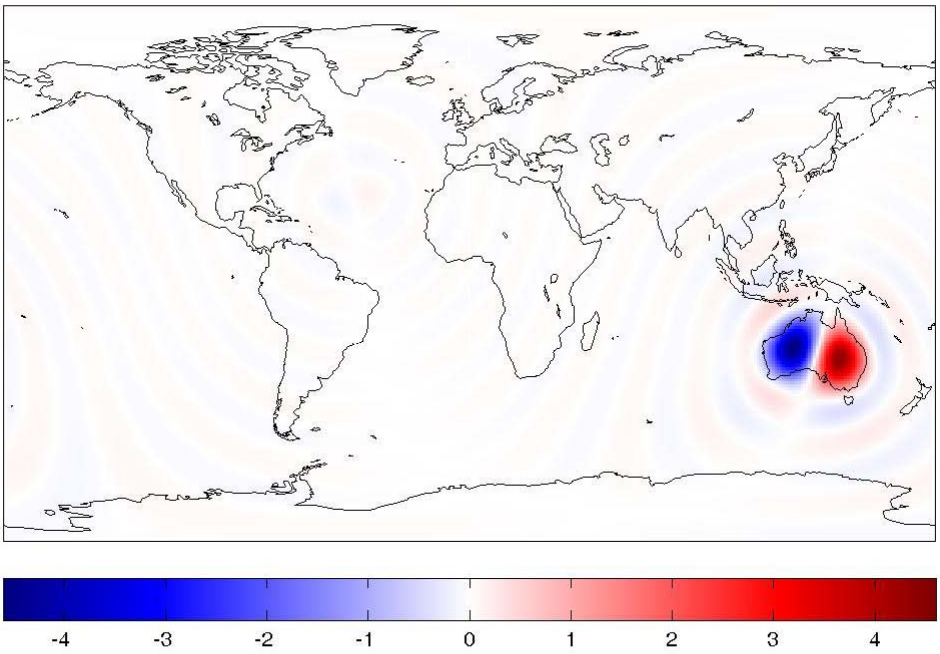
Universität Stuttgart

Geodätisches Institut

Sphärische Slepian-Funktionen



Studienarbeit im Studiengang
Geodäsie und Geoinformatik
an der Universität Stuttgart

Zhiguo Deng

Stuttgart, Mai 2007

Betreuer:
Dipl. -Ing Oliver Baur, & Dr. -Ing Matthias Weigelt, Universität Stuttgart

Prüfer:
Prof. Dr.-Ing. Nico Sneeuw, Universität Stuttgart

Selbstständigkeitserklärung

Hiermit erkläre ich, Zhiguo Deng, dass ich die von mir eingereichte Studienarbeit zum Thema

„Sphärische Slepian-Funktionen“

selbstständig verfasst und nur unter der angegebenen Hilfsmittel und Quellen verwendet habe.

Datum, Ort:_____

Unterschrift:_____

Zhiguo Deng

Zusammenfassung

Sphärische Slepian-Funktionen

Die am 17.03.2002 gestartete Satellitenmission GRACE liefert monatlich die Erdschwerefeldkoeffizienten c_{lm} und s_{lm}. Wegen deren sehr hohen Genauigkeit können die Daten der GRACE-Mission für die Hydrologie verwendet werden.

Um die Störeinflüsse außerhalb unserer interessierten Region eliminiert zu werden, verwenden wir die kugelförmigen Slepian-Funktionen.

Diese Studienarbeit beschäftigt sich mit der Berechnung der kugelförmigen Slepian-Funktionen auf einer interessierten beliebigen Region. Unter der Verwendung der Programme von Frederik J. Simons wird die kugelförmigen Slepian-Funktionen untersucht, und die Programme dokumentiert werden.

Die kugelförmigen Slepian-Funktionen können nach der Lösung des algebraischen Eigenwertproblems lösen. Die Elemente der Kernmatrix D des lokalisierten Gebiets werden durch das Gauss-Legendre Integrations-Polynom approximiert.

Schlüsselwörter:

- Slepian-Funktionen auf einer beliebigen Region
- Kugelflächenfunktionen
- bandbegrenztes Signal
- raumbegrenztes Signal
- Gauss-Legendre Integrations-Polynom

1. Motivation 2
2. Grundlagen 3
2.1 GRACE 3
2.2 Slepian-Funktionen 4
3. Aufstellung der Slepian-Funktion 7
4. Aufstellung der Programme von Frederik J. Simons 11
4.1. Hauptprogramme : 11
4.1.1. Slepian (Basis Funktionen) 11
4.1.2. glmalpha (Auslösung des EVPs) 11
4.2. Berechne der Matrix D des lokalisierten Gebiets 12
4.3. kernelc (Berechnung der **D** Matrix) 13
4.3.1. Spline (Bestimmung der Grenzpunkte des lokalisierten Gebiets) 15
4.3.2. gausslegendrecof (Koeffizienten für G-L Integrations-Polynom) 18
4.3.3. dphregion (Vorverarbeitung des untersuchenden Gebiets) 20
4.4. Kugelflächenfunktionen 23
4.4.1. ylm (Kugelflächenfunktionen) 23
4.4.2. xlm (zugehörig normierten Legendre Funktionen) 24
5. Quellen- und Literaturverzeichnis 26
6. Programme 27
6.1. Slepian 27
6.2. glmalpha 28
6.3. kernelc 32
6.4. Plotcont 36
6.5. Australia 38
6.6. bezier 40
6.7. gausslegendrecof 40
6.8. legendreroots 42
6.9. dphregion 42
6.10. phicurve 45
6.11. ylm 47
6.12. xlm 48
6.13. addmon (Definition for Indexing arrays) 50

1. Motivation

GRACE (Gravity Recovery and Climate Experiment) ist eine amerikanisch-deutsche Kleinsatelliten-Mission bestehend aus zwei Satelliten, die im Formationsflug durch Messung ihres Abstandes im Mikrowellenbereich Variationen in der Gravitation bestimmen. Sie liefert monatlich die Erdschwerefeldkoeffizienten c_{lm} und s_{lm}. Damit kann ein neues Modell des Erdgravitationsfeldes mit sehr hoher Genauigkeit erstellt werden. Ferner können die monatlichen Masseveränderungen der Oberflächendichte der Erde bestimmt werden. Der größte Anteil sind die Änderungen des Wasserspeichervermögens der Erde. So können die Daten der GRACE-Mission für die Hydrologie verwendet werden.

Unser Interesse liegt nun in bestimmten Gebieten der Erde, z.B. einem Kontinent, oder Staat. Aber bei der Berechnung der Änderungen des Wasserspeichervermögens eines bestimmten Gebiets gibt es die Störeinflüsse durch Massen des außerhalb zu untersuchenden Gebiets. Diese Störeinflüsse können eliminiert werden, indem die so genannten Slepian-Funktionen als Filter benutzt werden.

Slepian-Funktionen haben einige wichtige Eigenschaften.

1. Sie sind othogonal Innerhalb dem untersuchenden Gebiet.
2. Sie sind othonormal auf der ganzen Kugel.
3. Sie haben eine finite Dimension im Spektrumsraum.
4. Die Eigenschaften dieser Funktionen können sehr gut approximiert werden, und sich wesentlich leichter berechnen lassen.

Innerhalb einem untersuchenden Gebiet sind die Slepian-Funktionswerte größer Null und außerhalb dieses Gebiets möglichst gleich Null.

In meiner Studienarbeit werden die Slepian-Funktionen durch die Programme von Frederik J. Simons untersucht, und die Programme werden dokumentiert.

2. Grundlagen

2.1 GRACE

Die Satellitenmission GRACE ist ein Gemeinschaftsprojekt der amerikanischen Raumfahrtbehörde NASA in Zusammenarbeit mit dem Deutschen Zentrum für Luft- und Raumfahrt (DLR). Ziel der GRACE-Mission ist die Aufzeichnung von Veränderungen des Erdschwerefelds über einen Zeitraum von fünf Jahren sowie die Erstellung von hochgenauen Erdschwerefeldmodellen.

Missionsparameter	
Start:	17.03.2002, 10.21 MEZ
Ort:	Plesetsk Cosmodrome, Russland
Trägerrakete:	Rocket, Breeze-KM
Orbit:	500 km Höhe, 95 Minuten pro Erdumrundung
Gewicht der GRACE-Satelliten:	2 x 500 Kilogramm
Messsignal:	K-Band-Frequenzen
Datenübertragungssignal:	S-Band-Frequenzen
Zusätzliche Sensoren:	Sternenkameras
	Akzelerometern
	GPS-Antennen

Quelle: http://op.gfz-potsdam.de/grace/index_GRACE.html

Die GRACE-Mission besteht aus zwei, bis auf die Funkfrequenzbänder, baugleichen Satelliten, auch Zwillingssatelliten genannt. Das zentrale Beobachtungsgerät zur Positionierung der Satelliten und zur Durchführung von adiookkultationsmessungen ist der von NASA/JPL entwickelte *GPS-Blackjack-Empfänger* (Dunn et al., 2003). Dieses Gerät besteht aus einer 16-Kanal-Empfängereinheit und drei Antennen. Der Empfänger liefert fast ununterbrochen alle 10 Sekunden eine Navigationslösung mit einem mittleren Fehler von etwa ±5 m.
Um die extreme Genauigkeit der GRACE-Mission erst zu ermöglichen, wird das von NASA/JPL gebaute Zweifrequenz-K-Band-Entfernungsmesssystem (HAIRS) mit *GPS-Blackjack-Empfänger* verbunden. Damit können der Abstand und die Abstandsänderungen zwischen den zwei Satelliten sehr genau gemessen werden.

Die zwei Satellitenbahnen sind vom integrierten Effekt der Massenverteilungen und der Massenbewegungen abhängig. Und durch diese Effekte werden die zwei Satellitenbahnen an den leichten unterschiedlichen Phasen beeinflusst und unterschiedlich gestört. Dieser Störungsunterschiede verursachen die

Streckenänderungen Zwischen den zwei Satelliten, die werden durch die K-Band sehr genau (< 1 µm/s) gemessen. Diese Messwerte können zusammen mit GPS Beobachtungen verwendet werden, um Inhalt der höheren Frequenz des Gravitationssignals zu beobachten, um die Genauigkeit der Schätzung des Schwerfeldes der Erd zu erhöhen.

Nach Untersuchung von F. Flechtner (GFZ), Die Genauigkeit des Geoids mit GRACE ist Millimeter. In der folgenden Tabelle gezeigt die Höhestörungen des geschätzte Geoids, die von 30 Tage – GRACE - Daten abgeleitet werden:

Harmonische Grad	Geoid Höhefehler [mm] (per Grad)	Geoid Höhefehler [mm] (Kumulativ (ab n=3))
n=2	<0.10	-
3 <= n <= 10	< 0.01	<0.02
10 <= n <= 70	< 0.15	< 0.40
70 <= n <= 100	< 1.50	< 3.50
100 <= n <= 150	< 65.0	< 200.0

Quelle: http://op.gfz-potsdam.de/grace/index_GRACE.html

2.2 Slepian-Funktionen

Auf der Kugel kann ein reales quadratisches integrierbares Signal durch die harmonische Kugelflächenfunktionen entwickeln und in den Frequenzraum transformieren. Hier muss ich noch zwei Begriffe erklären.

Ein raumbegrenztes Signal:

Es handelt sich um ein raumbegrenztes Signal, Wenn das Signals außerhalb eines bestimmenden Raumgebiets gleich Null ist.

Ein bandbegrenztes Signal:

Es handelt sich um ein bandbegrenztes Signal, Wenn die Amplitude des Signals, nach Transformation in den Frequenzraum außerhalb eines bestimmenden Frequenzgebiets gleich Null ist.

Es gibt kein Signal, das gleichzeitig bandbegrenzt und raumbegrenzt ist, haben Slepian, Landau und Pollak In den sechziger Jahren ein grundlegendes Problem in der Informationstheorie gelöst, dass ein gegebenes Signal in Zeit- und Frequenzraum optimal zu konzentrieren, So genannt Slepian-Funktionen. Nach der weiten Entwicklung (*Albertella A., Sans`o F., Sneeuw N. (1999), und* Simons F.J., Dahlen, F.A., Wieczorek M.A. (2006)) können die Slepian-Funktionen auf der Kugel verwendet werden.

Wir verwenden $S_L = \{ g : \langle g_l^2 \rangle = 0 \ f\text{鬱} \ L < l < \infty \}$, um die bandbegrenzten Funktionen mit Bandbreit $\boldsymbol{L}$ zu bezeichnen. $\langle g_l^2 \rangle = \int_\Omega g_l^2 \, d\Omega$ ist Durchschnittenergie.

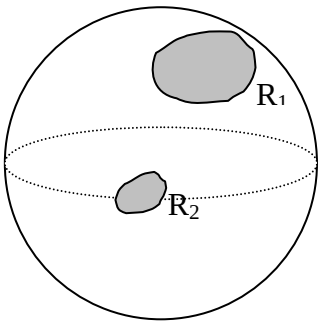
(1) $$g(\theta,\varphi)=\sum_{l=0}^{L}\sum_{m=-l}^{l} g_{lm} Y_{lm}(\theta,\varphi)$$

Mit Kugelflächenfunktion $Y_{lm}(\theta, \varphi)$, die sind auch die Basisfunktionen.

(2) $$Y_{lm}(\theta,\varphi) = \begin{cases} \sqrt{2} X_{lm}(\theta)\cos m\varphi & \text{if } -l \le m < 0 \\ X_{l0}(\theta) & \text{if } m = 0 \\ \sqrt{2} X_{lm}(\theta)\sin m\varphi & \text{if } 0 < m \le l \end{cases}$$

$X_{lm}(\theta, \varphi)$ ist die zugehörig normierten Legendre Funktionen.

Es gibt zwei Teilräumen auf der Einheitskugel Ω, $R = R_1 \cup R_2 \cup \ldots$ wir interessiert sind, und Ω-R. Wir verwenden $S_R = \{ h : h = 0 \ f\text{鬱} \ \Omega \notin R \}$, um die raumbegrenzten Funktionen in definierten Region R zu bezeichnen.



S_L hat eine $(L+1)^2$ Dimension, da der Vektor der Koeffizienten $\mathbf{g} = (g_{00} \ \mathrm{L} \ g_{LL})^T$ der Kugelflächenfunktionen insgesamt $\sum_{l=0}^{L}(2l+1) = (L+1)^2$ Elemente hat.

Es wird versucht, dass jene bandbegrenzten Funktionen $g(\hat{\mathbf{r}}) \in S_L$ optimal innerhalb einer räumlichen Region R konzentriert werden

Um die räumliche Konzentration einer bandbegrenzten Funktion innerhalb einer Region R zu maximieren, maximieren wir das Verhältnis der (halb) Normen:

(3) $$\lambda = \frac{\|g\|_R^2}{\|g\|_\Omega^2} = \frac{\int_R g^2 d\Omega}{\int_\Omega g^2 d\Omega} = \mathrm{Maximum}$$

Einsatz die Gleichung 1 in der Gleichung 3, kann λ so dargestellt werden:

(4) $$\lambda = \frac{\sum_{l=0}^{L}\sum_{m=-l}^{l} g_{lm} \sum_{l'=0}^{L}\sum_{m'=-l'}^{l'} D_{lm,l'm'} g_{l'm'}}{\sum_{l=0}^{L}\sum_{m=-l}^{l} g_{lm}^2}$$

Mit Definition

(5) $$D_{lm,l'm'} = \int_R Y_{lm} Y_{l'm'} d\Omega$$

Und es ist eine $(L+1)^2 \times (L+1)^2$ Matrix.

(6) $$\mathrm{D} = \begin{pmatrix} D_{00,00} & \mathrm{K} & D_{00,LL} \\ \mathrm{M} & \mathrm{O} & \mathrm{M} \\ D_{LL,00} & \mathrm{L} & D_{LL,LL} \end{pmatrix}$$

Mit Gleichung 6 kann die Gleichung 4 umformuliert werden.

(7) $$\lambda = \frac{\mathbf{g}^{\mathrm{T}} \mathbf{D}\, \mathbf{g}}{\mathbf{g}^{\mathrm{T}} \mathbf{g}} = \mathrm{Maximum}$$

Nach Umformt bekommen wir ein $(L+1)^2 \times (L+1)^2$ algebraisches Eigenwertproblem.

(8) $$\mathbf{D}\,\mathbf{g} = \lambda \mathbf{g}$$

Weil die Matrix D nach ihre Definition real, symmetrisch und positiv ist, sind die $(L+1)^2$ Eigenwerte λ und deren Eigenvektoren g real. Die Eigenwerte λ sind nach absteigend sortiert $0 < \lambda_{(L+1)^2} \leq \ldots \lambda_2 \leq \lambda_1 < 1$. Die Eigenvektoren g sind othogonal, nach der Normierung wird Die Eigenvektoren **g** orthonormal sein.

3. Aufstellung der Slepian-Funktion

In den Programmen von Frederik J. Simons wird die Slepian-Funktion für das gewählte Gebiet Australien berechnet.

Die untersuchten maximalen Grade sind 6, 10, 12, 18, 20, 24 und 30.

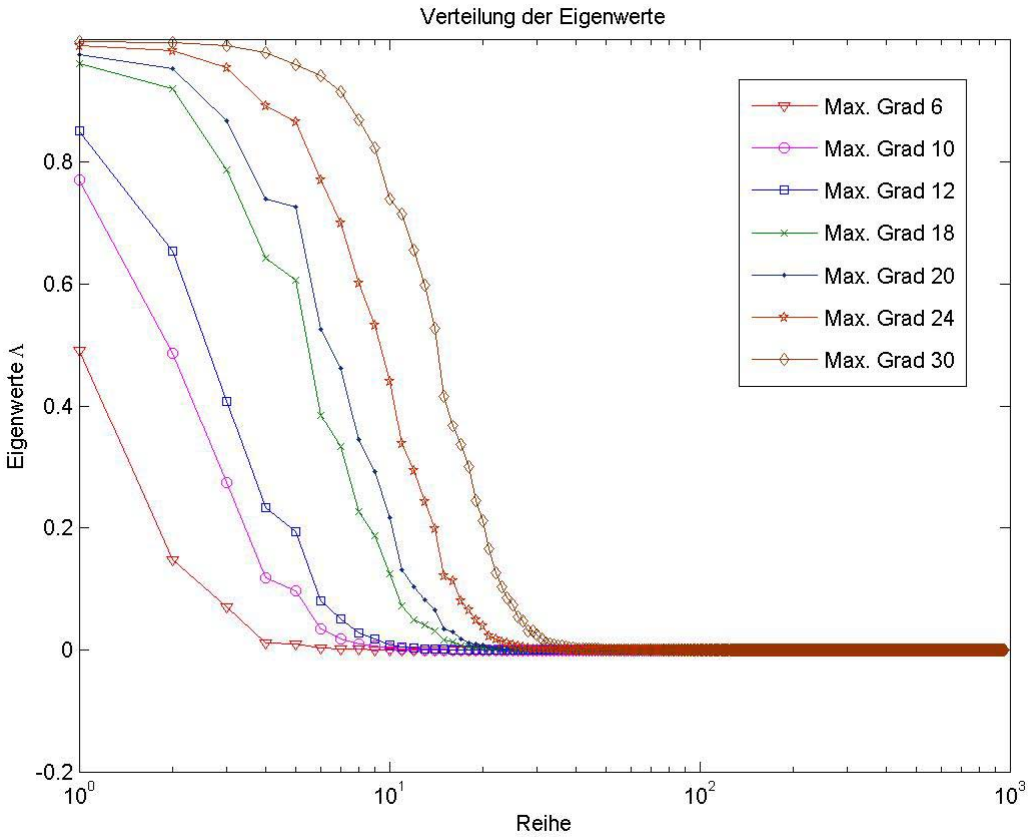


Abbildung 1: Zusammenstellung der Eigenwerte für Australien

Die Abbildung 1 zeigt uns die Verteilung der Eigenwerte für maximalen Grade 6, 10, 12, 18, 20, 24 und 30.

Weil Eigenwerte λ ist ein Maß, wie gut deren zugehörige $g(\hat{\mathbf{r}}) \in S_L$ innerhalb einer Region ***R*** konzentriert. Die Eigenfunktionen **g**, die gut innerhalb der Region ***R*** konzentriert werden, haben Eigenwerte λ nahe Einheit, während die, die schlecht konzentriert werden, Eigenwerte λ nahe null haben.

Für einen bestimmten Grad ***L*** ist die Summe der Eigenwerte so gegeben:

(12) $$N = \sum_{\alpha=1}^{(L+1)^2} \lambda_\alpha = \mathrm{tr}\,\mathbf{D} = \sum_{l=0}^{L} \sum_{m=-l}^{l} D_{lm,lm} = \int_R D(\hat{\mathbf{r}},\hat{\mathbf{r}}')\,d\Omega = (L+1)^2 \frac{\mathbf{A}}{4\pi}$$

N ist so genannt Shannon-Nummer. Dann wird die Gesamtzahl signifikanten Eigenwerten gut durch die gerundete Summe ***N*** approximiert. Aber für Praxis wird die Gesamtzahl signifikanten Eigenwerten nach Genauigkeitsanforderung entscheiden.

z.B. Bei $L = 10$ für Australien $A/4\pi = 1.50\%$ ist N =1.81, die gerundete Gesamtzahl ist 2. Aber die ersten zwei Eigenwerten sind λ_1=0.7707 und λ_2=0.4863, (λ_1+λ_1)/N= 69.4%. Das bedeutet, nur 69.4% Energie kann durch die ersten zwei Eigenvektoren repräsentiert werden. Weitere Eigenwerten werden auch als signifikanten Eigenwerten akzeptiert, bis zu angeforderte Genauigkeit erreicht wird.

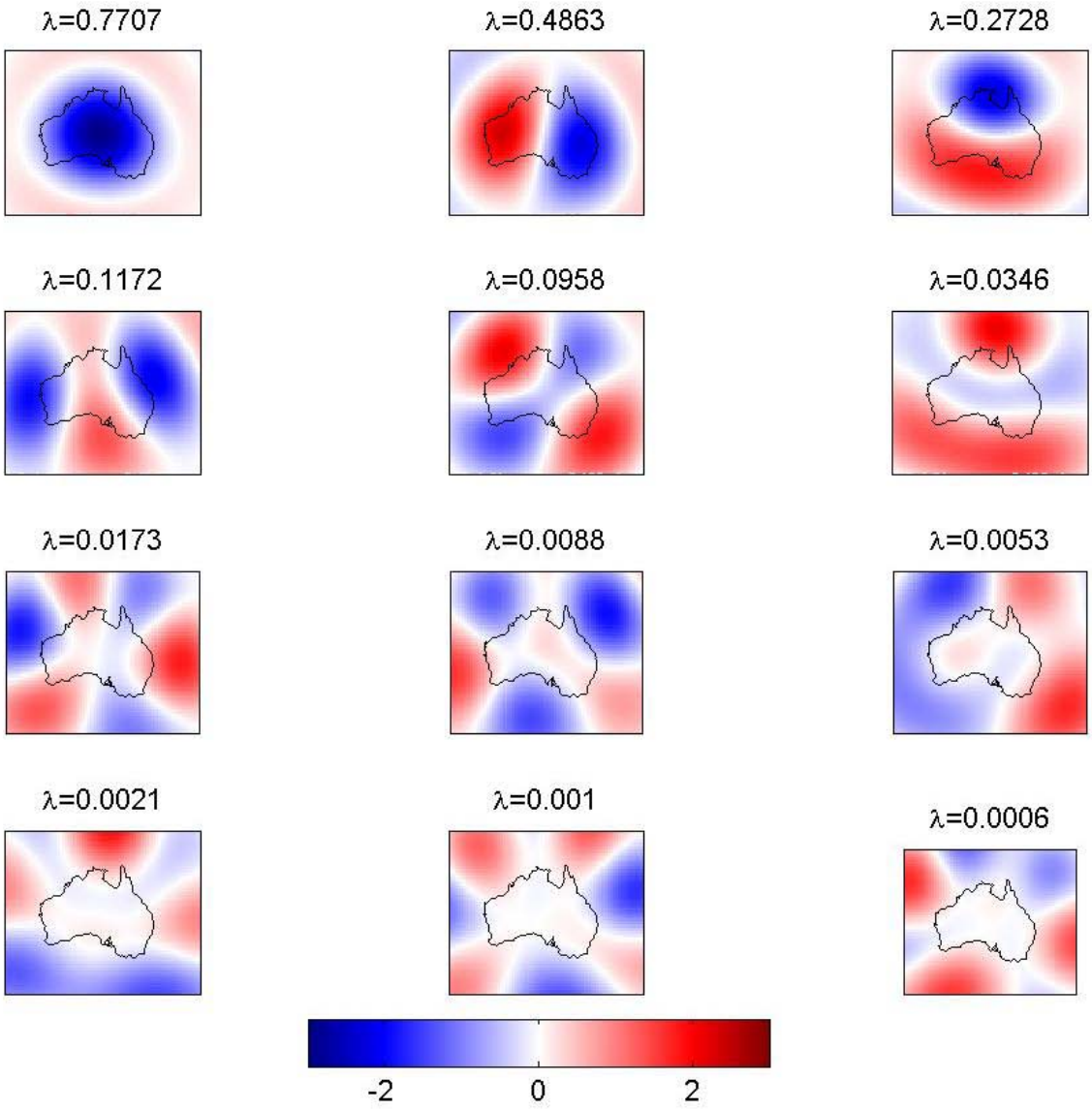


Abbildung 2: die Slepian-Funktionen (Max. Grad 10) für Australien

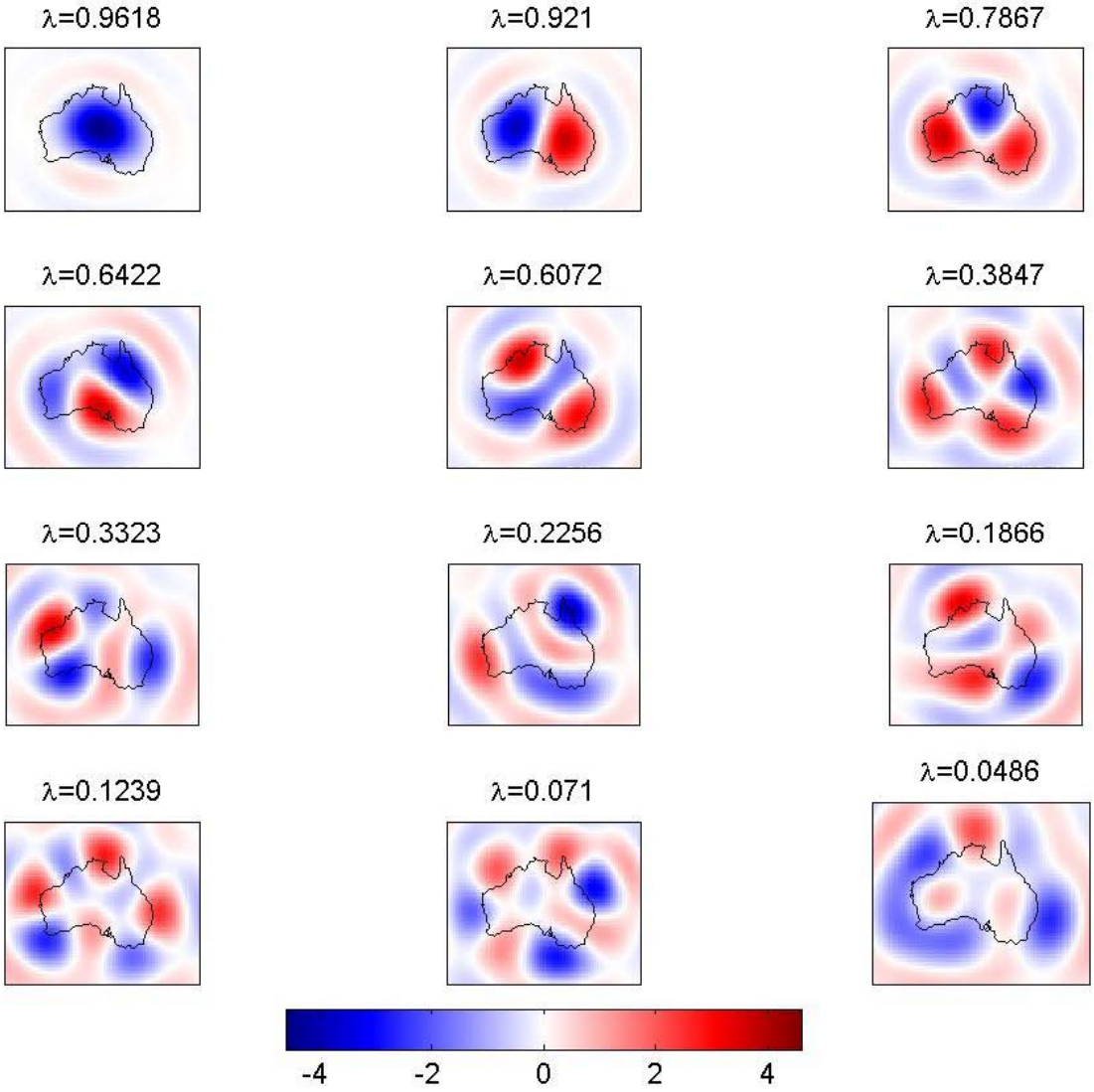


Abbildung 3: die Slepian-Funktionen (Max. Grad 18) für Australien

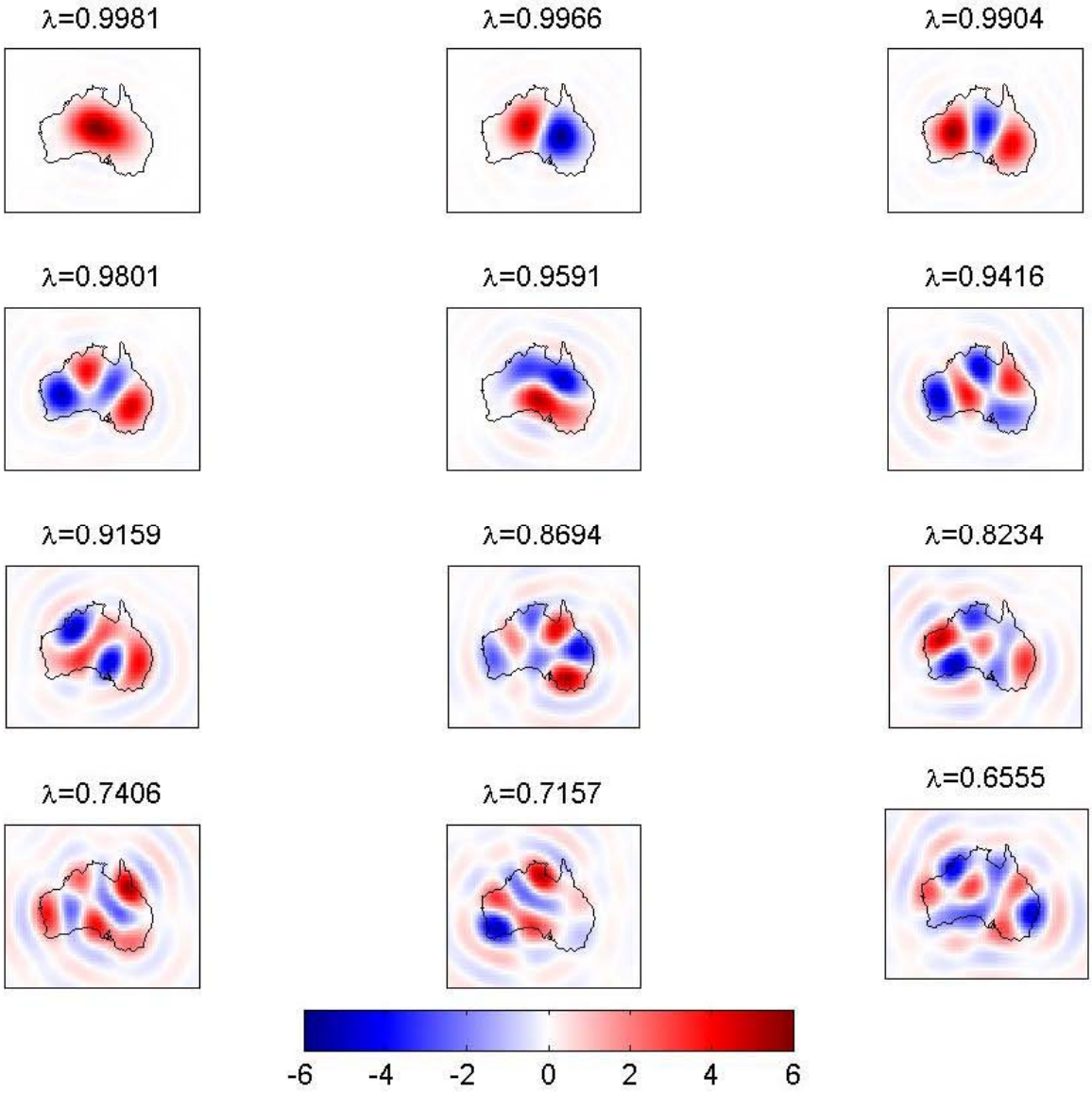
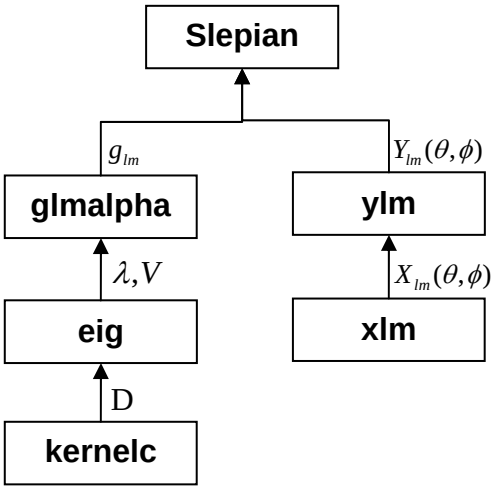


Abbildung 4: die Slepian-Funktionen (Max. Grad 30) für Australien

In den Abbildungen 2 bis 4 werden die Werte von Slepian-Funktionen von Australien für maximalen Grade 6, 10, 12, 18, 20, 24 und 30 dargestellt. Die Abbildungen zeigen uns, dass je größer der Max. Grad ist, konzentrieren die Slepian-Funktionen besser in der untersuchenden Region, und sind mehr Eigenwerte λ signifikanten, und ist die Übergang von fast Eins zu Null mehr schärfer.

4. Aufstellung der Programme von Frederik J. Simons

4.1. Hauptprogramme :



4.1.1. **Slepian** (Basis Funktionen)

In diesem Programm wird die Slepian-Funktion für lokalisierte Gebiet **R** berechnet. Mit der Gleichung:

$$g(\theta,\varphi)=\sum_{l=0}^{L}\sum_{m=-l}^{l} g_{lm}Y_{lm}(\theta,\varphi)$$

Für jeder 1°x1° Pix auf den Kugel wird die Slepian-Funktionswerte berechnet.

4.1.2. **glmalpha** (Auslösung des EVPs)

Hier werden die Eigenwerte und deren Eigenvektoren der Matrix **D** verwendeten Gebiets berechnet, und dann werden die Eigenwerte absteigend sortiert.

$$\mathbf{Dg} = \lambda\mathbf{g}$$

function [G,V,EL,EM,N]=glmalpha(TH,L,sord)

Eingaben:

TH	Breite, für kugelförmigen Fleck z.B. Breite 10° für Nordpol. Oder für ein gewünschtes Gebiet: ‚england', ‚eurasia', ‚namerica', ‚australia', ‚greenland', ‚africa', ‚samerica', ‚amazon', ‚orinoco'
L	maximale Grad (Bandweit)
sord	1 Single polar kugelförmige Fleck [Default] 2 Doubles polar kugelförmige Fleck N mal Verdichtung der Grenzpunke mit Spline Interpolation [Default: 10]

Ausgaben:

G die sortierte Eigenvektore.

V die absteigende Eigenwerte

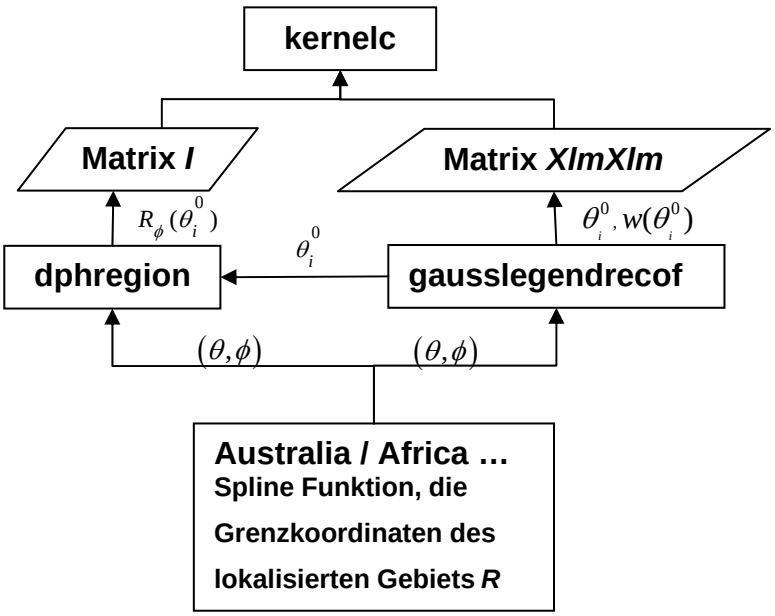
EL Index von G nach l

EM Index von G nach m

N der Shannon Nummer

Bei Aufruf der Funktion [G,V,EL,EM,N] = glmalpha ('australia',18,10) werden die Eigenwerte V und deren Eigenvektoren G für Australid Grad 18 berechnet. 10-mal werden die originalen Grenzpunkte von Australia verdichtet. G ist ein 361x361 Matrix, V ist ein 361 Vektor, N gleicht 5,4091.

4.2. Berechne der Matrix D des lokalisierten Gebiets



$\theta_i^0 \quad i = 1\mathrm{L}\ n$ Breite, der Nullstellen für das Polynom vom Grad $\boldsymbol{L}$ Gauss-Legendre Integration.

$w(\theta_i^0)_{1\times n}$ die Gewichte für das Polynom vom Grad $\boldsymbol{L}$ Gauss-Legendre Integration.

$R_\phi(\theta_i^0)$ die Länge Intervall für die passende $\theta_i^0 \quad i = 1\mathrm{L}\ n$

(θ,ϕ) die Grenzkoordinaten des lokalisierten Gebiets $\boldsymbol{R}$.

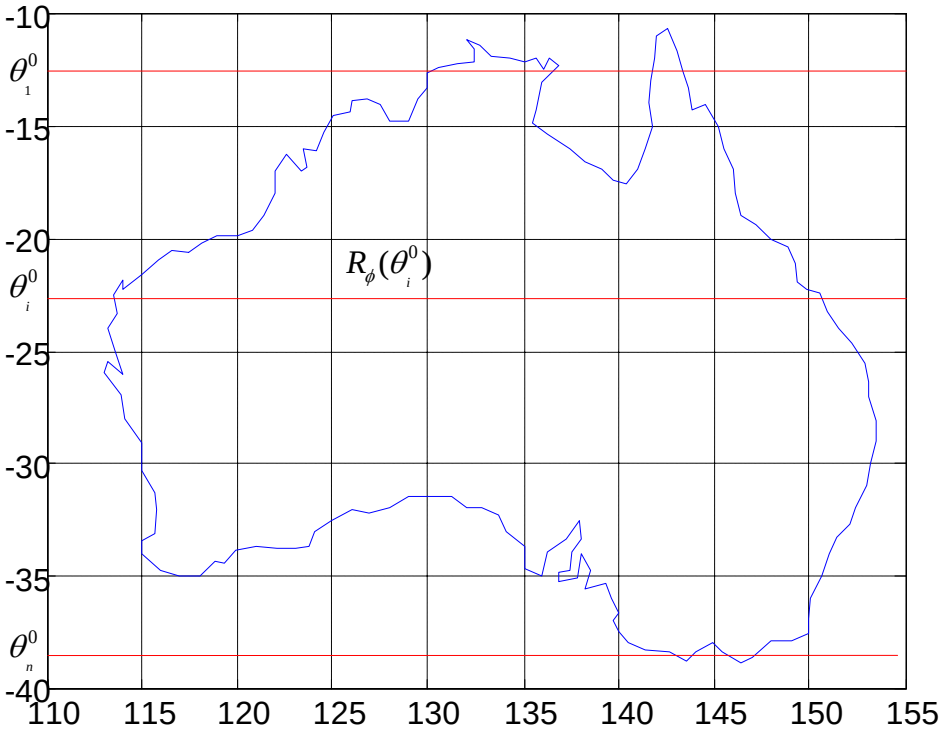


Abbildung 9: die Nullstellen der Gauss-Legendre Integration für Australien

Hier wird Australien als ein Beispiel genommen, x- Achse beträgt Länge in °, y-Achse beträgt Breite in °.

4.3. **kernelc** (Berechnung der **D** Matrix)

In Funktion Kernelc werden die **D** Matrix des lokalisierten Gebiets berechnet.

Function [Klmlmp, XY]=kernelc(Lmax,dom,pars,ngl)

Eingaben:

Lmax	maximale Grad (Bandweit)
dom	kugelförmige Fleck (wird in ‚pars' definiert) ‚patch' [Default: ‚patch'] oder gewünschte Gebiet: ‚england', ‚eurasia', ‚namerica', ‚australia', ‚greenland', ‚africa', ‚samerica', ‚amazon', ‚orinoco'
pars	der Parameter nur für ‚patch' [th0, ph0, thR] th0 Breit vom Mittel des Deckels, in rad ph0 Lange vom Mittel des Deckels, in rad thR Radius des Deckels, in rad
ngl	der approximierte Grad der Gauss-Legendre Integration [Default:200]

Ausgaben:

Klmlmp	Der lokalisierten Kerne D Matrix, deren Eigenfunktion wir brauchen, Index ist so: Grad [0 1 1 1 2 2 2 2 2] Ordnung [0 0 -1 1 0 -1 1 -2 2]
XY	Die Grenzkoordinaten des lokalisierten Gebiet.

In der Funktion [Klmlmp, XY]=kernelc(18, 'australia') wird die Matrix D (361x361) des Eigenwertproblems berechnet. XY sind die verdichteten Grenzkoordinaten von Australia.

$$\mathrm{D}=\begin{pmatrix} D_{00,00} & \mathrm{K} & D_{00,LL} \\ \mathrm{M} & \mathrm{O} & \mathrm{M} \\ D_{LL,00} & \mathrm{L} & D_{LL,LL} \end{pmatrix}_{(L+1)^2\times(L+1)^2}$$

nach Gauss-Legendre-Integration Funktion wird $D_{lm,\,l'm'}$ so approximiert.

$$D_{lm,\,l'm'}=\int_R Y_{lm}Y_{l'm'}d\Omega \approx \sum_{i=1}^{n} w(\theta_i^0)\cdot X_{lm,\,l'm'}(\theta_i^0)\cdot i_{mm'}(\theta_i^0)$$

Mit:

$\theta_i^0 \quad i=1\mathrm{L}\ n$	Die Nullstellen für das Polynom vom Grad *L* Gauss-Legendre Integration, In dem Programm wird als ***x*** genannt.
$w(\theta_i^0)_{1\times n}$	Die Gewichte für das Polynom vom Grad *L* Gauss-Legendre Integration, In dem Programm wird als ***w*** genannt.

Diese beide Koeffizienten können durch die Subfunktion function [w,x,N] = gausslegendrecof (l,method,intv) bekommen werden.

$i_{mm'}(\theta_i^0)$ wird so definiert:

$$i_{mm'}(\theta_i^0)=\begin{cases} \int_{R_\phi(\theta_i^0)} \cos(m\phi)\cos(m'\phi)d\phi & wenn, m<0, und\ \ m'<0 \\ \int_{R_\phi(\theta_i^0)} \cos(m\phi)\sin(m'\phi)d\phi & wenn, m<0, und\ \ m'>0 \\ \int_{R_\phi(\theta_i^0)} \sin(m\phi)\cos(m'\phi)d\phi & wenn, m>0, und\ \ m'<0 \\ \int_{R_\phi(\theta_i^0)} \sin(m\phi)\sin(m'\phi)d\phi & wenn, m>0, und\ \ m'>0 \end{cases}$$

In dem Programm wird $i_{mm'}(\theta_i^0)$ im Matrix ***I*** gespeichert.

$$I = \begin{pmatrix} i_{00}(\theta_1^0) & \mathrm{K} & i_{LL}(\theta_1^0) \\ \mathrm{M} & \mathrm{O} & \mathrm{M} \\ i_{00}(\theta_n^0) & \mathrm{L} & i_{LL}(\theta_n^0) \end{pmatrix}_{n\times(L+1)^2}$$

Für das restliche Teil $X_{lm,\,l'm'}(\theta_i^0)$ wird nach das untere Form berechnet.

$$X_{lm,\,l'm'}(\theta_i^0) = X_{lm}(\theta_i^0) X_{l'm'}(\theta_i^0)$$

$$X_{lm}(\theta_i^0) = (-1)^m \left(\frac{2l+1}{4\pi}\right)^{1/2} \left[\frac{(l-m)!}{(l+m)!}\right]^{1/2} P_{lm}(\cos\theta_i^0)$$

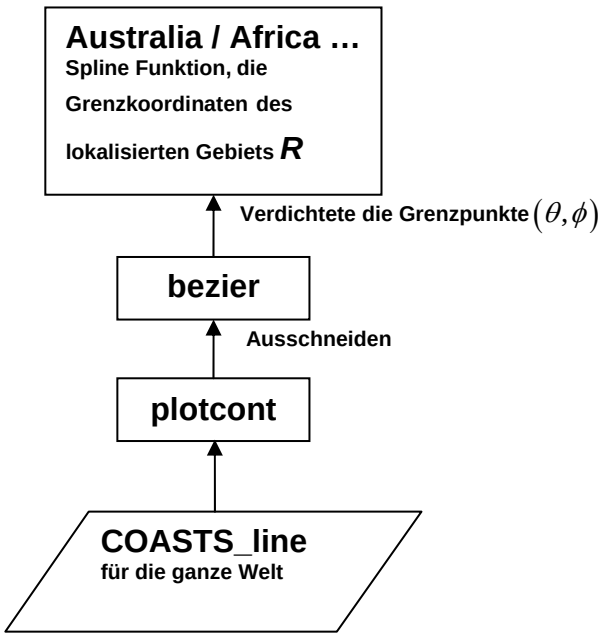
In dem Programm wird der Matrix ***XlmXlm*** berechnet.

$$XlmXlm = \begin{pmatrix} X_{00,00}(\theta_1^0) & \mathrm{K} & X_{00,LL}(\theta_1^0) & X_{10,LL}(\theta_1^0) & \mathrm{K} & X_{LL,LL}(\theta_1^0) \\ \mathrm{M} & \mathrm{O} & \mathrm{M} & \mathrm{M} & \mathrm{O} & \mathrm{M} \\ X_{00,00}(\theta_n^0) & \mathrm{L} & X_{00,LL}(\theta_n^0) & X_{10,LL}(\theta_n^0) & \mathrm{L} & X_{LL,LL}(\theta_n^0) \end{pmatrix}_{n\times\left[\frac{\left(\frac{(L+1)L}{2}+L+1\right)^2+\left(\frac{(L+1)L}{2}+L+1\right)}{2}\right]}$$

Um die Speicherplatz zu sparen wird es im Matrix ***XlmXlm*** nur für 0 ≤ m≤ l, 0 ≤ m' ≤ l gegeben, da $X_{lm,\,l'm'}(\theta_i^0) = X_{l-m,\,l'm'}(\theta_i^0) = X_{lm,\,l'-m'}(\theta_i^0) = X_{l-m,\,l'-m'}(\theta_i^0)$.

Endlich kann ***D*** die Matrix des Eigenwertproblems berechnet und gespeichert werden, in dem Programm wird ***D*** als ***KlmKlm*** genannt.

4.3.1. Spline (Bestimmung der Grenzpunkte des lokalisierten Gebiets)



(θ,ϕ) Die Grenzkoordinaten des lokalisierten Gebiets ***R***.

4.3.1.1. **Plotcont**(Abscheiden die Grenzpunkte des lokalisierten Gebiets)

Die Datei COASTS_line.mat erhalten die Punkte der Küstenlinien auf der ganze Welt

In diese Programme werden die Grenzpunkte ***R*** aus Datei für die ausgeschnitten.

Function varargout=plotcont(cll,cmn,res,ofs)

Eingaben:

cll	[Länge Breite] die Koordinaten des oben links Knoten für das interessiere Gebiet.
cmn	[Länge Breite] die Koordinaten des unten rechts Knoten für das interessiere Gebiet.
Res	0 normale Küstenlinien (Default)
	1 etwas höherer Auflösung für die Küstenlinien mit See
	2 Global Mollweide Projektion
	3 3D Koordinaten (für Ausgabe)
	4 3D Koordinaten, nur für Northern Hemisphäre
	5 die Küstenlinien mit hoher Auflösung für California
	6 einfache Begrenzungen für Australia
ofs	offset für Länge, z.B. 360 Grad [nur für 0,1,5,6]

Ausgaben:

axlim	:	Definition für Achse .
handl	:	Definition für Linien .
XYZ	:	Die Koordinaten der Grenzpunkte (2D Oder 3D)

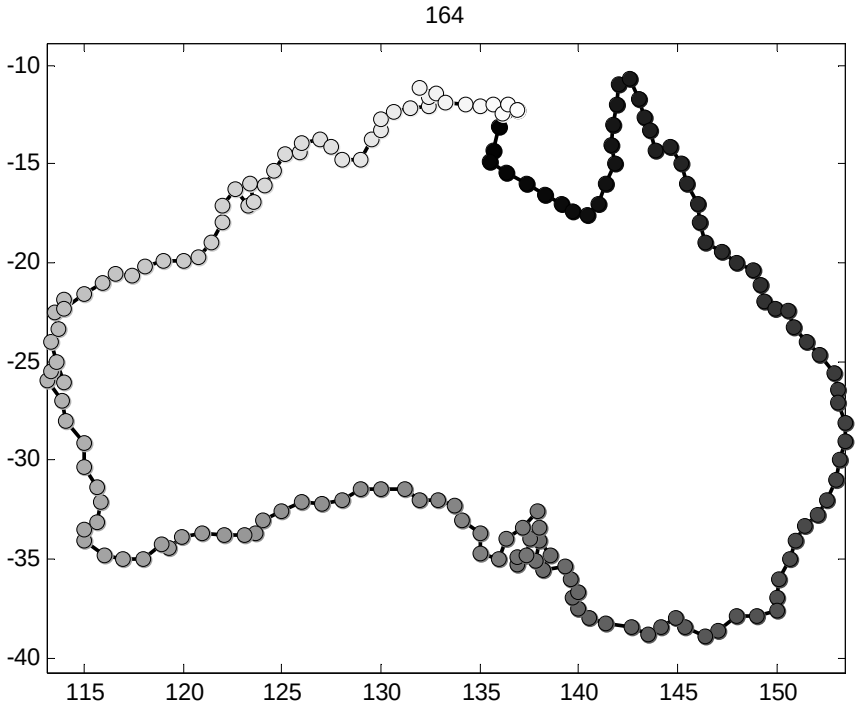


Abbildung 10: die originale Grenzpunkte von Australien

Abb. 10 wird Australien als ein Beispiel, x- Achs beträgt Länge in Grad, y- Achs beträgt Breite in Grad.
Insgesamt werden 164 Grenzpunkte aus Datei COASTS_line.mat für die ganze Welt mit Kommando XY=Plotcont([112 -10.5],[154 -39],[ ],360) aufgefunden.

4.3.1.2 Australia (Auffinden und Verarbeitung der Grenzpunkte)

Hier wird das ausschnitte Fernster des lokalisierten Gebiets definiert, und die Verdichtungsgrade eingegeben.

function varargout=australia(res,scal)

Eingaben:

res res mal Verdichtung der Küstenpunke mit Spline Interpolation [Default: 0]

scal Skala [Default: 0]

Ausgaben:

XY die Koordinaten (θ, ϕ) der Grenzpunkte lokalisierten Gebiet.

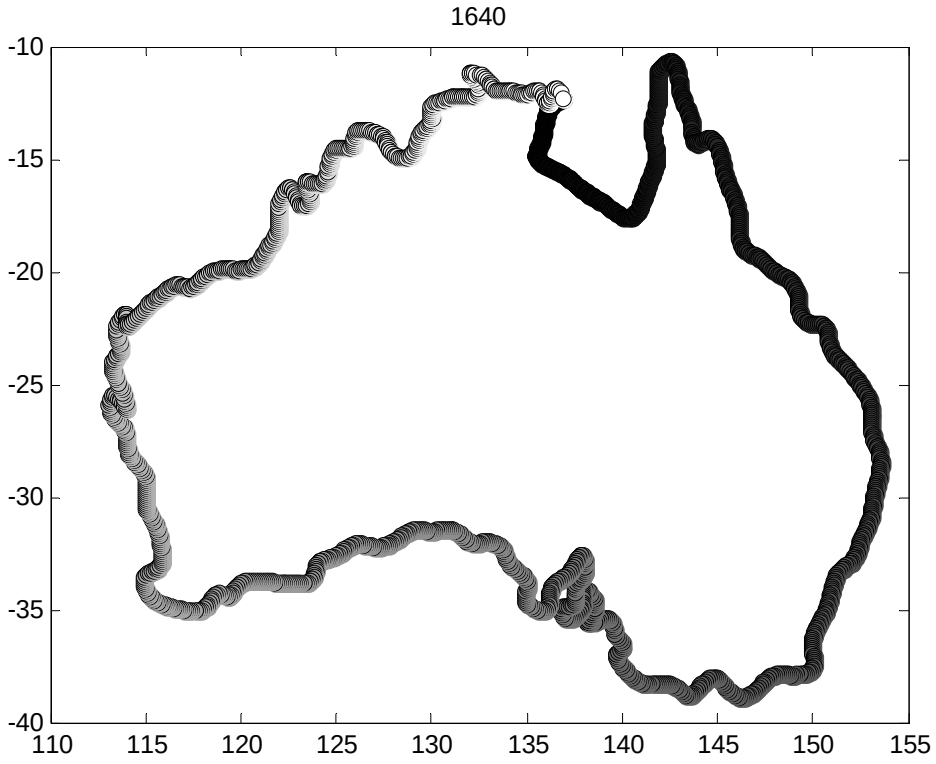
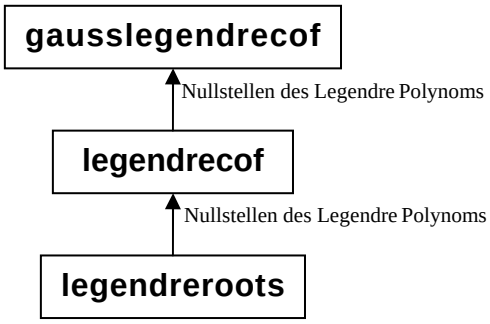


Abbildung 11: die verdichte Grenzepunkte für Australien

In function Xyb=bezier (XY,10) werden 164 originale Grenzpunkte nach Spline – Interpolation 10 mal verdicht.
Die Eingaben und Ausgaben in function varargout=africa(res,scal) sind gleich wie oben.
Beim isolierten lokalisierten Gebiet können die Grenzpunkte nach gleicher Methode wie oben bekommen. z.B. auch Afrika, England.
In andere Fallen werden die Grenzpunkte zuerst aus gegebener Datei geladen, dann können sie verdichtet werden. z.B. Amazon.

4.3.2. **gausslegendrecof** (Koeffizienten für G-L Integrations-Polynom)



Das Gauss-Legendre Integrations-Polynom hat 2*n-1 Grad Genauigkeit, A_k und x_k sind die Legendre Gewichte und Nullstellen.

$$\int_a^b f(x)dx \approx \sum_{k=0}^{n} A_k f(x_k)$$

4.3.2.1. **legendreroots** (Roots von Legendre Polynoms)

In diese Programme werden die Nullstellen des N-Grade Legendre Polynoms $\left|X^{0}_{Legendre}\right| \le 1$ berechnet.

function [r,J]=legendreroots(N);

Eingaben:

N	der Oder des Legendre Polynoms.

Ausgaben:

r	die Nullstellen des Legender Polynoms, zwischen -1 und 1.
J	der Jacobi Matrix, seine Eigenwerte sind die Nullstellen.

Der Jacobi Matrix $J = \begin{pmatrix} 0 & j_1 & 0 \\ j_1 & \mathrm{O} & j_{N-1} \\ 0 & j_{N-1} & 0 \end{pmatrix}_{N \times N}$ mit $j_i = \frac{i}{\sqrt{4i^2 - 1}}$

Die Nullstellen des Legendre Polynoms $r = eig(J)$

Weil wir später nur die Nullstellen des Legendre Polynoms brauchen, liefern die Funktion function [R,C,Cnorm]=legendrecof(L,method) einfach die Ergebnisse weiter zur function [w,x,N]=gausslegendrecof(l,method,intv)

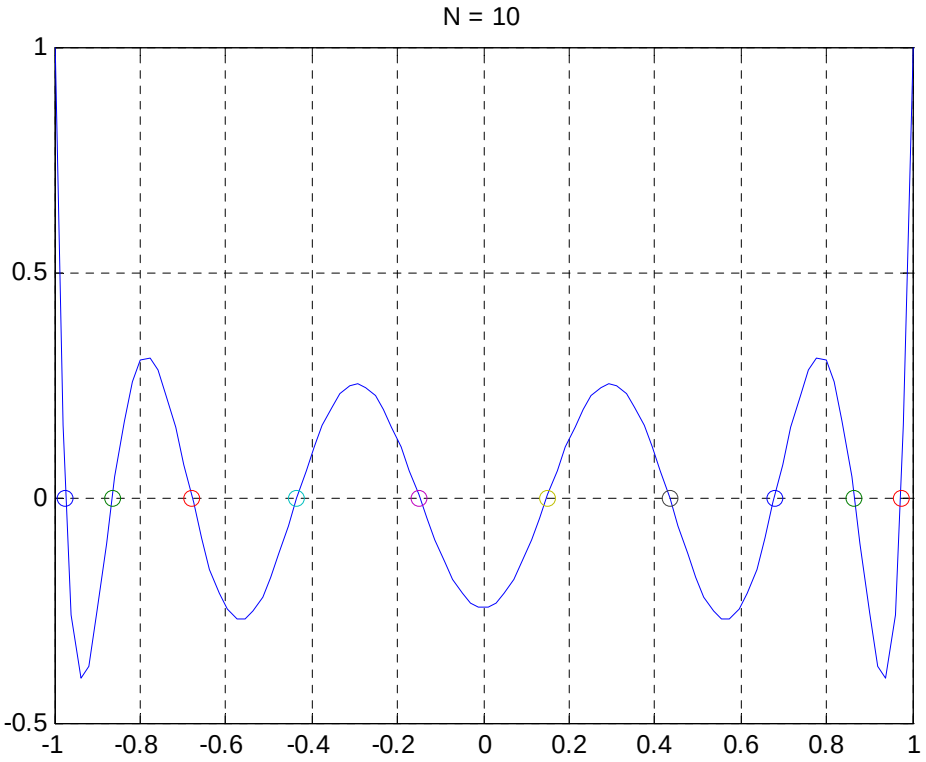


Abbildung 12: die Nullstellen des Legendre Polynoms N = 10

4.3.2.2. gausslegendreco (Verarbeitung der Koeffizienten)

In diese Programme werden die Nullstellen des Legendre Polynoms $\left| X^0_{Legendre} \right| \leq 1$ im $(\theta_{nord}, \theta_{s\text{ü}d})$ Integrationsintervall skaliert.

function [w,x,N]=gausslegendrecof(l,method,intv)

Eingaben:

L	Grad von Polynom für Integrand
method	'jacobi' stabler Algorithmus (nur mit Legendre Nullstelle)[Default]
	'cofrec' Algorithmus für l<40 (mit Nullstelle und Koeffizienten)
intv	das Integrationsintervall, die Nullstelle und ihre Gewichte[w,x] werden skaliert.

Ausgaben:

w	Gewichte.
x	die Nullstellen des Legender Polynoms von Grad N)
N	Zahl der gesamte Punkte in der Integration (length(x))

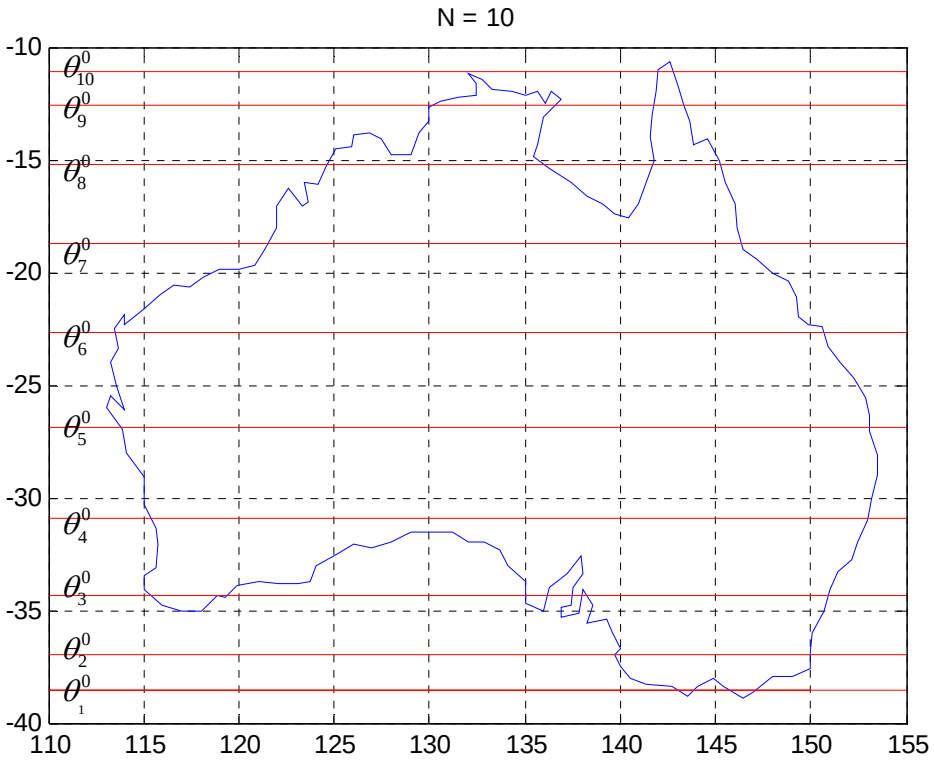
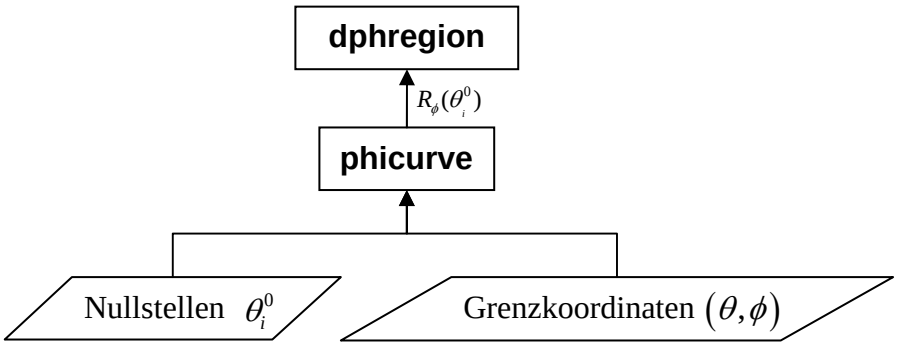


Abbildung 13: die Nullstellen des Legendre Polynoms N = 10 für Australien

Deren Gewicht *w* = [0.9401 2.1073 3.0891 3.7967 4.1669 4.1669 3.7967 3.0891 2.1073 0.9401]'

4.3.3. **dphregion** (Vorverarbeitung des untersuchenden Gebiets)



$\theta_i^0 \quad i = 1\mathrm{L}\ n$ die Breite, der Nullstellen für das Polynom vom Grad $\boldsymbol{L}$ Gauss-Legendre Integration.

$R_\phi(\theta_i^0)$ die Länge Intervall für die passende $\theta_i^0 \quad i = 1\mathrm{L}\ n$

(θ,ϕ) die Grenzkoordinaten des lokalisierten Gebiets $\boldsymbol{R}$.

4.3.3.1.dphregion

In diese Programme werden die gebrauchte $R_\phi(\theta_i^0)$ Parameter um zu berechnen vorbreitet und zum Subprogramme geliefert.

function [phint,thp,php]= dphregion(th,N,region)

Eingabe:

th	die Breite, die Nullstellen für das Polynom vom Grad L Gauss-Legendre Integration, in Grad
N	N mal Verdichtung der Küstenpunke mit Spline Interpolation [Default: 10]
region	Name des gewünschte Gebietes: 'africa', 'namerica', 'greenland', 'australia', 'eurasia', oder ein Matrix mit XY/[Länge ,Breite] Koordinaten

Ausgabe:

phint	Integrationsinterval der Länge für $R_\phi(\theta_i^0)$ gewünschtes Gebiet
thp	Breitematrix für graphische Darstellung
php	Längematrix für graphische Darstellung

4.3.3.2. phicurve

In diese Programme werden die Integrationsintervalle berechnet.

function [phint,thp,php]= phicurve(thph,th)

Eingabe:

thph	Die Koordinaten [Länge, Breite] der Grenzepunkte
th	Breite, die Nullstellen für das Polynom vom Grad L Gauss-Legendre Integration.

Ausgabe:

phint	Integrationsinterval der Länge für $R_\phi(\theta_i^0)$ gewünschtes Gebiet
thp	Breitematrix für graphische Darstellung
php	Längematrix für graphische Darstellung

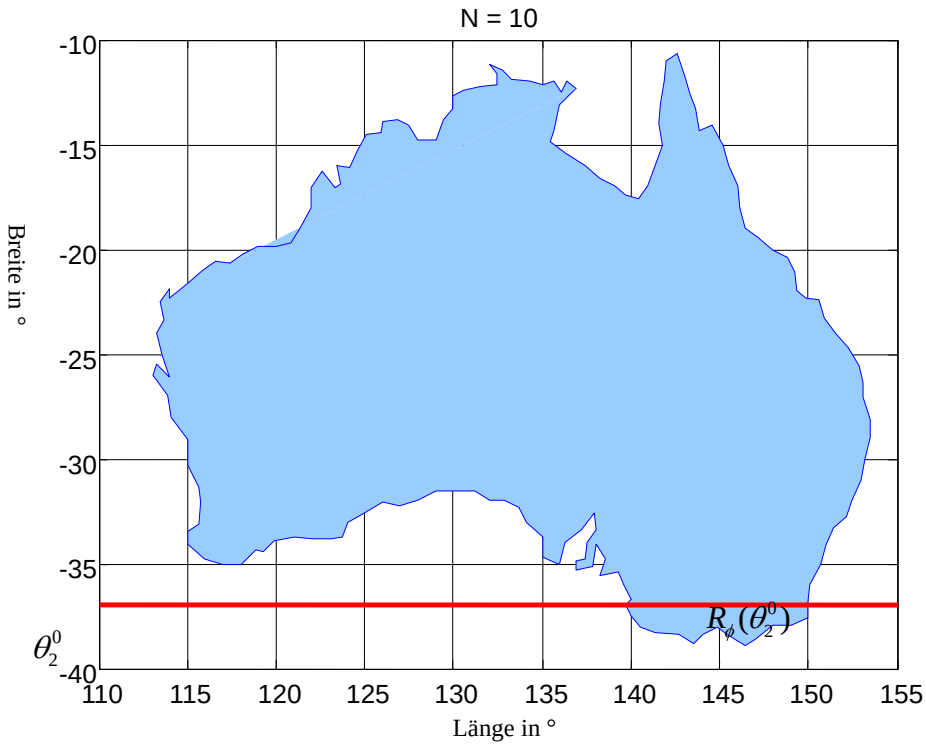


Abbildung 14: eine Nullstelle für Australien

Hier wird Australien als ein Beispiel. Die rote Linie ist für konstante Breite θ_2^0

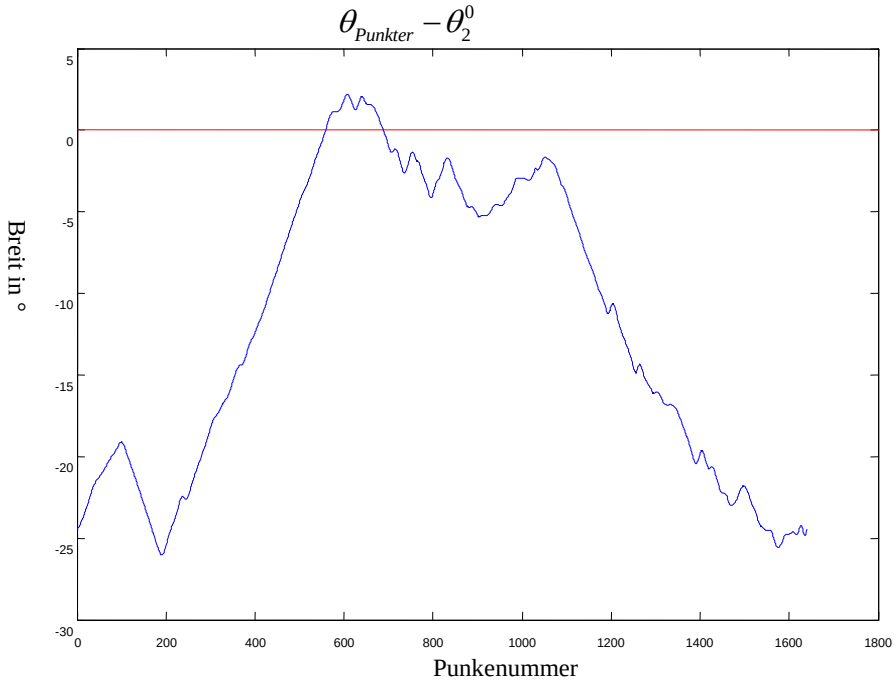


Abbildung 15: ein Nullstellenprofil

Hier werden die Breite der jeder Punkte von Nullstellen reduziert, und finden die geschnittene Punktenummer und deren Länge.
Nach Matlab© Kommando diff(sign()) kann die geschnittenen Punkte deutliche erkennbar werden.

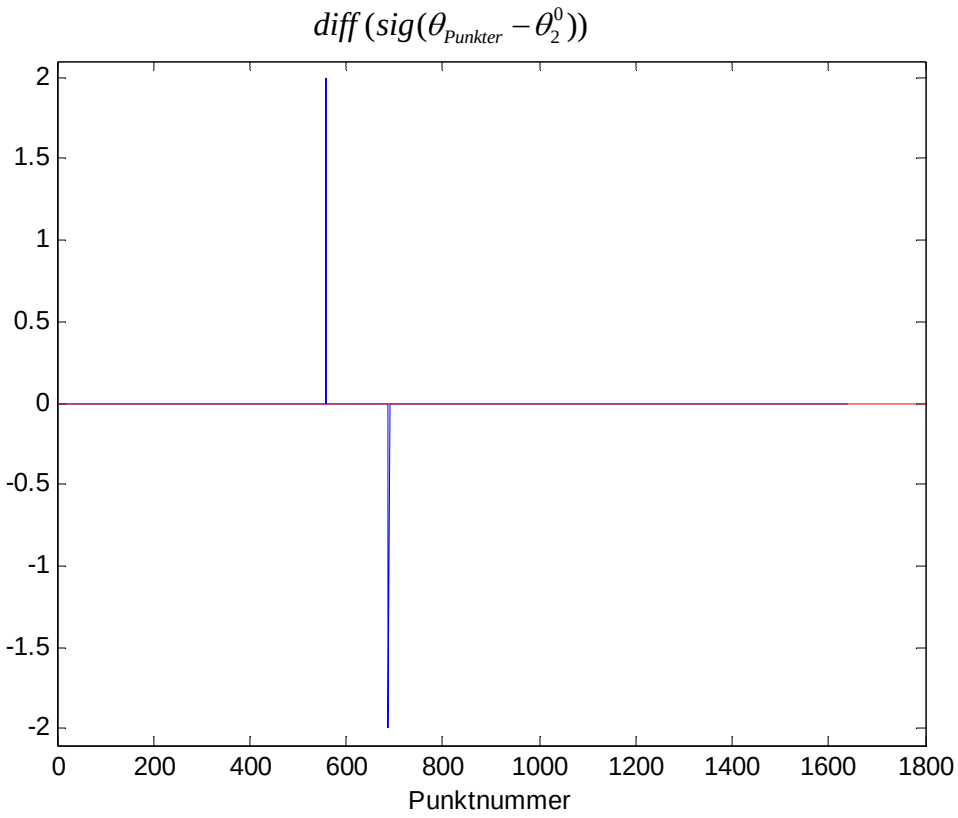
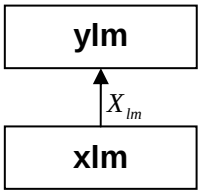


Abbildung 16: nach Verarbeitung des Nullstellenprofils

Wenn der geschnittene Punkt nicht genau ein Grenzpunkt ist, dann wird die Länge des Schnittpunktes durch lineare Interpolation berechnet.

4.4. Kugelflächenfunktionen



4.4.1. **ylm** (Kugelflächenfunktionen)

Die normierten Kugelflächenfunktionen werden in die Programme berechnet.

```
function [Y,theta,phi]=ylm(l,m,theta,phi,check,tol,blox)
```

Eingabe:

l	Grade (0 <= l <= Infinit) [Default: zufällig]
m	Ordnung (-l <= m <= l) [Default: alle -l <= m <= l]
	l und m können Vectors sein, aber können nicht gleichzeitig.
theta	Breite (0 <= theta <= pi) [Default: 181 linear Verteilung]
phi	Länge (0 <= phi <= pi) [Default: 361 linear Verteilung]
check	1 optionale Kontrolle für Normalisierung bei Gauss-Legendre Quadratur
	0 keine Kontrolle
tol	Tolerante für optionale Normalisierungskontrolle.
blox	0 Standard (lm) Ordnung, l=0:L, m=-l:l
	1 Block-diagonal Ordnung, m=-L:L, l=abs(m):L

Ausgabe:

Y	der Matrix der Normierten Kugelflächenfunktionen mit Dimension von
	length(theta) x length(phi) x max(length(m),length(l)) oder
	(L+1)^2 x (length(theta)*length(phi))
	Wenn Sie Grade[0 L] und Ordnung [] eingeben: m=-L:L,
theta	die Breite, wo die Kugelflächenfunktionen berechnet werden.
phi	die Länge, wo die Kugelflächenfunktionen berechnet werden.

Nach der unteren Formel werden die Normierten Kugelflächenfunktionen berechnet.

$$Y_{lm}(\theta,\varphi)=\begin{cases}\sqrt{2}X_{lm}(\theta)\cos m\varphi & \text{if } -l\le m<0\\ X_{l0}(\theta) & \text{if } m=0\\ \sqrt{2}X_{lm}(\theta)\sin m\varphi & \text{if } 0<m\le l\end{cases}$$

4.4.2. **xlm** (zugehörig normierten Legendre Funktionen)

Die zugehörig normierten Legendre Funktionen werden in die Programme berechnet.

function [X,theta]=xlm(l,m,theta,check,tol,blox)

Eingabe:

l	Grade (0 <= l <= Infinit) [Default: random]
m	Ordnung (-l <= m <= l) [Default: aller -l <= m <= l]
theta	Breite (0 <= theta <= pi) [Default: 181 linear Verteilung]
check	1 optionale Kontrolle für Normalisierung bei Gauss-Legendre Quadratur
	0 keine Kontrolle
tol	Tolerante für optionale Normalisierungskontrolle.
blox	0 Standard (lm) Ordnung, l=0:L, m=-l:l

1 Block-diagonal Ordnung, m=-L:L, l=abs(m):L

Ausgabe:

X die zugehörig normierten Legendre Funktionen

theta die Breite, wo die zugehörig normalisiere Legendre Funktionen berechnet werden.

Formel:

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

$$P_{lm}(\cos\varphi) = \frac{1}{2^l l!}(1-\cos\varphi^2)^{m/2} \left(\frac{d}{d(\cos\varphi)}\right)^{l+m} (\cos\varphi^2 - 1)^l$$

5. Quellen- und Literaturverzeichnis

[1] Simons F.J., Dahlen F.A., and Wieczorek M.A. (2006): Spationspectral concentration on a sphere. SIAM Review, 48(3), (2006), pp. 504-536

[2] Homepage von Simons F.J. in Princeton University (2006), http://geoweb.princeton.edu/people/simons/software.html

[3] O. Baur and N. Sneeuw (2006): The slepian approach revisited: Dealing with the polar gap in satellite based geopotential recovery.

[4] Deutsches Zentrum für Luft und Raumfahrt (2007): GRACE _ Gravity Recovery and Climate Experiment. http://www.dlr.de/grace (01. 05. 2007)

[5] GeoForschungsZentrum Potsdam (2006): The GRACE Mission. http://www.gfz-potsdam.de/pb1/op/grace/index_grace.html

[6] A. Albertella, N. Sneeuw (1999);The analysis of gradiometric data with Slepian functions, Physics and Chemistry of the Earth, MS No.: G12-00

6. Programme

6.1. Slepian

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%                  Plotten der Slepianfunktionen                       %%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%


clear all;
close all;
clc
format long g;
load geoid;


% TH       Angular extent of the spherical cap, in degrees OR
%          'england', 'eurasia',  'namerica', 'australia', 'greenland'
%          'africa', 'samerica', 'amazon', 'orinoco'
% L        Bandwidth (maximum angular degree)

defval('TH','australia')
defval('L',30)
defval('sord',10)
defval('blox',0)

% Einlesen der absteigend sortierten Eigenwerte
[Glma,V,EL,EM]=glmalpha(TH,L,sord,blox);

Ylm=ylm([0 L],[],[],[],blox);

XY=eval(sprintf('%s(%i)',TH,sord));

figure;

BigL=(L+1)^2;
g_slepian=repmat(NaN,181,360);

i=1;

while i<=12

    for lambda_d = 1:360

        if lambda_d <= 180
            lambda_g=180+lambda_d;
        else
            lambda_g=lambda_d-180;
        end

        for phi_d=1:181
            g_slepian(phi_d,lambda_g)=Glma(:,i)'*Ylm(:,182-phi_d+(lambda_d-
1)*181);
        end
    end


    % Berechnung der minimalen und maximalen Werte aller N
```

```
    colmin = min(min(g_slepian) );
    colmax = max(max(g_slepian));

    %Berechnung der Potenz zur Grenzfindung
    tmpi = fix(abs(log10(abs(colmin))))+1;
    tmpx = fix(abs(log10(abs(colmax))))+1;

    cmi = (fix(abs(colmin)*10^tmpi)/10^tmpi+1/10^tmpi)*(-1);
    cmx = fix(abs(colmax)*10^tmpx)/10^tmpx+1/10^tmpx;

    % Bestimmung der Grenzen fuer die colorbar
    if abs(cmi)>cmx,
        cmx = abs(cmi);
    else
        cmi = -cmx;
    end

    subplot(4,3,i);
    axesm ('eqdcylin','FLatLimit',[-50 0],'FLonLimit',[100 170]);
    plotm(XY(:,2), XY(:,1)-360,'k');
    set(gca,'clim',[cmi cmx]);
    colormap(redblue1(1000));%Übergang der colormap von blau über weiß nach
rot
    %gridm on;
    meshm(g_slepian,geoidlegend,size(g_slepian))
    title(['\lambda=' num2str(roundn(V(i),-4)) ],'fontsize',10);
    axis image;
    i=i+1;
end % Ende Schleife i

saveas(gcf,['Slepian' num2str(L) ],'jpg');
```

6.2. glmalpha

```
function [G,V,EL,EM,N,GM2AL,MTAP,IMTAP]=glmalpha(TH,L,sord,blox,upco,resc)
% [G,V,EL,EM,N,GM2AL,MTAP,IMTAP]=GLMALPHA(TH,L,sord,blox,upco,resc)
%
% Returns an (lm)X(alpha) matrix with spherical harmonic coefficients of
% the BANDLIMITED Slepian functions of the single or double polar cap, or
% of a geographical region of interest. Only in this case are the
% eigenvalues automatically sorted. The matrix G is orthogonal, G'*G is
% the identity.
%
% INPUT:
%
% TH       Angular extent of the spherical cap, in degrees OR
%          'england', 'eurasia',  'namerica', 'australia', 'greenland'
%          'africa', 'samerica', 'amazon', 'orinoco'
% L        Bandwidth (maximum angular degree)
% sord     1 Single polar cap [default]
%          2 Double polar cap
%          N Splining smoothness for geographical regions [default: 10]
%
% The following options are only for axisymmetric polar caps.
%
% blox     0 Standard (lm) ordering, l=0:L, m=-l:l [default]
%          1 Block-diagonal ordering, m=[0 -1 1 -2 2 ... -L L], l=abs(m):L
% upco     +ve fraction of unit radius for upward continuation [default: 0]
%          -ve fraction of unit radius for downward continuation
```

```
% resc     0 Not rescaled [default]
%          1 Rescaled to have a unit integral over the unit sphere
%            (this is only relevant for the down/upward continued functions)
%
% OUTPUT:
%
% G        The unitary matrix of localization coefficients
% V        The eigenvalues in this ordering (not automatically sorted)
% EL       The vector with spherical harmonic degrees as first index of G
% EM       The vector with spherical harmonic orders as first index of G
% N        The Shannon number
% GM2AL    The sum over all orders of the squared coefficients
% MTAP     The order of the eigentapers, if the region is axisymmetric
% IMTAP    The rank withing that particular order of the eigentapers
%
% EXAMPLE:
%
% Note that using ADDMOUT you can get this back to block-diagonal form
% G=glmalpha; [a,b,c,bl]=addmout(18); imagesc(G(bl,:))
% difer(G'*G-eye(size(G)))
%
% SEE ALSO:
%
% ADDMOUT, ADDMON, KERNELC, LOCALIZATION
%
% Last modified by fjsimons-at-alum.mit.edu, 07.06.2006

% Should be able to update this to retain the rank order per m as well as
% the global ordering

defval('TH',30)
defval('L',18)
defval('blox',0)
defval('upco',0)
defval('resc',0)

% Just get the file names here
if upco==0 & resc==0
  if ~isstr(TH) % POLAR CAPS
    defval('sord',1) % SINGLE OR DOUBLE CAP
    fname=fullfile(getenv('IFILES'),'GLMALPHA',...
           sprintf('glmalpha-%i-%i-%i-%i.mat',TH,L,sord,blox));
    % Initialize ordering matrices
    MTAP=repmat(0,1,(L+1)^2);
    IMTAP=repmat(0,1,(L+1)^2);

  else % GEOGRAPHICAL REGIONS
    defval('sord',10) % SPLINING SMOOTHNESS
    fname=fullfile(getenv('IFILES'),'GLMALPHA',...
           sprintf('glmalpha-%s-%i.mat',TH,L));
    defval('MTAP',NaN) % If not, calculate order per taper
    defval('IMTAP',NaN) % And rank ordering within that taper
    defval('xver',0) % For excessive verification of the geographical case
  end
else
  fname='neveravailable';
  defval('xver',0) % For excessive verification of the upco'd case
end

if exist(fname,'file')==2 % & 1==3
  load(fname)
```

```
else
  % Initialize matrices
  G=repmat(0,(L+1)^2,(L+1)^2);
  V=repmat(0,1,(L+1)^2);

  % Find indices into G belonging to the orders
  [EM,EL,mz,blkm]=addmout(L);

  % Find increasing column index; that's how many belong to this order
  alpha=cumsum([1 L+1 gamini(L:-1:1,2)]);

  % For GEOGRAPHICAL REGIONS
  if isstr(TH)
    % Calculates the localization kernel for this domain
    [Klmlmp,XY]=kernelc(L,TH,sord);

    % Calculates the eigenfunctions/values for this localization problem
    [G,V]=eig(Klmlmp);
    [V,isrt]=sort(sum(real(V),1));
    V=fliplr(V);
    G=G(:,fliplr(isrt));

    [a,b,c,d,e,f,ems,els,R1,R2]=addmon(L);
    % This indexes the orders of G back as 0 -101 -2-1012 etc
    G=G(R1,:);
    % Check indexing
    difer(els(R1)-EL)
    difer(ems(R1)-EM)

    % Calculate Shannon number and compare this with the theory
    N=sum(V);

    % Is the Shannon number right?
    difer((L+1)^2*Klmlmp(1)-N)

    % Check if the expansion of a basis function is indeed either 1 or 0
    if xver==1
      disp('Excessive verification')
      % Is the area right?
      difer(Klmlmp(1)-spharea(TH),4)
      % This is a bit double up... but it's only for excessive verification
      [V1,C]=localization(L,TH,sord);
      difer(V-V1')
      for index=1:length(C)
    salpha=G'*C{index}(R2);
    % Only one of these functions should get "hit"
    difer(sum(abs(salpha)>1e-9)-1)
      end
    end
    % For AXISYMMETRIC REGIONS
  else
    if blox~=0 & blox~=1
      error('Specify valid block-sorting option ''blox''')
    end
    % For the SINGLE or DOUBLE POLAR CAPS
    for m=0:L
      % Same results for +/- m
      if sord==1
    [E,Vg,th,C,T,Vp]=grunbaum(TH,L,m,[]);
      elseif sord==2
```

```
[E,Vg,th,C,T,Vp]=grunbaum2(TH,L,m,[]);
  else
error('Specify single or double polar cap')
  end
  if upco~=0
if upco>0
  % The upward continuation matrix
  A=diag((1+upco).^[-(m:L)-1]);
elseif upco<0
  % The downward continuation matrix
  A=diag((1+abs(upco)).^[(m:L)+1]);
end

if xver==1
  % This should give the same result, more or less, less accurate
  if sord==1
    [a,Vs,c,d,Cs,e,f,g,h,j,D]=sdwcap(TH,L,m,0,-1);
  else
    [a,Vs,c,Cs,e,f,D]=sdwcap2(TH,L,m,0,-1);
  end
  % This should give the eigenvalues again, which we'd had from
      % orthocheck
  warning off
  % Check difference integration eigenvalues and those from
      % kernel
  difer(Vp(:)-diag((C'*D*C)./(C'*C)))
  % Check difference integration and diagonalization eigenvalues
  difer(Vp(:)-Vs(:))
  warning on
  Vc=diag((C'*A*C*diag(Vp)*C'*A*C));
  Vp0=Vp;
end

% Upward continuation from 1 to 1+a or from 1+a to 1:
% New eigenfunctions, same name
C=A*C;
% Calculate new eigenvalues, same name
[jk1,jk2,jk3,Vp,nofa]=orthocheck(C,[],TH/180*pi,m,sord,1);

% Make sure these are sorted, since that's not automatically the case
% [Vp,ind]=sort(Vp,'descend');
% C=C(:,ind);
% Current thinking is: do NOT resort, as you'll want to compare the
% best at a=0 with whatever it becomes later!

if xver==1
  warning off
  % Check difference integration eigenvalues and those from kernel
  difer(Vp(:)-diag((C'*D*C)./(C'*C)))
  warning on
  % Check how many Vp>Vp0
  disp(sprintf('%i/%i eigenvalues greater',sum(Vp(:)>Vp0(:)), ...
            length(Vp0)))
  disp(sprintf('Shannon number greater: %i',sum(Vp)>sum(Vp0)))
end
if resc==1
  % Rescale these functions to have an integral to unity over the
  % sphere; note: this doesn't make the set orthonormal of course
  C=C*diag(1./nofa);
end
  end
```

```
      % Distribute this at the right point in the huge matrix
      if m>0
    G(EM==-m,alpha(2*m):alpha(2*m+1)-1)=C;
    V(alpha(2*m):alpha(2*m+1)-1)=Vp;
    MTAP(alpha(2*m):alpha(2*m+1)-1)=-m;
    % It's all neatly ordered here, downgoing within every order
    IMTAP(alpha(2*m):alpha(2*m+1)-1)=1:length(Vp);
      end
      % Duplicate for the other order in case the region is axisymmetric
      G(EM==m,alpha(2*m+1):alpha(2*m+2)-1)=C;
      V(alpha(2*m+1):alpha(2*m+2)-1)=Vp;
      MTAP(alpha(2*m+1):alpha(2*m+2)-1)=m;
      % It's all neatly ordered here, downgoing within every order
      IMTAP(alpha(2*m+1):alpha(2*m+2)-1)=1:length(Vp);;
    end
    % Calculate the Shannon number and compare it to the theory
    N=sum(V);
    difer(N-(L+1)^2*sord*(1-cos(TH/180*pi))/2);

    % Compute the sum over all orders of the squared coefficients
    % Thus works when they have not been blocksorted yet.
    GM2AL=repmat(0,(L+1)^2,L+1);
    for l=0:L
      b=(l-1+1)^2+1;
      e=(l+1)^2;
      GM2AL(:,l+1)=sum(G(b:e,:).^2,1)';
    end
    % Make sure that the sum over all degrees is 1
    difer(sum(GM2AL,2)-1)

    % This is not blockdiagonal, unless you make it thus
    if blox==1
      G=G(blkm,:);
      EM=EM(blkm);
      EL=EL(blkm);
    end
  end
  % Save the results
  eval(sprintf('save %s G V EL EM N',fname))
%      'save %s G V EL EM N GM2AL',fname))

end
```

6.3. kernelc

```
function [Klmlmp,XY]=kernelc(Lmax,dom,pars,ngl)
% [Klmlmp,XY]=KERNELC(Lmax,dom,pars,ngl)
%
% Calculation of the localization matrix for some domain on the sphere.
% NOT FOR POLAR PATCHES! AND NOT GOOD FOR NEAR-POLAR PATCHES!
% NOT FOR REGIONS CONTAINING THE NORTH POLE OR THE SOUTH POLE!
%
% INPUT:
%
% Lmax       Maximum angular degree (bandwidth)
% dom        'patch'   Spherical patch [default], with specs in 'pars'
%                      Note: better use GRUNBAUM / PLM2ROT in this case
%            'england', 'eurasia',  'namerica', 'australia', 'greenland'
```

```
%            'africa', 'samerica', 'amazon', 'orinoco', with specs in
'pars'
% pars       [th0,ph0,thR] for 'patch'
%                 th0  Colatitude of the cap center, in radians
%                 ph0  Longitude of the cap center, in radians
%                 thR  Radius of the cap, in radians
%            N  splining smoothness for geographical regions [default: 10]
% ngl        The degree of the Gauss-Legendre integration [default: 200]
%
% OUTPUT:
%
% Klmlmp     The localization kernel whose eigenfunctions we want,
%            indexed as: degree  [0  1  1  1  2  2  2  2  2]
%                        order   [0  0 -1  1  0 -1  1 -2  2]
%            The function LOCALIZATION later reindexes this in LMCOSI
%            fashion. Note you can use ADDMOUT and ADDMON to modify.
% XY         The outlines of the region into which you are localizing
%
%
% EXAMPLE:
%
% L=18;
% [Klmlmp,XY]=kernelc(L,'australia');
%
% See also LOCALIZATION, SDWREGIONS, GLMALPHA, DLMLMP
%
% Last modified by fjsimons-at-alum.mit.edu, 10/22/2006

t0=clock;

defval('Lmax',18);
defval('dom','patch')
defval('ngl',200)

fnpl=sprintf('%s/WREG-%s-%i.mat',...
         fullfile(getenv('IFILES'),'KERNELC'),dom,Lmax);

if exist(fnpl,'file')==2 & ~strcmp(dom,'patch')
  eval(sprintf('load %s',fnpl))
  disp(sprintf('%s loaded by KERNELC',fnpl))
else
  if strcmp(dom,'patch')
    defval('pars',[90 75 30]*pi/180);
    % For future reference
    th0=pars(1); ph0=pars(2); thR=pars(3);
    if th0==0
       error('Not for polar caps! Use GRUNBAUM or SDWCAP instead')
    end
    if thR>th0
       error('Not for near-polar caps! Use GRUNBAUM, SDWCAP, then rotate')
    end
    % Convert all angles to degrees for CAPLOC only
    [lon,lat]=caploc([ph0 pi/2-th0]/pi*180,thR/pi*180,100,1);
    % Northern and Southern points, in radians
    thN=(th0-thR);
    thS=(th0+thR);
    XY=[lon lat];
  else
    defval('pars',10);
    XY=eval(sprintf('%s(%i)',dom,pars));
    thN=90-max(XY(:,2)); thN=thN*pi/180;
```

```
  thS=90-min(XY(:,2)); thS=thS*pi/180;
end

% No more set-up after this point
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Introduce and dimensionalize variables and arrays
[dems,dels,mz,lmc,mzin,mzout,bigm,bigl]=addmon(Lmax);
dimK=(Lmax+1)^2; lenm=length(dems);
Klmlmp=repmat(NaN,dimK,dimK);

% Calculating different Gauss-Legendre points for all possible product
% degrees is not a good idea since they get multiplied by more
% functions of theta
intv=cos([thS thN]);
nGL=max(ngl,2*Lmax);
[w,x,N]=gausslegendrecof(nGL,[],intv);
disp(sprintf('%i Gauss-Legendre points and weights calculated',N))

% First calculate the Legendre functions themselves
% Note that dimK==sum(dubs)
dubs=repmat(2,lenm,1); dubs(mz)=1; comdi=[];
% First, calculate all the Legendre functions themselves
Xlm=repmat(NaN,length(x),lenm);

% Calculate the Legendre polynomials
ind=0;
for l=0:Lmax
  Xlm(:,ind+1:ind+l+1)=[legendre(l,x(:)','sch')*sqrt(2*l+1)]';
  ind=ind+l+1;
end

% Calculate the Legendre products for all combinations of l and m
XlmXlmp=repmat(NaN,length(x),((lenm^2)+lenm)/2);
GLint=repmat(NaN,((lenm^2)+lenm)/2,1);
index=0;
h=waitbar(0,'Calculating all Legendre products');
for lm1dex=1:lenm
  l1=dels(lm1dex);
  m1=dems(lm1dex);
  pos1=1/2*(l1)^2+1/2*l1+m1+1;
  comdi=[comdi ; dubs(lm1dex:lenm)];
  % Note that the last index will be ((lenm^2)+lenm)/2
  for lm2dex=lm1dex:lenm
    l2=dels(lm2dex);
    m2=dems(lm2dex);
    index=index+1;
    pos2=1/2*(l2)^2+1/2*l2+m2+1;
    % Calculate products of Legendre functions
    XlmXlmp(:,index)=Xlm(:,pos1).*Xlm(:,pos2);
    waitbar(2*index/((lenm^2)+lenm),h);
  end
end
delete(h)
disp('Legendre products calculated')

% In our ordering, the -1 precedes 1 and stands for the cosine term
comdex=[1:((lenm^2)+lenm)/2]';
coss=gamini(comdex,comdi);
% Need a vector of length "index" that points to the right combination in
% XlmXlmp for the next array we are designing. First, find the positions
% we've been missing
```

```
h=[dimK:-1:1']'; k=find(dems); kk=k+[1:length(k)]';
% Where to insert other elements
inpo=[indeks(cumsum(skip(h,kk)),k)+1]';
% How many elements to insert
inel=h(kk);
% Which elements to insert
beg=inpo-h(k)+[1:length(inel)]';
ent=inpo-h(k)+inel+[0:length(inel)-1]';
ins=[];
for ind=1:length(beg)
  ins=[ins coss(beg(ind):ent(ind))];
end
% And how to do it
bigo=insert(coss,ins,gamini(inpo,inel));

% Get the integration info for the domain
switch dom
 case 'patch'
  % Get the parameters of the dom
  phint=dphpatch(acos(x),thR,th0,ph0);
 otherwise
  defval('Nk',10);
  % Now we may have multiple pairs
  phint=dphregion(acos(x)*180/pi,Nk,dom);
  phint=phint*pi/180;
end

% The number of elements that will be calculated is
nel=(dimK^2+dimK)/2;
% Calculate matrix elements
index=0;
undex=0;
andex=1;
h=waitbar(0,'Evaluating all integrals and assembling matrix');
for lm1dex=1:dimK
  l1=bigl(lm1dex);
  m1=bigm(lm1dex);
  index=index+1;
  ondex=0;
  I=repmat(NaN,length(x),dimK-lm1dex+1);
  for lm2dex=lm1dex:dimK
    l2=bigl(lm2dex);
    m2=bigm(lm2dex);
    ondex=ondex+1;
    undex=undex+1;
    waitbar(undex/nel,h);
    % Now evaluate the longitudinal integrals at the GL points
    if m1>0 & m2>0
  I(:,ondex)=sinsin(acos(x),m1,m2,phint);
    elseif m1<=0 & m2<=0
  I(:,ondex)=coscos(acos(x),m1,m2,phint);
    elseif m1>0 & m2<=0
  I(:,ondex)=sincos(acos(x),m1,m2,phint,pars);
    elseif m1<=0 & m2>0
  I(:,ondex)=sincos(acos(x),m2,m1,phint);
    end
  end
  % Calculate the entire integral and distribute over the kernel
  Klmlmp(index,lm1dex:dimK)=...
  (w(:)'*(XlmXlmp(:,bigo(andex:undex)).*I));
```

```
    % And symmetrize them
    Klmlmp(lm1dex+1:dimK,index)=Klmlmp(index,lm1dex+1:dimK)';
    andex=undex+1;

    % Verify right away that the first value correctly gives the area of
    % the patch
    if lm1dex==1
      if strcmp(dom,'patch')
    apo=abs(2*pi*(1-cos(thR))-Klmlmp(1))/(2*pi*(1-cos(thR)));
    disp(sprintf('Area of the patch approximated to within %5.2f %s',...
             apo*100,'%'))
    if apo*100>1;
      error('Something wrong with the area element: radians/degrees ?')
    end
      else
    disp(sprintf('Area of the domain approximated as %8.3e',...
             Klmlmp(1)))
      end
    end
  end

  % To make this exactly equivalent to Tony's \ylm:
  Klmlmp=Klmlmp/4/pi;

  % This then makes Klmlmp(1) the fractional area on the sphere

  delete(h)
  disp(sprintf('KERNELC (Matrix)  took %8.4f s',etime(clock,t0)))
  % Save this now
  if ~strcmp(dom,'patch')
    eval(sprintf('save %s Lmax Klmlmp dom ngl XY',fnpl))
  end
end
```

6.4. Plotcont

```
function varargout=plotcont(c11,cmn,res,ofs)
% [axlim,handl,XYZ]=PLOTCONT(c11,cmn,res,ofs)
%
% INPUT:
%
% c11   : [x,y]/[lon,lat] of upper left node
% cmn   : [x,y]/[lon,lat] of bottom right node
% res   : 0 regular coastlines (default)
%       : 1 (slightly) higher-resolution coastlines and lakes, too
%       : 2 Global Mollweide projection
%       : 3 Three-dimensional coordinates (to return)
%       : 4 Three-dimensional coordinates, Northern hermisphere only
%       : 5 High-resolution coast line of California (including Baja)
%       : 6 Simple restriction to Australia
% ofs   : Longitude offset, e.g. 360 degrees [only for 0,1,5,6]
%
% Longitude ranges from 0 to 360; plots are centered on the PACIFIC.
%
% OUTPUT:
%
% axlim  : axis that fits snugly around the data.
% handl  : handle to the plotted line, and for Mollweide,
%          also the handle to the box around it
% XYZ    : the actual data points plotted (2D or 3D)
```

```
%
% AUSTRALIA:
% plotcont([90 10],[180 -60])
%
% WORLD:
% plotcont([0 90],[360 -90])
%
% Saved matrix as space-saving unsigned integer
% - but that translates the NaN's into some  high number - take that out.
% Note how A==NaN does not return the NaN's!
%
% You'll have to maintain your own databases of continental outlines!
%
% Last modified by fjsimons-at-alum.mit.edu, 18.05.2006

defval('res',0)
defval('ofs',0)

switch res
  case 5
   % Overrride map for California
   defval('c11',[235.4320 43]);
   defval('cmn',[252.9990 22.8699]);
 case 6
  defval('c11',[90 10])
  defval('cmn',[180 -60])
  res=1;
 otherwise
  defval('c11',[0 90])
  defval('cmn',[360 -90])
end

% Where are the data kept? Create a directory $IFILES/COASTS
% with $IFILES set as an environment variable...
defval('ddir',fullfile(getenv('IFILES'),'COASTS'))

% Load the data sets
switch res
 case 0
  fid=fopen(fullfile(ddir,'cont.mtl'),'r','b');
  cont=fread(fid,[5217 2],'uint16');
  fclose(fid);
 case {1,3,4}
  fid=fopen(fullfile(ddir,'cost.mtl'),'r','b');
  cont=fread(fid,[9598 2],'uint16');
  fclose(fid);
 case 2
  load(fullfile(ddir,'conm'))
  hold on
  handl{1}=plot(conxm,conym,'k');
  axlim=[];
  handl{2}=plot(xbox,ybox,'k');
  XYZ=[conxm conym];
 case 5
  cont=load(fullfile(ddir,'California'));
  cont=cont.cont;
end

% Recast data in good form
switch res
  case {0,1,3,4}
```

```
   cont=cont/100-90;
   cont(cont==max(max(cont)))=NaN;
end

switch res
 case {0,1,5}
  % Restrict to the map requested
  lon=cont(:,1); lat=cont(:,2);
  lon(~(lon>=c11(1) & lon<=cmn(1)))=NaN;
  lat(~(lat<=c11(2) & lat>=cmn(2)))=NaN;
  % Plot these continental outlines
  hold on
  handl=plot(lon+ofs,lat,'k','Linewidth',1);
  axlim=[min(lon([~isnan(lon) & ~isnan(lat)]))...
     max(lon([~isnan(lon) & ~isnan(lat)]))...
     min(lat([~isnan(lon) & ~isnan(lat)]))...
     max(lat([~isnan(lon) & ~isnan(lat)]))];
  XYZ=[lon+ofs lat];
 case {3,4}
  % Convert to spherical coordinates
  lon=cont(:,1)/180*pi;
  lat=cont(:,2)/180*pi;
  rad=repmat(1.001,size(lat));
  [xx,yy,zz]=sph2cart(lon,lat,rad);
  XYZ=[xx yy zz];
  if res==4
    XYZ=XYZ(zz>0,:);
  end
  % Now need to take out the annoying connecting lines
  % Distance between two consecutive points
  xx=XYZ(:,1); yy=XYZ(:,2); zz=XYZ(:,3);
  d=sqrt((xx(2:end)-xx(1:end-1)).^2+(yy(2:end)-yy(1:end-1)).^2);
  % D-level: minimum distance to lift pen... variable
  dlev=3;
  p=find(d>dlev*nanmean(d));
  % Now right after each of these positions need to insert a NaN;
  nx=insert(xx,NaN,p+1);
  ny=insert(yy,NaN,p+1);
  nz=insert(zz,NaN,p+1);
  XYZ=[nx(:) ny(:) nz(:)];

  % And plot this, too
  handl=plot3(XYZ(:,1),XYZ(:,2),XYZ(:,3),'k');
  axlim=[];
end

% Generate output
vars={'axlim' 'handl' 'XYZ'};
for index=1:nargout
  varargout{index}=eval(vars{index});
end
axis equal
hold off
```

6.5. Australia

```
function varargout=australia(res,scal)
% XY=AUSTRALIA(res,scal)
% AUSTRALIA(...) % Only makes a plot
%
% Finds the coordinates of Australia
```

```
%
% INPUT:
%
% res      0 The standard, default values
%          N Splined values at N times the resolution
% scal     Scale this to something else
%
% OUTPUT:
%
% XY       Closed-curved coordinates of the continent
%
% Last modified by fjsimons-at-alum.mit.edu, June 3rd, 2004

defval('res',0)
defval('scal',0)
if res==0
  fnpl=fullfile(getenv('IFILES'),'COASTS','Australia.mat');
else
  fnpl=fullfile(getenv('IFILES'),'COASTS',sprintf('Australia-%i.mat',res));
end

if exist(fnpl,'file')==2
  load(fnpl)
  if scal~=0
    XY(:,1)=scale(XY(:,1),[200 330]);
    XY(:,2)=scale(XY(:,2),[46 -70]);
  end
  if nargout==0
    plot(XY(:,1),XY(:,2),'k-'); axis equal; grid on
  else
    varargout{1}=XY;
  end
else
  if res==0
    % First part %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
    c11=[112 -10.5];
    cmn=[154 -39];
    clf
    [axlim,handl,XY]=plotcont(c11,cmn,[],360);
    delete(handl)
    % Get rid of common NaNs
    XY=XY(~isnan(XY(:,1)) & ~isnan(XY(:,2)),:);

    % Now make sure the distances aren't huge
    xx=XY(:,1); yy=XY(:,2);
    d=sqrt((xx(2:end)-xx(1:end-1)).^2+(yy(2:end)-yy(1:end-1)).^2);
    dlev=3;
    p=find(d>dlev*nanmean(d));
    nx=insert(xx,NaN,p+1);
    ny=insert(yy,NaN,p+1);
    XY=[nx(:) ny(:)];

    % Experiment with
    % a=[16:271];
    % plot(XY(a,1),XY(a,2));
    XY=XY(2:end,:);
    % Experiment with -> See "check this out"

    % Form closed contour
    XY=XY(~isnan(XY(:,1)) & ~isnan(XY(:,2)),:);
```

```
    plot(XY(:,1)-360,XY(:,2),'LineW',2,'Color','k');

    axis equal
%     axis([110 160 -40 -10])

    % Check this out %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
    hold on
    for index=1:length(XY)
      pm(index)=plot(XY(index,1)-360,XY(index,2),'o');
      set(pm(index),'MarkerE','k','MarkerF',[1 1 1]*(index-1)/length(XY))
      title(num2str(index))
      pause(0.02)
    end
    % Check this out %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
    eval(sprintf('save %s XY',fnpl))
  else
    XY=australia(0);
    XYb=bezier(XY,res);
    XY=XYb;
    eval(sprintf('save %s XY',fnpl))
  end
end
```

6.6. bezier

```
function XYb=bezier(XY,N)
% XYb=BEZIER(XY)
%
% An attempt at smoothing a coastline by B-spline fitting.
%
% INPUT:
%
% XY        The set of points, make sure XY(end,:)=XY(1,:)
% N         Number of times this needs to be smoothed
%
% http://www.me.cmu.edu/faculty1/shimada/gm98/project/ivan/project/
% Fujio Yamaguci "Curves and Surfaces in Computer Aided Geometric
% Design", Springer-Verlag, Berlin, 1988
% http://mathworld.wolfram.com/CubicSpline.html
%
% Last modified by fjsimons-at-alum.mit.edu, June 4rd, 2004

n=size(XY,1);

% Calculate cumulative distance between these points
% This is the "knot vector"
t=cumsum([0 ; grcdist(XY(1:end-1,:),XY(2:end,:))]);
t=t/t(n);

% New, finer, linear distance between the points
tb=linspace(0,1,n*N);

XYb=spline(t,XY',tb)';
```

6.7. gausslegendrecof

```
function [w,x,N]=gausslegendrecof(l,method,intv)
% [w,x,N]=gausslegendrecof(l,method,intv)
%
% Weights w and abscissas x for Gauss-Legendre integration of a
% polynomial function f(x) of degree l. The approximation w(:)'*f(x(:))
% to the integral W(x)f(x), W(x)=1 will be exact on [-1 1] and need only
```

```
% be rescaled to the desired interval. If intv is specified,
% w(:)'*f(x(:)) will do, as we do the scaling for you.
%
% INPUT:
%
% l           Degree of polynomial in the integrand
% method      'jacobi' stable algorithm (Legendre roots only) [default]
%             'cofrec' algorithm for l<40 (roots and coefficients)
% intv        If integration interval specified, returns scaled [w,x]
%             This may be an Nx2 matrix if you want multiple intervals
%
% OUTPUT:
%
% w           Weights (vector, or matrix in case of more than one TH)
% x           Abscissas (roots of Legendre polynomial of degree N)
% N           Number of points in the integration (length(x))
%
% The abcissa's are the roots of a Legendre polynomials defined on the
% same interval. In particular then, this routine can be used to
% integrate (products) of Legendre functions themselves, which is useful
% in the analysis of spherical harmonics. This returns the N-point
% Gauss-Legendre integration points and weights, which is accurate for
% polynomials up to degree 2*N-1. Note that these things are symmetric so
% for the N-point integration there are only ceil(N/2) unique
% weights. The weights are positive, symmetric, and should sum to 2.
%
% EXAMPLE: (from Numerical Recipes [qgaus, p. 141, Chapter 4.5])
%
% [w,x]=gausslegendrecof(19);
%
% Last modified by fjsimons-at-alum.mit.edu, 09/28/2006

defval('method','jacobi')
defval('intv',[])

% Figure out what the N will need to be
N=ceil((l+1)/2);

switch method
 case 'jacobi'
  % Find roots of Legendre polynomial
  x=legendrecof(N,method);
  % Find weight values from the derivative of the Legendre polynomial
  % Calculate Legendre function at N-1; note Pl(1)=1, which is not
  % the normalization we want in order to make the inner product of the
  % spherical harmonic be 4 pi! Now, for m=0, the inner product is
  % 2/(2N+1). These belong to the standard, listed Legendre polynomials.
  Pl=rindeks(legendre(N-1,x,'sch'),1);
  Pdiff=-N*Pl(:)./(x(:).^2-1);
 case 'cofrec'
  % Find roots of Legendre polynomial of degree N in [-1 1]
  % These are the abscissa points
  [x,C1]=legendrecof(N,method);
  % Find the value of the derivative of the Legendre function at these
  % points, by involving the lower-degree polynomial as well
  [y,C2]=legendrecof(N-1,method);
  x=sort(x);
  % This is the equation for the derivative of the Legendre polynomial
  % as found in Numerical Recipes and other texts. Wolfram has this
  % equation under Legendre-Gauss Quadrature, not under Legendre
Polynomials.
```

```
  % Pdiff=N*(x.*polyval(C1,x)-polyval(C2,x))./(x.^2-1);
  % But we are noting that we are evaluating this at the roots of C1
  % so we can save one computation
  % Note we are using coefficients of Legendre Polynomials that integrate
  % to 2/(2L+1), i.e. the "classical ones"
  Pdiff=-N*polyval(C2,x)./(x.^2-1);
end

% Find weights
% Numerical Recipes, Equation (4.5.16)
w=2./(1-x.^2)./Pdiff.^2;

% Scale to interval
if ~isempty(intv)
  if length(intv(:))==2
    a=intv(1);
    b=intv(2);
    % Rescale nodes
    x=a+(x+1)/2*(b-a);
    % Rescale weights
    w=w*(b-a)/2;
  else
    a=intv(:,1);
    b=intv(:,2);
    xg=x;wg=w;
    x=repmat(NaN,length(xg),length(a));
    w=repmat(NaN,length(wg),length(a));
    for index=1:length(a)
      x(:,index)=a(index)+(xg+1)/2*(b(index)-a(index));
      w(:,index)=wg*(b(index)-a(index))/2;
    end
  end
end
```

6.8. legendreroots

```
function [r,J]=legendreroots(N);
% [r,J]=LEGENDREROOTS(N);
%
% The function r = legendreroots(N) computes the roots of the
% Legendre polynomial of degree N.
% Also returns the Jacobi matrix whose eigenvalues they are.
%
% J.A.C. Weideman, S.C. Reddy 1998.
%
% fjsimons-at-alum.mit.edu is still wondering if you can use the
% Jacobi matrix to back out the coefficients of the Polynomials.

n = [1:N-1];                    %  Indices
d = n./sqrt(4*n.^2-1);          %  Create subdiagonals
J = diag(d,1)+diag(d,-1);       %  Create Jacobi matrix
r = sort(eig(sparse(J)));       %  Compute eigenvalues
```

6.9. dphregion

```
function varargout=dphregion(th,N,region)
% [phint,thp,php]=DPHREGION(th,N,region)
%
% Finds the longitude crossings at a given colatitude
% of the coastlines of some region
%
% INPUT:
%
```

```
% th         Colatitudes where you want it evaluated, in degrees
% N          Smoothness of the spline version
% region     A string: 'england', 'africa', 'namerica', 'greenland',
%                      'australia', 'eurasia', or
%            A matrix with XY/[lon,lat] coordinates
%
% OUTPUT:
%
% phint      Longitudinal interval(s) defining the integration domain
% thp        Colatitude matrix for plotting
% php        Longitude matrix for plotting
%
% EXAMPLE:
%
% dphregion('demo1',[],'africa')
% dphregion('demo2',[],'eurasia')
% dphregion('demo2',10)
%
% Last modified by fjsimons-at-alum.mit.edu, June 4th, 2004

if ~isstr(th)
  % First, get the coordinates of region; these are sorted
  defval('N',10);
  if isstr(region)
    XY=eval(sprintf('%s(%i)',region,N));
    % Now in colatitude
    XY(:,2)=90-XY(:,2);
  else
    XY=region;
  end

  % Calculate the crossings of XY at th
  [phint,thp,php]=phicurve([XY(:,2) XY(:,1)],th);

  % Distribute output
  varn={'phint','thp','php'};
  for index=1:nargout
    varargout{index}=eval(varn{index});
  end
  % AFTER THIS NOTHING BUT DEMOS %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
elseif strcmp(th,'demo1')
  defval('N',10)
  defval('region','africa')
  eval(sprintf('%s(%i)',region,N));
  eval(sprintf('XY=%s(%i);',region,N));
  thN=90-max(XY(:,2));
  thS=90-min(XY(:,2));
  XY(:,2)=90-XY(:,2);
  TH=thN+rand*(thS-thN);
  hold on
  XL=xlim;
  [phint,thp,php]=phicurve([XY(:,2) XY(:,1)],TH);
  pl=plot(php,90-thp,'g-','LineW',2);
  plot(XL,[90-TH 90-TH],'k--')
  p=plot(phint,90-TH,'o');
  set(p,'MarkerE','k','MarkerF','y')
  t=title(num2str(length(phint)),'FontS',30);
  hold off
  if nargout==1
    phint=dphi;
  end
```

```
elseif strcmp(th,'demo2')
  % Plot this for the time being
  defval('region','africa')
  defval('N',200)
  ah(2)=subplot(212);
  ah(1)=subplot(211);
  Nk=100;
  eval(sprintf('%s(%i)',region,Nk));
  eval(sprintf('XY=%s(%i);',region,Nk));
  thN=90-max(XY(:,2));
  thS=90-min(XY(:,2));
  XY(:,2)=90-XY(:,2);
  thetas=linspace(thN,thS,N);
  hold on; p=[];
  XL=xlim;
  for ind=1:length(thetas)
    TH=thetas(ind);
    l=plot(XL,[90-TH 90-TH],'k-');
    % Add eps so it is never zero, essentially
    xmth=XY(:,2)-TH;
    dsx=diff(sign(xmth));
    % Now it can be the one before, or after the crossing, how about
    colf=find(dsx);
    % This now returns the one on the negative side of the line
    colx=colf+(dsx(colf)==-2);
    colx2=colx-(dsx(colf)==-2)+(dsx(colf)==2);
    L(ind)=length(colx);
    if mod(L(ind),2)
      warning(sprintf('Cannot find pairs of crossings at th=%8.3f',TH))
    end
    axes(ah(2))
    plot(XY(:,1),xmth,'-+'); hold on; grid on
    xls=[min(XY(colx,1))-range(XY(colx,1))/3 ...
     max(XY(colx,1))+range(XY(colx,1))/3];
    % We want the interpolated point at which it becomes exactly zero
    xlim(xls)
    plot(xls,[0 0],'k')
    ylim([-1 1]/100)
    % Then one point was exactly hit, this is the thN case
    if all(colx==colx2)
      dph=XY([colx(2) colx2(2)]); % This works
    else
      for ond=1:L(ind)
    dph(ond)=interp1(xmth([colx(ond) colx2(ond)]),...
             XY([colx(ond) colx2(ond)],1),0,'linear');
      end
    end
    plot(XY(colx,1),xmth(colx),'bs');
    plot(XY(colx2,1),xmth(colx2),'rv');
    plot(dph,repmat(0,size(dph)),'g*');
    clear dph
    hold off
    for indo=1:length(colx)
      axes(ah(1))
      p(indo)=plot(XY(colx(indo),1),90-XY(colx(indo),2),'o');
    end
    if ~isempty(p)
      set(p,'MarkerE','k','MarkerF','y')
      % t=text(362,58,num2str(L(ind)),'FontS',30);
      t=title(num2str(L(ind)),'FontS',30);
      pause(0.05)
      delete([l p t]); p=[];
```

```
    end
  end
  hold off
  disp(sprintf('Number of uneven crosses %i',sum(mod(L,2))))
else
  error('Speficy valid demo')
end
```

6.10. phicurve

```
function varargout=phicurve(thph,th)
% [phint,thp,php]=PHICURVE(thph,th)
%
% Finds the longitude crossings and (thus) integration domains of a
% closed curve parameterized in colatitude/longitude space at certain
% query points of colatitude. Note that this is a hatching tool.
%
% INPUT:
%
% thph            Colatitude/Longitude of the closed curve
% th              Colatitude at which crossings are required
%
% OUTPUT:
%
% phint           A matrix with crossings/intervals and zeroes, of
%                 dimensions MxN where M=length(th) and N can be anything
%                 depending on the number of oscillations of the curve
% thp             Colatitude matrix for plotting, if possible
% php             Longitude matrix for plotting, if possible
%
% EXAMPLE:
%
% phicurve('demo1','africa') % A geographic region
% phicurve('demo2') % A random blob
%
% Last modified by fjsimons-at-alum.mit.edu, June 3rd, 2004

if ~isstr(thph)
  % For every th, find the relevant phint
  xmth=repmat(thph(:,1),1,length(th))-repmat(th(:)',length(thph(:,1)),1);
  dsx=diff(sign(xmth));
  if sum((dsx(:)==0)-1)==0
    error('Specify at least one colatitude within the data range')
  end
  % Now it can be the one before, or after the crossing, how about
  [colf,colj]=find(dsx);
  colr=sub2ind(size(dsx),colf,colj);
  % This now returns the one on the negative side of the line
  colx=colf+(dsx(colr)==-2);
  colx2=colx-(dsx(colr)==-2)+(dsx(colr)==2);
  L=length(colx);
  if mod(L,2)
    error(sprintf('Cannot find pairs of crossings'))
  end
  % Then one point was exactly hit, this is the thN or thS case
  if length(colx)==2 & all(colx==colx2)
    phint=thph([colx(2) colx2(2)],2);
    thp=[th th];
    php=phint;
  else
    for ond=1:L
      % In case you have a node immediate followed by a crossing
```

```
      if colx(ond)==colx2(ond)
    phint(ond)=NaN;
      else
    phint(ond)=interp1(xmth([colx(ond) colx2(ond)],colj(ond)),...
               thph([colx(ond) colx2(ond)],2),0,'linear');
      end
    end
    % Debate whether this is useful or not wrt to node/crossing
    %    phint=phint(~isnan(phint));
    % ACTUALLY, IF THE NAN'S ARE NOT CONSECUTIVE PAIRS GET SPECIAL CASE

    % Now rearrange back to the number of requested points
    % But there could be points with more or less than 2 crossings
    % Maximum number of times a crossing is repeated
    [a,b]=degamini(colj);
    rowj=colj;
    colj=matranges(reshape([repmat(1,length(b),1) b']',length(b)*2,1))';
    pint=repmat(NaN,length(th),max(b));
    subsi=(colj-1)*length(th)+rowj;
    pint(subsi)=phint;
    if length(b)==length(th)
      wt=0;
      thp=reshape(gamini(th,b),2,length(phint)/2);
    else
      wt=1;
      thp=[];
    end
    % Need to sort since contour may be given in any order
    phint=sort(pint,2);
    if wt==0
      php=reshape(phint(subsi),2,length(colj)/2);
    else
      php=[];
    end
    % Make them zero so the integral doesn't do anything
    phint(isnan(phint))=0;
  end
  varn={'phint','thp','php'};
  for index=1:nargout
    varargout{index}=eval(varn{index});
  end
elseif strcmp(thph,'demo1')
  defval('N',10)
  defval('th','africa')
  region=th;
  eval(sprintf('XY=%s(%i);',region,N));
  thph=[90-XY(:,2) XY(:,1)];
  Nth=rand*300;
  th=linspace(min(thph(:,1)),max(thph(:,1)),Nth);
  [phint,thp,php]=phicurve(thph,th);
  plot(php,90-thp,'k-'); hold on
  plot(thph(:,2),90-thph(:,1),'k-');
  hold off
  axis equal; grid on
elseif strcmp(thph,'demo2')
  [x,y]=blob(1,1);
  thph=[y(:) x(:)];
  Nth=ceil(rand*300);
  th=linspace(min(thph(:,1)),max(thph(:,1)),Nth);
  [phint,thp,php]=phicurve(thph,th);
  plot(php,90-thp,'k-'); hold on
  plot(thph(:,2),90-thph(:,1),'k-');
```

```
  hold off
  axis equal; grid on
  title(sprintf('Number of crossings %i',Nth))
end
```

6.11. ylm

```
function [Y,theta,phi,norms]=ylm(l,m,theta,phi,check,tol,blox)
% [Y,theta,phi,norms]=YLM(l,m,theta,phi,check,tol,blox)
%
% Calculates normalized real spherical harmonics, DT (B.72).
%
% INPUT:
%
% l      degree (0 <= l <= infinity) [default: random]
% m      order (-l <= m <= l)        [default: all orders 0<=l]
%        l and m can be vectors, but not both at the same time
% theta  colatitude (0 <= theta <= pi)   [default: 181 linearly spaced]
% phi    longitude (0 <= theta <= pi)    [default: 361 linearly spaced]
% check  1 optional normalization check by Gauss-Legendre quadrature
%        0 no normalization check
% tol    Tolerance for optional normalization checking
% blox   0 Standard (lm) ordering, l=0:L, m=-l:l
%        1 Block-diagonal ordering, m=-L:L, l=abs(m):L
%
% OUTPUT:
%
% Y      The real spherical harmonics at the desired argument(s):
%           As matrix with dimensions of
%           length(theta) x length(phi) x max(length(m),length(l)) OR
%           (L+1)^2 x (length(theta)*length(phi)) if you put in
%           a degreel=[0 L] and an order []: lists orders -l to l.
% theta  The latitude(s), which you might or not have specified
% phi    The longitude(s), which you might or not have specified
% norms  The normalization matrix, which should be the identity matrix
%
% See also: XLM
%
% EXAMPLES:
%
% plotplm(ylm(2,-1,[],[],1),[],[],1); colormap(flipud(gray(7)))
%
% Last modified by fjsimons-at-alum.mit.edu, 19.05.2006

% Default values
defval('l',round(rand*10))
defval('m',[])
defval('theta',linspace(0,pi,181))
defval('phi',linspace(0,2*pi,360))
defval('check',0)
defval('tol',1e-10)
defval('blox',0)

% Make sure phi is a row vector
phi=phi(:)';

% Revert back to cos(theta)
mu=cos(theta);
```

```
% If the degrees go from 0 to some L and m is empty, know what to do
if min(l)==0 & max(l)>0 & isempty(m)
  [PH,TH]=meshgrid(phi,theta);
  Y=repmat(NaN,(max(l)+1)^2,length(TH(:)));
  for thel=0:max(l)
    theY=reshape(ylm(thel,-thel:thel,theta,phi,check,tol),...
         length(TH(:)),2*thel+1)';
    Y(thel^2+1:(thel+1)^2,:)=theY;
  end
  theta=TH(:);
  phi=PH(:);
  norms=[];
  if blox==1
    [dems,dels,mz,blkm]=addmout(max(l));
    Y=Y(blkm,:);
  end
  return
end

% Error handling common to PLM, XLM, YLM
[l,m,mu,check,tol]=pxyerh(l,m,mu,check,tol);

% Straight to the calculation, check normalization on the XLM
[X,theta,norms]=xlm(l,abs(m),theta,check);

% Initialize the matrix with the spherical harmonics
Y=repmat(NaN,[length(theta) length(phi) max(length(m),length(l))]);

% Make the longitudinal phase: ones, sines or cosines, sqrt(2) or not
P=diag(sqrt(2-(m(:)==0)))*...
  cos(m(:)*phi-pi/2*(m(:)>0)*ones(size(phi)));

% Make the real spherical harmonics
if prod(size(l))==1 & prod(size(m))==1
 Y=X'*P;
else
  for index=1:max(length(m),length(l))
    Y(:,:,index)=X(index,:)'*P(index,:);
  end
end
```

6.12. xlm

```
function [X,theta,norms,dems]=xlm(l,m,theta,check,tol,blox)
% [X,theta,norms,dems]=XLM(l,m,theta,check,tol,blox)
%
% Calculates normalized (associate) Legendre functions, DT (B.58/B.60).
%
% INPUT:
%
% l      degree (0 <= l <= infinity) [default: random]
% m      order (-l <= m <= l)        [default: all orders 0<=l]
%        l and m can be vectors, but not both at the same time
% theta  argument (0 <= theta <= pi) [default: 181 linearly spaced]
% check  1 optional normalization check by Gauss-Legendre quadrature
%        0 no normalization check [default]
% tol    Tolerance for optional normalization checking
% blox   0 Standard (lm) ordering, l=0:L, m=-l:l
%        1 Block-diagonal ordering, m=-L:L, l=abs(m):L
```

```
%
% OUTPUT:
%
% X      The associated normalized Legendre function at the desired
argument(s):
%           as a scalar or a row vector with length(theta) columns, OR
%           as a matrix with length(m) rows and length(theta) columns, OR
%           as a matrix with length(l) rows and length(theta) columns, OR
%           as a 3D matrix of size [length(l) size(theta), OR
%           (L+1)^2 x length(theta) if you put in
%           a degree l=[0 L] and an order []: lists orders -l to l.
% theta  The argument(s), which you might or not have specified
% norms  The normalization matrix, which should be the identity matrix
% dems   The orders to which the Xlms belong (to verify block sorting)
%
% EXAMPLES:
%
% plot(xlm([0:5],0)')
% plot(xlm(5,[])')
%
% Last modified by fjsimons-at-alum.mit.edu, 10/07/2006

% For single m, this will accept 2D theta's and return 3D results X
% See Research Notebook VI p 77ff

% Default values
defval('l',round(rand*10))
defval('m',[])
defval('theta',linspace(0,pi,181))
% Never make this one or it will reach internal recursion limit
defval('check',0)
defval('tol',1e-10)
defval('blox',0)

% Revert back to cos(theta)
mu=cos(theta);

switch check
 case 0
  % If the degrees go from 0 to some L and m is empty, know what to do
  if min(l)==0 & max(l)>0 & isempty(m)
    X=repmat(NaN,(max(l)+1)^2,length(theta));
    for thel=0:max(l)
      X(thel^2+1:(thel+1)^2,:)=xlm(thel,-thel:thel,theta,check,tol);
    end
    norms=[];
    [dems,dels,mz,blkm]=addmout(max(l));
    if blox==1
      X=X(blkm,:);
      dems=dems(blkm);
    end
    return
  end

  % Error handling common to PLM, XLM, YLM
  [l,m,mu,check,tol]=pxyerh(l,m,mu,check,tol);

  % Calculation for m>0
  if prod(size(l))==1 & prod(size(m))==1 % SINGLE L AND M
    % Note that Matlab 'sch' has the sqrt(2-(m==0)) in there so we get
    % rid of it; this option also has gotten rid of the Condon-Shortley
```

```
    % phase which we now need to put back in
    X=(-1)^m*sqrt(2*l+1)/sqrt(2-(m==0))/sqrt(4*pi)*...
      rindeks(legendre(l,mu,'sch'),abs(m)+1);
    if m<0
      X=(-1)^m*X;
    end
  elseif prod(size(l))==1 % SINGLE L MULTIPLE M
    X=sqrt(2*l+1)/sqrt(4*pi)*...
      (rindeks(legendre(l,mu,'sch'),abs(m)+1)'*...
       diag(((-1).^m)./sqrt(2-(m==0))))';
    for index=find(m<0)
      X(index,:)=(-1)^m(index)*X(index,:);
    end
  elseif prod(size(m))==1 % MUTIPLE L SINGLE M
    if min(size(mu))==1 % VECTOR ARGUMENT
      X=repmat(NaN,[length(l) length(mu) 1]);
      ini=1;
    else % MATRIX ARGUMENT
      X=repmat(NaN,[length(l) size(mu)]);
      % The following is to bypass a dimensionality convention
      X(1,:,:)=shiftdim((-1)^m/sqrt(2-(m==0))/sqrt(4*pi)*...
            legendre(0,mu,'sch'),-1);
      ini=2;
    end
    for index=ini:length(l)
      X(index,:,:)=(-1)^m*sqrt(2*l(index)+1)/sqrt(2-(m==0))/sqrt(4*pi)*...
      rindeks(legendre(l(index),mu,'sch'),abs(m)+1);
    end
    if m<0
      X=(-1)^m*X;
    end
  else
    error('Specify valid option')
  end
  norms=[];
 case 1
  % Check normalization
  norms=pxynrm(l,m,tol,'X');
  % Still also need to get you the answer at the desired argument
  X=xlm(l,m,theta,0);
end
```

6.13. addmon (Definition for Indexing arrays)

```
function  [dems,dels,mz,lmcosi,mzi,mzo,bigm,bigl,rinm,ronm,demin]=addmon(L,m)
% [dems,dels,mz,lmcosi,mzi,mzo,bigm,bigl,rinm,ronm,demin]=ADDMON(L,m)
%
% Returns spherical harmonic degree and order indexing arrays.
% For arrays where m=0:l.
%
% INPUT:
%
% L         Maximal degree of the expansion
% m         An order requested [default: NaN, nothing requested]
%
% OUTPUT:
%
% dems      Vector of orders  m involved in the expansion, [0 01 012]
% dels      Vector of degrees l involved in the expansion, [0 11 222]
% mz        Index to the m=0 locations for the zonal coefficients
% lmcosi    Matrix with zeroes where cos/sine coefficients will go
% mzi       Insertion location of m=0 sin terms in vector not listing them
% mzo       Running index into lmcosi(3:4) returning vector without
```

```
%           the sin coefficient at m=0, effectively unwrapping the
%           cosine and sine elements and reverting back to +/-m
% bigm      Single vector with ordered orders, [0 0-11 0-11-22] etc
% bigl      Single vector with ordered degrees, [0 111 22222 3333333] etc
% rinm      Vector reindexing bigm back to [0 -101 -2-1012 -3-2-10123] etc
% ronm      Running index into lmcosi(3:4) unwrapping the orders as in
ADDMOUT
% demin     The index into vector dems listing all orders that belongs to
%           the order m given at the input
%
% SEE ALSO:
%
% XYZ2PLM, PLM2XYZ, MATRANGES, ADDMOUT, ADDMIN, ADDMUP
%
% EXAMPLE:
%
% [M,L,mz,BM]=addmout(10); [a,b,c,d,e,f,M2,L2,rinm]=addmon(10);
% difer(M-M2(rinm))
% difer(L-L2(rinm))
% difer(M(BM)-M2(rinm(BM)))
% difer(L(BM)-L2(rinm(BM)))
%
% EXAMPLE:
%
% Find the positions of the order m in an lmcosi matrix for degree L
%
% L=15; m=round(rand*L);
% [dems,dels,mz,lmcosi,mzi,mzo,bigm,bigl,rinm,demin]=addmon(L,m);
% dems(demin)
%
% Last modified by fjsimons-at-alum.mit.edu, 18.05.2006

defval('m',NaN)

matr=(repmat(0:L,2,1)'*diag([0 1]))';
dems=matranges(matr(:)')';
els=0:L;
dels=gamini(els,els+1)';

if nargout>=3
  mz=[1 cumsum(els(1:end-1)+1)+1]';
  if nargout>=4
    lmcosi=[dels dems zeros(length(dels),2)];
    if nargout>=5
      mzi=(mz*2+1)-[1:length(mz)]';
    if nargout>=6
      % Just number the [dels dems] vector sequentially down the rows
      indo=reshape(1:length(dems)*2,length(dems),2);
      % And stick a zero in where the sine coefficient goes
      indo(length(dels)+mz)=0; indo=indo';
      % Simply pick the nonzero elements on this index array
      mzo=indo(~~indo);
      % Note that this never skips more than two positions
      difer(unique(diff(sort(mzo)))-[1 2 3]')
      if nargout>=7
    fl=[dels dels];
    bigl=fl(mzo);
    if nargout>=8
      fm=[-dems dems];
      bigm=fm(mzo);
    if nargout>=9
```

```
      if  L>=2
        % Must do the first couple of degrees myself since gamini(x,0) is
bad
        lopos=cumsum(2*els(1:end-1)+1)+1;
        ka=[repmat(1,1,L-1) ; els(3:end)-1 ; repmat(1,1,L-1); els(3:end)];
        ko=[[1 0 -1 2] gamini(repmat([0 -2 -1 2],1,L-1),ka(:)')]';
        ko(lopos)=[2:2:2*L];
        rinm=cumsum(ko);
      elseif L==0
        rinm=1;
      elseif L==1
        rinm=[1 3 2 4]';
      end
      if nargout>=10
        ronm=mzo(rinm);
        if nargout>=11
          demin=addmup(m-1:L-1)+m+1;
        end
      end
    end
    end
      end
    end
    end
  end
end
```