sphericalrht

Release 2.0.1

George Halal

Apr 29, 2023

CONTENTS:

1 Input Parameters **3**
- 1.1 Required . . . 3
- 1.2 Optional . . . 3

2 Usage **5**
- 2.1 Example 1 . . . 5
- 2.2 Example 2 . . . 6
- 2.3 Reading the results . . . 6

3 References & Attribution **7**

4 Links **9**

Contents

- *Installation*
- *Input Parameters*
  - *Required*
  - *Optional*
- *Usage*
  - *Example 1*
  - *Example 2*
  - *Reading the results*
- *References & Attribution*
- *Links*

Run the following in a terminal to install:

```
$ pip install sphericalrht
```

If installing on a computing cluster, you may want to run the following instead:

```
$ pip install sphericalrht --user
```

Add the `--upgrade` option to upgrade to a newer version.

Note that the `sphericalrht` package requires at least Python 3.7.

CHAPTER

ONE

INPUT PARAMETERS

1.1 Required

in_map: Union[str, Tuple[np.ndarray, str]]
: either a path to the input intensity map or a tuple of the input intensity map as an array along with its name as a str, which is used for saving log file, alms, spherical RHT cube, and Stokes Q/U maps. The rest of the input options will be appended to this name when saving. The input map ordering is assumed to be RING.

nside: int
: output NSIDE for intensity and Stokes Q/U maps.

out_dir: str
: directory to save log file, alms, spherical RHT cube, and Stokes Q/U maps in COSMO/Healpix convention.

1.2 Optional

wlen: int
: convolution kernel window diameter [arcmins] (the scale at which to measure the orientation).

fwhm: float
: scale [arcmins] for the unsharp mask applied to pick out filamentary structure.

thresh: float
: threshold fraction of the window diameter between 0-1 applied to the result of the convolution. Higher thresholds focus on the main orientations only, while lower thresholds take more orientations into account, weighted by their intensity.

norients: int
: angular resolution given by the number of orientations to consider. We have found empirically that `norients` = 25 is sufficient for most applications.

mask: Union[str, np.ndarray]
: either a path to the map or an array of the map pixels. This defines the mask for maps that are not defined over the entire sky. The map ordering is assumed to be RING.

weighting: Union[str, np.ndarray]
: either a path to the map or an array of the map pixels. This is used as the weight for the output Stokes Q/U maps. The map ordering is assumed to be RING.

overwrite: bool
: whether to overwrite outputs of same name if they already exist.

split_factor: int
: number of data splits to save on runtime memory usage. Default value is based on the requested NSIDE (1 for

NSIDE < 4096). If your job runs out of memory and you can't request more memory per job, increase the split factor.

CHAPTER

TWO

USAGE

The code runs in parallel on as many CPUs as available, so feel free to request many CPUs when submitting a job. The runtime and memory increase as O(`nside`^{2} * `norients`). The `split_factor` can be used to decrease the memory complexity and increase the time complexity as O(`split_factor`). The other input parameters have a negligible effect on the time and memory complexity.

The polarization convention is COSMO with the polarization being perpendicular to the filaments in the map in the plane of the sky. To obtain polarization maps corresponding to the IAU convention, multiply U by -1. To obtain polarization maps parallel to the orientation of the filaments (e.g. magnetic field orientation) in the COSMO convention, multiply both Q and U by -1.

2.1 Example 1

Here's one way to run the algorithm with all the input parameters:

```
from sphericalrht import CubeAndStokes

cube_and_stokes = CubeAndStokes(
    in_map="/path/to/map_name.fits",
    nside=1024,
    out_dir="/path/to/output_dir",
    wlen=75,
    fwhm=30,
    thresh=0.7,
    norients=25,
    weighting="/path/to/weighting_map.fits"
    overwrite=False,
    split_factor=1)

cube_and_stokes.build_and_save()
```

2.2 Example 2

If your input map is an array instead of a .fits file, you can enter a tuple with the array in the first entry and the name as the second entry as shown in this example that uses only the required input parameters:

```
import h5py

with h5py.File("/path/to/map_name.h5", "r") as f:
    intensity = f["I"][:, 0]

from sphericalrht import CubeAndStokes

cube_and_stokes = CubeAndStokes(
    in_map=(intensity, "map_name"),
    nside=1024,
    out_dir="/path/to/output_dir")

cube_and_stokes.build_and_save()
```

2.3 Reading the results

```
# Load the output maps
import healpy as hp

out_name = "map_name_nside1024_wlen75_fwhm30_thresh0.7_norients25"

I, Q, U = hp.read_map(
    f"/path/to/output_dir/IQU_{out_name}.fits", field=(0,1,2))


# If you'd like, you can also load the output of
# all orientation angles for each pixel
import h5py

with h5py.File("/path/to/output_dir/{out_name}.h5") as cube_file:
    spherical_rht_out = cube_file["spherical_rht_cube"][:, PIXEL_INDEX]
```

CHAPTER

THREE

REFERENCES & ATTRIBUTION

The paper introducing this package is in preparation. If you make use of this code in your research, please contact halalgeorge@gmail.com for discussing proper citations.

CHAPTER

FOUR

LINKS