Memo: SkyH5 file format

Bryna Hazelton, and the pyradiosky team

October 5, 2023

Contents

1 Introduction

This memo introduces a new HDF5[1] based file format of a `SkyModel` object in `pyradiosky`[2], a python package that provides objects and interfaces for representing diffuse, extended and compact astrophysical radio sources. Here, we describe the required and optional elements and the structure of this file format, called *SkyH5*.

We assume that the user has a working knowledge of HDF5 and the associated python bindings in the package `h5py`[3], as well as `SkyModel` objects in `pyradiosky`. For more information about HDF5, pleasevisit `https://portal.hdfgroup.org/display/HDF5/HDF5`. For more information about the parameters present in a `SkyModel` object, please visit `https://pyradiosky.readthedocs.io/en/latest/skymodel.html`. Examples of how to interact with `SkyModel` objects in `pyradiosky` are available at `http://pyradiosky.readthedocs.io/en/latest/tutorial.html`.

Note that throughout the documentation, we assume a row-major convention (i.e., C-ordering) for the dimension specification of multi-dimensional arrays. For example, for a two-dimensional array with shape (N, M), the M-dimension is varying fastest, and is contiguous in memory. This convention is the same as Python and the underlying C-based HDF5 library. Users of languages with the opposite column-major convention (i.e., Fortran-ordering, seen also in MATLAB and Julia) must transpose these axes.

2 Overview

A SkyH5 object contains data representing catalogs and maps of astrophysical radio sources, including the associated metadata necessary to interpret them. A SkyH5 file contains two

[1] `https://www.hdfgroup.org/`
[2] `https://github.com/RadioAstronomySoftwareGroup/pyradiosky`
[3] `https://www.h5py.org/`

primary HDF5 groups: the `Header` group, which contains the metadata, and the `Data` group, which contains the Stokes parameters representing the flux densities or brightness temperatures of the sources (as well as some optional arrays the same size as the stokes parameters data). Datasets in the `Data` group are can be passed through HDF5's compression pipeline, to reduce the amount of on-disk space required to store the data. However, because HDF5 is aware of any compression applied to a dataset, there is little that the user has to explicitly do when reading data. For users interested in creating new files, the use of compression is optional in the SkyH5 format, because the HDF5 file is self-documenting in this regard.

Many of the datasets in SkyH5 files have units associated with them (represented as `astropy Quantity` objects on the `SkyModel` object). The units are stored as attributes on the datasets with the name "unit". Datasets that derive from other `astropy` objects (e.g. `astropy Time, astropy EarthLocation, astropy Latitude, astropy Longitude`) also have an "object_type" attribute indicating the object type.

In the discussion below, we discuss required and optional datasets in the various groups. We note in parenthesis the corresponding attribute of a `SkyModel` object. Note that in nearly all cases, the names are coincident, to make things as transparent as possible to the user.

3 Header

The `Header` group of the file contains the metadata necessary to interpret the data. We begin with the required parameters, then continue to optional ones. Unless otherwise noted, all datasets are scalars (i.e., not arrays). The precision of the data type is also not specified as part of the format, because in general the user is free to set it according to the desired use case (and HDF5 records the precision and endianness when generating datasets). When using the standard `h5py`-based implementation in `pyradiosky`, this typically results in 32-bit integers and double precision floating point numbers. Each entry in the list contains **(1)** the exact name of the dataset in the HDF5 file, in boldface, **(2)** the expected datatype of the dataset, in italics, **(3)** a brief description of the data, and **(4)** the name of the corresponding attribute on a `SkyModel` object, italicized and in parentheses at the end of the entry.

Note that string datatypes should be handled with care. See the Appendix in the UVH5 memo (`https://github.com/RadioAstronomySoftwareGroup/pyuvdata/blob/main/docs/references/uvh5_memo.pdf`) for appropriately defining them for interoperability between different HDF5 implementations.

3.1 Required Parameters

- **component_type**: *string* The type of components in the sky model. The options are: "healpix" and "point". If component_type is "healpix", the components are the pixels in a HEALPix map in units compatible with K or Jy/sr. If the component_type is "point", the components are point-like sources, or point like components of extended sources, in units compatible with Jy or K sr. Some additional parameters are required depending on the component type. (*component_type*)

- **Ncomponents**: *int* The number of components in the sky model. This can be the number of individual compact sources, or it can include components of extended sources, or the number of pixels in a map. (*Ncomponents*)

- **spectral_type**: *string* This describes the type of spectral model for the components. The options are:

  1. **spectral_index** The convention for the spectral index is $I = I_0 \frac{f}{f_0}^{\alpha}$, where I_0 is the stokes parameter at the reference_frequency parameter f_0 and α is the spectral_index parameter. Note that the spectral index is assumed to apply in the units of the stokes parameter (i.e. there is no additive factor of 2 applied to convert between brightness temperature and flux density units). If the spectral model uses a spectral index, the reference_frequency and spectral_index parameters are required.
  2. **subband** The subband spectral model is used for catalogs with multiple flux measurements at different frequencies (i.e. GLEAM `https://www.mwatelescope.org/science/galactic-science/gleam/`). For subband spectral models, the freq_array and freq_edge_array parameters are required to give the nominal (usually the central) frequency and the top and bottom of each subband respectively.
  3. **flat** The flat spectral model assumes no spectral flux dependence, which can be useful for testing. Since the flux is assumed to be the same at all frequencies it does not require any extra parameters to be defined.
  4. **full** The full spectral model is used for catalogs with flux values at multiple frequencies that are not expected to have flux correlations as a function of frequency, so cannot not be interpolated to frequencies not included in the catalog. This is a good representation for e.g. Epoch of Reionization signal cubes. For full spectral models, the freq_array parameter is required to give the frequencies.

  (*spectral_type*)

- **Nfreqs**: *int* Number of frequencies if spectral_type is "full" or "subband", 1 otherwise. (*Nfreqs*)

- **history**: *string* The history of the catalog. (*history*)

3.2 Optional Parameters

- **name**: *string* The name for each component. This is a one-dimensional array of size (Ncomponents). Note this is **required** if the component_type is "point". (*name*)
- **skycoord**: A nested dataset that contains the information to create an `astropy SkyCoord` object representing the component positions. Note this is **required** if the component_type is "point". The keys must include:
  - **frame**: *string* The name of the coordinate frame (e.g. "icrs", "fk5", "galactic"). Must be a frame supported by `astropy`. Note that only one frame is allowed, which applies to all the components.
  - **representation_type**: *string* The representation type, one of "spherical", "cartesian" or "cylindrical". This sets what the coordinate names can be. It is most common to set this to "spherical" and specify latitudinal and longitudinal coordinates (e.g. **ra**, **dec**) and optionally distance coordinates. It is also possible to use other representations such as cartesian or cylindrical, e.g. for "cartesian" the coordinates would be specified in x, y, and z. See the `astropy SkyCoord` docs for more details.
  - Coordinate names (e.g. **ra**, **dec**, **alt**, **az**): *float* Two or three such components must be present, which ones are required depend on the frame and representation_type. These are one-dimensional arrays of size (Ncomponents).

  And may include any other attributes accepted as input parameters for an `astropy SkyCoord` object (e.g. obstime, equinox, location). Each of these datasets may have "unit" and "object_type" attributes and may be either a scalar or a one-dimensional array of size (Ncomponents) as appropriate. (*skycoord*)
- **nside**: *int* The HEALPix nside parameter. Note this is **required** if the component_type is "healpix" and should not be defined otherwise. (*nside*)
- **hpx_order**: *string* The HEALPix pixel ordering convention, either "ring" or "nested". Note this is **required** if the component_type is "healpix" and should not be defined otherwise. (*hpx_order*)
- **hpx_frame**: A nested dataset that contains the information to describe an `astropy` coordinate frame giving the HEALPix coordinate frame. This is similar to the skycoord dataset described above but it does not contain the representation_type or the coordinate names. Note this is **required** if the component_type is "healpix" and should not be defined otherwise. The keys must include:
  - **frame**: *string* The name of the coordinate frame (e.g. "icrs", "fk5", "galactic"). Must be a frame supported by `astropy`.

And may include any other scalar attributes accepted as input parameters for an `astropy SkyCoord` object (e.g. obstime, equinox, location). Each of these datasets may have "unit" and "object_type" attributes as appropriate. (*hpx_frame*)

- **hpx_inds**: *int* The HEALPix indices for the included components. Does not need to include all the HEALPix pixels in the map. This is a one-dimensional array of size (Ncomponents). Note this is **required** if the component_type is "healpix" and should not be defined otherwise. (*hpx_inds*)
- **freq_array**: *float* Frequency array giving the nominal (or central) frequency in a unit that can be converted to Hz. Note this is **required** if the spectral_type is "full" or "subband" and should not be defined otherwise. (*freq_array*)
- **freq_edge_array**: *float* Array giving the frequency band edges in a unit that can be converted to Hz. This is a two dimensional array with shape (2, Nfreqs). The zeroth index in the first dimension holds the lower band edge and the first index holds the upper band edge. Note this is **required** if the spectral_type is "subband" and should not be defined otherwise. (*freq_edge_array*)
- **reference_frequency**: *float* Reference frequency giving the frequency at which the flux in the stokes parameter was measured in a unit that can be converted to Hz. This is a one-dimensional array of size (Ncomponents). Note this is **required** if the spectral_type is "spectral_index" and should not be defined if the spectral_type is "full" or "subband". (*reference_frequency*)
- **spectral_index**: *float* The spectral index describing the flux evolution with frequency, see details in the spectral_type description above. This is a one-dimensional array of size (Ncomponents). Note this is **required** if the spectral_type is "spectral_index" and should not be defined otherwise. (*spectral_index*)
- **extended_model_group**: *string* Identifier that groups components of an extended source model. This is a one-dimensional array of size (Ncomponents), with empty strings for point sources that are not components of an extended source. (*extended_model_group*)

3.3 Extra Columns

`SkyModel` objects support "extra columns", which are additional arbitrary per-component arrays of metadata that are useful to carryaround with the data but which are not formally supported as a reserved keyword in the `Header`. In a SkyH5 file, extra columns are handled by creating a datagroup called **extra_columns** inside the `Data` datagroup. When possible, these quantities should be HDF5 datatypes, to support interoperability between SkyH5 readers. Inside of the extra_columns datagroup, each extra columns is saved as a key-value pair using a dataset, where the name of the extra columns is the name of the dataset and its corresponding array is saved in the dataset. The “unit” and “object_type” HDF5 attributes are used in the same way as for the other header items (see **??**), but it is not recommended to use other attribute names due to the lack of support inside of `pyradiosky` for ensuring the attributes are properly saved when reading and writing SkyH5 files.

4 Data

In addition to the `Data` datagroup in the root namespace, there must be one called `Data`. This datagroup saves the Stokes parameters representing the flux densities or brightness temperatures of the sources and some optional arrays that are the same size. They are also all expected to be the same shape: (4, Nfreqs, Ncomponents) where the first dimension indexes the polarization direction, ordered (I, Q, U, V). The **stokes** dataset must be present in this datagroup and it must have a “unit” attribute that is equivalent to Jy or K str if the component_type is “point” or equivalent to Jy/str or K if the component_type is “healpix”. In addition, this datagroup may also contain a **stokes_error** dataset that gives the standard error on the stokes values and should have the same “unit” attribute as the stokes dataset and a **beam_amp** dataset that gives the beam amplitude at the source position for the instrument that made the measurement.