Altitude-Cognizant Scalar Slepian Functions

H. A. Werth

Contents

0 Introduction: description of task

The scalar spherical harmonics are a family of real-valued functions which provide a basis in spherical coordinates for real-valued functions defined on the sphere of unit radius. In this context, to be defined on the unit sphere is to be a function of the position of the radial vector $\mathbf{r}$. Scalar Slepian functions are specialized linear combinations of these spherical harmonics. They form an alternative basis, so that any function expressible as a combination of spherical-harmonics may be rewritten in terms of the Slepian functions.

While the spherical harmonics are a global basis, in the sense that they may be used to express functions on their whole domain, the Slepian functions are computed locally in order to represent functions on a region R of the sphere. The utility of the Slepian basis is in that, for a chosen region and a chosen bandwidth, they are the most concentrated within each. (For a proof (?) of this, see [Simons 2006].) The limitation of the bandwidth is a condition on the number of spherical harmonics used in each function, an important consideration in computation. Since a Slepian function is a linear combination of spherical harmonics, to "compute/construct a Slepian function" is to solve for a vector of spherical-harmonic coefficients.

The Slepian functions are designed to represent and analyse experimental data. In this document we will be concerned with locally-collected satellite data which are values of the negative first derivative of some potential field on a planet's surface. The task of interest is data inversion – we want to use the radial derivative data to solve for the potential field, which is mathematically approximated by a real-valued

function of radial position. Since we assume the data is available only regionally, we aim to represent (a local portion of) the potential field as a linear combination of Slepiann functions. So, to "solve/invert for the potential field" is, in practice, to determine a vector of coefficients in the Slepian basis.

The Slepian functions we will be using are "altitude-cognizant". They are best-concentrated like the classical Slepian functions, but their construction explicitly takes into account that the data are recorded at an elevation above the source of the potential field. One solves for the spherical-harmonic coefficients of the Slepian basis functions under the assumption that the coefficients are elevated to the average altitude of data collection. The data inversion process involves first representing the potential field at the level of the data and then "downward continuing" that representation to the planet's surface. For a detailed account of both the derivation and application of the altitude-cognizant Slepian functions, see [Plattner and Simons 2017].

The purpose of this document is to present the matlab code for all the steps of the data inversion process, from generating random data to solving for the potential field.

1 Create model potential field

We need to base our creation of random data on a model potential field, so that we may compare the inverted potential field to this model field in the final step.

Create field:

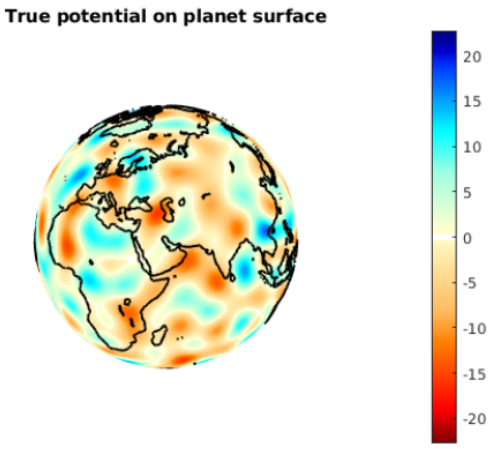
```
Lmax=20
rnd_lmcosi=plm2rnd(Lmax,0);
coef=lmcosi2coef(rnd_lmcosi);
```

The "field" is a linear comination of spherical harmonics, whose coefficients are given in the matrix output `coef`.

Plot field:

```
lmcosi=coef2lmcosi(coef/sqrt(4*pi));
clf;
res = 0.5;
plotplm(lmcosi,[],[],2,res)
title('True potential on planet surface')
cax=caxis
caxis([-1 1]*max(abs(cax)))
kelicol(1)
colorbar
```

It is necessary to include the factor `sqrt(4*pi)` whenever plotting fields generated with `plm2rnd` using `plotplm`.



2 Evaluate the field's derivative at random points

Choose parameters for data location:

```
dom= 'africa'
N= 2000
planetrad=6371
satrad=6371+400
varalt= 100
```

`N` is the maximum number of data points. Each datum is recorded at a satellite altitude; `satrad` is the average of these altitudes. `varalt` is the optional range of variation for satellite altitude.

Get random data locations:

```
[lon, lat] = randinpoly(dom,N);
cola = 90-lat
rad = satrad +(rand(length(lon),1)-0.5)*varalt;
theta= cola*pi/180
phi = lon*pi/180
```

This sequence of steps generates up to `N` random points first within `dom`, and then, for each point, generates a random satellite altitude which has a uniformly random distribution in `[satrad-varalt/2 satrad+varalt/2]`.

Evaluate the derivative at random points:

```
data=rGscal(coef,theta,phi,rad,planetrad)';
```

The function `rGscal` computes and evaluates the field's negative radial derivative point-wise, taking into account the altitude of the points. The values contained in the output matrix `data` form our "collected data".

2.1 Compute spherical-harmonic coefficients for Slepian basis functions

```
[G,V]=glmalphaup(dom,Lmax,satrad,planetrad);
```

The altitude-cognizant Slepian basis functions depend on `data` only through the region `dom` and the average altitude `satrad` of their collection. `G` is the matrix of coefficients and `V` the vector of eigenvalues (for more information on the output of `glmalphaup`, refer to its `help` section).

This step is relegated to a subsection because it is not necessary to perform in the course of data inversion. The function `LocalIntField`, used in the next section, computes the appropriate Slepian basis automatically.

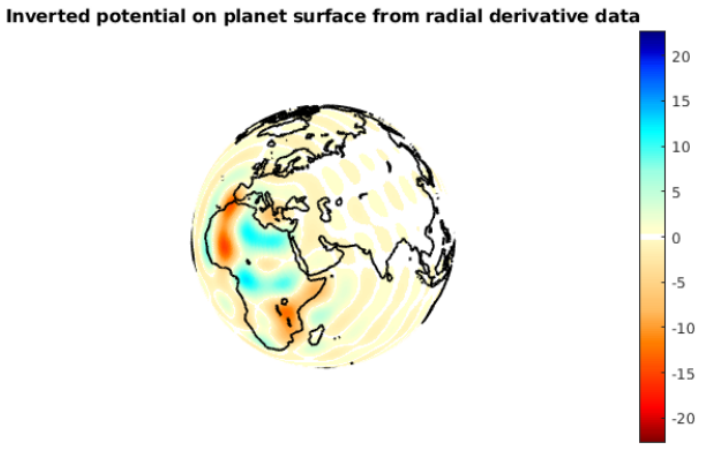
3 Solve for the potential field

```
J=40
result_coef=LocalIntField(data,rad,theta,phi,dom,Lmax,J,planetrad,satrad);
```

The choice of `J` dictates how many of the Slepian basis functions are used to represent the potential field in the inversion, which the function `LocalIntField` wholly accomplishes. Larger `J` means the computation is more sensitive to noise but results in higher spatial resolution.

Plot inverted field:

```
figure
result_lmcosi=coef2lmcosi(result_coef/sqrt(4*pi));
plotplm(result_lmcosi,[],[],2,res)
title('Inverted potential on planet surface from radial derivative data')
caxis([-1 1]*max(abs(cax)))
kelicol(1)
colorbar
```



In order to plot, we had first to convert the coefficients to `lmcosi` format, as usual. Note how the inverted field is expressed in the local basis while the true field with which we began was in the global basis.

Plot difference of true and inverted field:

```
clf;
diff_lmcosi=coef2lmcosi((coef - result_coef)/sqrt(4*pi));
plotplm(diff_lmcosi,[],[],2,res)
title('Difference between true and inverted potential fields')
caxis([-1 1]*max(abs(cax)))
kelicol(1)
colorbar
```