Scalar Spherical Harmonics and Slepian Functions

H. A. Werth

Contents

1 Plot a single spherical harmonic function

We will demonstrate plotting a spherical-harmonic on a sphere, in a standard Matlab plot, on a Mollweide projection, and on random points of a sphere.

First, designate a spherical-harmonic to be plotted:

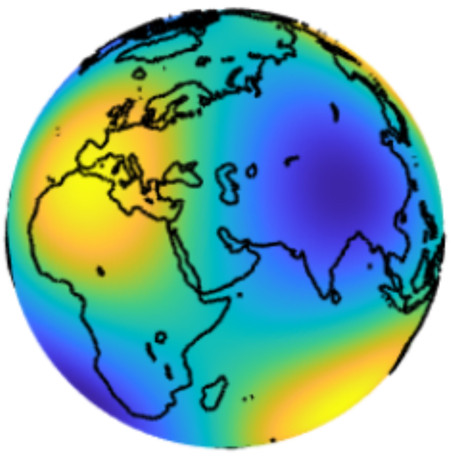
For example,

`l` = 3; `m` = -2;

0 `l` fixes the degree and `m` fixes the order.

1.1 Plot on sphere

```
l = 3; m = -2;
lon = 0:0.5:360;
lat = -90:0.5:90;
Y = ylm(l, m, pi/180*(90-lat), pi/180*lon);
figure;
plotplm(Y, pi/180*lon, pi/180*lat,2)
```



1. Create a grid on the sphere

```
lon = 0:0.5:360; lat = -90:0.5:90;
```

This creates a coordinate point every half-degree.

2. Calculate the values of the function for coordinate points on the sphere

```
Y = ylm(l, m, pi/180*(90-lat), pi/180*lon);
```

The function `slepian_alpha/ylm.m` evaluates the spherical harmonic function of degree `l` and order `m` at every point `pi/180*(90-lat)`, `pi/180*lon` on the grid. We name the vector of the spherical-harmonic values `Y`.

Note that `90-lat` is needed to convert latitude to colatitude and `pi/180` is needed to convert degrees to radians.

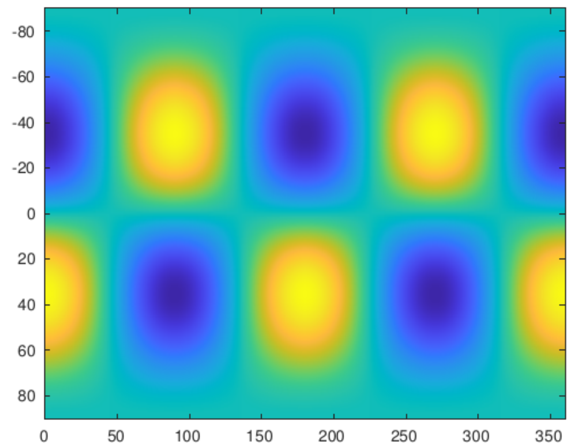
3. Plot

```
figure; plotplm(Y, pi/180*lon, pi/180*lat,2)
```

The function `slepian_alpha/plotplm.m` is here used to plot the vector `Y` using the grid specified by `lon` and `lat` in step 1. The input `2` dictates that the graph be on a sphere.

1.2 Plot in standard Matlab plot

```
l = 3; m = -2;
lon = 0:0.5:360;
lat = -90:0.5:90;
Y = ylm(l, m, pi/180*(90-lat), pi/180*lon);
imagesc(lon, lat, Y)
```

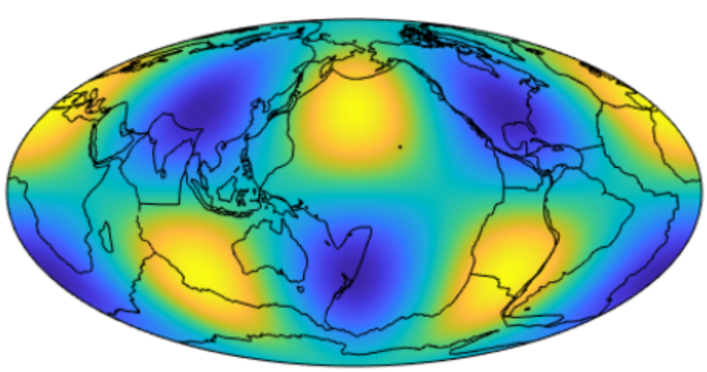


Do steps 1 and 2, and then run

```
imagesc(lon, lat, Y)
```

1.3 Plot on Mollweide projection

```
l = 3; m = -2;
lon = 0:0.5:360;
lat = -90:0.5:90;
Y = ylm(l, m, pi/180*(90-lat), pi/180*lon);
figure;
plotplm(Y, pi/180*lon, pi/180*lat,1)
```



Do steps 1 and 2, and then run

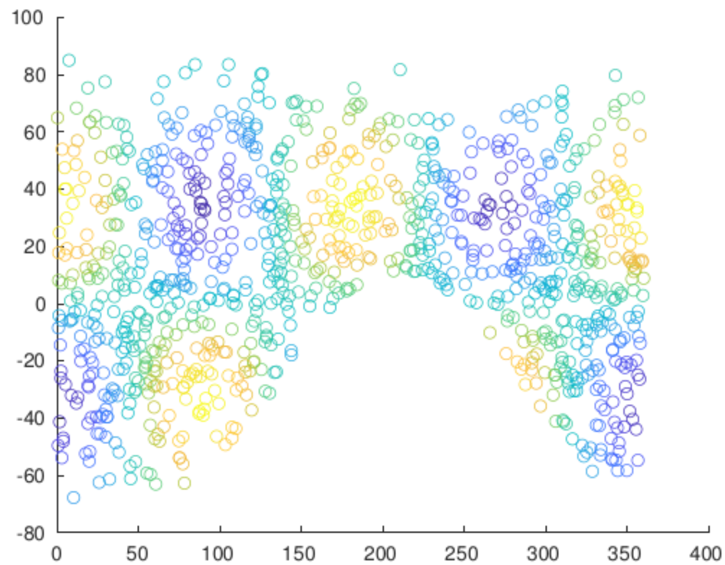
```
figure;
```

```
plotplm(Y, pi/180*lon, pi/180*lat,1)
```

The input `1` dictates that the graph be on the Mollweide projection.

1.4 Plot for random points on a sphere

```
l = 3; m = -2;
TH = 120; lon0 = 30; cola0 = 40; N=1000;
[lon, lat] = randpatch(N,TH,lon0,cola0);
Y = ylm(l, m, pi/180*(90-lat), pi/180*lon,[],[],[],1);
scatter(lon, lat, [], Y)
```



1. Generate a subset of the sphere consisting of random points

In particular, we will create `N` randomly-generated coordinate points within a spherical cap of opening angle `TH` and centered at longitude `lon0` and colatitude `cola0`

For example,

```
TH = 120; lon0 = 30; cola0 = 40; N=1000;
```

```
[lon, lat] = randpatch(N,TH,lon0,cola0);
```

The function `slepian_alpha/randpatch.m` creates the set of random points within the spherical cap of the specified values. We name those coordinate points `[lon,lat]`.

2. Calculate the values of the spherical harmonic at those points

```
Y = ylm(l, m, pi/180*(90-lat), pi/180*lon,[],[],[],1);
```

`ylm.m` takes the arguments `l`, `m`, `pi/180*(90-lat)`, `pi/180*lon` as before. Run `help ylm` for information on all eight arguments.

3. Plot

If necessary, use the Matlab command

```
clf;
```

To clear existing figures, and then run the Matlab command

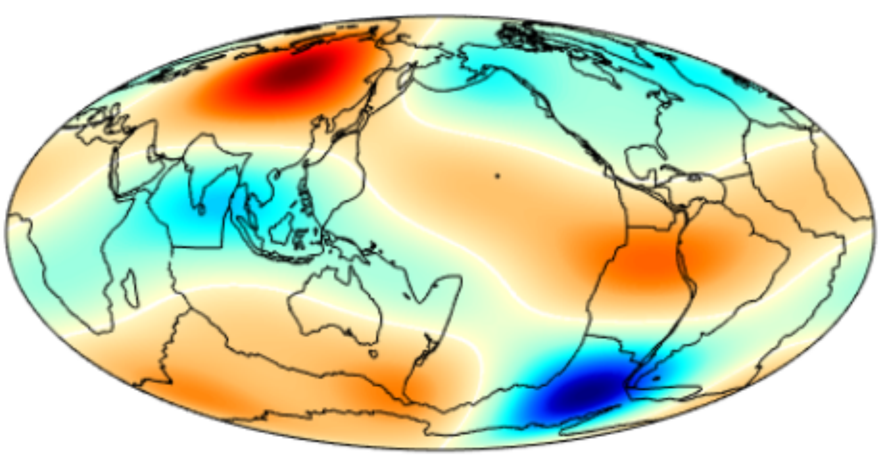
```
scatter(lon, lat, [], Y)
```

To create a scatter plot of circles having locations `[lon, lat]`. Here, `[]` indicates the default value for circle size and the vector of spherical-harmonic values `Y` is used to determine circle color.

Please see `Ch_01` in the `.edu` folder for more detailed information.

2 Plot a linear combination of spherical harmonics

```
lon = 0:0.5:360;
lat = -90:0.5:90;
Y1=ylm(3,1,pi/180*(90-lat),pi/180*lon);
Y2=ylm(1,-1,pi/180*(90-lat),pi/180*lon);
Y3=ylm(5,-2,pi/180*(90-lat),pi/180*lon);
Y4=4*Y1-0.2*Y2+2*Y3;
plotplm(Y4, pi/180*lon, pi/180*lat,1);
kelicol(1)
```



This task is a simple variation on the first.

Let us define three spherical harmonics:

```
lon = 0:0.5:360;

lat = -90:0.5:90;

Y1=ylm(3,1,pi/180*(90-lat),pi/180*lon);

Y2=ylm(1,-1,pi/180*(90-lat),pi/180*lon);

Y3=ylm(5,-2,pi/180*(90-lat),pi/180*lon);
```

Next, create a vector which is a linear combination of these three. For example,

```
Y4=4*Y1-0.2*Y2+2*Y3;
```

To plot the function, use `plotplm.m`. For example,

```
plotplm(Y4, pi/180*lon, pi/180*lat,1)
```

If you're interested in another color scheme, try out

```
kelicol(1)
```

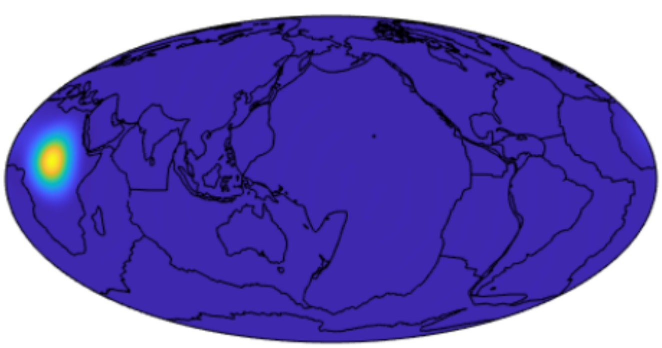
Please see `Ch_01` in the `.edu` folder for more detailed information.

3 Create and plot scalar Slepian functions

3.1 Named region and polar cap

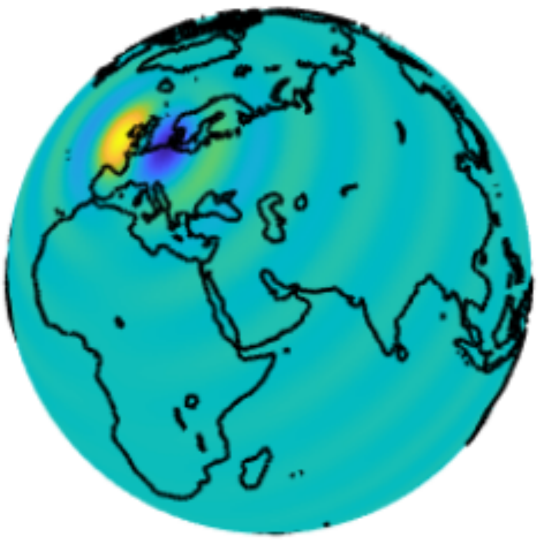
Named region example 1:

```
[G] = glmalpha('africa',20,[],0);
lmcs = coef2lmcosi(G(:,1),1);
data=plm2xyz(lmcs,0.5);
plotplm(data, [], [], 1, 0.5)
```



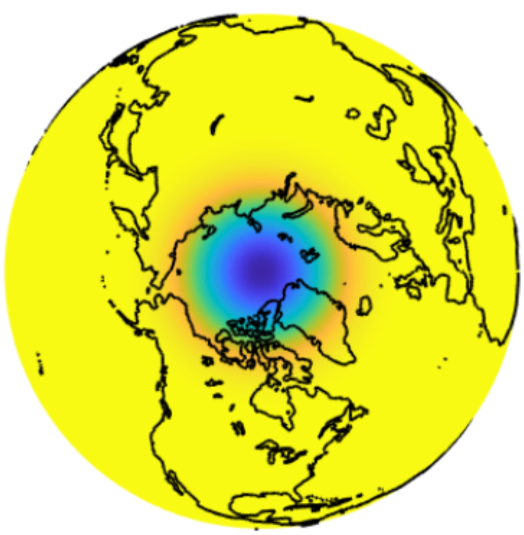
Named region example 2:

```
[G] = glmalpha('england',20,[],0);
lmcs = coef2lmcosi(G(:,3),1);
data=plm2xyz(lmcs,0.5);
plotplm(data, [], [], 2, 0.5)
```



Polar cap example:

```
[G] = glmalpha(40,20,1,0)
lmcs = coef2lmcosi(G(:,1),1);
data=plm2xyz(lmcs,0.5);
plotplm(data, [], [], 2, 0.5)
view(2)
```



1. Generate the coefficients of the function

We will use the function `glmalpha.m`, which will essentially compute for us the best spatially-concentrated Slepian functions given two constraints. Those constraints are the first input, `TH`, and the second, `L`. `TH` can be either a named region or the opening angle in degrees of a polar cap. `L` is the bandwidth.

You may choose among the named regions 'england', 'eurasia', 'namerica', 'australia', 'greenland', 'africa', 'samerica', 'amazon', 'orinoco', 'antarctica', 'contshelves', and 'alloceans'.

For example,

```
[G] = glmalpha('africa',20,[],0);
```

2. Plot

`[G]` in both cases is a matrix whose nth column holds the coefficients of the nth-best spatially-concentrated Slepian function. In order to plot the nth function we need to convert the nth column of `[G]` into the form recognized by the function `plotplm.m`, the so-called "lmcosi" format.

```
lmcs = coef2lmcosi(G(:,1),1);
```

Here we have chosen to use the first column `G(:,1)`. The second input `1` is necessary when the coefficients are calculated using `glmalpha`.

Now we input the (lmcosi-formatted) matrix `lmcs` and a resolution into the function `plm2xyz`. We choose the resolution to be 0.5 and name the ouput "data":

```
data=plm2xyz(lmcs,0.5);
```

To plot, run

```
plotplm(data, [], [], 1, 0.5)
```

The input `1` specifies Mollweide projection and the input `0.5` is just the resolution again.

The same sequence is used to plot a Slepian function on a polar cap, except for a change in the inputs to `glmalpha.m` Specifically, we will now let `TH` denote an opening angle in degrees.

For example, let `TH = 40`.

```
[G] = glmalpha(40,20,2,0)
```

The command

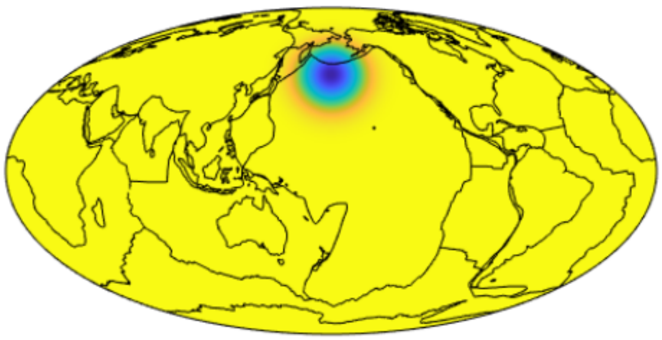
```
view(2)
```

may be used to rotate the spherical figure so that the North pole is faced toward the viewer.

We suggest reading the `help` section for the relevant functions and/or Chapter 2 Section 1 in the folder .edu for a detailed discussion.

3.2 Rotated polar cap

```
[G] = glmalphaptoJ(40,20,180,45,0,10)
lmcs = coef2lmcosi(G(:,1),1);
data=plm2xyz(lmcs,0.5);
plotplm(data, [], [], 1, 0.5)
```



The general method of calculuating coefficients for a rotated polar cap is the same as above, but involves slightly different commands.

We will need to use the function `glmalphaptoJ` and provide the following inputs:

`TH`: opening angle

`L`: maximum spherical harmonic degree (bandwidth)

`phi`: longitude in degrees

`theta`: colatitude in degrees

`omega`: rotation of the region itself, if any

`J`: number of Slepian functions to be calculated

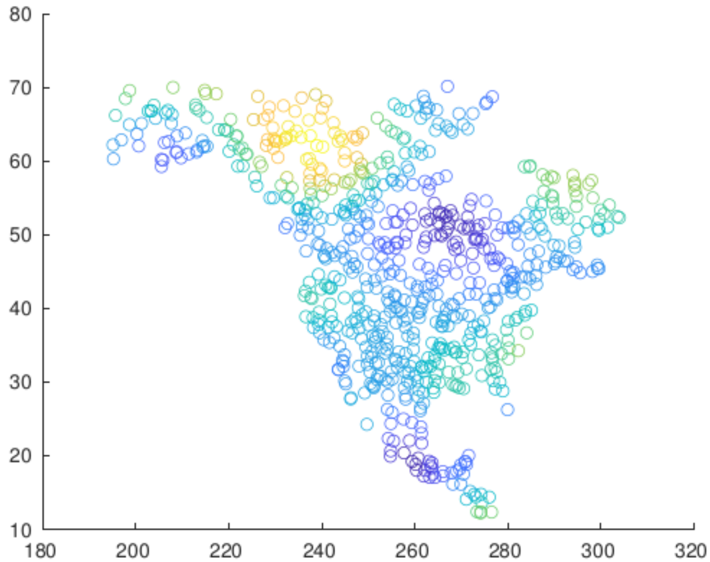
4 Compute spherical-harmonic coefficients from regional data

The goal is to be able to calculate spherical-harmonic coefficients that describe observations when we only have these observations within a region.

In a first step we create these observations from random spherical-harmonic coefficients.

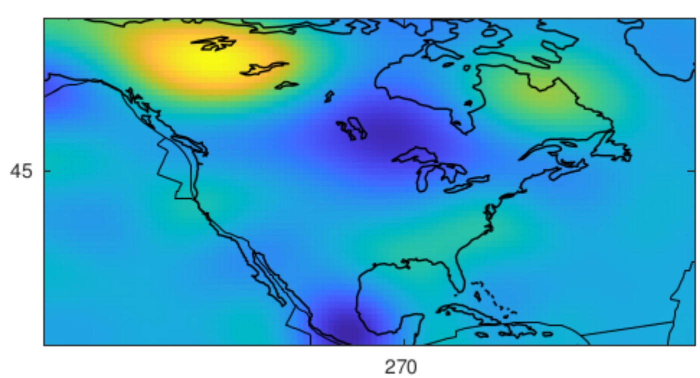
```
N=5000
dom='namerica'
[lon,lat]=randinpoly(dom,N);
Lmax=20
```

```
lmcosi=plm2rnd(Lmax,0)
data = plm2xyz(lmcosi,lat,lon);
subplot(2,1,1)
scatter(lon,lat,[],data)
```



In a second step we try to obtain the spherical-harmonic coefficients from the observations.

```
[G,V]=glmalpha(dom,Lmax);
Y=ylm([0 Lmax],[],(90-lat)*pi/180,lon*pi/180+pi,[],[],[],1
J = round(1.5*(Lmax+1)^2*spharea(dom));
Geval = G(:,1:J)'*Y;
gcoef = (Geval*Geval')\(Geval*data);
coef = G(:,1:J)*gcoef;
subplot(2,1,2)
plotplm(coef2lmcosi(coef,1),[],[],4,1)
```



Step 1.

First choose a region and create N random data locations within the region. For example,

```
N=5000
```

```
dom='namerica'
```

```
[lon,lat]=randinpoly(dom,N);
```

Next, create random spherical-harmonic coefficients using the function plm2rnd.m, which makes random coefficients for spherical harmonics up to degree L and stores them in an lmcosi matrix. For this example, let Lmax=L=20.

```
Lmax=20
```

```
lmcosi=plm2rnd(Lmax,0)
```

Now evaluate the Slepian function determined by these random coefficients at these locations

```
data = plm2xyz(lmcosi,lat,lon);
```

To look at the data,

```
subplot(2,1,1)
```

```
scatter(lon,lat,[],data)
```

Step 2.

We will now treat data as a collection of observations and try to find the Slepian function which best fits the observations. We already know the function of best fit in this example; it is that whose spherical-harmonic coefficients are given in lmcosi and whose values are the entries of data. Hence this Step should recover for us these coefficients.

First we will use glmalpha to compute the coefficients of the best spatially-concentrated Slepian functions over our chosen region dom and chosen bandwidth Lmax. From this set of functions we will choose the one which best fits data.

```
[G,V]=glmalpha(dom,Lmax);
```

First evaluate the spherical harmonics of degrees 0 to Lmax at the data points:

```
Y=ylm([0 Lmax],[],(90-lat)*pi/180,lon*pi/180+pi,[],[],[],1
```

Next evaluate the Slepian functions at the data points. All of these evaluated Slepian functions are linear combinations of the above evaluated spherical harmonics; they differ by their coefficients, which are stored in the matrix [G,V]. Therefore, in order to evaluate a Slepain function whose coefficients come from the nth vector of [G,V], we would run something like

```
eval=G(:,n)'*Y
```

(Recall that `transpose(A)=A'` in Matlab.)

But we want to evaluate the functions for the first `J` vectors of `[G,V]` simultaneously (in order that we may compare them for best fit to `data`), so we run

```
J = round(1.5*(Lmax+1)^2*spharea(dom));
```

```
Geval = G(:,1:J)'*Y;
```

See that we have replaced the column argument `n` with `1:J`

To determine which of these `J` functions best fits `data`, we must solve

```
Geval*gcoef = data
```

where

```
gcoef = (Geval*Geval')\(Geval*data);
```

One will recognize this as the least-squares method of solving an overdetermined linear system.

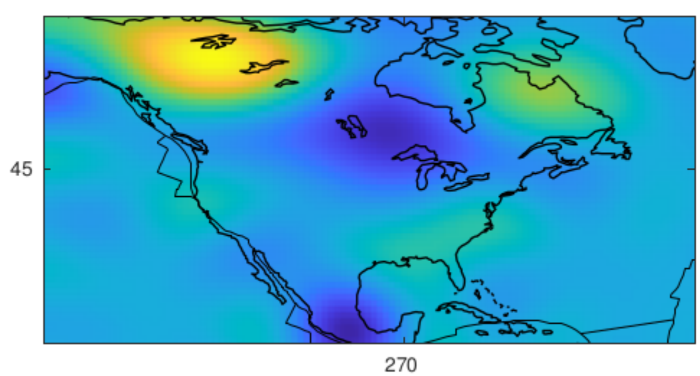
The vector of best-fit coefficients `gcoef` is currently in the Slepian basis. To translate it into the spherical-harmonic basis, run

```
coef = G(:,1:J)*gcoef;
```

To view the function whose coefficients are given in `coef`,

```
subplot(2,1,2)
```

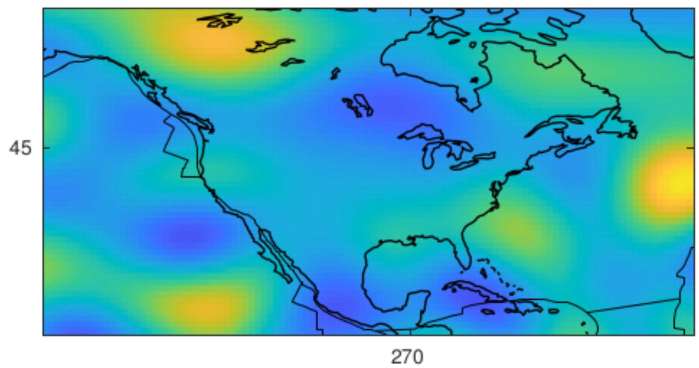
```
plotplm(coef2lmcosi(coef,1),[],[],4,1)
```



To compare this with the graph of function whose values are the entries in `data`, run

```
plotplm(lmcosi,[],[],4,1)
```

And zoom in on the region namerica.



You can also compare the vectors of coefficients `lmcosi2coef(lmcosi,1)` and `coef` directly.