Rendering and line-tracing codes for the spherical coordinate system

The purpose of the rendering code is to compute the intersections between the voxels (specified by their spherical coordinates) and given straight line(s) of sight. Namely, the voxels are specified by the coordinates of their boundaries with respect to the longitude φ, latitude θ and radial distance r, i.e.

$$\varphi_i = \left\{\varphi_0, \varphi_1, \ldots, \varphi_{N_\varphi - 1}\right\},$$

$$\theta_j = \left\{\theta_0, \theta_1, \ldots, \theta_{N_\theta - 1}\right\},$$

$$r_k = \left\{r_0, r_1, \ldots, r_{N_r - 1}\right\}.$$

All coordinate arrays are assumed to be strictly ascending; the nodes do not need to be evenly spaced, but the φ, θ and r grids are assumed to be independent on each other. The coordinate system used is shown in Figure 1, with $0 \le \varphi < 360^\circ$ (this implies $\varphi_{N_\varphi - 1} - \varphi_0 < 360^\circ$), $-90^\circ < \theta < 90^\circ$ (the latitude is relative to the equator, so that -90° and $+90^\circ$ correspond to the southern and northern poles, respectively; the values of $\pm 90^\circ$ should not be used for the grid nodes because the coordinate system becomes degenerate at the poles), and $r > 0$.

The purpose of the line-tracing code is to compute the magnetic field line(s) passing through given point(s) for a given vector magnetic field. The code uses the above-mentioned spherical coordinate system and grid, and the magnetic field $\mathbf{B} = (B_\varphi, B_\theta, B_r)$ is specified by the values of its components at the grid nodes, i.e. $(B_r)_{ijk} = B_r(\varphi_i, \theta_j, r_k)$, etc. Instead of (or in addition to) returning the array(s) of points specifying the field line(s), the code can compute the line(s) characteristics such as the length, maximum and average magnetic field along the line(s), and the magnetic field strength at the line(s) endpoints.

The codes are implemented as IDL functions which then call the respective Windows/Linux executable library.

1 Rendering code

The code is implemented as two IDL functions:

- `RenderSpherical` — single-thread computation (for one line of sight);
- `RenderSphericalMulti` — utilizes the multi-thread capabilities (can be applied to a number of lines of sight simultaneously).

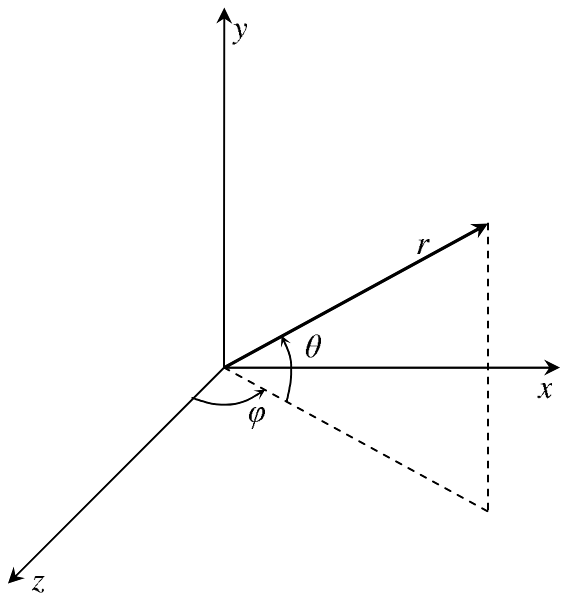

Figure 1: Spherical coordinate system.

1.1 RenderSpherical

This function has the following syntax:

```
Nvox = RenderSpherical(lon_arr, lat_arr, r_arr, p1, p2, rmin, $
                       index, dl, lon_ind, lat_ind, r_ind, $
                       [, /closed])
```

1.1.1 Input parameters

- `lon_arr` is the array of longitude grid points $\{\varphi_i\}$ [degrees], double-precision floating-point.
- `lat_arr` is the array of latitude grid points $\{\theta_j\}$ [degrees], double-precision floating-point.
- `r_arr` is the array of radial distance grid points $\{r_k\}$ [in the units of solar radius], double-precision floating-point.
- `p1`$= \{r, \varphi, \theta\}_{\text{start}}$ is a 3-element double-precision floating-point array specifying the start point of the line of sight (in spherical coordinates). The radial distance is in units of solar radius, and longitude and latitude are in degrees.
- `p2`$= \{r, \varphi, \theta\}_{\text{end}}$ is a 3-element double-precision floating-point array specifying the end point of the line of sight.
- `rmin` is the effective photosphere radius $r_{\min}$ [in the units of solar radius], double-precision floating-point scalar. If $r_{\min} > 0$, the lines of sight intersecting the sphere $r = r_{\min}$ are truncated so that only their parts that

can contribute to the observable emission (if any) are considered.
Note: $r_{\min}$ should not exceed the lower boundary of the data cube; if $r_{\min} > r_0$, r_0 will be used instead as the effective photosphere radius.

1.1.2 Return value

$N_{\rm vox}$ — the number of voxels intersected by the line of sight (can be zero if the line passes beyond the data cube or if all emission is absorbed in the layers below $r_{\min}$), long integer.

1.1.3 Output parameters

The output parameters are defined if $N_{\rm vox} > 0$. All data arrays are sorted accordingly to the emission propagation direction (i.e., from `p1` to `p2`).

- `index` is the $N_{\rm vox}$-element long-integer array of one-dimensional voxel identifiers (for the voxels intersected by the line of sight). For a voxel with the boundaries of $(\varphi_i, \varphi_{i+1})$, (θ_j, θ_{j+1}), (r_k, r_{k+1}), the corresponding one-dimensional identifier is computed as $\mathrm{id} = i + jN_\varphi + kN_\varphi N_\theta$.

- `dl` is the $N_{\rm vox}$-element double-precision floating-point array of the projected depths of the intersected voxels [in the units of solar radius], i.e., the lengths of the fragments of the line of sight falling within the corresponding voxels.

- `lon_ind` is the $N_{\rm vox}$-element double-precision floating-point array of the fractional indices of the line of sight midpoints within each of the intersected voxels, with respect to the longitude grid, i.e.:
  $\mathrm{ind}_\varphi = i + (\varphi_c - \varphi_i)/(\varphi_{i+1} - \varphi_i)$ for $\varphi_i \le \varphi_c < \varphi_{i+1}$,
  where $\varphi_{\rm c}$ is the midpoint coordinate.

- `lat_ind` is the $N_{\rm vox}$-element double-precision floating-point array of the fractional indices of the line of sight midpoints within each of the intersected voxels, with respect to the latitude grid, i.e.:
  $\mathrm{ind}_\theta = j + (\theta_c - \theta_j)/(\theta_{j+1} - \theta_j)$ for $\theta_j \le \theta_c < \theta_{j+1}$,
  where $\theta_{\rm c}$ is the midpoint coordinate.

- `r_ind` is the $N_{\rm vox}$-element double-precision floating-point array of the fractional indices of the line of sight midpoints within each of the intersected voxels, with respect to the radial distance grid, i.e.:
  $\mathrm{ind}_r = k + (r_c - r_k)/(r_{k+1} - r_k)$ for $r_k \le r_c < r_{k+1}$,
  where $r_{\rm c}$ is the midpoint coordinate.

The fractional indices are intended to be used for computing the values of φ, θ, r, magnetic field, etc. at the midpoints by interpolation — e.g., using the IDL routine `interpolate`.

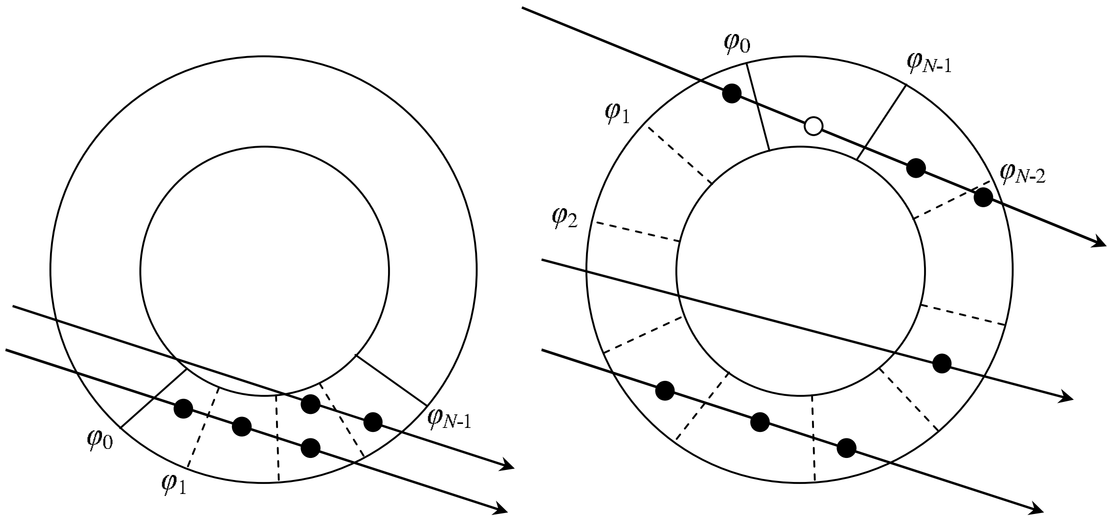

Figure 2: Voxels in the "open" and "closed" models.

1.1.4 Keywords

If the keyword `/closed` is off, the data cube is assumed to cover a limited range of longitudes, so that $\varphi_0 \le \varphi < \varphi_{N_\varphi-1}$; this is a typical situation for an isolated active region. In Figure 2, this corresponds to considering only the voxels marked by filled dots. The resulting values of the fractional index ind_φ always fall into the range $0 \le \mathrm{ind}_\varphi < N_\varphi - 1$.

If the keyword `/closed` is on, the data cube is assumed to cover the entire possible range of longitudes $0 \le \varphi < 360°$; this is a typical situation for the PFSS extrapolation. In this case, the region with $\varphi < \varphi_0$ or $\varphi \ge \varphi_{N_\varphi-1}$ is considered as additional voxels (marked by empty dot in Figure 2). If these voxels are intersected by the line of sight, they have the following characteristics:

- The one-dimensional voxel identifier (`index`) is computed using the same formula as above with $i = N_\varphi - 1$.
- The fractional index ind_φ is computed as

$$\mathrm{ind}_\varphi = N_\varphi - 1 + \frac{(\varphi_c + 360° - \varphi_{N_\varphi-1}) \bmod 360°}{\varphi_0 + 360° - \varphi_{N_\varphi-1}},$$

  so that $0 \le \mathrm{ind}_\varphi < N_\varphi$.

Other dimensions (θ and r) are unaffected by this keyword.

1.2 `RenderSphericalMulti`

This function has the following syntax:

```
Nvox_m = RenderSphericalMulti(NLOS, lon_arr, lat_arr, r_arr, $
                              p1_m, p2_m, rmin, $
```

```
                                    index_m, dl_m, $
                                    lon_ind_m, lat_ind_m, r_ind_m, $
                                    [, /closed])
```

1.2.1 Input parameters

- `NLOS` is the number of lines of sight, long integer.
- `lon_arr`, `lat_arr`, `r_arr`, `rmin` are the same as in the previous function `RenderSpherical`.
- `p1_m` is a 3×`NLOS`-element double-precision floating-point array, each jth column of which `p1_m[*, j]` specifies the start point of the corresponding jth line of sight, like in the function `RenderSpherical`.
- `p2_m` — same as `p1_m`, for the end points of the lines of sight.

1.2.2 Return value

The function returns the `NLOS`-element long-integer array `Nvox_m`$=\{N_{\rm vox}\}$ specifying the numbers of intersected voxels for each line of sight (some elements can be zero, if no intersections have been found).

1.2.3 Output parameters

All output parameters (`index_m`, `dl_m`, `lon_ind_m`, `lat_ind_m`, `r_ind_m`) are two-dimensional $N_{\rm max}$×`NLOS`-element arrays (long-integer for `index_m` and double-precision floating-point for others), where $N_{\rm max}$ is a number sufficient to accommodate all results, i.e. $N_{\rm max} \geq \max(N_{\rm vox})$. Each jth column of each of these arrays is equivalent to the corresponding parameter of the function `RenderSpherical`: e.g., the value `index_m[i, j]` represents the one-dimensional index of the ith intersected voxel along the jth line of sight, etc.; these values are defined only for $i <$`Nvox_m[j]`. Each j column of the output arrays is sorted according to the emission propagation direction.

1.2.4 Keywords

Same as in the function `RenderSpherical`.

1.3 Notes

Both the start and end points of the line(s) of sight should be located beyond the data cube boundaries and beyond the sphere with $r = r_{\rm min}$; otherwise, the results returned can be incorrect.

2 Line-tracing code

The code is implemented as four IDL functions:

- `LineTrace` — single-thread computation (for one field line), returns the field line itself and its characteristics;
- `LineTraceMulti` — utilizes the multi-thread capabilities (can be applied to a number of field lines simultaneously), returns both the field lines themselves and their characteristics;
- `LineTraceShort` — single-thread computation (for one field line), returns the field line characteristics only;
- `LineTraceShortMulti` — utilizes the multi-thread capabilities (can be applied to a number of field lines simultaneously), returns the field lines characteristics only.

The `Short` versions of the functions work faster and consume much less memory.

2.1 `LineTrace`

This function has the following syntax:

```
N = LineTrace(lon_arr, lat_arr, r_arr, Bph, Bth, Br, $
              lon0, lat0, r0, ds, $
              l_lon, l_lat, l_r, $
              lineclosed, L, Bmax, Bavg, B1, B2)
```

2.1.1 Input parameters

- `lon_arr` is the array of longitude grid points $\{\varphi_i\}$ [degrees], double-precision floating-point.
- `lat_arr` is the array of latitude grid points $\{\theta_j\}$ [degrees], double-precision floating-point.
- `r_arr` is the array of radial distance grid points $\{r_k\}$ [in the units of solar radius], double-precision floating-point.
- `Bph` is the three-dimensional array specifying the longitudinal component of the magnetic field B_φ at the grid nodes, double-precision floating-point.
- `Bth` is the three-dimensional array specifying the latitudinal component of the magnetic field B_θ at the grid nodes, double-precision floating-point.

- `Br` is the three-dimensional array specifying the longitudinal component of the magnetic field B_r at the grid nodes, double-precision floating-point.
- `lon0`, `lat0` and `r0` specify the coordinates of the initial point of the field line φ_* [degrees], θ_* [degrees] and r_* [in the units of solar radius], respectively; double-precision floating-point.
- `ds` is the integration step along the field line (i.e., the distance between the adjacent points along the line) [in the units of solar radius], double-precision floating-point scalar.

2.1.2 Return value

N — the number of found points along the field line (or zero if the initial point is located outside the boundaries of the data cube), long integer.

2.1.3 Output parameters

The output parameters are defined if $N > 0$.

- `l_lon` is the N-element double-precision floating-point array specifying the longitudes φ_l [degrees] of the found points along the field line.
- `l_lat` is the N-element double-precision floating-point array specifying the latitudes θ_l [degrees] of the found points along the field line.
- `l_r` is the N-element double-precision floating-point array specifying the radial coordinates r_l [in the units of solar radius] of the found points along the field line.
- `lineclosed` — 1 if the field line is closed (i.e., both its endpoints are located at the lower boundary of the data cube $r = r_0$), 0 otherwise; long integer.
- `L` is the length of the field line L [in the units of solar radius], double-precision floating-point.
- `Bmax` is the maximum magnetic field strength $B_{\max}$ along the field line, double-precision floating-point.
- `Bavg` is the average magnetic field strength $\langle B\rangle = \int B(s)\,\mathrm{d}s/L$ along the field line, double-precision floating-point.
- `B1` and `B2` are the magnetic field strengths at the field line endpoints, double-precision floating-point.

2.2 LineTraceMulti

This function has the following syntax:

```
N_m = LineTraceMulti(Nlines, lon_arr, lat_arr, r_arr, $
                     Bph, Bth, Br, $
                     lon0_m, lat0_m, r0_m, ds, $
                     l_lon_m, l_lat_m, l_r_m, $
                     lineclosed_m, L_m, Bmax_m, Bavg_m, $
                     B1_m, B2_m)
```

2.2.1 Input parameters

- `Nlines` is the number of field lines, long integer.
- `lon_arr`, `lat_arr`, `r_arr`, `Bph`, `Bth`, `Br` and `ds` are the same as in the previous function `LineTrace`.
- `lon0_m`, `lat0_m` and `r0_m` are the `Nlines`-element double-precision floating-point arrays specifying the coordinates of the initial points of the field lines: (`lon0_m[j]`, `lat0_m[j]`, `r0_m[j]`) provide the coordinates $(\varphi_*, \theta_*, r_*)$ for the jth line in the same way as in the previous function.

2.2.2 Return value

The function returns the `Nlines`-element long-integer array `N_m`$= \{N\}$ specifying the number of points in each field line (some elements can be zero if the corresponding initial point is located outside the boundaries of the data cube).

2.2.3 Output parameters

- `l_lon_m`, `l_lat_m` and `l_r_m` are two-dimensional $N_{\max}\times$`Nlines` double-precision floating-point arrays specifying the coordinates of the found points along the field lines so that (`l_lon_m[i, j]`, `l_lat_m[i, j]`, `l_r_m[i, j]`) provide the coordinates (φ, θ, r) of the ith point along the jth field line, in the same way as in the previous function `LineTrace`; these values are defined only for $i <$`N_m[j]`. $N_{\max}$ is a number sufficient to accommodate all results, i.e. $N_{\max} \geq \max(N)$.
- `lineclosed_m`, `L_m`, `Bmax_m`, `Bavg_m`, `B1_m` and `B2_m` are one-dimensional `Nlines`-element arrays (long-integer for `lineclosed_m` and double-precision floating-point for other parameters) specifying various parameters of the respective field lines, in the same way as in the previous function `LineTrace`; they are defined only for the lines with `N_m[j]`> 0.

2.3 `LineTraceShort`

This function has the following syntax:

```
N = LineTraceShort(lon_arr, lat_arr, r_arr, Bph, Bth, Br, $
                   lon0, lat0, r0, ds, $
                   lineclosed, L, Bmax, Bavg, B1, B2)
```

It is similar to the function `LineTrace` with the only difference that the list of output parameters does not contain the coordinates of the points along the field line φ_l, θ_l and r_l; i.e., this function only determines whether the field line is closed and computes the line length, maximum and average field strengths along the line and the magnetic field strengths at the line endpoints.

2.4 `LineTraceShortMulti`

This function has the following syntax:

```
N_m = LineTraceShortMulti(Nlines, lon_arr, lat_arr, r_arr, $
                          Bph, Bth, Br, $
                          lon0_m, lat0_m, r0_m, ds, $
                          lineclosed_m, L_m, Bmax_m, Bavg_m, $
                          B1_m, B2_m)
```

It is similar to the function `LineTraceMulti` with the only difference that the list of output parameters does not contain the coordinates of the points along the field lines φ_l, θ_l and r_l; i.e., this function only determines whether the field lines are closed and computes the line lengths, maximum and average field strengths along the lines and the magnetic field strengths at the endpoints of the lines.

2.5 Notes

When computing the field line, the code starts from the supplied initial point and performs numerical integration firstly in the direction of the local magnetic field $\mathbf{B}$ and then in the opposite direction $-\mathbf{B}$. The integration stops when

a) The field line crosses a boundary of the considered data cube.

b) A point with zero magnetic field is encountered.

c) The number of integration steps (in either forward or backward direction) exceeds a certain limit which is currently set to 10^6.

The latter limitation was introduced to deal with an error in some magnetic field extrapolation codes when the condition $\nabla\mathbf{B} = 0$ is not always strictly satisfied, which can result in artificial “magnetic null points” and infinite loops in the line-tracing algorithm. This also means that extremely small values of the integration step `ds` should not be used.

The ability of the field lines to cross the gap between $\varphi_{N_\varphi-1}$ and φ_0 (similar to the `/closed` keyword of the rendering code) has not been implemented yet.

3 Notes

- The executable libraries (`*.dll` or `*.so` for Windows or Linux, respectively) are assumed to be located in the same folder as the IDL routines `*.pro`.
- The types and sizes of the input parameters must match exactly the above specifications; otherwise, the results are completely inpredictable (the codes currently do not check the validity of the supplied parameters).