Mpole Reference

Montana State University

May 25, 2006

Matthew Sos & Dana Longcope

Version 2.4

Contents

Chapter 1

Introduction

The Mpole package is a collection of IDL routines to implement the Magnetic Charge Topology (MCT) models and the Minimum Current Corona (MCC) model.[2,4] The package includes numerical algorithms to find and manipulate key elements of the topology of such models, such as magnetic null points, separatrix surfaces, and separator field lines. There are also routines for data visualization and output of results. The details of these models are presented in the references. In particular, the examples presented in this manual are those found in reference [4].

1.1 Finding and Visualizing Poles

In magnetic charge topology models the coronal field is determined from a set of point charges. The charges may be a purely theoretical construction, or an approximation of an observed photospheric field. The complete set of charge positions and strengths (fluxes) are contained in a *poles* data structure, along with other data pertaining to the set as a whole. The prefered method of creating the structure is to read it from a file. There is a program `rd_pns` which reads ASCII files with a specific format, column formatted text with special tags identifying each section. It is tradition to give such files the extension `.pns`, for *poles-nulls-separators*, since the file format permits the specification of all the basic topological features of MCT. Because it is an ASCII file it may be created simply with a text editor, although it is usually necessary to do this only for the poles. An example data file containing only poles appears in example 1.1.0.1. This example is the one analyzed in published articles on the MCC.[3,4] (Lines beginning with % are skipped; they are used here as comments for our own information.) The command

```
rd_pns, 'example.pns', pls
```

Example 1.1.0.1 A simple data file containing six magnetic sources.

```
BEGIN POLES
% lab      x      y      z     Phi
   P1    0.75   0.0    0.0    1.0
   P2     0.3   0.9    0.0    0.5
   P3   -0.75   0.6    0.0    0.25
   N4    -0.9   0.0    0.0   -0.8
   N5    0.75  -0.45   0.0   -0.25
   N6    -1.5   0.75   0.0   -0.7
END POLES
```

will read the sources from a file `example.pns` into the variable `pls`, replacing the variable's existing contents. As with any IDL variable, its structure can be examined using

```
help, pls, /struct
```

Once the structure of poles has been defined, the pole locations may be plotted using

```
show_poles, pls
```

1.2 Finding and Visualizing Magnetic Null Points

Magnetic nulls, (i.e. points of zero magnetic field) may be found for a given charge configuration. The procedure `all_nulls` is intended to locate all of the null points for a given `poles` structure.

```
nls = all_nulls( pls )
```

will locate the nulls associated with the poles structure `pls`, placing them in the structure-array `nls`. IDL will prind the following cryptic statements on the screen

```
Sources:
     3 positive,    3 negative
Nulls:
     2 positive (B),    2 negative (A)
     4 prone
 --------------
2d Poincare index:
  CHECKS OK
3d Poincare index:
  CHECKS OK
```

Their meaning will be explained below.

The structure containing the nulls can be examined using

```
help, nls, /struct
```

while

```
show_nulls, nls
```

will draw their locations on the current plot.

The number of null points in a field, even a simple one, is difficult to know in advance. One theorem,[4] which goes a type of *Poincaré index theorem*, states that that N charges must have at least $N-2$ nulls (at least $N-1$ if the total charge of the set is not zero). There are still more elaborate theorems vailable, a summary of which appear in [1]. The program `index_check` runs through a set of these and prints the diostics to the screen. It is invoked by typing

```
index_check, pls, nls
```

This program is invoked automatically by `all_nulls`, which produces the printout described above.

The `index_check` program counts the number of sources of each sign and the number of nulls of each type. These counts are displayed, and compared to the predictions of two different index theorems. If either of the index theorem gives an unxpected result (i.e. is not `OK`) `index_check` will make its best guess about which nulls are missing. It is often the case that some nulls elude `all_nulls` in spite of its ambitious name. The program `xadd_nulls` permits you to add missing nulls, provided you can point to an approximate location with the cursor.

Armed with a set of poles and of nulls it is possible to graph the footprint of the field's skeleton[4] using the command

```
show_fp, pls, nls, xr=[-2,2], yr=[-2,2]
```

Note how the keywords `xr` and `yr` force a less claustrophic frame for the plot.

1.3 Visualizing Separatrix Surfaces

The fan surfaces of the nulls form the skeleton of the field, dividing it into its domains. To plot the separatrix surface from one or more particular null atop an exisiting plot (say of the footprint) type

```
show_sepx, pls, nls[0]
show_sepx, pls, nls[1]
```

Each command draws sample field lines from the separatrix surfaces that particular null.

To see a perspective view of this piece of the field's skeleton you must use 3d versions of some of the programs, and use `/threed` options of others. Typing the commands

```
show_3dpoles, pls
show_3dnulls, nls
show_sepx, pls, nls[0], /threed
```

accomplishes this.

1.4 Finding and Visualizing Separators

The separator line itself can be found using the function `all_seprs()`, which returns all separators associated with a given configuration of poles and nulls. (Once again, this program is quite fallible, and many separators will elude its search.)

```
spr = all_seprs( pls, nls )
```

Though finding separators is computationally expensive, plotting them is not. The command

```
show_sepr, pls, nls, spr
```

will draw the separator lines in `spr` over the current plot. (If you don't have an exisiting plot, you may create one by typing `show_fp, pls, nls`.)

The properties of the separators,[2,5] such as enclosed flux, length, and fiducial current $I^\star$, may be computed using the command

```
si = sepr_info( pls, nls, spr, /str )
```

This creates a structure called `si` whose fields contain the information.

1.5 Connectivity and the Domain Matrix

The connectivity of the field can be represented by the domain matix or adjacency matrix D_{ij}. This is represented by an $(N+1) \times (N+1)$ array `dm[i,j]`. The fastest way to calculate this is the program `domain_matrix`

```
dm = domain_matrix( pls, nls )
```

In this matrix row and column with index `0` represents ∞ (distant sources) so the connection between poles `0` and `3`, charges $P1$ and $N4$, is to be found in `dm[1,4]`. This is 1 in the example, indicating that the sources are connected. The matrix may be represented graphically over an exisiting plot of the poles by typing

```
show_dm, pls, dm
```

A concise summary table can be printed on the screen by typing

```
dm_print, pls, dm
```

This results in the tabular array printed to the screen

```
     N4  N5  N6
P1    1   1   1
P2    1   0   1
P3    1   0   1
```

showing that $N4$ and $N6$ each connect to all positive sources, while $N5$ connects only to $P1$.

A slower and less accurate, but more straight-forward alternative is to use Monte-Carlo integration: begin random field lines from each source and see where they end up. This is done by the typing

```
cm = connectivity( pls )
```

The connecivity matrix, `cm`, contains floating-point values which are the approximation of the flux in each domain. The connectivity can be listed in tabular form by invoking the command

```
cm_list, cm, pls
```

This print something approximating the following summary to the screen

```
Largest connection by source
                    flux    frac.  deg.  col. error
    P1 -->     N4    0.554   56.9%   3    -2.7324%
    P2 -->     N6    0.389   75.7%   2     2.9173%
    P3 -->     N4    0.152   61.5%   2    -1.2236%
    N4 -->     P1    0.554   66.7%   3     3.7989%
    N5 -->     P1    0.260  100.0%   1     4.1244%
    N6 -->     P2    0.389   60.5%   3    -8.0713%

Largest connections
                    flux    frac.   cummulative
    P1 -->     N4    0.554  31.9176%  31.9176%
    P2 -->     N6    0.389  22.4575%  54.3751%
    P1 -->     N5    0.260  15.0104%  69.3855%
    P1 -->     N6    0.159   9.1597%  78.5452%
    P3 -->     N4    0.152   8.7502%  87.2954%
    P2 -->     N4    0.125   7.2153%  94.5107%
    P3 -->     N6    0.095   5.4893% 100.0000%
```

The actual numbers will always be slightly different since the `connectivity` uses a random number generator in its Monte Carlo calculation of the connectivity matrix. The top segment is a list of all sources along with the source to which it shares the plurarility of its flux. The bottom provides a list of connections in order of decreasing size. If there are very many connection only the top 25 are shown (or some other number set by keyword).

If you have a structure, `spr`, containing *all* of the separators it is possible to use it to find the domain matrix:

```
dm = domain_matrix( pls, nls, spr )
```

This is the fastest method possible, however, it will omit domains which are "leaves" in the domain graph. Calling it with the `\leaf` keyword set will include these, but at the cost of slightly longer run time. In general this longer time is still considerably less than running `domain_matrix` without the separator, so it is a nice compromise. Recall, however, that if some separators are missing from `spr` then this version cannot hope to provide you with a complete list of domains.

The domain graph is plotted, in schematic form, by the function

```
dm_schematic, pls, dm
```

The function `null_graph` will plot the null graph, showing the nulls and separators. A side-by-side plot of both the domain graph and null graph gives a concise summary of the field's connectivity:

```
sum_graph, pls, nls, spr
```

In addition to theorems concerning the numbers of null points there are also theorems relating the numbers of separators and numbers of domains.[1] Calling `index_check` with the separator structure as the third argument will print the old assessment of the null count and then add the text

```
Euler Charcateristic:
  D =    7 domains
   1 unbroken fans
   ==>    0 coronal domains
```

It has used the numbers and types of sources, nulls and separators to determine that there should be $D = 7$ different domains present. The result of `sum_graph` just produced seems to indicate that there are only six. The problem here is the absence of "leaf" domains in the plot: they are always counted in the Euler characteristic. Leaf domains can be included in the plot in two ways. Either by using the keyword `\leaf`

```
sum_graph, pls, nls, spr, /leaf
```

or by including the domain matrix, `dm`, as a fourth argument. This second method will only work, however, if the domain matrix has been calculated in a way which will include leaf domains.

1.6 Tracing Field Lines

Individual magnetic field lines may be found from any charge configuration, given an initial starting point using `fl_from_point`. If you have less specific desires you can choose to draw a random selection of field lines from a sepcified domain, say the one connecting pole 0 `P1` to pole 3 `N4` by typing

```
show_domain, pls, 0, 3
```

On an existing 3d plot you can draw a sample of all field lines by typing

```
show_random_3dlines, pls
```

To trace a single field line from an initial point, say $(x, y, z) = (0, 0, 0.1)$, type

```
fl = fl_from_point( pls, [0,0,0.1] )
```

will calculate the field line originating from the point in the charge configuration `pls`.

As field lines are simply arrays of vertices, they can be displayed using the IDL direct graphics routines. For example, the command

```
PLOTS, fl(0,*), fl(1,*), fl(2,*), /t3d
```

will draw the field line `fl` on an exisiting 3d plot.

1.7 The Viewing Transformation

In Magnetic Charge Topology models, the photosphere is represented as the plane at $z = 0$. Observed photospehric fields must be modeled in the *tangent plane* approximation. The results may be mapped onto the sky using a viewing transformation contained in a structure `view`. This structure provides the transformation which takes points in the tangent plane, (x, y) are Megameters (North, West) from the point of tangency, to the plane of the sky, (x, y) are arc-seconds (North, West) of disk center. The command

```
view = view_xform( 20.0, 30.0, /degrees )
```

establishes a transformation which puts the origin of `pls` at 20°N and 30°W on the solar disk. To view the sources as they appear in the sky type

```
show_poles, pls, view=view
```

To see the same region one hour later type

```
show_poles, pls, view=solar_rotate_view( view, 3600.0 )
```

A common situation is to have a set of poles whose locations were determined from a magnetogram. When located from a magnetogram the poles have (x, y) coordinates in the plane of the sky, and an implicit z-ccordinate (along the line-of-sight) which would place them on the solar surface. Provided (x, y) are in arcseconds from disk center the command

```
view = disk2tan_plane( pls )
```

calculates the appropriate viewing transformation *and changes the values in the structure* `pls` so that its coordinates are now in the tangent plane. In order to view the poles as they appeared on the sky it will hereafter be necessary to type

```
show_poles, pls, view=view
```

otherwise, they will be shown as viewed from above the tangent plane. Mpole provides no reverse transformation since none of the MCT calculations should ever be done in plane-of-the-sky coordinates. All two-dimensional plotting routines will accept the `view` keyword.

When poles are saved for future use (see next section) it is important to save the viewing transformation if you ever want to plot things as they would appear in the sky. The sample file `ar930605.pns` contains the viewing transformation used when the magnetogram-derived poles were mapped onto the tangent plane. Reading the poles with the command

```
rd_pns, 'ar930605.pns', pls, view=view
```

will fill the structure `pls` with the 20 sources in the file, and will read the veiwing transformation into the variable `view`. Viewing the structure

```
** Structure <1aa628>, 8 tags, length=88, data length=84, refs=1:
   MAT             FLOAT     Array[3, 3]
   DISP            FLOAT     Array[3]
   LAT             FLOAT          -15.7515
   CMD             FLOAT           18.7504
   P               FLOAT           0.00000
   B               FLOAT        -0.0943472
   RAD             FLOAT           954.693
   DATE            STRING    '1993-06-05T13:59:46:000Z'
```

we see that the point of tangency is at $15^\circ\!.75S$ and $18^\circ\!.75W$. The string `view.date` contains the time of observation (in standardized `yyyy-mm-ddThh:mm:ss.fffZ` format). When such a string exists the view may be advanced in time by specifying the desired date and time, in the same format. For example the command

```
nv = solar_rotate_view( view, to='1993-06-07T14:00:00.0000Z' )
```

creates a new viewing transformation whose point of tangency is at $15^\circ\!.75S$ and $45^\circ\!.47W$.

1.8 Storage and Retrieval of Results

Poles, nulls, and separators may be saved to disk and retrieved later for further analysis.

```
write_pns, 'example2.pns', pls, nls, spr, view=view
```

writes all the information into an expanded `example.pns` file. This may be read back in by the command

```
rd_pns, 'example2.pns', pls, nls, spr, view=view
```

1.9 A Sample Session

There is nothing approaching a "typical" operating procedure for the Mpole programs since these versatile programs can be used in a limitless variety of different calculations. In the interest of demonstration, however, we present here a sequence of commands to read in a set of poles and analyze the connectivity of their field. This sequence combines, for the most part, commands which have been explained above.

The first steps read in the poles from a file, find the nulls, plot the footprint and try to add any missing nulls manually (although in this simple case none are missing)

```
rd_pns, 'example.pns', pls
nls = all_nulls( pls )
show_fp, pls, nls, xr=[-2,2], yr=[-2,2]
xadd_nulls, pls, nls
show_fp, pls, nls, xr=[-2,2], yr=[-2,2]
```

The next set of commands find the separators, plot these above the exisiting footprint, then display the whole thing three-dimensionally, including field lines from domains $P3$–$N4$ and $P2$–$N6$.

```
spr = all_seprs( pls, nls )
show_sepr, pls, nls, spr
show_3dpoles, pls, xr=[-2,2], yr=[-2,2]
show_3dnulls, nls
show_sepr, pls, nls, spr, /threed
show_domain, pls, 3, 2, /threed
show_domain, pls, 5, 1, /threed
```

The final sequence finds the connectivity of the field and displays it in three different ways

```
dm = domain_matrix( pls, nls )
dm_print, pls, dm
polka_map, pls, xr=[-2,2], yr=[-2,2], max=0.2
show_dm, pls, dm
window, 1
sum_graph, pls, nls, spr, dm
```

Chapter 2

Data Structures

Mpole routines operate on a variety of data structures. These structures have counterparts in magnetic charge topology, but most also include components for convenient numerical analysis. The following sections describe the major Mpole data structures.

2.1 Poles

A fundamental feature of Mpole models is some configuration of magnetic sources. The *poles structure* represents such configurations. Variables of this type contain a set of magnetic charges, their locations and strengths, and characteristics of the configuration as a whole. A poles variable `pls` has the following form:

pls.lab	An array of strings labeling each source in the configuration.
pls.x	An array of floats giving the x coordinate of each source.
pls.y	An array of floats giving the y coordinate of each source.
pls.z	An array of floats giving the z coordinate of each source.
pls.q	An array of floats giving the charge of each source.
pls.phi	An array of floats giving the flux of each source.
pls.bmax	The characterisitic maximum field strength.
pls.drmax	The characterisitic small distance.
pls.coc	An array of floats giving the coordinates of the center of charge.
pls.rmax	The characteristic maximum distance from `pls.coc`. Beyond this distance the field is approximated by a 2-term mutlipole expansion.
pls.mpole	The monopole term in the multipole expansion.
pls.dpole	The dipole term in the multipole expansion.
pls.alpha	A place holder for α in linear-force-free fields. (These are not yet implimented).

2.2 Nulls

In contrast with the poles structure, which contains information about an entire set of charges, the *null structure* describes only a single null.

A null variable `nl` has the following form:

nl.x	float array(3): the x,y,z coordinares of the null
nl.b	float `B(x)` (should be small, since this is meant to be a null).
nl.lam	float array(3) the eigenvalues (sorted) of the matrix $M_{ij} = \partial B_i/\partial x_j$.
nl.e0	float array(3) the unit vector of the spine
nl.e1	float array(3) one of the fan directions
nl.e2	float array(3) other fan direction
nl.type	character, either 'A' or 'B'
nl.ends	int array(2) the indices of charges (referring to the arrays in `pls`) at the ends of the spines. -1 for infinity. sorted in ascending order
nl.label	string giving the name of the null
nl.theta0	float the angle begining the circle on the plane, after that circle increases by π.

Arrays of nulls are used to represent all the null points for a given charge configuration.

2.3 Separators

A separator is a magnetic field line found at the intersection of two separatrix surfaces. Rather than storing the entire line, Mpole finds the separator by following a field lines from initial angles θ_A and θ_B from the fans of the nulls at each of its ends. The separator is a strcuture containing this information.

spr.anull	`int`: The index of the A-type (negative) null (the index refers to the array of `null` structures).
spr.bnull	`int`: The index of the B-type (positive) null (the index refers to the array of `null` structures).
spr.atheta	`double`: θ_A, the angle within the fan of the A-null, defined relative to `null.theta0`
spr.btheta	`double`: θ_B, the angle within the fan of the B-null, defined relative to `null.theta0`
spr.poles	array of 4 `int`s giving the indices of the spine charges of the two nulls.
spr.label	`char` a label for the separator
spr.psi	the flux enclosed by the separator
spr.i0	the fiducial current defined by the potential field separator
spr.len	the length of the separator field line

The user must ensure that a given array of separator-structures does not become inconsistant with the null points that generated it, since the separators are defined in terms of a specific set of nulls.

2.4 Field Lines

Magnetic field lines are represented by Mpole as arrays of vertices. When connected piecewise, the resulting line segments approximate the shape of a continuous curve. The number of vertices, and thus the size of the array, varies depending on user preferences and the complexity of the line.

Complete field lines take the form of a $3 \times n$ array, where n is the number of segments defining the line. The location of each vertex is given in Cartesian coordinates. For example, the straight line connecting the points $[x, y, z] = [0, 0, 0]$ and $[x, y, z] = [1, 1, 1]$ would be expressed in Mpole as $\begin{bmatrix} 0 & 0 & 0 \\ 1 & 1 & 1 \end{bmatrix}$.

2.5 Viewing Transformation

The *viewing transformation* has the following form: The viewing trasnformation tanes a vector in tangent-plane coordinates $\mathbf{x}_{tp}$ to plane-of-sky coordinates $\mathbf{x}_{pos}$ according to

$$\mathbf{x}_{pos} \;=\; \texttt{mat} \cdot \mathbf{x}_{tp} \;+\; \texttt{disp}$$

or in IDL

```
xpos = view.mat # xtp + view.disp
```

The information for this transformation is stored int he structure `view`

view.mat	the 3×3 transformation matrix
view.disp	the displacement vector.
view.lat	solar latitude of the point of tangency (degrees)
view.cmd	central merdian distance of the point of tangency (degrees)
view.b	the solar b angle (degrees)
view.p	the p angle (degrees)
view.rad	apparent solar radius (arcseconds)
view.scale	the linear scale factor.
view.date	a string giving the date in strandard format: `yyyy-mm-ddThh:mm:ss:fffZ`

2.6 Domain and Connectivity Matrices

Information about the domains formed by a set of N poles are stored in $(N + 1) \times (N + 1)$ matrices. Row/column i stores infomation about the pole

indexed by $i-1$ in the poles structure. Row/column 0 stores information about connections to ∞. For example, the *domain matrix*, `dm[i,j]`, returned by `domain_matrix.pro`, is `1` if poles `i-1` and `j-1` are interconnected, and `0` if they are not. Likewise, `dmn[0,i]=dmn[i,0]` indicates whether pole `i-1` connects to ∞.

2.7 Data Storage

Mpole stores data to disk in simple text files that can be created and modified by hand. Each file has a number of tagged sections containing the neccessary fields to build its associated Mpole variable. Data files can contain poles, nulls, separators, and viewing transformations.

Each section in the data file begins with the text `BEGIN` followed by the data type in capital letters. Similarly, the end of a section is marked by the text `END`, again followed by the data type in capital letters. Between these two tages are rows of data, the format of which depends on the section type. The row format for each section is as follows:

Section	Column 1	Column 2	Column 3	Column 4	Column 5	Column 6
Poles	Label	x coordinate	y coordinate	z coordinate	magnetic flux	N/A
Nulls	Array index	x coordinate	y coordinate	z coordinate	Type	Label
Separators	Array index	Index of "A" null	Index of "B" null	"A" angle	"B" angle	Label

Mpole routines will ignore any line in a data file that begins with `%`, which is useful for inserting comments. Data files created with `write_pns` will contain comments at the beginning of each section identifying the contents of each data column.

Chapter 3

Routines by Function

3.1 Input and Output Routines

cm2csv

given a set of poles and an array containing the connectivity matrix, output in connectivity matrix as a .csv (comma separated values) file.

Calling Sequence:

cm2csv, pls, cm

Inputs:

pls the poles structure (pls.phi & pls.lab are used)

cm connectvity matrix – int matrix which is 0 for no connection and ¿0 for connection.

Keyword Parameters:

minflux=minflux the minimum flux to include (poles whose connections total less than this are not included)

file=file the name of an output file ['cm.csv']

cm_list

given a set of poles and an array containing the connection matrix (returned by connectivity) print out a summary of the connections

Calling Sequence:

cm_list, cm, [pls]

Keyword Parameters:

ntop=ntop print, the ntop largest connections. [25]

err=err if set, this is the error matrix from connectivity

format=format the format for printing the fluxes

dm2tex

given a set of poles and an array containing the domain matrix, produce a LaTeX table (on the screen) with the tabel.

Calling Sequence:
dm2tex, pls, dm

Inputs:
pls the poles structure (pls.phi & pls.lab are used)

dm domain matrix – int matrix which is 0 for no connection and ¿0 for connection. its value will be used to determined the pinted character.

Keyword Parameters:
vstring = vstring a string array w/ N elements where N = max(dm)+1. vstring(dm(i,j)) will be printed at each location. default = ['0', '1', '2', ...]

dm_print

given a set of poles and an array containing the domain matrix, print a table

Calling Sequence:
dm_print, pls, dm

Inputs:
pls the poles structure (pls.phi & pls.lab are used)

dm domain matrix – int matrix which is 0 for no connection and ¿0 for connection. its value will be used to determined the pinted character.

Keyword Parameters:
vstring = vstring a string array w/ N elements where N = max(dm)+1. vstring(dm(i,j)) will be printed at each location. default = ['0', '1', '2', ...]

col_space=cs spacing between column. default is 2 setting to 0 will make the heading labels run together

latex_poles

output a table in LaTeX format summarizing the poles

Calling Sequence:
latex_poles, pls

Inputs:
pls the structure of poles

Keyword Parameters:
view = view is used to translate poles into heliographic coords

tan_plane if set coordinates are recoreded in Mm

pwr = pwr write fluxes in units of 10^{pwr} Mx

tot if set place a total at the bottom of each side

rd_pns

read in a .pns file containg poles, nulls & separators. return all of these as the appropriate structures/arrays. a file may contain only poles, or poles & nulls. and the remaining structures will not be set upon return. old-style .mp files, containing only poles may be read in. a line @— will separate multiple entries in a single file. if keyword multi is set then read through multi-1 of these lines.

Calling Sequence:
rd_pns, file_name, pls [, nulls [, sepr]]

Keyword Parameters:
view = view will return the viewing transformation (if present)

dm = dm will return the domain matrix (if present)

multi is set read through multi-1 different entries if there are not this many entries return multi set to -1.

norefine if set the null positions will not be refined

info if set the separator information will be stored in the sturcture. this takes additional time.

norelabel if set the nulls will not be relabeled

noends if set the end-points of spines will not be found.

rmax = rmax (fix rmax)

write_pns

output a file containing poles, nulls & separators

Calling Sequence:
write_pls, file_name, pls [, nls [, sepr]]

Inputs:
file_name string the name of the file to be created. file_name = '-' means write to std out

pls the structure containing all poles.

nls array of structures containing the nulls

sepr float array (3, n) contaning the separators

Keyword Parameters:

view = view the viewing transformation.

append if set the information is added to the end of the file, and can be read using the multi option of rd_pns

dm=dm the domain matrix

3.2 Calculational Routines

all_nulls

try to find as many nulls as possible using a combination of reasonable initial guess locations. remove duplicate nulls. label the nulls Ai or Bj for negative/positive nulls (using relabel_nulls). return an array of null structures.

Calling Sequence:

nls = all_nulls(pls)

Keyword Parameters:

noasymp unless set a zero will be sought based on the asymptotic field

Outputs:

nls the array of null structures.

all_seprs

given poles & nulls find all of the separators and return them as an array of spr structures. fill in structure with sepr_info

Calling Sequence:

sprs = all_seprs(pls, nulls)

Keyword Parameters:

ref_max the number of refinements in the null-scan

no_info if set, exclude the calculation of the sepr_info

cribbon

Calculate the ribbon along the separator. This is a structure containing several (3,n) arrays. cr.x - is the separator itself a (3,n) array cr.w - an array of the nomralized widths s.t. Delta = w * sqrt(—I—/I_0) cr.pos - array of vectors chosen to represent the edge of a current sheet w/ current +I_* cr.neg - array of vectors chosen to represent the edge of a current sheet w/ current -I_* cr.i0 - the fiducial current I_* for the separator

Calling Sequence:
cr = cribbon(pls, fl) or cr = cribbon(pls, nls, spr)

Inputs:
pls structure of poles

nls an array of separator structures

spr a single separator structure (refering to pls & nls)

fl the field line array to use instead of the spr

connectivity

calculate domain fluxes for a set of sources. uses a Monter-Carlo method to determine which poles are connected to which other poles. returns an array cm(i,j) which the approximate flux connecting i to j i=0 refers to infinity. the array returned is symmetric: cm(i,j) = cm(j,i) uses a bayesian estimate of the flux connecting i to j when m_i of N_i and m_j of N_j line connect the connecting value with maximum liklihood is cm(i,j) = (m_i + m_j)/(N_i/Phi_i + N_j/Phi_j)

Calling Sequence:
cm = connectivity(pls)

Keyword Parameters:
nlines=nlines total number of lines to initiate from each source. [300]

phicut=phicut if set nlines is set to 1.5*phi/phicut so 95\be used

noneg if set field lines from negative poles will not be used

doonly = name_list initialize field lines from only these poles.

err = err returns the statistical error in each value

max_steps = max_steps the maximum number of steps to take [10000]

current_solve

solve the equation governing the separator currents: i * L alog(e i0 / —i—) + M.i + psi = psi0 where M is the mutual induction matrix, psi are the separator vacuum fluxes and psi0 is the value they are required to have. this cannot be solved if any element of the vector (psi0 - psi)/(i0*L) has magntiude greater than one. in this case psi0 is altered to make the equation solvable and psi0 WILL BE CHANGED UPON RETURN.

Calling Sequence:
i = current_solve(psi0, si [, m])

Inputs:
psi0 a vector of desired fluxes. if this is unobtainable it is changed to one which can be obtained.

si the structure of separator info. returned from sepr_info

Optional Inputs:

m the mutual induction matrix. returned by mut_induct_mat

Keyword Parameters:

sweep if set a solution is constructed by sweeping the matrix m to its final value from 0.0. this will provide a unique solution related to the one w/o induction matrix.

init=i an initial guess for the current. this will be disabled by the sweep keyword.

erg=erg will return the energy of the current system.

domain_matrix

list the flux domains for a set of sources. Uses the nulls to decide which separatrices exist. This is used to fill in a matrix dm(i,j) = dm(j,i) with 1 when i connects to j and 0 otherwise. row/column 0 stands for infinity. row/column i stands for pole i-1

Calling Sequence:

dm = domain_matrix(pls, nls, [spr])

Optional Inputs:

spr the separator structure. if present, use spr.anull and spr.bnull to deduce connections

Keyword Parameters:

planar if set use only the photospheric separatrices

leaf if this is set AND spr is present the leaf domains are added using one fan trace from each null *not* connected to a separator.

mut_induct_mat

return a an array containing the mutual induction matrix for the separators given.

Calling Sequence:

m = mut_induct_mat(pls, nls, sepr)

Inputs:

pls poles structure

nls array of nulls structures

spr array of separator structures

Keyword Parameters:

verbose if set will display a graph showing the progress

Outputs:
m the an NxN mutual inductance matrix

xadd_nulls

put up a window displaying poles and nulls (and possibly gamma lines), and add new potential nulls.

Calling Sequence:
xadd_nulls, pls, nls

Inputs:
pls the list of poles

nls the null array. will be changed on output.

Keyword Parameters:
z=z put the initial guess at his level [default: z=0]

view=view a viewing transformation

xselect_fl

on a window showing poles etc. use the mouse to select field lines at various heights.

Calling Sequence:
xselect_fl, pls

Inputs:
pls (optional) the list of poles

Keyword Parameters:
z0 = [zlist] the values of z to use for the possible field lines. default is [10,20,30,40]. one field line will be integrated for each height.

view=view nulls are displayed using viewing trasnform view.

quiet if set the lines are plotted w/o listing the initial points.

xselect_null

put up a window displaying poles and nulls (and possibly gamma lines), and select a subset of the nulls with the cursor. return only that subset.

Calling Sequence:
new_nulls = xselect_nulls(nls [, pls])

Inputs:
nls the complete list of nulls (from which to select

pls (optional) the list of poles

Keyword Parameters:

gamma draw the gamma lines (spines)

over nulls are already displayed

omit rather than selecting nulls to include, select nulls to omit

view=view nulls are displayed using viewing trasnform view.

3.3 Graphics Routines

dm_schematic

graph the domain matrix schematically. the graph will be two columns (positive left of negatives) with the lines connecting between them.

Calling Sequence:

dm_schematic, pls, dm

Keyword Parameters:

title=title the title to print at the top

nosum if set the summary legent is suppressed

linestyles=linetsyles an array of length max(dm) st. connection i,j is plotted with line style linestyle(dm(i,jI)-1) [default is all 0s]

min_deg=min_deg plot only poles with degree above min_deg min_deg=1 omits unconnected poles

nophi if set the total fluxes will be suppressed

format=format the format for printing the fluxes

fill_dmn_fp

overplot a specific domain footprint with a filled region. the footprint must be a simple 4-sided region with opposite poles on diagonal vertices and prone nulls (of any type) at the other vertices. the boundaries of the region consist of spines and separatrix traces. *** if the region does not fit these criteria nothing will be plotted ***

Calling Sequence:

fill_dmn_fp, pls, i1, i2, null1, null2

Inputs:

pls the complete poles structure

i1, i2 indices from pls which form vertices of the region. OR then names of the poles given as strings. these must match an entry in pls.lab exactly.

null1, null2 the poles (complete structures) which form the other vertices

Keyword Parameters:

shrink=fctr shrink the region by a factor fctr (0¡fctr¡1) for aesthetic purposes.

shade=shade the color with which to shade the region.

view=view a viewing transformation to apply.

Example:

IDL¿ show_fp, pls, nls IDL¿ fill_dmn_fp, pls, 'P03', 'N06', nls(2), nls(7)

pls_phi_hist

plot a histogram summarizing the fluxes in all poles a particular structure.

Calling Sequence:

pls_phi_plot, pls

Inputs:

pls the poles stucture to be summarized. pls.phi is used (and possibly scaled) and pls.lab to label the bins

Keyword Parameters:

scl=scl multiply fluxes by scl. [1.0]

mx if set then scl=1.0e16 meaning the fluxes are assumed be in G Mmˆ2 and the plot will be in Mx.

phimax=phimax the range of the plot.

nlab=nlab label the largest nlab poles along the bottom. [5]

nmax=nmax include only nmax largest positive and nmax larges negative sources. default is to plot all of them.

title=title title for the top of the plot.

polka_map

plot locations of each pole in a poles structure as disks whose area is proportional to their flux.

Calling Sequence:

polka_map, pls

Keyword Parameters:

over if not set a bounding box will be constructed. and a greyscale will be rendered

nolab omit labels

view = view will transform according to transformation in structure view. title=title

maxrad = maxrad if set this will define the maximum radius in data units [default = 0.1*xrange]

flux_scl=flux_scl the value of the flux to assign the maximum radius [default is maximum flux]

axis=ax if set to 1 no axis will be draw if set to 2 a box will be drawn

show_3dnulls

graph all the nulls in an array of null-structures

Calling Sequence:
show_3dnulls, nulls

Keyword Parameters:
label if set, display the labels of each null

show_3dpoles

plot locations of each pole in a poles structure — in 3d

Calling Sequence:
show_3dpoles, pls

Keyword Parameters:
xrange = [x0, x1] force the xrange of plot window

yrange = [y0, y1] force the yrange of plot window

zrange = [z0, z1] force the zrange of plot window

over if not set a bounding box will be constructed.

nolab omit labels

zaxis = za the zaxis will go from 0 to za default is the full zrange

ax = ax the x-rotation for the 3d projection

az = az the x-rotation for the 3d projection

show_connectivity

graph the connectivity on an exisiting graph of poles

Calling Sequence:
show_connectivity, pls, c_mat

Keyword Parameters:
min_flux = min_flux the minimum flux to indicate [0.5]

count = count return the number of connection

inf if set, connections to infinity are depicted as arrows linestyle=linestyle

show_cr_skel

draw the skeleton of a current ribbon over an exisiting plot

Calling Sequence:
show_cr_skel, cr

Inputs:
cr structure containing the current ribbon as returned by cribbon

Keyword Parameters:
i = i the current (unormalized)

norm_i = ni the normalized current i/i_*

view = view will transform according to transformation in structure view.

threed show field line in 3d

ribs=n if set the "ribs" of the skeleton are shown. show every n ribs

show_dm

graph the domain matrix on an exisiting graph of poles

Calling Sequence:
show_dm, pls, dm

Keyword Parameters:
linestyle = linestyle linestyle used to l]plot lines

mindegree = md will only show edges connecting to vertices of degree md or greater

inf if set arrows are used to show linkages to infinity

show_domain

show field lines from the domain ia/ib. nlines field lines are started at ia & traced until they hit another pole. only those which terminate at ib are shown.

Calling Sequence:
show_domain, pls, ia, ib

Keyword Parameters:
ntries = ntries the number of lines in the monte carlo integral [100]

nlines = nlines the number of lines to draw. if unspecified then every one fromthe MC integral is drawn

threed same as graph, but for 3d rendering

rad = rad radius to begin field lines [2*pls.drmax]

view = view project with view transfrom

max_segs = max_segs set to the maximum number of segments in a line [1000]

show_flux_tube

graph a set of field lines which start in a ring about the point x0. the ring has radius r and is oriented normal to the magnetic field at point x0. there are nlines field lines in the ring

Calling Sequence:
show_flux_tube, pls, x0

Inputs:
pls the poles structure

x0 3-element array (float) of the initial point.

Keyword Parameters:
nlines=nlines number of lines to use [30]

radius=radius radius of circle [10*pls.drmax]

threed if set, render in 3d

view=view use view structure

circle if set draw the circle around x0

clip if set clip the field lines

dir if set to +1 ot -1 only that direction will be traced

show_fp

graph the locations of sources, nulls and footprints of the field. footprints are shown as dashed and solid lines. solid lines denot photospheric spines, dashed lines are the footprints of separatrices. dotted lines are the spines of coronal null points.

Calling Sequence:
show_fp, pls, nls

Keyword Parameters:
view = view will transform according to transformation in structure view.

nosepx do not show the separatrix field lines

nogamma do not show the gamma-lines (spines)

view = view perform the viewing transformation

threed show the field lines in 3d

clip if set, each of the lines in clipped

max_segs set to the maximum number of segments in a line [1000]

label label the nulls

axis=ax if set to 1 no axis will be draw if set to 2 a box will be drawn

over plot is made over the exisiting plot

color if set the pos/neg poles are shown in white/black this is useful when plotting over a grey-scale magnetogram

mg setting this is equivalent to /over, /color which is useful when overplotting on a magnetogram

show_loop_skeleton

given a field line and a set of radii, draw a sekeloon consisting of rings arounf each point in the field line

Calling Sequence:
show_loop_skeleton, fl, rad

Keyword Parameters:
view = view will transform according to transformation in structure view.

threed show field line in 3d

skip=skip if set then only every skip ribs are rendered. [1]

show_nulls

graph all the nulls in an array of null-structures

Calling Sequence:
show_nulls, nulls

Keyword Parameters:
label if set, display the labels of each null

new draw new set of axes.

view = view will transform according to transformation in structure view.

show_poles

plot locations of each pole in a poles structure

Calling Sequence:
show_poles, pls

Keyword Parameters:
xrange = [x0, x1] force the xrange of plot window

yrange = [y0, y1] force the yrange of plot window

over if not set a bounding box will be constructed.

color if set the pos/neg poles are shown in white/black this is useful when plotting over a grey-scale magnetogram

nolab omit labels on the poles

axis=ax if set to 1 no axis will be draw if set to 2 a box will be drawn

view = view - will transform according to transformation in structure view.

mg setting this is equivalent to /over, /color which is useful when overplotting on a magnetogram

show_random_3dlines

graph nlines different field lines

Calling Sequence:
show_random_3dlines, pls

Keyword Parameters:
nlines=nlines the number of lines to draw. [30]

show_sepr

draw all the separators in array sepr

Calling Sequence:
show_sepr, pls, nls, spr

Keyword Parameters:
view = view will transform according to transformation in structure view.

threed show field line in 3d

shadow if set (& in 3d mode) a projection onto z=0 is also shown

clip clip the field line

show_sepx

graph field lines from the separatrix of null.

Calling Sequence:
show_sepx, pls, null

Inputs:
pls the poles structure

null a single null structure

Keyword Parameters:

view = view set to the viewing transformation

thrnge = [th0, th1] a 2-element array defining the range of theta to be used. default is [null.theta0, null.theta0 + !pi]

thvals = thv an array of theta values

nlines = n the number of lines to draw. default is 30

threed if set the field lines are rendered in 3d.

clip if set this will clip the field lines

sum_graph

draw the domain graph and null graph for a set of poles, nulls and separators

Calling Sequence:

sum_graph, pls, nls, spr, [dm]

Inputs:

psl the poles structure

nls an array of null structures

spr an array of separators. spr.anull and spr.bnull are unsed to index the nls connected by each separator. this is used to generate the domain matrix if it is not passed in separately

Optional Inputs:

dm the domain matrix

Keyword Parameters:

title=title a title for the top of the plot

noorphans if set the null graph will omit nulls w/o separators.

leaf if keyword is set, and spr is present, use the slower domain_matrix algorithm which finds leaf domains. from nulls w/ unbroken fans.

voronoi_pls

a graph of the voronoi tesselation formed by a set of poles

Calling Sequence:

voronoi_pls, pls

Keyword Parameters:

fill if set the positive regions will be filled using the default line-fill pattern.

3.4 Analysis Routines

dm_schematic

graph the domain matrix schematically. the graph will be two columns (positive left of negatives) with the lines connecting between them.

Calling Sequence:
dm_schematic, pls, dm

Keyword Parameters:
title=title the title to print at the top

nosum if set the summary legent is suppressed

linestyles=linetsyles an array of length max(dm) st. connection i,j is plotted with line style linestyle(dm(i,jI)-1) [default is all 0s]

min_deg=min_deg plot only poles with degree above min_deg min_deg=1 omits unconnected poles

nophi if set the total fluxes will be suppressed

format=format the format for printing the fluxes

domain_char

calculate flux and volume of the flux domain connecting poles ia to ib. monte carlo integration is used to perform the integrals. field lines are initialized at random on pole ia and traced to their terminus.

Calling Sequence:
res = domain_char(pls, ia, ib, add=add)

Keyword Parameters:
n = n the number of lines in the monte carlo integral [100]

graph if set each of the lines connecting ia-¿ib are rendered

threed same as graph, but for 3d rendering

rad radius to begin field lines [2*pls.drmax]

index_check

check poles and nulls against two Poincare indices involving the numbers of various sources & null points: P2d = prone - upright - sources + 2 P3d = B-type - A-type - positive + negative if all nulls are present both indices will be zero. if P2d is positive there are at least P2d prone nulls missing. if it is negative, there are at least -P2d upright nulls missing. etc. [see Longcope & Klapper, ApJ v.579, p.468 (2002)] the program prints out the numbers of each type & reports on the conclusions derived from the indices. if the separator is passed in it also reports the number of domains topologically required. domains = separators + sources - coronal-nulls - 1 [Beveridge & Longocpe Sol Phys v.227, p.193 (2005)]

Calling Sequence:
index_check, pls, nls, [spr]

Inputs:
pls the poles structure

nls the array of nulls structures

Optional Inputs:
spr the array of separator structures

line_info

return a 3 X N array (float) where each point contains [len, B, bˆ . zˆ]

Calling Sequence:
info = line_info(pls, fl)

null_graph

use the null & separator structures to plot a schematic graph with pos. nulls in left column, negative nulls in right column connected by a line for each separator

Calling Sequence:
null_graph, nls, spr

Inputs:
nls an array of null structures

spr an array of separators. spr.anull and spr.bnull are unsed to index the nls connected by each separator

Keyword Parameters:
title=title the title to print at the top

nosum if set the summary legent is suppressed

noorphans if set the graph will omit nulls w/o separators.

pls_phi_hist

plot a histogram summarizing the fluxes in all poles a particular structure.

Calling Sequence:
pls_phi_plot, pls

Inputs:
pls the poles stucture to be summarized. pls.phi is used (and possibly scaled) and pls.lab to label the bins

Keyword Parameters:
scl=scl multiply fluxes by scl. [1.0]

mx if set then scl=1.0e16 meaning the fluxes are assumed be in G Mm^2 and the plot will be in Mx.

phimax=phimax the range of the plot.

nlab=nlab label the largest nlab poles along the bottom. [5]

nmax=nmax include only nmax largest positive and nmax larges negative sources. default is to plot all of them.

title=title title for the top of the plot.

sepr_info

return an array (or structure) containing magnetic information on each of the magnetic separators in the array sepr. the information: [length, flux, I_0, s, hel] I_0: I_*/c in units of flux/length For Mx & cm I_0 has units of Mx/cm = 10 Amps (i.e. 10*I_0 is the current in Amps) For 1.0e16 Mx & Mm I_0 has units of 1.0e9 Amps hel: relative helicty per current. The relative helicity of the entire field is the sum over all separators as H_R = \sum_i hel(i) cur(i)

Calling Sequence:

info = sepr_info(pls, nulls, sepr)

Keyword Parameters:

structure if set the information is handed back as a structure, rather than an array.

lineclose if set the flux is calculated using a return path making a straight line between the null points

sum_graph

draw the domain graph and null graph for a set of poles, nulls and separators

Calling Sequence:

sum_graph, pls, nls, spr, [dm]

Inputs:

psl the poles structure

nls an array of null structures

spr an array of separators. spr.anull and spr.bnull are unsed to index the nls connected by each separator. this is used to generate the domain matrix if it is not passed in separately

Optional Inputs:

dm the domain matrix

Keyword Parameters:

title=title a title for the top of the plot

noorphans if set the null graph will omit nulls w/o separators.

leaf if keyword is set, and spr is present, use the slower domain_matrix algorithm which finds leaf domains. from nulls w/ unbroken fans.

3.5 Viewing Transforms

disk2tan_plane

translate a set of poles from disk to tangent plane coordinates. poles pls have x, y coordinates in terms of location on the disk (radius rad) & a z coordinate giving distance above surface (same units as x & y). translate this to coordinates (in Mm) on a tangent plane centered at heliographic coordinates (lat, cmd) and disk location (x0, y0) return the inverse transformation as a structure view containing the matrix, the displacement & the heliographic coodinates (in degrees) of the transformation the disk location xd is related to tangent_plane location xtp by xd = view.mat # xtp + view.disp xd now contains a z coordinates indicating vertical location

Calling Sequence:

view = disk2tan_plane(pls)

Inputs:

pls - the poles. will be changed upon return

Keyword Parameters:

rad = rad radius of disk [default = 925 arcsec]

b = b the b-angle (deg) for heliographic coods. [default = 0]

p = p position angle (deg) for heliographic coods. [default = 0]

center_hel = (lat, cmd) force center to be at there heliographic coordinates

center_disk = (x0, y0) force center to be at this disk location (default = center of charge)

date if set this string contains the date of the transformation

vert if set the fluxes in disk corrdinates are assumed to be vertical field Bz integrated over pixels rather than line-of-sight field.

mg = mg a data structure which will be used to replace keywords rad, b, p and date in one call. the structure

must contain fields mg.b0, mg.p, mg.rad, mg.date

new_tan_plane

translate a set of poles from one tangent plane to a new tangent plane. change pole the pole structure and the viewing structure. the transformation is specified by its cantact-point

Calling Sequence:

new_tan_plane, pls, view, new_center [, options] new_tan_plane, pls, view, match=view2

Inputs:

pls the poles. will be changed upon return.

view the viewing transformation - changed upon return.

new_center the new center float(2) specified in heliographic or disk coordinates

Keyword Parameters:

match = view2 advance tangent plane of this view to the date/time of view, and use this center to do the tangent plane. if this option is set the new_center argument is not used.

hel if set interpret center as (lat, cmd) in heliographic coordinates [defult]

disk if set interpret center as (x0, y0) in arcseconds from center of disk

tp if set interpret center = (x0, y0) in the tangent plane

solar_rotate_view

given a structure for viewing at a tangent plane. advance the view by a fixed time interval (in seconds) according to the solar rotation. the center of the of the tangent plane is (lat, cmd), b angle and p angle will be given in degrees. the view structure contains the following elements mat: the 3 x 3 transformation matrix disp: the displacement vector. transformation from tan-plane coordinates xtp to disk coordinates xd: xd = view.mat # xtp + view.disp b: the b angle (degrees) p: the p angle (degrees) rad: apparent solar radius (arcseconds)

Calling Sequence:

new_view = solar_rotate_view(view, time)

Inputs:

view = view the old view

time time (in seconds)

Keyword Parameters:

to = date if set advance the view to a date specified by the string date from view.date

rad=rad this new radius will be used rather thant the one in view. this permits a change in perspective as between SOHO and an Earth-orbiting satellite

b0=b0 this new b0 will be used rather thant the one in view. this permits a change in perspective as between SOHO and an Earth-orbiting satellite

shift=[dx,dy] shift the view by a [dx,dy] arcseconds. this permits co-alignment

view2time

from a single viewing trasnformation or an array return a float (or array of floats) denoting the time from some reference point. unless otherwise specified the time will be in hours and the reference time will be the begining of the day of the first element.

Calling Sequence:
time = view2time(view)

Inputs:
view a view-xform structure, or array of structures.

Keyword Parameters:
days if set the time is converted to decimal days

from=datestr a string giving the reference tim

Outputs:
time a float or float array

Chapter 4

Alphabetical List of Routines

4.1 Top level Routines

all_nulls

try to find as many nulls as possible using a combination of reasonable initial guess locations. remove duplicate nulls. label the nulls Ai or Bj for negative/positive nulls (using relabel_nulls). return an array of null structures.

Calling Sequence:

nls = all_nulls(pls)

Keyword Parameters:

noasymp unless set a zero will be sought based on the asymptotic field

Outputs:

nls the array of null structures.

all_seprs

given poles & nulls find all of the separators and return them as an array of spr structures. fill in structure with sepr_info

Calling Sequence:

sprs = all_seprs(pls, nulls)

Keyword Parameters:

ref_max the number of refinements in the null-scan

no_info if set, exclude the calculation of the sepr_info

cm2csv

given a set of poles and an array containing the connectivity matrix, output in connectivity matrix as a .csv (comma separated values) file.

Calling Sequence:
cm2csv, pls, cm

Inputs:
pls the poles structure (pls.phi & pls.lab are used)

cm connectvity matrix – int matrix which is 0 for no connection and ¿0 for connection.

Keyword Parameters:
minflux=minflux the minimum flux to include (poles whose connections total less than this are not included)

file=file the name of an output file ['cm.csv']

cm_list

given a set of poles and an array containing the connection matrix (returned by connectivity) print out a summary of the connections

Calling Sequence:
cm_list, cm, [pls]

Keyword Parameters:
ntop=ntop print, the ntop largest connections. [25]

err=err if set, this is the error matrix from connectivity

format=format the format for printing the fluxes

connectivity

calculate domain fluxes for a set of sources. uses a Monter-Carlo method to determine which poles are connected to which other poles. returns an array cm(i,j) which the approximate flux connecting i to j i=0 refers to infinity. the array returned is symmetric: cm(i,j) = cm(j,i) uses a bayesian estimate of the flux connecting i to j when m_i of N_i and m_j of N_j line connect the connecting value with maximum liklihood is cm(i,j) = (m_i + m_j)/(N_i/Phi_i + N_j/Phi_j)

Calling Sequence:
cm = connectivity(pls)

Keyword Parameters:
nlines=nlines total number of lines to initiate from each source. [300]

phicut=phicut if set nlines is set to 1.5*phi/phicut so 95\be used

noneg if set field lines from negative poles will not be used

doonly = name_list initialize field lines from only these poles.

err = err returns the statistical error in each value

max_steps = max_steps the maximum number of steps to take [10000]

cribbon

Calculate the ribbon along the separator. This is a structure containing several (3,n) arrays. cr.x - is the separator itself a (3,n) array cr.w - an array of the nomralized widths s.t. Delta = w * sqrt(—I—/I_0) cr.pos - array of vectors chosen to represent the edge of a current sheet w/ current +I_* cr.neg - array of vectors chosen to represent the edge of a current sheet w/ current -I_* cr.i0 - the fiducial current I_* for the separator

Calling Sequence:

cr = cribbon(pls, fl) or cr = cribbon(pls, nls, spr)

Inputs:

pls structure of poles

nls an array of separator structures

spr a single separator structure (refering to pls & nls)

fl the field line array to use instead of the spr

current_solve

solve the equation governing the separator currents: i * L alog(e i0 / —i—) + M.i + psi = psi0 where M is the mutual induction matrix, psi are the separator vacuum fluxes and psi0 is the value they are required to have. this cannot be solved if any element of the vector (psi0 - psi)/(i0*L) has magntiude greater than one. in this case psi0 is altered to make the equation solvable and psi0 WILL BE CHANGED UPON RETURN.

Calling Sequence:

i = current_solve(psi0, si [, m])

Inputs:

psi0 a vector of desired fluxes. if this is unobtainable it is changed to one which can be obtained.

si the structure of separator info. returned from sepr_info

Optional Inputs:

m the mutual induction matrix. returned by mut_induct_mat

Keyword Parameters:

sweep if set a solution is constructed by sweeping the matrix m to its final value from 0.0. this will provide a unique solution related to the one w/o induction matrix.

init=i an initial guess for the current. this will be disabled by the sweep keyword.

erg=erg will return the energy of the current system.

disk2tan_plane

translate a set of poles from disk to tangent plane coordinates. poles pls have x, y coordinates in terms of location on the disk (radius rad) & a z coordinate giving distance above surface (same units as x & y). translate this to coordinates (in Mm) on a tangent plane centered at heliographic coordinates (lat, cmd) and disk location (x0, y0) return the inverse transformation as a structure view containing the matrix, the displacement & the heliographic coodinates (in degrees) of the transformation the disk location xd is related to tangent_plane location xtp by xd = view.mat # xtp + view.disp xd now contains a z coordinates indicating vertical location

Calling Sequence:
view = disk2tan_plane(pls)

Inputs:
pls - the poles. will be changed upon return

Keyword Parameters:
rad = rad radius of disk [default = 925 arcsec]

b = b the b-angle (deg) for heliographic coods. [default = 0]

p = p position angle (deg) for heliographic coods. [default = 0]

center_hel = (lat, cmd) force center to be at there heliographic coordinates

center_disk = (x0, y0) force center to be at this disk location (default = center of charge)

date if set this string contains the date of the transformation

vert if set the fluxes in disk corrdinates are assumed to be vertical field Bz integrated over pixels rather than line-of-sight field.

mg = mg a data structure which will be used to replace keywords rad, b, p and date in one call. the structure

must contain fields mg.b0, mg.p, mg.rad, mg.date

dm2tex

given a set of poles and an array containing the domain matrix, produce a LaTeX table (on the screen) with the tabel.

Calling Sequence:
dm2tex, pls, dm

Inputs:
pls the poles structure (pls.phi & pls.lab are used)

dm domain matrix – int matrix which is 0 for no connection and ¿0 for connection. its value will be used to determined the pinted character.

Keyword Parameters:

vstring = vstring a string array w/ N elements where N = max(dm)+1. vstring(dm(i,j)) will be printed at each location. default = ['0', '1', '2', ...]

dm_print

given a set of poles and an array containing the domain matrix, print a table

Calling Sequence:

dm_print, pls, dm

Inputs:

pls the poles structure (pls.phi & pls.lab are used)

dm domain matrix – int matrix which is 0 for no connection and ¿0 for connection. its value will be used to determined the pinted character.

Keyword Parameters:

vstring = vstring a string array w/ N elements where N = max(dm)+1. vstring(dm(i,j)) will be printed at each location. default = ['0', '1', '2', ...]

col_space=cs spacing between column. default is 2 setting to 0 will make the heading labels run together

dm_schematic

graph the domain matrix schematically. the graph will be two columns (positive left of negatives) with the lines connecting between them.

Calling Sequence:

dm_schematic, pls, dm

Keyword Parameters:

title=title the title to print at the top

nosum if set the summary legent is suppressed

linestyles=linetsyles an array of length max(dm) st. connection i,j is plotted with line style linestyle(dm(i,jI)-1) [default is all 0s]

min_deg=min_deg plot only poles with degree above min_deg min_deg=1 omits unconnected poles

nophi if set the total fluxes will be suppressed

format=format the format for printing the fluxes

domain_char

calculate flux and volume of the flux domain connecting poles ia to ib. monte carlo integration is used to perform the integrals. field lines are initialized at random on pole ia and traced to their terminus.

Calling Sequence:
res = domain_char(pls, ia, ib, add=add)

Keyword Parameters:
n = n the number of lines in the monte carlo integral [100]

graph if set each of the lines connecting ia-¿ib are rendered

threed same as graph, but for 3d rendering

rad radius to begin field lines [2*pls.drmax]

domain_matrix

list the flux domains for a set of sources. Uses the nulls to decide which separatrices exist. This is used to fill in a matrix dm(i,j) = dm(j,i) with 1 when i connects to j and 0 otherwise. row/column 0 stands for infinity. row/column i stands for pole i-1

Calling Sequence:
dm = domain_matrix(pls, nls, [spr])

Optional Inputs:
spr the separator structure. if present, use spr.anull and spr.bnull to deduce connections

Keyword Parameters:
planar if set use only the photospheric separatrices

leaf if this is set AND spr is present the leaf domains are added using one fan trace from each null *not* connected to a separator.

fill_dmn_fp

overplot a specific domain footprint with a filled region. the footprint must be a simple 4-sided region with opposite poles on diagonal vertices and prone nulls (of any type) at the other vertices. the boundaries of the region consist of spines and separatrix traces. *** if the region does not fit these criteria nothing will be plotted ***

Calling Sequence:
fill_dmn_fp, pls, i1, i2, null1, null2

Inputs:
pls the complete poles structure

i1, i2 indices from pls which form vertices of the region. OR then names of the poles given as strings. these must match an entry in pls.lab exactly.

null1, null2 the poles (complete structures) which form the other vertices

Keyword Parameters:

shrink=fctr shrink the region by a factor fctr (0¡fctr¡1) for aesthetic purposes.

shade=shade the color with which to shade the region.

view=view a viewing transformation to apply.

Example:

IDL¿ show_fp, pls, nls IDL¿ fill_dmn_fp, pls, 'P03', 'N06', nls(2), nls(7)

index_check

check poles and nulls against two Poincare indices involving the numbers of various sources & null points: P2d = prone - upright - sources + 2 P3d = B-type - A-type - positive + negative if all nulls are present both indices will be zero. if P2d is positive there are at least P2d prone nulls missing. if it is negative, there are at least -P2d upright nulls missing. etc. [see Longcope & Klapper, ApJ v.579, p.468 (2002)] the program prints out the numbers of each type & reports on the conclusions derived from the indices. if the separator is passed in it also reports the number of domains topologically required. domains = separators + sources - coronal-nulls - 1 [Beveridge & Longocpe Sol Phys v.227, p.193 (2005)]

Calling Sequence:

index_check, pls, nls, [spr]

Inputs:

pls the poles structure

nls the array of nulls structures

Optional Inputs:

spr the array of separator structures

inv_lam_func

return the inverse of function \Lamda(u) = u ln(e / —u—) = 12.745 f(u/12.745)

Calling Sequence:

u = inv_lam_func(lambda)

latex_poles

output a table in LaTeX format summarizing the poles

Calling Sequence:
latex_poles, pls

Inputs:
pls the structure of poles

Keyword Parameters:
view = view is used to translate poles into heliographic coords
tan_plane if set coordinates are recoreded in Mm
pwr = pwr write fluxes in units of 10^{pwr} Mx
tot if set place a total at the bottom of each side

line_info

return a 3 X N array (float) where each point contains [len, B, b^ . z^]

Calling Sequence:
info = line_info(pls, fl)

mut_induct_mat

return a an array containing the mutual induction matrix for the separators given.

Calling Sequence:
m = mut_induct_mat(pls, nls, sepr)

Inputs:
pls poles structure
nls array of nulls structures
spr array of separator structures

Keyword Parameters:
verbose if set will display a graph showing the progress

Outputs:
m the an NxN mutual inductance matrix

new_tan_plane

translate a set of poles from one tangent plane to a new tangent plane. change pole the pole structure and the viewing structure. the transformation is specified by its cantact-point

Calling Sequence:
new_tan_plane, pls, view, new_center [, options] new_tan_plane, pls, view, match=view2

Inputs:

pls the poles. will be changed upon return.

view the viewing transformation - changed upon return.

new_center the new center float(2) specified in heliographic or disk coordinates

Keyword Parameters:

match = view2 advance tangent plane of this view to the date/time of view, and use this center to do the tangent plane. if this option is set the new_center argument is not used.

hel if set interpret center as (lat, cmd) in heliographic coordinates [defult]

disk if set interpret center as (x0, y0) in arcseconds from center of disk

tp if set interpret center = (x0, y0) in the tangent plane

null_graph

use the null & separator structures to plot a schematic graph with pos. nulls in left column, negative nulls in right column connected by a line for each separator

Calling Sequence:

null_graph, nls, spr

Inputs:

nls an array of null structures

spr an array of separators. spr.anull and spr.bnull are unsed to index the nls connected by each separator

Keyword Parameters:

title=title the title to print at the top

nosum if set the summary legent is suppressed

noorphans if set the graph will omit nulls w/o separators.

pixel2tan_plane

translate a set of poles from to tangent plane coordinates from coordinates given in terms of pixels relative to some arbitrary point. the location on the disk will be inferred by specifying (lat,cmd) for some reference pixel, ref_pixel, in the image (assumed to be center-of-charge) and the pixel scale (assumed to be 1 arcsecond). poles pls have x, y coordinates in terms of location on the disk (radius rad) & once again the z coordinate gives distance above surface (same units as x & y). translate this to coordinates (in Mm) on a tangent plane centered at heliographic coordinates (lat,cmd)

and pixle-location ref_pixel. return the inverse transformation as a structure view containing the matrix, the displacement & the heliographic coodinates (in degrees) of the transformation the pixel-location xp is related to tangent_plane location xtp by xp = view.mat # xtp + view.disp xp now contains a z coordinates indicating vertical location

Calling Sequence:
view = pixel2tan_plane(pls, lat, cmd)

Inputs:
pls the poles. will be changed upon return

lat heliographic latitude of ref_pixel

cmd heliographic longitude of ref_pixel

Keyword Parameters:
rad = rad radius of disk [default = 925 arcsec]

b = b the b-angle (deg) for heliographic coods. [default = 0]

p = p position angle (deg) for heliographic coods. [default = 0]

ref_pixel 2 element array containin the pixel to which lat and cmd refer

pix_scale size of pixels in arcseconds

pls_phi_hist

plot a histogram summarizing the fluxes in all poles a particular structure.

Calling Sequence:
pls_phi_plot, pls

Inputs:
pls the poles stucture to be summarized. pls.phi is used (and possibly scaled) and pls.lab to label the bins

Keyword Parameters:
scl=scl multiply fluxes by scl. [1.0]

mx if set then scl=1.0e16 meaning the fluxes are assumed be in G Mm^2 and the plot will be in Mx.

phimax=phimax the range of the plot.

nlab=nlab label the largest nlab poles along the bottom. [5]

nmax=nmax include only nmax largest positive and nmax larges negative sources. default is to plot all of them.

title=title title for the top of the plot.

polka_map

plot locations of each pole in a poles structure as disks whose area is proportional to their flux.

Calling Sequence:
polka_map, pls

Keyword Parameters:
over if not set a bounding box will be constructed. and a grey-scale will be rendered

nolab omit labels

view = view will transform according to transformation in structure view. title=title

maxrad = maxrad if set this will define the maximum radius in data units [default = 0.1*xrange]

flux_scl=flux_scl the value of the flux to assign the maximum radius [default is maximum flux]

axis=ax if set to 1 no axis will be draw if set to 2 a box will be drawn

rd_pns

read in a .pns file containg poles, nulls & separators. return all of these as the appropriate structures/arrays. a file may contain only poles, or poles & nulls. and the remaining structures will not be set upon return. old-style .mp files, containing only poles may be read in. a line @— will separate multiple entries in a single file. if keyword multi is set then read through multi-1 of these lines.

Calling Sequence:
rd_pns, file_name, pls [, nulls [, sepr]]

Keyword Parameters:
view = view will return the viewing transformation (if present)

dm = dm will return the domain matrix (if present)

multi is set read through multi-1 different entries if there are not this many entries return multi set to -1.

norefine if set the null positions will not be refined

info if set the separator information will be stored in the sturcture. this takes additional time.

norelabel if set the nulls will not be relabeled

noends if set the end-points of spines will not be found.

rmax = rmax (fix rmax)

sepr_info

return an array (or structure) containing magnetic information on each of the magnetic separators in the array sepr. the information: [length, flux, I_0, s, hel] I_0: I_*/c in units of flux/length For Mx & cm I_0 has units

of Mx/cm = 10 Amps (i.e. 10*I_0 is the current in Amps) For 1.0e16 Mx & Mm I_0 has units of 1.0e9 Amps hel: relative helicty per current. The relative helicity of the entire field is the sum over all separators as H_R = \sum_i hel(i) cur(i)

Calling Sequence:
info = sepr_info(pls, nulls, sepr)

Keyword Parameters:
structure if set the information is handed back as a structure, rather than an array.

lineclose if set the flux is calculated using a return path making a straight line between the null points

show_3dnulls

graph all the nulls in an array of null-structures

Calling Sequence:
show_3dnulls, nulls

Keyword Parameters:
label if set, display the labels of each null

show_3dpoles

plot locations of each pole in a poles structure — in 3d

Calling Sequence:
show_3dpoles, pls

Keyword Parameters:
xrange = [x0, x1] force the xrange of plot window

yrange = [y0, y1] force the yrange of plot window

zrange = [z0, z1] force the zrange of plot window

over if not set a bounding box will be constructed.

nolab omit labels

zaxis = za the zaxis will go from 0 to za default is the full zrange

ax = ax the x-rotation for the 3d projection

az = az the x-rotation for the 3d projection

show_connectivity

graph the connectivity on an exisiting graph of poles

Calling Sequence:
show_connectivity, pls, c_mat

Keyword Parameters:

min_flux = min_flux the minimum flux to indicate [0.5]

count = count return the number of connection

inf if set, connections to infinity are depicted as arrows linestyle=linestyle

show_cr_skel

draw the skeleton of a current ribbon over an exisiting plot

Calling Sequence:

show_cr_skel, cr

Inputs:

cr structure containing the current ribbon as returned by cribbon

Keyword Parameters:

i = i the current (unormalized)

norm_i = ni the normalized current i/i_*

view = view will transform according to transformation in structure view.

threed show field line in 3d

ribs=n if set the "ribs" of the skeleton are shown. show every n ribs

show_dm

graph the domain matrix on an exisiting graph of poles

Calling Sequence:

show_dm, pls, dm

Keyword Parameters:

linestyle = linestyle linestyle used to l]plot lines

mindegree = md will only show edges connecting to vertices of degree md or greater

inf if set arrows are used to show linkages to infinity

show_domain

show field lines from the domain ia/ib. nlines field lines are started at ia & traced until they hit another pole. only those which terminate at ib are shown.

Calling Sequence:

show_domain, pls, ia, ib

Keyword Parameters:

ntries = ntries the number of lines in the monte carlo integral [100]

nlines = nlines the number of lines to draw. if unspecified then every one fromthe MC integral is drawn

threed same as graph, but for 3d rendering

rad = rad radius to begin field lines [2*pls.drmax]

view = view project with view transfrom

max_segs = max_segs set to the maximum number of segments in a line [1000]

show_flux_tube

graph a set of field lines which start in a ring about the point x0. the ring has radius r and is oriented normal to the magnetic field at point x0. there are nlines field lines in the ring

Calling Sequence:

show_flux_tube, pls, x0

Inputs:

pls the poles structure

x0 3-element array (float) of the initial point.

Keyword Parameters:

nlines=nlines number of lines to use [30]

radius=radius radius of circle [10*pls.drmax]

threed if set, render in 3d

view=view use view structure

circle if set draw the circle around x0

clip if set clip the field lines

dir if set to +1 ot -1 only that direction will be traced

show_fp

graph the locations of sources, nulls and footprints of the field. footprints are shown as dashed and solid lines. solid lines denot photospheric spines, dashed lines are the footprints of separatrices. dotted lines are the spines of coronal null points.

Calling Sequence:

show_fp, pls, nls

Keyword Parameters:

view = view will transform according to transformation in structure view.

nosepx do not show the separatrix field lines

nogamma do not show the gamma-lines (spines)

view = view perform the viewing transformation

threed show the field lines in 3d

clip if set, each of the lines in clipped

max_segs set to the maximum number of segments in a line [1000]

label label the nulls

axis=ax if set to 1 no axis will be draw if set to 2 a box will be drawn

over plot is made over the exisiting plot

color if set the pos/neg poles are shown in white/black this is useful when plotting over a grey-scale magnetogram

mg setting this is equivalent to /over, /color which is useful when overplotting on a magnetogram

show_gamma_lines

graph all the gamma lines for a set of nulls

Calling Sequence:

show_gamma_lines, pls, nulls

Keyword Parameters:

view = view - will transform according to transformation in structure view. nosepx - do not show the separatrix field lines nogamma - do not show the gamma field lines view - perform the viewing transformation threed - show the field lines in 3d clip - if set, each of the lines in clipped max_segs - set to the maximum number of segments in a line [1000]

show_loop_skeleton

given a field line and a set of radii, draw a sekeloon consisting of rings arounf each point in the field line

Calling Sequence:

show_loop_skeleton, fl, rad

Keyword Parameters:

view = view will transform according to transformation in structure view.

threed show field line in 3d

skip=skip if set then only every skip ribs are rendered. [1]

show_nulls

graph all the nulls in an array of null-structures

Calling Sequence:
show_nulls, nulls

Keyword Parameters:
label if set, display the labels of each null

new draw new set of axes.

view = view will transform according to transformation in structure view.

show_poles

plot locations of each pole in a poles structure

Calling Sequence:
show_poles, pls

Keyword Parameters:
xrange = [x0, x1] force the xrange of plot window

yrange = [y0, y1] force the yrange of plot window

over if not set a bounding box will be constructed.

color if set the pos/neg poles are shown in white/black this is useful when plotting over a grey-scale magnetogram

nolab omit labels on the poles

axis=ax if set to 1 no axis will be draw if set to 2 a box will be drawn

view = view - will transform according to transformation in structure view.

mg setting this is equivalent to /over, /color which is useful when overplotting on a magnetogram

show_random_3dlines

graph nlines different field lines

Calling Sequence:
show_random_3dlines, pls

Keyword Parameters:
nlines=nlines the number of lines to draw. [30]

show_sepr

draw all the separators in array sepr

Calling Sequence:
show_sepr, pls, nls, spr

Keyword Parameters:
view = view will transform according to transformation in structure view.

threed show field line in 3d

shadow if set (& in 3d mode) a projection onto z=0 is also shown

clip clip the field line

show_sepx

graph field lines from the separatrix of null.

Calling Sequence:
show_sepx, pls, null

Inputs:
pls the poles structure

null a single null structure

Keyword Parameters:
view = view set to the viewing transformation

thrnge = [th0, th1] a 2-element array defining the range of theta to be used. default is [null.theta0, null.theta0 + !pi]

thvals = thv an array of theta values

nlines = n the number of lines to draw. default is 30

threed if set the field lines are rendered in 3d.

clip if set this will clip the field lines

solar_rotate_view

given a structure for viewing at a tangent plane. advance the view by a fixed time interval (in seconds) according to the solar rotation. the center of the of the tangent plane is (lat, cmd), b angle and p angle will be given in degrees. the view structure contains the following elements mat: the 3 x 3 transformation matrix disp: the displacement vector. transformation from tan-plane coordinates xtp to disk coordinates xd: xd = view.mat # xtp + view.disp b: the b angle (degrees) p: the p angle (degrees) rad: apparent solar radius (arcseconds)

Calling Sequence:
new_view = solar_rotate_view(view, time)

Inputs:
view = view the old view

time time (in seconds)

Keyword Parameters:

to = date if set advance the view to a date specified by the string date from view.date

rad=rad this new radius will be used rather thant the one in view. this permits a change in perspective as between SOHO and an Earth-orbiting satellite

b0=b0 this new b0 will be used rather thant the one in view. this permits a change in perspective as between SOHO and an Earth-orbiting satellite

shift=[dx,dy] shift the view by a [dx,dy] arcseconds. this permits co-alignment

sum_graph

draw the domain graph and null graph for a set of poles, nulls and separators

Calling Sequence:

sum_graph, pls, nls, spr, [dm]

Inputs:

psl the poles structure

nls an array of null structures

spr an array of separators. spr.anull and spr.bnull are unsed to index the nls connected by each separator. this is used to generate the domain matrix if it is not passed in separately

Optional Inputs:

dm the domain matrix

Keyword Parameters:

title=title a title for the top of the plot

noorphans if set the null graph will omit nulls w/o separators.

leaf if keyword is set, and spr is present, use the slower domain_matrix algorithm which finds leaf domains. from nulls w/ unbroken fans.

tan_plane2disk

translate a set of poles from the tangent plane to disk coordinates (arcseconds from disk center). the transoforamtion is specified in the structure view. returns a new viewing transformation which does the composite transformation.

Calling Sequence:

new_view = tan_plane2disk(pls, view)

Inputs:

pls the poles. will be changed upon return

view the structure specifying the transformation

view_xform

returns a structure for viewing at a tangent plane. the origin of the tangent plane is (lat, cmd), b angle and p angle will be given in degrees. the view structure contains the following elements mat: the 3 x 3 transformation matrix disp: the displacement vector. transformation from tan-plane coordinates xtp to disk coordinates xd: xd = view.mat # xtp + view.disp b: the b angle (degrees) p: the p angle (degrees) rad: apparent solar radius (arcseconds) scale: the linear scale factor. date: a string containing the date & time of the observation

Calling Sequence:

view = view_xform(lat, cmd)

Inputs:

lat - the latitude cmd - the longitude

Keyword Parameters:

rad = rad radius of disk [default = 925 arcsec]

b = b the b-angle (deg) for heliographic coods. [default = 0]

p = p position angle (deg) for heliographic coods. [default=0] degrees - if set lat & cmd are taken to be in degrees date = date_str - the string containg date & time

voronoi_pls

a graph of the voronoi tesselation formed by a set of poles

Calling Sequence:

voronoi_pls, pls

Keyword Parameters:

fill if set the positive regions will be filled using the default line-fill pattern.

write_pns

output a file containing poles, nulls & separators

Calling Sequence:

write_pls, file_name, pls [, nls [, sepr]]

Inputs:

file_name string the name of the file to be created. file_name = '-' means write to std out

pls the structure containing all poles.

nls array of structures containing the nulls

sepr float array (3, n) contaning the separators

Keyword Parameters:

view = view the viewing transformation.

append if set the information is added to the end of the file, and can be read using the multi option of rd_pns

dm=dm the domain matrix

xadd_nulls

put up a window displaying poles and nulls (and possibly gamma lines), and add new potential nulls.

Calling Sequence:

xadd_nulls, pls, nls

Inputs:

pls the list of poles

nls the null array. will be changed on output.

Keyword Parameters:

z=z put the initial guess at his level [default: z=0]

view=view a viewing transformation

xselect_fl

on a window showing poles etc. use the mouse to select field lines at various heights.

Calling Sequence:

xselect_fl, pls

Inputs:

pls (optional) the list of poles

Keyword Parameters:

z0 = [zlist] the values of z to use for the possible field lines. default is [10,20,30,40]. one field line will be integrated for each height.

view=view nulls are displayed using viewing trasnform view.

quiet if set the lines are plotted w/o listing the initial points.

xselect_null

put up a window displaying poles and nulls (and possibly gamma lines), and select a subset of the nulls with the cursor. return only that subset.

Calling Sequence:
new_nulls = xselect_nulls(nls [, pls])

Inputs:
nls the complete list of nulls (from which to select

pls (optional) the list of poles

Keyword Parameters:
gamma draw the gamma lines (spines)

over nulls are already displayed

omit rather than selecting nulls to include, select nulls to omit

view=view nulls are displayed using viewing trasnform view.

4.2 Lower Level Routines

advance_dif_rot

advance cmd according to differential rotation law \[\Omega = a + b \sin^2(lat) \] a = 14.552, b = -2.84 degrees per day (sidereal)

Calling Sequence:
advance_diff_rot, lc, time

Inputs:
lc [lat, cmd] in radians unless degree flag is set

time time (in seconds)

KEYWORD PARAMETRS

degree if set the lat and cmd are interpretted in degrees (default is radians)

asym_fl_map

given a point at or beyond the asymptotic field limit —r1— ¿ pls.rmax, return the point at the same radius which the asymptotic field maps to. use the multi-pole expansion in pls. if the field line does not return then closes = 0 upon return

Calling Sequence:
r2 = asym_fl_map(pls, r1, closes=closes)

Keyword Parameters:
closes - set to 0 if the field does not close

avg_bp

return ¡B'¿

Calling Sequence:
ab = avg_bp(pls, fl)

calc_sepr_info

calculate the properties of each separator and add it to the spr structure. if there are multiple separators do all of them.

Calling Sequence:
calc_sepr_info, pls, nulls, sepr

clip_fl

clip a field line to fit in 3d window

Calling Sequence:
fl = clip_fl(fl)

closest_pole

return the index of the pole closest to the point x

Calling Sequence:
i = closest_pole(pls, x)

create_null

locate a null-point, given an initial guess. return a null structure null: null.x - float array(3) the x,y,z coordinares of the null null.b - float —B(x)— null.lam - float array(3) the eigenvalues of bprime (sorted) null.e0 - float array(3) the spine direction null.e1 - float array(3) one of the span direcitons null.e2 - float array(3) other spine direction null.type - character, either 'A' or 'B' null.char - character, 'P' (prone), 'U' (upright), 'C' (coronal) of 'M' (mirror) null.ends - int array(2) the indices of charges at ends of the spines. -1 for infinity. sorted in ascending order null.label - string giving the name of the null null.theta0 - float the angle begining the circle on the plane. after that circle increases by !pi

Calling Sequence:
null = create_null(pls, x) KEYWORD: noend - if set the spines will not be traced

create_pls

create a blank poles structure, with np poles. pls.x - float array (np) of x-locations pls.y - float array (np) of y-locations pls.z - float array (np) of z-locations pls.q - float array (np) of charges of the poles pls.phi - float array (np) of fluxes = 2*!pi*pls.q pls.lab - string array (np) of labels pls.rmax - float — characterisitic maximum distance pls.bmax - float — characterisitic maximum field strength pls.drmax - float — characterisitic small distance pls.coc - float array (3) is the center-of-(unsigned)charge pls.mpole - float array with the monopole moment of set of poles pls.dpole - float array (3) with the dipole moment of set of poles relative to the COC

Calling Sequence:
pls = create_pls(np)

create_spr

create a blank separator structure spr structure: spr.anull - (int) index of the A-type null spr.bnull - (int) index of the B-type null spr.atheta - (float) angle from the A-null spr.btheta - (float) angle fromt he B-null spr.poles - (int array 4) list of poles ++– spr.label - (string) the label for the separator fields filled in by spr_info spr.psi - the flux enclosed between separator and z=0 spr.len - the length of the separator spr.zmax - maximum height of separator spr.i0 - the characteristic current

Calling Sequence:
spr = create_spr([array])

Inputs:
array (optional) the 4-element float array used in v1.0. this will be used to fill in anull, bnull, atheta and btheta fields of structure

eval_a

at a point, x(0:2) evaluate the vector potential due to the set of poles given by pls. for a point charge at the origin the vector potential is 1 - cos(th) (1 - z/r) A = ———- phi^ = ——— z^ X r r sin(th) x^2 + y^2

Calling Sequence:
a = eval_a(pls, x)

eval_b

at a point, x(0:2) evaluate the magnetic field due to the set of poles given by pls

Calling Sequence:
b = eval_b(pls, x)

eval_b2p

at a point, x(0:2) evaluate the 2nd derivative tensor of the magnetic field due to the set of poles given by pls d^2 B_i d^3 d(i,j,k) = ——— = - ————— chi(x) dx_j dx_k dx_i dx_j dx_k

Calling Sequence:
b = eval_b2p(pls, x)

eval_bprime

at a point, x(0:2) evaluate the derivative matrix of the magnetic field due to the set of poles given by pls

Calling Sequence:
b = eval_b(pls, x)

eval_z

at a point, x(0:2) evaluate the Auxilliary field Z(x) s.t. curl(Z) = A the vector potential. for a point charge at the origin the vector potential is 1 - cos(th) (1 - z/r) A = ———- phi^ = ——— z^ X r r sin(th) x^2 + y^2 1 - cos(th) r z - r^2 z^ Z = ———- theta^ = ——— sin(th) —r—(—r— + z)

Calling Sequence:
z = eval_z(pls, x)

find_null

Newton-Raphson to find a null, given a guess. Return the converged null in place of the guess

Calling Sequence:
find_null, pls, x

Keyword Parameters:
twod - restricts search to x,y plane stat - return the status of the search. stat=0 if failed

fl_from_point

return a single field line, traced from from point x0. default is to plot in both directions.

Calling Sequence:
fl = fl_from_point(pls, x0)

Keyword Parameters:
max_segs set to the maximum number of segments
seg_size number of integration steps per segment [10]
dir if set to +1 ot -1 only that direction will be traced

fl_intgrt

integrate the field line from initial point x0 until termination by one of several criteria

Calling Sequence:
done = fl_intgrt(pls, x0)

Inputs:
pls - the structure of poles x0 - the initial point for the field line will be change on return

done has the following value 0 = failure 1 = len_max is exceeded 2 = cang_min exceeded 3 = bmax exceeded 4 = plane crossed 5 = dist_max exeeded 6 = rad_max exeeded

Keyword Parameters:

RESULT dir - trace in direction dir (default is +1.0) max_step - the maximum integration step to take len_max - maximum integrated distance of line dist_max - maximum distance from x0 step_max - maximum size of a step (default = pls.drmax) max_steps - maximum number of steps (default = 1000) cang_min - the minimum cosine of the normal angle cross = cpv = float array(3,2) = [[xctr], [dir]] if set end the tracing when the field line crosses the plane defined by point xctr and direction vector dir

fl_view_xform

given a field line perfom the specified viewing transformation. the field line is returned transformed.

Calling Sequence:

fl_view_xform, fl, view

gamma_line

return one of the gamma lines from null, including its two ends

Calling Sequence:

fl = gamma_line(pls, null)

init_sepx_line

begin a field line at position theta along the given null. locally the field has the structure B = lam1 (r.e1) e1 + lam2 (r.e2) e2 where —lam1— ¡ —lam2— are the eigenvalues of bprime the field lines can then be defined in terms of a = lam1/lam2, so that 0 ¡= a ¡= 1 r = (cos(th))ˆa rhoˆa e1 + sin(th) rho e2 where rho increases outward from a field line & is the largest physical distance at an d r / d rho = rhoˆ{-1} [a (r.e1) e1 + (r.e1) e2] = (rho lam2)ˆ{-1} B

Calling Sequence:

x = init_spex_line, pls, null, theta

is_pole

return the index of the pole identified by label lab if no pole matches return -1. print, pls.lab(is_pole(pls, 'N01')) N01

Calling Sequence:

i = is_pole(pls, lab)

line_piece

return a section of a field line, with a fixed number of points, uniformly distributed. input and output are float arrays fl(3,*)

Calling Sequence:
fl = line_piece(ofl, zmin=zmin, n=n) ARGUUMENTS: ofl - the old field line

Keyword Parameters:
zmin - the lowest altitude for the field line n - the number of points in the field line ref = ref. fi non-zero the points will be exponentially concentrated at the ends with maximum ratio ref (¿ 1.0)

null_scan

given a single null compile a list of all charge-pairs (by index) contacted by the null. if the keyword th is set, it will be returned with a list of all angles theta bracketing the separator.;+

Calling Sequence:
inds = null_scan(pls, null)

Inputs:
pls - the structure of poles null - a single null structure

Keyword Parameters:
twopi - scan over all 2pi radians, rather than simply pi nth - the number of segments to check ref_max - the number of times to refine a boundary [5]

Outputs:
inds - intarr(2,n) lists the pairs of poles bracketed by separators.

pole_of_theta

return the pole index found from integrating away from a given null at angle theta. If the trace extends past the afl radius continue using asym_fl_map.

Calling Sequence:
i = pole_of_theta(pls, null, th_val)

Inputs:
pls the poles structure

null a single null structure

th_val the angle from which to sent a fan field line from the null point.

random_field_line

return a single field line, selected at random. probability is based on flux.

Calling Sequence:
fl = random_field_line(pls)

relabel_nulls

define the null labels according to a given scheme. sequential: nls(5).lab = 'A06' or 'B06' polar: nls(5).lab = 'P03-P15' etc.

Calling Sequence:
relabel_nulls, nls, [pls], [keywords]

Keyword Parameters:
sequential use sequenctial labeling [default]

polar use polar labeling

sort put the nulls into an order based on spine sources

relabel_poles

assign pole labels in sequential order pls.lab(5) = 'P06' or 'N06'

Calling Sequence:
relabel_poles, pls,

Keyword Parameters:
sort if set the poles are sorted in decreasing order of magnetitude order = order - if initially set to an array the array will be returned with values in the new order.

sepr_line

return a separator field line from nullA - nullB. defined by the angle thA and thB

Calling Sequence:
fl = sepr_line(pls, nulls, sepr)

sepr_plane

return a point, and a triad of vectors defining a plane normal to the separator. this is returned as a 3 X 4 float array. result(*,0) is the origin of the new coordinate system. result(*,1) and result(*,2) are vectors in the normal plane and result(*,2) is the normal vector (parallel to B).

Calling Sequence:
result = sepr_plane(pls, nulls, sepr)

sepx_line

return a separatix field line from null along direction theta

Calling Sequence:
fl = sepx_line(pls, null, theta)

Keyword Parameters:

max_segs=max_segs - the maximum number of segments to use

tan_plane_matrix

a 3 x 3 matrix translating neighbors of the point (lat, cmd) on the disk, to a tangent plane.

Calling Sequence:

tpmat = tan_plane_matrix(lat, cmd). matrix takes a tangent vector in disk coordinates [vx, vy, vz] = tmat # [vw, vn, vr]

Inputs:

lat - latitude of point cmd - the longitude

Keyword Parameters:

b = b the b-angle (deg) for heliographic coods. [default=0]

p = p position angle (deg) for heliographic coods. [default=0] degrees; if set the lat & cmd are in degrees (default is radians)

update_nulls

given a set of poles and a set of nulls from an related set of poles update the nulls. use their location as initial guesses.

Calling Sequence:

new_nulls = update_nulls(pls, nls)

view2time

from a single viewing trasnformation or an array return a float (or array of floats) denoting the time from some reference point. unless otherwise specified the time will be in hours and the reference time will be the begining of the day of the first element.

Calling Sequence:

time = view2time(view)

Inputs:

view a view-xform structure, or array of structures.

Keyword Parameters:

days if set the time is converted to decimal days

from=datestr a string giving the reference tim

Outputs:

time a float or float array

Bibliography

[1] C. Beveridge and D. W. Longcope. On three-dimensional magnetic skeleton elements due to discrete flux sources. *Solar Phys.*, 227:193–206, 2005.

[2] D. W. Longcope. Topology and current ribbons: A model for current, reconnection and flaring in a complex, evolving corona. *Solar Phys.*, 169:91–121, 1996.

[3] D. W. Longcope. Separator current sheets: Generic features in minimum-energy magnetic fields subject to flux constraints. *Phys. Plasmas*, 8:5277–5290, 2001.

[4] D. W. Longcope and I. Klapper. A general theory of connectivity and current sheets in coronal magnetic fields. *ApJ*, 579:468–481, 2002.

[5] D. W. Longcope and T. Magara. A comparison of the minimum current corona to a magnetohydrodynamic simulation of quasi-static coronal evolution. *ApJ*, 608:1106–1123, 2004.