Rendering code for the spherical coordinate system

The purpose of this 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 $r > 0$.

Computing the values of magnetic field, plasma density, etc. at the points along the line(s) of sight is beyond the scope of this code; however, some implementation details followed the assumption that, e.g., the coronal magnetic field is specified by its values at the above mentioned grid nodes, so that the values at other points can be found via interpolation.

The code is implemented as IDL functions `RenderSpherical` (single-thread computation) and `RenderSphericalMulti` (utilizes the multi-thread capabilities), which then call the respective Windows/Linux executable library.

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 Input parameters

- `lon_arr` is the array of longitude grid points $\{\varphi_i\}$ [degrees], double-precision floating-point.

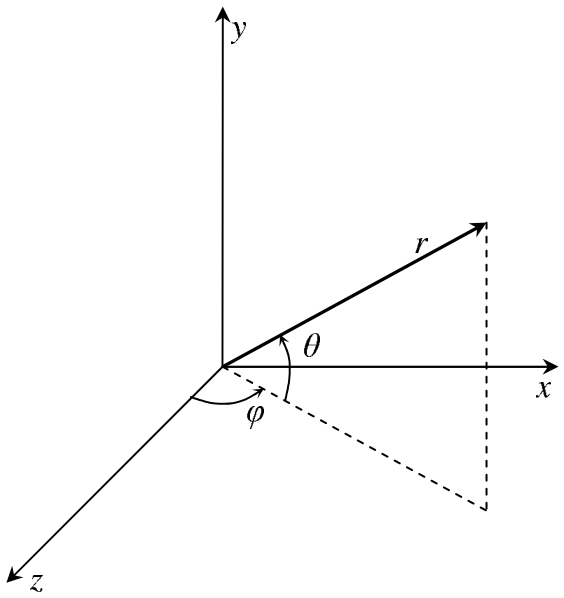

Figure 1: Spherical coordinate system.

- `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`= $\{r, \varphi, \theta\}_{\rm start}$ 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`= $\{r, \varphi, \theta\}_{\rm end}$ is a 3-element double-precision floating-point array specifying the end point of the line of sight.

- `rmin` is the effective photosphere radius $r_{\rm min}$ [in the units of solar radius], double-precision floating-point scalar. If $r_{\rm min} > 0$, the lines of sight intersecting the sphere $r = r_{\rm min}$ are truncated so that only their parts that can contribute to the observable emission (if any) are considered.
  Note: $r_{\rm min}$ should not exceed the lower boundary of the data cube; if $r_{\rm min} > r_0$, r_0 will be used instead as the effective photosphere radius.

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 $r_{\rm min}$), long integer.

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`.

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

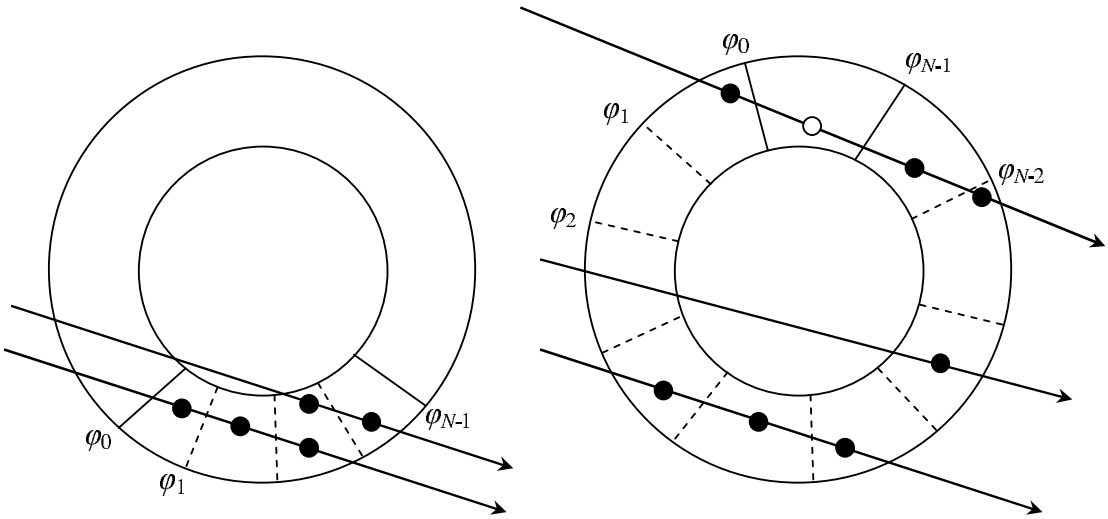

Figure 2: Voxels in the "open" and "closed" models.

PFSS extrapolation. In this case, the region with $\varphi < \varphi_0$ or $\varphi \geq \varphi_{N_\varphi - 1}$ 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

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

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

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

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])
```

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.

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).

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.

2.4 Keywords

Same as in the function `RenderSpherical`.

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.
- The executable libraries (`dll` or `so` for Windows or Linux) are assumed to be located in the same folder as the IDL routines `RenderSpherical.pro` and/or `RenderSphericalMulti.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).