Plotting on the Sphere

Grady Wright

Contents

In this tutorial we review some techniques for plotting scalar-valued functions and vector fields on the surface of the sphere.

```
LW = 'linewidth'; lw = 1; FS = 'FontSize'; fs = 12;
FC = 'FaceColor'; fc = [0.93 0.93 0.93]; EC = 'EdgeColor'; ec = 'none';
vw = [70 25];
```

Longitude-Latitude plots

The sphere can, of course, be parameterized in spherical coordinates consisting of a longitudinal (or azimuthal) coordinate λ and latitudinal (or elevation) coordinate θ. Here we use the Matlab convention that $-\pi \leq \lambda \leq \pi$ and $-\pi/2 \leq \theta \leq \pi/2$. Since this is a logically rectangular coordinate system, we can use the standard surface plotting routines of Matlab to visualize a function. The first step is to create a "grid" of the domain:

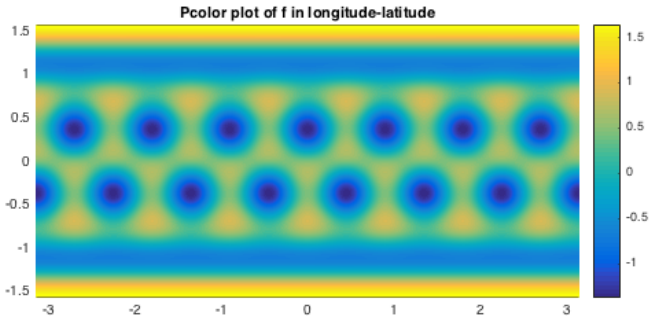
```
res = 201;
lambda = linspace(-pi,pi,res);
theta = linspace(-pi/2,pi/2,ceil(res/2));
[L,T] = meshgrid(lambda,theta);
```

Next we need a function defined on the sphere and sampled at the grid locations. We will use a combination of spherical harmonics Y_8^0 and Y_8^7, which are available from the `sphHarm` function in the **spherepts** package:

```
f = sphHarm(8,0,L,T)+sphHarm(8,7,L,T);
```

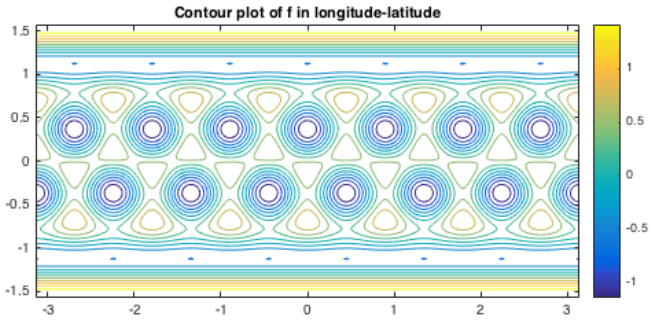
We can visualize this function using `pcolor` as

```
pcolor(L,T,f);
colorbar, shading interp, daspect([1 1 1]);
title('Pcolor plot of f in longitude-latitude')
```



Alternatively, we can make a contour plot of the function with 12 contour lines as

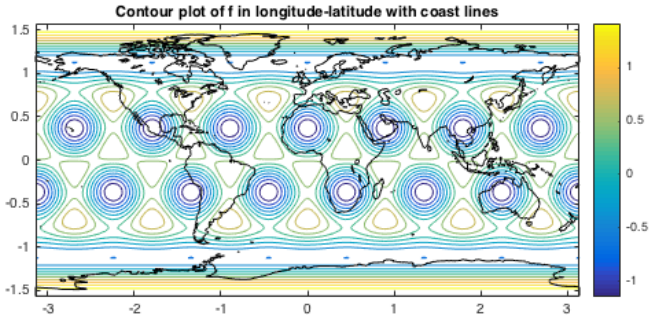
```
lvls = 12;
contour(L,T,f,lvls,LW,lw);
colorbar, daspect([1 1 1]);
title('Contour plot of f in longitude-latitude')
```



We can add in a plot of the coast lines of the land on Earth to plots in longitude

and latitude using the `plotCoastLines` function in the **spherepts** package.

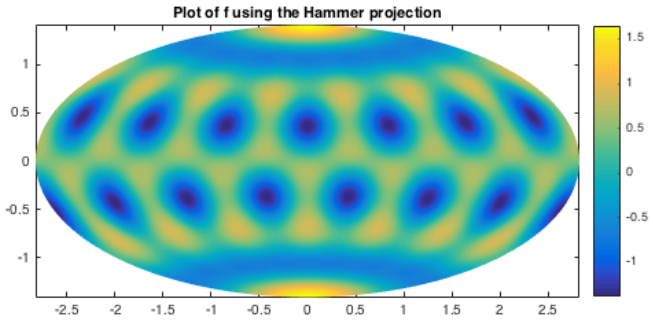
```
hold on;
plotCoastLines(0,'k-');
title('Contour plot of f in longitude-latitude with coast lines')
hold off;
```



Plots using the Hammer projection

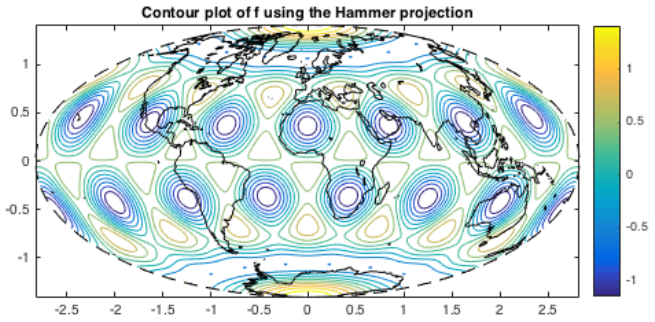
Another common way to plot data on the sphere is using a Hammer projection of the sphere, which is a 2D projection developed by Ernst Hammer. To make a Hammer plot, the spherical coordinates first need to be mapped to the Hammer coordinates. This can be done using the `sph2hammer` function in the **spherepts** package. Then the function can be plotted using the `pcolor` as above.

```
[HX,HY] = sph2hammer(L,T);
clf;
pcolor(HX,HY,f);
title('Plot of f using the Hammer projection')
colorbar, shading interp, daspect([1 1 1]);
```



A contour plot can also be made in a straightforward manner in the Hammer projection and the land masses can be added to the plot by switching the plot type in the `plotCoastLines` function:

```
clf;
contour(HX,HY,f,lvls,LW,lw);
title('Contour plot of f using the Hammer projection')
colorbar, daspect([1 1 1]);
hold on;
plotCoastLines(2,'k-');
hold off;
```



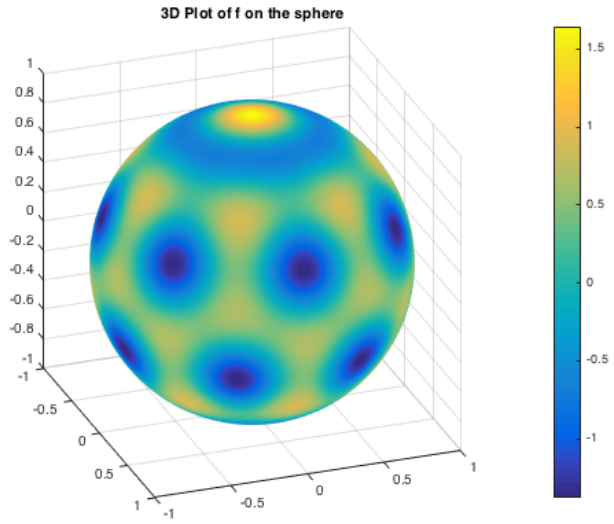
3D plots on the sphere

Plots in longitude-latitude or the Hammer projection do not quite give an accurate representation of a function on the sphere because of the unnatural "stretching" that occurs from mapping the surface of a 3D sphere to a 2D plane. A more natural way to view functions is instead directly on the surface of the sphere in 3D. To do this we must convert the spherical coordinates to Cartesian coordinates:

```
[X,Y,Z] = sph2cart(L,T,1);
```

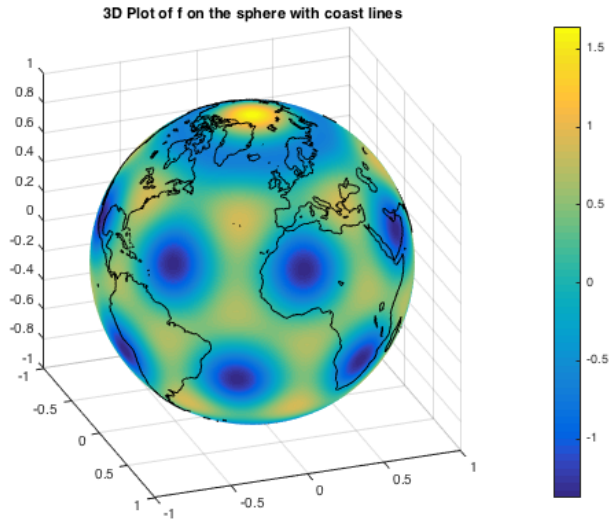
Now the `surf` command can be used to produce the desired plot:

```
clf;
surf(X,Y,Z,f);
colorbar; shading interp; daspect([1 1 1]); axis tight; view(vw);
title('3D Plot of f on the sphere')
```



The land masses can also be included on this plot.

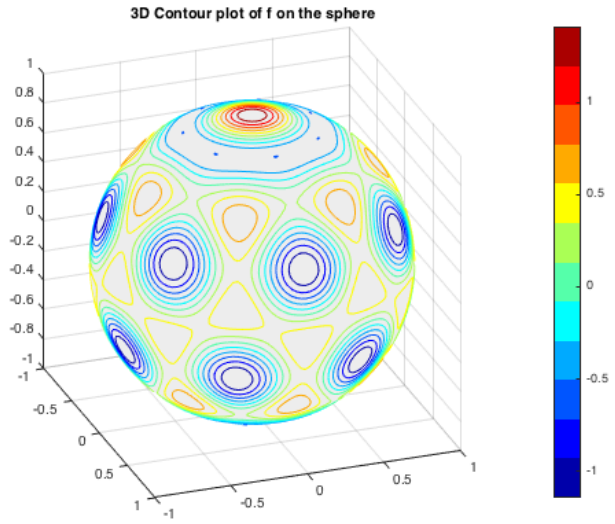
```
hold on;
plotCoastLines(1,'k-');
title('3D Plot of f on the sphere with coast lines')
hold off;
```



Drawing contours on the surface of the sphere is more tricky than just calling the `contour` function, but nevertheless can be done. The idea is to first compute

the contours in longitude-latitude, as above, and then map the contour lines to the sphere. Here is a code for doing this:

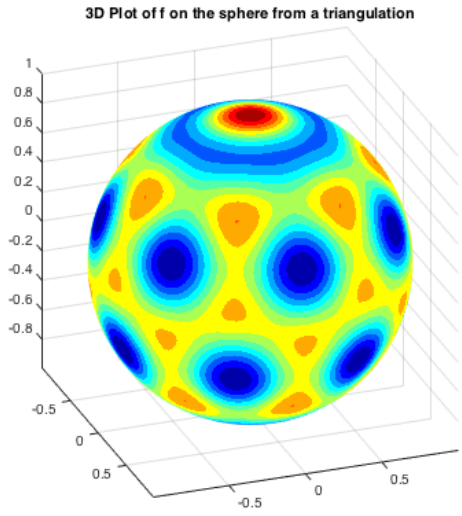
```
c_f = contour(L,T,f,lvls);
[~,cl] = size(c_f);
clf;
surf(X,Y,Z,FC,fc,EC,ec); % Create a solid sphere to plot the results on
hold on;
cmap = jet(lvls);        % Colormap for the contours
k = 1;
cnum = 1;
clvl = c_f(1,k);
cmin = clvl;
while k < cl             % Draw each contour line.
   kl = c_f(2,k);
   v = k+1:k+kl;
   xv = cos(c_f(2,v)).*cos(c_f(1,v));
   yv = cos(c_f(2,v)).*sin(c_f(1,v));
   zv = sin(c_f(2,v));
   if c_f(1,k) ~= clvl
       cnum = cnum+1;
       clvl = c_f(1,k);
   end
   plot3(xv,yv,zv,'-',LW,lw,'Color',cmap(cnum,:)), hold on;
   k = k+kl+1;
end
title('3D Contour plot of f on the sphere')
colormap(cmap); colorbar; caxis([cmin clvl]);
axis tight; daspect([1 1 1]); view(vw);
```



3D Plots from triangulations:

If the data is sufficiently sampled on the sphere and the sample points are quasi-uniformly distributed, then a surface plot of the data can be obtained from a triangulation of the sample points. The function `delaunaySph` in the **spherepts** package can be used to compute a triangulation of the sample points and this can be combined with the `trisurf` command in Matlab to obtain a plot. We illustrate this below for N=10000 minimum energy sample sites.

```
clf;
x = getMinEnergyNodes(10000); z = x(:,3); y = x(:,2); x = x(:,1);
f = sphHarm(8,0,x,y,z)+sphHarm(8,7,x,y,z);  % resample f at new points
tri = delaunaySph([x y z]);
trisurf(tri,x,y,z,f);
title('3D Plot of f on the sphere from a triangulation')
shading interp; axis tight; daspect([1 1 1]); view(vw);
```



Vector fields

Vector fields tangent to the sphere, such as the horizontal wind field, can also be plotted rather easily in Matlab using the `quiver` and `quiver3` function. Unlike the `pcolor`, `surf`, and `contour` commands used above, the `quiver` functions does not require the input data be sample on a logically rectangular grid. So for the examples below we will use sample points from the minimum energy nodes:

```
x = getMinEnergyNodes(43^2);
[lambda,theta] = cart2sphm(x);
```

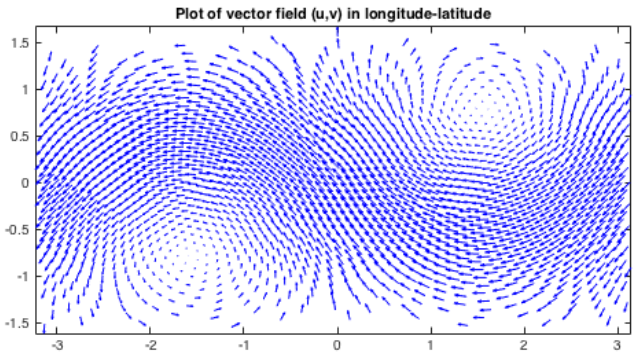
Plots in Longitude-Latitude

If the vector field is defined with respect to the the spherical coordinates system then a 2D plot of the vector field can be made using the `quiver` function. Consider the following vector field corresponding to solid body rotation with the rotation axis set at 45 degrees from the polar axis of the sphere:

```
alpha = pi/4;
u = -(cos(theta)*cos(alpha)-sin(theta).*sin(lambda).*sin(alpha));
v = cos(lambda).*sin(alpha);
```

A plot of this vector field can be made using

```
clf;
quiver(lambda,theta,u,v,'b-',LW,lw);
title('Plot of vector field (u,v) in longitude-latitude')
axis tight; daspect([1 1 1]);
```



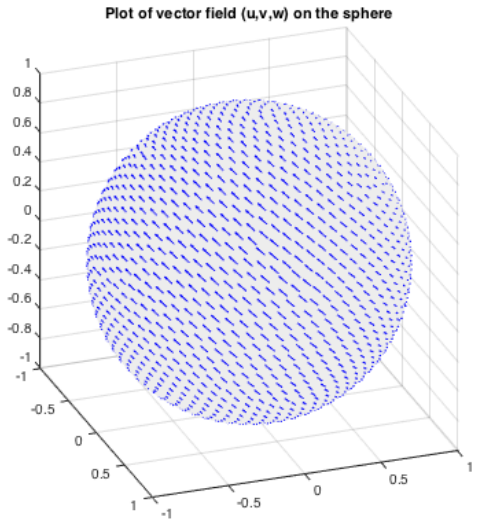
3D plots on the sphere

Plots of vector fields on the surface of the sphere can be made using the `quiver3` function. To make these plots, however, the vector field must be expressed with respect to the Cartesian coordinate system. The functions `sphv2cartv` and `cartv2sphv` in the **spherepts** package provide an easy way to switch between the spherical coordinate and Cartesian coordinate representations of the fields.

```
[x,y,z,u,v,w] = sphv2cartv(lambda,theta,u,v);
```

On the sphere the vector field now looks like

```
clf;
surf(X,Y,Z,FC,fc,EC,ec); % Create a solid sphere to plot the results on
hold on;
quiver3(x,y,z,u,v,w,'b-');
title('Plot of vector field (u,v,w) on the sphere')
axis tight; daspect([1 1 1]); view(vw);
```


Plot of vector field (u,v,w) on the sphere
1
0.8
0.6
0.4
0.2
0
-0.2
-0.4
-0.6
-0.8
-1
-1
-0.5
0
0.5
1
-1
-0.5
0
0.5
1