

# 1 Overview

This document describes the components of the mapping and tracking pipeline at the MPS from `seismo1` or `seismo2`. You can also refer to this website [www.mps.mpg.de/projects/seismo/Track%27n%27Map/Track%27n%27Map.html](http://www.mps.mpg.de/projects/seismo/Track%27n%27Map/Track%27n%27Map.html) It begins with an input file which defines all the non-fixed parameters. The fixed parameters are discussed in the relevant sections below. First set up the classpath for the java file.

```
> export CLASSPATH=/opt/pegasus/default/lib/pegasus.jar:/opt/pegasus/default/lib/vdl.jar:/data/seismo2/workflows/Assets/JAVA:/data/seismo2/workflows/Assets/JAVA/postgresql-9.0-801.jdbc4.jar
```

Then run the java file to create the DAX file which is submitted to Pegasus.

```
> java -classpath $CLASSPATH CreateDAXva1p09 input_parameter_file
```

Then simply follow the online instructions to run the pipeline with Pegasus.

The input parameter file must follow this structure with each new parameter on a new line:

`version=1.05` — make sure this is the correct version number

`daxfile=MapTrackDAX.dax` — don't touch this

`in_series=hmi_test.m_45s` — name of the input series you want to get data from  
`out_series=mps_schunker.m_hmi_lowres` — the name of the output series (this must exist - if not create one in step 1.)

`t_start=2010.11.11_00:00:00_TAI` — the start time of the data you want to use  
`missing_frames=b` — fill any missing frames with blank (there is also an interpolation option - not tested and not recommended)

`workflowname=trackmap`

`namespace=MPS`

`mapping_exec=drms_mapping`

`cube_exec=drms_cube_care`

`projection_type=postel` — type of projection. For now use only Postel.

`n_trac_regions=1` — the total number of tracked regions in time

`n_spg=1` — how many frames to overlap the cubes in time. Negative is overlap, positive puts gaps in between (ignored if `n_trac_regions=1`)

`n_phi=1` — the number of longitudes of the centre of mapped regions

`n_lambda=1` — the number of latitudes of the centre of mapped regions

`n_map_jobs=30` — the number of cpus to split this run over. No more than 30 is recommended for now.

`phi_cen_first_box=2.5337940` — Carrington longitude of centre of map at central

time (radians)  
 lambda\_cen\_first\_box=-0.33161256 — Carrington latitude of centre of map at  
 central time (radians)  
 dx=0.001 — pixel size in the horizontal direction in units of radians  
 dy=0.001 — pixel size in the vertical direction in units of radians  
 a0=-0.02893 — constant of differential rotation profile (for more information on  
 tracking rates see here)  
 a2=-0.3441 — coefficient for  $\sin^2$  of differential rotation profile  
 a4=-0.5037 — coefficient for  $\sin^4$  of differential rotation profile  
 spec=norm — specified tracking, see below  
 ar\_number=11084 — active region number (if not applicable put NA)

We now describe the different components - tracking, mapping and creating the datacube - in more detail.

## 2 Tracking

The tracking is dealt with in the java file `CreateDAXva1p09.java`, run by

```
> export CLASSPATH=/opt/pegasus/default/lib/pegasus.jar:/opt/pegasus/default/lib/vdl.jar
> java -classpath $CLASSPATH CreateDAXva1p09 input_parameter_file
```

The default tracking is latitudinally dependent at the Snodgrass rotation rate,  $\omega(\lambda) = a0 + a2 \sin^2 \lambda + a4 \sin^4 \lambda$ , where the coefficients are  $a0 = -0.02893$ ,  $a2 = -0.3441$ ,  $a4 = -0.5037 \mu\text{rad/sec}$ . Any other values of the coefficients can be set in the input parameter file. There are four options that can be specified in the input file for the tracking.

- **centre:** Tracks centred on the sub-observation point of HMI at the middle time. Ignores the "phi\_cen\_first\_box" and "lambda\_cen\_first\_box" parameters. Goes to the frame at the middle of the time range and gets OBS\_L0 and OBS\_B0, it then tracks forward and backward in time to beginning and end of the cube at the rate specified by the tracking coefficients (a0, a2, a4) at OBS\_B0.
- **eqcentre:** Tracks centred on the meridian at the equator at the middle time. Ignores the "phi\_cen\_first\_box" and "lambda\_cen\_first\_box" parameters. Goes to the frame at the middle of the time range and gets OBS\_L0, it then tracks forward and backward in time to beginning and end of the cube at the rate

specified by the tracking coefficients (a0, a2, a4) at the equator (ie. latitude is 0 degrees).

- **norm:** this is 'normal' tracking. It tracks at the latitude specified in "lambda\_cen\_first\_box" centred at the middle frame forward and backward in time to beginning and end of the time series OF THE FIRST CUBE. It does NOT check if it is on the visible side of the Sun.
- **fixed:** tracked at the Carrington rotation rate centred at the given co-ordinates.

The rotation rate is determined for the specified latitude given in the input parameter file `lambda_cen_firstbox` which is the central latitude of the middle frame of the first datacube.

The first file after the time given in the input parameter file for `t_start` is the first frame to be mapped.

The DRMS is queried for the cadence of the input series. The tracking is then calculated for each time step at that cadence (60 or 45 seconds).

### 3 Defining the Time Computational Grid

– The coordinates that define the 3D spacetime are time,  $t$ , the latitude  $\lambda(t)$ , and the longitude,  $\phi(t)$ .

1) The input data for the tracking remapping consists of a long sequential time series of fd MDI data. This already enforces the grid spacing in time to be approximately 60s. We could also include other grid spacings, but this would involve a lot of interpolation in time which is not the best for scientific analysis. The grid spacing in time for MDI data also dictates that the points to be tracked around must also lie on an MDI file and then the time at which this point is to be tracked forward and backwards must also start/end on an MDI file.

So the grid must have a spacing of 60s (approximately MDI time, we can also put a check to make sure it is 60 seconds as sometimes this may be out). We denote  $T$  as the total duration for the MDI observations to be tracked/remapped and  $N_T$  is the number of MDI files. The user must input a start-time and end-time. It is important that these times correspond precisely to the times in the MDI data files. We will put a check to ensure this. Users can use the DRMS to search for the data they are interested in and find this information. Thus, the total time is automatically related to the number of points according to  $T = dt (N_T - 1)$  where

for MDI data  $dt = 60\text{s}$ . In the future this will easily work for other data series such as HMI and GONG. The computational time labels for each MDI file is denoted

$$t^i = t^0 + i dt \quad \text{where} \quad 0 \leq i \leq N_T - 1 \quad (1)$$

The user needs to specify the initial MDI time,  $t^0$ , (or get this from the input MDI file itself resulting from a DRMS search), the tracking duration  $\delta t$ , and the time between each point to be tracked over,  $\Delta t$ . These quantities automatically determine the time-overlap (+ive i.e. there is overlap, 0 no overlap, and -ive there is a gap). The precise end time depends heavily on all these quantities, thus this is something the user cannot specify precisely. However, the user can specify an approximate total time to perform the calculation and the code will search for the nearest precise MDI time that can accommodate all the requirements.

From this we determine the number of grid points used for each individual tracking,  $N_{\text{Trac}}$  (will be odd). How many grid points in time each tracking cube is overlapping with other cubes is denoted  $N_{\text{ol}}$ . This can be positive if they do not overlap, 0 if they match perfectly, negative if they do overlap. It is given by

$$\begin{aligned} N_{\text{ol}} &= \frac{\Delta t - \delta t}{dt} \\ &= N_{\text{spg}} - 1 - (N_{\text{trac}} - 1) \\ &= N_{\text{spg}} - N_{\text{trac}} \end{aligned}$$

### 3.1 Specifying mapping intervals

The times at which the  $N_{\text{trac}}$  points will be tracked about are given by

Each tracking interval is defined by

$$m(N_{\text{spg}} + N_{\text{trac}}) \leq i \leq m(N_{\text{spg}} + N_{\text{trac}}) + (N_{\text{trac}} - 1), \quad \text{where} \quad 0 \leq m \leq N_{\text{trac}} - 1$$

Correspondingly the midpoint of each interval is

$$i = m(N_{\text{spg}} + N_{\text{trac}}) + (N_{\text{trac}} - 1)/2, \quad \text{where} \quad 0 \leq m \leq N_{\text{trac}} - 1.$$

(tracking region must have odd number of points in time)

## 4 Grid in Space

1) Let the mid-point of the first tracking interval be the reference time,  $t_{\text{ref}}$ , at which a uniform reference grid is defined. The reference grid is sliced into regions in both latitude  $N_\lambda$ , and longitude  $N_\phi$ , which are both specified by the user. The latitude runs from -90 to 90, whereas the longitude runs from 0 to 360. The overlap in both longitude and latitude are also specified by the user. This must be given in percentage (of area or length?). The distance between each latitude and longitude center points respectively are

$$\begin{aligned}\Delta\lambda &= \pi/N_\lambda \\ \Delta\phi &= 2\pi/N_\phi.\end{aligned}$$

Thus, the lat/long centers of each region of this reference frame are given by

$$\begin{aligned}\lambda_k^{\text{ref}} &= -\pi/2 + \Delta\lambda/2 + k \Delta\lambda & \text{where} & \quad 0 \leq k \leq N_\lambda - 1 \\ \phi_j^{\text{ref}} &= \Delta\phi/2 + j \Delta\phi & \text{where} & \quad 0 \leq j \leq N_\phi - 1\end{aligned}$$

The tracking code will generate a file to be later passed to the mapmdi code. Each line of this file is of the form

xxx.FITS, L0, B0, dx, dy, nx, ny

where L0 and B0 are the lat/long to be mapped about. dx and dy are user input and must be an input to the tracking code. nx and ny are usually a user input for the mapmdi code. However, here the tracking code calculates these quantities based on the number of regions specified in lat/long.

input from the user to the tracking code must therefore be

- $dx$ : degrees/pixel in the resultant mapped region, in the longitudinal direction; user input for tracking code then passed directly to mapmdi
- $dy$ : degrees/pixel in the resultant mapped region, in the latitudinal direction; user input for tracking code then passed directly to mapmdi
- $n_x$ : integer number of full pixels in the resultant mapped region, in the longitudinal direction (does this account for rotations in SOHO for example, done so using the header)
- $n_y$ : integer number of full pixels in the resultant mapped region, in the latitudinal direction.

- overlap: the percentage in length ( $o_x$  and  $o_y$  in the  $x$  and  $y$  directions respectively) that the regions overlap with respect to total length
- $z_{x/y}$ : in percentage, it is the extension (in both x/y directions respectively) required to the non-overlapping regions where  $z_{x/y} = 0$ .
- $n_x = \frac{\Delta\phi}{dx}(1 + 2z_x)$
- $n_y = \frac{\Delta\lambda}{dy}(1 + 2z_y)$

The relationship between  $z_{x/y}$  and the overlap is

$$z_{x/y} = \frac{o_{x/y}}{2(1 - o_{x/y})} \quad \text{where} \quad o_{x/y} \neq 1 \quad (2)$$

for  $x$  and  $y$  directions respectively.

For example, a user can specify a  $dy$  of 1 degree per pixel in the resultant mapped file. Then if the user also specified tracking input as three regions for the latitudinal region, then each region has 60 degrees each and are separated by  $\Delta\lambda = 60$ . Thus, the tracking code will calculate that 60 pixels are needed to accomplish this. According to the formula above we have

$$n_y = \frac{\Delta\lambda}{dy} = 60/1 = 60 \quad (3)$$

In the case where this is not a perfect integer, we should round this up to the nearest integer (we could also check that if a square is requested, then it remains a square). So, following on from the last example, a user could choose a high resolution of  $dy=0.5$  degrees per pixel and since the number of regions is still 3, then it would take  $n_y$  of 120 pixels to achieve this. We need to also give a blank file with dimensions  $n_x n_y$  and also with  $dx/dy$  information in its header.

The overlap factor must also be an input to the tracking code

If there is zero overlapping (in space), then the boundaries of all regions will align perfectly. If negative overlapping is requested (i.e. space between each region) we just need to create the corresponding parameters and pass these to Charlie's code. However, if positive overlapping is requested, i.e. regions overlap in some way, then this needs a careful treatment.  $dy$  and  $dx$  are the scaling (degrees/pixel) of the final mapped observations., the overlap is given by  $o_x$  and  $o_y$  in percent.

## 5 Mapping

Code name: `drms_mapping.c`

Call sequence: `drms_mapping [-gdcmv] inlist=... outlist=...`

`-g`: geometry flag

`-d`: doppler flag

`-c`: continuum flag

`-m`: magnetogram flag

`-v`: verbose

`inlist (string)`: input list containing files to be mapped  
(no default)

`outlist (string)`: output list containing sequential filenames of the mapped patches

default value: `cube.list`

The ‘inlist’ contains the DRMS series; T\_REC of the data frame; the segment name; the id of the datacube this map will belong to; the central longitude of the map; the central latitude of the map; offset of the x-centre of the postel map in pixels; offset of the y-centre of the postel map in pixels; number of pixels in the x-direction for the resulting map; number of pixels in the y-direction for the resulting map; x-axis scale (radians per pixel); y-axis scale (radians per pixel); name of the output file; type of mapping.

e.g.

`hmi.v_45s[2011.05.04_11:23:15_TAI] Dopplergram 1`

`67.47873934450685 0.0 0.0 0.0 256 256 5.0E-4 5.0E-4 map0_mj14_cube0_wfid1.fits`  
`-postel`

The ‘outlist’ at this stage is the name of the file that the names of the mapped fits files are written to.

- Opens input list containing all relevant mapping details and output list (see above). If either does not exist, exists.
- Reads in keywords from the header of the file (can deal with either MDI or HMI). XX list necessary keywords XX
- Reads in the full data array.
- Measures the solar rotation from this array.

- Removes the solar rotation by fitting a two-dimensional plane to the array.
- Magnify the array if desired (this is not implemented in the pipeline as it takes up significant memory) onto a finer grid, using Fourier interpolation. This reduces the effects of aliasing and the Moire effect.
- The array is then padded reducing the effects of truncating at the limb when the Fourier interpolator is used.
- Loops over the number of patches to be mapped in one array.
- Compute the Postel projected maps (perpendicular and line-of-sight projections are available but not tested) for each central latitude and longitude, and write out.
- If Carrington longitude and latitude of each point in the Postel map is required (given by the `-g` parameter) it is computed and written out.

## 5.1 Postel projection

The function `postel` in `drms_mapping` computes the Postel projection.

First the array is rotated so that the specified central latitude and longitude of the resulting Postel map is at the centre (see Appendix A). This takes into account the P-angle and the B-angle.

Then the array is mapped from the CCD coordinates to the Postel map coordinates (see Appendix C).

## 6 Creating the datacube

This reads the header information from the first mapped input file and writes this information to the header for the datacube. XX This should be corrected, so that it takes data from the *middle* frame XX.

## A Define the rotation matrices.

The co-ordinates are defined with the  $x$  and  $y$  axis in the plane of the image ( $x$  horizontal and to the right and  $y$  vertically upwards) with the  $z$ -axis coming out of

the plane. We want to rotate the axis so that a point  $C$  with longitudinal distance from the meridian and latitude co-ordinates given by  $l$  and  $b$  respectively (i.e.  $l = L0 - l_c$  where  $l_c$  is the actual Carrington longitude of point  $C$ ).

First rotate about the  $z$ -axis to correct for the P-angle:

$$R_z(P) = \begin{bmatrix} \cos P & \sin P & 0 \\ -\sin P & \cos P & 0 \\ 0 & 0 & 1 \end{bmatrix} \quad (4)$$

Then rotate by angle of  $\pi/2 - B0$  about the  $x$ -axis to align the solar North pole with the  $z$ -axis using:

$$R_x(\pi/2 - B0) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos(\pi/2 - B0) & \sin(\pi/2 - B0) \\ 0 & -\sin(\pi/2 - B0) & \cos(\pi/2 - B0) \end{bmatrix} \quad (5)$$

to get

$$R_z(P)R_x(\pi/2 - B0) = \begin{bmatrix} \cos P & \sin P \cos(\pi/2 - B0) & \sin P \sin(\pi/2 - B0) \\ -\sin P & \cos P \cos(\pi/2 - B0) & \cos P \sin(\pi/2 - B0) \\ 0 & -\sin(\pi/2 - B0) & \cos(\pi/2 - B0) \end{bmatrix}. \quad (6)$$

Then rotate about the  $z$ -axis to align with the longitudinal distance of the point  $C$  from the meridian (which lies along the  $zy$ -plane),  $l$  using:

$$R_z(l) = \begin{bmatrix} \cos l & \sin l & 0 \\ -\sin l & \cos l & 0 \\ 0 & 0 & 1 \end{bmatrix} \quad (7)$$

to get

Then rotate about the  $x$ -axis to align with the latitude of point  $C$  using:

$$R_x(b - \pi/2) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos(b - \pi/2) & \sin(b - \pi/2) \\ 0 & -\sin(b - \pi/2) & \cos(b - \pi/2) \end{bmatrix} \quad (8)$$

to get

$$R = R_z(P)R_x(\pi/2 - B0)R_z(l)R_x(b - \pi/2) \quad (9)$$

where the components of  $R$  are

$$\begin{aligned}
R_{xx} &= \cos l \cos P - \sin l \sin P \cos(\pi/2 - B0) \\
R_{xy} &= \cos(b - \pi/2) (\sin l \cos P + \cos l \sin P \cos(\pi/2 - B0)) \\
&\quad - \sin(b - \pi/2) \sin P \sin(\pi/2 - B0) \\
R_{xz} &= \sin(b - \pi/2) (-\sin l \cos P + \cos l \sin P \cos(\pi/2 - B0)) \\
&\quad + \cos(b - \pi/2) \sin P \sin(\pi/2 - B0) \\
R_{yx} &= -\sin P \cos l - \sin l \cos P \cos(\pi/2 - B0) \\
R_{yy} &= \cos(b - \pi/2) (-\sin l \sin P + \cos l \cos P \cos(\pi/2 - B0)) \\
&\quad - \sin(b - \pi/2) \cos P \sin(\pi/2 - B0) \\
R_{yz} &= \sin(b - \pi/2) (-\sin l \sin p + \cos l \cos P \cos(\pi/2 - B0)) \\
&\quad + \cos(b - \pi/2) \cos P \sin(\pi/2 - B0) \\
R_{zx} &= \sin l \sin(\pi/2 - B0) \\
R_{zy} &= -\cos(b - \pi/2) \cos l \sin(\pi/2 - B0) - \sin(b - \pi/2) \cos(\pi/2 - B0) \\
R_{zz} &= -\sin(b - \pi/2) \cos l \sin(\pi/2 - B0) + \cos(\pi/2 - B0) \cos(b - \pi/2)
\end{aligned} \tag{10}$$

which simplifies to

$$\begin{aligned}
R_{xx} &= \cos l \cos P - \sin l \sin P \sin B0 \\
R_{xy} &= \sin b (\sin l \cos P + \cos l \sin P \sin B0) \\
&\quad + \cos b \sin P \cos B0 \\
R_{xz} &= -\cos b (-\sin l \cos P + \cos l \sin P \sin B0) \\
&\quad + \sin b \sin P \cos B0 \\
R_{yx} &= -\sin P \cos l - \sin l \cos P \sin B0 \\
R_{yy} &= \sin b (-\sin l \sin P + \cos l \cos P \sin B0) \\
&\quad + \cos b \cos P \cos B0 \\
R_{yz} &= -\cos b (-\sin l \sin p + \cos l \cos P \sin B0) \\
&\quad + \sin b \cos P \cos B0 \\
R_{zx} &= \sin l \cos B0 \\
R_{zy} &= -\sin b \cos l \cos B0 + \cos b \sin B0 \\
R_{zz} &= \cos b \cos l \cos B0 + \sin B0 \sin b
\end{aligned} \tag{11}$$

and can be re-written as

$$\begin{aligned}
R_{xx} &= \cos l \cos P - \sin l \sin P \sin B0 \\
R_{xy} &= \sin b \sin l \cos P + \sin P (\cos b \cos B0 + \cos l \sin B0 \sin b) \\
R_{xz} &= + \cos b \sin l \cos P + \sin P (\sin b \cos B0 - \cos b \cos l \sin B0) \\
R_{yx} &= - \sin P \cos l - \sin l \cos P \sin B0 \\
R_{yy} &= - \sin b \sin l \sin P + \cos P (\cos b \cos B0 + \cos l \sin B0 \sin b) \\
R_{yz} &= \cos b \sin l \sin P + \cos P (\sin b \cos B0 - \cos b \cos l \sin B0) \\
R_{zx} &= \sin l \cos B0 \\
R_{zy} &= - \sin b \cos l \cos B0 + \cos b \sin B0 \\
R_{zz} &= \cos b \cos l \cos B0 + \sin B0 \sin b
\end{aligned} \tag{12}$$

where the **red** symbols indicate a different sign to “map\_mdi.c”. The Eqns. 12 (other than the incorrect signs) appear in the `postel` function in the `drms_mapping` code.

## B The oblate Sun

There is also some scaling to account for the Sun not being a perfect sphere using the semi-major and semi-minor axis. Let the semi-major axis of the Sun be  $R_{\text{major}}$  (and  $\psi$  in radians) and the semi-minor axis be  $R_{\text{minor}}$  in pixels. These are given in the header of the images.

$$S_{xx} = 0.5 ((R_{\text{major}} + R_{\text{minor}}) - (R_{\text{major}} - R_{\text{minor}}) \cos(2\psi)) \tag{13}$$

$$S_{xy} = 0.5 ((R_{\text{major}} - R_{\text{minor}}) \sin(2\psi)) \tag{14}$$

$$S_{yx} = 0.5 ((R_{\text{major}} - R_{\text{minor}}) \sin(2\psi)) \tag{15}$$

$$S_{yy} = 0.5 ((R_{\text{major}} + R_{\text{minor}}) + (R_{\text{major}} - R_{\text{minor}}) \cos(2\psi)) \tag{16}$$

$$\tag{17}$$

## C Postel to CCD equations

We begin with the Postel map we want to create. The horizontal and vertical coordinates on in the Postel projection are  $\xi$  and  $\eta$  in units of heliocentric radians. The pixel scale is  $P$  in units of radians per pixel. Use  $\rho$  as the distance from the center of the map (again heliocentric radians) and  $\phi$  as the angle counter-clockwise

from the  $\xi$  axis:

$$\rho = \sqrt{\eta^2 + \xi^2}, \quad (18)$$

$$\phi = \arctan(\eta/\xi). \quad (19)$$

Let  $m = \rho/P$  be  $\rho$  in units of pixels.

We need to find the where a point with Postel coordinates  $\rho$  and  $\phi$  is located on the CCD. On the CCD let the horizontal direction be  $x$  and the vertical direction be  $y$ .

$$r = \sqrt{x^2 + y^2}, \quad (20)$$

$$\phi = \arctan(y/x). \quad (21)$$

The angle  $\phi$  is the same on the CCD as the Postel. Need to determine where  $\rho$  is on the CCD i.e.  $r(\rho)$ .

$$\frac{r}{f} = \tan \alpha = \frac{R_\odot \sin \rho}{D - R_\odot \cos \rho} \quad (22)$$

$$r = f \frac{R_\odot \sin \rho}{D - R_\odot \cos \rho}, \quad (23)$$

where  $\alpha$  is the angle between the sub-observation point, the observer and the coordinate to be mapped. Notice that we haven't yet said what the focal length  $f$  is. We will come back to this.

Thus the point on the Sun given by  $\rho$  and  $\phi$  is located on the CCD at the point:

$$x = r(\rho) \cos \phi \quad (24)$$

$$y = r(\rho) \sin \phi. \quad (25)$$

Define  $\alpha_{\max}$  as the angle between the sub-observation point, the centre of the CCD and the observed limb (see Figure 3). Use  $r_\odot$  as the distance to limb on the CCD. From looking at the figure, we see  $\sin \alpha_{\max} = R_\odot/D$ . The size of the Sun on the CCD is  $r_\odot = f \tan \alpha_{\max}$ . Use  $Y$  and  $Z$  is the dimensionless coordinates from Figure 3.

Then put into above equations:

$$r/r_\odot = \frac{\cos \alpha_{\max}}{\sin \alpha_{\max}} \frac{Y}{1/\sin \alpha_{\max} - Z} \quad (26)$$

$$r/r_\odot = \frac{Y (1 - \sin^2 \alpha_{\max})^{1/2}}{1 - \sin \alpha_{\max} Z} \quad (27)$$

$$r/r_\odot = p(Z)Y. \quad (28)$$

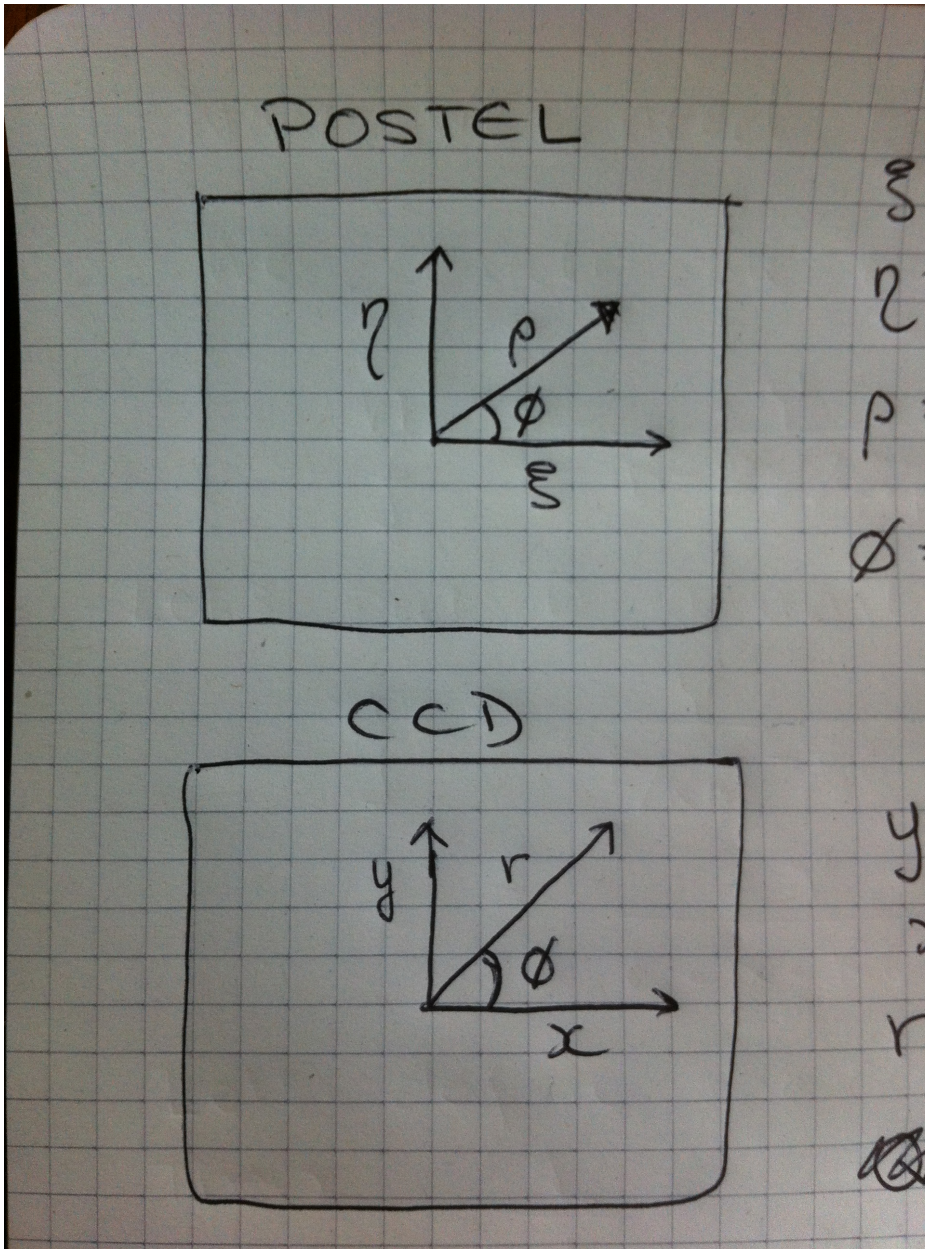


Figure 1: Defining the coordinates on the Postel map and the CCD.

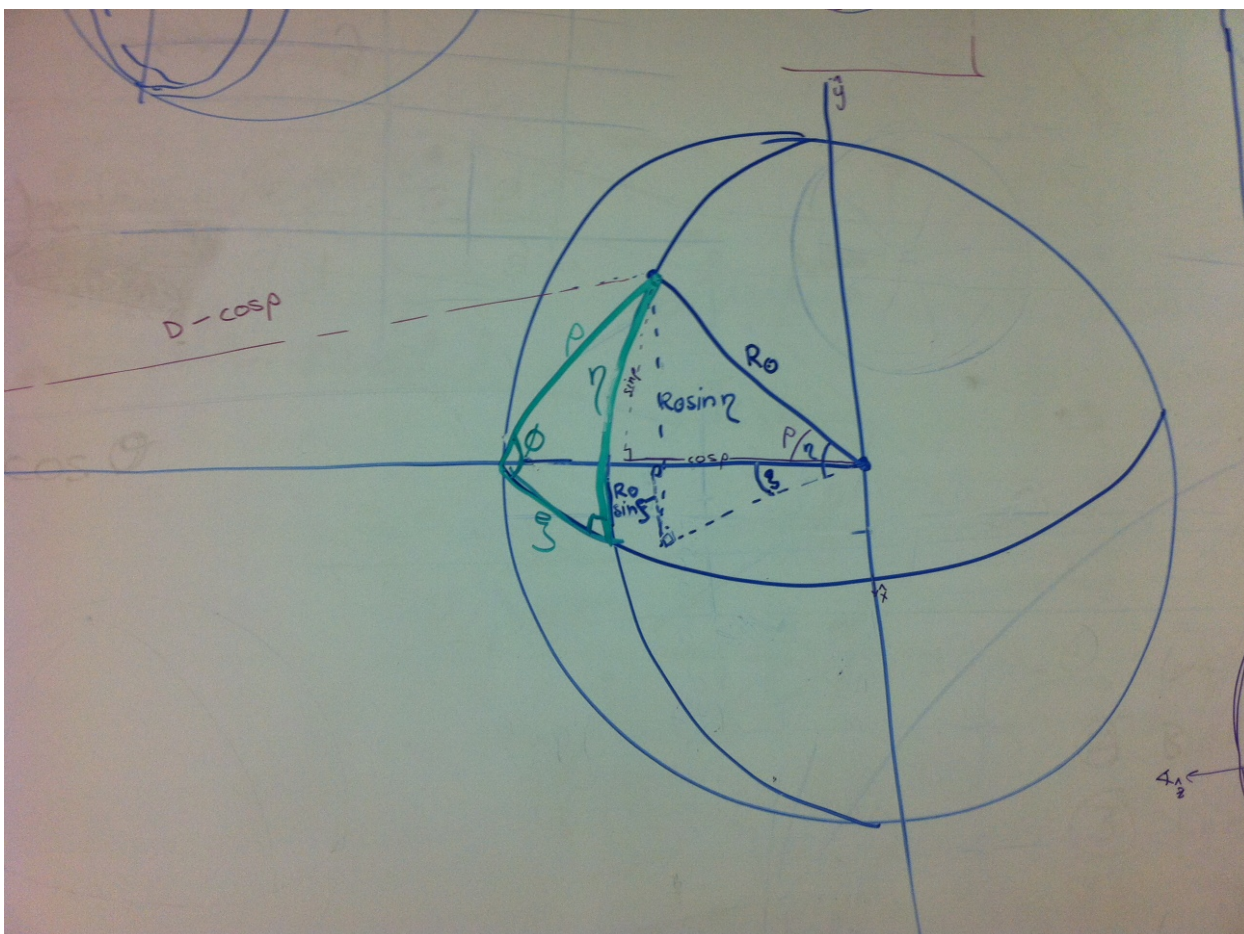


Figure 2: To illustrate the three-dimensional coordinates for the Postel projected map.

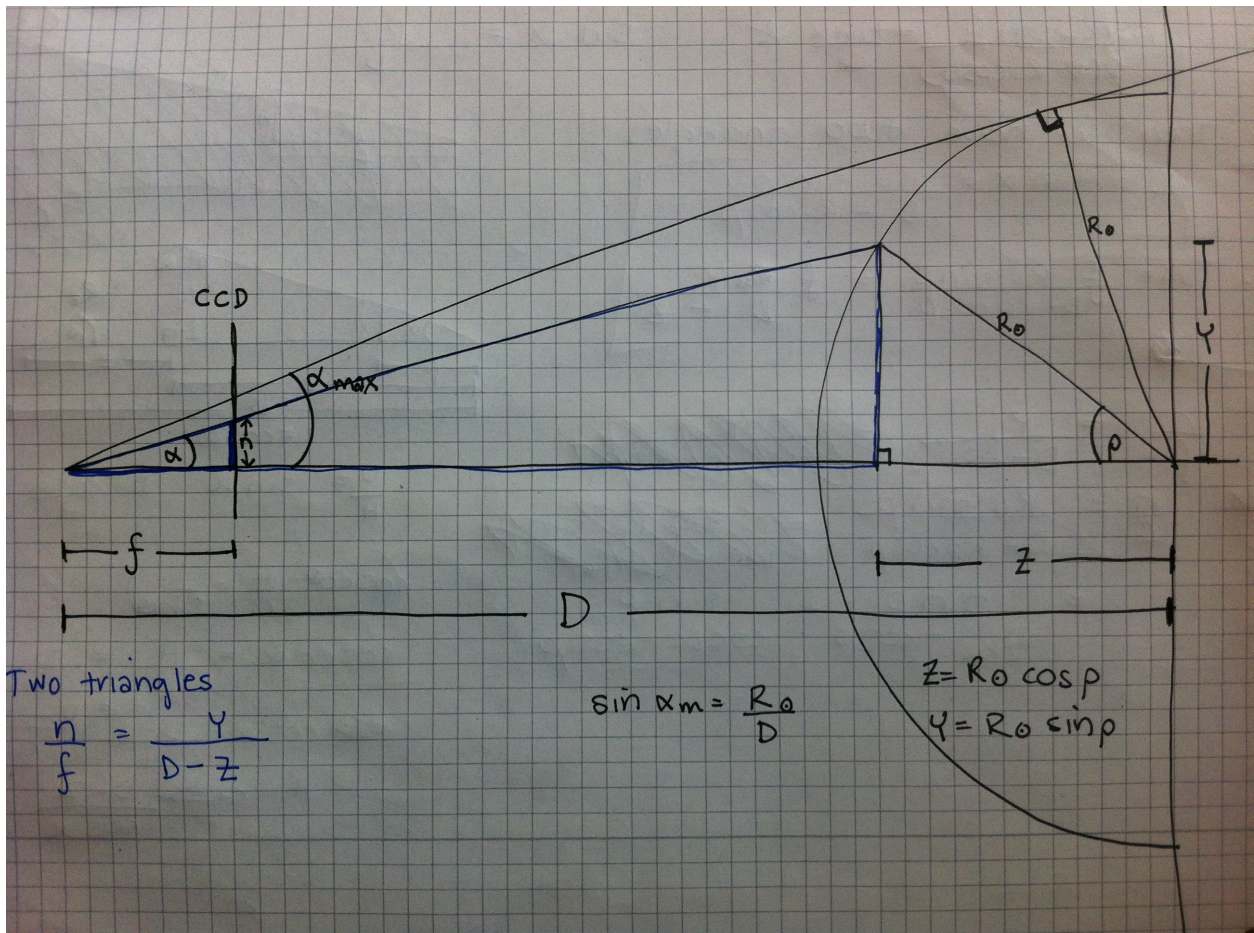


Figure 3: The geometry involved in deriving the scaling factor (Charlie calls it parallax) due to the finite distance from the Sun. The similar triangles with which we equate the ratio of the sides are outlined in blue.

Define “parallax” factor  $p(Z) = (1 - \sin^2 \alpha_{\max})^{1/2} / (1 - \sin \alpha_{\max} Z)$ .

## C.1 Postel projection

Insert the Postel equations here and more derivation.

Define the Postel map coordinates,  $\eta$ ,  $\xi$ ,  $\phi$  and  $\rho$  as before in units of radians (see Figure 1) i.e.  $\phi = \arctan(\eta/\xi)$  and  $\rho = \sqrt{\xi^2 + \eta^2}$ .

Then

$$\zeta = \cos \rho \quad (29)$$

$$\rho = \sin \rho \quad (30)$$

notice that  $\rho$  is redefined! So that

$$\xi_{\text{new}} = \rho \cos \phi \quad (31)$$

$$\eta_{\text{new}} = \rho \sin \phi \quad (32)$$

which is exactly taken from the `drms_mapping` code.

## C.2 Final equations

What appears in the code are as follows. Rotate about the  $z$ -axis.

$$z = (R_{zx}\xi_{\text{new}} + R_{zy}\eta_{\text{new}} + R_{zz}\zeta) \quad (33)$$

Then rotate about the  $x$  and  $y$  axis and scale for the parallax.

$$x_t = (R_{xx}\xi_{\text{new}} + R_{xy}\eta_{\text{new}} + R_{xz}\zeta) \frac{(1 - \sin^2 \alpha_{\max})^{1/2}}{(1 - \sin \alpha_{\max} z)} \quad (34)$$

$$y_t = (R_{yx}\xi_{\text{new}} + R_{yy}\eta_{\text{new}} + R_{yz}\zeta) \frac{(1 - \sin^2 \alpha_{\max})^{1/2}}{(1 - \sin \alpha_{\max} z)} \quad (35)$$

and then account for any oblateness

$$x = x_0 + S_{xx}x_t + S_{xy}y_t \quad (36)$$

$$y = y_0 + S_{yx}x_t + S_{yy}y_t \quad (37)$$

where  $x_0$  and  $y_0$  are where the centre of the CCD is defined in pixels. These equations are what appear in the `postel` function in the `drms_mapping` code.

Intermediate results for rotation matrices, for debugging:

$$\begin{aligned}
& R_z(P)R_x(\pi/2 - B)R_z(l) \\
&= \begin{bmatrix} \cos l \cos P - \sin l \sin P \cos(\pi/2 - B) & \sin l \cos P + \cos l \sin P \cos(\pi/2 - B) & \sin P \sin(\pi/2 - B) \\ -\sin P \cos l - \sin l \cos P \cos(\pi/2 - B) & -\sin l \sin P + \cos l \cos P \cos(\pi/2 - B) & \cos P \sin(\pi/2 - B) \\ \sin l \sin(\pi/2 - B) & -\cos l \sin(\pi/2 - B) & \cos(\pi/2 - B) \end{bmatrix}
\end{aligned} \tag{38}$$