X,Y,Z or F value at point position
cccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
subroutine cpt_dat_vals(nd,npm,np,nt,ppos,nb,BC,BS,xyzf)
        implicit none
        integer :: i,nd,npm,np,nb,nt(*)
        real*8 :: ppos(nd+1,*),bc(*),XYZf(*)
        real*8, allocatable :: dw(:)
        external BS
        allocate(dw(1:nb))
        do i=1,np
          call cpt_dat(nt(i),nb,BS,BC,ppos(1,i),xyzf(i),dw)
        enddo
        deallocate(dw)
        return
        end
cccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
c subroutine cptnormeq
c Vincent Lesur  10/02/2005
c       New version, should be used with F90
c       Compute the normal equations of the problem
c       calling recursive subroutines
c       input:
c          npmax          number max of data point with correlated errors
c          npm            Maximum total number of data points
c          npt            total number of data points
c          ipg            where to start in data file!
c          nd             space dimension
c          ppos           data point position in ndD
c          nb             Number or base function to use
c          FM             misfit function (like l2_norm.f)
c          BS             the "Base functions" subroutine to use
c          bc             Estimation of Base function coefficients
c          icov/jcov      integer arrays describing cov format
c          cov            Covariance matrix in SLAP column format
c          ddat           data vector
c          nt             vector indicating data type
c          xyzf           result of forward modelling
c       output:
c          gg             Matrice of Normal equations (nb,nb)
```

```
c          bb             SCDM of Normal equations    (nb)
cccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
        subroutine cptnormeq(npmax,npm,npt,ipg,nd,ppos,nb,FM,BS,bc
     >                       ,icov,jcov,cov,ddat,nt,xyzf,gg,bb)
        implicit none
        integer ip,npt,np,npmax,npm,nd,nb,icov(*),jcov(*)
        integer id,i,j,ipg,ipl,nt(*)
        real*8 ddat(*),xyzf(*),cov(*),ppos(nd+1,*),bc(*)
        real*8 gg(*),bb(*)
        real*8, allocatable :: dwgh(:),drot(:),vmf(:)
        real*8, allocatable :: ddif(:),dw(:),aa(:)
        real*8 FM
        external FM,BS
        allocate(dwgh(1:npmax),drot(1:npmax*npmax),vmf(1:npmax))
        allocate(ddif(1:npmax),dw(npmax),aa(1:npmax*nb))
         do i=1,nb
          do j=1,nb
            gg(j+(i-1)*nb)=0.0d0
          enddo
          bb(i)=0.0d0
        enddo
        ip=1
        do while (ip.le.npt)
          ipl=ipg+ip-1
c  calculate np and id values  (id=0 => diagonal cov)
          np=min0(npt-ip+1,npmax-2)
          if(np.gt.npmax)then
            write(*,*)'WARNING: Cannot fully expend COV',ipg,ipl
            np=npmax
          endif
         call cptnp(ipl,icov,jcov,np)
c Calculate the equations of condition
          call mkArows(np,npm,nt(ipl),nd,nb,ppos(1,ipl),bs,bc,aa)
c  calculate the delta data
          do i=1,np
            ddif(i)=ddat(ipl+i-1)-xyzf(ipl+i-1)
          enddo
c  Calculate the covariance matrix and id values  (id=0 => diagonal cov)
          call mkcovm(ipl,np,icov,jcov,cov,id,dwgh,drot)
```

```
c  Apply rotation on on AA and  dd
          if(id.ne.0)then
            call AtB(np,np,nb,drot,aa,dw)
            call AtB(np,np,1,drot,ddif,dw)
          endif
c Calculate msft vector
          do i=1,np
            vmf(i)=ddif(i)*dsqrt(dwgh(i))
          enddo
c Update the G matrix and B vector
          call concoct_G(FM,nb,np,vmf,dwgh,aa,gg)
          call concoct_B(FM,nb,np,vmf,dwgh,aa,ddif,bb)
          ip=ip+np
        enddo
        deallocate(ddif,dw,aa)
        deallocate(dwgh,drot,vmf)
        return
        end
cccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
c subroutine cptnp
c V. Lesur 20/02/2002
c
c        modified 23/02/04: GOTO removed
c
c       calculate the number of datapoint that should be treated
c       at the same time to build the normal equation
c
C       =================== S L A P Column format ==================
C
C       This routine  requires that  the matrix A  be stored in  the
C       SLAP Column format.  In this format the non-zeros are stored
C       counting down columns (except for  the diagonal entry, which
C       must appear first in each  "column")  and are stored  in the
C       real array A.  In other words, for each column in the matrix
C       put the diagonal entry in A.  Then put in the other non-zero
C       elements going down   the  column (except  the diagonal)  in
C       order.  The IA array holds the row  index for each non-zero.
C       The JA array holds the offsets into the IA, A arrays for the
C       beginning of   each    column.    That  is,    IA(JA(ICOL)),
```

```
C       A(JA(ICOL)) points to the beginning of the ICOL-th column in
C       IA and  A.  IA(JA(ICOL+1)-1),  A(JA(ICOL+1)-1) points to the
C       end  of   the ICOL-th  column.  Note   that  we  always have
C       JA(N+1) = NELT+1, where  N  is the number of columns in  the
C       matrix and  NELT   is the number of non-zeros in the matrix.
C
C       Here is an example of the  SLAP Column  storage format for a
C       5x5 Matrix (in the A and IA arrays '|'  denotes the end of a
C       column):
C
C           5x5 Matrix      SLAP Column format for 5x5 matrix on left.
C                              1  2  3    4  5    6  7    8    9 10 11
C       |11 12  0  0 15|   A: 11 21 51 | 22 12 | 33 53 | 44 | 55 15 35
C       |21 22  0  0  0|  IA:  1  2  5 |  2  1 |  3  5 |  4 |  5  1  3
C       | 0  0 33  0 35|  JA:  1  4  6    8  9   12
C       | 0  0  0 44  0|
C       |51  0 53  0 55|
C       =========== End S L A P Column format description ===========
c
c       input:
c       icov/jcov integer array describing the covariance matrix
c       ic              present working dolumn
c       np              minimum number of points in this iteration
c
c       output:
c       np              number of data point in this itteration
cccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
          subroutine cptnp(ic,icov,jcov,np)
          implicit none
          integer ic,icov(*),jcov(*),np,icm,i
          icm=ic+np-1
          i=ic-1
          do while (i.lt.icm)
           i=i+1
           icm=max0(icov(jcov(i+1)-1),icm)
          enddo
          np=icm-ic+1
          return
          end
```

```
ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
c subroutine cptstd
c Vincent Lesur  27/02/2002
c
c       modified 23/02/04: GOTO removed
c       modified 09/02/05: remove work array, includes mkcovm
c                          Should be compile using F90
c
c       Compute the STD
c
c       input:
c          npmax          number max of data point with correlated errors
c          npt            total number of data points
c          icov/jcov      integer arrays describing cov format
c          cov            Covariance matrix in SLAP column format
c          ddat           data vector
c          xyzf           result of forward modelling
c          fstd           std function
c
c       output:
c          std            STD value
c
ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
        subroutine cptstd(npmax,npt,icov,jcov,cov,ddat,xyzf,fstd,std)
        implicit none
        integer ip,npt,np,npmax,icov(*),jcov(*),id,i
        real*8 std,ddat(*),xyzf(*),cov(*)
        real*8, allocatable :: dwgh(:),drot(:),ddif(:),dw(:)
        real*8 fstd
        external fstd
        allocate(dwgh(1:npmax),drot(1:npmax*npmax))
        allocate(ddif(1:npmax),dw(npmax))
        std=0.0d0
        ip=1
        do while (ip.le.npt)
c  calculate np
          np=min0(npt-ip+1,npmax-2)
          call cptnp(ip,icov,jcov,np)
          if(np.gt.npmax)then
```

```
c           write(*,*)'WARNING: Cannot fully expend COV',ip,np,npmax
            np=npmax
          endif
c  Calculate the covariance matrix and id values  (id=0 => diagonal cov)
          call mkcovm(ip,np,icov,jcov,cov,id,dwgh,drot)
c  calculate the delta data
          do i=1,np
            ddif(i)=ddat(ip+i-1)-xyzf(ip+i-1)
          enddo
c  Apply rotation on dd
          if(id.ne.0)then
            call AtB(np,np,1,drot,ddif,dw)
          endif
c  Calculate STD
          dw(1)=std
          std=fstd(ip-1,np,dw(1),ddif,dwgh)
          ip=ip+np
        enddo
        deallocate(dwgh,drot,ddif,dw)
        return
        end
ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
c               Subroutine damp_rien
c                       V. Lesur 27/05/2002
c             compute the damping matrix "Dt.R.D",
c               scale it by dl(1)  and add it to the normal matrix GG
c               scale ir by dl(2) and substract "Dt.R.D.bc" to BB
c             Assume a time variation of 2 years
c      input
c      output
c        GG = GG + dl(1) * Dt.R.D
c        BB = BB - dl(2) * Dt.R.D * bc
ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
         subroutine damp_rien(dl,nb,GG,BB,bc)
         integer nb
         real*8 dl(*),gg(nb,*),bc(*),BB(*)
         return
         end
cccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
```

```
c       subroutine damp_spline.f      V. Lesur 07/03/2007
c       Integration in time of the product of two spline function
c       n_th derivatives.
c       The splines function are of order "io", the number of
c       derivatives is given by "idrv". The splines are defined by
c       "nk" knots "tk", the number of splines function is nspl=tk-io.
c       The output is presented as a matrix D(nspl,nspl) where the
c       element i,j correspond to the integration between tk(io) and
c       tk(nk-io+1) of the product of the idrv_th derivatives of the
c       spline function i and j.
c       note: the matrix D is necessarily symmetric but that is not
c       accounted for in the calculus.
c       input:
c         io      integer      spline order
c         nk      integer      number of knots
c         tk(*)   real*8       array of knots positions
c         idrv    integer      order of the derivative [0:io-1]
c       output:
c         damp(*) real*8       array for integrated spline products
c                             [dim min = (nk-io)*(nk-io)]
cccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
        subroutine damp_spline(io,nk,tk,idrv,damp)
        implicit none
        integer io,nk,idrv
        real*8  tk(*),damp(*)
        integer i,j,it,ip,nspl,mpd,nip
        real*8  dd,dp,d1,d2,dw((io+1)*(io+1)),spl(io,io)
        real*8, allocatable :: xx(:),ww(:)
        nspl=nk-io
        damp(1:nspl*nspl)=0.0d0
c   Find max spline pol. deg.
        mpd=io-1-idrv
c   Number of int. points  nip>=(2*mpd+1)/2
        nip=mpd+1
        if(nip.gt.0)then
c   Find int. points and weight
        allocate (xx(1:nip),ww(1:nip))
        d1=-1.d0
        d2=1.d0
```

```
      call gauleg(d1,d2,xx,ww,nip)
c   for each spline
      do i=1,nspl
c   intervals containing the spline it=i,i+io-1
        do it=max0(i,io),min0(i+io-1,nspl)
c   sampling points in the interval for integration
          d1=(tk(it+1)-tk(it))/2.d0
          d2=(tk(it+1)+tk(it))/2.d0
          do ip=1,nip
            dp=d1*xx(ip)+d2
c   idrv derivative of all non zero splines at these points
            j=idrv+1
            call DBSPVD(tk,io,j,dp,it,io,spl,dw)
c   element of the damping matrix (column wise)
            do j=it-io+1,it
              dd=spl(i-it+io,idrv+1)*spl(j-it+io,idrv+1)
              dd=dd*ww(ip)*(tk(it+1)-tk(it))/2.d0
              damp(j+(i-1)*nspl)=damp(j+(i-1)*nspl)+dd
            enddo
          enddo
        enddo
      enddo
      deallocate (xx,ww)
      endif
      return
      end
cccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
c   degree_sincos functions
c   Calulate de cosinus and sinus of an angle given in degree
cccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
       real*8 function dsind(theta)
       implicit none
       real*8 theta,d2r
       d2r=4.d0*datan(1.d0)/180.d0
       dsind=dsin(theta*d2r)
       return
       end
cccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
       real*8 function dcosd(theta)
```

```
      implicit none
      real*8 theta,d2r
      d2r=4.d0*datan(1.d0)/180.d0
      dcosd=dcos(theta*d2r)
      return
      end
ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
      real*8 function dtand(theta)
      implicit none
      real*8 theta,d2r
      d2r=4.d0*datan(1.d0)/180.d0
      dtand=dtan(theta*d2r)
      return
      end
ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
      real*8 function dasind(val)
      implicit none
      real*8 val,d2r
      d2r=4.d0*datan(1.d0)/180.d0
      dasind=dasin(val)/d2r
      return
      end
ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
      real*8 function dacosd(val)
      implicit none
      real*8 val,d2r
      d2r=4.d0*datan(1.d0)/180.d0
      dacosd=dacos(val)/d2r
      return
      end
ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
      real*8 function datand(val)
      implicit none
      real*8 val,d2r
      d2r=4.d0*datan(1.d0)/180.d0
      datand=datan(val)/d2r
      return
      end
ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
```

```
         real*8 function datan2d(val1,val2)
         implicit none
         real*8 val1,val2,d2r
         d2r=4.d0*datan(1.d0)/180.d0
         datan2d=datan2(val1,val2)/d2r
         return
         end
cccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
c   subroutine diagcov
c V. Lesur 09/02/2005
c       previously mkwgh,should be compiled using F90
c       diagonalise the covariance matrix
c       and calculate the weight matrix : WGH
c       Correlation coefficients less than dlim*variance are set to 0
c       Variances larger than min(variance)/dlim set to infinity
c       input:
c       id      indicator of known diagonal matrix (0 => diag; 1=> non diag)
c np Number of data points
c       cm      npXnp covariance matrix (Symetrique real)
c       output:
c       wgh     weight matrix (inverse of eigenvalues of cm)
c       cm      eigen vector of covariance matrix
c       id      may be corrected to 0
cccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
        subroutine diagcov(id,np,cm,wgh)
        implicit none
        integer id,np,il,ic
        real*8 cm(np,*),wgh(*),dlim,wmin
        real*8, allocatable :: dw(:)
        dlim=1.d-8
c Check if cm diag matrix
        if(id.eq.1)then
          il=0
          do while (il.lt.np.and.id.eq.1)
            il=il+1
            ic=il
            do while (ic.lt.np.and.id.eq.1)
              ic=ic+1
              if(cm(il,ic).ne.0.0d0)then
```

```
                if(dabs(cm(il,ic)).gt.cm(il,il)*dlim
     >         .or.dabs(cm(il,ic)).gt.cm(ic,ic)*dlim)then
                  id=0
                else
                  cm(il,ic)=0.0d0
                endif
              endif
            enddo
          enddo
          id=1-id
        endif
c  Case of diagonal matrix
        if(id.eq.0)then
          do il=1,np
            if(cm(il,il).gt.0.0d0)then
              wgh(il)=1.d0/cm(il,il)
            else
              wgh(il)=0.0d0
            endif
          enddo
c  Case of NON diagonal matrix
        else
c Calculate eigen values and vectors
          allocate(dw(1:np))
          call eigenvv(np,cm,wgh,dw)
          deallocate(dw)
c Inverse eigen values
          wmin=wgh(1)
          do il=2,np
            wmin=dmin1(wmin,wgh(il))
          enddo
          wmin=wmin/dlim
          do il=1,np
            if(wgh(il).gt.wmin)then
              wgh(il)=0.0d0
            else
              wgh(il)=1.d0/wgh(il)
            endif
          enddo
```

```
        endif
        return
        end
cccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
c subroutine displayeigv
c Vincent Lesur  09/10/2003
c       Display eigenvalues only of the normal matrix
c       input:
c               nb            Number or base function to use
c               BB(*)         Eigenvalues of the last normal matrix
cccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
        subroutine displayeigv(nom,nb,bb)
        implicit none
        real*8  BB(*)
        real*8, allocatable :: dw(:)
        integer i,ned,nb,nv
        character nom*100
        allocate (dw(1:nb))
        do i=1,nb
          dw(i)=BB(i)
        enddo
        call dsort(dw,BB,nb,1)
c  Save iegenvalues
        open(10,file=nom)
          do i=1,nb
            write(10,'(i5,2e15.7)')i,BB(i),dw(i)
          enddo
        close(10)
        deallocate (dw)
        return
        end
cccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
c subroutine dy2mjd
c       V. Lesur 16/06/06
c tranform a date given in decimal year in Julian days from 00:00
c on 01/01/2000
c subroutine called: IDAYSINYR
c input:
c        dy          decimal year (real*8)
```

```
c output:
c        djd         julian day (real*8)
ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
        subroutine dy2mjd(dy,djd)
        implicit none
        real*8 dy,djd
        integer iy,nd,i,IDAYSINYR
        real*8 ddiy
c year and day number in year
        iy=idint(dy)
        ddiy=dble(IDAYSINYR(iy))
c number of days from 01/01/2000 at 00:00 to 01/01/iy at 00:00
        nd=0
        if(iy.ge.2000)then
          do i=2000,iy-1
            nd=nd+IDAYSINYR(i)
          enddo
        else
          do i=iy,1999
             nd=nd+IDAYSINYR(i)
          enddo
          nd=-nd
        endif
c add days, hours, minutes and seconds
        djd=dble(nd)+(dy-dble(iy))*ddiy
        return
        end
cccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
c subroutine eig_solve
c Vincent Lesur 06/06/01
c   resoud un system lineaire Ax=b  ou A est symetrique et definie positive
c   (i.e. Toutes les valeurs propres sont reelles positives)
c   A est factorisee en A= V.D.V' et x calule par x= V.inv(D).V'.b
c   Regularisation:
c     les valeurs propres < (valeur propre max)/condition number  sont posees = 0
c     les valeurs propres < 10 * min(0.0d0,valeur propre min) sont posees = 0
c       (rem: valeurs propres peuvent etre negatives par instabilites numeriques)
c   input:
c       nbp Number of unknown and leading dim of A
```

```
c dd(nbp,*) Matrix A
c db(nbp) scdm
c       cn              Largest condition number acceptable
c   output:
c dd(nbp,*) Eigenvectors of A (column wise)
c       de(nbp) Eigenvalues of A
c       ds(nbp) Vector solution
ccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
subroutine eig_solve2(nbp,cn,dd,de,db,ds)
        implicit none
        integer nbp,i,j,icpt
        real*8 dd(nbp,*),de(*),db(*),ds(*),dw1,dw2,cn
c     calculate eigen-values and vectors (ds used as work array)
        call eigenvv(nbp,dd,de,ds)
c     calculate cut-off value
        dw1=0.d0
        dw2=1.d30
        do i=1,nbp
          dw1=dmax1(dw1,de(i))
          dw2=dmin1(dw2,de(i))
        enddo
        write(*,'(A,e10.3)')'Largest eigenvalue       = ',dw1
        write(*,'(A,e10.3)')'Smallest eigenvalue      = ',dw2
        dw2=dmax1(dw1/cn,dw2)
        write(*,'(A,e10.3)')'Condition number         = ',dw1/dw2
c       dw2=dmin1(0.0d0,dw2)
c       dw2=dmax1(dw1/cn,-10.d0*dw2)
        dw2=dmax1(dw1/cn,0.d0)
        write(*,'(A,e10.3)')'Smallest eigenvalue used = ',dw2
c    sol=V*(1/L)*Vt.At.W.b
        do j=1,nbp
          ds(j)=0.0d0
        enddo
        icpt=0
        do i=1,nbp
          dw1=0.0d0
          if(de(i).ge.dw2)then
            icpt=icpt+1
            do j=1,nbp
```

```
              dw1=dw1+dd(j,i)*db(j)
            enddo
            dw1=dw1/de(i)
            do j=1,nbp
              ds(j)=ds(j)+dd(j,i)*dw1
            enddo
          endif
        enddo
        write(*,*)
        write(*,*)'Total number of eigenvalues: ',nbp
        write(*,*)'Number of used eigenvalues : ',icpt
c
        return
        end
cccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
c subroutine eigenvv Vincent Lesur 11/04/2001
c       calcul les valeur-propres et vecteur propres
c       d'une matrice reelle SYMETRIQUE
c       algorithm de GOLUB avec shift implicite
c       la matrice est tri-diagonalisee en utilisant des rotations
c       de GIVENS avec calculs de racines carrees (tres grande stabilite)
c       L'algorithm de GOLUB utilise uniquement des rotations de GIVENS
c    => CE CODE N'EST PAS OPTIMUM EN VITESSE MAIS EST TRES STABLE
c     subroutines appelees: trig.f mkr.f
c     entree:
c       da matrice symetrique reelle
c       nn taille de la matrice
c       dw tab de travail (dim min=nn)
c     sortie:
c       dval valeur propres de la matrice
c       da   vecteur propres
c     pour le calcul des valeur propres uniquement voir eigenv.f
cccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
subroutine eigenvv(nn,da,dval,dw)
        implicit none
        real*8 da(*),dval(*),dw(*),dvar,alpha,beta,gama,dlda
        integer i,j,len,nn,iter,itermax,k,li,l,lj
        character yon
        itermax=30*nn
```

```
c    reduction tridiagonale
        call trig(da,nn)
c    sauvegarde de la diagonale et sub-diagonale
        do i=1,nn-1
          dval(i)=da(i+(i-1)*nn)
          dw(i)=da(1+i+(i-1)*nn)
        enddo
        dval(nn)=da(nn*nn)
        dw(nn)=0.d0
c    calcul de la matrice de rotation R (dans da)
        call mkr(nn,da)
c    Main
        iter=1
        do j=1,nn-1
          len=9999
          do while (len.gt.1.and.iter.le.itermax)
            i=j
            yon='y'
            do while (yon.eq.'y')
              alpha=dabs(dval(i))+dabs(dval(i+1))
              if (alpha+dabs(dw(i)).eq.alpha)then
                dw(i)=0.d0
                yon='n'
              else
                i=i+1
                if(i.gt.nn-1)yon='n'
              endif
            enddo
            len=i-j+1
            if(len.gt.1) then
c   Set the off diag element J to ZERO
              k=j+1
              dlda=(dval(i-1)-dval(i))/2.d0
              dlda=dlda/dw(i-1)
              dlda=dw(i-1)*(dlda-dsign(dsqrt(dlda**2+1.d0),dlda))
              dlda=dval(i)+dlda
              dvar=dsqrt((dval(j)-dlda)**2+(dw(j))**2)
              alpha=(dval(j)-dlda)/dvar
              beta=dw(j)/dvar
```

```
            dvar=beta*(dval(j))-alpha*dw(j)
            dval(j)=alpha*(dval(j))+beta*dw(j)
            gama=alpha*dw(j)+beta*(dval(k))
            dval(k)=beta*dw(j)-alpha*(dval(k))
            dw(j)=beta*dval(j)-alpha*gama
            dval(j)=alpha*dval(j)+beta*gama
            dval(k)=beta*dvar-alpha*dval(k)
            dw(i)=beta*dw(k)
            dw(k)=-alpha*dw(k)
c   Update eigen-vectors
            do l=1,nn
              lj=l+(j-1)*nn
              li=lj+nn
              gama=alpha*da(lj)+beta*da(li)
              da(li)=beta*da(lj)-alpha*da(li)
              da(lj)=gama
            enddo
c   Restore tri-diag form
            do k=j,i-2
              dvar=dsqrt(dw(k)**2+dw(i)**2)
              alpha=dw(k)/dvar
              beta=dw(i)/dvar
              dw(k)=dvar
              dvar=beta*dval(k+1)-alpha*dw(k+1)
              dval(k+1)=alpha*dval(k+1)+beta*dw(k+1)
              gama=alpha*dw(k+1)+beta*dval(k+2)
              dval(k+2)=beta*dw(k+1)-alpha*dval(k+2)
              dw(k+1)=gama
              dval(k+1)=alpha*dval(k+1)+beta*dw(k+1)
              dw(k+1)=alpha*dvar+beta*dval(k+2)
              dval(k+2)=beta*dvar-alpha*dval(k+2)
              dw(i)=beta*dw(k+2)
              dw(k+2)=-alpha*dw(k+2)
c   Update eigen-vectors
              do l=1,nn
                lj=l+k*nn
                li=lj+nn
                gama=alpha*da(lj)+beta*da(li)
                da(li)=beta*da(lj)-alpha*da(li)
```

```
                da(lj)=gama
              enddo
            enddo
            dw(i)=0.0d0
            iter=iter+1
          endif
        enddo
      enddo
return
    end
cccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
c subroutine expcovm
c V. Lesur 09/02/2005
c   modified:
c      25/08/2005 id itroduced in parameter list (vbfl)
c       previously subroutine mkcm
c       expend the covariance matrix associated with np points.
c       Matrix elements cm(i,j) where (i or j > np) are set to 0.
c       The LOWER triangle of the covariance matrix (including digonal
c       is input in the SLAP COLUMN Format describe below:
C       ================== S L A P Column format ==================
C       This routine  requires that  the matrix A  be stored in  the
C       SLAP Column format.  In this format the non-zeros are stored
C       counting down columns (except for  the diagonal entry, which
C       must appear first in each  "column")  and are stored  in the
C       real array A.  In other words, for each column in the matrix
C       put the diagonal entry in A.  Then put in the other non-zero
C       elements going down   the  column (except  the diagonal)  in
C       order.  The IA array holds the row  index for each non-zero.
C       The JA array holds the offsets into the IA, A arrays for the
C       beginning of   each    column.    That  is,    IA(JA(ICOL)),
C       A(JA(ICOL)) points to the beginning of the ICOL-th column in
C       IA and  A.  IA(JA(ICOL+1)-1),  A(JA(ICOL+1)-1) points to the
C       end  of   the ICOL-th  column.  Note   that  we  always have
C       JA(N+1) = NELT+1, where  N  is the number of columns in  the
C       matrix and  NELT   is the number of non-zeros in the matrix.
C       Here is an example of the  SLAP Column  storage format for a
C       5x5 Matrix (in the A and IA arrays '|'  denotes the end of a
C       column):
```

```
C           5x5 Matrix      SLAP Column format for 5x5 matrix on left.
C                              1  2  3    4  5    6  7    8    9 10 11
C       |11 12  0  0 15|   A: 11 21 51 | 22 12 | 33 53 | 44 | 55 15 35
C       |21 22  0  0  0|  IA:  1  2  5 |  2  1 |  3  5 |  4 |  5  1  3
C       | 0  0 33  0 35|  JA:  1  4  6    8  9   12
C       | 0  0  0 44  0|
C       |51  0 53  0 55|
C       =========== End S L A P Column format description ===========
c       input:
c         data(np)        data values
c         cov(*)          covariance matrix in SLAP column format
c         icov/jcov       integer array describing COV matrix
c         np              Number of point in data
c         ic              present working column
c         id              id=0 for diag cov matrix =1 otherwise
c       output:
c         cm              covariance matrix for these np data
c         id              id=0 for diag cov matrix =1 otherwise
cccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
  subroutine expcovm(id,ic,np,icov,jcov,cov,cm)
          implicit none
          integer id,ic,icov(*),jcov(*),np,i,j,k,l
          real*8 cov(*),cm(np,*)
c Initialisation
         do i=1,np
           do j=1,np
             cm(i,j)=0.0d0
           enddo
         enddo
c building
         id=0
         do i=1,np
           k=ic+i-1
           cm(i,i)=cov(jcov(k))
           do j=jcov(k)+1,jcov(k+1)-1
             l=icov(j)-ic+1
             if(l.le.np)then
               id=1
               cm(i,l)=cov(j)
```

```
            cm(l,i)=cov(j)
          endif
        enddo
      enddo
      return
      end
ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
c subroutine formalstd
c Vincent Lesur  17/01/2003
c       Computes the formal std values of an inverse problem
c       given the GG matrix of the inverse problem.
c       The GG matrix may have been built using pbi.f
c       input:
c               nb              Number or base function to use
c               dl              max condition number for inverse problem
c               GG/EE           eigenvector and values of the inverse problem
c               stdt            weighted misfit to the data
c       output:
c               var(*)          array of formal std for parameters
ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
        subroutine formalstd(nb,dl,EE,GG,stdt,var)
        implicit none
        integer nb,i,j
        real*8  EE(*),GG(nb,*),var(*),eemin,dl,stdt
c  Compute the inverse problem matrix from GG & EE
        eemin=0.d0
        do i=1,nb
          eemin=dmax1(eemin,EE(i))
        enddo
        eemin=dmax1(eemin/dl,0.d0)
        do i=1,nb
          var(i)=0.0d0
          do j=1,nb
            if(EE(j).gt.eemin)then
             var(i)=var(i)+GG(i,j)*GG(i,j)/EE(j)
            endif
          enddo
          var(i)=stdt*dsqrt(var(i))
        enddo
```

```
        return
        end
c====================================================================
c   *** GAULEG - find zeroes and weights for Gauss Quadrature ***
c     probably a Gubbins subroutine originally
c     check 31/05/2004 up to n=96 against  Abrm&Stegun p. 919:
c                      GOTO removed
c                      error on starting value z corrected
c                      maximum number of iteration introduced
c   PARAMETERS
c        x1,x2: end points of integration
c        x/w:   arrays of zeroes of Legendre function of degree n
c               corresponding weights.
c        n:     degree of integration
c====================================================================
      subroutine gauleg(x1,x2,x,w,n)
      implicit real*4 (a-c,e-h,o-z)
      implicit real*8 (d)
      implicit integer*4 (i-n)
      parameter (itmax=10)
      real*8 x(*),w(*),dpi,deps
     +       ,x1,x2,xm,xl,p1,p2,p3,pp,z,z1
      deps=1.d-15
      dpi=4.d0*datan(1.d0)
      m=int((n+1)/2.)
      x(m+1)=0.0d0
      xm=0.5d0*(x2+x1)
      xl=0.5d0*(x2-x1)
      do i=1,m
c Search x value by gradient methods
        z=dcos(dpi*dble(2*i)/dble(2*n+1))
        z1=dcos(dpi*dble(2*i-1)/dble(2*n+1))
        it=0
        do while (dabs(z-z1).gt.deps.and.it.le.itmax)
          it=it+1
c Recurence on Pl(cos(z))
          p1=1.d0
          p2=0.0d0
          do j=1,n
```

```
            p3=p2
            p2=p1
            p1=((2.d0*j-1.d0)*z*p2-(dble(j)-1.d0)*p3)/dble(j)
          enddo
c Calculate derivative relative to cos(z)
          pp=n*(z*p1-p2)/(z*z-1.d0)
c Newton's iteration
          z1=z
          z=z1-p1/pp
        enddo
        if(it.ge.itmax)write(*,*)'Warning: convergence for zero:',i
c Found saved + weight calculated (twice by symmetry)
        x(i)=xm-xl*z
        x(n+1-i)=xm+xl*z
        w(i)=xl*2.d0/((1.d0-z*z)*pp*pp)
        w(n+1-i)=w(i)
      enddo
9999  return
      end
      subroutine gdtogc (cltgd,alt,cltgc,r,cd,sd)
cccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
c     FUNCTION - conversion of position from geodetic, i.e. spheroidal,
c                to geocentric, i.e. spherical, coordinates using WGS84
c                spheroid
c     First author: unknown
c     21/01/2003-Modified to better meet FORTRAN std. V. Lesur
c     ARGUMENTS -
c     Name         Type  I/O  Description
c     ----         ----  ---  -----------
c     cltgd        dble   I   geodetic colatitude (deg)
c     alt          dble   I   altitude above sea-level (km)
c     cltgc        dble   O   geocentic colatitude (deg)
c     r            dble   O   radius from centre of earth (km)
c     cd           dble   O   cos(difference in angle between geodetic
c                             and geocentric latitude)
c     sd           dble   O   sin(difference in angle between geodetic
c                             and geocentric latitude)
cccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
      implicit none
```

```
      real*8 cltgd,alt,cltgc,r,cd,sd
      real*8 ct,st,a2,b2,one,two,three,rho
      real*8 dtor
      dtor=4.d0*datan(1.d0)/180.d0
      one   = cltgd*dtor
      ct    = dcos(one)
      st    = dsin(one)
      a2    = 40680631.6d0
      b2    = 40408296.0d0
      one   = a2*st*st
      two   = b2*ct*ct
      three = one + two
      rho   = dsqrt(three)
      r     = dsqrt(alt*(alt + 2.0d0*rho) + (a2*one + b2*two)/three)
      cd    = (alt + rho)/r
      sd    = (a2 - b2)/rho*ct*st/r
      one   = ct
      ct    = ct*cd -  st*sd
      st    = st*cd + one*sd
      cltgc = datan2(st,ct)/dtor
      return
      end
cccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
c subroutine GEO_SMm
c       V. Lesur 21/08/06
c Transform a vector in SM system of coordinates from a vector in
c GEO system of coordinates.
c input:
c         vx,vy,vx    (REAL*8)  vector coordinates in GEO
c         djd         (REAL*8)  time in days from 00:00 UT the 01/01/2000
c output:
c         vx,vy,vx    (REAL*8)  vector coordinates in SM
cccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
        subroutine GEO_SMm(djd,vx,vy,vz)
        implicit none
        real*8 djd,vx,vy,vz
        real*8 dt0,ut,dth,eps,dm,dl,dsl
        real*8 g10,g11,h11,dd,psi,d0,d5,dmu,d2r
        d2r=4.d0*datan(1.d0)/180.d0
```

```
c   Find the greenwich mean sideral time
        dt0=dint(djd)
        ut=(djd-dt0)*24.d0
        dt0=(dt0-0.5d0)/36525.0d0
        dth=100.461d0+36000.77d0*dt0+15.04107d0*ut
c   Find the obliquity of the ecliptic
        eps=23.439d0-0.013d0*dt0
        eps=-eps
c   Find Sun's ecliptic longitude
        dm=357.528d0+35999.050d0*dt0+0.04107d0*ut
        dl=280.460d0+36000.772d0*dt0+0.04107d0*ut
        dsl=dl+(1.915d0-0.0048d0*dt0)*dsin(dm*d2r)
     >        +0.02d0*dsin(2.d0*dm*d2r)
        dsl=-dsl
c   Dipole coeff in GEO for djd
        d0=(1827.0d0-djd)/1827.d0
        d5=djd/1827.d0
        g10=-(29556.8d0*d5+29619.4d0*d0)
        g11=-(1671.8d0*d5+1728.2d0*d0)
        h11=5080.0d0*d5+5186.1d0*d0
c   Normalised & angles
        dd=g10**2
        dd=dd+g11**2
        dd=dd+h11**2
        dd=dsqrt(dd)
        g10=g10/dd !z0
        g11=g11/dd !x0
        h11=h11/dd !y0
        call ortho_rot(3,dth,g11,h11,g10)
        call ortho_rot(1,eps,g11,h11,g10)
        call ortho_rot(3,dsl,g11,h11,g10)
        psi=datan(h11/g10)/d2r
        dd=dsqrt(h11**2+g10**2)
        dmu=datan2(g11,dd)/d2r
c   Apply rotations
        call ortho_rot(3,dth,vx,vy,vz)
        call ortho_rot(1,eps,vx,vy,vz)
        call ortho_rot(3,dsl,vx,vy,vz)
        call ortho_rot(1,psi,vx,vy,vz)
```

```
        call ortho_rot(2,dmu,vx,vy,vz)
        return
        end
cccccccccccccccccccccccccccccccccccccccccccccccccccccc
c Function l1_norm
c                  V. Lesur
c       Build from the misfit vector MV compute the misfit function
c       for the probability density function associated the L1-norm
c     input:
c mv misfit vector
c       i Element of the misfit vector to use
c     output:  misfit function value
cccccccccccccccccccccccccccccccccccccccccccccccccccccccc
real*8 function l1_norm(i,mv)
        implicit real*8 (a-h,o-z)
        implicit integer*4 (i-n)
        integer i
        real*8 mv(*),dnu,amv
        dnu=1.d-5
        amv=dabs(mv(i))
        if(amv.lt.dnu)amv=dnu
        l1_norm=dsqrt(2.d0)/amv
        return
        end
cccccccccccccccccccccccccccccccccccccccccccccccccccccc
c Function l1_std
c                  V. Lesur 20/02/2002
c       Compute the std assoiciated with an L1 norme
c       for np points added to a total of npt points
c     input:
c       npt     total number of point used to compute std
c       std     values of std computed with npt points
c       np      number of point added in this itteration
c mv misfit vector dimmin np
c       wgh vector of 1/variances dimmin np
c     output:
cccccccccccccccccccccccccccccccccccccccccccccccccccccccc
real*8 function l1_std(npt,np,std,mv,wgh)
        implicit real*8 (a-h,o-z)
```

```
        implicit integer*4 (i-n)
        integer npt,np,i
        real*8 mv(*),std,wgh(*),sr2
        sr2=dsqrt(2.d0)
        std=std*npt/sr2
        do i=1,np
          std=std+dabs(mv(i))*dsqrt(wgh(i))
        enddo
        l1_std=sr2*std/(npt+np)
        return
        end
cccccccccccccccccccccccccccccccccccccccccccccccccccc
c Function l2_norm
c                 V. Lesur
c       Build from the misfit vector MV compute the misfit function
c       for the probability density function associated the L2-norm
c     input:
c   mv misfit vector
c     i Element of the misfit vector to use
c     output:  misfit function value
ccccccccccccccccccccccccccccccccccccccccccccccccccccccc
    real*8 function l2_norm(i,mv)
        implicit none
        integer i
        real*8 mv(*)
        l2_norm=1.d0
        return
        end
cccccccccccccccccccccccccccccccccccccccccccccccccccc
c Function l2_std
c                 V. Lesur 20/02/2002
c       Compute the std assoiciated with an L2 norm
c       for np points added to a total of npt points
c     input:
c       npt     total number of point used to compute std
c       std     values of std computed with npt points
c       np      number of point added in this itteration
c mv misfit vector dimmin np
c       wgh vector of 1/variances dimmin np
```

```
c     output:
c       std     modified
cccccccccccccccccccccccccccccccccccccccccccccccccccccccc
real*8 function l2_std(npt,np,std,mv,wgh)
        implicit none
        integer npt,np,i
        real*8 mv(*),std,wgh(*)
        std=dble(npt)*std**2
        do i=1,np
          std=std+wgh(i)*mv(i)**2
        enddo
        l2_std=dsqrt(std/dble(npt+np))
        return
        end
cccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
C**** Subroutines in misc.f (copied from MISCSUBS.FOR (VAX), D.J.Kerridge)
C     List last revised 25 January 1999 : E.Clarke
C**** Y2K initial check : 25th January 1999, E.Clarke
C**** Y2K final check   :
C     SUBROUTINE DMS(JD,JM,JS,DEG,IC)
C     SUBROUTINE DMSP(JD1,JM1,JS1,JD2,JM2,JS2,JD3,JM3,JS3,DEG3)
C     SUBROUTINE DMSM(JD1,JM1,JS1,JD2,JM2,JS2,JD3,JM3,JS3,DEG3)
C     SUBROUTINE DAMON(ID,IM,IY,ND,MODE)
C     SUBROUTINE ETRAP(ECODE,ID,IM,IY,ND,MODE)
C     SUBROUTINE DAYSIN(MON,IYEAR,NDAYS)
C     SUBROUTINE DAYSINMON(IY,NDM)
C     FUNCTION   IDAYSINYR(IY)
C     SUBROUTINE DOW(ID,M,IY,DAY)
C     SUBROUTINE MNVAR(ARR,NPTS,FLAG,RMN,VAR,NMISS)
C     SUBROUTINE MNMX(ARR,NPTS,FLAG,RMIN,RMAX)
C     SUBROUTINE IDIVREM(NUM,IDIV,IQUO,IREM)
C     SUBROUTINE GETLEN(STRING,NCHAR)
C     SUBROUTINE CHCASE(STRING,CASE)
C     SUBROUTINE RMEAN(ARR,AFIL,NARR,RMISS,FILT,NFILT)
C     SUBROUTINE BROT(IDAY,MON,IYR,IROT,IDNO)
C     FUNCTION ROUND(X1,X2)
C     FUNCTION IROUND(I1,I2)
C     SUBROUTINE FDMON(IY,IFDM)
C     SUBROUTINE AUTOC(X,NX,LAG,ACF,XVR)
```

```
C     SUBROUTINE LASTMON(ID,IM,IY,ND,NDYR)
C     SUBROUTINE THISMON(ID,IM,IY,ND,NDYR)
C     SUBROUTINE DATERD(ID,IM,IY,ND,NDYR)
c     subroutine tdaterd(id,im,iy,nd,ndyr)
c     subroutine stat(x,n,flag,xmin,xmax,xrms,xsd,nmiss)
C>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
C**** Subroutine DMS to convert from degrees (JD), minutes (JM) and
C     seconds (JS) to decimal degrees (DEG) when IC = 1, and from decimal
C     degrees to degrees minutes and seconds when IC = 2.
C**** No 'allowed range' checks are made on the input data, so the input
C     can be in minutes or seconds for conversion to decimal degrees.
C     eg. JD=JS=0 and JM=420 will lead to DEG=7.0
C     The output format for negative angles is:
C     a. -12 34 56 when the degrees are non-zero
C     b.  00-34 56 when degrees = 0
C     c   00 00-56 when degrees = 0 and minutes = 0
C**** The input parameters are unchanged on exit.
C**** Last revised 5th August 1987  DJK
      SUBROUTINE DMS(JD,JM,JS,DEG,IC)
      INTEGER IC,ISIG,JD,JM,JS
      REAL DEG,DG,DR,EM,SC,SIG
      IF(IC .EQ. 2) GOTO 1
      IF(IC .NE. 1) THEN
         WRITE(6,*) 'IC should be 1 or 2, it has been passed as',IC
         STOP
      END IF
C**** IC = 1, so convert from dms to decimal degrees
      SIG = 1.0
      IF(JD .LT. 0 .OR. JM .LT. 0 .OR. JS .LT. 0) SIG = -1.0
      DG  = IABS(JD)
      EM  = IABS(JM)
      SC  = IABS(JS)
      DEG = (DG + EM/60. + SC/3600.)*SIG
      RETURN
C**** IC = 2, so convert from decimal degrees to dms
    1 SIG = SIGN(1.1,DEG)
      DR  = ABS(DEG)
      JD  = INT(DR)
      DG  = JD
```

```
      JM  = INT(60.*(DR-DG))
      EM  = JM
      JS  = INT((DR - DG - EM/60.)*3600. + .5)
      IF(JS .EQ. 60) THEN
         JS = 0
         JM = JM + 1
      END IF
      IF(JM .EQ. 60) THEN
         JM = 0
         JD = JD+1
      END IF
CC**** Assign the correct sign to the result
      ISIG = INT(SIG)
      IF(JD .NE. 0) THEN
         JD = JD*ISIG
      ELSE
            IF(JM .NE. 0) THEN
               JM = JM*ISIG
            ELSE
               JS = JS*ISIG
            END IF
      END IF
      RETURN
      END
C**** Subroutine DMSP to add two angles supplied in degrees, minutes and
C     seconds.  The result, (angle3 = angle1+angle2), is returned in both
C     degrees minutes and seconds and decimal degrees.
C**** This subroutine calls subroutine DMS
C**** Last revised 22nd June 1987 : DJK
      SUBROUTINE DMSP(JD1,JM1,JS1,JD2,JM2,JS2,JD3,JM3,JS3,DEG3)
      INTEGER JD1,JM1,JS1,JD2,JM2,JS2,JD3,JM3,JS3
      REAL DEG1,DEG2,DEG3
      CALL DMS(JD1,JM1,JS1,DEG1,1)
      CALL DMS(JD2,JM2,JS2,DEG2,1)
      DEG3 = DEG1 + DEG2
      CALL DMS(JD3,JM3,JS3,DEG3,2)
      RETURN
      END
ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
```

```
C**** Subroutine DMSM to subtract two angles supplied in degrees, minutes
C     and seconds.  The result, (angle3 = angle1-angle2), is returned in
C     both degrees minutes and seconds and decimal degrees.
C     This subroutine calls subroutine DMS
C**** Last revised 22nd June 1987 : DJK
Ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
      SUBROUTINE DMSM(JD1,JM1,JS1,JD2,JM2,JS2,JD3,JM3,JS3,DEG3)
      INTEGER JD1,JM1,JS1,JD2,JM2,JS2,JD3,JM3,JS3
      REAL DEG1,DEG2,DEG3
      CALL DMS(JD1,JM1,JS1,DEG1,1)
      CALL DMS(JD2,JM2,JS2,DEG2,1)
      DEG3 = DEG1 - DEG2
      CALL DMS(JD3,JM3,JS3,DEG3,2)
      RETURN
      END
ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
C**** Subroutine DAMON to convert day (ID), month (IM), year (IY) to day
C     number when MODE=1 and to compute day and month from day number and
C     year when MODE=2.  The full year should be specified, eg. 1988 not
C     just 88.  In the Gregorian calendar, introduced in 1582 but not
C     adopted in English-speaking countries until 1752, years which are
C     are divisible by 100 are leap years only if they are also divisible
C     by 400.
C**** Last revised 11th February 1988 : DJK (adapted from PRR subroutine)
C**** Y2K initial check : 25th January 1999, E.Clarke
C**** Y2K final check   :
Cccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
      SUBROUTINE DAMON(ID,IM,IY,ND,MODE)
      INTEGER MD,ID,IM,IY,ND,MODE
      DIMENSION MD(13)
      DATA MD /0,31,59,90,120,151,181,212,243,273,304,334,365/
      IF(MODE .NE. 1 .AND. MODE .NE. 2) CALL ETRAP(1,ID,IM,IY,ND,MODE)
      IF(MODE .EQ. 1) THEN
         IF(ID .GT. 31 .OR. ID .LT. 1)  CALL ETRAP(2,ID,IM,IY,ND,MODE)
         IF(IM .GT. 12 .OR. IM .LT. 1)  CALL ETRAP(3,ID,IM,IY,ND,MODE)
        IF(IY .LT. 1582) CALL ETRAP(4,ID,IM,IY,ND,MODE)
         IF(ID .GT. MD(IM+1)-MD(IM)) THEN
            IF(IM .NE. 2 .OR. (IM .EQ. 2 .AND. ID .NE. 29))
     &         CALL ETRAP(5,ID,IM,IY,ND,MODE)
```

```
          IF((IM .EQ. 2 .AND. ID .EQ. 29) .AND. (MOD(IY,100) .EQ. 0
     &          .AND. MOD(IY,400) .NE. 0))
     &          CALL ETRAP(5,ID,IM,IY,ND,MODE)
        END IF
      ELSE
        IF(ND .GT. 366 .OR. ND .LT. 1) CALL ETRAP(6,ID,IM,IY,ND,MODE)
      END IF
      IF(MODE .EQ. 1) THEN
        ND  = MD(IM) + ID
        IF(MOD(IY,4) .EQ. 0 .AND. IM .GT. 2) THEN
          IF(MOD(IY,100) .EQ. 0 .AND. MOD(IY,400) .NE. 0) THEN
          ELSE
            ND = ND+1
          END IF
        END IF
      ELSE
        L = 0
        IF(MOD(IY,4) .EQ. 0 .AND. ND .GT. 59) THEN
          IF(MOD(IY,100) .EQ. 0 .AND. MOD(IY,400) .NE. 0) THEN
          ELSE
            L = 1
          END IF
        END IF
        NND = ND-L
        IF(NND .EQ. 366) CALL ETRAP(7,ID,IM,IY,ND,MODE)
        DO 1 IL=12,1,-1
        IM  = IL
        ID  = NND-MD(IM)
        IF(ID .GE. 1) GOTO 2
    1   CONTINUE
    2   IF(IM .EQ. 2) ID = ID+L
      END IF
      RETURN
      END
C**** Subroutine ETRAP for DAMON
      SUBROUTINE ETRAP(ECODE,ID,IM,IY,ND,MODE)
      INTEGER ECODE,MODE,ID,IM,IY,ND
      GOTO (10,20,30,40,50,60,70) ECODE
   10 WRITE(6,*) ' MODE must be either 1 or 2'
```

```
      WRITE(6,*) ' MODE =',MODE,'  has been passed to DAMON'
      STOP
   20 WRITE(6,*) 'The day of the month has been passed as',ID
      STOP
   30 WRITE(6,*) 'The month has been passed as',IM
      STOP
   40 WRITE(6,*) 'The Gregorian calendar began in 1582'
      WRITE(6,*) 'The year has been passed as',IY
      STOP
   50 WRITE(6,*) 'The day number and month are inconsistent',ID,IM
      STOP
   60 WRITE(6,*) 'The day number has been passed as',ND
      STOP
   70 WRITE(6,*) 'There is no day 366 in year',IY
      STOP
      END
Cccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
C**** Subroutine DAYSIN to return the number of days in month MON in
C   year IYEAR
C**** Y2K initial check : 25th January 1999, E.Clarke
C**** Y2K final check   :
cccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
      SUBROUTINE DAYSIN(MON,IYEAR,NDAYS)
      DIMENSION MDAYS(12)
      DATA MDAYS /31,-99,31,30,31,30,31,31,30,31,30,31/
C
      CALL DAMON(31,12,IYEAR,N,1)
      IF(N .EQ. 365) THEN
         MDAYS(2) = 28
      ELSE IF (N .EQ. 366) THEN
         MDAYS(2) = 29
      ELSE
         WRITE(6,'(1X,A)') 'Error in subroutine DAYSIN'
         STOP
      END IF
      NDAYS = MDAYS(MON)
      RETURN
      END
C>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
```

```
C**** Subroutine DAYSINMON to return the number of days in each month in
C     year IY in the array NDM.  Function IDAYSINYR is called.
C**** 26th September 1990
C**** Y2K initial check : 25th January 1999, E.Clarke
C**** Y2K final check   :
C>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
      SUBROUTINE DAYSINMON(IY,NDM)
      IMPLICIT NONE
      INTEGER NDM,IY,ND,IDAYSINYR
      DIMENSION NDM(12)
      NDM(1)  =  31
      NDM(2)  = -99
      NDM(3)  =  31
      NDM(4)  =  30
      NDM(5)  =  31
      NDM(6)  =  30
      NDM(7)  =  31
      NDM(8)  =  31
      NDM(9)  =  30
      NDM(10) =  31
      NDM(11) =  30
      NDM(12) =  31
      ND = IDAYSINYR(IY)
      IF(ND .EQ. 365) THEN
         NDM(2) = 28
      ELSE IF(ND .EQ. 366) THEN
         NDM(2) = 29
      ELSE
         WRITE(6,'(/1X,A/)') '>>> Error in subroutine DAYSINMON <<<'
         STOP
      END IF
      RETURN
      END
C>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
C**** Subroutine FDMON to return the day number for the first day in each
C     month in year IY in the array NDM.  Subroutine DAYSINMON is called.
C**** 26th September 1990
C**** Y2K initial check : 25th January 1999, E.Clarke
C**** Y2K final check   :
```

```
c>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
      SUBROUTINE FDMON(IY,IFDM)
      IMPLICIT NONE
      INTEGER IFDM,NDM,IY,IL
      DIMENSION NDM(12),IFDM(12)
      CALL DAYSINMON(IY,NDM)
      IFDM(1) = 1
      DO 10 IL = 2,12
         IFDM(IL) = IFDM(IL-1) + NDM(IL-1)
   10 CONTINUE
      RETURN
      END
C>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
C**** Subroutine DOW to compute the day of the week given the date in
C     day (ID), month (M) and year (IY) format.  The JULDAY function
C     from 'Numerical Recipes' is used.  The day is written into the
C     character variable DAY.
C     (Note: the JULDAY function can be used independently of the
C      subroutine to obtain the Julian day number.)
C**** Last revised 26th September 1988 : DJK
C**** Y2K initial check : 25th January 1999, E.Clarke
C**** Y2K final check   :
C>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
      SUBROUTINE DOW(ID,M,IY,DAY)
      INTEGER ID,IY,M,JDAY
      CHARACTER*9 DAYS(7),DAY
      DATA DAYS /'Sunday','Monday','Tuesday','Wednesday','Thursday',
     &           'Friday','Saturday'/
      JDAY = JULDAY(ID,M,IY)
      IREM = MOD(JDAY+1,7) + 1
      WRITE(DAY,'(A)') DAYS(IREM)
      RETURN
      END
      FUNCTION JULDAY(ID,MM,IYYY)
      PARAMETER (IGREG=15+31*(10+12*1582))
      IF(IYYY.EQ.0) THEN
         WRITE(6,'(1X,A)') 'There is no Year Zero'
         STOP
      END IF
```

```
      IF (IYYY.LT.0) IYYY=IYYY+1
      IF (MM.GT.2) THEN
        JY=IYYY
        JM=MM+1
      ELSE
        JY=IYYY-1
        JM=MM+13
      ENDIF
      JULDAY=INT(365.25*JY) + INT(30.6001*JM) + ID + 1720995
      IF (ID+31*(MM+12*IYYY) .GE. IGREG) THEN
        JA=INT(0.01*JY)
        JULDAY=JULDAY + 2 - JA + INT(0.25*JA)
      ENDIF
      RETURN
      END
C>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
C**** Function IDAYSINYR to return the number of days in the year IY.
C     Subroutine DAMON is used.
C**** 26th September 1990
C**** Y2K initial check : 25th January 1999, E.Clarke
C**** Y2K final check   :
      FUNCTION IDAYSINYR(IY)
      IMPLICIT NONE
      INTEGER IDAYSINYR,IY,IDAY,MON,MODE,ND
      DATA IDAY,MON,MODE /31,12,1/
      CALL DAMON(IDAY,MON,IY,ND,MODE)
      IDAYSINYR = ND
      RETURN
      END
C>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
C**** Subroutine MNVAR to calculate the mean (RMN) and variance (VAR)
C     of the NPTS elements passed to ARR(NPTS).
C     FLAG is the value used to flag missing data.
C*CC**** Last revised 12-06-1989 : DJK
C
      SUBROUTINE MNVAR(ARR,NPTS,FLAG,RMN,VAR,NMISS)
      IMPLICIT NONE
      INTEGER IL,NPTS,NST,IOUT,NMISS
      REAL ARR,DEL,RMN,VAR,FLAG,AMN,TINY
```

```
      DIMENSION ARR(NPTS)
      PARAMETER (IOUT = 6, TINY = 0.01)
      IF(NPTS .EQ. 1) THEN
         RMN =  ARR(1)
         VAR =  FLAG
      ELSE
         RMN   = 0.0
         NMISS =   0
         NST   =   0
   10    NST   = NST + 1
         IF(NST .EQ. NPTS+1) THEN
            RMN = FLAG
            VAR = FLAG
            RETURN
         END IF
         DEL   = ABS(ARR(NST) - FLAG)
         IF(DEL .LT. TINY) THEN
            NMISS = NMISS + 1
            GOTO 10
         ELSE
            AMN   = ARR(NST)
         END IF
C
         DO 20 IL = NST+1,NPTS
            DEL = ABS(ARR(IL) - FLAG)
            IF(DEL .GT. TINY) THEN
               RMN = RMN + ARR(IL)-AMN
            ELSE
               NMISS = NMISS + 1
            END IF
   20    CONTINUE
         RMN = AMN + RMN/REAL(NPTS-NMISS)
C
         VAR = 0.0
         DO 30 IL=1,NPTS
            DEL = ABS(ARR(IL) - FLAG)
            IF(DEL .GT. 0.01) VAR = VAR + (RMN-ARR(IL))*(RMN-ARR(IL))
   30    CONTINUE
         IF(NPTS-NMISS-1 .GT. 0) THEN
```

```
            VAR = VAR/REAL(NPTS-NMISS-1)
         ELSE
            VAR = FLAG
         END IF
      END IF
      RETURN
      END
C
C**** Subroutine MNMX to find the smallest (RMIN) and largest (RMAX)
C     elements in the NPTS passed into array ARR(NPTS).
C     FLAG is the value used to flag missing data.
C
C**** Last revised 27-09-1990 : DJK
C
      SUBROUTINE MNMX(ARR,NPTS,FLAG,RMIN,RMAX)
      IMPLICIT NONE
      INTEGER IL,NPTS,NDAT
      REAL ARR,DEL,RMIN,RMAX,FLAG,TINY
      DIMENSION ARR(NPTS)
      DATA TINY /0.1/
C
      RMIN = FLAG
      RMAX = FLAG
      NDAT = 0
      DO 10 IL=1,NPTS
         DEL  = ABS(ARR(IL) - FLAG)
         IF(DEL .GT. TINY) THEN
            NDAT = NDAT + 1
            IF(NDAT .EQ. 1) THEN
               RMIN = ARR(IL)
               RMAX = ARR(IL)
            ELSE
               RMIN  = MIN(RMIN,ARR(IL))
               RMAX  = MAX(RMAX,ARR(IL))
            END IF
         END IF
   10 CONTINUE
C
      RETURN
```

```
      END
C
C**** Subroutine IDIVREM to perform integer division and return a
C     remainder.  IDIV goes into NUM IQUO times with remainder IREM.
C
C**** 29th January 1988 : DJK
C
      SUBROUTINE IDIVREM(NUM,IDIV,IQUO,IREM)
      INTEGER NUM,IDIV,IQUO,IREM
C
      IQUO = NUM/IDIV
      IREM = MOD(NUM,IDIV)
C
      RETURN
      END
C
C**** Subroutine GETLEN to find the number of 'significant' characters,
C     NCHAR, in STRING.  NCHAR is found by counting back from the end
C     of STRING until a non-space character is found.
C
C**** 23rd August1988 : DJK
C
      SUBROUTINE GETLEN(STRING,NCHAR)
      INTEGER NCHAR
      CHARACTER*(*) STRING
      NCHAR = LEN(STRING) + 1
   10 NCHAR = NCHAR - 1
      IF(STRING(NCHAR:NCHAR) .NE. ' ') RETURN
      GOTO 10
      END
C
C**** Subroutine CHCASE to change the case of a character variable
C     STRING according to the specifier CASE.  Only alphabetic
C     characters are affected.  A string with a mixture of upper and
C     lower case on entry will leave with all characters as specified
C     by CASE.
C
C**** Last revised 11th August 1988 : DJK
C
```

```
      SUBROUTINE CHCASE(STRING,CASE)
      CHARACTER*(*) STRING
      CHARACTER*5 CASE
C
      IF(CASE .NE. 'UPPER' .AND. CASE .NE. 'lower') THEN
         WRITE(6,'(/1X,4A/)') 'Case conversion specifier passed to',
     &   ' CHCASE as ''',CASE,''''
         WRITE(6,'(1X,A/)') 'It must be either ''UPPER'' or ''lower'''
         STOP
      END IF
C
      NCHAR = LEN(STRING)
      IF(CASE .EQ. 'UPPER') THEN
         DO 10 IL = 1,NCHAR
            IASC  = ICHAR(STRING(IL:IL))
            IF(IASC .GE. 97 .AND. IASC .LE. 122) THEN
               IASC = IASC - 32
               STRING(IL:IL) = CHAR(IASC)
            END IF
   10       CONTINUE
      ELSE
         DO 20 IL = 1,NCHAR
            IASC  = ICHAR(STRING(IL:IL))
            IF(IASC .GE. 65 .AND. IASC .LE. 90) THEN
               IASC = IASC + 32
               STRING(IL:IL) = CHAR(IASC)
            END IF
   20       CONTINUE
      END IF
C
      RETURN
      END
C
C**** Subroutine RMEAN to filter (running mean) the data in array ARR(NARR).
C     The filter coefficients are supplied in array FILT(NFILT) and the filtered
C     data are written into array AFIL(NARR).  RMISS is the value used to show
C     missing data and this value is put into AFIL in the element at the
C     centre of the running mean.
C
```

```
C**** Last revised 12th September 1988 : DJK
C
      SUBROUTINE RMEAN(ARR,AFIL,NARR,RMISS,FILT,NFILT)
      INTEGER IL,ISW,IWT,JL,NMISS,NFILT,NST,NFN,NARR
      REAL ARR,AFIL,FILT,RM,RMISS
      DIMENSION ARR(NARR),AFIL(NARR),FILT(NFILT)
C
C**** Check for a symmetric filter
      IF(MOD(NFILT,2) .EQ. 0) THEN
         WRITE(6,'(/1X,2A)') 'The number of filter coefficients ',
     &                       'must be odd'
         WRITE(6,'(1X,A,I4/)') 'The value supplied is',NFILT
         STOP
      END IF
C
      NMISS = NFILT/2
      NST   = NMISS + 1
      NFN   = NARR  - NMISS
      DO 20 IL = 1,NARR
         RM   = 0.0
         ISW  =   0
         IWT  =   0
         IF(IL.GE.NST .AND. IL.LE.NFN) THEN
            DO 10 JL = IL-NMISS,IL+NMISS
               IWT = IWT + 1
               IF(ABS(ARR(JL)-RMISS) .LT. 0.1) ISW = 1
               RM  = RM + ARR(JL)*FILT(IWT)
   10          CONTINUE
         AFIL(IL)  = RM
C**** ISW=1 indicates that data is missing in ARR within the range of the
C     filter. The filtered value is written as RMISS.
         IF(ISW .EQ. 1) AFIL(IL) = RMISS
         ELSE
            AFIL(IL) = RMISS
         END IF
   20    CONTINUE
C
      RETURN
      END
```

```
C
C**** Subroutine BROT to calculate the Bartels rotation number and the
C     day number in the rotation for the date supplied.
C
C     Day 1 of rotation 1 is 8th February 1832.
C
C**** Last revised 13th February 1990 : DJK
C**** Corrected 6th Jan 1992 (it had done multiples of 27 incorrectly)
C
C**** Y2K initial check : 25th January 1999, E.Clarke
C**** Y2K final check   :
C
      SUBROUTINE BROT(IDAY,MON,IYR,IROT,IDNO)
C
      CALL DAMON(7,2,1832,NDAY0,1)
      IYR1 = IYR
      CALL DAMON(IDAY,MON,IYR,NDAY2,1)
      IF(IYR1 .EQ. 1832) THEN
         NDAY1 = 0
         NDAY2 = NDAY2 - NDAY0
      ELSE
         CALL DAMON(31,12,1832,NDAY1,1)
         NDAY1 = NDAY1 - NDAY0
      END IF
C
      NDAY4 = 0
      IF(IYR .GT. 1832) THEN
         DO 10 IL = 1833,IYR-1
            CALL DAMON(31,12,IL,NDAY3,1)
            NDAY4 = NDAY4 + NDAY3
   10    CONTINUE
      END IF
C
      NDAYT = NDAY1 + NDAY4 + NDAY2
      CALL IDIVREM(NDAYT,27,IROT,IREM)
      IF(IREM .EQ. 0) THEN
         IDNO = 27
      ELSE
         IROT = IROT + 1
```

```
         IDNO = IREM
      END IF
      RETURN
      END
C>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
C
C**** Function ROUND to return the closest multiple of X2 to X1,
C     (eg. X1=1234.5, X2=10, then ROUND=1230.0)
C
C**** Last revised 20th April 1990
C
      FUNCTION ROUND(X1,X2)
      REAL ROUND,X1,X2,X3,S
      S  = SIGN(1.,X1)
      X1 = ABS(X1)
      X3 = MOD(X1,X2)
      ROUND = X1-X3
      IF(X3 .GT. X2/2) ROUND = ROUND+X2
      X1    = X1*S
      ROUND = ROUND*S
      RETURN
      END
C>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
C
C**** Function IROUND to return the closest multiple of I2 to I1,
C     (eg. I1=1234, I2=10, then IROUND=1230)
C
C**** Last revised 20th April 1990
C
      FUNCTION IROUND(I1,I2)
      INTEGER IROUND,I1,I2,I3,IS
      IS = SIGN(1,I1)
      I1 = ABS(I1)
      I3 = MOD(I1,I2)
      IROUND = I1-I3
      IF(I3 .GT. I2/2) IROUND = IROUND+I2
      I1     = I1*IS
      IROUND = IROUND*IS
      RETURN
```

```
      END
C>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
C
C**** Subroutine AUTOC to compute the autocorrelation coefficients for
C     the NX data in array X up from lag 1 to LAG.  The coefficients are
C     returned in array ACF
C
C**** 21 Nov 1991 : DJK
C
      SUBROUTINE AUTOC(X,NX,LAG,ACF,XVR)
      IMPLICIT NONE
      INTEGER IL,KL,NX,LAG,NT
      DOUBLE PRECISION XMN,XVR,AMN,ACOV
      REAL X,ACF
      DIMENSION X(NX),ACF(LAG)
C
C**** Calculate the mean and variance of the data
      XMN = 0.D0
      XVR = 0.D0
      AMN = DBLE(X(1))
      DO 10 IL = 2,NX
         XMN   = XMN + DBLE(X(IL))-AMN
   10    CONTINUE
         XMN   = AMN + XMN/DBLE(NX)
C
      DO 20 IL = 1,NX
         XVR   = XVR + (DBLE(X(IL))-XMN)**2
   20 CONTINUE
C
C**** Now compute the autocorrelation coefficients
      DO 40 KL  = 1,LAG
        ACOV    = 0.D0
        NT      = NX - KL
           DO 30 IL = 1,NT
              ACOV  = ACOV + (DBLE(X(IL))-XMN)*(DBLE(X(IL+KL))-XMN)
   30      CONTINUE
        ACF(KL) = REAL(ACOV/XVR)
   40   CONTINUE
C
```

```
        RETURN
        END
C>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
C
C**** Subroutine LASTMON to return the date of the 1st day of last month
C     in the form of the day, month, year and day number as well as the
C     number of days in the year
C
C**** Last Revised : 21st September 1995, E Clarke
C
C**** Y2K initial check : 25th January 1999, E.Clarke
C**** Y2K final check   :
C
      SUBROUTINE LASTMON(ID,IM,IY,ND,NDYR)
      IMPLICIT NONE
      INTEGER ID,IM,IY,ND,NDYR,NDATE(3)
C
      CALL IDATE(NDATE)
      IM = NDATE(2) - 1
      IF (IM.EQ.0) THEN
         IM = 12
         IY = NDATE(3) - 1
      ELSE
         IY = NDATE(3)
      ENDIF
      ID = 1
      CALL DAMON(ID,IM,IY,ND,1)
      CALL DAMON(31,12,IY,NDYR,1)
      RETURN
      END
C
C>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
C
C**** Subroutine THISMON to return the date of the 1st day of this month
C     in the form of the day, month, year and day number as well as the
C     number of days in the year
C
C**** Last Revised : 21st September 1995, E Clarke
C
```

```
C**** Y2K initial check : 25th January 1999, E.Clarke
C**** Y2K final check   :
C
      SUBROUTINE THISMON(ID,IM,IY,ND,NDYR)
      IMPLICIT NONE
      INTEGER ID,IM,IY,ND,NDYR,NDATE(3)
C
      CALL IDATE(NDATE)
      IM = NDATE(2)
      IY = NDATE(3)
      ID = 1
      CALL DAMON(ID,IM,IY,ND,1)
      CALL DAMON(31,12,IY,NDYR,1)
      RETURN
      END
C
C>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
C
C**** Subroutine DATERD to return 'yesterdays' date in the form of the
C     day, month, year and day number as well as the number of days in
C     the year. Todays date is obtained first from the Sun system.
C
C**** Last Revised : 28th July 1995, E Clarke
C
C**** Y2K initial check : 25th January 1999, E.Clarke
C**** Y2K final check   :
C
      SUBROUTINE DATERD(ID,IM,IY,ND,NDYR)
      IMPLICIT NONE
      INTEGER ID,IM,IY,ND,NDYR,NDATE(3)
C
      CALL IDATE(NDATE)
      ID = NDATE(1)
      IM = NDATE(2)
      IY = NDATE(3)
      CALL DAMON(ID,IM,IY,ND,1)
      IF(ND .EQ. 1) THEN
        IY = IY-1
        IM = 12
```

```
          ID = 31
          CALL DAMON(ID,IM,IY,ND,1)
        ELSE
           ND = ND-1
        END IF
        CALL DAMON(ID,IM,IY,ND,2)
        CALL DAMON(31,12,IY,NDYR,1)
        RETURN
        END
C
C>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>

C*******************************************************************************
C Routine:   LEAP2     Author: AWP Thomson         Date: 20-07-94
C
C Function:  Gets number of days in specified year
C
C Notes:     FIX FOR Y2K - WON'T WORK IN 2090-2100 !! WILL WORK FOR 4 DIGIT YEAR
C
C*******************************************************************************
      SUBROUTINE LEAP2(YEAR,IDAYSINYR)
      IMPLICIT   NONE
      INTEGER    YEAR,IDAYSINYR
      LOGICAL    LEAPYEAR
      IF(YEAR.LT.100)  YEAR=YEAR+1900
CY2K
      IF(YEAR.LE.1990) YEAR=YEAR+100
CY2K
      LEAPYEAR=.FALSE.
      IF(MOD(YEAR,4).NE.0) THEN
           LEAPYEAR =.FALSE.
      ELSE
           IF(MOD(YEAR,100).NE.0) THEN
            LEAPYEAR=.TRUE.
           ELSE
            IF(MOD(YEAR,400).NE.0) THEN
              LEAPYEAR=.FALSE.
            ELSE
              LEAPYEAR=.TRUE.
```

```
           END IF
          END IF
      END IF
      IDAYSINYR=365
      IF(LEAPYEAR) IDAYSINYR=366
      RETURN
      END
C*******************************************************************************
C Routine:   DAYNUMTOMON     Author: AWP Thomson         Date: 20-07-94
C
C Function:  Gets month and day from year and daynumber
C
C Notes:     Checks supplied daynumber fits expected number of days for year.
C Notes:     FIX FOR Y2K - MAY NOT WORK IN 2099-2100 !!!
C
C*******************************************************************************
      SUBROUTINE DAYNUMTOMON(YERIN,DAYIN,YER,MON,DAY)
      IMPLICIT   NONE
      INTEGER    YER,DAYNUM,YERNUM,MON,DAY,IDAYSINYR,I,DAYINMON1,
     1           YERIN,DAYIN,DAYINMON
      DIMENSION  DAYINMON1(12),DAYINMON(12)
      DATA       DAYINMON1/31,59,90,120,151,181,
     1                    212,243,273,304,334,365/

      YERNUM=YERIN
      DAYNUM=DAYIN
      YER=0
      MON=0
      DAY=0
      CALL LEAP2(YERNUM,IDAYSINYR)
      IF(DAYNUM.LE.0) THEN
       YERNUM=YERNUM-1
CY2K
       IF(YERNUM.LT.0) YERNUM=YERNUM+100
CY2K
       CALL LEAP2(YERNUM,IDAYSINYR)
       DAYNUM=DAYNUM+IDAYSINYR
      END IF
      IF(DAYNUM.GT.IDAYSINYR) THEN
```

```
       YERNUM=YERNUM+1
CY2K
       IF((YERNUM.GE.100).AND.(YERNUM.LT.1900))YERNUM=YERNUM-100
CY2K
       DAYNUM=DAYNUM-IDAYSINYR
      END IF
      YER=YERNUM
      CALL LEAP2(YERNUM,IDAYSINYR)
      DAYINMON(1)=DAYINMON1(1)
      IF(IDAYSINYR.GT.365) THEN
       DO 10 I=2,12
        DAYINMON(I)=DAYINMON1(I)+1
10     CONTINUE
      ELSE
       DO 11 I=2,12
        DAYINMON(I)=DAYINMON1(I)
11     CONTINUE
      END IF
      DO 20 I=1,12
       IF(DAYNUM.LE.DAYINMON(I)) THEN
        MON=I
        IF(I.GT.1) THEN
         DAY=DAYNUM-DAYINMON(I-1)
        ELSE
         DAY=DAYNUM
        END IF
        GOTO 30
       END IF
20    CONTINUE
30    CONTINUE
      RETURN
      END
C*******************************************************************************
C Routine:   MONTODAYNUM     Author: AWP Thomson         Date: 20-07-94
C
C Function:  Gets year and daynumber from day, month and year
C
C Notes:
C
```

```
C****************************************************************************
      SUBROUTINE MONTODAYNUM(YER,MON,DAY,YEROUT,DAYOUT)
      IMPLICIT   NONE
      INTEGER    YER,DAYNUM,YERNUM,MONNUM,
     1           MON,DAY,IDAYSINYR,I,DAYINMON1,
     1           YEROUT,DAYOUT,DAYINMON
      DIMENSION  DAYINMON1(12),DAYINMON(12)
      DATA       DAYINMON1/31,59,90,120,151,181,
     1                    212,243,273,304,334,365/

      YERNUM=YER
      MONNUM=MON
      DAYNUM=DAY
      YEROUT=0
      DAYOUT=0
      CALL LEAP2(YERNUM,IDAYSINYR)
      DAYINMON(1)=DAYINMON1(1)
      IF(IDAYSINYR.GT.365) THEN
       DO 10 I=2,12
        DAYINMON(I)=DAYINMON1(I)+1
10     CONTINUE
      ELSE
       DO 11 I=2,12
        DAYINMON(I)=DAYINMON1(I)
11     CONTINUE
      END IF
      YEROUT=YERNUM
      IF(MONNUM.GT.1) THEN
       DAYOUT=DAYINMON(MONNUM-1)+DAYNUM
      ELSE
       DAYOUT=DAYNUM
      ENDIF
      RETURN
      END
C Yes I know that there looks as if there is a coding error, but for some
C reason this works. I dont know why.
      SUBROUTINE IDATESUB(IMON,IDAY,IYER)
      INTEGER JARRAY(3),IDAY,IMON,IYER
      CALL IDATE(JARRAY)
```

```
      IDAY=JARRAY(1)
      IMON=JARRAY(2)
      IYER=JARRAY(3)
      RETURN
      END
c
c>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
c
c>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
c
c**** Subroutine tdaterd to return 'todays' date in the form of the
c     day, month, year and day number as well as the number of days in
c     the year from the UNIX system date.
c
c**** Last Revised : 19th January 1996, E.Clarke
c
      subroutine tdaterd(id,im,iy,nd,ndyr)
      implicit none
      integer id,im,iy,nd,ndyr,ndate
      dimension ndate(3)
c
      call idate(ndate)
      id = ndate(1)
      im = ndate(2)
      iy = ndate(3)
c
      call damon(id,im,iy,nd,1)
      call damon(31,12,iy,ndyr,1)
      return
      end
c
c>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
c
c>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
c
c**** Subroutine stat to calculate various descriptive statistics
c     on the data array passed.
c
c**** Last Revised : 2nd March 1999, E.Clarke
```

```
c
      subroutine stat(x,n,flag,xmin,xmax,xrms,xsd,nmiss)
      implicit none
      integer n,i,npts,nmiss
      real x(n),flag,xmin,xmax,xmn,xrms,xsd,var
      npts = 0
      xmn = 0.0
      xrms = 0.0
      xmax = -0.9e22
      xmin =  0.9e22

      do 10 i=1,n
         if(x(i).ne.flag) then
            npts = npts + 1
            xmn = xmn + x(i)
            xrms = xrms + x(i)*x(i)
            if(x(i).gt.xmax) xmax = x(i)
            if(x(i).lt.xmin) xmin = x(i)
         endif
   10 continue
      nmiss = n - npts
      if(npts.ne.0) then
         xmn = xmn/npts
         xrms = sqrt( xrms/npts )
      else
         xmn = flag
         xrms = flag
      endif
      if(npts.le.1) then
         xmax = flag
         xmin = flag
      endif

      if(xmn.ne.flag) then
         var = 0.0
         do 20 i=1,n
            if(x(i).ne.flag) then
               var = var + (xmn-x(i))*(xmn-x(i))
            endif
```

```
   20    continue
         if(n-nmiss-1 .gt. 0) then
            var = var/real(n-nmiss-1)
            xsd = sqrt(var)
         else
            xsd = flag
         endif
      else
         xsd = flag
      endif

c      if(nmiss.gt.(n/10)) then
c         xmn  = flag
c         xsd  = flag
c         xmin = flag
c         xmax = flag
c      endif

      return
      end
c
c>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
c
c>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
c
c**** Subroutine stdev to calculate the standard deviation (std)
c     of the npts elements passed to arr(npts).
c     Flag is the value used to flag missing data.
c
c*cc**** Last revised 31-08-1999 : EC
c
      subroutine stdev(arr,npts,flag,std,nmiss)
      implicit none
      integer il,npts,nst,iout,nmiss
      real arr,del,rmn,var,std,flag,amn,tiny
      dimension arr(npts)
      parameter (iout = 6, tiny = 0.01)

      if(npts .eq. 1) then
```

```
          std =  flag
       else
          rmn   = 0.0
          nmiss =   0
          nst   =   0
10        nst   = nst + 1
          if(nst .eq. npts+1) then
             std = flag
             return
          end if
          del   = abs(arr(nst) - flag)
          if(del .lt. tiny) then
             nmiss = nmiss + 1
             goto 10
          else
             amn   = arr(nst)
          end if
c
          do 20 il = nst+1,npts
             del = abs(arr(il) - flag)
             if(del .gt. tiny) then
                rmn = rmn + arr(il)-amn
             else
                nmiss = nmiss + 1
             end if
20        continue
          rmn = amn + rmn/real(npts-nmiss)
c
          var = 0.0
          do 30 il=1,npts
             del = abs(arr(il) - flag)
             if(del .gt. 0.01) var = var + (rmn-arr(il))*(rmn-arr(il))
30        continue
          if(npts-nmiss-1 .gt. 0) then
             var = var/real(npts-nmiss-1)
             std = sqrt(var)
          else
             var = flag
             std = flag
```

```
         end if
      end if
c
      return
      end
c
cccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
c       subroutine mk_lf_0
c               V. Lesur 18/11/2004
c
c
c Computes Legendre polynomial (m=0) using 3.7.14 page 109 in
c  Backus, Parker and Constable, Foundations of Geomagnetism,
c  Cambridge University Press, 1996
c
c Outputs (* sqrt(L+0.5)) to be compared for lm=10000 with Table 1 in :
c Smith Olver and Lozier, 1981, Extended-Range Airthmetic and Normalized
c    Legendre Polyniomials, ACM Transactions on Mathematical Software,
c    Vol 7, No 1,93-105.
c
c       input:
c         nl            degree max
c         dc            cos(colatitude)
c       output:
c         dlf           legendre polynomial from 0 to nl
c                      i.e. needs array dim(nl+1)
c
cccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
        subroutine  mk_lf_0(nl,dc,dlf)
c
        implicit real*8 (a-h,o-z)
        implicit integer*4 (i-n)
c
        integer nl,il
        real*8 dlf(*),dc,dbeta,dalpha
c
        dlf(1)=1.d0
        dlf(2)=dc
c
```

```
        do il=2,nl
          dalpha=dble(2*il-1)*dlf(il)*dc
          dbeta=dble(il-1)*dlf(il-1)
c
          dlf(il+1)=dalpha-dbeta
          dlf(il+1)=dlf(il+1)/dble(il)
        enddo
c
        return
        end
cccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
c  subroutine mk_lf_dlf
c               V. Lesur 19 June 2005
c
c       Computes legendre Functions and their derivatives along theta
c
c       CALLED: mklf_F.f
c
c       limitations of mklf_F apply
c
c       Fast version: tests reduced to minimum
c
c       input:
c         nm/nl         order and degree max
c         dc/ds         cos(colatitude)/sin(colatitude)
c       output:
c         dlf           legendre function from nm to nl
c         ddlf          derivative of legendre function from nm to nl
c
cccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
        subroutine mk_lf_dlf(nm,nl,ds,dc,dlf,ddlf)
c
        implicit real*8 (a-h,o-z)
        implicit integer*4 (i-n)
c
c
        integer nm,nl,il,jl
        real*8 ds,dc,dlf(*),ddlf(*),d1,d2
c
```

```
        call mklf_F(nm,nl,ds,dc,dlf)
c
        if (ds.eq.0.0d0) then
          if(nm.ne.1) then
            do il=1,nl+1
              ddlf(il)=0.0d0
            enddo
          else
            ddlf(1)=-1.0d0
            ddlf(2)=-dsqrt(3.0d0)
            do il=3,nl+1
              d1=(2*il-1)/dsqrt(dble(il*il-1))
              d2=dsqrt(dble((il-1)**2-1)/(il*il-1))
              ddlf(il)=d1*ddlf(il-1)-d2*ddlf(il-2)
            enddo
          endif
        else
          do il=nl,nm,-1
            jl=il-nm+1
            if(il-nm.gt.0)then
              d1=dsqrt(dble((il-nm)*(il+nm)))
              d2=dble(il)
              ddlf(jl)=(d2*dc*dlf(jl)-d1*dlf(jl-1))/ds
            else
              ddlf(1)=nm*dc*dlf(1)/ds
            endif
          enddo
        endif
c
        return
        end
cccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
c  subroutine mk_lf_dlf_0
c V. Lesur 19 June 2005
c
c Computes legendre Polynomials and their derivatives along theta
c
c       CALLED: mk_lf_0.f
c
```

```
c       input:
c       nl degree max
c       dc/ds     cos(colatitude)/sin(colatitude)
c output:
c   dlf legendre polynomial from 0 to nl dim=nl+1
c   ddlf derivative of legendre polynomial from 0 to nl
c                 dim=nl+1
c
cccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
  subroutine mk_lf_dlf_0(nl,ds,dc,dlf,ddlf)
c
        implicit real*8 (a-h,o-z)
        implicit integer*4 (i-n)
c
        integer nl,il
        real*8 ds,dc,ddlf(*),dlf(*)
c
        call mk_lf_0(nl,dc,dlf)
        if (ds.eq.0.0d0) then
          do il=1,nl+1
            ddlf(il)=0.0d0
          enddo
        else
          do il=nl,1,-1
            ddlf(il+1)=dble(il)*(dc*dlf(il+1)-dlf(il))/ds
          enddo
          ddlf(1)=0.0d0
        endif
c
        return
        end
ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
c         subroutine mk_wave_cubic_t
c          Vincent Lesur       15/03/2007
c
c  build a translated wavelet based on cubic polynomials
c
c  Find the alpa(l) for
c        F_ij(tht,phy)=SUM(L:Lm) alpha(l) PL(cos(colat))
```

```
c
c  alphas are normalised such that 1=SUM(0:Lm) (l+1) alpha(l) PL(cos(0))
c                              i.e. 1=SUM(0:Lm) (l+1) alpha(l)
c
c  alpha(0:lm)=0 see below for lm definition
c
c  input:
c      nl maximum value of the degree + 1
c      lm       minimum value of the degree - 1  (>-1)
c  output:
c      al(*)    alpha values (dim min: nl)
c
ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
        subroutine mk_wave_cubic_t(nl,lm,alph)
c
        implicit none
c
        integer nl,lm
        real*8 alph(*)
c
        integer i,nl1,nl2
        real*8 dlm,dw,cubicp
c
        alph(1:nl)=0.0d0
c
c     Calculate the alpha's (difference of scaled cubic polynomials)
        dlm=dble(nl-lm)
        nl1=idint(dlm+1.d0)
        do i=1,nl1-1
          dw=dble(i-1)/dlm
          cubicp=(dw-1.d0)**2
          cubicp=cubicp*(2.d0*dw+1.d0)
          alph(lm+i)=cubicp
        enddo
c
        dlm=dble(nl-lm)/2.d0
        nl2=idint(dlm+1.d0)
        do i=1,nl2-1
          dw=dble(i-1)/dlm
```

```
          cubicp=(dw-1.d0)**2
          cubicp=cubicp*(2.d0*dw+1.d0)
          alph(lm+i)=alph(lm+i)-cubicp
        enddo
c
c     Normalisation
        dw=0.0d0
        do i=1,nl
          dw=dw+dble(i)*alph(i)
        enddo
        do i=1,nl
          alph(i)=alph(i)/dw
        enddo
c
        return
        end
cccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
c
c subroutine mkArows
c V. Lesur 09/02/2005
c
c       previouslu mkcondmat, should be compile using F90
c
c       calculate the condition matrix for NP data points.
c       call recurcive subroutines
c
c input:
c   NP Number of data points
c         npm           leading dim of ppos
c         NT(NP) Type of data (1->X,2->Y,3->Z,4->F ...) dim min (NP)
c         ND space dimension
c         nb            Number of base functions
c         ppos point position in ndD
c         BS            the "Base functions" subroutine to use
c         XYZF          X,Y,Z component and total field dim min (NP,4)
c
c       output:
c   AA(NP,NB) matrix of conditions
c
```

```
ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
subroutine mkArows(np,npm,nt,nd,nb,ppos,bs,bc,AA)
c
        implicit none
c
integer np,npm,nt(*),nd,nb
        real*8 ppos(nd+1,*),AA(np,*),bc(*)
        real*8, allocatable :: row(:)
c
        integer ip,ib,j
c
        external bs
        allocate(row(1:nb))
c
        do ip=1,np
          call BS('i',nt(ip),nb,bc,ppos(1,ip),row)
          do ib=1,nb
            AA(ip,ib)=row(ib)
          enddo
        enddo
        deallocate(row)
c
        return
end
ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
c subroutine mkcovm
c                V. Lesur 09/02/2005
c
c   modified:
c      25/08/2005 id itroduced in parameter list of expcovm (vbfl)
c
c       should be compiled using F90
c
c       Computes the covariance matrix for np points
c       and perform an eigenvector/eigenvalue decomposition
c
c       input:
c         ip             pointer to first data point
c         np          number of points
```

```
c         icov/jcov      Integer arrays describing cov format
c         cov            Covariance matrix in SLAP column format
c
c       output:
c         id             id=0 for diagonal cov matrix
c         dwgh(*)  inverse of variances
c         drot(*)        Rotations to transform cov in dwgh
c
ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
        subroutine mkcovm(ip,np,icov,jcov,cov,id,dwgh,drot)
c
        implicit none
c
        integer ip,np,icov(*),jcov(*),id
        real*8 cov(*),dwgh(*),drot(*)
c
        id=1
        if(np.eq.1)id=0
c
c  Calculate the covariance matrix CM for np data points
        call expcovm(id,ip,np,icov,jcov,cov,drot)
c
c  Diagonalise the covariance matrix
        call diagcov(id,np,drot,dwgh)
c
        return
        end
cccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
c  subroutine mkdlf
c V. Lesur 01/02/2002
c
c Computes the derivative along theta of the legendre function
c       at a given order "m" from degree m to nl (nl > m).
c       Code based on the MKSSP subroutine writen by V.Lesur 9/17/97
c
c       CALLED: mklf.f
c
c       limitations of mklf appl
c
```

```
c       input:
c         nm/nl order and degree max
c         dc/ds         cos(colatitude)/sin(colatitude)
c output:
c   ddlf derivative of legendre function from nm to nl
c
cccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
subroutine mkdlf(nm,nl,ds,dc,ddlf)
c
        implicit none
c
        integer nm,nl,il,jl
        real*8 ds,dc,ddlf(*),d1,d2
c
        if (ds.eq.0.0d0) then
          if(nm.ne.1) then
            do il=1,nl+1
              ddlf(il)=0.0d0
            enddo
          else
            ddlf(1)=-1.0d0
            ddlf(2)=-dsqrt(3.0d0)
            do il=3,nl+1
              d1=(2*il-1)/dsqrt(dble(il*il-1))
              d2=dsqrt(dble((il-1)**2-1)/(il*il-1))
              ddlf(il)=d1*ddlf(il-1)-d2*ddlf(il-2)
            enddo
          endif
        else
c
          call mklf(nm,nl,ds,dc,ddlf)
c
          do il=nl,nm,-1
            jl=il-nm+1
            if(il-nm.gt.0)then
              d1=dsqrt(dble((il-nm)*(il+nm)))
              d2=dble(il)
              ddlf(jl)=(d2*dc*ddlf(jl)-d1*ddlf(jl-1))/ds
            else
```

```
          ddlf(1)=nm*dc*ddlf(1)/ds
        endif
      enddo
    endif
c
    return
    end
cccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
c  subroutine mkdlf_F
c V. Lesur 01/02/2002
c
c Computes the derivative along theta of the legendre function
c       at a given order "m" from degree m to nl (nl > m).
c       Code based on the MKSSP subroutine writen by V.Lesur 9/17/97
c
c       CALLED: mklf_F.f
c
c       limitations of mklf_F apply
c
c       Fast version: tests reduced to minimum
c
c       input:
c         nm/nl order and degree max
c         dc/ds         cos(colatitude)/sin(colatitude)
c output:
c   ddlf derivative of legendre function from nm to nl
c
cccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
subroutine mkdlf_F(nm,nl,ds,dc,ddlf)
c
        implicit real*8 (a-h,o-z)
        implicit integer*4 (i-n)
c
c
        integer nm,nl,il,jl
        real*8 ds,dc,ddlf(*),d1,d2
c
        if (ds.eq.0.0d0) then
          if(nm.ne.1) then
```

```
            do il=1,nl+1
              ddlf(il)=0.0d0
            enddo
          else
            ddlf(1)=-1.0d0
            ddlf(2)=-dsqrt(3.0d0)
            do il=3,nl+1
              d1=(2*il-1)/dsqrt(dble(il*il-1))
              d2=dsqrt(dble((il-1)**2-1)/(il*il-1))
              ddlf(il)=d1*ddlf(il-1)-d2*ddlf(il-2)
            enddo
          endif
        else
c
          call mklf_F(nm,nl,ds,dc,ddlf)
c
          do il=nl,nm,-1
            jl=il-nm+1
            if(il-nm.gt.0)then
              d1=dsqrt(dble((il-nm)*(il+nm)))
              d2=dble(il)
              ddlf(jl)=(d2*dc*ddlf(jl)-d1*ddlf(jl-1))/ds
            else
              ddlf(1)=nm*dc*ddlf(1)/ds
            endif
          enddo
        endif
c
99999   return
        end
cccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
c  subroutine mkdlf_F0
c V. Lesur 18/11/2004
c
c Computes the derivative along theta of the legendre Polynomial
c
c       CALLED: mklf_F0.f
c
c       input:
```

```
c         nl degree max
c         dc/ds         cos(colatitude)/sin(colatitude)
c output:
c   ddlf derivative of legendre polynomial from 0 to nl
c                       i.e. dim=nl+1
c
cccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
subroutine mkdlf_F0(nl,ds,dc,ddlf)
c
        implicit real*8 (a-h,o-z)
        implicit integer*4 (i-n)
c
        integer nl,il
        real*8 ds,dc,ddlf(*)
c
        if (ds.eq.0.0d0) then
          do il=1,nl+1
            ddlf(il)=0.0d0
          enddo
        else
          call mklf_F0(nl,dc,ddlf)
          do il=nl,1,-1
            ddlf(il+1)=dble(il)*(dc*ddlf(il+1)-ddlf(il))/ds
          enddo
          ddlf(1)=0.0d0
        endif
c
99999   return
        end
cccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
c subroutine mklf
c V. Lesur 01/02/2001
c
c       This is a subroutine based on MKS.f I wrote back in 1997
c
c       Computes the Schmidt normalised legendre functions dlf for
c       a given order "m" from degree m to lm (lm >= m).
c
c       Underflow are avoided by writing small reals in the format:
```

```
c       d1*dfloat(ibase)**(iepsi*nm*iexp)
c       where ibase amd iepsi are machine dependent and may need to be
c       adjusted.
c
c       The subsroutine has been tested on Driscoll&Healy sampling
c       theorem points up to degree 512 on a DECalpha station
c       It needs to be tested again on other platform
c
c       Olver&Smith subroutines subroutines should be used for
c       degree >> 256
c
c       Recurrence 3.7.28 Fundations of geomagetism, Backus 1996
c
c    inputs:
c dc/ds           cos(colatitude)/sin(colatitude)
c       nm/lm           order and degree max
c    outputs:
c       dlf             Legendre functions [dim min: max (2,lm-nm+1)]
c
cccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
        subroutine  mklf(nm,lm,ds,dc,dlf)
c
        implicit none
c
        integer nm,lm,ierr,il,ii,ibase,iexp,iepsi
        real*8 dlf(*),ds,dc,dnm,d1,d2,dbeta,dalpha,dlim
c
        real*8 dgamln
        external dgamln
c
c Machine constants
c
        iepsi=-1021                     !smallest exposant
        ibase=2                         !Base representation
c
c Parameter set up
c
        iepsi=iepsi+1                   !to work slightly off the limit
        ii=iepsi
```

```
      if(nm.ne.0) ii=nint(float(iepsi)/float(nm))
      dlim=dble(ibase)**(ii)          !limit too small number

c
c Start Recurence
c
c l=nm,  l=nm+1
c
      dnm=dble(nm) !dble real for nm
      d1=dgamln(2*dnm+1.0d0,ierr)     ! d1=log(fact(2dnm))
      d2=dgamln(dnm+1.0d0,ierr)       ! d2=log(fact(dnm))
      if(ierr.ne.0)then
        write(*,*)'mklf_F: Cannot computes normalisation cst !'
        stop
      endif
c
      d2=0.5d0*d1-d2                  !d2=sqrt(fact(2dnm))/fact(dnm)
      d2=d2-dnm*dlog(2.0d0) ! is the
      d2=dexp(d2) ! normalisation cst.
      if(nm.ne.0)d2=d2*dsqrt(2.0d0) !special case  m=0
c
      d1=ds
      if(d1.ne.0.0d0) then
c
        iexp=0                        !
        do while (dabs(d1).le.dlim)         !
          iexp=iexp+1                 ! Scaling small numbers.
          d1=d1*dfloat(ibase)**(-ii)  !
        enddo                         !
c
        d1=d1**nm
      else
        if(nm.eq.0)d1=1.d0
      endif
c
      dlf(1)=d1*d2 !leg. func. (m,m)
      dlf(2)=dlf(1)*dc*dsqrt(dble(2*nm+1)) !leg. func. (m+1,m)
c
c l=nm+2....lm
```

```
c
        do il=2,lm-nm !l=nm+il-1
c
          d1=dble((il-1)*(il+2*nm-1)) !(l-m)*(l+m)
          d2=dble((il)*(il+2*nm)) !(l-m+1)*(l+m+1)
          dbeta=dsqrt(d1/d2) ! recurrence coeff.
          d1=dble(2*(il+nm)-1) !2l+1
          dalpha=d1/dsqrt(d2) !
c
          dlf(il+1)=dalpha*dlf(il)*dc !
     +                    -dbeta*dlf(il-1) !leg. func. (nm+il-1,nm)
        enddo
c
c Scaling back to normal
c
        d1=dfloat(ibase)**(ii*nm)              !scaling factor
        dlim=dfloat(ibase)**(iepsi)            !my zero limit
c
        dlim=dlim/d1
        do while (iexp.ne.0)
          do il=1,lm-nm+1
            if(dabs(dlf(il)).lt.dlim)dlf(il)=0.0d0
            dlf(il)=dlf(il)*d1
          enddo
          iexp=iexp-1
        enddo
c
        return
        end
cccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
c subroutine mklf_F
c V. Lesur 01/02/2001
c
c    Modified 23/02/04 : GOTO removed
c
c       computes the Schmidt normalised legendre functions dlf for
c       a given order "m" from degree m to lm (lm > m).
c       Fast version: minimium check, do not avoid underflows
c
```

```
c       recurrence 3.7.28 Fundations of geomagetism, Backus 1996
c
c    inputs:
c dc/ds           cos(colatitude)/sin(colatitude)
c       nm/lm           order and degree max
c    outputs:
c       dlf             Legendre functions [dim min: max (2,lm-nm+1)]
c
ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
        subroutine  mklf_F(nm,lm,ds,dc,dlf)
c
        implicit real*8 (a-h,o-z)
        implicit integer*4 (i-n)
c
        parameter (lmax=128)
c
        integer nm,lm,ierr,il
        real*8 dlf(*),ds,dc,dnm,d1,d2,dbeta,dalpha
c
c    start calculs
c
c     l=nm,  l=nm+1
c
        dnm=dble(nm) !dble real for nm
        d1=dgamln(2*dnm+1.0d0,ierr)      ! d1=log(fact(2dnm))
        d2=dgamln(dnm+1.0d0,ierr)        ! d2=log(fact(dnm))
        if (ierr.ne.0) then
          write(*,*)'mklf_F: Cannot compute normalisation const !'
          stop
        endif
c
        d2=0.5d0*d1-d2                   !d2=sqrt(fact(2dnm))/fact(dnm)
        d2=d2-dnm*dlog(2.0d0) !
        d2=dexp(d2) !normalisation cst.
        if (nm.ne.0) d2=d2*dsqrt(2.0d0) !special case  m=0
c
        d1=ds
        if(d1.ne.0.0d0) then
          d1=d1**nm
```

```
        else
          if(nm.eq.0)d1=1.d0
        endif
c
        dlf(1)=d1*d2 !leg. func. (m,m)
        dlf(2)=dlf(1)*dc*dsqrt(dble(2*nm+1)) !leg. func. (m+1,m)
c
c     l=nm+2....lm
c
        do il=2,lm-nm !l=nm+il-1
c
          d1=dble((il-1)*(il+2*nm-1)) !(l-m)*(l+m)
          d2=dble((il)*(il+2*nm)) !(l-m+1)*(l+m+1)
          dbeta=dsqrt(d1/d2) ! recurrence coeff.
          d1=dble(2*(il+nm)-1) !2l+1
          dalpha=d1/dsqrt(d2) !
c
          dlf(il+1)=dalpha*dlf(il)*dc !
     +                    -dbeta*dlf(il-1) !leg. func. (nm+il-1,nm)
        enddo
c
        return
        end
cccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
c       subroutine mklf_F0
c               V. Lesur 18/11/2004
c
c
c Computes Legendre polynomial (m=0) using 3.7.14 page 109 in
c  Backus, Parker and Constable, Foundations of Geomagnetism,
c  Cambridge University Press, 1996
c
c Outputs (* sqrt(L+0.5)) to be compared for lm=10000 with Table 1 in :
c Smith Olver and Lozier, 1981, Extended-Range Airthmetic and Normalized
c    Legendre Polyniomials, ACM Transactions on Mathematical Software,
c    Vol 7, No 1,93-105.
c
c       input:
c         nl            degree max
```

```
c         dc            cos(colatitude)
c       output:
c         dlf           legendre polynomial from 0 to nl
c                      i.e. needs array dim(nl+1)
c
cccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
       subroutine  mklf_F0(lm,dc,dlf)
c
        implicit real*8 (a-h,o-z)
        implicit integer*4 (i-n)
c
        integer lm,il
        real*8 dlf(*),dc,dbeta,dalpha
c
        dlf(1)=1.d0
        dlf(2)=dc
c
        do il=2,lm
c
          dalpha=dble(2*il-1)*dlf(il)*dc
          dbeta=dble(il-1)*dlf(il-1)
c
          dlf(il+1)=dalpha-dbeta
          dlf(il+1)=dlf(il+1)/dble(il)
        enddo
c
        return
        end
c-----------------------------------------------------------------------
c             procedure mkr
c         vincent Lesur   04/2001
c         Calcul la matrice rotation R tq R'*A = tri-diag
c
c       entree :    da          rotations a appliquer
c (output from trig)
c                   nn          nombre de ligne
c
c       sortie :    da          matrice de rotation R
c
```

```
c-------------------------------------------------------------------------
             subroutine mkr(nn,da)
c
        implicit none
c
        real*8 da(*),alpha,beta,gama
        integer nn,k,i,ik,j,jk,ji
c
c     debut
        da(nn*nn)=1.d0
c
c      boucle colonnes
        do k=nn-2,1,-1
c
c      initialise a Id
          do i=k+2,nn
            da(i+k*nn)=0.d0
            da(1+k+(i-1)*nn)=0.d0
          enddo
          da(1+k+k*nn)=1.d0
c
          do i=nn,k+2,-1
            ik=i+(k-1)*nn
            if (da(ik).ne.0.0d0) then
c      construction rotation
              if(dabs(da(ik)).eq.1.d0)then
                 alpha=0.0d0
                 beta=da(ik)
              else
              if (dabs(da(ik)).lt.1.0d0) then
                  alpha=dsqrt(1.0d0/(1.d0+(2.d0*da(ik))**2))
                  beta=alpha*2.d0*da(ik)
              else
                  alpha=dsqrt(1.0d0/(1.d0+(da(ik)/2.d0)**2))
                  beta=alpha*da(ik)/2.d0
              endif
              endif
c      right multiplication in reverse order => left multiplication
              do j=k+1,nn
```

```
                jk=1+k+(j-1)*nn
                ji=i+(j-1)*nn
                gama=alpha*da(jk)+beta*da(ji)
                da(ji)=beta*da(jk)-alpha*da(ji)
                da(jk)=gama
              enddo
            endif
          enddo
        enddo
c
c first column/row
        do i=2,nn
          da(i)=0.d0
          da(1+(i-1)*nn)=0.d0
        enddo
        da(1)=1.d0
c
        return
        end
cccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
c   subroutine mkwgh
c V. Lesur 18/02/2002
c
c   Modified 23/02/04: GOTO removed
c
c       diagonalise the covariance matrix
c       and calculate the weight matrix : WGH
c
c       Correlation coefficients less than dlim*variance are set to 0
c       Variances larger than min(variance)/dlim set to infinity
c
c       input:
c       id      indicator of known diagonal matrix (0 => diag; 1=> non diag)
c np Number of data points
c       CM      npXnp covariance matrix (Symetrique real)
c       dw      Work array dim min= np
c
c       output:
c       wgh     weight matrix (inverse of eigenvalues of CM)
```

```
c       CM      eigen vector of covariance matrix
c       id      may be corrected to 0
c       dw      work array
c
cccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
        subroutine mkwgh(id,np,CM,wgh,dw)
c
        implicit none
c
        integer id,np,il,ic
        real*8 CM(np,*),wgh(*),dw(*),dlim,wmin
c
        dlim=1.d-5
c
c Check if CM diag matrix
        if(id.eq.1)then
          il=0
          do while (il.lt.np.and.id.eq.1)
            il=il+1
            ic=il
            do while (ic.lt.np.and.id.eq.1)
              ic=ic+1
              if(CM(il,ic).ne.0.0d0)then
                if(dabs(CM(il,ic)).gt.CM(il,il)*dlim
     >         .or.dabs(CM(il,ic)).gt.CM(ic,ic)*dlim)then
                  id=0
                else
                  CM(il,ic)=0.0d0
                endif
              endif
            enddo
          enddo
        endif
        id=1-id
c
c  Case of diagonal matrix
        if(id.eq.0)then
c         wmin=CM(1,1)
c         do il=2,np
```

```
c           wmin=dmin1(wmin,CM(il,il))
c         enddo
c         wmin=wmin/dlim
c         do il=1,np
c           if(CM(il,il).gt.wmin)then
c             wgh(il)=0.0d0
c           else
c             wgh(il)=1.d0/CM(il,il)
c           endif
c         enddo
          do il=1,np
            if(CM(il,il).gt.0.0d0)then
              wgh(il)=1.d0/CM(il,il)
            else
              wgh(il)=0.0d0
            endif
          enddo
c
c  Case of NON diagonal matrix
        else
c
c Calculate eigen values and vectors
          call eigenvv(np,CM,wgh,dw)
c
c Inverse eigen values
          wmin=wgh(1)
          do il=2,np
            wmin=dmin1(wmin,wgh(il))
          enddo
          wmin=wmin/dlim
          do il=1,np
            if(wgh(il).gt.wmin)then
              wgh(il)=0.0d0
            else
              wgh(il)=1.d0/wgh(il)
            endif
          enddo
c
        endif
```

```
c
99999   return
        end
cccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
c subroutine opt_d_l3
c Vincent Lesur  13/01/2005
c
c vbfl: modified 24/08/05 for:
c         1- use the transpose of ppos and remove ddat from calling parameters
c vbfl: modified 14/03/07 for:
c         1- preconditioner
c
c       Should be used with F90
c       calling recurcive subroutines
c
c       non-parallel optimiser using non-gradient method
c
c       Computes parameters BC from a set of NPT data points.
c       The "base function" subroutine used is BS
c       The norm used for minimisation correspond to the FM misfit function
c       The STD is computed using fstd function and
c       the damping is done using the Fdamp subroutine
c
c       All these subroutines & functions are user provided and shoud be
c        declared external in the calling program.
c
c       The normal equations are built by adding the contributions of groups
c       of uncorrelated data points. The Data covarience matrix, input in a
c       SLAP column format, is used to determine groups of uncorrelated data
c
c       The inversion process may be non-linear => First estimation of the
c       solution must be given.
c
c       CALLED: cpt_dat_vals, cptstd, cptnp, cptnormeq
c         &     fdamp (external), eig_solve2, eigen_vv
c
c       depending on itmax:
c         itmax > 0       find parameters to fit the data
c         itmax = 0       do the forward problem only
```

```
c         itmax < 0       compute eigen values and vectors of normal equations
c
c       input:
c               path          where to write output files
c               itmax         Maximum number of itterations
c               npmax         number max of data point with correlated errors
c               npm           Maximum total number of data points
c               nd            space dimension
c               npt           Total Number of data
c               ppos          data point position in ndD
c               nb            Number or base function to use
c               BC            First estimation of Base function coefficients
c               dl(3)         damping factor & max condition number acceptable
c               FM            misfit function (like l2_norm.f)
c               BS            Base subroutine to use
c               fstd          std Function
c               fdamp         damping Subroutine
c               nt(*)         data type
c               cov(*)        covariance matrix in SLAP Column format
c               icov/jcov     Integer vector describing cov format
c               stdt          STD Target
c
c       output:
c               BC(*)         Base function coefficients
c               xyzf(*)       results of forward modelling
c               stdt          estimated STD
c               GG            eigen vectors of the last normal matrix
c               BB            eigen values of the last normal matrix
c
cccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
        subroutine opt_d_l3(path,itmax,npmax,npm,nd,npt,ppos,nb,bc,dl
     >  ,FM,BS,fstd,FDAMP,nt,cov,icov,jcov,stdt,xyzf,BB,GG)
c
        implicit none
c
        integer itmax,nd,npt,nb,nt(*),icov(*),jcov(*),i,j,ip,npmax,npm
        real*8  ppos(*),bc(*),cov(*),stdt,dl(*),xyzf(*)
        real*8  BB(*),GG(*)
        real*8, allocatable :: dw(:),dsol(:),eigv(:),ddat(:)
```

```
        character*2 yon_iterate,yon_build,yon_solve
        character path*100
c
        real*8 FM,FSTD
        external FM,BS,FSTD,FDAMP
c
        integer it
        real*8 std
        character cit*2
c
        real*4 t,tarray(2),dtime
c
        std=0.0d0
        it=0
        yon_iterate='ok'
        if(itmax.gt.0)then
          yon_build='in'
          yon_solve='in'
        elseif(itmax.eq.0)then
          yon_build='no'
          yon_solve='no'
        else
          yon_build='ok'
          yon_solve='no'
        endif
c
        do while (yon_iterate.eq.'ok')
          it=it+1
c
c  All: set up
          allocate (ddat(1:npt))
          do i=1,npt
            ddat(i)=ppos(i*(nd+1))
          enddo
c
c  All: Forward modelling & std estimate
          t=dtime(tarray)
          call cpt_dat_vals(nd,npm,npt,nt,ppos,nb,BC,BS,xyzf)
          call cptstd(npmax,npt,icov,jcov,cov,ddat,xyzf,fstd,std)
```

```
          t=dtime(tarray)
          write(*,'(A,e9.2,A,e9.2)')'Forward modelling: user time'
     >        ,tarray(1),' system time',tarray(2)
          write(*,*)
          write(*,*)'std =',std,'  & it   =',it-1
          write(*,*)
c
c  All: What to do next ?
          if(std.gt.stdt.and.yon_build.eq.'in') yon_build='ok'
c
c  All: build normal equations if std > stdt or itmax < 0
          if (yon_build.eq.'ok') then
c
c  MP: start clock
            t=dtime(tarray)
c
c  SP: build their parts of normal equations
            ip=1
    write(*,'(A)') 'Here'
            call cptnormeq(npmax,npm,npt,ip,nd
     >       ,ppos,nb,FM,BS,bc,icov,jcov,cov,ddat,nt,xyzf,GG,BB)
        write(*,'(A)') 'Here'
c
c  MP: stop clock
            t=dtime(tarray)
            write(*,'(A,e9.2,A,e9.2)')'Normal equations: user time'
     >          ,tarray(1),' system time',tarray(2)
            write(*,*)
c
c  MP: Damp the normal equations
            if (it<2) call fdamp(dl,nb,GG,BB,bc)
c
c  All: What to do next ?
            if(yon_solve.eq.'in')yon_solve='ok'
          endif
          deallocate(ddat)
c
c  MP: Now solve the normal equations: it's a MP only
          if (yon_solve.eq.'ok') then
```

```
c
c  MP: Solve
          allocate(dsol(1:nb),eigv(1:nb),dw(1:nb))
          t=dtime(tarray)
          call precond('d',nb,GG,BB,dsol,dw)
          call eig_solve2(nb,dl(3),GG,eigv,BB,dsol)
          call precond('i',nb,GG,BB,dsol,dw)
          t=dtime(tarray)
          write(*,'(A,e9.2,A,e9.2)')'solving system: user time'
     >          ,tarray(1),' system time',tarray(2)
c
c  MP: Update parameters
          do i=1,nb
            BC(i)=BC(i)+dsol(i)
            BB(i)=eigv(i)
          enddo
          deallocate(dsol,eigv,dw)
c
c  MP: and save
          cit(2:2)=char(48+it-int(it/10.)*10)
          cit(1:1)=char(48+int(it/10.))
          i=0
          do while (path(i+1:i+1).ne.' ')
            i=i+1
          enddo
          open(10,file=path(1:i)//'model_it'//cit//'.out')
            write(10,*)'# model at iteration : ',cit
            do i=1,nb
              write(10,*)i,BC(i)
            enddo
          close(10)
        endif
c
c  All: What to do next iteration ?
        if(it.lt.itmax) then
          if(yon_solve.eq.'ok')then
            yon_iterate='ok'
            yon_build='in'
            yon_solve='in'
```

```
c
            else
              yon_iterate='no'
            endif
          else
            if(yon_solve.eq.'ok')then
              yon_iterate='ok'
              yon_build='no'
              yon_solve='no'
            else
              yon_iterate='no'
            endif
          endif
c
c  All: End of itterative process
        enddo
c
c  MP: SVD of GG with eigenvalues in BB and QUIT
        if (itmax.lt.0) then
          allocate(dw(1:nb))
          call eigenvv(nb,GG,BB,dw)
          deallocate(dw)
        endif
c
        stdt=std
c
        return
        end
cccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
c
c   subroutine ortho_rot     v.lesur 1999
c
c    Give the coordinates of a vector in a system of
c    coordinates that is rotated by an angle theta around
c    the axis X, Y or Z of the original system
c
c    Theta positif clockwise
c    axis order (x,y,z) (y,z,x) (z,x,y)
c
```

```
c    input
c       dth       real*8 rotation angle in degree
c       n         integer rotation axis (1=X, 2=Y 3=Z)
c       vx,vy,vz  real*8 Vector coordinates in original SOC
c
c    output
c       vx,vy,vx  real*8 New vector Coordinates
cccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
subroutine ortho_rot(n,dth,vx,vy,vz)
c
        implicit none
c
        integer n
        real*8 dth,vx,vy,vz
        real*8 dct,dst,dd,d2r
c
        d2r=4.d0*datan(1.d0)/180.d0
c
        dct=dcos(dth*d2r)
        dst=dsin(dth*d2r)
c
        if(n.eq.1)then
          dd=dct*vy-dst*vz
          vz=dst*vy+dct*vz
          vy=dd
        elseif(n.eq.2)then
          dd=dct*vz-dst*vx
          vx=dst*vz+dct*vx
          vz=dd
        else
          dd=dct*vx-dst*vy
          vy=dst*vx+dct*vy
          vx=dd
        endif
c
        return
        end
cccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
c         subroutine precond.f
```

```
c                                 V.Lesur 14.03.2007
c
c     For a symmetric definit positive matrix G and a linear system
c     Gx=B, depending on the cswch either:
c      1- cswch='d'  Compute a preconditioner M as the inverse of a
c                    diagonal matrix which elements are sqrt(|g(i,i)|)
c                    left and right multiply G with M
c                    left multiply B with M
c      2- cswch='i'  left multiply x by M
c
c      if g(i,i)=0 M(i,i)=1
c      the operations are :   M.G.M.inv(M).x= M.G.M.x'=M.B
c                             x=M.x'
c
c      input:
c         cswch   CHARACTER  switch
c         nb      INTEGER    Leading dimm of G
c         G(nb,*) REAL*8     matrix symmetric and definit positive
c         B(*)    REAL*8     Left hand side
c         X(*)    REAL*8     Solution
c         M(*)    REAL*8     element of the diagonal preconditioner
c      output:
c cswch='d'
c         G(nb,*) REAL*8     matrix symmetric and definit positive
c         B(*)    REAL*8     Left hand side
c         M(*)    REAL*8     element of the diagonal preconditioner
c cswch='i'
c         X(*)    REAL*8     Solution

cccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
        subroutine precond(cswch,nb,G,B,X,M)
c
        implicit none
c
        integer nb
        real*8 G(nb,*),B(*),X(*),M(*),cond,dd
        integer i,j
        character cswch
c
```

```
         cond=1.d12
c
         if (cswch.eq.'d')then
c
c   Defining M
           dd=0.d0
           do i=1,nb
             M(i)=dabs(G(i,i))
             dd=dmax1(dd,M(i))
           enddo
           do i=1,nb
             if(M(i).lt.dd/cond)M(i)=dd/cond
             M(i)=1.d0/dsqrt(M(i))
           enddo
c
c   Compute M.G.M
           do i=1,nb
             G(i,i)=M(i)*G(i,i)*M(i)
             do j=i+1,nb
               G(i,j)=M(i)*G(i,j)*M(j)
               G(j,i)=G(i,j)
             enddo
c
c   Compute M.B
             B(i)=M(i)*B(i)
           enddo
         else
c
c   Compute X
           do i=1,nb
             X(i)=M(i)*X(i)
           enddo
         endif
c
         return
         end
cccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
c subroutine read_model
c V. Lesur 21/01/2003
```

```
c
c       13/07/04:   modified for refyear in Julian days & gotos removed
c
c       read a model file with a maximum of nl model coefficients
c
c       The first lines of the file may be used for comments.
c       These comment lines should all start with '# '
c       The line before the first data line starts with '##'.
c       The first line after should contain the reference date of the model
c       after, start the model and on each line:
c          a dummy character (usually g or h)
c          two dummy integer (usually l and m)
c          the model parameter value
c          a dummy real (usually the parameter variance)
c
c
c       input:
c         nom file name to read
c         nl            Maximum number of line to read
c
c       output:
c         refyear       ref year for model in Julian days
c         rar           real array (nr)
c         nl            number of data line
c
cccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
subroutine read_model(nom,refyear,rar,nl)
c
        implicit none
c
character nom*100,ligne*80,cd
        integer nl,i,id
        real*8 rar(*),rd,refyear,ry
c
open(10,file=nom,status='old')
c
        ligne(1:2)='  '
        do while (ligne(1:2).ne.'##')
          read(10,*)ligne
```

```
        enddo
c
        read(10,*)ry
        call dy2mjd(ry,refyear)
c
        i=0
        do while (i.le.nl)
          i=i+1
          read(10,*,END=22222)cd,id,id,rar(i),rd
        enddo
c
        write(*,*)'ERROR READING INPUT FILE (NL > NLMAX)'
        close(10)
        stop
c
22222   nl=i-1
        close(10)
c
        return
        end
cccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
c       subroutine set_FG_sampling
c                                    V. Lesur 07/07/2006
c
c       Calculate the sampling points and weights for the
c       Fourier-Gauss sampling theorem on the Sphere
c
c       The number of sampling point is np=(lmax+1)*(2*lmax+1)
c       input:
c          lmax     INTEGER    Maximum spherical harmonic degree
c       output:
c          np       INTEGER    Number of sampling points
c          vrt(2,*) REAL*8     Array of vertex positions
c          wght(*)  Real*8     Weight associated with each vertex
c
cccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
        subroutine set_FG_sampling(lmax,np,vrt,wght)
c
        implicit none
```

```
c
        real*8 vrt(2,*),wght(*)
        integer lmax,np
c
        real*8 dw1,dw2
        integer nn,ilo,ila
        real*8, allocatable :: xx(:),ww(:)
c
        real*8 dacosd
        external dacosd
c
        nn=lmax+1
        dw1=-1.0d0
        dw2=1.0d0
c
c  Find the zeros and weight
        allocate (xx(1:nn),ww(1:nn))
        call gauleg(dw1,dw2,xx,ww,nn)
c
c  Defind sampling in Longitude
        dw2=360.0d0/dble(2*nn-1)
c
c  Define the grid
        np=0
        do ila=1,nn
          do ilo=1,2*nn-1
            np=np+1
c
            vrt(1,np)=dacosd(xx(ila))
            vrt(2,np)=dble(ilo-1)*dw2
            wght(np)=ww(ila)
c
          enddo
        enddo
c
        deallocate (xx,ww)
        return
        end
cccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
```

```
c       subroutine set_FG_sampling
c                                    V. Lesur 07/07/2006
c
c       Calculate the sampling points and weights for the
c       Fourier-Gauss sampling theorem on the Sphere
c
c       The number of sampling point is np=(lmax+1)*(2*lmax+1)
c       input:
c          lmax     INTEGER    Maximum spherical harmonic degree
c       output:
c          np       INTEGER    Number of sampling points
c          vrt(2,*) REAL*8     Array of vertex positions
c          wght(*)  Real*8     Weight associated with each vertex
c
cccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
        subroutine set_FG_sampling_rsc(lmax,np,vrt,lonx,lonn,latx,latn)
c
        implicit none
c
        real*8 vrt(2,*),lonx,lonn,latx,latn
        integer lmax,np
c
        real*8 dw1,dw2,dlon,dlat
        integer nn,ilo,ila
        real*8, allocatable :: xx(:),ww(:)
c
        real*8 dacosd
        external dacosd
c
        nn=lmax+1
        dw1=-1.0d0
        dw2=1.0d0
        dlon=lonx-lonn
        dlat=latx-latn
        write(*,'(6f8.2)')lonx,lonn,latx,latn,dlon,dlat
c
c  Find the zeros and weight
        allocate (xx(1:nn),ww(1:nn))
        call gauleg(dw1,dw2,xx,ww,nn)
```

```
c
c  Defind sampling in Longitude
        dw2=360.0d0/dble(2*nn-1)
c
c  Define the grid
        np=0
        do ila=1,nn
          do ilo=1,2*nn-1
            np=np+1
c
            vrt(1,np)=dacosd(xx(ila))
            vrt(2,np)=dble(ilo-1)*dw2
c Shift centres to ROI
            vrt(1,np)=vrt(1,np)*dlat/180.0d0+latn
      vrt(2,np)=vrt(2,np)*dlon/360.0d0+lonn
          enddo
        enddo
        write(*,*)dlat
c
        deallocate (xx,ww)
        return
        end
           FUNCTION REPS(i)
C       DONNE LA PRECISION MACHINE EN SIMPLE PRECISION
           REPS=1.E-3
   10      TEST=1.E0+REPS
           IF (TEST.GT.1.E0)THEN
             REPS=REPS/2.
             GOTO 10
           ENDIF
           REPS=REPS*2.E0
           RETURN
           END
C
C
C
           FUNCTION DEPS(i)
C       DONNE LA PRECISION MACHINE EN DOUBLE PRECISION
           IMPLICIT REAL*8 (D)
```

```
      DEPS=1.D-3
 20   DTEST=1.D0+DEPS
      IF (DTEST.GT.1.D0)THEN
        DEPS=DEPS/2.
        GOTO 20
      ENDIF
      DEPS=DEPS*10.D0
      RETURN
      END
C
C
C
      FUNCTION RUF(i)
C     PLUS PETIT REEL SIMPLE PRECISION POSITIF
C     DEPENT DE LA MACHINE
c     RUF=1.17549429E-38
      RUF=1.17549429E-37
      RETURN
      END
C
C
      FUNCTION DUF(i)
C     PLUS PETIT REEL DOUBLE PRECISION POSITIF
C     DEPEND DE LA MACHINE
      IMPLICIT REAL*8 (D)
      DUF=2.2250738585072012D-308
      RETURN
      END
C
C
C
      FUNCTION ROF(i)
C     PLUS GRAND REEL SIMPLE PRECISION POSITIF
C     DEPENT DE LA MACHINE
c     ROF=3.40282356E38
      ROF=3.40282356E37
      RETURN
      END
C
```

```
C
          FUNCTION DOF(i)
C       PLUS GRAND REEL DOUBLE PRECISION POSITIF
C       DEPEND DE LA MACHINE
          IMPLICIT REAL*8 (D)
c         DOF=1.7976931348623158D308
          DOF=1.7976931348623158D307
          RETURN
          END
ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
c subroutine SPH_GEO
c
c       V. Lesur 14/09/05
c
c Transform a vector in GEO system of coordinates from a vector in
c SPH system of coordinates.
c
c CALLED: ortho_rot.f
c
c input:
c         vx,vy,vx    (REAL*8)  vector coordinates in SPH
c         dclt,dln    (REAL*8)  Colat and Long of centre SPH system
c
c output:
c         vx,vy,vx    (REAL*8)  vector coordinates in GEO
c
ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
        subroutine SPH_GEO(dclt,dln,vx,vy,vz)
c
        implicit none
c
        real*8 dclt,dln,vx,vy,vz
c
c   Apply rotation around Y and then Z
        call ortho_rot(2,dclt,vx,vy,vz)
        call ortho_rot(3,dln,vx,vy,vz)
c
        return
        end
```

```
c-----------------------------------------------------------------------
c            procedure trig (tri-diagonalisation par rotation de Givens)
c        vincent Lesur   29.03.2001
c
c        modified: 16/04/2004  1- enddo replace labels+continue
c                              2- all implicite to real*8
c
c        Calcul les matrices Q et T tq QAQ'=T pour A matrice reelle symetrique
c        ou Q est une matrice de rotation unitaire (Givens)
c           Q' est la transposee de Q
c        et T est une matrice tridiagonale (symetrique reelle).
c _       _
c |alpha beta  |
c     la matrice Q elementaire  Q= |       |
c |beta -alpha|
c -      -
c       input: da matrice symmetric carre de dim NXN
c              nn       N
c
c       output: da      matrice symmetric tridiagonal. triangle inferieur
c                       utilise pour stocker les rotations. triangle
c                       superieur mis a zero
c
c-----------------------------------------------------------------------
             subroutine trig(da,nn)
c
        implicit none
c
        real*8
     +       da(*),depsi,dsg,alpha,beta,gama,dvar
        integer*4
     +       nn,k,k1,k2,i,i1,i2,j,jj
c
        real*8 deps
        external deps
c
        depsi=deps(1)
c
c     debut
```

```
c
c      boucle sur les colonnes
        do k=1,nn-2
          k1=1+k+(k-1)*nn  !element sub-diag
          k2=k+k*nn  !element sup-diag
          dsg=dsign(1.0d0,da(k1))
c
c      boucle sur les lignes (K+2 a MM pour tri_diag)
          do i=k+2,nn
            i1=i+(k-1)*nn
            i2=k+(i-1)*nn
c
            if(dabs(da(i1)).gt.depsi) then
              dvar=dsqrt((da(k1))**2+(da(i1))**2)
              alpha=dabs(da(k1)/dvar)
              beta=dsg*da(i1)/dvar
c
c     stokage et affectation
              if(dabs(da(k1)).eq.0.d0) then
                da(i1)=dsign(1.0d0,da(i1))
              else
                if(dabs(da(k1)).gt.dabs(da(i1)))then
                  da(i1)=da(i1)/da(k1)/2.d0
                else
                  da(i1)=2.d0*da(i1)/da(k1)
                endif
              endif
              da(i2)=0.0d0
              da(k1)=dsg*dvar
              da(k2)=da(k1)
c
c     rotation sur les colonnes k,k+1,k+2,...
              do j=1,nn-k
                jj=j*nn
                gama=alpha*da(k1+jj)+beta*da(i1+jj)
                da(i1+jj)=beta*da(k1+jj)-alpha*da(i1+jj)
                da(k1+jj)=gama
              enddo
c
```

```
c     rotation transposee sur les lignes
              gama=alpha*da(k2+1)+beta*da(i2+1)
              da(i2+1)=beta*da(k2+1)-alpha*da(i2+1)
              da(k2+1)=gama
              gama=beta*da(k2+i-k)-alpha*da(i2+i-k)
              do j=2,nn-k
                jj=j*nn
                da(i2+j)=da(i1+jj)
                da(k2+j)=da(k1+jj)
              enddo
              da(i2+i-k)=gama
c
            else
              da(i1)=0.0d0
              da(i2)=0.0d0
            endif
          enddo
        enddo
c
        return
        end
cccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
c subroutine write_comp
c V. Lesur 07/03/2002
c
c       write a file of nl data line.
c       The line before the first data line will be '##########'
c       previous lines must start with a '#' and may contain comments
c       data lines have nd+3 real records:
c       point position (nd real), data, fit, (data-fit)
c       The data point printed are selected by type
c
c       input:
c         nom     file name to write
c         nl            number of data line to write
c         npm           number max of data point
c         nd            number of space dimmension
c         nt(*)         data type (dim nl)
c         ts            type selected
```

```
c         ddat(nd+1,*)  position and data (dim (nd+1,npm))
c         xyzf(*)       fit to data (dim nl)
c
ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
      subroutine write_comp(nom,nl,npm,nd,nt,ts,ddat,xyzf)
c
        implicit none
c
      character nom*100,theformat*15
        integer i,j,nl,npm,nd,nt(*),ts,nb,nd1,nd2
        real*8 ddat(*),xyzf(*),adif,sdif
c
      open(10,file=nom,status='unknown')
c
        write(10,'(A)')'##########'
c
        nb=0
        adif=0.0d0
        sdif=0.0d0
        nd1=int((nd+3)/10.)
        nd2=(nd+3)-nd1*10
        theformat='('//char(nd2+48)//'e15.7)'
        if(nd1.ne.0)theformat='('//char(nd1+48)//char(nd2+48)//'e15.7)'
        nd1=nd+1
c
        i=0
        do while (i.lt.nl)
          i=i+1
          if(nt(i).eq.ts)then
            nb=nb+1
            write(10,theformat)(ddat(j+(i-1)*nd1),j=1,nd1)
     >                ,xyzf(i),ddat(i*nd1)-xyzf(i)
            adif=adif+ddat(i*nd1)-xyzf(i)
            sdif=sdif+(ddat(i*nd1)-xyzf(i))**2
          endif
        enddo
c
        close(10)
c
```

```
        adif=adif/nb
        sdif=dsqrt((sdif-nb*adif**2)/nb)
c
        write(*,'(2(A,f11.2))')'Residual average : ',adif
     >                       ,' with L2 std : ',sdif
c
        return
        end
cccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
c subroutine write_comp
c V. Lesur 07/03/2002
c
c       write a file of nl data line.
c       The line before the first data line will be '##########'
c       previous lines must start with a '#' and may contain comments
c       data lines have nd+3 real records:
c       point position (nd real), data, fit, (data-fit)
c       The data point printed are selected by type
c
c       input:
c         nom file name to write
c         nl            number of data line to write
c         npm           number max of data point
c         nd            number of space dimmension
c         nt(*)         data type (dim nl)
c         ts            type selected
c         ddat(nd+1,*)  position and data (dim (nd+1,npm))
c         xyzf(*)       fit to data (dim nl)
c
cccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
subroutine write_comp(nom,nl,npm,nd,nt,ts,ddat,xyzf)
c
        implicit none
c
character nom*100,theformat*15
        integer i,j,nl,npm,nd,nt(*),ts,nb,nd1,nd2
        real*8 ddat(*),xyzf(*),adif,sdif,dy
c
open(10,file=nom,status='unknown')
```

```
c
c        write(10,'(A)')'##########'
c
         nb=0
         adif=0.0d0
         sdif=0.0d0
         nd1=int((nd+3)/10.)
         nd2=(nd+3)-nd1*10
         theformat='('//char(nd2+48)//'e15.7)'
         if(nd1.ne.0)theformat='('//char(nd1+48)//char(nd2+48)//'e15.7)'
         nd1=nd+1
c
         i=0
         do while (i.lt.nl)
           i=i+1
           if(nt(i).eq.ts)then
             nb=nb+1
             call mjd2dy(ddat(4+(i-1)*nd1),dy)
             write(10,theformat)(ddat(j+(i-1)*nd1),j=1,3),dy
                        ,(ddat(j+(i-1)*nd1),j=5,nd1))
      >                 ,xyzf(i),ddat(i*nd1)-xyzf(i)
             adif=adif+ddat(i*nd1)-xyzf(i)
             sdif=sdif+(ddat(i*nd1)-xyzf(i))**2
           endif
         enddo
c
         close(10)
c
         adif=adif/nb
         sdif=dsqrt((sdif-nb*adif**2)/nb)
c
         write(*,'(2(A,f11.2))')'Residual average : ',adif
      >                         ,' with L2 std : ',sdif
c
         return
         end
cccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
c subroutine write_comps
c V. Lesur 07/03/2002
```

```
c           Anne        20/07/2002
c
c       write a file of nl data line.
c       The line before the first data line will be '##########'
c       previous lines must start with a '#' and may contain comments
c       data lines have nd+3 real records:
c       point position (nd real), data, fit, (data-fit)
c       The data point printed are selected by type
c
c       input:
c         nom file name to write
c         nl             number of data line to write
c         npm            number max of data point
c         nd             number of space dimmension
c         nt(*)          data type (dim nl)
c         ddat(nd+1,*)  position and data (dim (nd+1,npm))
c         xyzf(*)        fit to data (dim nl)
c
cccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
        subroutine write_comps(nom,nl,npm,nd,nt,ddat,xyzf,ds,dt,lw)
c
        implicit none
c
        character nom*100,theformat*15
        integer i,j,nl,npm,nd,nt(*),ts,nb,nd1,nd2, lw
        real*8 ddat(*),xyzf(*),adif,sdif,a(3),s(3),ra,dt,ds
c
        open(10,file=nom
     >      ,status='old',access='append')
c       write(10,'(A)')"#lw Residual average x,y,z with L2 std in x,y,z"
c       write(10,'(A)')'##########'
c
        do ts=1,3
        a(ts)=0.0d0
        s(ts)=0.0d0
        nb=0
        adif=0.0d0
        sdif=0.0d0
        nd1=int((nd+3)/10.)
```

```
      nd2=(nd+3)-nd1*10
      theformat='('//char(nd2+48)//'e15.7)'
      if(nd1.ne.0)theformat='('//char(nd1+48)//char(nd2+48)//'e15.7)'
      nd1=nd+1
c
      i=0
      do while (i.lt.nl)
        i=i+1
        if(nt(i).eq.ts)then
          nb=nb+1
c          write(10,theformat)(ddat(j+(i-1)*nd1),j=1,nd1)
c    >                ,xyzf(i),ddat(i*nd1)-xyzf(i)
          adif=adif+ddat(i*nd1)-xyzf(i)
          sdif=sdif+(ddat(i*nd1)-xyzf(i))**2
        endif
      enddo
c
      adif=adif/nb
      sdif=dsqrt((sdif-nb*adif**2)/nb)
c
      a(ts)=adif
      s(ts)=sdif
c      write(*,'(2(A,f11.2))')'*Residual average : ',a(ts)
c    >                        ,' with L2 std : ',s(ts)
      end do
c
      write(10,'(2e15.5,i5,6f15.7)')ds,dt,lw,(a(i),i=1,3),(s(i),i=1,3)
      close(10)
      return
      end
ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
c subroutine write_model4
c V. Lesur 27/05/2006
c
c       write a file of with a maximum of nl data line.
c       For each line the records are:
c          a character (g or h)
c          two integer (l and m)
c          the model coeff value
```

```
c          a dummy real
c
c       The line before the first data line starts with '##'.
c       The first lines of the file may be used for comments.
c       These comment lines should all start with '# '
c
c       input:
c         nom file name to write
c         nb            number of coeff
c         model(*)      model to write
c         var(*)        formal std associated with model
c         refyear       reference year in julian days
c         std           calculated STD
c
ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
        subroutine write_model5(nom,nb,refyear,model,var,std)
c
        implicit none
c
character nom*100
        integer nb,i
        real*8 model(*),var(*),refyear,ry,std
c
open(10,file=nom)
c
        write(10,'(A,i5)')'# Number of coeff   :',nb
        write(10,'(A,e17.5)')'# STD to the data  :',std
        write(10,'(A)')'############'
c
        call mjd2dy(refyear,ry)
        write(10,'(f11.2)')ry
c
        do i=1,nb
          write(10,'(A,2I5,2f20.6)')'x',i,0,model(i)
     >                                       ,var(i)
        enddo
c
        close(10)
        return
```

```
        end
cccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
c Subroutine Xsph_bi0
c Vincent Lesur 30/04/2002
c       For a all spherical harmonique between degree 1 and nlie
c       and a  colatitude, longitude, radius computes for
c       internal sources:
c       X component basis  (X horizontal North)
c
c       These basis is the derivative of a potential
c       defined on the sphere of radius "ra"
c       (See Foundations of Geomagnetism, Backus 1996 page 110 and 125)
c
c       No time dependence
c
c       Spherical harmonics are set by increasing degree "l", for
c       a given degree by increasing order "m" (l=0,m=0 excluded)
c       nu=(l**2+2*(m-1))   FOR cosine terms if m.ne.0
c       nu=(l**2+2*m-1)     For sine terms or m=0
c
c       colat,long,radius given in degree,degree,km
c
c       input:
c         nlie          end of degree internal
c         pos           positions in space and time (colat,long,radius)
c
c       output:
c         be Base funtion value
c
cccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
subroutine Xsph_bi0(nlie,pos,be)
c
        implicit none
c
        integer nu,il,im,nlie
        real*8 ry,pos(*),be(*),rc,rs,ra,dw
        real*8 ds,dc,dt,dcosd,dsind
        real*8, allocatable :: ddlf(:)
c
```

```
      external dcosd,dsind
      allocate(ddlf(1:nlie+1))
c
      ra=6371.2d0
      rc=dcosd(pos(1))
      rs=dsind(pos(1))
c
c   im=0
      im=0
      call mkdlf_F(im,nlie,rs,rc,ddlf)
      do il=1,nlie
        nu=(il*il-1)
        be(nu+1)=ddlf(il-im+1)*(ra/pos(3))**(il+2)
      enddo
c   im.ne.0
      do im=1,nlie
        call mkdlf_F(im,nlie,rs,rc,ddlf)
        dc=dcosd(im*pos(2))
        ds=dsind(im*pos(2))
        do il=im,nlie
          dw=ddlf(il-im+1)*(ra/pos(3))**(il+2)
          nu=il*il+2*(im-1)
          be(nu+1)=dw*dc
          be(nu+2)=dw*ds
        enddo
      enddo
      deallocate(ddlf)
c
      return
      end
ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
c             subroutine XYlocal_bt.f   V. Lesur 05/02/2007
c
c       derived from localc_XYZ_g.f     V. Lesur  19 June 2005
c
c       For harmonic functions i=is,ie defined by:
c       F_i(clat,lon) = ra SUM(l=0:Lm) alpha(l)
c                          {SUM(m=0:l) Y*lm(clat_i,lon_i) Ylm(clat,lon)}
c      computes their associated toroidal field along:
```

```
c      X (X horizontal North), Y (Y horizontal East)  at (clat,lon).
c      The Z (Z vectical down) component is always =0
c
c      if the angle between (clat_i,lon_i) and (clat,lon) is larger than
c      cuta the field is assumed to be 0
c
c      The field values along XY are saved in the arrays Xv,Yv with
c      dim. min :  nb=ie.
c
c      CALL to:
c       mkdlf_F0 that computes a Legendre polynomial derivative
c
c       input:
c         is/ie   INTEGER      starting/ending center for F_i(clat,lon)
c         lw      INTEGER      Maximum SH degree in Wavelet
c         ppos(3) REAL*8       array of positions in space
c                              ppos(1)=colat (deg)
c                              ppos(2)=long  (deg)
c         alph(*)  REAL*8      array of alpha values
c         cntr(2,*)REAL*8      array of functions centers
c         cuta     REAL*8      cut angle
c
c       output:
c         xv(nb) REAL*8   Toroidal field in North direction
c         yv(nb) REAL*8   Toroidal field in East direction
c
cccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
        subroutine XYlocal_bt(is,ie,lw,alph,cntr,cuta,ppos,xv,yv)
c
        implicit none
c
        integer is,ie,lw
        real*8 ppos(*),xv(*),yv(*),alph(*),cntr(2,*),cuta
c
        integer i,j,ic
        real*8 dth,dph,dx,dy,dz,dw
        real*8 dxn,dyn,dzn,dxd,dyd,dzd
        real*8 dxe,dye,dze,dgth
        real*8 dst,dct,dsp,dcp,dsti,dcti,dspi,dcpi
```

```
      real*8 dcosd,dsind,ccuta
      real*8, allocatable :: ddlf(:)
c
      allocate (ddlf(1:lw+1))
c
c  Set initial valuyes
      ccuta=dcosd(cuta)
c
      dth=ppos(1)
      dph=ppos(2)
      dst=dsind(dth)
      dct=dcosd(dth)
      dsp=dsind(dph)
      dcp=dcosd(dph)
c
c  Position of data point in cartesian soc
      dxd=dst*dcp
      dyd=dst*dsp
      dzd=dct
c
c  Direction of North vector in cartesian soc
      dxn=-dct*dcp
      dyn=-dct*dsp
      dzn=dst
c
c  Direction of East vector in cartesian soc
      dxe=-dsp
      dye=dcp
      dze=0.0d0
c
c   For each F centre
      do ic=is,ie
c
        dth=cntr(1,ic)
        dph=cntr(2,ic)
        dsti=dsind(dth)
        dcti=dcosd(dth)
        dspi=dsind(dph)
        dcpi=dcosd(dph)
```

```
c
c   Position of data point in soc where Z pointing to (dth,dph)
         dw=dxd*dcpi+dyd*dspi
         dy=-dxd*dspi+dyd*dcpi
         dx=dcti*dw-dzd*dsti
         dz=dsti*dw+dzd*dcti
c
c   COS and SIN of data point in soc where Z pointing to (dth,dph)
         dct=dz
         dst=dsqrt(1.d0-dct**2)
         dcp=0.0d0
         dsp=0.0d0
         if(dst.ne.0.0d0)then
           dcp=dx/dst
           dsp=dy/dst
         endif
c
c   Theta Gradient of F at data point
         if(dct.ge.ccuta)then
           call mkdlf_F0(lw,dst,dct,ddlf)
           dgth=0.0d0
           do j=2,lw+1                              !j=l+1
             dgth=dgth+ddlf(j)*alph(j)
           enddo
c
c   North vector in soc where Z pointing to (dth,dph)
           dw=dxn*dcpi+dyn*dspi
           dy=-dxn*dspi+dyn*dcpi
           dx=dcti*dw-dzn*dsti
c
c   Calculate the rotational in (dx,dy,dz) (i.e. North) direction
           xv(ic)=-dsp*dx+dcp*dy
           xv(ic)=-dgth*xv(ic)
c
c   East vector in soc where Z pointing to (dth,dph)
           dw=dxe*dcpi+dye*dspi
           dy=-dxe*dspi+dye*dcpi
           dx=dcti*dw-dze*dsti
c
```

```
c   Calculate the rotational in (dx,dy,dz) (i.e. East) direction
            yv(ic)=-dsp*dx+dcp*dy
            yv(ic)=-dgth*yv(ic)
          else
c
c    gardients assumed to be zero
            xv(ic)=0.0d0
            yv(ic)=0.0d0
          endif
        enddo
c
        deallocate (ddlf)
c
        return
        end
cccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
c             subroutine XYZlocal_bio.f   V. Lesur 05/02/2007
c
c       derived from localc_XYZ_g.f     V. Lesur  19 June 2005
c
c       For the current harmonic functions i=is,ie defined at
c       radius ri=6481.3km (i.e.110km altitude) by:
c       F_i(clat,lon) = SUM(l=0:Lm) alpha(l)
c                          {SUM(m=0:l) Y*lm(clat_i,lon_i) Ylm(clat,lon)}
c       computes the associated magnetic field along X (X horizontal
c       North), Y (Y horizontal East) and Z (Z vertical down) at
c       (clat,lon,radius).
c
c      if the angle between (clat_i,lon_i) and (clat,lon) is larger than
c      cutag the current are assumed to be 0
c
c      The magnetic field along XYZ are saved in the arrays Xv,Yv,Zv
c      with dim. min :  nb=ie
c
c      CALL to:
c       mk_lf_dlf_0 that computes a Legendre polynomial and its
c       derivative
c
c       input:
```

```
c         is/ie   INTEGER      starting/ending center for F_i(clat,lon)
c         lw      INTEGER      Maximum SH degree in Wavelet
c         ppos(3) REAL*8       array of positions in space
c                              ppos(1)=colat (deg)
c                              ppos(2)=long  (deg)
c                              ppos(3)=radius (km)
c         alph(*)  REAL*8      array of alpha values
c         cntr(2,*)REAL*8      array of functions centers
c         cuta     REAL*8      cut angle
c
c       output:
c         xv(nb) REAL*8   ionosphere field in North direction
c         yv(nb) REAL*8   ionosphere field in East direction
c         zv(nb) REAL*8   ionosphere field in East direction
c
cccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
        subroutine XYZlocal_bio(is,ie,lw,alph,cntr,cuta,ppos,xv,yv,zv)
c
        implicit none
c
        integer is,ie,lw
        real*8 ppos(*),xv(*),yv(*),zv(*),alph(*),cntr(2,*),cuta
c
        integer i,j,ic
        real*8 ri,rr,dth,dph,dx,dy,dz,dw
        real*8 dxn,dyn,dzn,dxd,dyd,dzd
        real*8 dxe,dye,dze,dgth,dgr
        real*8 dst,dct,dsp,dcp,dsti,dcti,dspi,dcpi
        real*8 dcosd,dsind,ccuta
        real*8, allocatable :: ddlf(:),dlf(:)
c
        allocate(ddlf(1:lw+1),dlf(1:lw+1))
c
c  Set initial valuyes
        ri=6481.2d0
        ccuta=dcosd(cuta)
c
        dth=ppos(1)
        dph=ppos(2)
```

```
      dst=dsind(dth)
      dct=dcosd(dth)
      dsp=dsind(dph)
      dcp=dcosd(dph)
c
c  Position of data point in cartesian soc
      dxd=dst*dcp
      dyd=dst*dsp
      dzd=dct
c
c  Direction of North vector in cartesian soc
      dxn=-dct*dcp
      dyn=-dct*dsp
      dzn=dst
c
c  Direction of East vector in cartesian soc
      dxe=-dsp
      dye=dcp
      dze=0.0d0
c
c   For each F centre
      do ic=is,ie
c
        dth=cntr(1,ic)
        dph=cntr(2,ic)
        dsti=dsind(dth)
        dcti=dcosd(dth)
        dspi=dsind(dph)
        dcpi=dcosd(dph)
c
c   Position of data point in soc where Z pointing to (dth,dph)
        dw=dxd*dcpi+dyd*dspi
        dy=-dxd*dspi+dyd*dcpi
        dx=dcti*dw-dzd*dsti
        dz=dsti*dw+dzd*dcti
c
c   COS and SIN of data point in soc where Z pointing to (dth,dph)
        dct=dz
        dst=dsqrt(1.d0-dct**2)
```

```
          dcp=0.0d0
          dsp=0.0d0
          if(dst.ne.0.0d0)then
            dcp=dx/dst
            dsp=dy/dst
          endif
c
c   Theta and Radial Gradient of F at data point
          if(dct.ge.ccuta)then
            call mk_lf_dlf_0(lw,dst,dct,dlf,ddlf)
            dgth=0.0d0
            dgr=0.0d0
            if(ri.gt.ppos(3))then
              rr=ppos(3)/ri
              do j=2,lw+1                              !j=l+1
                dw=alph(j)*rr**(j-2)
                dgth=dgth-dble(j)*ddlf(j)*dw/dble(2*j-1)
                dgr=dgr+dble(j)*dble(j-1)*dlf(j)*dw/dble(2*j-1)
              enddo
            else
              rr=ri/ppos(3)
              do j=2,lw+1                              !j=l+1
                dw=alph(j)*rr**(j+1)
                dgth=dgth+dble(j-1)*ddlf(j)*dw/dble(2*j-1)
                dgr=dgr+dble(j-1)*dble(j)*dlf(j)*dw/dble(2*j-1)
              enddo
            endif
c
c   North vector in soc where Z pointing to (dth,dph)
            dw=dxn*dcpi+dyn*dspi
            dy=-dxn*dspi+dyn*dcpi
            dx=dcti*dw-dzn*dsti
            dz=dsti*dw+dzn*dcti
c
c   Calculate the gradient in (dx,dy,dz) (i.e. North) direction
            xv(ic)=dct*dcp*dx+dct*dsp*dy-dst*dz
            xv(ic)=-dgth*xv(ic)
c
c   East vector in soc where Z pointing to (dth,dph)
```

```
             dw=dxe*dcpi+dye*dspi
             dy=-dxe*dspi+dye*dcpi
             dx=dcti*dw-dze*dsti
             dz=dsti*dw+dze*dcti
c
c   Calculate the gradient in (dx,dy,dz) (i.e. East) direction
             yv(ic)=dct*dcp*dx+dct*dsp*dy-dst*dz
             yv(ic)=-dgth*yv(ic)
c
c   Calculate the gradient in radial direction
             zv(ic)=-dgr
           else
c
c    gardients assumed to be zero
             xv(ic)=0.0d0
             yv(ic)=0.0d0
             zv(ic)=0.0d0
           endif
         enddo
c
         deallocate(ddlf,dlf)
c
         return
         end
cccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
c             subroutine XYZlocal_core.f   V. Lesur 05/05/2008
c
c       derived from localc_XYZ_g.f     V. Lesur  19 June 2005
c
c       For the harmonic functions i=is,ie defined at radius ra=3485.km
c       (i.e. Earth's core radius) by:
c       F_i(clat,lon) = ra SUM(l=0:Lm) alpha(l) (ra/r)**(l+1)
c         (ra/r_i)**(l+2) {SUM(m=0:l) Y*lm(clat_i,lon_i) Ylm(clat,lon)}
c       computes the associated magnetic field along X (X horizontal
c       North), Y (Y horizontal East) and Z (Z vertical down) at
c       (clat,lon,radius).
c
c      if the angle between (clat_i,lon_i) and (clat,lon) is larger than
c      cutag the current are assumed to be 0
```

```
c
c      The magnetic field along XYZ are saved in the arrays Xv,Yv,Zv
c      with dim. min :  nb=ie
c
c      CALL to:
c       mk_lf_dlf_0 that computes a Legendre polynomial and its
c       derivative
c
c       input:
c         is/ie   INTEGER      starting/ending center for F_i(clat,lon)
c         lw      INTEGER      Maximum SH degree in Wavelet
c         ppos(3) REAL*8       array of positions in space
c                              ppos(1)=colat (deg)
c                              ppos(2)=long  (deg)
c                              ppos(3)=radius (km)
c         alph(*)  REAL*8      array of alpha values
c         cntr(3,*)REAL*8      array of functions centers (colat,long,radius)
c         ra       REAL*8      function reference radius
c         cuta     REAL*8      cut angle
c
c       output:
c         xv(nb) REAL*8   crustal field in North direction
c         yv(nb) REAL*8   crustal field in East direction
c         zv(nb) REAL*8   crustal field in East direction
c
cccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
        subroutine XYZlocal_core(is,ie,lw,alph,cntr,ra,cuta,ppos
     >                                                     ,xv,yv,zv)
c
        implicit none
c
        integer is,ie,lw
        real*8 ppos(*),xv(*),yv(*),zv(*),alph(*),cntr(3,*),ra,cuta
c
        integer i,j,ic
        real*8 rr,rc,dth,dph,dx,dy,dz,dw,dd
        real*8 dxn,dyn,dzn,dxd,dyd,dzd
        real*8 dxe,dye,dze,dgth,dgr
        real*8 dst,dct,dsp,dcp,dsti,dcti,dspi,dcpi
```

```
      real*8 dcosd,dsind,ccuta
      real*8, allocatable :: ddlf(:),dlf(:)
c
      allocate(ddlf(1:lw+1),dlf(1:lw+1))
c
c  Set initial valuyes
      ccuta=dcosd(cuta)
c
      dth=ppos(1)
      dph=ppos(2)
      dst=dsind(dth)
      dct=dcosd(dth)
      dsp=dsind(dph)
      dcp=dcosd(dph)
c
c  Position of data point in cartesian soc
      dxd=dst*dcp
      dyd=dst*dsp
      dzd=dct
c
c  Direction of North vector in cartesian soc
      dxn=-dct*dcp
      dyn=-dct*dsp
      dzn=dst
c
c  Direction of East vector in cartesian soc
      dxe=-dsp
      dye=dcp
      dze=0.0d0
c
c   For each F centre
      do ic=is,ie
c
        dth=cntr(1,ic)
        dph=cntr(2,ic)
        dsti=dsind(dth)
        dcti=dcosd(dth)
        dspi=dsind(dph)
        dcpi=dcosd(dph)
```

```
c
c   Position of data point in soc where Z pointing to (dth,dph)
          dw=dxd*dcpi+dyd*dspi
          dy=-dxd*dspi+dyd*dcpi
          dx=dcti*dw-dzd*dsti
          dz=dsti*dw+dzd*dcti
c
c   COS and SIN of data point in soc where Z pointing to (dth,dph)
          dct=dz
          if(dct.gt.1.d0)dct=1.d0
          dst=dsqrt(1.d0-dct**2)
          dcp=0.0d0
          dsp=0.0d0
          if(dst.ne.0.0d0)then
            dcp=dx/dst
            dsp=dy/dst
          endif
c
c   Theta and Radial Gradient of F at data point
          if(dct.ge.ccuta)then
            call mk_lf_dlf_0(lw,dst,dct,dlf,ddlf)
            dgth=0.0d0
            dgr=0.0d0
            rr=ra/ppos(3)
            rc=ra/cntr(3,ic)
            do j=2,lw+1                              !j=l+1
              dw=rc**(j+1)
              dw=dw*alph(j)*rr**(j+1)
              dgth=dgth+ddlf(j)*dw
              dgr=dgr+dble(j)*dlf(j)*dw
            enddo
c
c   North vector in soc where Z pointing to (dth,dph)
            dw=dxn*dcpi+dyn*dspi
            dy=-dxn*dspi+dyn*dcpi
            dx=dcti*dw-dzn*dsti
            dz=dsti*dw+dzn*dcti
c
c   Calculate the gradient in (dx,dy,dz) (i.e. North) direction
```

```
            xv(ic)=dct*dcp*dx+dct*dsp*dy-dst*dz
            xv(ic)=-dgth*xv(ic)
c
c   East vector in soc where Z pointing to (dth,dph)
            dw=dxe*dcpi+dye*dspi
            dy=-dxe*dspi+dye*dcpi
            dx=dcti*dw-dze*dsti
            dz=dsti*dw+dze*dcti
c
c   Calculate the gradient in (dx,dy,dz) (i.e. East) direction
            yv(ic)=dct*dcp*dx+dct*dsp*dy-dst*dz
            yv(ic)=-dgth*yv(ic)
c
c   Calculate the gradient in radial direction
            zv(ic)=-dgr
          else
c
c    gardients assumed to be zero
            xv(ic)=0.0d0
            yv(ic)=0.0d0
            zv(ic)=0.0d0
          endif
        enddo
c
        deallocate(ddlf,dlf)
c
        return
        end
cccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
c             subroutine XYZlocal_coret.f   V. Lesur 05/05/2008
c
c       For the harmonic functions i=is,ie defined at radius ra=3485.km
c       (i.e. Earth's core radius) by:
c       F_i(clat,lon) = ra SUM(l=0:Lm) alpha(l) (ra/r)**(l+1)
c         (ra/r_i)**(l+2) {SUM(m=0:l) Y*lm(clat_i,lon_i) Ylm(clat,lon)}
c       computes the associated magnetic field along X (X horizontal
c       North), Y (Y horizontal East) and Z (Z vertical down) at
c       (clat,lon,radius).
c
```

```
c      A time dependence is introduced by B-splines of oder io
c
c      if the angle between (clat_i,lon_i) and (clat,lon) is larger than
c      cutag the current are assumed to be 0
c
c      The magnetic field along XYZ are saved in the arrays Xv,Yv,Zv
c      with dim. min :  nb=ie*(nk-io) where "nk" is the number of
c      spline nodes.
c
c      CALL to:
c       XYZlocal_core.f that computes the same thing without time
c       dependence
c
c       input:
c         io      INTEGER      spline order
c         nk      INTEGER      number of spline knots
c         tk      REAL*8       array of spline knot positions
c         ik      INTEGER      spline knot tk(ik)=< dt <tk(ik+1)
c         ry      REAL*8       reference year
c         is/ie   INTEGER      starting/ending center for F_i(clat,lon)
c         lw      INTEGER      Maximum SH degree in Wavelet
c         ppos(4) REAL*8       array of positions in space
c                              ppos(1)=colat (deg)
c                              ppos(2)=long  (deg)
c                              ppos(3)=radius (km)
c                              ppos(4)=time (MJD)
c         alph(*)  REAL*8      array of alpha values
c         cntr(3,*)REAL*8      array of functions centers (colat,long,radius)
c         ra       REAL*8      function reference radius
c         cuta     REAL*8      cut angle
c
c       output:
c         xv(nb) REAL*8   crustal field in North direction
c         yv(nb) REAL*8   crustal field in East direction
c         zv(nb) REAL*8   crustal field in East direction
c
cccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
       subroutine XYZlocal_coret(io,nk,tk,ik,ry,is,ie,lw,alph,cntr,ra
     >                                            ,cuta,ppos,xv,yv,zv)
```

```
c
      implicit none
c
      integer is,ie,lw,nk,ik,io,nb
      real*8 ppos(*),xv(*),yv(*),zv(*)
      real*8 ry,tk(*),alph(*),cntr(3,*),ra,cuta
c
      integer i,iw,ic,it,itm
      real*8 tw(io),dt
      real*8, allocatable :: dw(:),xx(:),yy(:),zz(:)
c
      itm=nk-io
      nb=ie*itm
      iw=int(float(io+1)*float(io+2)/2.)
      allocate(dw(1:iw),xx(1:ie),yy(1:ie),zz(1:ie))
c
      dt=(ppos(4)-ry)/365.25d0
      xv(1:nb)=0.d0
      yv(1:nb)=0.d0
      zv(1:nb)=0.d0
c
c  Define the vectors without time dependence
      call XYZlocal_core(is,ie,lw,alph,cntr,ra,cuta,ppos,xx,yy,zz)
c
c  Define splines
      call DBSPVD(tk,io,1,dt,ik,io,tw,dw)
c
c  define XV,YV,ZV
      do ic=is,ie
        do i=1,io
          it=ik-io+i
          xv((ic-1)*itm+it)=xx(ic)*tw(i)
          yv((ic-1)*itm+it)=yy(ic)*tw(i)
          zv((ic-1)*itm+it)=zz(ic)*tw(i)
        enddo
      enddo
c
      deallocate(dw,xx,yy,zz)
      return
```

```
      end
cccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
c            subroutine XYZlocal_coretd.f   V. Lesur 05/11/2008
c
c       For the harmonic functions i=is,ie defined at radius ra=3485.km
c       (i.e. Earth's core radius) by:
c       F_i(clat,lon) = ra SUM(l=0:Lm) alpha(l) (ra/r)**(l+1)
c         (ra/r_i)**(l+2) {SUM(m=0:l) Y*lm(clat_i,lon_i) Ylm(clat,lon)}
c       computes the associated magnetic field along X (X horizontal
c       North), Y (Y horizontal East) and Z (Z vertical down) at
c       (clat,lon,radius) and its time derivatives.
c
c      A time dependence is introduced by B-splines of order io
c
c      if the angle between (clat_i,lon_i) and (clat,lon) is larger than
c      cutag the current are assumed to be 0
c
c      The magnetic field along XYZ are saved in the arrays Xv,Yv,Zv
c      with dim. min :  nb=ie*(nk-io) where "nk" is the number of
c      spline nodes.
c
c      CALL to:
c       XYZlocal_core.f that computes the same thing without time
c       dependence
c
c       input:
c         io      INTEGER      spline order
c         nk      INTEGER      number of spline knots
c         tk      REAL*8       array of spline knot positions
c         ik      INTEGER      spline knot tk(ik)=< dt <tk(ik+1)
c         ry      REAL*8       reference year
c         is/ie   INTEGER      starting/ending center for F_i(clat,lon)
c         lw      INTEGER      Maximum SH degree in Wavelet
c         ppos(4) REAL*8       array of positions in space
c                              ppos(1)=colat (deg)
c                              ppos(2)=long  (deg)
c                              ppos(3)=radius (km)
c                              ppos(4)=time (MJD)
c         alph(*)  REAL*8      array of alpha values
```

```
c         cntr(3,*)REAL*8      array of functions centers (colat,long,radius)
c         ra       REAL*8      function reference radius
c         cuta     REAL*8      cut angle
c         idrv     INTEGER     derivative number
c
c       output:
c         xv(nb) REAL*8   crustal field in North direction
c         yv(nb) REAL*8   crustal field in East direction
c         zv(nb) REAL*8   crustal field in East direction
c
cccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
        subroutine XYZlocal_coretd(io,nk,tk,ik,ry,idrv
     >                                        ,is,ie,lw,alph,cntr,ra
     >                                           ,cuta,ppos,xv,yv,zv)
c
        implicit none
c
        integer is,ie,lw,nk,ik,io,nb,idrv
        real*8 ppos(*),xv(*),yv(*),zv(*)
        real*8 ry,tk(*),alph(*),cntr(3,*),ra,cuta
c
        integer i,iw,ic,it,itm
        real*8 dt
        real*8, allocatable :: dw(:),xx(:),yy(:),zz(:),tw(:,:)
c
        itm=nk-io
        nb=ie*itm
        iw=int(float(io+1)*float(io+2)/2.)
        allocate(dw(1:iw),xx(1:ie),yy(1:ie),zz(1:ie),tw(1:io,1:idrv+1))
c
        dt=(ppos(4)-ry)/365.25d0
        xv(1:nb)=0.d0
        yv(1:nb)=0.d0
        zv(1:nb)=0.d0
c
c  Define the vectors without time dependence
        call XYZlocal_core(is,ie,lw,alph,cntr,ra,cuta,ppos,xx,yy,zz)
c
c  Define splines
```

```
        i=idrv+1
        call DBSPVD(tk,io,i,dt,ik,io,tw,dw)
c
c  define XV,YV,ZV
        do ic=is,ie
          do i=1,io
            it=ik-io+i
            xv((ic-1)*itm+it)=xx(ic)*tw(i,idrv+1)
            yv((ic-1)*itm+it)=yy(ic)*tw(i,idrv+1)
            zv((ic-1)*itm+it)=zz(ic)*tw(i,idrv+1)
          enddo
        enddo
c
        deallocate(dw,xx,yy,zz)
        return
        end
cccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
c             subroutine XYZlocal_crust.f   V. Lesur 05/02/2007
c
c       derived from localc_XYZ_g.f     V. Lesur  19 June 2005
c
c       For the current harmonic functions i=is,ie defined at
c       radius ra=6371.3km (i.e. Earth's average surface) by:
c       F_i(clat,lon) = SUM(l=0:Lm) alpha(l)
c                          {SUM(m=0:l) Y*lm(clat_i,lon_i) Ylm(clat,lon)}
c       computes the associated magnetic field along X (X horizontal
c       North), Y (Y horizontal East) and Z (Z vertical down) at
c       (clat,lon,radius).
c
c      if the angle between (clat_i,lon_i) and (clat,lon) is larger than
c      cutag the current are assumed to be 0
c
c      The magnetic field along XYZ are saved in the arrays Xv,Yv,Zv
c      with dim. min :  nb=ie
c
c      CALL to:
c       mk_lf_dlf_0 that computes a Legendre polynomial and its
c       derivative
c
```

```
c       input:
c         is/ie   INTEGER      starting/ending center for F_i(clat,lon)
c         lw      INTEGER      Maximum SH degree in Wavelet
c         ppos(3) REAL*8       array of positions in space
c                              ppos(1)=colat (deg)
c                              ppos(2)=long  (deg)
c                              ppos(3)=radius (km)
c         alph(*)  REAL*8      array of alpha values
c         cntr(2,*)REAL*8      array of functions centers
c         cuta     REAL*8      cut angle
c
c       output:
c         xv(nb) REAL*8   crustal field in North direction
c         yv(nb) REAL*8   crustal field in East direction
c         zv(nb) REAL*8   crustal field in East direction
c
cccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
        subroutine XYZlocal_crust(is,ie,lw,alph,cntr,cuta,ppos,xv,yv,zv)
c
        implicit none
c
        integer is,ie,lw
        real*8 ppos(*),xv(*),yv(*),zv(*),alph(*),cntr(2,*),cuta
c
        integer i,j,ic
        real*8 ra,rr,dth,dph,dx,dy,dz,dw
        real*8 dxn,dyn,dzn,dxd,dyd,dzd
        real*8 dxe,dye,dze,dgth,dgr
        real*8 dst,dct,dsp,dcp,dsti,dcti,dspi,dcpi
        real*8 dcosd,dsind,ccuta
        real*8, allocatable :: ddlf(:),dlf(:)
c
        allocate(ddlf(1:lw+1),dlf(1:lw+1))
c
c  Set initial valuyes
        ra=6371.2d0
        ccuta=dcosd(cuta)
c
        dth=ppos(1)
```

```
      dph=ppos(2)
      dst=dsind(dth)
      dct=dcosd(dth)
      dsp=dsind(dph)
      dcp=dcosd(dph)
c
c  Position of data point in cartesian soc
      dxd=dst*dcp
      dyd=dst*dsp
      dzd=dct
c
c  Direction of North vector in cartesian soc
      dxn=-dct*dcp
      dyn=-dct*dsp
      dzn=dst
c
c  Direction of East vector in cartesian soc
      dxe=-dsp
      dye=dcp
      dze=0.0d0
c
c   For each F centre
      do ic=is,ie
c
        dth=cntr(1,ic)
        dph=cntr(2,ic)
        dsti=dsind(dth)
        dcti=dcosd(dth)
        dspi=dsind(dph)
        dcpi=dcosd(dph)
c
c   Position of data point in soc where Z pointing to (dth,dph)
        dw=dxd*dcpi+dyd*dspi
        dy=-dxd*dspi+dyd*dcpi
        dx=dcti*dw-dzd*dsti
        dz=dsti*dw+dzd*dcti
c
c   COS and SIN of data point in soc where Z pointing to (dth,dph)
        dct=dz
```

```
        dst=dsqrt(1.d0-dct**2)
        dcp=0.0d0
        dsp=0.0d0
        if(dst.ne.0.0d0)then
          dcp=dx/dst
          dsp=dy/dst
        endif
c
c   Theta and Radial Gradient of F at data point
        if(dct.ge.ccuta)then
          call mk_lf_dlf_0(lw,dst,dct,dlf,ddlf)
          dgth=0.0d0
          dgr=0.0d0
          rr=ra/ppos(3)
          do j=2,lw+1                              !j=l+1
            dw=alph(j)*rr**(j+1)
            dgth=dgth+ddlf(j)*dw
            dgr=dgr+dble(j)*dlf(j)*dw
          enddo
c
c   North vector in soc where Z pointing to (dth,dph)
          dw=dxn*dcpi+dyn*dspi
          dy=-dxn*dspi+dyn*dcpi
          dx=dcti*dw-dzn*dsti
          dz=dsti*dw+dzn*dcti
c
c   Calculate the gradient in (dx,dy,dz) (i.e. North) direction
          xv(ic)=dct*dcp*dx+dct*dsp*dy-dst*dz
          xv(ic)=-dgth*xv(ic)
c
c   East vector in soc where Z pointing to (dth,dph)
          dw=dxe*dcpi+dye*dspi
          dy=-dxe*dspi+dye*dcpi
          dx=dcti*dw-dze*dsti
          dz=dsti*dw+dze*dcti
c
c   Calculate the gradient in (dx,dy,dz) (i.e. East) direction
          yv(ic)=dct*dcp*dx+dct*dsp*dy-dst*dz
          yv(ic)=-dgth*yv(ic)
```

```
c
c   Calculate the gradient in radial direction
            zv(ic)=-dgr
          else
c
c    gardients assumed to be zero
            xv(ic)=0.0d0
            yv(ic)=0.0d0
            zv(ic)=0.0d0
          endif
        enddo
c
        deallocate(ddlf,dlf)
c
        return
        end
cccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
c             subroutine XYZlocal_X.f   V. Lesur 21/01/2009
c
c       This is a very rough and slow subroutine to test the
c       local modelling approach. It can replace directly the
c       XYZlocal_core subroutine (i.e. it has the same input and
c       outputs)
c
c       USE the Karnel in North direction (X)
c
c       input:
c         is/ie   INTEGER      starting/ending center for F_i(clat,lon)
c         lw      INTEGER      Maximum SH degree in Wavelet
c         ppos(3) REAL*8       array of positions in space
c                              ppos(1)=colat (deg)
c                              ppos(2)=long  (deg)
c                              ppos(3)=radius (km)
c         alph(*)  REAL*8      array of alpha values
c         cntr(3,*)REAL*8      array of functions centers (colat,long,radius)
c         ra       REAL*8      not used
c         cuta     REAL*8      not used
c
c       output:
```

```
c         xx(nb) REAL*8   field in North direction
c         xy(nb) REAL*8   field in East direction
c         xz(nb) REAL*8   field in East direction
c
ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
        subroutine XYZlocal_X(is,ie,lw,alph,cntr,ra,cuta,ppos
     >                                                     ,xx,xy,xz)
c
        implicit none
c
        integer is,ie,lw
        real*8 ppos(*),xx(*),xy(*),xz(*),alph(*),cntr(3,*),ra,cuta
c
        integer i,j,k,ic
        real*8, allocatable :: bx(:),cx(:),cy(:),cz(:)
c
        i=lw*(lw+2)
        allocate(bx(1:i))
        allocate(cx(1:i),cy(1:i),cz(1:i))
c
c  define vector for data point
        call XYZsph_bi0(lw,ppos,cx,cy,cz)
c
c   For each F centre
        do ic=is,ie
c
        call Xsph_bi0(lw,cntr(1,ic),bx)
c
c   Calculate the gradient in North direction
          xx(ic)=0.0d0
          do i=1,lw
            k=i*i
            xx(ic)=xx(ic)+alph(i+1)*bx(k)*cx(k)
            do j=1,i
              k=i*i+2*(j-1)+1
              xx(ic)=xx(ic)+alph(i+1)*bx(k)*cx(k)
              xx(ic)=xx(ic)+alph(i+1)*bx(k+1)*cx(k+1)
            enddo
          enddo
```

```
c
c   Calculate the gradient in East direction
          xy(ic)=0.0d0
          do i=1,lw
            k=i*i
            xy(ic)=xy(ic)+alph(i+1)*bx(k)*cy(k)
            do j=1,i
              k=i*i+2*(j-1)+1
              xy(ic)=xy(ic)+alph(i+1)*bx(k)*cy(k)
              xy(ic)=xy(ic)+alph(i+1)*bx(k+1)*cy(k+1)
            enddo
          enddo
c
c   Calculate the gradient in radial direction
          xz(ic)=0.0d0
          do i=1,lw
            k=i*i
            xz(ic)=xz(ic)+alph(i+1)*bx(k)*cz(k)
            do j=1,i
              k=i*i+2*(j-1)+1
              xz(ic)=xz(ic)+alph(i+1)*bx(k)*cz(k)
              xz(ic)=xz(ic)+alph(i+1)*bx(k+1)*cz(k+1)
            enddo
          enddo
        enddo
c
        deallocate(cx,cy,cz)
        deallocate(bx)
c
        return
        end
ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
c             subroutine XYZlocal_coret.f   V. Lesur 05/05/2008
c
c       For the harmonic functions i=is,ie defined at radius ra=3485.km
c       (i.e. Earth's core radius) by:
c       F_i(clat,lon) = ra SUM(l=0:Lm) alpha(l) (ra/r)**(l+1)
c         (ra/r_i)**(l+2) {SUM(m=0:l) Y*lm(clat_i,lon_i) Ylm(clat,lon)}
c       computes the associated magnetic field along X (X horizontal
```

```
c        North), Y (Y horizontal East) and Z (Z vertical down) at
c        (clat,lon,radius).
c
c       A time dependence is introduced by B-splines of oder io
c
c       if the angle between (clat_i,lon_i) and (clat,lon) is larger than
c       cutag the current are assumed to be 0
c
c       The magnetic field along XYZ are saved in the arrays Xv,Yv,Zv
c       with dim. min :  nb=ie*(nk-io) where "nk" is the number of
c       spline nodes.
c
c       CALL to:
c        XYZlocal_core.f that computes the same thing without time
c        dependence
c
c        input:
c          io      INTEGER      spline order
c          nk      INTEGER      number of spline knots
c          tk      REAL*8       array of spline knot positions
c          ik      INTEGER      spline knot tk(ik)=< dt <tk(ik+1)
c          ry      REAL*8       reference year
c          is/ie   INTEGER      starting/ending center for F_i(clat,lon)
c          lw      INTEGER      Maximum SH degree in Wavelet
c          ppos(4) REAL*8       array of positions in space
c                               ppos(1)=colat (deg)
c                               ppos(2)=long  (deg)
c                               ppos(3)=radius (km)
c                               ppos(4)=time (MJD)
c          alph(*)  REAL*8      array of alpha values
c          cntr(3,*)REAL*8      array of functions centers (colat,long,radius)
c          ra       REAL*8      function reference radius
c          cuta     REAL*8      cut angle
c
c        output:
c          xv(nb) REAL*8   crustal field in North direction
c          yv(nb) REAL*8   crustal field in East direction
c          zv(nb) REAL*8   crustal field in East direction
c
```

```
cccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
      subroutine XYZlocal_Xt(io,nk,tk,ik,ry,is,ie,lw,alph,cntr,ra
     >                                            ,cuta,ppos,xv,yv,zv)
c
      implicit none
c
      integer is,ie,lw,nk,ik,io,nb
      real*8 ppos(*),xv(*),yv(*),zv(*)
      real*8 ry,tk(*),alph(*),cntr(3,*),ra,cuta
c
      integer i,iw,ic,it,itm
      real*8 tw(io),dt
      real*8, allocatable :: dw(:),xx(:),yy(:),zz(:)
c
      itm=nk-io
      nb=ie*itm
      iw=int(float(io+1)*float(io+2)/2.)
      allocate(dw(1:iw),xx(1:ie),yy(1:ie),zz(1:ie))
c
      dt=(ppos(4)-ry)/365.25d0
      xv(1:nb)=0.d0
      yv(1:nb)=0.d0
      zv(1:nb)=0.d0
c
c  Define the vectors without time dependence
      call XYZlocal_X(is,ie,lw,alph,cntr,ra,cuta,ppos,xx,yy,zz)
c
c  Define splines
      call DBSPVD(tk,io,1,dt,ik,io,tw,dw)
c
c  define XV,YV,ZV
      do ic=is,ie
        do i=1,io
          it=ik-io+i
          xv((ic-1)*itm+it)=xx(ic)*tw(i)
          yv((ic-1)*itm+it)=yy(ic)*tw(i)
          zv((ic-1)*itm+it)=zz(ic)*tw(i)
        enddo
      enddo
```

```
c
        deallocate(dw,xx,yy,zz)
        return
        end
cccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
c             subroutine XYZlocal_XYZtd.f   V. Lesur 05/11/2008
c
c       For the harmonic functions i=is,ie defined at radius ra=6371.2km
c       (i.e. Earth's reference radius) by:
c       dir.F_i(clat,lon) = ra**2 SUM(l=0:Lm) alpha(l) (ra/r)**(l+1)
c            {SUM(m=0:l) dir.GRAD[(ra/r_i)**(l+1) Y*lm(clat_i,lon_i)]
c                                                        Ylm(clat,lon)}
c       computes the associated magnetic field along X (X horizontal
c       North), Y (Y horizontal East) and Z (Z vertical down) at
c       (clat,lon,radius) and its time derivatives. the direction "dir"
c       is either X, Y or Z depending on the input.
c
c      A time dependence is introduced by B-splines of order io
c
c      if the angle between (clat_i,lon_i) and (clat,lon) is larger than
c      cutag the current are assumed to be 0
c
c      The magnetic field along XYZ are saved in the arrays Xv,Yv,Zv
c      with dim. min :  nb=ie*(nk-io) where "nk" is the number of
c      spline nodes.
c
c      CALL to:
c       XYZlocal_X.f that computes the same thing without time
c       dependence for dir='X'
c       XYZlocal_Y.f that computes the same thing without time
c       dependence for dir='Y'
c       XYZlocal_Z.f that computes the same thing without time
c       dependence for dir='Z'
c
c       input:
c         dir     CHARACTER    'X', 'Y' or 'Z' depending on direction
c         io      INTEGER      spline order
c         nk      INTEGER      number of spline knots
c         tk      REAL*8       array of spline knot positions
```

```
c          ik      INTEGER      spline knot tk(ik)=< dt <tk(ik+1)
c          ry      REAL*8       reference year
c          is/ie   INTEGER      starting/ending center for F_i(clat,lon)
c          lw      INTEGER      Maximum SH degree in Wavelet
c          ppos(4) REAL*8       array of positions in space
c                               ppos(1)=colat (deg)
c                               ppos(2)=long  (deg)
c                               ppos(3)=radius (km)
c                               ppos(4)=time (MJD)
c          alph(*)  REAL*8      array of alpha values
c          cntr(3,*)REAL*8      array of functions centers (colat,long,radius)
c          ra       REAL*8      function reference radius
c          cuta     REAL*8      cut angle
c          idrv     INTEGER     derivative number
c
c        output:
c          xv(nb) REAL*8   crustal field in North direction
c          yv(nb) REAL*8   crustal field in East direction
c          zv(nb) REAL*8   crustal field in East direction
c
cccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
        subroutine XYZlocal_XYZtd(dir,io,nk,tk,ik,ry,idrv
     >                                         ,is,ie,lw,alph,cntr,ra
     >                                            ,cuta,ppos,xv,yv,zv)
c
        implicit none
c
        integer is,ie,lw,nk,ik,io,nb,idrv
        real*8 ppos(*),xv(*),yv(*),zv(*)
        real*8 ry,tk(*),alph(*),cntr(3,*),ra,cuta
        character dir
c
        integer i,iw,ic,it,itm
        real*8 dt
        real*8, allocatable :: dw(:),xx(:),yy(:),zz(:),tw(:,:)
c
        itm=nk-io
        nb=ie*itm
        iw=int(float(io+1)*float(io+2)/2.)
```

```
      allocate(dw(1:iw),xx(1:ie),yy(1:ie),zz(1:ie),tw(1:io,1:idrv+1))
c
      dt=(ppos(4)-ry)/365.25d0
      xv(1:nb)=0.d0
      yv(1:nb)=0.d0
      zv(1:nb)=0.d0
c
c  Define the vectors without time dependence
      if(dir.eq.'X')then
        call XYZlocal_X(is,ie,lw,alph,cntr,ra,cuta,ppos,xx,yy,zz)
      elseif(dir.eq.'Y')then
        call XYZlocal_Y(is,ie,lw,alph,cntr,ra,cuta,ppos,xx,yy,zz)
      else
        call XYZlocal_Z(is,ie,lw,alph,cntr,ra,cuta,ppos,xx,yy,zz)
      endif
c
c  Define splines
      i=idrv+1
      call DBSPVD(tk,io,i,dt,ik,io,tw,dw)
c
c  define XV,YV,ZV
      do ic=is,ie
        do i=1,io
          it=ik-io+i
          xv((ic-1)*itm+it)=xx(ic)*tw(i,idrv+1)
          yv((ic-1)*itm+it)=yy(ic)*tw(i,idrv+1)
          zv((ic-1)*itm+it)=zz(ic)*tw(i,idrv+1)
        enddo
      enddo
c
      deallocate(dw,xx,yy,zz)
      return
      end
cccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
c             subroutine XYZlocal_Y.f   V. Lesur 21/01/2009
c
c       This is a very rough and slow subroutine to test the
c       local modelling approach. It can replace directly the
c       XYZlocal_core subroutine (i.e. it has the same input and
```

```
c       outputs)
c
c       USE the Karnel in East direction (Y)
c
c       input:
c         is/ie   INTEGER      starting/ending center for F_i(clat,lon)
c         lw      INTEGER      Maximum SH degree in Wavelet
c         ppos(3) REAL*8       array of positions in space
c                              ppos(1)=colat (deg)
c                              ppos(2)=long  (deg)
c                              ppos(3)=radius (km)
c         alph(*)  REAL*8      array of alpha values
c         cntr(3,*)REAL*8      array of functions centers (colat,long,radius)
c         ra       REAL*8      not used
c         cuta     REAL*8      not used
c
c       output:
c         yx(nb) REAL*8   field in North direction
c         yy(nb) REAL*8   field in East direction
c         yz(nb) REAL*8   field in East direction
c
cccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
        subroutine XYZlocal_Y(is,ie,lw,alph,cntr,ra,cuta,ppos
     >                                                        ,yx,yy,yz)
c
        implicit none
c
        integer is,ie,lw
        real*8 ppos(*),yx(*),yy(*),yz(*),alph(*),cntr(3,*),ra,cuta
c
        integer i,j,k,ic
        real*8, allocatable :: by(:),cx(:),cy(:),cz(:)
c
        i=lw*(lw+2)
        allocate(by(1:i))
        allocate(cx(1:i),cy(1:i),cz(1:i))
c
c  define vector for data point
        call XYZsph_bi0(lw,ppos,cx,cy,cz)
```

```
c
c   For each F centre
        do ic=is,ie
c
          call Ysph_bi0(lw,cntr(1,ic),by)
c
c   Calculate the gradient in North direction
          yx(ic)=0.0d0
          do i=1,lw
            k=i*i
            yx(ic)=yx(ic)+alph(i+1)*by(k)*cx(k)
            do j=1,i
              k=i*i+2*(j-1)+1
              yx(ic)=yx(ic)+alph(i+1)*by(k)*cx(k)
              yx(ic)=yx(ic)+alph(i+1)*by(k+1)*cx(k+1)
            enddo
          enddo
c
c   Calculate the gradient in East direction
          yy(ic)=0.0d0
          do i=1,lw
            k=i*i
            yy(ic)=yy(ic)+alph(i+1)*by(k)*cy(k)
            do j=1,i
              k=i*i+2*(j-1)+1
              yy(ic)=yy(ic)+alph(i+1)*by(k)*cy(k)
              yy(ic)=yy(ic)+alph(i+1)*by(k+1)*cy(k+1)
            enddo
          enddo
c
c   Calculate the gradient in radial direction
          yz(ic)=0.0d0
          do i=1,lw
            k=i*i
            yz(ic)=yz(ic)+alph(i+1)*by(k)*cz(k)
            do j=1,i
              k=i*i+2*(j-1)+1
              yz(ic)=yz(ic)+alph(i+1)*by(k)*cz(k)
              yz(ic)=yz(ic)+alph(i+1)*by(k+1)*cz(k+1)
```

```
          enddo
        enddo
      enddo
c
      deallocate(cx,cy,cz)
      deallocate(by)
c
      return
      end
cccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
c            subroutine XYZlocal_coret.f   V. Lesur 05/05/2008
c
c       For the harmonic functions i=is,ie defined at radius ra=3485.km
c       (i.e. Earth's core radius) by:
c       F_i(clat,lon) = ra SUM(l=0:Lm) alpha(l) (ra/r)**(l+1)
c         (ra/r_i)**(l+2) {SUM(m=0:l) Y*lm(clat_i,lon_i) Ylm(clat,lon)}
c       computes the associated magnetic field along X (X horizontal
c       North), Y (Y horizontal East) and Z (Z vertical down) at
c       (clat,lon,radius).
c
c      A time dependence is introduced by B-splines of oder io
c
c      if the angle between (clat_i,lon_i) and (clat,lon) is larger than
c      cutag the current are assumed to be 0
c
c      The magnetic field along XYZ are saved in the arrays Xv,Yv,Zv
c      with dim. min :  nb=ie*(nk-io) where "nk" is the number of
c      spline nodes.
c
c      CALL to:
c       XYZlocal_core.f that computes the same thing without time
c       dependence
c
c       input:
c         io      INTEGER      spline order
c         nk      INTEGER      number of spline knots
c         tk      REAL*8       array of spline knot positions
c         ik      INTEGER      spline knot tk(ik)=< dt <tk(ik+1)
c         ry      REAL*8       reference year
```

```
c         is/ie   INTEGER      starting/ending center for F_i(clat,lon)
c         lw      INTEGER      Maximum SH degree in Wavelet
c         ppos(4) REAL*8       array of positions in space
c                              ppos(1)=colat (deg)
c                              ppos(2)=long  (deg)
c                              ppos(3)=radius (km)
c                              ppos(4)=time (MJD)
c         alph(*)  REAL*8      array of alpha values
c         cntr(3,*)REAL*8      array of functions centers (colat,long,radius)
c         ra       REAL*8      function reference radius
c         cuta     REAL*8      cut angle
c
c       output:
c         xv(nb) REAL*8   crustal field in North direction
c         yv(nb) REAL*8   crustal field in East direction
c         zv(nb) REAL*8   crustal field in East direction
c
cccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
        subroutine XYZlocal_Yt(io,nk,tk,ik,ry,is,ie,lw,alph,cntr,ra
     >                                            ,cuta,ppos,xv,yv,zv)
c
        implicit none
c
        integer is,ie,lw,nk,ik,io,nb
        real*8 ppos(*),xv(*),yv(*),zv(*)
        real*8 ry,tk(*),alph(*),cntr(3,*),ra,cuta
c
        integer i,iw,ic,it,itm
        real*8 tw(io),dt
        real*8, allocatable :: dw(:),xx(:),yy(:),zz(:)
c
        itm=nk-io
        nb=ie*itm
        iw=int(float(io+1)*float(io+2)/2.)
        allocate(dw(1:iw),xx(1:ie),yy(1:ie),zz(1:ie))
c
        dt=(ppos(4)-ry)/365.25d0
        xv(1:nb)=0.d0
        yv(1:nb)=0.d0
```

```
      zv(1:nb)=0.d0
c
c  Define the vectors without time dependence
      call XYZlocal_Y(is,ie,lw,alph,cntr,ra,cuta,ppos,xx,yy,zz)
c
c  Define splines
      call DBSPVD(tk,io,1,dt,ik,io,tw,dw)
c
c  define XV,YV,ZV
      do ic=is,ie
        do i=1,io
          it=ik-io+i
          xv((ic-1)*itm+it)=xx(ic)*tw(i)
          yv((ic-1)*itm+it)=yy(ic)*tw(i)
          zv((ic-1)*itm+it)=zz(ic)*tw(i)
        enddo
      enddo
c
      deallocate(dw,xx,yy,zz)
      return
      end
cccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
c             subroutine XYZlocal_Z.f   V. Lesur 21/01/2009
c
c       This is a very rough and slow subroutine to test the
c       local modelling approach. It can replace directly the
c       XYZlocal_core subroutine (i.e. it has the same input and
c       outputs)
c
c       USE the Karnel in DOWN direction (Z)
c
c       input:
c         is/ie   INTEGER      starting/ending center for F_i(clat,lon)
c         lw      INTEGER      Maximum SH degree in Wavelet
c         ppos(3) REAL*8       array of positions in space
c                              ppos(1)=colat (deg)
c                              ppos(2)=long  (deg)
c                              ppos(3)=radius (km)
c         alph(*)  REAL*8      array of alpha values
```

```
c         cntr(3,*)REAL*8      array of functions centers (colat,long,radius)
c         ra       REAL*8      not used
c         cuta     REAL*8      not used
c
c       output:
c         zx(nb) REAL*8   field in North direction
c         zy(nb) REAL*8   field in East direction
c         zz(nb) REAL*8   field in East direction
c
cccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
        subroutine XYZlocal_Z(is,ie,lw,alph,cntr,ra,cuta,ppos
     >                                                        ,zx,zy,zz)
c
        implicit none
c
        integer is,ie,lw
        real*8 ppos(*),zx(*),zy(*),zz(*),alph(*),cntr(3,*),ra,cuta
c
        integer i,j,k,ic
        real*8, allocatable :: bz(:),cx(:),cy(:),cz(:)
c
        i=lw*(lw+2)
        allocate(bz(1:i))
        allocate(cx(1:i),cy(1:i),cz(1:i))
c
c  define vector for data point
        call XYZsph_bi0(lw,ppos,cx,cy,cz)
c
c   For each F centre
        do ic=is,ie
c
        call Zsph_bi0(lw,cntr(1,ic),bz)
c
c   Calculate the gradient in North direction
          zx(ic)=0.0d0
          do i=1,lw
            k=i*i
            zx(ic)=zx(ic)+alph(i+1)*bz(k)*cx(k)
            do j=1,i
```

```
            k=i*i+2*(j-1)+1
            zx(ic)=zx(ic)+alph(i+1)*bz(k)*cx(k)
            zx(ic)=zx(ic)+alph(i+1)*bz(k+1)*cx(k+1)
          enddo
        enddo
c
c   Calculate the gradient in East direction
        zy(ic)=0.0d0
        do i=1,lw
          k=i*i
          zy(ic)=zy(ic)+alph(i+1)*bz(k)*cy(k)
          do j=1,i
            k=i*i+2*(j-1)+1
            zy(ic)=zy(ic)+alph(i+1)*bz(k)*cy(k)
            zy(ic)=zy(ic)+alph(i+1)*bz(k+1)*cy(k+1)
          enddo
        enddo
c
c   Calculate the gradient in radial direction
        zz(ic)=0.0d0
        do i=1,lw
          k=i*i
          zz(ic)=zz(ic)+alph(i+1)*bz(k)*cz(k)
          do j=1,i
            k=i*i+2*(j-1)+1
            zz(ic)=zz(ic)+alph(i+1)*bz(k)*cz(k)
            zz(ic)=zz(ic)+alph(i+1)*bz(k+1)*cz(k+1)
          enddo
        enddo
      enddo
c
      deallocate(cx,cy,cz)
      deallocate(bz)
c
      return
      end
cccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
c             subroutine XYZlocal_coret.f   V. Lesur 05/05/2008
c
```

```
c       For the harmonic functions i=is,ie defined at radius ra=3485.km
c       (i.e. Earth's core radius) by:
c       F_i(clat,lon) = ra SUM(l=0:Lm) alpha(l) (ra/r)**(l+1)
c         (ra/r_i)**(l+2) {SUM(m=0:l) Y*lm(clat_i,lon_i) Ylm(clat,lon)}
c       computes the associated magnetic field along X (X horizontal
c       North), Y (Y horizontal East) and Z (Z vertical down) at
c       (clat,lon,radius).
c
c      A time dependence is introduced by B-splines of oder io
c
c      if the angle between (clat_i,lon_i) and (clat,lon) is larger than
c      cutag the current are assumed to be 0
c
c      The magnetic field along XYZ are saved in the arrays Xv,Yv,Zv
c      with dim. min :  nb=ie*(nk-io) where "nk" is the number of
c      spline nodes.
c
c      CALL to:
c       XYZlocal_core.f that computes the same thing without time
c       dependence
c
c       input:
c         io      INTEGER      spline order
c         nk      INTEGER      number of spline knots
c         tk      REAL*8       array of spline knot positions
c         ik      INTEGER      spline knot tk(ik)=< dt <tk(ik+1)
c         ry      REAL*8       reference year
c         is/ie   INTEGER      starting/ending center for F_i(clat,lon)
c         lw      INTEGER      Maximum SH degree in Wavelet
c         ppos(4) REAL*8       array of positions in space
c                              ppos(1)=colat (deg)
c                              ppos(2)=long  (deg)
c                              ppos(3)=radius (km)
c                              ppos(4)=time (MJD)
c         alph(*)  REAL*8      array of alpha values
c         cntr(3,*)REAL*8      array of functions centers (colat,long,radius)
c         ra       REAL*8      function reference radius
c         cuta     REAL*8      cut angle
c
```

```
c       output:
c         xv(nb) REAL*8   crustal field in North direction
c         yv(nb) REAL*8   crustal field in East direction
c         zv(nb) REAL*8   crustal field in East direction
c
cccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
        subroutine XYZlocal_Zt(io,nk,tk,ik,ry,is,ie,lw,alph,cntr,ra
     >                                            ,cuta,ppos,xv,yv,zv)
c
        implicit none
c
        integer is,ie,lw,nk,ik,io,nb
        real*8 ppos(*),xv(*),yv(*),zv(*)
        real*8 ry,tk(*),alph(*),cntr(3,*),ra,cuta
c
        integer i,iw,ic,it,itm
        real*8 tw(io),dt
        real*8, allocatable :: dw(:),xx(:),yy(:),zz(:)
c
        itm=nk-io
        nb=ie*itm
        iw=int(float(io+1)*float(io+2)/2.)
        allocate(dw(1:iw),xx(1:ie),yy(1:ie),zz(1:ie))
c
        dt=(ppos(4)-ry)/365.25d0
        xv(1:nb)=0.d0
        yv(1:nb)=0.d0
        zv(1:nb)=0.d0
c
c  Define the vectors without time dependence
        call XYZlocal_Z(is,ie,lw,alph,cntr,ra,cuta,ppos,xx,yy,zz)
c
c  Define splines
        call DBSPVD(tk,io,1,dt,ik,io,tw,dw)
c
c  define XV,YV,ZV
        do ic=is,ie
          do i=1,io
            it=ik-io+i
```

```
            xv((ic-1)*itm+it)=xx(ic)*tw(i)
            yv((ic-1)*itm+it)=yy(ic)*tw(i)
            zv((ic-1)*itm+it)=zz(ic)*tw(i)
          enddo
        enddo
c
        deallocate(dw,xx,yy,zz)
        return
        end
cccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
c             subroutine XYZradial_dipoles.f   V. Lesur 08/08/2008
c
c       derived from XYZlocal_crust.f
c
c       For the harmonic functions i=is,ie defined at
c       radius ra=6371.3km (i.e. Earth's average surface) by a
c       radial dipole:
c       F_i(clat,lon) = SUM(l=0:Lm)
c                          {SUM(m=0:l) Y*lm(clat_i,lon_i) Ylm(clat,lon)}
c       computes the associated magnetic field along X (X horizontal
c       North), Y (Y horizontal East) and Z (Z vertical down) at
c       (clat,lon,radius).
c
c      if the angle between (clat_i,lon_i) and (clat,lon) is larger than
c      cutag the current are assumed to be 0
c
c      The magnetic field along XYZ are saved in the arrays Xv,Yv,Zv
c      with dim. min :  nb=ie
c
c       input:
c         is/ie   INTEGER      starting/ending center for F_i(clat,lon)
c         ppos(3) REAL*8       array of positions in space
c                              ppos(1)=colat (deg)
c                              ppos(2)=long  (deg)
c                              ppos(3)=radius (km)
c         cntr(2,*)REAL*8      array of dipole positions
c         cuta     REAL*8      cut angle
c
c       output:
```

```
c         xv(nb) REAL*8   crustal field in North direction
c         yv(nb) REAL*8   crustal field in East direction
c         zv(nb) REAL*8   crustal field in East direction
c
cccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
        subroutine XYZradial_dipoles(is,ie,cntr,cuta,ppos,xv,yv,zv)
c
        implicit none
c
        integer is,ie
        real*8 ppos(*),xv(*),yv(*),zv(*),cntr(2,*),cuta
c
        integer i,j,ic
        real*8 ra,rr,dth,dph,dx,dy,dz,dw
        real*8 dxn,dyn,dzn,dxd,dyd,dzd
        real*8 dxe,dye,dze,dgth,dgr
        real*8 dst,dct,dsp,dcp,dsti,dcti,dspi,dcpi
        real*8 dcosd,dsind,ccuta
c
c  Set initial valuyes
        ra=6371.2d0
        ccuta=dcosd(cuta)
c
        dth=ppos(1)
        dph=ppos(2)
        dst=dsind(dth)
        dct=dcosd(dth)
        dsp=dsind(dph)
        dcp=dcosd(dph)
c
c  Position of data point in cartesian soc
        dxd=dst*dcp
        dyd=dst*dsp
        dzd=dct
c
c  Direction of North vector in cartesian soc
        dxn=-dct*dcp
        dyn=-dct*dsp
        dzn=dst
```

```
c
c  Direction of East vector in cartesian soc
     dxe=-dsp
     dye=dcp
     dze=0.0d0
c
c   For each F centre
     do ic=is,ie
c
       dth=cntr(1,ic)
       dph=cntr(2,ic)
       dsti=dsind(dth)
       dcti=dcosd(dth)
       dspi=dsind(dph)
       dcpi=dcosd(dph)
c
c   Position of data point in soc where Z pointing to (dth,dph)
       dw=dxd*dcpi+dyd*dspi
       dy=-dxd*dspi+dyd*dcpi
       dx=dcti*dw-dzd*dsti
       dz=dsti*dw+dzd*dcti
c
c   COS and SIN of data point in soc where Z pointing to (dth,dph)
       dct=dz
       dw=dct**2
       if(dw.lt.0.0d0)dw=0.0d0
       dst=dsqrt(1.d0-dw)
       dcp=0.0d0
       dsp=0.0d0
       if(dst.ne.0.0d0)then
         dcp=dx/dst
         dsp=dy/dst
       endif
c
c   Theta and Radial Gradient of radial dipole at data point
       if(dct.ge.ccuta)then
         rr=ra/ppos(3)
         dw=dsqrt(1.d0-2.d0*rr*dct+rr**2)
c
```

```
           dgr=dct/(dw**3)
           dgr=dgr+3.d0*(rr-dct)*(1.d0-rr*dct)/(dw**5)
           dgr=dgr/(rr**3)
c
           dgth=-dst/(dw**3)
           dgth=dgth+3.d0*(rr-dct)*(rr*dst)/(dw**5)
           dgth=dgth/(rr**3)
c
c   North vector in soc where Z pointing to (dth,dph)
           dw=dxn*dcpi+dyn*dspi
           dy=-dxn*dspi+dyn*dcpi
           dx=dcti*dw-dzn*dsti
           dz=dsti*dw+dzn*dcti
c
c   Calculate the gradient in (dx,dy,dz) (i.e. North) direction
           xv(ic)=dct*dcp*dx+dct*dsp*dy-dst*dz
           xv(ic)=-dgth*xv(ic)
c
c   East vector in soc where Z pointing to (dth,dph)
           dw=dxe*dcpi+dye*dspi
           dy=-dxe*dspi+dye*dcpi
           dx=dcti*dw-dze*dsti
           dz=dsti*dw+dze*dcti
c
c   Calculate the gradient in (dx,dy,dz) (i.e. East) direction
           yv(ic)=dct*dcp*dx+dct*dsp*dy-dst*dz
           yv(ic)=-dgth*yv(ic)
c
c   Calculate the gradient in radial direction
           zv(ic)=-dgr
         else
c
c    gardients assumed to be zero
           xv(ic)=0.0d0
           yv(ic)=0.0d0
           zv(ic)=0.0d0
         endif
       enddo
c
```

```
        return
        end
cccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
c Subroutine XYZsph_bi0
c Vincent Lesur 21/01/2009
c
c       For a all spherical harmonique between degree 1 and nlie
c       and a  colatitude, longitude, radius computes for
c       internal sources:
c       X component basis  (X horizontal North)
c       Y component basis  (Y horizontal East)
c       Z component basis  (Z Vertical Down)
c
c       These basis is the derivative of a potential
c       defined on the sphere of radius "ra"
c       (See Foundations of Geomagnetism, Backus 1996 page 110 and 125)
c
c       No time dependence
c
c       Spherical harmonics are set by increasing degree "l", for
c       a given degree by increasing order "m" (l=0,m=0 excluded)
c       nu=(l**2+2*(m-1))   FOR cosine terms if m.ne.0
c       nu=(l**2+2*m-1)     For sine terms or m=0
c
c       colat,long,radius given in degree,degree,km
c
c       input:
c         nlie          end of degree internal
c         pos           positions in space and time (colat,long,radius)
c
c       output:
c         bx X Base funtion value
c         by Y Base funtion value
c         bz Z Base funtion value
c
cccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
subroutine XYZsph_bi0(nlie,pos,bx,by,bz)
c
        implicit none
```

```
c
      integer nu,il,im,nlie
      real*8 pos(*),bx(*),by(*),bz(*),rc,rs,ra,dr,dw
      real*8 ds,dc,dt,dcosd,dsind
      real*8, allocatable :: dlf(:),ddlf(:)
c
      external dcosd,dsind
      allocate(dlf(1:nlie+1),ddlf(1:nlie+1))
c
      ra=6371.2d0
      rc=dcosd(pos(1))
      rs=dsind(pos(1))
c
c   im=0
      im=0
      call mk_lf_dlf(im,nlie,rs,rc,dlf,ddlf)
      do il=1,nlie
        nu=(il*il-1)
        dr=(ra/pos(3))**(il+2)
        bx(nu+1)=ddlf(il-im+1)*dr
        by(nu+1)=0.0d0
        bz(nu+1)=-dlf(il-im+1)*dble(il+1)*dr
      enddo
c   im.ne.0
      do im=1,nlie
        call mk_lf_dlf(im,nlie,rs,rc,dlf,ddlf)
        dc=dcosd(im*pos(2))
        ds=dsind(im*pos(2))
        do il=im,nlie
          nu=il*il+2*(im-1)
          dr=(ra/pos(3))**(il+2)
          dw=ddlf(il-im+1)*dr
          bx(nu+1)=dw*dc
          bx(nu+2)=dw*ds
          dw=dlf(il-im+1)/rs
          dw=dw*dble(im)*dr
          by(nu+1)=dw*ds
          by(nu+2)=-dw*dc
          dw=dlf(il-im+1)*dble(il+1)*dr
```

```
            bz(nu+1)=-dw*dc
            bz(nu+2)=-dw*ds
          enddo
        enddo
        deallocate(dlf,ddlf)
c
        return
        end
ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
c Subroutine Ysph_bi0
c Vincent Lesur 30/04/2002
c
c       For a all spherical harmonique between degree 1 and nlie
c       and a  colatitude, longitude, radius computes for
c       internal sources:
c       Y component basis  (Y horizontal East)
c
c       These basis is the derivative of a potential
c       defined on the sphere of radius "ra"
c       (See Foundations of Geomagnetism, Backus 1996 page 110 and 125)
c
c       No time dependence
c
c       Spherical harmonics are set by increasing degree "l", for
c       a given degree by increasing order "m" (l=0,m=0 excluded)
c       nu=(l**2+2*(m-1))   FOR cosine terms if m.ne.0
c       nu=(l**2+2*m-1)     For sine terms or m=0
c
c       colat,long,radius given in degree,degree,km
c
c       input:
c         nlie          end of degree internal
c         pos           positions in space and time (colat,long,radius)
c
c       output:
c         be            Y Base funtion value
c
ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
subroutine Ysph_bi0(nlie,pos,be)
```

```
c
        implicit none
c
        integer nu,il,im,nlie
        real*8 pos(*),be(*),rc,rs,ra,dw
        real*8 ds,dc,dt,dcosd,dsind
        real*8, allocatable :: dlf(:)
c
        external dcosd,dsind
        allocate(dlf(1:nlie+1))
c
        ra=6371.2d0
        rc=dcosd(pos(1))
        rs=dsind(pos(1))
c
c   im=0
        im=0
        do il=1,nlie
          nu=(il*il-1)
          be(nu+1)=0.d0
        enddo
c   im.ne.0
        do im=1,nlie
          call mklf_F(im,nlie,rs,rc,dlf)
          dc=-im*dsind(im*pos(2))
          ds=im*dcosd(im*pos(2))
          do il=im,nlie
            dw=dlf(il-im+1)/rs
            dw=dw*(ra/pos(3))**(il+2)
            nu=(il*il+2*(im-1))
            be(nu+1)=-dw*dc
            be(nu+2)=-dw*ds
          enddo
        enddo
c
        deallocate(dlf)
        return
        end
cccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
```

```
c Subroutine Zsph_bi0
c Vincent Lesur 30/04/2002
c
c       For a all spherical harmonique between degree 1 and nlie
c       and a  colatitude, longitude, radius computes for
c       internal sources:
c       Z component basis  (Z Vertical Down)
c
c       These basis is the derivative of a potential
c       defined on the sphere of radius "ra"
c       (See Foundations of Geomagnetism, Backus 1996 page 110 and 125)
c
c       No time dependence
c
c       Spherical harmonics are set by increasing degree "l", for
c       a given degree by increasing order "m" (l=0,m=0 excluded)
c       nu=(l**2+2*(m-1))   FOR cosine terms if m.ne.0
c       nu=(l**2+2*m-1)     For sine terms or m=0
c
c       colat,long,radius given in degree,degree,km
c
c       input:
c         nlie          end of degree internal
c         pos           positions in space and time (colat,long,radius)
c
c       output:
c         be Base funtion value
c
ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
subroutine Zsph_bi0(nlie,pos,be)
c
        implicit none
c
        integer nu,il,im,nlie
        real*8 pos(*),be(*),rc,rs,ra,dw
        real*8 dc,ds,dt,dcosd,dsind
        real*8, allocatable :: dlf(:)
c
        external dcosd,dsind
```

```
        allocate(dlf(1:nlie+1))
c
        ra=6371.2d0
        rc=dcosd(pos(1))
        rs=dsind(pos(1))

        im=0
        call mklf_F(im,nlie,rs,rc,dlf)
        do il=1,nlie
          nu=(il*il-1)
          be(nu+1)=-dlf(il-im+1)*(il+1)*(ra/pos(3))**(il+2)
        enddo
        do im=1,nlie
          call mklf_F(im,nlie,rs,rc,dlf)
          dc=dcosd(dble(im)*pos(2))
          ds=dsind(dble(im)*pos(2))
          do il=im,nlie
            dw=dlf(il-im+1)*(il+1)*(ra/pos(3))**(il+2)
            nu=(il*il+2*(im-1))
            be(nu+1)=-dw*dc
            be(nu+2)=-dw*ds
          enddo
        enddo
        deallocate(dlf)
        return
        end
SUBROUTINE DBSPVD (T, K, NDERIV, X, ILEFT, LDVNIK, VNIKX, WORK)
C***BEGIN PROLOGUE  DBSPVD
C***PURPOSE  Calculate the value and all derivatives of order less than
C            NDERIV of all basis functions which do not vanish at X.
C***LIBRARY   SLATEC
C***CATEGORY  E3, K6
C***TYPE      DOUBLE PRECISION (BSPVD-S, DBSPVD-D)
C***KEYWORDS  DIFFERENTIATION OF B-SPLINE, EVALUATION OF B-SPLINE
C***AUTHOR  Amos, D. E., (SNLA)
C***DESCRIPTION
C
C     Written by Carl de Boor and modified by D. E. Amos
C
```

```
C     Abstract    **** a double precision routine ****
C
C         DBSPVD is the BSPLVD routine of the reference.
C
C         DBSPVD calculates the value and all derivatives of order
C         less than NDERIV of all basis functions which do not
C         (possibly) vanish at X.  ILEFT is input such that
C         T(ILEFT) .LE. X .LT. T(ILEFT+1).  A call to INTRV(T,N+1,X,
C         ILO,ILEFT,MFLAG) will produce the proper ILEFT.  The output of
C         DBSPVD is a matrix VNIKX(I,J) of dimension at least (K,NDERIV)
C         whose columns contain the K nonzero basis functions and
C         their NDERIV-1 right derivatives at X, I=1,K, J=1,NDERIV.
C         These basis functions have indices ILEFT-K+I, I=1,K,
C         K .LE. ILEFT .LE. N.  The nonzero part of the I-th basis
C         function lies in (T(I),T(I+K)), I=1,N).
C
C         If X=T(ILEFT+1) then VNIKX contains left limiting values
C         (left derivatives) at T(ILEFT+1).  In particular, ILEFT = N
C         produces left limiting values at the right end point
C         X=T(N+1).  To obtain left limiting values at T(I), I=K+1,N+1,
C         set X= next lower distinct knot, call INTRV to get ILEFT,
C         set X=T(I), and then call DBSPVD.
C
C     Description of Arguments
C         Input      T,X are double precision
C          T       - knot vector of length N+K, where
C                    N = number of B-spline basis functions
C                    N = sum of knot multiplicities-K
C          K       - order of the B-spline, K .GE. 1
C          NDERIV  - number of derivatives = NDERIV-1,
C                    1 .LE. NDERIV .LE. K
C          X       - argument of basis functions,
C                    T(K) .LE. X .LE. T(N+1)
C          ILEFT   - largest integer such that
C                    T(ILEFT) .LE. X .LT.  T(ILEFT+1)
C          LDVNIK  - leading dimension of matrix VNIKX
C
C         Output     VNIKX,WORK are double precision
C          VNIKX   - matrix of dimension at least (K,NDERIV) contain-
```

```
C                   ing the nonzero basis functions at X and their
C                   derivatives columnwise.
C          WORK    - a work vector of length (K+1)*(K+2)/2
C
C     Error Conditions
C         Improper input is a fatal error
C
C***REFERENCES  Carl de Boor, Package for calculating with B-splines,
C                 SIAM Journal on Numerical Analysis 14, 3 (June 1977),
C                 pp. 441-472.
C***ROUTINES CALLED  DBSPVN, XERMSG
C***REVISION HISTORY  (YYMMDD)
C   800901  DATE WRITTEN
C   890531  Changed all specific intrinsics to generic.  (WRB)
C   890831  Modified array declarations.  (WRB)
C   890831  REVISION DATE from Version 3.2
C   891214  Prologue converted to Version 4.0 format.  (BAB)
C   900315  CALLs to XERROR changed to CALLs to XERMSG.  (THJ)
C   920501  Reformatted the REFERENCES section.  (WRB)
C***END PROLOGUE  DBSPVD
C
      INTEGER I,IDERIV,ILEFT,IPKMD,J,JJ,JLOW,JM,JP1MID,K,KMD, KP1, L,
     1 LDUMMY, M, MHIGH, NDERIV
      DOUBLE PRECISION FACTOR, FKMD, T, V, VNIKX, WORK, X
C     DIMENSION T(ILEFT+K), WORK((K+1)*(K+2)/2)
C     A(I,J) = WORK(I+J*(J+1)/2),  I=1,J+1  J=1,K-1
C     A(I,K) = WORK(I+K*(K-1)/2)  I=1.K
C     WORK(1) AND WORK((K+1)*(K+2)/2) ARE NOT USED.
      DIMENSION T(*), VNIKX(LDVNIK,*), WORK(*)
C***FIRST EXECUTABLE STATEMENT  DBSPVD
      IF(K.LT.1) GO TO 200
      IF(NDERIV.LT.1 .OR. NDERIV.GT.K) GO TO 205
      IF(LDVNIK.LT.K) GO TO 210
      IDERIV = NDERIV
      KP1 = K + 1
      JJ = KP1 - IDERIV
      CALL DBSPVN(T, JJ, K, 1, X, ILEFT, VNIKX, WORK, IWORK)
      IF (IDERIV.EQ.1) GO TO 100
      MHIGH = IDERIV
```

```
      DO 20 M=2,MHIGH
        JP1MID = 1
        DO 10 J=IDERIV,K
          VNIKX(J,IDERIV) = VNIKX(JP1MID,1)
          JP1MID = JP1MID + 1
   10   CONTINUE
        IDERIV = IDERIV - 1
        JJ = KP1 - IDERIV
        CALL DBSPVN(T, JJ, K, 2, X, ILEFT, VNIKX, WORK, IWORK)
   20 CONTINUE
C
      JM = KP1*(KP1+1)/2
      DO 30 L = 1,JM
        WORK(L) = 0.0D0
   30 CONTINUE
C     A(I,I) = WORK(I*(I+3)/2) = 1.0       I = 1,K
      L = 2
      J = 0
      DO 40 I = 1,K
        J = J + L
        WORK(J) = 1.0D0
        L = L + 1
   40 CONTINUE
      KMD = K
      DO 90 M=2,MHIGH
        KMD = KMD - 1
        FKMD = KMD
        I = ILEFT
        J = K
        JJ = J*(J+1)/2
        JM = JJ - J
        DO 60 LDUMMY=1,KMD
          IPKMD = I + KMD
          FACTOR = FKMD/(T(IPKMD)-T(I))
          DO 50 L=1,J
            WORK(L+JJ) = (WORK(L+JJ)-WORK(L+JM))*FACTOR
   50     CONTINUE
          I = I - 1
          J = J - 1
```

```
          JJ = JM
          JM = JM - J
   60   CONTINUE
C
        DO 80 I=1,K
          V = 0.0D0
          JLOW = MAX(I,M)
          JJ = JLOW*(JLOW+1)/2
          DO 70 J=JLOW,K
            V = WORK(I+JJ)*VNIKX(J,M) + V
            JJ = JJ + J + 1
   70     CONTINUE
          VNIKX(I,M) = V
   80   CONTINUE
   90 CONTINUE
  100 RETURN
C
C
  200 CONTINUE
c      CALL XERMSG ('SLATEC', 'DBSPVD', 'K DOES NOT SATISFY K.GE.1', 2,
c     +   1)
c      RETURN
  205 CONTINUE
c      CALL XERMSG ('SLATEC', 'DBSPVD',
c     +   'NDERIV DOES NOT SATISFY 1.LE.NDERIV.LE.K', 2, 1)
c      RETURN
  210 CONTINUE
c      CALL XERMSG ('SLATEC', 'DBSPVD',
c     +   'LDVNIK DOES NOT SATISFY LDVNIK.GE.K', 2, 1)
c      RETURN
      Write(*,*)'ERROR in DBSPVD'
      stop
c
      END

ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
      SUBROUTINE DBSPVN (T, JHIGH, K, INDEX, X, ILEFT, VNIKX, WORK,
     +   IWORK)
C***BEGIN PROLOGUE  DBSPVN
```

```
C***PURPOSE  Calculate the value of all (possibly) nonzero basis
C            functions at X.
C***LIBRARY   SLATEC
C***CATEGORY  E3, K6
C***TYPE      DOUBLE PRECISION (BSPVN-S, DBSPVN-D)
C***KEYWORDS  EVALUATION OF B-SPLINE
C***AUTHOR  Amos, D. E., (SNLA)
C***DESCRIPTION
C
C     Written by Carl de Boor and modified by D. E. Amos
C
C     Abstract    **** a double precision routine ****
C         DBSPVN is the BSPLVN routine of the reference.
C
C         DBSPVN calculates the value of all (possibly) nonzero basis
C         functions at X of order MAX(JHIGH,(J+1)*(INDEX-1)), where T(K)
C         .LE. X .LE. T(N+1) and J=IWORK is set inside the routine on
C         the first call when INDEX=1.  ILEFT is such that T(ILEFT) .LE.
C         X .LT. T(ILEFT+1).  A call to DINTRV(T,N+1,X,ILO,ILEFT,MFLAG)
C         produces the proper ILEFT.  DBSPVN calculates using the basic
C         algorithm needed in DBSPVD.  If only basis functions are
C         desired, setting JHIGH=K and INDEX=1 can be faster than
C         calling DBSPVD, but extra coding is required for derivatives
C         (INDEX=2) and DBSPVD is set up for this purpose.
C
C         Left limiting values are set up as described in DBSPVD.
C
C     Description of Arguments
C
C         Input      T,X are double precision
C          T       - knot vector of length N+K, where
C                    N = number of B-spline basis functions
C                    N = sum of knot multiplicities-K
C          JHIGH   - order of B-spline, 1 .LE. JHIGH .LE. K
C          K       - highest possible order
C          INDEX   - INDEX = 1 gives basis functions of order JHIGH
C                          = 2 denotes previous entry with work, IWORK
C                              values saved for subsequent calls to
C                              DBSPVN.
```

```
C          X       - argument of basis functions,
C                    T(K) .LE. X .LE. T(N+1)
C          ILEFT   - largest integer such that
C                    T(ILEFT) .LE. X .LT.  T(ILEFT+1)
C
C         Output     VNIKX, WORK are double precision
C          VNIKX   - vector of length K for spline values.
C          WORK    - a work vector of length 2*K
C          IWORK   - a work parameter.  Both WORK and IWORK contain
C                    information necessary to continue for INDEX = 2.
C                    When INDEX = 1 exclusively, these are scratch
C                    variables and can be used for other purposes.
C
C     Error Conditions
C         Improper input is a fatal error.
C
C***REFERENCES  Carl de Boor, Package for calculating with B-splines,
C                 SIAM Journal on Numerical Analysis 14, 3 (June 1977),
C                 pp. 441-472.
C***ROUTINES CALLED  XERMSG
C***REVISION HISTORY  (YYMMDD)
C   800901  DATE WRITTEN
C   890831  Modified array declarations.  (WRB)
C   890831  REVISION DATE from Version 3.2
C   891214  Prologue converted to Version 4.0 format.  (BAB)
C   900315  CALLs to XERROR changed to CALLs to XERMSG.  (THJ)
C   920501  Reformatted the REFERENCES section.  (WRB)
C***END PROLOGUE  DBSPVN
cccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
C
      INTEGER ILEFT, IMJP1, INDEX, IPJ, IWORK, JHIGH, JP1, JP1ML, K, L
      DOUBLE PRECISION T, VM, VMPREV, VNIKX, WORK, X
C     DIMENSION T(ILEFT+JHIGH)
      DIMENSION T(*), VNIKX(*), WORK(*)
C     CONTENT OF J, DELTAM, DELTAP IS EXPECTED UNCHANGED BETWEEN CALLS.
C     WORK(I) = DELTAP(I), WORK(K+I) = DELTAM(I), I = 1,K
C***FIRST EXECUTABLE STATEMENT  DBSPVN
      IF(K.LT.1) GO TO 90
      IF(JHIGH.GT.K .OR. JHIGH.LT.1) GO TO 100
```

```
      IF(INDEX.LT.1 .OR. INDEX.GT.2) GO TO 105
      IF(X.LT.T(ILEFT) .OR. X.GT.T(ILEFT+1)) GO TO 110
      GO TO (10, 20), INDEX
   10 IWORK = 1
      VNIKX(1) = 1.0D0
      IF (IWORK.GE.JHIGH) GO TO 40
C
   20 IPJ = ILEFT + IWORK
      WORK(IWORK) = T(IPJ) - X
      IMJP1 = ILEFT - IWORK + 1
      WORK(K+IWORK) = X - T(IMJP1)
      VMPREV = 0.0D0
      JP1 = IWORK + 1
      DO 30 L=1,IWORK
        JP1ML = JP1 - L
        VM = VNIKX(L)/(WORK(L)+WORK(K+JP1ML))
        VNIKX(L) = VM*WORK(L) + VMPREV
        VMPREV = VM*WORK(K+JP1ML)
   30 CONTINUE
      VNIKX(JP1) = VMPREV
      IWORK = JP1
      IF (IWORK.LT.JHIGH) GO TO 20
C
   40 RETURN
C
C
   90 CONTINUE
c     CALL XERMSG ('SLATEC', 'DBSPVN', 'K DOES NOT SATISFY K.GE.1', 2,
c    +   1)
c     RETURN
  100 CONTINUE
c     CALL XERMSG ('SLATEC', 'DBSPVN',
c    +   'JHIGH DOES NOT SATISFY 1.LE.JHIGH.LE.K', 2, 1)
c     RETURN
  105 CONTINUE
c     CALL XERMSG ('SLATEC', 'DBSPVN', 'INDEX IS NOT 1 OR 2', 2, 1)
c     RETURN
  110 CONTINUE
c     CALL XERMSG ('SLATEC', 'DBSPVN',
```

```
c    +   'X DOES NOT SATISFY T(ILEFT).LE.X.LE.T(ILEFT+1)', 2, 1)
c     RETURN
      write(*,*)"ERROR in DBSPVN"
      stop
      END

ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
      DOUBLE PRECISION FUNCTION DGAMLN (Z, IERR)
C*****BEGIN PROLOGUE  DGAMLN
C*****SUBSIDIARY
C*****PURPOSE  Compute the logarithm of the Gamma function
C*****LIBRARY   SLATEC
C*****CATEGORY  C7A
C*****TYPE      DOUBLE PRECISION (GAMLN-S, DGAMLN-D)
C*****KEYWORDS  LOGARITHM OF GAMMA FUNCTION
C*****AUTHOR  Amos, D. E., (SNL)
C*****DESCRIPTION
C
C               **** A DOUBLE PRECISION ROUTINE ****
C         DGAMLN COMPUTES THE NATURAL LOG OF THE GAMMA FUNCTION FOR
C         Z.GT.0.  THE ASYMPTOTIC EXPANSION IS USED TO GENERATE VALUES
C         GREATER THAN ZMIN WHICH ARE ADJUSTED BY THE RECURSION
C         G(Z+1)=Z*G(Z) FOR Z.LE.ZMIN.  THE FUNCTION WAS MADE AS
C         PORTABLE AS POSSIBLE BY COMPUTING ZMIN FROM THE NUMBER OF BASE
C         10 DIGITS IN A WORD, RLN=MAX(-ALOG10(R1MACH(4)),0.5E-18)
C         LIMITED TO 18 DIGITS OF (RELATIVE) ACCURACY.
C
C         SINCE INTEGER ARGUMENTS ARE COMMON, A TABLE LOOK UP ON 100
C         VALUES IS USED FOR SPEED OF EXECUTION.
C
C     DESCRIPTION OF ARGUMENTS
C
C         INPUT      Z IS DOUBLE PRECISION
C           Z      - ARGUMENT, Z.GT.0.0D0
C
C         OUTPUT      DGAMLN IS DOUBLE PRECISION
C           DGAMLN  - NATURAL LOG OF THE GAMMA FUNCTION AT Z.NE.0.0D0
C           IERR    - ERROR FLAG
C                     IERR=0, NORMAL RETURN, COMPUTATION COMPLETED
```

```
C                     IERR=1, Z.LE.0.0D0,    NO COMPUTATION
C
C
C***REFERENCES  COMPUTATION OF BESSEL FUNCTIONS OF COMPLEX ARGUMENT
C                 BY D. E. AMOS, SAND83-0083, MAY, 1983.
C***ROUTINES CALLED  D1MACH, I1MACH
C***REVISION HISTORY  (YYMMDD)
C   830501  DATE WRITTEN
C   830501  REVISION DATE from Version 3.2
C   910415  Prologue converted to Version 4.0 format.  (BAB)
C   920128  Category corrected.  (WRB)
C   921215  DGAMLN defined for Z negative.  (WRB)
C***END PROLOGUE  DGAMLN
cccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
      DOUBLE PRECISION CF, CON, FLN, FZ, GLN, RLN, S, TLG, TRM, TST,
     * T1, WDTOL, Z, ZDMY, ZINC, ZM, ZMIN, ZP, ZSQ, D1MACH
      INTEGER I, IERR, I1M, K, MZ, NZ, I1MACH
      DIMENSION CF(22), GLN(100)
C           LNGAMMA(N), N=1,100
      DATA GLN(1), GLN(2), GLN(3), GLN(4), GLN(5), GLN(6), GLN(7),
     1     GLN(8), GLN(9), GLN(10), GLN(11), GLN(12), GLN(13), GLN(14),
     2     GLN(15), GLN(16), GLN(17), GLN(18), GLN(19), GLN(20),
     3     GLN(21), GLN(22)/
     4     0.00000000000000000D+00,     0.00000000000000000D+00,
     5     6.93147180559945309D-01,     1.79175946922805500D+00,
     6     3.17805383034794562D+00,     4.78749174278204599D+00,
     7     6.57925121201010100D+00,     8.52516136106541430D+00,
     8     1.06046029027452502D+01,     1.28018274800814696D+01,
     9     1.51044125730755153D+01,     1.75023078458738858D+01,
     A     1.99872144956618861D+01,     2.25521638531234229D+01,
     B     2.51912211827386815D+01,     2.78992713838408916D+01,
     C     3.06718601060806728D+01,     3.35050734501368889D+01,
     D     3.63954452080330536D+01,     3.93398841871994940D+01,
     E     4.23356164607534850D+01,     4.53801388984769080D+01/
      DATA GLN(23), GLN(24), GLN(25), GLN(26), GLN(27), GLN(28),
     1     GLN(29), GLN(30), GLN(31), GLN(32), GLN(33), GLN(34),
     2     GLN(35), GLN(36), GLN(37), GLN(38), GLN(39), GLN(40),
     3     GLN(41), GLN(42), GLN(43), GLN(44)/
     4     4.84711813518352239D+01,     5.16066755677643736D+01,
```

```
5     5.47847293981123192D+01,     5.80036052229805199D+01,
6     6.12617017610020020D+01,     6.45575386270063311D+01,
7     6.78897431371815350D+01,     7.12570389671680090D+01,
8     7.46582363488301644D+01,     7.80922235533153106D+01,
9     8.15579594561150372D+01,     8.50544670175815174D+01,
A     8.85808275421976788D+01,     9.21361756036870925D+01,
B     9.57196945421432025D+01,     9.93306124547874269D+01,
C     1.02968198614513813D+02,     1.06631760260643459D+02,
D     1.10320639714757395D+02,     1.14034211781461703D+02,
E     1.17771881399745072D+02,     1.21533081515438634D+02/
 DATA GLN(45), GLN(46), GLN(47), GLN(48), GLN(49), GLN(50),
1     GLN(51), GLN(52), GLN(53), GLN(54), GLN(55), GLN(56),
2     GLN(57), GLN(58), GLN(59), GLN(60), GLN(61), GLN(62),
3     GLN(63), GLN(64), GLN(65), GLN(66)/
4     1.25317271149356895D+02,     1.29123933639127215D+02,
5     1.32952575035616310D+02,     1.36802722637326368D+02,
6     1.40673923648234259D+02,     1.44565743946344886D+02,
7     1.48477766951773032D+02,     1.52409592584497358D+02,
8     1.56360836303078785D+02,     1.60331128216630907D+02,
9     1.64320112263195181D+02,     1.68327445448427652D+02,
A     1.72352797139162802D+02,     1.76395848406997352D+02,
B     1.80456291417543771D+02,     1.84533828861449491D+02,
C     1.88628173423671591D+02,     1.92739047287844902D+02,
D     1.96866181672889994D+02,     2.01009316399281527D+02,
E     2.05168199482641199D+02,     2.09342586752536836D+02/
 DATA GLN(67), GLN(68), GLN(69), GLN(70), GLN(71), GLN(72),
1     GLN(73), GLN(74), GLN(75), GLN(76), GLN(77), GLN(78),
2     GLN(79), GLN(80), GLN(81), GLN(82), GLN(83), GLN(84),
3     GLN(85), GLN(86), GLN(87), GLN(88)/
4     2.13532241494563261D+02,     2.17736934113954227D+02,
5     2.21956441819130334D+02,     2.26190548323727593D+02,
6     2.30439043565776952D+02,     2.34701723442818268D+02,
7     2.38978389561834323D+02,     2.43268849002982714D+02,
8     2.47572914096186884D+02,     2.51890402209723194D+02,
9     2.56221135550009525D+02,     2.60564940971863209D+02,
A     2.64921649798552801D+02,     2.69291097651019823D+02,
B     2.73673124285693704D+02,     2.78067573440366143D+02,
C     2.82474292687630396D+02,     2.86893133295426994D+02,
D     2.91323950094270308D+02,     2.95766601350760624D+02,
```

```
     E     3.00220948647014132D+02,     3.04686856765668715D+02/
      DATA GLN(89), GLN(90), GLN(91), GLN(92), GLN(93), GLN(94),
     1     GLN(95), GLN(96), GLN(97), GLN(98), GLN(99), GLN(100)/
     2     3.09164193580146922D+02,     3.13652829949879062D+02,
     3     3.18152639620209327D+02,     3.22663499126726177D+02,
     4     3.27185287703775217D+02,     3.31717887196928473D+02,
     5     3.36261181979198477D+02,     3.40815058870799018D+02,
     6     3.45379407062266854D+02,     3.49954118040770237D+02,
     7     3.54539085519440809D+02,     3.59134205369575399D+02/
C             COEFFICIENTS OF ASYMPTOTIC EXPANSION
      DATA CF(1), CF(2), CF(3), CF(4), CF(5), CF(6), CF(7), CF(8),
     1     CF(9), CF(10), CF(11), CF(12), CF(13), CF(14), CF(15),
     2     CF(16), CF(17), CF(18), CF(19), CF(20), CF(21), CF(22)/
     3     8.33333333333333333D-02,    -2.77777777777777778D-03,
     4     7.93650793650793651D-04,    -5.95238095238095238D-04,
     5     8.41750841750841751D-04,    -1.91752691752691753D-03,
     6     6.41025641025641026D-03,    -2.95506535947712418D-02,
     7     1.79644372368830573D-01,    -1.39243221690590112D+00,
     8     1.34028640441683920D+01,    -1.56848284626002017D+02,
     9     2.19310333333333333D+03,    -3.61087712537249894D+04,
     A     6.91472268851313067D+05,    -1.52382215394074162D+07,
     B     3.82900751391414141D+08,    -1.08822660357843911D+10,
     C     3.47320283765002252D+11,    -1.23696021422692745D+13,
     D     4.88788064793079335D+14,    -2.13203339609193739D+16/
C
C             LN(2*PI)
      DATA CON                    /     1.83787706640934548D+00/
C
C***FIRST EXECUTABLE STATEMENT  DGAMLN
      IERR=0
      IF (Z.LE.0.0D0) GO TO 70
      IF (Z.GT.101.0D0) GO TO 10
      NZ = Z
      FZ = Z - NZ
      IF (FZ.GT.0.0D0) GO TO 10
      IF (NZ.GT.100) GO TO 10
      DGAMLN = GLN(NZ)
      RETURN
   10 CONTINUE
```

```
      WDTOL = D1MACH(4)
      WDTOL = MAX(WDTOL,0.5D-18)
      I1M = I1MACH(14)
      RLN = D1MACH(5)*I1M
      FLN = MIN(RLN,20.0D0)
      FLN = MAX(FLN,3.0D0)
      FLN = FLN - 3.0D0
      ZM = 1.8000D0 + 0.3875D0*FLN
      MZ = ZM + 1
      ZMIN = MZ
      ZDMY = Z
      ZINC = 0.0D0
      IF (Z.GE.ZMIN) GO TO 20
      ZINC = ZMIN - NZ
      ZDMY = Z + ZINC
   20 CONTINUE
      ZP = 1.0D0/ZDMY
      T1 = CF(1)*ZP
      S = T1
      IF (ZP.LT.WDTOL) GO TO 40
      ZSQ = ZP*ZP
      TST = T1*WDTOL
      DO 30 K=2,22
        ZP = ZP*ZSQ
        TRM = CF(K)*ZP
        IF (ABS(TRM).LT.TST) GO TO 40
        S = S + TRM
   30 CONTINUE
   40 CONTINUE
      IF (ZINC.NE.0.0D0) GO TO 50
      TLG = LOG(Z)
      DGAMLN = Z*(TLG-1.0D0) + 0.5D0*(CON-TLG) + S
      RETURN
   50 CONTINUE
      ZP = 1.0D0
      NZ = ZINC
      DO 60 I=1,NZ
        ZP = ZP*(Z+(I-1))
   60 CONTINUE
```

```
      TLG = LOG(ZDMY)
      DGAMLN = ZDMY*(TLG-1.0D0) - LOG(ZP) + 0.5D0*(CON-TLG) + S
      RETURN
C
C
   70 CONTINUE
      DGAMLN = D1MACH(2)
      IERR=1
      RETURN
      END

cccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
      SUBROUTINE DS2Y (N, NELT, IA, JA, A, ISYM)
C*****BEGIN PROLOGUE  DS2Y
C*****PURPOSE  SLAP Triad to SLAP Column Format Converter.
C            Routine to convert from the SLAP Triad to SLAP Column
C            format.
C*****LIBRARY   SLATEC (SLAP)
C*****CATEGORY  D1B9
C*****TYPE      DOUBLE PRECISION (SS2Y-S, DS2Y-D)
C*****KEYWORDS  LINEAR SYSTEM, SLAP SPARSE
C*****AUTHOR  Seager, Mark K., (LLNL)
C             Lawrence Livermore National Laboratory
C             PO BOX 808, L-60
C             Livermore, CA 94550 (510) 423-3141
C             seager@llnl.gov
C***DESCRIPTION
C
C *Usage:
C     INTEGER N, NELT, IA(NELT), JA(NELT), ISYM
C     DOUBLE PRECISION A(NELT)
C
C     CALL DS2Y( N, NELT, IA, JA, A, ISYM )
C
C *Arguments:
C N      :IN       Integer
C         Order of the Matrix.
C NELT   :IN       Integer.
C         Number of non-zeros stored in A.
```

```
C IA     :INOUT    Integer IA(NELT).
C JA     :INOUT    Integer JA(NELT).
C A      :INOUT    Double Precision A(NELT).
C         These arrays should hold the matrix A in either the SLAP
C         Triad format or the SLAP Column format.  See "Description",
C         below.  If the SLAP Triad format is used, this format is
C         translated to the SLAP Column format by this routine.
C ISYM   :IN       Integer.
C         Flag to indicate symmetric storage format.
C         If ISYM=0, all non-zero entries of the matrix are stored.
C         If ISYM=1, the matrix is symmetric, and only the lower
C         triangle of the matrix is stored.
C
C *Description:
C       The Sparse Linear Algebra Package (SLAP) utilizes two matrix
C       data structures: 1) the  SLAP Triad  format or  2)  the SLAP
C       Column format.  The user can hand this routine either of the
C       of these data structures.  If the SLAP Triad format is give
C       as input then this routine transforms it into SLAP Column
C       format.  The way this routine tells which format is given as
C       input is to look at JA(N+1).  If JA(N+1) = NELT+1 then we
C       have the SLAP Column format.  If that equality does not hold
C       then it is assumed that the IA, JA, A arrays contain the
C       SLAP Triad format.
C
C       =================== S L A P Triad format ===================
C       This routine requires that the  matrix A be   stored in  the
C       SLAP  Triad format.  In  this format only the non-zeros  are
C       stored.  They may appear in  *ANY* order.  The user supplies
C       three arrays of  length NELT, where  NELT is  the number  of
C       non-zeros in the matrix: (IA(NELT), JA(NELT), A(NELT)).  For
C       each non-zero the user puts the row and column index of that
C       matrix element  in the IA and  JA arrays.  The  value of the
C       non-zero   matrix  element is  placed  in  the corresponding
C       location of the A array.   This is  an  extremely  easy data
C       structure to generate.  On  the  other hand it   is  not too
C       efficient on vector computers for  the iterative solution of
C       linear systems.  Hence,   SLAP changes   this  input    data
C       structure to the SLAP Column format  for  the iteration (but
```

```
C       does not change it back).
C
C       Here is an example of the  SLAP Triad   storage format for a
C       5x5 Matrix.  Recall that the entries may appear in any order.
C
C           5x5 Matrix      SLAP Triad format for 5x5 matrix on left.
C                              1  2  3  4  5  6  7  8  9 10 11
C       |11 12  0  0 15|   A: 51 12 11 33 15 53 55 22 35 44 21
C       |21 22  0  0  0|  IA:  5  1  1  3  1  5  5  2  3  4  2
C       | 0  0 33  0 35|  JA:  1  2  1  3  5  3  5  2  5  4  1
C       | 0  0  0 44  0|
C       |51  0 53  0 55|
C
C       ================== S L A P Column format ==================
C
C       This routine  requires that  the matrix A  be stored in  the
C       SLAP Column format.  In this format the non-zeros are stored
C       counting down columns (except for  the diagonal entry, which
C       must appear first in each  "column")  and are stored  in the
C       double precision array A.   In other words,  for each column
C       in the matrix put the diagonal entry in  A.  Then put in the
C       other non-zero  elements going down  the column (except  the
C       diagonal) in order.   The  IA array holds the  row index for
C       each non-zero.  The JA array holds the offsets  into the IA,
C       A arrays  for  the  beginning  of each   column.   That  is,
C       IA(JA(ICOL)),  A(JA(ICOL)) points   to the beginning  of the
C       ICOL-th   column    in    IA and   A.      IA(JA(ICOL+1)-1),
C       A(JA(ICOL+1)-1) points to  the  end of the   ICOL-th column.
C       Note that we always have  JA(N+1) = NELT+1,  where N is  the
C       number of columns in  the matrix and NELT  is the number  of
C       non-zeros in the matrix.
C
C       Here is an example of the  SLAP Column  storage format for a
C       5x5 Matrix (in the A and IA arrays '|'  denotes the end of a
C       column):
C
C           5x5 Matrix      SLAP Column format for 5x5 matrix on left.
C                              1  2  3    4  5    6  7    8    9 10 11
C       |11 12  0  0 15|   A: 11 21 51 | 22 12 | 33 53 | 44 | 55 15 35
```

```
C       |21 22  0  0  0|  IA:  1  2  5 |  2  1 |  3  5 |  4 |  5  1  3
C       | 0  0 33  0 35|  JA:  1  4  6     8  9    12
C       | 0  0  0 44  0|
C       |51  0 53  0 55|
C
C***REFERENCES  (NONE)
C***ROUTINES CALLED  QS2I1D
C***REVISION HISTORY  (YYMMDD)
C   871119  DATE WRITTEN
C   881213  Previous REVISION DATE
C   890915  Made changes requested at July 1989 CML Meeting.  (MKS)
C   890922  Numerous changes to prologue to make closer to SLATEC
C           standard.  (FNF)
C   890929  Numerous changes to reduce SP/DP differences.  (FNF)
C   910411  Prologue converted to Version 4.0 format.  (BAB)
C   910502  Corrected C***FIRST EXECUTABLE STATEMENT line.  (FNF)
C   920511  Added complete declaration section.  (WRB)
C   930701  Updated CATEGORY section.  (FNF, WRB)
C***END PROLOGUE  DS2Y
C     .. Scalar Arguments ..
      INTEGER ISYM, N, NELT
C     .. Array Arguments ..
      DOUBLE PRECISION A(NELT)
      INTEGER IA(NELT), JA(NELT)
C     .. Local Scalars ..
      DOUBLE PRECISION TEMP
      INTEGER I, IBGN, ICOL, IEND, ITEMP, J
C     .. External Subroutines ..
      EXTERNAL QS2I1D
C***FIRST EXECUTABLE STATEMENT  DS2Y
C
C         Check to see if the (IA,JA,A) arrays are in SLAP Column
C         format.  If it's not then transform from SLAP Triad.
C
      IF( JA(N+1).EQ.NELT+1 ) RETURN
C
C         Sort into ascending order by COLUMN (on the ja array).
C         This will line up the columns.
C
```

```
      CALL QS2I1D( JA, IA, A, NELT, 1 )
C
C         Loop over each column to see where the column indices change
C         in the column index array ja.  This marks the beginning of the
C         next column.
Cccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
CVD$R NOVECTOR
      JA(1) = 1
      DO 20 ICOL = 1, N-1
         DO 10 J = JA(ICOL)+1, NELT
            IF( JA(J).NE.ICOL ) THEN
               JA(ICOL+1) = J
               GOTO 20
            ENDIF
 10      CONTINUE
 20   CONTINUE
      JA(N+1) = NELT+1
C
C         Mark the n+2 element so that future calls to a SLAP routine
C         utilizing the YSMP-Column storage format will be able to tell.
C
      JA(N+2) = 0
C
C         Now loop through the IA array making sure that the diagonal
C         matrix element appears first in the column.  Then sort the
C         rest of the column in ascending order.
C
      DO 70 ICOL = 1, N
         IBGN = JA(ICOL)
         IEND = JA(ICOL+1)-1
         DO 30 I = IBGN, IEND
            IF( IA(I).EQ.ICOL ) THEN
C
C              Swap the diagonal element with the first element in the
C              column.
C
               ITEMP = IA(I)
               IA(I) = IA(IBGN)
               IA(IBGN) = ITEMP
```

```
            TEMP = A(I)
            A(I) = A(IBGN)
            A(IBGN) = TEMP
            GOTO 40
          ENDIF
 30     CONTINUE
 40     IBGN = IBGN + 1
        IF( IBGN.LT.IEND ) THEN
          DO 60 I = IBGN, IEND
            DO 50 J = I+1, IEND
              IF( IA(I).GT.IA(J) ) THEN
                ITEMP = IA(I)
                IA(I) = IA(J)
                IA(J) = ITEMP
                TEMP = A(I)
                A(I) = A(J)
                A(J) = TEMP
              ENDIF
 50         CONTINUE
 60       CONTINUE
        ENDIF
 70   CONTINUE
      RETURN
C------------- LAST LINE OF DS2Y FOLLOWS ----------------------------
      END

ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
      SUBROUTINE DSORT (DX, DY, N, KFLAG)
C***BEGIN PROLOGUE  DSORT
C***PURPOSE  Sort an array and optionally make the same interchanges in
C            an auxiliary array.  The array may be sorted in increasing
C            or decreasing order.  A slightly modified QUICKSORT
C            algorithm is used.
C***LIBRARY   SLATEC
C***CATEGORY  N6A2B
C***TYPE      DOUBLE PRECISION (SSORT-S, DSORT-D, ISORT-I)
C***KEYWORDS  SINGLETON QUICKSORT, SORT, SORTING
C***AUTHOR  Jones, R. E., (SNLA)
C           Wisniewski, J. A., (SNLA)
```

```
C***DESCRIPTION
C
C   DSORT sorts array DX and optionally makes the same interchanges in
C   array DY.  The array DX may be sorted in increasing order or
C   decreasing order.  A slightly modified quicksort algorithm is used.
C
C   Description of Parameters
C      DX - array of values to be sorted   (usually abscissas)
C      DY - array to be (optionally) carried along
C      N  - number of values in array DX to be sorted
C      KFLAG - control parameter
C            =  2  means sort DX in increasing order and carry DY along.
C            =  1  means sort DX in increasing order (ignoring DY)
C            = -1  means sort DX in decreasing order (ignoring DY)
C            = -2  means sort DX in decreasing order and carry DY along.
C
C***REFERENCES  R. C. Singleton, Algorithm 347, An efficient algorithm
C                 for sorting with minimal storage, Communications of
C                 the ACM, 12, 3 (1969), pp. 185-187.
C***ROUTINES CALLED  XERMSG
C***REVISION HISTORY  (YYMMDD)
C   761101  DATE WRITTEN
C   761118  Modified to use the Singleton quicksort algorithm.  (JAW)
C   890531  Changed all specific intrinsics to generic.  (WRB)
C   890831  Modified array declarations.  (WRB)
C   891009  Removed unreferenced statement labels.  (WRB)
C   891024  Changed category.  (WRB)
C   891024  REVISION DATE from Version 3.2
C   891214  Prologue converted to Version 4.0 format.  (BAB)
C   900315  CALLs to XERROR changed to CALLs to XERMSG.  (THJ)
C   901012  Declared all variables; changed X,Y to DX,DY; changed
C           code to parallel SSORT. (M. McClain)
C   920501  Reformatted the REFERENCES section.  (DWL, WRB)
C   920519  Clarified error messages.  (DWL)
C   920801  Declarations section rebuilt and code restructured to use
C           IF-THEN-ELSE-ENDIF.  (RWC, WRB)
C***END PROLOGUE  DSORT
cccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
C     .. Scalar Arguments ..
```

```
      INTEGER KFLAG, N
C     .. Array Arguments ..
      DOUBLE PRECISION DX(*), DY(*)
C     .. Local Scalars ..
      DOUBLE PRECISION R, T, TT, TTY, TY
      INTEGER I, IJ, J, K, KK, L, M, NN
C     .. Local Arrays ..
      INTEGER IL(21), IU(21)
C     .. External Subroutines ..
c     EXTERNAL XERMSG
C     .. Intrinsic Functions ..
      INTRINSIC ABS, INT
C***FIRST EXECUTABLE STATEMENT  DSORT
      NN = N
      IF (NN .LT. 1) THEN
c        CALL XERMSG ('SLATEC', 'DSORT',
c    +      'The number of values to be sorted is not positive.', 1, 1)
c        RETURN
         WRITE(*,*)"ERROR in DSORT"
         STOP
      ENDIF
C
      KK = ABS(KFLAG)
      IF (KK.NE.1 .AND. KK.NE.2) THEN
c        CALL XERMSG ('SLATEC', 'DSORT',
c    +      'The sort control parameter, K, is not 2, 1, -1, or -2.', 2,
c    +      1)
c        RETURN
         WRITE(*,*)"ERROR in DSORT"
         STOP
      ENDIF
C
C     Alter array DX to get decreasing order if needed
C
      IF (KFLAG .LE. -1) THEN
         DO 10 I=1,NN
            DX(I) = -DX(I)
   10    CONTINUE
      ENDIF
```

```
C
      IF (KK .EQ. 2) GO TO 100
C
C     Sort DX only
C
      M = 1
      I = 1
      J = NN
      R = 0.375D0
C
   20 IF (I .EQ. J) GO TO 60
      IF (R .LE. 0.5898437D0) THEN
         R = R+3.90625D-2
      ELSE
         R = R-0.21875D0
      ENDIF
C
   30 K = I
C
C     Select a central element of the array and save it in location T
C
      IJ = I + INT((J-I)*R)
      T = DX(IJ)
C
C     If first element of array is greater than T, interchange with T
C
      IF (DX(I) .GT. T) THEN
         DX(IJ) = DX(I)
         DX(I) = T
         T = DX(IJ)
      ENDIF
      L = J
C
C     If last element of array is less than than T, interchange with T
C
      IF (DX(J) .LT. T) THEN
         DX(IJ) = DX(J)
         DX(J) = T
         T = DX(IJ)
```

```
C
C        If first element of array is greater than T, interchange with T
C
         IF (DX(I) .GT. T) THEN
            DX(IJ) = DX(I)
            DX(I) = T
            T = DX(IJ)
         ENDIF
      ENDIF
C
C     Find an element in the second half of the array which is smaller
C     than T
C
   40 L = L-1
      IF (DX(L) .GT. T) GO TO 40
C
C     Find an element in the first half of the array which is greater
C     than T
C
   50 K = K+1
      IF (DX(K) .LT. T) GO TO 50
C
C     Interchange these elements
C
      IF (K .LE. L) THEN
         TT = DX(L)
         DX(L) = DX(K)
         DX(K) = TT
         GO TO 40
      ENDIF
C
C     Save upper and lower subscripts of the array yet to be sorted
C
      IF (L-I .GT. J-K) THEN
         IL(M) = I
         IU(M) = L
         I = K
         M = M+1
      ELSE
```

```
            IL(M) = K
            IU(M) = J
            J = L
            M = M+1
         ENDIF
         GO TO 70
C
C        Begin again on another portion of the unsorted array
C
   60 M = M-1
      IF (M .EQ. 0) GO TO 190
      I = IL(M)
      J = IU(M)
C
   70 IF (J-I .GE. 1) GO TO 30
      IF (I .EQ. 1) GO TO 20
      I = I-1
C
   80 I = I+1
      IF (I .EQ. J) GO TO 60
      T = DX(I+1)
      IF (DX(I) .LE. T) GO TO 80
      K = I
C
   90 DX(K+1) = DX(K)
      K = K-1
      IF (T .LT. DX(K)) GO TO 90
      DX(K+1) = T
      GO TO 80
C
C        Sort DX and carry DY along
C
  100 M = 1
      I = 1
      J = NN
      R = 0.375D0
C
  110 IF (I .EQ. J) GO TO 150
      IF (R .LE. 0.5898437D0) THEN
```

```
            R = R+3.90625D-2
         ELSE
            R = R-0.21875D0
         ENDIF
C
  120 K = I
C
C     Select a central element of the array and save it in location T
C
      IJ = I + INT((J-I)*R)
      T = DX(IJ)
      TY = DY(IJ)
C
C     If first element of array is greater than T, interchange with T
C
      IF (DX(I) .GT. T) THEN
         DX(IJ) = DX(I)
         DX(I) = T
         T = DX(IJ)
         DY(IJ) = DY(I)
         DY(I) = TY
         TY = DY(IJ)
      ENDIF
      L = J
C
C     If last element of array is less than T, interchange with T
C
      IF (DX(J) .LT. T) THEN
         DX(IJ) = DX(J)
         DX(J) = T
         T = DX(IJ)
         DY(IJ) = DY(J)
         DY(J) = TY
         TY = DY(IJ)
C
C        If first element of array is greater than T, interchange with T
C
         IF (DX(I) .GT. T) THEN
            DX(IJ) = DX(I)
```

```
            DX(I) = T
            T = DX(IJ)
            DY(IJ) = DY(I)
            DY(I) = TY
            TY = DY(IJ)
         ENDIF
      ENDIF
C
C     Find an element in the second half of the array which is smaller
C     than T
C
  130 L = L-1
      IF (DX(L) .GT. T) GO TO 130
C
C     Find an element in the first half of the array which is greater
C     than T
C
  140 K = K+1
      IF (DX(K) .LT. T) GO TO 140
C
C     Interchange these elements
C
      IF (K .LE. L) THEN
         TT = DX(L)
         DX(L) = DX(K)
         DX(K) = TT
         TTY = DY(L)
         DY(L) = DY(K)
         DY(K) = TTY
         GO TO 130
      ENDIF
C
C     Save upper and lower subscripts of the array yet to be sorted
C
      IF (L-I .GT. J-K) THEN
         IL(M) = I
         IU(M) = L
         I = K
         M = M+1
```

```
      ELSE
         IL(M) = K
         IU(M) = J
         J = L
         M = M+1
      ENDIF
      GO TO 160
C
C     Begin again on another portion of the unsorted array
C
  150 M = M-1
      IF (M .EQ. 0) GO TO 190
      I = IL(M)
      J = IU(M)
C
  160 IF (J-I .GE. 1) GO TO 120
      IF (I .EQ. 1) GO TO 110
      I = I-1
C
  170 I = I+1
      IF (I .EQ. J) GO TO 150
      T = DX(I+1)
      TY = DY(I+1)
      IF (DX(I) .LE. T) GO TO 170
      K = I
C
  180 DX(K+1) = DX(K)
      DY(K+1) = DY(K)
      K = K-1
      IF (T .LT. DX(K)) GO TO 180
      DX(K+1) = T
      DY(K+1) = TY
      GO TO 170
C
C     Clean up
C
  190 IF (KFLAG .LE. -1) THEN
         DO 200 I=1,NN
            DX(I) = -DX(I)
```

```
  200    CONTINUE
      ENDIF
      RETURN
      END

cccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
SUBROUTINE QS2I1D (IA, JA, A, N, KFLAG)
C***BEGIN PROLOGUE  QS2I1D
C***SUBSIDIARY
C***PURPOSE  Sort an integer array, moving an integer and DP array.
C            This routine sorts the integer array IA and makes the same
C            interchanges in the integer array JA and the double pre-
C            cision array A.  The array IA may be sorted in increasing
C            order or decreasing order.  A slightly modified QUICKSORT
C            algorithm is used.
C***LIBRARY   SLATEC (SLAP)
C***CATEGORY  N6A2A
C***TYPE      DOUBLE PRECISION (QS2I1R-S, QS2I1D-D)
C***KEYWORDS  SINGLETON QUICKSORT, SLAP, SORT, SORTING
C***AUTHOR  Jones, R. E., (SNLA)
C           Kahaner, D. K., (NBS)
C           Seager, M. K., (LLNL) seager@llnl.gov
C           Wisniewski, J. A., (SNLA)
C***DESCRIPTION
C     Written by Rondall E Jones
C     Modified by John A. Wisniewski to use the Singleton QUICKSORT
C     algorithm. date 18 November 1976.
C
C     Further modified by David K. Kahaner
C     National Bureau of Standards
C     August, 1981
C
C     Even further modification made to bring the code up to the
C     Fortran 77 level and make it more readable and to carry
C     along one integer array and one double precision array during
C     the sort by
C     Mark K. Seager
C     Lawrence Livermore National Laboratory
C     November, 1987
```

```
C     This routine was adapted from the ISORT routine.
C
C     ABSTRACT
C         This routine sorts an integer array IA and makes the same
C         interchanges in the integer array JA and the double precision
C         array A.
C         The array IA may be sorted in increasing order or decreasing
C         order.  A slightly modified quicksort algorithm is used.
C
C     DESCRIPTION OF PARAMETERS
C        IA - Integer array of values to be sorted.
C        JA - Integer array to be carried along.
C         A - Double Precision array to be carried along.
C         N - Number of values in integer array IA to be sorted.
C     KFLAG - Control parameter
C           = 1 means sort IA in INCREASING order.
C           =-1 means sort IA in DECREASING order.
C
C***SEE ALSO  DS2Y
C***REFERENCES  R. C. Singleton, Algorithm 347, An Efficient Algorithm
C                 for Sorting With Minimal Storage, Communications ACM
C                 12:3 (1969), pp.185-7.
C***ROUTINES CALLED  XERMSG
C***REVISION HISTORY  (YYMMDD)
C   761118  DATE WRITTEN
C   890125  Previous REVISION DATE
C   890915  Made changes requested at July 1989 CML Meeting.  (MKS)
C   890922  Numerous changes to prologue to make closer to SLATEC
C           standard.  (FNF)
C   890929  Numerous changes to reduce SP/DP differences.  (FNF)
C   900805  Changed XERROR calls to calls to XERMSG.  (RWC)
C   910411  Prologue converted to Version 4.0 format.  (BAB)
C   910506  Made subsidiary to DS2Y and corrected reference.  (FNF)
C   920511  Added complete declaration section.  (WRB)
C   920929  Corrected format of reference.  (FNF)
C   921012  Corrected all f.p. constants to double precision.  (FNF)
C***END PROLOGUE  QS2I1D
cccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
CVD$R NOVECTOR
```

```
CVD$R NOCONCUR
C     .. Scalar Arguments ..
      INTEGER KFLAG, N
C     .. Array Arguments ..
      DOUBLE PRECISION A(N)
      INTEGER IA(N), JA(N)
C     .. Local Scalars ..
      DOUBLE PRECISION R, TA, TTA
      INTEGER I, IIT, IJ, IT, J, JJT, JT, K, KK, L, M, NN
C     .. Local Arrays ..
      INTEGER IL(21), IU(21)
C     .. External Subroutines ..
C     EXTERNAL XERMSG
C     .. Intrinsic Functions ..
      INTRINSIC ABS, INT
C***FIRST EXECUTABLE STATEMENT  QS2I1D
      NN = N
      IF (NN.LT.1) THEN
c        CALL XERMSG ('SLATEC', 'QS2I1D',
c    $      'The number of values to be sorted was not positive.', 1, 1)
         WRITE(*,*)'SLATEC', 'QS2I1D',
     $      'The number of values to be sorted was not positive.'
         RETURN
      ENDIF
      IF( N.EQ.1 ) RETURN
      KK = ABS(KFLAG)
      IF ( KK.NE.1 ) THEN
c        CALL XERMSG ('SLATEC', 'QS2I1D',
c    $      'The sort control parameter, K, was not 1 or -1.', 2, 1)
         WRITE(*,*)'SLATEC', 'QS2I1D',
     $      'The sort control parameter, K, was not 1 or -1.'
         RETURN
      ENDIF
C
C     Alter array IA to get decreasing order if needed.
C
      IF( KFLAG.LT.1 ) THEN
         DO 20 I=1,NN
            IA(I) = -IA(I)
```

```
 20      CONTINUE
      ENDIF
C
C     Sort IA and carry JA and A along.
C     And now...Just a little black magic...
      M = 1
      I = 1
      J = NN
      R = .375D0
 210  IF( R.LE.0.5898437D0 ) THEN
         R = R + 3.90625D-2
      ELSE
         R = R-.21875D0
      ENDIF
 225  K = I
C
C     Select a central element of the array and save it in location
C     it, jt, at.
C
      IJ = I + INT ((J-I)*R)
      IT = IA(IJ)
      JT = JA(IJ)
      TA = A(IJ)
C
C     If first element of array is greater than it, interchange with it.
C
      IF( IA(I).GT.IT ) THEN
         IA(IJ) = IA(I)
         IA(I)  = IT
         IT     = IA(IJ)
         JA(IJ) = JA(I)
         JA(I)  = JT
         JT     = JA(IJ)
         A(IJ)  = A(I)
         A(I)   = TA
         TA     = A(IJ)
      ENDIF
      L=J
C
```

```
C     If last element of array is less than it, swap with it.
C
      IF( IA(J).LT.IT ) THEN
         IA(IJ) = IA(J)
         IA(J)  = IT
         IT     = IA(IJ)
         JA(IJ) = JA(J)
         JA(J)  = JT
         JT     = JA(IJ)
         A(IJ)  = A(J)
         A(J)   = TA
         TA     = A(IJ)
C
C     If first element of array is greater than it, swap with it.
C
         IF ( IA(I).GT.IT ) THEN
            IA(IJ) = IA(I)
            IA(I)  = IT
            IT     = IA(IJ)
            JA(IJ) = JA(I)
            JA(I)  = JT
            JT     = JA(IJ)
            A(IJ)  = A(I)
            A(I)   = TA
            TA     = A(IJ)
         ENDIF
      ENDIF
C
C     Find an element in the second half of the array which is
C     smaller than it.
C
  240 L=L-1
      IF( IA(L).GT.IT ) GO TO 240
C
C     Find an element in the first half of the array which is
C     greater than it.
C
  245 K=K+1
      IF( IA(K).LT.IT ) GO TO 245
```

```
C
C     Interchange these elements.
C
      IF( K.LE.L ) THEN
         IIT   = IA(L)
         IA(L) = IA(K)
         IA(K) = IIT
         JJT   = JA(L)
         JA(L) = JA(K)
         JA(K) = JJT
         TTA   = A(L)
         A(L)  = A(K)
         A(K)  = TTA
         GOTO 240
      ENDIF
C
C     Save upper and lower subscripts of the array yet to be sorted.
C
      IF( L-I.GT.J-K ) THEN
         IL(M) = I
         IU(M) = L
         I = K
         M = M+1
      ELSE
         IL(M) = K
         IU(M) = J
         J = L
         M = M+1
      ENDIF
      GO TO 260
C
C     Begin again on another portion of the unsorted array.
C
  255 M = M-1
      IF( M.EQ.0 ) GO TO 300
      I = IL(M)
      J = IU(M)
  260 IF( J-I.GE.1 ) GO TO 225
      IF( I.EQ.J ) GO TO 255
```

```
      IF( I.EQ.1 ) GO TO 210
      I = I-1
  265 I = I+1
      IF( I.EQ.J ) GO TO 255
      IT = IA(I+1)
      JT = JA(I+1)
      TA =  A(I+1)
      IF( IA(I).LE.IT ) GO TO 265
      K=I
  270 IA(K+1) = IA(K)
      JA(K+1) = JA(K)
      A(K+1)  =  A(K)
      K = K-1
      IF( IT.LT.IA(K) ) GO TO 270
      IA(K+1) = IT
      JA(K+1) = JT
      A(K+1)  = TA
      GO TO 265
  300 IF( KFLAG.LT.1 ) THEN
         DO 310 I=1,NN
            IA(I) = -IA(I)
 310     CONTINUE
      ENDIF
      RETURN
      END
```

Appendix D: FORTRAN programs for spherical cap harmonic analysis of potential and general fields

These programs were made available by G.V. Haines (https://doi.org/10.1016/0098-3004(88)90027-1). They were used for chapter 7.

```
 PROGRAM SCHFIT

C     SPHERICAL CAP HARMONIC LEAST SQUARES FIT TO EITHER:
C     (1) POTENTIAL FIELD COMPONENTS NORTH X, EAST Y, VERTICAL DOWN Z; OR
C     (2) A GENERAL FUNCTION Z ON A SPHERICAL CAP SURFACE.
C     IN BOTH CASES, THE COEFFICIENTS ARE POLYNOMIALS IN TIME.

C     SUBPROGRAMS REQUIRED -  FNM, LEGFUN, SUMPRD, STEPREG, SPHNEWF
C             AND POSSIBLY -  GEOCENF, FIELD, PRTRES

!      include 'scparms'

!C      PARAMETER   (IBASIS=2,KINT=-1,LINT=-1,KEXT=10,LEXT=0)
      PARAMETER   (IBASIS=2)
      PARAMETER   (KINT=6, KEXT=0, KMAX=22)
      PARAMETER   (LINT=0, LEXT=-1, KT=0)
!C      PARAMETER   (KMAX=MAX(KINT,KEXT),KT=MAX(LINT,LEXT)) !!!!

C     Typically for magnetic field analyses of external field only:
```

```
C      ibasis=2, kint=-1, lint=-1, kext=6 to 10, lext=0
C     internal field only:
C      ibasis=2, kint=6 to 10, lint=0, kext=0, lext=-1
C     with internal and external field (not recommended for sparse data):
C      ibasis=2, kint=6 to 10, lint=0, kext=6 to 10, lext=0

C     IBASIS = 1  TO USE 1 SET OF BASIS FUNCTIONS (K-M=EVEN)
C                 (FOR FITTING VERTICAL FIELD DATA, OR GENERAL FUNCTION
C                  NOT DIFFERENTIABLE IN LATITUDE)
C              2  TO USE 2 SETS OF BASIS FUNCTIONS
C                 (FOR FITTING THREE-COMPONENT DATA, OR GENERAL FUNCTION
C                  DIFFERENTIABLE IN LATITUDE)
C     KINT  =  MAXIMUM SPATIAL INDEX FOR INTERNAL SOURCES
C     LINT  =  TEMPORAL DEGREE FOR INTERNAL SOURCES
C     KEXT  =  MAXIMUM SPATIAL INDEX FOR EXTERNAL SOURCES
C     LEXT  =  TEMPORAL DEGREE FOR EXTERNAL SOURCES

C     NOTE:    FOR NO INTERNAL SOURCES, PUT KINT=-1,LINT=-1.
C              FOR NO EXTERNAL SOURCES, PUT KEXT= 0,LEXT=-1.

C     EXAMPLE:  FOR MAIN FIELD, INTERNAL SOURCES, PUT LINT=KEXT=0,LEXT=-1.
C     FOR LINEAR TIME TERMS, INTERNAL SOURCES,  PUT LINT=1,KEXT=0,LEXT=-1.

      PARAMETER   (KB=2-IBASIS/2)
      PARAMETER   (NDIM=(((KINT+1)*(KINT+KB))/KB)*(LINT+1)
     *            +((KEXT*(KEXT+KB+1))/KB)*(LEXT+1)+1)

      DIMENSION   S(NDIM),SS(NDIM,NDIM)
      DIMENSION   XX(NDIM),XY(NDIM),XZ(NDIM)
      DIMENSION   FN(0:KMAX,0:KMAX), CONST(0:KMAX,0:KMAX)
      DIMENSION   CML(KMAX), SML(KMAX)
      DIMENSION   DELT(0:KT)
      DIMENSION   B(NDIM)
      DIMENSION   SIG(NDIM),IVAR(NDIM),BW(NDIM),SB(NDIM),ICBLNK(NDIM)
      CHARACTER*4 IE,ID1,ID2
      CHARACTER*60 FILEOUT,FILEREF
      EQUIVALENCE (XX,SIG), (XY,IVAR), (XZ,SB)

      ID1='DATA'
```

```
      ID2='DATA'

      IFIT =  1
C     PUT IFIT = -1 TO FIT A GENERAL FUNCTION OF LATITUDE AND LONGITUDE.
C                 0 TO FIT GEOCENTRIC VERTICAL POTENTIAL FIELD DATA.
C                 1 TO FIT THREE-COMPONENT POTENTIAL FIELD DATA.
      IF (IFIT .LT. 0)  THEN

         IF (KEXT .NE. 0)  GO TO 999
         IF (LEXT .NE. -1)  GO TO 999
      END IF

      ICEN = 1
C     PUT ICEN = 1 TO CONVERT INPUT DATA FROM GEODETIC TO GEOCENTRIC
C                  (INPUT DATA ARE GEODETIC LATITUDE, LONGITUDE, ALTITUDE)
C                0 FOR NO GEODETIC TO GEOCENTRIC CONVERSION
C                  (INPUT DATA ARE GEOCENTRIC LAT, LONG, RADIAL DISTANCE)
      IF (IFIT .LE. 0)  ICEN = 0

      IREF = 0
C     PUT IREF = 1 TO SUBTRACT REFERENCE FIELD FROM INPUT DATA
C                0 FOR NO SUBTRACTION.

      RE = 6371.2
C     PUT RE = REFERENCE RADIUS OF THE SPHERICAL CAP.

C     INPUT SPHERICAL CAP POLE (SUBROUTINE SPHNEWF)
        FLATO=-27.25!-28.0
        FLONO=24.75!24.0
      CALL SPHINF (FLATO,FLONO,DUM,DUM,DUM,DUM,DUM,DUM)

      TZERO = 2007.
C     TIME T, FOR SECULAR VARIATION POLYNOMIAL, WILL BE RELATIVE TO TZERO.

      ISUMS = 0
C     PUT ISUMS =  0  TO NEITHER PRINT NOR STORE SUMS OF SQUARES AND PRODUCTS
C                  1  TO STORE THEM
C                  2  TO STORE AND PRINT THEM
C                  3  TO PRINT THEM
```

```
      IRESID = 1
C     PUT IRESID = 1 FOR RESIDUALS TO BE CALCULATED AND PRINTED
C                  0 FOR NO RESIDUALS

      IFILES = 1
C     PUT IFILES = NUMBER OF END-OF-FILE RECORDS ON DATA FILE

      IPR = 1
C     Changed: Print only first input record in original form
C     SET IPR TO NUMBER OF INPUT RECORDS TO BE PRINTED.

C     READ IN PARAMETER FILE (REAL DEGREES, CONSTANTS)
C     (INPUT TO SUBROUTINE FNM MUST HAVE FN'S UP TO KMAX OR HIGHER)
      CALL FNM (KMAX,IBASIS,FN,CONST,THETA)
      CAPLAT = 90. - THETA

c      print*, 'SCPARMS.DAT gelesen!'
C     CHANGE CAPLAT IF DATA BEYOND THETA IS TO BE INCLUDED

      IF (IREF .EQ. 0)  GO TO 20

   20 NP1 = NDIM
      NP  = NDIM-1
      IBO = 0
      IENTER = 0
      NOB = 0
      !file of input data to fit
c      print*, 'Name of input xyz-data file:'
c      read '(A)', FILEOUT

      FILEOUT = 'sch_res2007_2008_03_01outX.txt'
      !FILEOUT = 'schaxyzal200512.dat'
      IT1 = 1
      OPEN (IT1,FILE=FILEOUT,STATUS='OLD')
      ! file of global reference
      !FILEREF='griRef_out.txt'
      FILEREF='igrfbcgal200512.dat'
      IT2 = 22
```

```
      OPEN (IT2,FILE=FILEREF,STATUS='OLD')

      IF (IRESID .EQ. 0)  GO TO 90
      IT91 = 91
      IT93 = 93
      IT94 = 94
      IT95 = 95
      OPEN (IT91,FILE='SCRES.BIN',FORM='UNFORMATTED')
      OPEN (IT93,FILE='XSCRES.BIN',FORM='UNFORMATTED')
      OPEN (IT94,FILE='YSCRES.BIN',FORM='UNFORMATTED')
      OPEN (IT95,FILE='ZSCRES.BIN',FORM='UNFORMATTED')

   90 IFILE  =  1
!C     SET DEFAULT FOR ALT AND T, IF NOT IN READ LIST
      IF (ICEN .NE. 0)  ALT = 400.
      IF (ICEN .EQ. 0)  ALT = RE
      T = TZERO
C     READ DATA FILE
  100 read(it1,*,end=200) flat,flon,falt,x,y,z
      ALT=falt
      IF (ICEN .EQ. 0)  GO TO 60
C     CONVERT FROM GEODETIC TO GEOCENTRIC
      CALL GEOCENF (FLAT,ALT,X,Z,GCLAT,R,BN,BV)
      GO TO 65
   60 GCLAT = FLAT
      R = ALT
      BN = X
      BV = Z
   65 IF (IREF .EQ. 0)  GO TO 70
C     REMOVE GLOBAL REFERENCE FIELD
      read(it2,*)flat,flon,falt,XREF,YREF,ZREF
      CALL GEOCENF (FLAT,ALT,XREF,ZREF,GCLAT,R,BNREF,BVREF)

      BN = BN - BNREF
      Y = Y - YREF
      BV = BV - BVREF

c     ROTATE SPHERICAL COORDINATE SYSTEM
```

```
 70 CALL SPHNEWF (GCLAT,FLON,BN,Y,SCLAT,SCLON,SCX,SCY)

    IF (IPR .EQ. 0)  GO TO 80
    IF (IPR .LT. 0)  GO TO 75
    IPR = -IPR

75  continue

103 FORMAT (4HFLAT,4X,4HFLON,5X,3HALT,
   *        7X,1HX,7X,1HY,7X,1HZ,3X,5HSCLAT,3X,5HSCLON,7X,1HR,
   *        5X,3HSCX,5X,3HSCY,6X,2HBV/)

    IPR = IPR + 1

 80 IF (SCLAT .LT. CAPLAT)  GO TO 100
    DELT(0) = 1.
    DEL = T - TZERO
    DO 102 I=1,KT
102 DELT(I) = DELT(I-1)*DEL
    IF (IFIT .GE. 0)  THEN
       AOR = RE/R
       AR = AOR**2
       IF (KEXT .GT. 0)  AOR3 = AOR*AR
    ELSE
       AR = -1.
       END IF

    K = 0
    IF (KINT .LT. 0)  GO TO 106
    DO 105 I=0,LINT
    ART = AR*DELT(I)
    K = K + 1
    IF (IFIT .LE. 0)  GO TO 105
    XX(K) =  0.
    XY(K) =  0.
105 XZ(K) =  -ART
106 COLAT = 90. - SCLAT
    DO 150 N=1,KMAX
    IF (N .GT. 1)  GO TO 115
```

```
      CL = COS(SCLON*3.1415927/180.0)
      SL = SIN(SCLON*3.1415927/180.0)
      CML(1) = CL
      SML(1) = SL
        GO TO 120
  115 SML(N) = SL*CML(N-1) + CL*SML(N-1)
      CML(N) = CL*CML(N-1) - SL*SML(N-1)
  120 CONTINUE
      DO 150 M=0,N
      IF (IBASIS .EQ. 2)  GO TO 121
      NMM = N-M
      IF ((NMM/2)*2 .NE. NMM)  GO TO 150
  121 FFN = FN(N,M)
      CALL LEGFUN (M,FFN,CONST(N,M),COLAT,P,DP,PMS,0)
      IF (IFIT .GE. 0)  THEN
         AR = AOR**(FFN+2.)
      ELSE

         AR = 1.
         FFN = -2.
      END IF
      IF (N .GT. KINT)  GO TO 135
      DO 130 I=0,LINT
      IF (M .NE. 0)  GO TO 125
      K = K + 1
      ART = AR*DELT(I)
      XZ(K) = -(FFN+1.)*ART*P
      IF (IFIT .LE. 0)  GO TO 130
      XX(K) =  ART*DP
      XY(K) =  0.
      GO TO 130
  125 K = K + 2
      ART = AR*DELT(I)
      ZT = -(FFN+1.)*ART*P
      XZ(K-1) =  ZT*CML(M)
      XZ(K) =  ZT*SML(M)
      IF (IFIT .LE. 0)  GO TO 130
      XT =  ART*DP
      XX(K-1) =  XT*CML(M)
```

```
    XX(K) =  XT*SML(M)
    YT =  ART*PMS
    XY(K-1) =  YT*SML(M)
    XY(K) = -YT*CML(M)
130 CONTINUE
135 IF (N .GT. KEXT)  GO TO 150
    RA = AOR3/AR
    DO 145 I=0,LEXT
    IF (M .NE. 0)  GO TO 140
    K = K + 1
    RAT = RA*DELT(I)
    XZ(K) =  FFN*RAT*P
    IF (IFIT .EQ. 0)  GO TO 145
    XX(K) =  RAT*DP
    XY(K) =  0.
    GO TO 145
140 K = K + 2
    RAT = RA*DELT(I)
    ZT =  FFN*RAT*P
    XZ(K-1) =  ZT*CML(M)
    XZ(K) =  ZT*SML(M)
    IF (IFIT .EQ. 0)  GO TO 145
    XT =  RAT*DP
    XX(K-1) =  XT*CML(M)
    XX(K) =  XT*SML(M)
    YT =  RAT*PMS
    XY(K-1) =  YT*SML(M)
    XY(K) = -YT*CML(M)
145 CONTINUE
150 CONTINUE
    !find out why this equality K=NP is to be satisfied
    !KINT must be equal to KMAX
    IF (K .NE. NP)  THEN
        GO TO 999
    END IF
    IF (IFIT .LE. 0)  GO TO 155
    XX(NP1) =  SCX
    CALL SUMPRD (IBO,NP1,NOBS,XX,S,SS,IENTER)
    XY(NP1) =  SCY
```

```
      CALL SUMPRD (IBO,NP1,NOBS,XY,S,SS,IENTER)
  155 XZ(NP1) =  BV
      CALL SUMPRD (IBO,NP1,NOBS,XZ,S,SS,IENTER)
      IF (IRESID .EQ. 0)  GO TO 100
C     SAVE VARIABLES TO COMPUTE RESIDUALS IN SUBROUTINE PRTRES.
      ID = T
      IF (IFIT .LE. 0)  GO TO 160
C     Store values to compute residuals for single components
      WRITE (IT93)  ID,ID1,(XX(K),K=1,NP1)
      WRITE (IT94)  ID,ID1,(XY(K),K=1,NP1)
      WRITE (IT95)  ID,ID1,(XZ(K),K=1,NP1)

      WRITE (IT91)  ID,ID1,(XX(K),K=1,NP1)
      WRITE (IT91)  ID,ID1,(XY(K),K=1,NP1)
  160 WRITE (IT91)  ID,ID1,(XZ(K),K=1,NP1)
      GO TO 100
C     END OF PROCESSING FILE=IFILE ON TAPE=IT1.
  200 NOB = NOBS - NOB
      NOB = NOBS
      IF (IFILE .GE. IFILES)  GO TO 300
      IFILE = IFILE + 1
      GO TO 100
C     END OF PROCESSING TAPE=IT1.
  300 CLOSE (IT1,STATUS='KEEP')

      IF (NOBS .LE. 0)  GO TO 999
      IF (IRESID .EQ. 0)  GO TO 310
      ENDFILE IT91
      ENDFILE IT93
      ENDFILE IT94
      ENDFILE IT95
C     STORE/PRINT SUMS OF SQUARES AND PRODUCTS MATRIX
  310 IF (ISUMS .LE. 0)  GO TO 320
      IF (ISUMS .GE. 3)  GO TO 315
      IT2 = 2
      OPEN (IT2,FILE='SUMSQUARES.DAT',STATUS='UNKNOWN')
      WRITE (IT2)  NOBS,NP1
      WRITE (IT2)  ((SS(I,J),J=I,NP1),I=1,NP1)
      CLOSE (IT2,STATUS='KEEP')
```

```
      IF (ISUMS .EQ. 1)  GO TO 320
 315  continue

  202 FORMAT (1X,10E13.6)
C     PERFORM STEPWISE REGRESSION.
  320 IBLNK =   1
      IORD  =   0
C       ** Original settings **
      FIN=3.5
      FOUT=3.5
      IPRT1 =   0
      IPRT2 =   2
      IPRTS =   0
      ITMAX = 2*NP
       CALL STEPREG (IBO,IBLNK,IORD,NOBS,NP1,S,SS,FIN,FOUT,ITMAX,B,
     *              SIG,IVAR,BW,SB,ICBLNK,IPRTS,IPRT1,IPRT2)

C     STORE SPHERICAL CAP HARMONIC COEFFICIENTS
      IT3 = 3
      OPEN (IT3,FILE='schcoef.dat',STATUS='UNKNOWN')
      write(*,*)FLATO,FLONO,THETA,RE,TZERO
      WRITE (IT3,245)  FLATO,FLONO,THETA,RE,TZERO,
     *                 IFIT,ICEN,IREF,IBASIS,KINT,LINT,KEXT,LEXT
  245 FORMAT (F8.2,F9.2,F7.2,2F8.1,8I5)
      K = 0
      LI = LINT + 1
      LE = LEXT + 1

      DO 270 N=0,KMAX !loop to write coefficients
      IF (IBASIS .EQ. 2)  GO TO 230
      IF ((N/2)*2 .NE. N)  GO TO 260
  230 IF (N .GT. KINT)  GO TO 259
      IE = 'I'
      KO = K + 1
      K  = K + LI
      KF = LI

  276 DO 247 J=KO,K
  247 IF (B(J) .NE. 0.)  GO TO 248
```

```
      GO TO 258
248   M = 0
      WRITE (IT3,250) IE,N,M,FN(N,M),CONST(N,M),(B(J),J=KO,K)
250   FORMAT (1X,A1,2I3,F9.4,E15.6,(F15.3,10X))
255   FORMAT (1X,A1,2I3,F9.4,E15.6,F10.3,3F20.3:/(22X,4F20.3))
258   IF (IE .EQ. 'E')  GO TO 260
259   IF (N .GT. KEXT)  GO TO 260
      IF (N .EQ. 0)  GO TO 260
      IE = 'E'
      KO = K + 1
      K  = K + LE
      KF = LE
      GO TO 276
260   DO 270 M=1,N
      IF (IBASIS .EQ. 2)  GO TO 240
      NMM = N - M
      IF ((NMM/2)*2 .NE. NMM)  GO TO 270
240   IF (N .GT. KINT)  GO TO 269
      IE = 'I'
      KO = K + 1
      K  = K + LI + LI
      KF = LI
261   DO 262 J=KO,K  !inner loop
262   IF (B(J) .NE. 0.)  GO TO 265
      GO TO 268
265   WRITE (IT3,280)  IE,N,M,FN(N,M),CONST(N,M),(B(J),J=KO,K)
280   FORMAT (1X,A1,2I3,F9.4,E15.6,(F15.3,F15.3))
285   FORMAT (1X,A1,2I3,F9.4,E15.6,8F10.3:/(32X,8F10.3))
268   IF (IE .EQ. 'E')  GO TO 270
269   IF (N .GT. KEXT)  GO TO 270
      IE = 'E'
      KO = K + 1
      K  = K + LE + LE
      KF = LE
      GO TO 261
270   CONTINUE   !end of the loop
      WRITE (IT3,255) ' ',-1,-1
      CLOSE (IT3,STATUS='KEEP')
       IF (IRESID .EQ. 0)  GO TO 999
```

```
C     COMPUTE AND PRINT SPHERICAL CAP HARMONIC RESIDUALS
C     Compute and show residuals for all components AND for single ones
 2301   FORMAT(20H all components        )
      CALL PRTRES (IT91,IBO,NOBS,NP1,XX,B)
 2303   FORMAT(20H X-component        )
      CALL PRTRES (IT93,IBO,NOBS,NP1,XX,B)
 2304   FORMAT(20H Y-component        )
      CALL PRTRES (IT94,IBO,NOBS,NP1,XX,B)
 2305   FORMAT(20H Z-component        )
      CALL PRTRES (IT95,IBO,NOBS,NP1,XX,B)
      CLOSE (IT91,STATUS='DELETE')
      CLOSE (IT93,STATUS='DELETE')
      CLOSE (IT94,STATUS='DELETE')
      CLOSE (IT95,STATUS='DELETE')
  999 STOP
      END   !END OF SCHFIT PROGRAM

c========================================================================
      PROGRAM SCHE

C     SPHERICAL CAP HARMONIC EXPANSION AT INTERVALS OF LAT, LONG, ALT.

C     SUBPROGRAMS USED:  GEOCENF, SPHNEWF, SCHNEV, FIELD.

      CHARACTER*60 fileout,infile,fileref,modelfile,modelxyzf

      PRINT 1
    1 FORMAT (14H1PROGRAM  SCHE)



C     INPUT PARAMETER AND COEFFICIENT FILE
      CALL SCHNEV (DUM,DUM,DUM,DUM,0,FLATO,FLONO,THETA)                 COEFFS

C     DEFINE SPHERICAL CAP POLE
      CALL SPHITF (FLATO,FLONO,DUM,DUM,DUM,DUM,DUM,DUM)

      IREF=0
C     PUT IREF .EQ. 0 IF NO REFERENCE FIELD IS TO BE ADDED.
C     PUT IREF .NE. 0 IF ONE IS TO BE ADDED.
```

```
C     PRINT 4,  IREF
C   4 FORMAT (/7X,4HIREF/1X,I10)

      IF (IREF .EQ. 0)  GO TO 7
C     READ (*,5,END=99)  TF,NMAX
C   5 FORMAT (F10.2,I5)
C     PRINT 6,  TF,NMAX
C   6 FORMAT (/9X,2HTF,6X,4HNMAX/1X,F10.2,I10)
C     INPUT COEFFICIENTS OF GLOBAL REFERENCE FIELD
C     LF = 0
C     CALL FIELD (DUM,DUM,DUM,TF,NMAX,LF,DUM,DUM,DUM)
C     LF = 1
      !model file in h,d, z and f components

       modelfile='schadhzfalm.dat'
        open(13,file=modelfile,status='replace')

       !model file in x,y, z and f components
        modelxyzf='schaxyzfalm.dat'
        open(15,file=modelxyzf,status='replace')
      ! file of global reference
        FILEREF='igrfbcgal.dat'

      IT2 = 22

      OPEN (IT2,FILE=FILEREF,STATUS='OLD')

C    7 PRINT*, 'Please give Latitude: Begin End Step'
C      READ (*,*) FLAT1,FLAT2,DLAT
C      PRINT*, 'Please give Longitude: Begin End Step'
C      READ (*,*) FLON1,FLON2,DLON
    7 FLAT1=-40.0
      FLAT2=-10.0
      DLAT=1.0
      FLON1=10.0
      FLON2=40.0
      DLON=1.0
      !model file in h,d, z and f components
       modelfile='schadhzfal.dat'
```

```
        open(13,file=modelfile,status='replace')

       !model file in x,y, z and f components
        modelxyzf='schaxyz_K6_18d14_0301.dat'
        open(15,file=modelxyzf,status='replace')
C     PRINT*, 'Please give FLAT1, FLAT2, DLAT, FLON1, FLON2, DLON,'
C     PRINT*, '            ALT1, ALT2, DALT:'

C     READ (*,*,END=99) FLAT1,FLAT2,DLAT,FLON1,FLON2,DLON,
C    *         ALT1,ALT2,DALT                                          INPUT
C  10 FORMAT(9F6.0,I5)

      IGRAT=0
C     PUT IGRAT .EQ. 0 FOR COMPUTATIONS AT LAT=FLAT1,FLAT2,DLAT
C                      AND LONG=FLON1,FLON2,DLON.
C               .NE. 0 FOR COMPUTATIONS AT LAT=FLAT1,FLAT2,DLAT
C                      AND LONG=FLON1,FLON2,DLAT/COS(LAT).

c      PRINT*, 'Please give L:'
c      print*, '   L=1: internal field only'
c      print*, '   L=2: external field only'
c      print*, '   L=3: both'
C     print*, '   ICEN=0: geocentric components on geocentric grid'
C     print*, '   ICEN=1: geodetic components, geodetic grid'

C     READ (*,12,END=99)  L,ICEN,IDHF,ICHANGE,T,DELT                   INPUT
c      READ (*,*,END=99)  L
      L=1
C  12 FORMAT (4I6,2F6.0)
   12 FORMAT (2I6)

C     Set IHDF,ICHANGE,T,DELT to zero by default
      IDHF=0
      ICHANGE=0
      T=0
      DELT=0
      ICEN=1
!      read(*,1234) infile
!      read(*,1234) fileout
```

```
 1234 format(a20)

C     Ask for and Open output file for (lat,lon,alt) model data
c      print*, 'Please give the name of the output file for the field'
c      print*, 'values:'
c      read '(A)',fileout
      fileout='testfit.dat'

      !file used to compute rms/original data to model
      infile='sch_res2007_2008_03_01.txt'
  !file used to compute data at a given altitude
      !infile='data_fixed_altitude.txt'

      IOUT=92
      OPEN (UNIT=IOUT, FILE=fileout)
      write (IOUT,*) 'long lat x y z'
      open(10,file=infile)

C     PUT L .EQ.  1 FOR INTERNAL POTENTIAL FIELD,
C           .EQ.  2 FOR EXTERNAL POTENTIAL FIELD,
C           .GE.  3 FOR BOTH INTERNAL AND EXTERNAL POTENTIAL FIELDS.
C           .EQ.  0 FOR GEOCENTRIC INTERNAL POTENTIAL FIELD.
C           .LE. -1 FOR GENERAL FUNCTION AND TWO SURFACE DERIVATIVES.
C     PUT ICEN .NE. 0  FOR GEODETIC COMPONENTS ON GEODETIC LAT-LONG-ALT GRID.
C              .EQ. 0  FOR GEOCENTRIC COMPONENTS ON GEOCENTRIC LAT-LONG-R GRID.
C     PUT IDHF .NE. 0  TO COMPUTE I,D,H,F
C              .EQ. 0  FOR NO I,D,H,F
C     PUT ICHANGE .EQ. 0 FOR FIELD VALUES AT TIME T.
C                 .NE. 0 FOR AVERAGE SECULAR CHANGE PER UNIT TIME
C                        FROM TIME T TO TIME T+DELT.

      IF (L .LE. 0)  ICEN = 0                                           NOTE
      IF (L .LT. 0)  IDHF = 0                                           NOTE
      IF (DELT .EQ. 0.)  ICHANGE = 0
      PRINT 14,  L,ICEN,IDHF,ICHANGE,T,DELT
   14 FORMAT (/10X,1HL,6X,4HICEN,6X,4HIDHF,3X,7HICHANGE,
     *         9X,1HT,6X,4HDELT/ 1X,4I10,2F10.2)
      LINE = 1
      NREC = 0
```

```
     !CHANGE ALT AS NECESSARY
     ! ALT=400.
      xsum=0.
      ysum=0.
      zsum=0.
      dsum=0.
      hsum=0.
      zsum=0.
      fsum=0.
      nn=0
   30 read(10,*,end=99) FLAT,FLON,falt,xx,yy,zz
       ALT=falt
       hh=sqrt(xx**2+yy**2)
       dd=asin(yy/hh)*180/3.1415927
       ff=sqrt(xx**2+yy**2+zz**2)

      IF (IGRAT .EQ. 0)  GO TO 35
      CFLAT = COS(FLAT*3.1415927/180.0)
      IF (CFLAT .NE. 0.)  GO TO 58
      DLON = 720.
      GO TO 35
   58 DLON = DLAT/CFLAT
      GO TO 35

   35 IF (ICEN .EQ. 0)  GO TO 36
C     CONVERT FROM GEODETIC TO GEOCENTRIC COORDINATES
C     CALL GEOCENF (FLAT,ALT,DUM,DUM,GCFLAT,R,DUM,DUM)
      CALL GEOCENF (FLAT,ALT,xx,yy,GCFLAT,R,DUM,DUM)
      GO TO 37
   36 GCFLAT = FLAT
      R = ALT

C     CONVERT SPHERICAL TO SPHERICAL CAP COORDINATE SYSTEM
   37 CALL SPHNEWF (GCFLAT,FLON,DUM,DUM,SCFLAT,SCFLON,DUM,DUM)

C     COMPUTE SPHERICAL CAP HARMONIC FIELD
      CALL SCHNEV (SCFLAT,SCFLON,R,T,L,BNSC,BESC,BV)

!C     ROTATE S.C. COORDINATES BACK TO SPHERICAL COORDINATES
```

```
      CALL SPHOLDF (DUM,DUM,BN,Y,DUM,DUM,BNSC,BESC)

      IF (IREF .EQ. 0)  GO TO 40
C     ADD GLOBAL REFERENCE FIELD TO S.C. RESIDUAL FIELD.
C     CALL FIELD (GCFLAT,FLON,R,TF,NMAX,LF,XREF,YREF,ZREF)
      read(it2,*) FLAT,FLON,FALT,xref,yref,zref

C     CONVERT FROM GEODETIC TO GEOCENTRIC COORDINATES
C      CALL GEOCENF (FLAT,ALT,DUM,DUM,GCFLAT,R,DUM,DUM)
      CALL GEOCENF(FLAT,ALT,XREF,ZREF,GCFLAT,R,BNREF,BVREF)
      BN=BN+BNREF
      BV=BV+BVREF
      Y =Y +YREF

C     BN = BN + XREF  !this case xref taken as geocentric
C     Y = Y + YREF
C     BV = BV + ZREF

   40 IF (ICEN .EQ. 0) GO TO 42
C     CONVERT CEOCENTRIC COORDINATES BACK TO GEODETIC
      CALL GEOCINV (DUM,DUM,X,Z,DUM,DUM,BN,BV)
      GO TO 44
   42 X = BN
      Z = BV
   44 CONTINUE

      IF (IDHF .EQ. 0) GO TO 50
      HSQ = X**2 + Y**2
      H = SQRT(HSQ)
      D = 0.
      IF (H .NE. 0.)  D=ATAN2(Y,X)*180.0/3.1415927
      F  = SQRT(HSQ +Z**2)
      FI = 90.
      IF (H .NE. 0.)  FI = ATAN(Z/H)*180.0/3.1415927

   50 IF (ICHANGE .EQ. 0)  GO TO 78
      CALL SCHNEV (SCFLAT,SCFLON,R,T+DELT,L,BNSC2,BESC2,BV2)
      CALL SPHOLDF (DUM,DUM,BN2,Y2,DUM,DUM,BNSC2,BESC2)
      IF (IREF .EQ.0)  GO TO 77
```

```
C     BN2 = BN2 + XREF
C     Y2 = Y2 + YREF
C     BV2 = BV2 + ZREF
   77 IF (ICEN .EQ. 0)  GO TO 52
      CALL GEOCINV (DUM,DUM,X2,Z2,DUM,DUM,BN2,BV2)
      GO TO 54
   52 X2 = BN2
      Z2 = BV2
   54 X = (X2-X)/DELT
      Y = (Y2-Y)/DELT
      Z = (Z2-Z)/DELT

      IF (IDHF .EQ. 0)  GO TO 78
      H2SQ = X2**2 + Y2**2
      H2 = SQRT(H2SQ)
      H = (H2-H)/DELT
      D2 = 0.
      IF (H2 .NE. 0.)  D2=ATAN2(Y2,X2)*180.0/3.1415927
      D = D2 - D
      IF (D .GT. 180.)  D=D-360.
      IF (D .LT. -180.)  D = D+360.
      D = D*60./DELT
      FI2 = 90.
      IF (H2 .NE. 0.)  FI2 = ATAN(Z2/H2)*180.0/3.1415927
      FI = (FI2-FI)*60./DELT
C     D AND I SECULAR CHANGE ARE IN MINUTES PER YEAR.
      F2 = SQRT(H2SQ+Z2**2)
      F = (F2 - F)/DELT

   78 IF (LINE .LT. 0)  GO TO 60
   80 FORMAT (1H1,4X,3HLAT,4X,4HLONG,7X,1HX,7X,1HY,7X,1HZ)
      LINE = -57
      IF (IDHF .EQ. 0)  GO TO 83
   82 FORMAT (1H+,53X,1HD,7X,1HI,7X,1HH,7X,1HF)
   83 continue
      GO TO 90
   60 LINE = LINE + 1

   90 continue
```

```
   81 FORMAT (1X,F7.1,F7.2,F8.2,3F8.0)

!C     Write data also to file fileout and modelfile
      xx=xx-x
      yy=yy-y
      zz=zz-z
      xsum=xsum+xx*xx
      ysum=ysum+yy*yy
      zsum=zsum+zz*zz
      h=sqrt(x**2+y**2)
      d=asin(y/h)*180/3.1415927
      f=sqrt(x**2+y**2+z**2)

      dd=dd-d
      hh=hh-h
      ff=ff-f
      dsum=dsum+dd*dd
      hsum=hsum+hh*hh
      fsum=fsum+ff*ff
      nn=nn+1
      write (IOUT,1111) FLON,FLAT,X,Y,Z,xx,yy,zz
      write (13,1113)flat,flon,d,h,z,f
      write (15,1114)FLAT,FLON,X,Y,Z,f
 1111 format (2F10.2,3F10.2,3f10.3)
 1113 format (2F8.3,F8.3,F8.1,F9.1,f8.1)
 1114 format (2F8.3,F8.1,F8.1,F9.1,f8.1)
      NREC = NREC + 1
      IF (IDHF .EQ. 0)  GO TO 30
      PRINT 84, D,FI,H,F
   84 FORMAT (1H+,46X,2F8.2,2F8.0)
      GO TO 30


 2222 format(4f12.5)
 2223 format(4f12.5)
 2224 format(3f12.5)

   99 CLOSE (UNIT=IOUT,STATUS='KEEP')
      close(10)
```

```
      write(*,*) '        rms       rmsX        rmsY        rmsZ'
      xfit=sqrt(xsum/nn)
      yfit=sqrt(ysum/nn)
      zfit=sqrt(zsum/nn)
      fit=sqrt((xsum+ysum+zsum)/(3*nn))
      write(*,2222) fit,xfit,yfit,zfit
      write(*,*) 'normal termination'
      write(15,*) '      rmsX        rmsY        rmsZ'
      write(15,2224)xfit,yfit,zfit
      close(15)
      !for modelfile in D, H and Z
       write(13,*) '     rmsD        rmsH        rmsZ       rmsF'
      dfit=sqrt(dsum/nn)
      hfit=sqrt(hsum/nn)
      zfit=sqrt(zsum/nn)
      ffit=sqrt(fsum/nn)
      write(13,2223) dfit,hfit,zfit,ffit
      close(13)
      END
c==============================================================
      PROGRAM FNROOT
C     OBTAIN REAL (NON-INTEGRAL) DEGREES OF LEGENDRE FUNCTIONS
C     FOR SPHERICAL CAP HARMONIC EXPANSION.

C     SUBPROGRAMS REQUIRED:  PDPFUN, RTMI
C     NOTE:  RTMI IS AN IBM SCIENTIFIC SUBROUTINE.

      EXTERNAL  PDPFUN
      COMMON    COLAT, M, IPDP, CON
      character*30 fileout

C     INPUT FROM KEYBOARD (added by O.A.)
c      print*, 'Please give COLAT KMAX:'
c      read*,COLAT, KMAX
       print*
c      print*,'Please give the name of the output file for the n_k(m):'
c      read '(A)', fileout
      fileout = 'scparms.dat'
```

```
      COLAT =  18.                                              ! SET UP
      IBASIS = 2                                                ! SET UP
      KMAX  = 22                                                ! SET UP

C     COLAT  = HALF-ANGLE OF SPHERICAL CAP, IN DEGREES.
C     IBASIS = 1  GIVES ONE SET OF BASIS FUNCTIONS (FOR FITTING FUNCTIONS
C                 THAT NEED NOT BE DIFFERENTIATED WITH RESPECT TO COLATITUDE)
C              2  GIVES TWO SETS (FOR FITTING FUNCTIONS THAT NEED TO BE
C                 DIFFERENTIATED WITH RESPECT TO COLATITUDE)
C     KMAX   = MAXIMUM INDEX AT WHICH REAL DEGREES ARE TO BE CALCULATED.

      IT1=1
      OPEN (UNIT=IT1,FILE=fileout,STATUS='OLD')
      WRITE (IT1,90)  KMAX,KMAX,COLAT,IBASIS
   90 FORMAT (2I3,F9.4,I5)
      MM = KMAX
      KM1   = KMAX - 1
      IPDPL = 3 - IBASIS
      RINC  = 180./COLAT

      DO 200 IPDP=2,IPDPL,-1
C     IPDP   = 1  GIVES ROOTS OF  P
C              2  GIVES ROOTS OF DP
      PRINT 1
    1 FORMAT (1H1,4X,1HK,4X,1HM,13X,2HRT,17X,3HFRT,2X,3HIER,17X,3HCON)
      IF (IPDP .EQ. 1)  MM = KM1
      DO 200 M=0,MM
      RT = 1.2*M - RINC/2.
C     USE 1.2*M RATHER THAN M IF NORMALIZING CONSTANT NOT COMPUTED AT N=M.
      K = M - IPDP
      IEND = 20

  100 RTL = RT + RINC/2.
      RTR = RTL + RINC
      EPS = 0.5E-4
      IF (RTR .GT. 1.)  EPS = EPS/RTR
      CALL RTMI (RT,FRT,PDPFUN,RTL,RTR,EPS,IEND,IER)
C     IER = 0   NO ERROR
```

```
C           1   NO CONVERGENCE AFTER IEND ITERATIONS
C           2   PDPFUN(RTL) SAME SIGN AS PDPFUN(RTR)
      IF (IER .NE. 2)  GO TO 145
      RT = (RTL+RTR)/2.
      GO TO 170
  145 K = K + 2
      PRINT 150, K,M,RT,FRT,IER,CON
  150 FORMAT (/1X,2I5,F15.7,F20.14,I5,E20.12)
      WRITE (IT1,160)  K,M,RT,CON
  160 FORMAT (2I3,F9.4,E15.6)
  170 IF (K .LT. KM1)  GO TO 100
  200 CONTINUE

      K = -1
      WRITE (IT1,160) K,K
      CLOSE (UNIT=IT1,STATUS='KEEP')
      STOP
      END
C   END OF SCHE PROGRAM
c==================================================================
c SUBROUTINES
c -----------------------------------------------------------------
      SUBROUTINE LEGFUN (M,FN,CONST,COLAT,P,DP,PMS,IPRT)
C     SERIES FORM FOR ASSOCIATED LEGENDRE FUNCTION P, ITS DERIVATIVE DP,
C     AND THE FUNCTION PMS=P*M/SIN(COLAT), IN POWERS OF (1-COS(COLAT))/2.
C     INTEGRAL ORDER M, REAL DEGREE FN, NORMALIZING CONSTANT CONST.
C     COLATITUDE COLAT IN DEGREES.
C     IPRT = 0     NO PRINT-OUT
C            1     PRINT PARAMETERS AND P SERIES
C            2     PRINT PARAMETERS AND DP SERIES
C            3     PRINT PARAMETERS AND BOTH P AND DP SERIES
C           -1     PRINT PARAMETERS ONLY
C     INPUT M,FN,CONST,COLAT,IPRT.   OUTPUT P,DP,PMS.
      SAVE
      REAL*8  FNN,AL,A,B,PNM,DPNM
      DIMENSION  AM(60), BM(60)
      DATA   JMAX/60/
      FNN = FN*(FN+1.)
      IF (COLAT .LT. 60.)  THEN
```

```
         X = SIN(COLAT/2.*3.1415927/180.0)**2
         C = 1. - 2.*X
      ELSE
         C = COS(COLAT*3.1415927/180.0)
         X = (1. - C)/2.
         END IF
      S = SIN(COLAT*3.1415927/180.0)
      IF (M .GT.1)  GO TO 20
      IF (M .LT. 0)  STOP
      AL = CONST
      GO TO 50
   20 AL = CONST*S**(M-1)

   50 PNM = AL
      DPNM = 0.
      J = 0
  100 J = J + 1
      JPM = J + M
      B = AL*((JPM-1)-FNN/JPM)
      DPNM = DPNM + B
      A = (B*X)/J
      PNM = PNM + A
      AL = A

C     STORE P OR DP SERIES FOR PRINTOUT.
      IF (IPRT .LE. 0)  GO TO 150
      IF (IPRT .EQ. 2)  GO TO 145
      AM(J) = A
      IF (IPRT .EQ. 1)  GO TO 150
  145 BM(J) = B

C     CHECK FOR TERMINATION OF SERIES.
  150 ABSA = ABS(A)
      IF (ABSA .LT. 1.E-15)  GO TO 110
      IF (ABSA .GT. 1.E+15)  GO TO 105
C     CHANGE CHECK LIMITS ACCORDING TO ACCURACY DESIRED AND ACCORDING
C     TO WORD SIZE OF COMPUTER.
C     FOR 32-BIT WORD, DOUBLE PRECISION, E-8 AND E+8 GIVE 7 DIGITS ACCURACY.
C     FOR 60-BIT WORD, DOUBLE PRECISION, E-15 AND E+15 GIVE 14 DIGITS ACCURACY.
```

```
C     FOR 60-BIT WORD, SINGLE PRECISION, E-8 AND E+7 GIVE 7 DIGITS ACCURACY.
C     (DOUBLE OR SINGLE PRECISION REFER TO FNN,AL,A,B,PNM,DPNM)
      IF (J .LT. JMAX)  GO TO 100
C     CONVERGENCE SLOW OR JMAX TOO SMALL
  105 CONTINUE
C     NUMERICAL ERROR UNACCEPTABLY LARGE DUE TO ADDING OF
C     LARGE AND SMALL NUMBERS.
  108 FORMAT (//12H ** ERROR **/1X,I5,F10.5,E15.7,I5,2D15.7)
      print*, 'Give higher desired accuracy in SUBROUTINE LEGFUN!'
      STOP

C     SERIES TRUNCATED SUCCESSFULLY.
  110 PS = PNM
      DPS = DPNM
      IF (M .NE. 0)  GO TO 115
      PMS = 0.
      P = PS
      DP = DPS*S/2.
      GO TO 120
  115 PMS = PS*M
      P = PS*S
      DP = DPS*S*S/2. + C*PMS
  120 CONTINUE

C     PRINT TERMS OF SERIES
      IF (IPRT .EQ. 0)  RETURN
  125 FORMAT (/1X,I5,F10.5,E20.12,F10.2,3F25.14,I5)
      IF (IPRT .LT. 0)  RETURN
      IF (IPRT .EQ. 2)  GO TO 135
  130 FORMAT (1X,16E8.1)
      IF (IPRT .EQ. 1)  RETURN
 135  continue
      RETURN
      END
!----------------------------------------------------------------
      SUBROUTINE FNM (KMAX,IBASIS,FN,CONST,COLAT)

C     READ PARAMETER FILE AND PUT DEGREE AND NORMALIZING CONSTANT
C     INTO ARRAYS FN AND CONST.
```

```
      SAVE
      DIMENSION    FN(0:KMAX,0:KMAX), CONST(0:KMAX,0:KMAX)
      CHARACTER*4  IZERO

      IT = 80
      OPEN (UNIT=IT,FILE='SCPARMS.DAT',STATUS='OLD')

      READ (IT,40,END=190)  KM,MM,THETA,IB

   40 FORMAT (2I3,F9.4,I5)
c      PRINT 50,  KM,MM,IB,THETA,KMAX,IBASIS
   50 FORMAT (/26H    KM   MM   IB     THETA,6X,4HKMAX,4X,6HIBASIS/
     *         1X,3I5,F10.2,2I10)
      IF (IBASIS .GT. IB)  GO TO 195
      IF (KMAX .GT. KM)  GO TO 195
      IF (MM .NE. KM)  GO TO 195
      COLAT = THETA

  100 READ  (IT,110,END=200)  K,M,FFN,CON
   write(*,*)K,M,FFN,CON
  110 FORMAT (2I3,F9.4,E15.6)
      IF (K .LT. 0)  GO TO 200
      IF (M .LT. 0)  GO TO 200
      IF (K .GT. KMAX)  GO TO 100
      IF (M .GT. K)  GO TO 100
      FN(K,M) = FFN
      CONST(K,M) = CON

      GO TO 100

  190 PRINT 191,  IT
  191 FORMAT (27H0*ERROR* NO FN DATA ON TAPE,I3)
  195 IFLAG = 1

      GO TO 350

  200 IFLAG = 0
      IVAR = 0
c      PRINT 205
```

```
  205 FORMAT (6H0    K, 4X,1HM, 8X,7HFN(K,M), 10X,10HCONST(K,M))
      DO 220 K=0,KMAX
      DO 220 M=0,K
      IF (IBASIS .EQ. 2)  GO TO 210
      KMM = K-M
      IF ((KMM/2)*2 .NE. KMM)  GO TO 220
  210 IZERO = ' '
      IF (K .NE. 0)  GO TO 212
      IF (M .NE. 0)  GO TO 212
      IF (FN(K,M) .EQ. 0.)  GO TO 214
      GO TO 213
  212 IF (FN(K,M) .NE. 0.)  GO TO 214
  213 IZERO = ' ***'
      IFLAG = 1
214   continue
c 214 PRINT 215,  K,M,FN(K,M),CONST(K,M),IZERO
  215 FORMAT (1X,2I5,F15.5,E20.7,1X,A4)
  220 CONTINUE

  350 CLOSE (UNIT=IT,STATUS='KEEP')
      IF (IFLAG .EQ. 1)  STOP
      RETURN
      END
c--------------------------------------------------------
      SUBROUTINE SUMPRD (IBO,NP1,N,X,SUM,S,IENTER)
C     CALCULATE SUMS (IF NECESSARY) AND SUMS OF SQUARES AND PRODUCTS.
      SAVE
      DIMENSION  X(NP1), SUM(NP1), S(NP1,NP1)
      IF (IENTER .GT. 0)  GO TO 20
      IENTER=3
      N=0
      DO 11 I=1,NP1
      IF (IBO .NE. 0)  SUM(I) = 0.
      DO 11 J=I,NP1
   11 S(I,J)=0.

   20 N=N+1
      DO 21 I=1,NP1
      IF (IBO .NE. 0)  SUM(I) = SUM(I) + X(I)
```

```
      DO 21 J=I,NP1
   21 S(I,J)=S(I,J)+X(I)*X(J)
      RETURN
      END
!-----------------------------------------------------------------------
      SUBROUTINE STEPREG (IBO,IBLNK,IORD,N,NP1,S,A,FIN,FOUT,ITMAX,BOUT,
     *                    SIG,IVAR,B,SB,ICBLNK,
     *                    IPRTS,IPRT1,IPRT2)

C     STEPWISE REGRESSION, GIVEN SUMS (IF NECESSARY) AND SUMS OF SQUARES
C     AND PRODUCTS.

C     REFERENCES - MATHEMATICAL METHODS FOR DIGITAL COMPUTERS, VOL. 1,
C                  EDITED BY A.RALSTON AND H.WILF.  CHAPTER 17 BY M.A. EFROYMSON
C                - APPLIED REGRESSION ANALYSIS BY N.R. DRAPER AND H. SMITH
C                  SECTIONS 6.4 AND 6.8.

C  INPUT PARAMETERS
C     IBO     = 0 FORCES REGRESSION THROUGH ORIGIN.
C     IBLNK   .LT. 0  ALLOWS PARAMETERS TO BE EXCLUDED FROM REGRESSION.
C     IORD    .NE. 0  PRE-ORDERS REGRESSION.
C     N       =  NUMBER OF OBSERVATIONS.
C     NP1     =  NUMBER OF INDEPENDENT VARIABLES (NP) + 1
C     S       =  MATRIX OF SUMS.  NOT NEEDED IF IBO=0.
C     A       =  MATRIX OF SUMS OF SQUARES AND PRODUCTS.
C                NOTE:  THIS MATRIX IS DESTROYED.
C     FIN     =  F LEVEL FOR ENTERING VARIABLE INTO REGRESSION.
C     FOUT    =  F LEVEL FOR TAKING VARIABLE OUT OF REGRESSION.
C                NOTE:  FIN MUST BE .GE. FOUT.
C     ITMAX   =  MAXIMUM NUMBER OF REGRESSION STEPS PERMITTED.
C     ICBLNK  =  BLANKING ARRAY,  INPUT IF IBLNK.LT.0.
C     IPRTS   =  0  TO PRINT NO MATRICES
C                1  TO PRINT ALL MATRICES
C               -1  TO PRINT ALL MATRICES EXCEPT COVARIANCE MATRICES.
C     IPRT1   =  FIRST STEP NUMBER TO HAVE COEFFICIENTS PRINTED.
C     IPRT2   =  LAST STEP NUMBER TO HAVE COEFFICIENTS PRINTED.
C                NOTE.  FINAL SET ARE PRINTED REGARDLESS OF IPRT1,IPRT2.
C  OUTPUT PARAMETERS
C     BOUT    =  OUTPUT MATRIX OF REGRESSION COEFFICIENTS.
```

```
C     SIG     =  MATRIX OF STANDARD DEVIATIONS.
C     IVAR    =  VARIABLE NUMBER FOR ELEMENTS OF B,SB.
C     B       =  MATRIX OF REGRESSION COEFFICIENTS, IN STEP-WISE ORDER.
C     SB      =  MATRIX OF STANDARD ERRORS OF COEFFICIENTS, IN STEP-WISE ORDER.
      SAVE
      REAL  MSE, MSR
      DIMENSION  S(NP1),BOUT(NP1)
      DIMENSION A(NP1,NP1),SIG(NP1),IVAR(NP1),B(NP1),SB(NP1),ICBLNK(NP1)

      NP = NP1 - 1
      ITN = 0
      IZERO = 0

C     PRINT PARAMETERS
c     PRINT 100,  IBO,IBLNK,IORD,N,NP1,FIN,FOUT,ITMAX,IPRTS,IPRT1,IPRT2
  100 FORMAT (20H STEPWISE REGRESSION//
     *        8X,3HIBO, 5X,5HIBLNK, 6X,4HIORD, 9X,1HN, 7X,3HNP1,
     *        7X,3HFIN, 6X,4HFOUT, 5X,5HITMAX, 5X,5HIPRTS,
     *        5X,5HIPRT1, 5X,5HIPRT2/ 1X,5I10,2F10.3,4I10)
      IF (FIN .LT. FOUT)   STOP
      IF (IPRTS .EQ. 0)  GO TO 101
C     PRINT INPUT MATRIX A
c      PRINT 115
c  115 FORMAT  (//62H0MATRIX OF SUMS OF SQUARES AND PRODUCTS, UNCORRECTED
c     * FOR MEANS)
c     DO 110 I=1,NP1
c 110 PRINT 116,  (A(I,J),I,J, J=I,NP1)
c 116 FORMAT (E17.8,3H A(,I2,1H,,I2,1H),E17.8,3H A(,I2,1H,,I2,1H),
c     *        E17.8,3H A(,I2,1H,,I2,1H),E17.8,3H A(,I2,1H,,I2,1H),
c     *        E17.8,3H A(,I2,1H,,I2,1H))
  101 CONTINUE
  131 FORMAT (43H0TOTAL SUM OF SQUARES, UNCORRECTED FOR MEAN, E29.12)
  132 FORMAT (41H0STANDARD DEVIATION, UNCORRECTED FOR MEAN, E31.12)

      IF (IBO .EQ. 0)  GO TO 117
      IF (IPRTS .EQ. 0)  GO TO 102
  112 FORMAT (15H0MATRIX OF SUMS )
  108 FORMAT (E17.8,3H S(,I2,1H),E20.8,3H S(,I2,1H),
     *   E20.8,3H S(,I2,1H),E20.8,3H S(,I2,1H),E20.8,3H S(,I2,1H))
```

```
  102 CONTINUE

C     COMPUTE MEAN OF EACH VARIABLE
      DO 114 J=1,NP1  !loop 1
  114 S(J) = S(J)/N   !end loop 1

      IF (IPRTS .EQ. 0)  GO TO 124

  118 FORMAT (6H0MEANS)
  124 CONTINUE
  133 FORMAT (27H0MEAN OF DEPENDENT VARIABLE, E45.12)

C     COMPUTE  RESIDUAL SUMS OF SQUARES AND CROSS PRODUCTS
      DO 113 I=1,NP1! loop 2
      DO 113 J=I,NP1
  113 A(I,J) = A(I,J) - N*S(I)*S(J) !end loop 2
      IF (IPRTS .EQ. 0)  GO TO 117
  137 FORMAT (  'OMATRIX OF SUMS OF SQUARES AND PRODUCTS, CORRECTED FOR
     *MEANS')

  117 CONTINUE
      SST = A(NP1,NP1)
      IF (IBO .EQ. 0)  GO TO 139
c      PRINT 134,  SST
  134 FORMAT (41H0TOTAL SUM OF SQUARES, CORRECTED FOR MEAN, E31.12)
c      PRINT 135,  SQRT(SST/(N-1))
  135 FORMAT (39H0STANDARD DEVIATION, CORRECTED FOR MEAN, E33.12)
  139 CONTINUE

C     COMPUTE STANDARD DEVIATION.  SET DIAGONALS OF CORRELATION MTRX=1.
C     NORMALIZE, THEN EXPAND UPPER TRIANGULAR MATRIX TO FULL
      DO 120 I=1,NP1 !loop 4
      SIG(I)=SQRT(A(I,I))
  120 A(I,I)=1.0    !end loop 4
      DO 121 I=1,NP  !loop 5
      II=I+1
      DO 121 J=II,NP1
      A(I,J)=A(I,J)/(SIG(I)*SIG(J))
  121 A(J,I)=A(I,J) ! end loop 5
```

```
      IF (IPRTS .EQ. 0)  GO TO 103
C     PRINT STANDARD DEVIATIONS AND CORRELATION MATRIX
c      PRINT 126
  126 FORMAT (20H0STANDARD DEVIATIONS)
      IF (IBO .NE. 0)  DF = SQRT(FLOAT(N-1))
      IF (IBO .EQ. 0)  DF = SQRT (FLOAT(N))
  122 FORMAT (19H0CORRELATION MATRIX )
      DO 123 I=1,NP !loop 6
      IP1 = I + 1 !end loop 6
  123 continue

  103 CONTINUE

C     SET UP A BLANKING ARRAY IF IBLNK .LT. 0
C     OTHERWISE, INCLUDE ALL VARIABLES IN REGRESSION.
      IF (IBLNK .LT. 0)  GO TO 109
      DO 107 I=1,NP !loop 7
  107 ICBLNK(I) = 1. !end loop 7
      GO TO 111
C     READ COEFFICIENT BLANKING ARRAY (ICBLNK).
C     IF ICBLNK(I) = 0,  DO NOT INCLUDE VARIABLE I IN THE REGRESSION.
  109 READ  104, (ICBLNK(I),I=1,NP)                          !INPUT
  104 FORMAT (80I1)
      NPL = NP
      IF (NP .GT. 50)  NPL = 50
c      PRINT 105,  (I,I=1,NPL)
  105 FORMAT (27H0COEFFICIENT BLANKING ARRAY/ 1X,50I2)
c      PRINT 106, (ICBLNK(I),I=1,NP)
  106 FORMAT (1X,50I2)
  111 CONTINUE

c      PRINT 401
  401 FORMAT (///6H ANOVA,8X,2HDF,24X,2HSS,22X,2HMS,13X,1HF,8X,6HTOTVAR,
     *        12X,2HSY,6X,4HNVAR,9X,6HFLEVEL)

C     COMPUTE DEGREES OF FREEDOM
      PHI = N - 1
      IF (IBO .EQ. 0)  PHI = N
```

```
      DFT = PHI

C     VARIABLES ARE ORDERED (1,2,3,...) IF IORD .NE. 0
      INOR = 0
      NORD = NP
      IF (IORD .NE. 0)  NORD = 1
      GO TO 129
  130 NORD = NORD + 1
      IF (NORD .GT. NP)  GO TO 175
      INOR = 0
      NOR = NORD
      GO TO 128

C     INITIALIZATION PROCEDURE FOR DETERMINING MOST SIGNIFICANT VARIABLE
C     TO BE ADDED  TO REGRESSION
C     COMPUTE STANDARD ERROR OF DEPENDENT VARIABLE
  125 INOR = 1
  129 NOR = 1
      SSE = A(NP1,NP1)*SST
      MSE = SSE/PHI
      SY = SQRT(MSE)
      SSR = SST - SSE
      DFR = DFT - PHI
      IF (DFR .LE. 0.)  GO TO 127
      MSR = SSR/DFR
      F = MSR/MSE
  127 CONTINUE
      VMIN=-10.E37
      NMIN=0
      NOIN=0
  128 VMAX=0.0
      NMAX=0
      DO 150 I=NOR,NORD !loop 8

      IF (A(I,I) .LE. .00001)  GO TO 150

C     COMPUTE VARIANCE
      V=A(I,NP1)*A(NP1,I)/A(I,I)
```

```
      IF (V) 142,150,143

  143 IF (V .LE. VMAX)  GO TO 150
      IF (ICBLNK(I) .EQ. 0)  GO TO 150
      VMAX=V
      NMAX=I
      GO TO 150

C     X(I) IS IN REGRESSION...COMPUTE COEFFICIENT B AND STAND. ERROR OF
C     COEFF.
  142 NOIN=NOIN+1
      IVAR(NOIN)=I
      B(NOIN)=A(I,NP1)*SIG(NP1)/SIG(I)

      SB(NOIN)=SY*SQRT(A(I,I))/SIG(I)
      IF (V .LE. VMIN)  GO TO 150
      VMIN=V
      NMIN=I
  150 CONTINUE    !end loop 8

      IF (INOR .EQ. 0)  GO TO 153
      IF (IBO .EQ. 0)  GO TO 160
C     COMPUTE CONSTANT
      TEMP=0.0
      IF (NOIN .LE. 0)  GO TO 156
      DO 151 I=1,NOIN  !loop 9
      II=IVAR(I)
  151 TEMP=TEMP+B(I )*S(II)   !end loop 9
  156 CONTINUE
      CONST = S(NP1) - TEMP
  160 CONTINUE

C     PRINT RESULTS FROM STEP NUMBER ITN.
c      PRINT 405, ITN,NVAR,FLEVEL
  405 FORMAT (12HOSTEP NUMBER,I4,92X,I10,F15.5)
      VTNTOT = (1.-A(NP1,NP1))*100.
c      PRINT 416, DFR,SSR,MSR,F,VTNTOT,SY, PHI,SSE,MSE
  416 FORMAT (11H REGRESSION,F6.0,E25.12,E24.12,3F14.5/
     *        9H RESIDUAL,F8.0,E25.12,E24.12)
```

```
      IF (ITN .LT. IPRT1)  GO TO 420
      IF (ITN .GT. IPRT2)  GO TO 420
  407 continue     !end of loop 9
c  407 PRINT 400
  400 FORMAT (9H VARIABLE,22X,11HCOEFFICIENT,10X,14HSTANDARD ERROR,
     *        7X,7HT VALUE,6X,7HF VALUE)
      IF (IBO .NE. 0)  PRINT 415, IZERO,CONST
  415 FORMAT (1H ,I8,E33.12,E24.12,2F14.5)
      DO 410 I=1,NOIN !loop 10
      T = B(I)/SB(I)
      TSQ = T**2
 410  continue !end loop 10
c  410 PRINT 415, IVAR(I),B(I),SB(I),T,TSQ
      IF (ITN .LT. 0)  RETURN
      IF (IPRTS .LE. 0)  GO TO 181
C     PRINT VARIANCE-COVARIANCE MATRIX
c      PRINT 179
  179 FORMAT (18H COVARIANCE MATRIX)
c      IF (IBO .NE. 0)  PRINT 178, MSE/N
  178 FORMAT (1H+,26X,E18.9,9H V(YMEAN))
      DO 180 II=1,NOIN !loop 11
      I = IVAR(II)
 180  continue    !end loop 11

  181 CONTINUE
  420 CONTINUE

      IF (ITN .GE. ITMAX)  GO TO 170

C     COMPARE F LEVELS...
  417 FLEVEL=VMIN*PHI/A(NP1,NP1)
      IF (FOUT+FLEVEL) 153,153,152
  152 K=NMIN
      PHI=PHI+1.0
      NVAR=-K
      GO TO 200
  153 FLEVEL=VMAX*(PHI-1.)/(A(NP1,NP1)-VMAX)
      IF (FIN-FLEVEL) 154,130,130
  154 K=NMAX
```

```
      PHI=PHI-1.0
      NVAR=K

C     CALCULATE NEW MATRIX
  200 ITN = ITN + 1
      DO 210 I=1,NP1 !loop 12
      IF (I .EQ. K)  GO TO 210
      DO 240 J=1,NP1
      IF (J .EQ. K)  GO TO 240
      A(I,J)=A(I,J) - A(I,K)*A(K,J)/A(K,K)
  240 CONTINUE
  210 CONTINUE !end loop 12
      DO 280 I=1,NP1
      IF (I .EQ. K)  GO TO 280
      A(I,K)=-A(I,K)/A(K,K)
  280 CONTINUE
      DO 320 J=1,NP1 !loop 13
      IF (J .EQ. K)  GO TO 320
      A(K,J)=A(K,J)/A(K,K)
  320 CONTINUE   !end loop 13
      A(K,K)=1.0/A(K,K)
      GO TO 125

  170 PRINT 172, ITN
  172 FORMAT (46HOMAXIMUM NUMBER OF ITERATIONS REACHED.   ITN =,I5)

  175 DO 176 I=1,NP !loop 14
  176 BOUT(I) = 0. !end loop 14
      DO 177 I=1,NOIN !loop 15
      II = IVAR(I)
  177 BOUT(II) = B(I) !end loop 15
       IF (IBO .NE. 0)  BOUT(NP1) = CONST

C     PRINT COEFFICIENTS FROM FINAL STEP IF NOT ALREADY PRINTED.
      IF (ITN .LT. IPRT1)  GO TO 350
      IF (ITN .GT. IPRT2)  GO TO 350
      RETURN
  350 ITN = -1
      GO TO 407
```

```
      END

c-----------------------------------------------------------------------
      SUBROUTINE SPHNEWF (FLAT1,FLON1,X1,Y1,FLAT2,FLON2,X2,Y2)
        SAVE
C     SUBROUTINE FOR TRANSFORMING GEOGRAPHIC COORDINATES (FLAT1,FLON1)
C     TO SPHERICAL COORDINATES (FLAT2,FLON2) ABOUT A NEW POLE (FLAT0,FLON0).
C     FIELD COMPONENTS GEOGRAPHIC NORTH X1 AND EAST Y1 ARE ROTATED TO
C     NORTH X2 AND EAST Y2 IN THE NEW COORDINATE SYSTEM.
C     FLON2 = 0. IS THE MERIDIAN THROUGH THE SOUTH GEOGRAPHIC POLE.

      DATA   FLAT0/90./

      REAL*8 DFLAT0,DFLAT1,DFLAT2,FLOND1,
     *       SIN0,COS0,SIN1,COS1,SIND1,COSD1,SIN2,COS2,SLON2,PROD

      IF (FLAT0 .LT. 90.)  GO TO 100
      FLAT2 = FLAT1
      FLON2 = FLON1
      GO TO 105
  100 IF (FLAT0 .GT. -90.)  GO TO 101
      FLAT2 = -FLAT1
      FLON2 = -FLON1
      GO TO 107

  101 IF (FLAT1 .LT. 90.)  GO TO 102
  131 FLAT2 = FLAT0
      FLON2 = 180.
      ANG = FLON0-FLON1+180.
      GO TO 103
  102 IF (FLAT1 .GT. -90.)  GO TO 104
  132 FLAT2 = -FLAT0
      FLON2 = 0.
      ANG = FLON1-FLON0
  103 COSROT = COS(ANG*3.1415927/180.0)
      SINROT = SIN(ANG*3.1415927/180.0)
      GO TO 225
  104 DFLAT1 = FLAT1
      SIN1 = sin(DFLAT1*3.1415927/180.0)
```

```
    COS1 = cos(DFLAT1*3.1415927/180.0)
    IF (COS1 .NE. 0. ) GO TO 133
    IF (SIN1 .GT. 0. ) GO TO 131
    GO TO 132

133 FLOND1 = FLON1 - FLON0
    COSD1 = COS(FLOND1*3.1415927/180.0)
    SIN2 = SIN0*SIN1 + COS0*COS1*COSD1
    IF (SIN2 .LT. 1.)  GO TO 106
108 FLAT2 = 90.
    FLON2 = 0.
105 X2 = X1
    Y2 = Y1
    COSROT = 1.
    SINROT = 0.
    RETURN
106 IF (SIN2 .GT. -1.)  GO TO 110
109 FLAT2 = -90.
    FLON2 = 0.
107 X2 = -X1
    Y2 = -Y1
    COSROT = -1.
    SINROT =  0.
    RETURN


110 DFLAT2 = ASIN(SIN2)*180.0/3.1415927
    FLAT2 = DFLAT2
    COS2 = COS(DFLAT2*3.1415927/180.0)
    IF ( COS2 .NE. 0. ) GO TO 111
    IF ( SIN2 .GT. 0. ) GO TO 108
    GO TO 109
111 PROD = SIN0*SIN2
    SIND1 = SIN(FLOND1*3.1415927/180.0)
    IF (SIN1 .NE. PROD)  GO TO 120
    IF (SIND1 .GT. 0.)  GO TO 115
112 FLON2 = -90.
    SLON2 = -1.
    GO TO 200
```

```
  115 FLON2 =  90.
      SLON2 = 1.
      GO TO 200
  120 SLON2 = (SIND1*COS1)/COS2
      IF ( SLON2 .LE. -1. ) GO TO 112
      IF ( SLON2 .GE. 1. ) GO TO 115
      IF (SIN1 .LT. PROD)  GO TO 125
      FLON2 = 180. - ASIN(SLON2)*180.0/3.1415927
      IF (FLON2 .GT. 180.)  FLON2 = FLON2 - 360.
      GO TO 200
  125 FLON2 = ASIN(SLON2)*180.0/3.1415927

C     Error detection (would give division by zero)
  200 IF (COS1.NE. 0. .AND. COS2.NE.0.)  GO TO 220
c      PRINT 205, FLAT1,FLON1,SIN1,COS1,SIN2,COS2,PROD
  205 FORMAT (/19H0SUBROUTINE SPHNEWF/12X,5HFLAT1,11X,5HFLON1,
     * 12X,4HSIN1,12X,4HCOS1,12X,4HSIN2,12X,4HCOS2,12X,4HPROD/1X,7E16.8)
      STOP
  220 COSROT = (SIN0-SIN1*SIN2)/COS1/COS2
      SINROT = COS0*SLON2/COS1
  225 X2 = X1*COSROT - Y1*SINROT
      Y2 = X1*SINROT + Y1*COSROT
      RETURN
c----------------------------------------------------------------------
C     TRANSFORM BACK FROM ROTATED COMPONENTS X2,Y2
C                   TO GEOGRAPHIC COMPONENTS X1,Y1.
      ENTRY SPHOLDF (FLAT1,FLON1,X1,Y1,FLAT2,FLON2,X2,Y2)
      X1 = X2*COSROT + Y2*SINROT
      Y1 = Y2*COSROT - X2*SINROT
      RETURN
c-----------------------------------------------------------------------
C     INPUT NEW POLE POSITION FLAT0,FLON0 IN GEOGRAPHIC COORDINATES.
C     RETURN FLAT0,FLON0 AS FLAT1,FLON1.
      ENTRY SPHINF(FLAT1,FLON1,X1,Y1,FLAT2,FLON2,X2,Y2)
c      print*, 'Please give FLAT0, FLON0 (midpoint of spherical cap'
c      print*, '  in geographical coordinates):'
c      READ(*,*)   FLAT0,FLON0
      FLAT0 = -27.25
      FLON0 = 24.75
```

```
   10 FORMAT (2F10.0)
      FLAT1 = FLAT0
      FLON1 = FLON0
      GO TO 20
c------------------------------------------------------------------------
C     DEFINE NEW POLE POSITION THROUGH CALL PARAMETERS FLAT1,FLON1.
      ENTRY SPHITF(FLAT1,FLON1,X1,Y1,FLAT2,FLON2,X2,Y2)
      FLAT0 = FLAT1
      FLON0 = FLON1
  20  continue
c   20 PRINT 30, FLAT0,FLON0
   30 FORMAT (1H0, 5X,5HFLAT0, 5X,5HFLON0 / 1X,2F10.3)
      DFLAT0 = FLAT0
      SIN0 = SIN(FLAT0*3.1415927/180.0)
      COS0 = COS(FLAT0*3.1415927/180.0)
      RETURN
      END
c----------------------------------------------------------------------------
      SUBROUTINE GEOCENF (GDLAT,GDALT,X,Z,GCLAT,R,BN,BV)
C     CONVERT FROM GEODETIC LATITUDE GDLAT, GEODETIC ALTITUDE GDALT
C             TO GEOCENTRIC LATITUDE GCLAT, GEOCENTRIC RADIAL DISTANCE R;
C     AND FROM GEODETIC NORTH AND VERTICAL DOWNWARD COMPONENTS X,Z
C         TO GEOCENTRIC NORTH AND VERTICAL DOWNWARD COMPONENTS BN,BV.
      SAVE
C     International ELLIPSOID
      DATA  A/6378.388/, ESQ/0.006722670022333/
      SLAT = SIN(GDLAT*3.1415927/180.0)
      CLAT = COS(GDLAT*3.1415927/180.0)
      FN = A/SQRT(1.-ESQ*SLAT**2)
      FNPH = FN + GDALT
      WC = FNPH*CLAT
      ZC = (FNPH-ESQ*FN)*SLAT
      R = SQRT(WC**2+ZC**2)
      IF (WC .LT. ABS(ZC)) THEN
          GCLAT = ACOS(WC/R)*180.0/3.1415927
          IF (ZC .LT .0.) GCLAT=-GCLAT
      ELSE
          GCLAT = ASIN(ZC/R)*180.0/3.1415927
          END IF
```

```
      D = GDLAT - GCLAT
      SD = SIN(D*3.1415927/180.0)
      CD = COS(D*3.1415927/180.0)
      BN = X*CD - Z*SD
      BV = X*SD + Z*CD
      RETURN
c-----------------------------------------------------------------------
C     CONVERT BACK FROM GEOCENTRIC BN,BV TO GEODETIC X,Z.
C     THIS CALL MUST FOLLOW A CALL TO GEOCENF.
      ENTRY GEOCINV (GDLAT,GDALT,X,Z,GCLAT,R,BN,BV)
      X = BN*CD + BV*SD
      Z = BV*CD - BN*SD
      RETURN
      END
c-------------------------------------------------------------------------
      SUBROUTINE PRTRES (ITAPE,IBO,NTOT,NP1,X,B)

C     PRINT RESIDUALS OF DATA USED IN REGRESSION.

      SAVE

      DIMENSION  X(NP1), B(NP1)

      REWIND ITAPE
      NP = NP1 - 1
      BO = 0.
      N = 0
      SUM = 0.
      SUMSQ = 0.
      PRINT 5
    5 FORMAT (22H1PRINTOUT OF RESIDUALS/18H0COEFFICIENTS USED/)
      NC = 0
      IF (IBO .EQ. 0)  GO TO 7
      BO = B(NP1)
c      PRINT 6,  NC,BO
    6 FORMAT (3H B(,I3,3H) =,1X,E18.10)
      NC = 1
    7 DO 8 I=1,NP
      IF (B(I) .EQ. 0.)  GO TO 8
```

```
c       PRINT 6,  I,B(I)
      NC = NC + 1
    8 CONTINUE
c       PRINT 9
    9 FORMAT (//3X,3HNUM,2X,5HIDENT,25X,8HOBSERVED, 15X,
     *    10HCALCULATED,15X,10HOBS - CALC /)

   10 READ(ITAPE,END=14)  NUM,IDENT,(X(I),I=1,NP1)
      YREG = BO
      DO 12 I=1,NP
   12 YREG=YREG+B(I)*X(I)
      YOBS=X(NP1)
      RESID=YOBS-YREG
c     PRINT 13,    NUM,IDENT,YOBS,YREG,RESID
   13 FORMAT (1X,I5,2X,A10,3X,3F25.1)
      N = N+1
      SUM = SUM + RESID
      SUMSQ = SUMSQ + RESID*RESID
      GO TO 10

   14 PRINT 16, IBO,NP,NC,NTOT,N,SUM,SUMSQ
   16 FORMAT (//8X,3HIBO, 8X,2HNP, 8X,2HNC, 6X,4HNTOT, 9X,1HN,
     *        17X,3HSUM, 15X,5HSUMSQ/ 1X,5I10,2E20.12)
      IF (NTOT .LE. 0)  RETURN
      IF (N .LE. 0)  RETURN
      FN = N
      SMEAN = SUM/FN
      PRINT 18, FN,SMEAN
   18 FORMAT (/9X,2HFN, 11X,4HMEAN/ 1X,F10.3,F15.5)
      DFR = NC
      DF = FN - (FN/NTOT)*DFR
      IF (DF .LE. 0.)  RETURN
      RMS = SQRT(SUMSQ/DF)
      STDEV = SQRT((SUMSQ-SUM*SUM/FN)/DF)
      PRINT 20, DF,RMS,STDEV
   20 FORMAT (/9X,2HDF, 12X,3HRMS, 10X,5HSTDEV/ 1X,F10.3,2F15.5)
      RETURN
      END
c----------------------------------------------------------------------
```

```
      SUBROUTINE SCHNEV (FLAT,FLON,R,T,L,BN,BE,BV)

C     WHEN L IS POSITIVE:
C     COMPUTES SPHERICAL CAP HARMONIC (GEOCENTRIC) FIELD COMPONENTS
C     HORIZONTAL NORTH BN,HORIZONTAL EAST BE,AND VERTICAL DOWNWARD BV.
C     WHEN L IS NEGATIVE:
C     COMPUTES GENERAL FUNCTION BV, ITS HORIZONTAL NORTH DERIVATIVE BN,
C     AND ITS HORIZONTAL EAST DERIVATIVE BE, ON SPHERICAL CAP SURFACE.
C     NOTE THAT THESE ARE METRICAL DERIVATIVES, AND BE IS THE
C     LONGITUDINAL DERIVATIVE DIVIDED BY SIN(COLATITUDE).

C     FLAT,FLON,R ARE GEOCENTRIC SPHERICAL CAP LATITUDE,LONGITUDE,RADIAL
C     DISTANCE; T IS TIME.

C     L =  0  ON FIRST CALL:  RETURNS SPHERICAL CAP POLE POSITION FLATO,FLONO
C             AND HALF-ANGLE THETA AS BN,BE, AND BV AFTER INITIALIZATION.
C             ON SUBSEQUENT CALLS:  ACTS AS L=1.
C          1  COMPUTES POTENTIAL FIELD COMPONENTS FROM INTERNAL COEFFICIENTS.
C          2  COMPUTES POTENTIAL FIELD COMPONENTS FROM EXTERNAL COEFFICIENTS.
C          3  COMPUTES FIELD FROM BOTH INTERNAL AND EXTERNAL COEFFICIENTS.
C         -1  COMPUTES GENERAL FUNCTION BV AND DERIVATIVES BN WITH RESPECT TO
C             LATITUDE AND BE WITH RESPECT TO LONGITUDE DIVIDED BY COS(LAT)
C             (R IS DUMMY VARIABLE IN THIS CASE).
C     NOTE:   SUBROUTINE IS INITIALIZED DURING FIRST CALL REGARDLESS OF L.

C     SUBPROGRAM USED:  LEGFUN
      SAVE
      PARAMETER   (KDIM=30,LDIM=3)                                      SET UP
      DIMENSION   FN(0:KDIM,0:KDIM), CONST(0:KDIM,0:KDIM)
      DIMENSION   CML(KDIM), SML(KDIM)
      DIMENSION   GNM(0:LDIM), HNM(0:LDIM)
      DIMENSION   DELT(0:LDIM)
      DIMENSION   BINT(0:KDIM,0:KDIM,0:LDIM), BEXT(0:KDIM,0:KDIM,0:LDIM)
      CHARACTER*1 IE
      DATA        IENTER /0/

      IF (IENTER .NE. 0)  GO TO 100

      IENTER = 1
```

```
C     READ COEFFICIENT FILE
      IT1 = 85
      OPEN (UNIT=IT1,FILE='schcoef.dat',STATUS='OLD')
      READ (IT1,240,END=999)  FLATO,FLONO,THETA,RE,TZERO,
     *             IFIT,ICEN,IREF,IB,KINT,LINT,KEXT,LEXT
  240 FORMAT (F8.2,F9.2,F7.2,2F8.1,8I5)
      PRINT 236,  FLATO,FLONO,THETA,RE,TZERO
  236 FORMAT (//7X,5HFLATO,5X,5HFLONO,5X,5HTHETA,8X,2HRE,5X,5HTZERO/
     *        2X,3F10.2,2F10.1)
      PRINT 238,  IFIT,ICEN,IREF,IB,KINT,LINT,KEXT,LEXT
  238 FORMAT (/4X,48H  IFIT  ICEN  IREF    IB  KINT  LINT  KEXT  LEXT/
     *         4X,8I6)
      KMAX = MAX(KINT,KEXT)
      IF (KMAX .GT. KDIM)  GO TO 999
      KT = MAX(LINT,LEXT)
      IF (KT .GT. LDIM)  GO TO 999
C      PRINT 242
C  242 FORMAT (/4X,1HK,3H  M,6X,3HFNN,10X,5HCONST,
C     *        <MIN(KT+1,5)>(7X,3HGNM,7X,3HHNM))
  247 READ (IT1,250,END=280) IE,N,M,FNN,CON,(GNM(J),HNM(J),J=0,KT)
  250 FORMAT (1X,A1,2I3,F9.4,E15.6,(F15.3,F15.3))
      IF (N .LT. 0)  GO TO 280
      IF (M .LT. 0)  GO TO 280
      IF (N .GT. KMAX)  GO TO 247
      IF (M .GT. KMAX)  GO TO 247
      IF (IE .EQ. 'I')  THEN
         IF (N .GT. KINT)  GO TO 247
         LJ = LINT
      ELSE
         IF (N .GT. KEXT)  GO TO 247
         LJ = LEXT
         END IF
      FN(N,M) = FNN
      CONST(N,M) = CON
      IF (M .GT. 0)  GO TO 300
      PRINT 255, IE,N,M, FNN,CON,(GNM(J),J=0,LJ)
  255 FORMAT (1X,A1,2I3,F9.4,E15.6,F10.3,4F20.3:/(22X,5F20.3))
      DO 301 J=0,LJ
      IF (IE .EQ. 'I')  THEN
```

```
      BINT(N,M,J)   = GNM(J)
   ELSE
      BEXT(N,M,J)   = GNM(J)
      END IF
301 CONTINUE
   GO TO 247
300 PRINT 260, IE,N,M,FNN,CON,(GNM(J),HNM(J),J=0,LJ)
260 FORMAT (1X,A1,2I3,F9.4,E15.6,10F10.3:/(32X,10F10.3))
   DO 302 J=0,LJ
   IF (IE .EQ. 'I')  THEN
      BINT(N,M,J)   = GNM(J)
      BINT(M-1,N,J) = HNM(J)
   ELSE
      BEXT(N,M,J)   = GNM(J)
      BEXT(M-1,N,J) = HNM(J)
      END IF
302 CONTINUE
   GO TO 247
280 CLOSE (UNIT=IT1,STATUS='KEEP')

   IF (L .NE. 0)  GO TO 100
   BN = FLATO
   BE = FLONO
   BV = THETA
   RETURN

100 IF (L .GE. 0)  THEN
      IF (IFIT .LT. 0)  GO TO 999
      AOR = RE/R
      AR = AOR**2
      IF (L .GT. 1)  GO TO 107
   ELSE
      IF (IFIT .GE. 0)  GO TO 999
      AR = -1.
      END IF
   KT = LINT
   GO TO 109
107 IF (KEXT .GT. 0)  AOR3 = AOR*AR
   IF (L .GT. 2)  GO TO 108
```

```
      KT = LEXT
      GO TO 109
  108 KT = MAX (LINT,LEXT)
  109 DELT(0) = 1.
      IF (KT .LE. 0)  GO TO 103
      DEL = T - TZERO
      DO 102 I=1,KT
  102 DELT(I) = DELT(I-1)*DEL
  103 X = 0.
      Y = 0.
      Z = 0.
      IF (L .EQ. 2)  GO TO 106
      IF (KINT .LT. 0)  GO TO 106
      GTI = 0.
      DO 105 I=0,LINT
  105 GTI = GTI + BINT(0,0,I)*DELT(I)
      Z = -AR*GTI
      N =  0
  106 COLAT = 90. - FLAT
      DO 150 N=1,KMAX
      IF (N .GT. 1)  GO TO 115
      CL = COS(FLON*3.1415927/180.0)
      SL = SIN(FLON*3.1415927/180.0)
      CML(1) = CL
      SML(1) = SL
      GO TO 120
  115 SML(N) = SL*CML(N-1) + CL*SML(N-1)
      CML(N) = CL*CML(N-1) - SL*SML(N-1)
  120 CONTINUE
      DO 150 M=0,N
      IF (IB .EQ. 2)  GO TO 121
      NMM = N - M
      IF ((NMM/2)*2 .NE. NMM)  GO TO 150
  121 FFN = FN(N,M)
      CALL LEGFUN (M,FFN,CONST(N,M),COLAT,P,DP,PMS,0)
      IF (L .GE. 0)  THEN
         AR = AOR**(FFN+2.)
      ELSE
         AR = 1.
```

```
      FFN = -2.
      DP = -DP
      PMS = -PMS
      END IF
    IF (M .NE. 0)  GO TO 130
    BT1 = 0.
    BT3 = 0.
    BT  = 0.
    IF (L .EQ. 2)  GO TO 123
    IF (N .GT. KINT)  GO TO 123
    GTI = 0.
    DO 122 I=0,LINT
122 GTI  = GTI  + BINT(N,M,I)*DELT(I)
    BT1  = AR*GTI
    BT3  = BT1
123 IF (L .LE. 1)  GO TO 125
    IF (N .GT. KEXT)  GO TO 125
    GTE = 0.
    DO 124 I=0,LEXT
124 GTE = GTE + BEXT(N,M,I)*DELT(I)
    BT  = AOR3/AR*GTE
    BT1 = BT1 + BT
125 X = X + BT1*DP
    Z = Z - (FFN*(BT3-BT)+BT3)*P
    GO TO 150
130 BT1 = 0.
    BT2 = 0.
    BT3 = 0.
    BT  = 0.
    IF (L .EQ. 2)  GO TO 133
    IF (N .GT. KINT)  GO TO 133
    GTI = 0.
    HTI = 0.
    DO 132 I=0,LINT
    GTI = GTI + BINT(N,M,I)*DELT(I)
132 HTI = HTI + BINT(M-1,N,I)*DELT(I)
    BT1 = AR*(GTI*CML(M) + HTI*SML(M))
    BT2 = AR*(GTI*SML(M) - HTI*CML(M))
    BT3 = BT1
```

```
  133 IF (L .LE. 1)  GO TO 135
      IF (N .GT. KEXT)  GO TO 135
      GTE = 0.
      HTE = 0.
      DO 134 I=0,LEXT
      GTE = GTE + BEXT(N,M,I)*DELT(I)
  134 HTE = HTE + BEXT(M-1,N,I)*DELT(I)
      RA = AOR3/AR
      BT = RA*(GTE*CML(M) + HTE*SML(M))
      BT1 = BT1 + BT
      BT2 = BT2 + RA*(GTE*SML(M) - HTE*CML(M))
  135 X = X + BT1*DP
      Y = Y + BT2*PMS
      Z = Z - (FFN*(BT3-BT)+BT3)*P
  150 CONTINUE
      BN = X
      BE = Y
      BV = Z
      RETURN
  999 STOP
      END
c-----------------------------------------------------------------------
      FUNCTION  PDPFUN (FN)

C     DRIVER PROGRAM FOR SUBROUTINE RTMI.  COMPUTES LEGENDRE FUNCTION P
C     OR ITS DERIVATIVE DP AND SCHMIDT NORMALIZING CONSTANT CONST
C     AT SPHERICAL CAP BOUNDARY COLATITUDE = COLAT.

C     SUBPROGRAM REQUIRED:  LEGFUN
      SAVE
      COMMON    COLAT, M, IPDP, CONST
      DATA      IENTER/-1/

      IF (M .GT. 0)  GO TO 10
      CONST = 1.
      GO TO 30

C     COMPUTE NORMALIZING CONSTANT USING STIRLINGS APPROXIMATION,
C     VALID FOR   FN .GT. FM .GT. 0.
```

```
   10 IF (IENTER .EQ. 0)  GO TO 20
      IENTER = 0
      FLN2 = ALOG(2.)
      FRPI = ALOG(ATAN2(0.,-1.))/2.
   20 FM = M
      R = FM/FN
      R1 = 1./R/R - 1.
      R2 = R1*R1
      R3 = R1*R2
      CONST = EXP((FN/2.+.25)*ALOG((1.+R)/(1.-R))
     *      + (FM/2.)*ALOG(R1) - ALOG(FM)/2. - FM*FLN2
     *      - FRPI - (1.+1./R1)/12./FM + (1.+3./R2+4./R3)/360./FM**3)

   30 CALL LEGFUN (M,FN,CONST,COLAT,P,DP,PMS,0)
      PDPFUN = DP
      IF (IPDP .EQ. 1)  PDPFUN = P

      RETURN
      END
c-----------------------------------------------------------------------
      SUBROUTINE RTMI (X,F,FCT,XLI,XRI,EPS,IEND,IER)

C     ..................................................................
C
C        PURPOSE
C           TO SOLVE GENERAL NONLINEAR EQUATIONS OF THE FORM FCT(X)=0
C           BY MEANS OF MUELLER-S ITERATION METHOD.
C
C        USAGE
C           CALL RTMI (X,F,FCT,XLI,XRI,EPS,IEND,IER)
C           PARAMETER FCT REQUIRES AN EXTERNAL STATEMENT.
C
C        DESCRIPTION OF PARAMETERS
C           X      - RESULTANT ROOT OF EQUATION FCT(X)=0.
C           F      - RESULTANT FUNCTION VALUE AT ROOT X.
C           FCT    - NAME OF THE EXTERNAL FUNCTION SUBPROGRAM USED.
C           XLI    - INPUT VALUE WHICH SPECIFIES THE INITIAL LEFT BOUND
C                    OF THE ROOT X.
C           XRI    - INPUT VALUE WHICH SPECIFIES THE INITIAL RIGHT BOUND
```

```
C                  OF THE ROOT X.
C           EPS    - INPUT VALUE WHICH SPECIFIES THE UPPER BOUND OF THE
C                  ERROR OF RESULT X.
C           IEND   - MAXIMUM NUMBER OF ITERATION STEPS SPECIFIED.
C           IER    - RESULTANT ERROR PARAMETER CODED AS FOLLOWS
C                   IER=0 - NO ERROR,
C                   IER=1 - NO CONVERGENCE AFTER IEND ITERATION STEPS
C                           FOLLOWED BY IEND SUCCESSIVE STEPS OF
C                           BISECTION,
C                   IER=2 - BASIC ASSUMPTION FCT(XLI)*FCT(XRI) LESS
C                           THAN OR EQUAL TO ZERO IS NOT SATISFIED.
C
C        REMARKS
C           THE PROCEDURE ASSUMES THAT FUNCTION VALUES AT INITIAL
C           BOUNDS XLI AND XRI HAVE NOT THE SAME SIGN. IF THIS BASIC
C           ASSUMPTION IS NOT SATISFIED BY INPUT VALUES XLI AND XRI, THE
C           PROCEDURE IS BYPASSED AND GIVES THE ERROR MESSAGE IER=2.
C
C        SUBROUTINES AND FUNCTION SUBPROGRAMS REQUIRED
C           THE EXTERNAL FUNCTION SUBPROGRAM FCT(X) MUST BE FURNISHED
C           BY THE USER.
C
C        METHOD
C           SOLUTION OF EQUATION FCT(X)=0 IS DONE BY MEANS OF MUELLER-S
C           ITERATION METHOD OF SUCCESSIVE BISECTIONS AND INVERSE
C           PARABOLIC INTERPOLATION, WHICH STARTS AT THE INITIAL BOUNDS
C           XLI AND XRI. CONVERGENCE IS QUADRATIC IF THE DERIVATIVE OF
C           FCT(X) AT ROOT X IS NOT EQUAL TO ZERO. ONE ITERATION STEP
C           REQUIRES TWO EVALUATIONS OF FCT(X). FOR TEST ON SATISFACTORY
C           ACCURACY SEE FORMULAE (3,4) OF MATHEMATICAL DESCRIPTION.
C           FOR REFERENCE, SEE G. K. KRISTIANSEN, ZERO OF ARBITRARY
C           FUNCTION, BIT, VOL. 3 (1963), PP.205-206.
C
C     ..................................................................

      EXTERNAL FCT
      SAVE

C     PREPARE ITERATION
```

```
      IER=0
      XL=XLI
      XR=XRI
      X=XL
      TOL=X
      F=FCT(TOL)
      IF(F)1,16,1
    1 FL=F
      X=XR
      TOL=X
      F=FCT(TOL)
      IF(F)2,16,2
    2 FR=F
      IF(SIGN(1.,FL)+SIGN(1.,FR))25,3,25
C
C     BASIC ASSUMPTION FL*FR LESS THAN 0 IS SATISFIED.
C     GENERATE TOLERANCE FOR FUNCTION VALUES.
    3 I=0
      TOLF=100.*EPS

C     START ITERATION LOOP
    4 I=I+1
C
C     START BISECTION LOOP
      DO 13 K=1,IEND
      X=.5*(XL+XR)
      TOL=X
      F=FCT(TOL)
      IF(F)5,16,5
    5 IF(SIGN(1.,F)+SIGN(1.,FR))7,6,7
C
C     INTERCHANGE XL AND XR IN ORDER TO GET THE SAME SIGN IN F AND FR
    6 TOL=XL
      XL=XR
      XR=TOL
      TOL=FL
      FL=FR
      FR=TOL
    7 TOL=F-FL
```

```
      A=F*TOL
      A=A+A
      IF(A-FR*(FR-FL))8,9,9
    8 IF(I-IEND)17,17,9
    9 XR=X
      FR=F
C
C     TEST ON SATISFACTORY ACCURACY IN BISECTION LOOP
      TOL=EPS
      A=ABS(XR)
      IF(A-1.)11,11,10
   10 TOL=TOL*A
   11 IF(ABS(XR-XL)-TOL)12,12,13
   12 IF(ABS(FR-FL)-TOLF)14,14,13
   13 CONTINUE
C     END OF BISECTION LOOP
C
C     NO CONVERGENCE AFTER IEND ITERATION STEPS FOLLOWED BY IEND
C     SUCCESSIVE STEPS OF BISECTION OR STEADILY INCREASING FUNCTION
C     VALUES AT RIGHT BOUNDS. ERROR RETURN.
      IER=1
   14 IF(ABS(FR)-ABS(FL))16,16,15
   15 X=XL
      F=FL
   16 RETURN
C
C     COMPUTATION OF ITERATED X-VALUE BY INVERSE PARABOLIC INTERPOLATION
   17 A=FR-F
      DX=(X-XL)*FL*(1.+F*(A-TOL)/(A*(FR-FL)))/TOL
      XM=X
      FM=F
      X=XL-DX
      TOL=X
      F=FCT(TOL)
      IF(F)18,16,18
C
C     TEST ON SATISFACTORY ACCURACY IN ITERATION LOOP
   18 TOL=EPS
      A=ABS(X)
```

```
      IF(A-1.)20,20,19
   19 TOL=TOL*A
   20 IF(ABS(DX)-TOL)21,21,22
   21 IF(ABS(F)-TOLF)16,16,22
C
C     PREPARATION OF NEXT BISECTION LOOP
   22 IF(SIGN(1.,F)+SIGN(1.,FL))24,23,24
   23 XR=X
      FR=F
      GO TO 4
   24 XL=X
      FL=F
      XR=XM
      FR=FM
      GO TO 4
C     END OF ITERATION LOOP

C     ERROR RETURN IN CASE OF WRONG INPUT DATA
   25 IER=2
      RETURN
      END
```

Appendix E: IDL scripts to analyse and plot results

I wrote the following IDL subroutines to analyse and plots results, this task is required for chapters 4, 5 and 6. And the second subroutine is also required for chapter 7.

```
;***********************************************************
pro twelve_month_run_mean,infile,outfile,n_records
;program to apply a 12 month running mean algorithm on
;secular variation data values stored in a file of
;4 columns of epoch, H, D and Z or epoch, X, Y and Z
;input:
       infile: input file of data to be processed
       n_records: number of records in the input file
;output:
       outfile: output file to store processed data
;
;Author: E. Nahayo
;***********************************************************
;number of records
;-----------------
n_records = 480
;name of the file holding data
;----------------------------
infile='c:\research\data\secVdata2001_2010.dat'
;name of the processed data file
;------------------------------
outfile='c:\research\data\Rm_secVdata2001_2010.dat'
openr,lun,infile,/get_lun
openw,lunw,outfile,/get_lun
```

```
data=fltarr(4,n_records)
;read data into the the array data
;---------------------------------
readf,lun,data
;initialise the variables to hold data and assign them
;0.0 value
;--------------------------------------------------------
sumx=0.0
sumy=0.0
sumz=0.0
epoch_s=0.0
out_n_records=n_records-12
for st=0, out_n_records-1 do begin
   for k=st,st+11 do begin
     epoch_s=epoch_s+data(0,k)
     sumx=sumx+data(1,k)
     sumy=sumy+data(2,k)
     sumz=sumz+data(3,k)
   endfor
   ;get the average values over 12 months
 ;---------------------------------------
   epochav=epoch_s/12
   xav=sumx/12
   yav=sumy/12
   zav=sumz/12
   ;save the new data record in the output file
;--------------------------------------------
   printf,lunw,format='(f9.3,3f6.1)',epochav,xav,yav,zav
   ;initialise variables and assign them 0.0 value
;-----------------------------------------------
   sumx=0.0
   sumy=0.0
   sumz=0.0
   epoch_s=0.0
endfor
free_lun,lun
free_lun,lunw
end
%-------------------------------------------------------------------------
```

```
pro contour_gridded_data,data_file,latmin,latmax,lonmin,lonmax,$
grid_interv,epoch,comp,model
;************************************************************
;program to plot the contours of field components of data on
;a specified area.
;data file has 6 columns of lat.,lon., x, y, z and f.
;The program outline the borders of the countries on map.
;input:
        data_file: input file of data to be contoured
;       latmin: minimum value of latitude
;       latmax: maximum value of latitude
;       lonmin: minimum value of longitude
;       lonmax: maximum value of longitude
;       grid_interv: grid space interval
;       epoch: epoch on which data was computed
;       comp: field component (X, Y or  Z) or total field F
;       model: the name of the model
;output:
        fileps: postscript file of the data contour
;        created as output of the program
;************************************************************
;constants
;---------------------------------------------------
data_file='c:\research\data\main_XYZF_data.dat'
epoch=2006.5
epochstr=string(epoch,'(f6.1)')
model=' SARM'
comp='Z'
grid_interv=2
latmin = -38 ;min latitude
latmax = -10 ;max latitude
lonmin = 10 ;min longitude
lonmax = 38 ;max longitude
;----------------------------------------------------
n_elts_lat=(latmax-latmin)/grid_interv+1 ;15 in this case
n_elts_lon=(lonmax-lonmin)/grid_interv+1 ;15 in this case
tot_nrows=n_elts_lat*n_elts_lon
fileps='c:\research\plots\'+epochstr+comp+'_'+model+'.ps'
```

```
;declare arrays to hold data
;--------------------------
lat_arr=fltarr(n_elts_lat)
lon_arr=fltarr(n_elts_lon)

;array to hold temporarily data to be contoured
;----------------------------------------------
;and 2 dim. array (n_elts_lon,n_elts_lat) cont_arr
temp_arr=fltarr(4,tot_nrows)
cont_arr=fltarr(n_elts_lon,n_elts_lat)

;define lat_arr and lon_arr arrays
;--------------------------------
lat=latmax
for s=0,n_elts_lat-1 do begin
      lat_arr(s)=lat
      lat=lat-grid_interv
endfor
lon=lonmin
for s=0,n_elts_lon-1 do begin
      lon_arr(s)=lon
      lon=lon+grid_interv
endfor

;read data into the temporary array temp_arr
;-------------------------------------------
openr,lun,data_file,/get_lun
for j=0,tot_nrows-1 do begin
   readf,lun,lat,lon,h,d,z,f,format='(2f8.3,4f10.2)'
   temp_arr(0,j)=h & temp_arr(1,j)=d & temp_arr(2,j)=z
   temp_arr(3,j)=f
endfor
free_lun,lun


;define cont_arr
;---------------
m=0
for i=0,n_elts_lat-1 do begin
```

```
 for j=0,n_elts_lon-1 do begin
       if comp eq 'X' then begin
      cont_arr(j,i)=temp_arr(0,m)
      m++
    endif else if comp eq 'Y' then begin
      cont_arr(j,i)=temp_arr(1,m)
      m++
    endif else if comp eq 'Z' then begin
      cont_arr(j,i)=temp_arr(2,m)
      m++
    endif else if comp eq 'F' then begin
      cont_arr(j,i)=temp_arr(3,m)
      m++
   endif
 endfor
endfor
;labels
;------
if comp eq 'X' then unitstr=model+': X (nT)'
if comp eq 'Y' then unitstr=model+': Y (nT)'
if comp eq 'Z' then unitstr=model+': Z (nT)'
if comp eq 'F' then unitstr=model+': F (nT)'

;contour data and create postscript file
;----------------------------------------
SET_PLOT, 'PS'

; Load discrete color table:
;---------------------------
tek_color;it has 32 colors

; Match color indices to colors we want to use:
;---------------------------------------------
black=0 & white=1 & red=2
green=3 & dk_blue=4 & lt_blue=5

;Set the PostScript device to *8* bits per color
;-----------------------------------------------
DEVICE, FILE=fileps,SCALE_FACTOR=1., BITS_PER_PIXEL=8, $
```

```
     XOFFSET=1.5,YOFFSET=2, XSIZE=20.0,YSIZE=25.0, /COLOR

MAP_SET,0,0,/ISOTROPIC,/MERCATOR,/HIRES,/NOBORDER,xmargin=[5,5],$
 ymargin=[4,4],LIMIT=[latmin, lonmin, latmax, lonmax],charsize=1.5,color=0
diflat=latmax-latmin
diflon=lonmax-lonmin

; Show national borders:
;-----------------------
MAP_CONTINENTS,/COUNTRIES,/HIRES,COLOR=0, MLINETHICK=1
XYOUTS,lonmin-(1.1/20.0)*diflon-.5,(latmin+latmax)/2, 'Latitude ()',$
        orientation=90, Align=0.5,charsize=2.,charthick=2,color=0
XYOUTS,(lonmin+lonmax)/2,latmin-(1.8/22.0)*diflat, 'Longitude ()', $
        Align=0.5 ,charsize=2.,charthick=2,color=0

;contour data for a selected field component
;-------------------------------------------
if comp eq 'Y' then begin
    levs=intarr(20)
   start=-8000
   for i=0,19 do begin
     levs(i)=start
     start=start+500
   endfor
  labs=string(levs,'(i5)')
  contour,cont_arr,lon_arr,lat_arr,levels=levs,color=0,$
     c_annotation=labs,/overplot,c_charsize=2.5,charthick=5
endif else if comp eq 'X' then begin
        levs=intarr(13)
    start=9800
    for i=0,12 do begin
      levs(i)=start
      start=start+500
    endfor
   labs=string(levs,'(i5)')
   contour,cont_arr,lon_arr,lat_arr,levels=levs,$
    charthick=0,color=0,c_annotation=labs,/overplot,$
     c_charsize=2.5
endif else if comp eq 'Z' then begin
```

```
  levs=intarr(20)
  start=-24000
  for i=0,19 do begin
    levs(i)=start
    start=start+500
  endfor
  labs=string(levs,'(i6)')
  contour,cont_arr,lon_arr,lat_arr,levels=levs,color=0,$
     c_annotation=labs,/overplot,c_charsize=2.5,charthick=5
  endif else if comp eq 'F' then begin
    levs=intarr(13)
  start=25000
  for i=0,12 do begin
    levs(i)=start
    start=start+500
  endfor
  labs=string(levs,'(i6)')
  contour,cont_arr,lon_arr,lat_arr,levels=levs,color=0,$
     c_annotation=labs,/overplot,c_charsize=2.5
endif
DEVICE, /CLOSE,scale_factor=1.0
```