nxarray

Aug 11, 2022

Contents

A software package for NeXus/HDF5 conversion from and to Python xarray.

`nxarray` extends xarray *DataArrays* and *Datasets* with a high-level python interface for NeXus/HDF5 file input and output.

CHAPTER 1

Installation

You can install nxarray with pip:

```
$ pip install nxarray
```

1.1 Prerequisites

nxarray is built on and depends on `nexusformat` and `xarray` packages:

- nexusformat
- xarray

CHAPTER 2

Usage

After installation, import nxarray with:

```
>>> import nxarray
```

Now the `nxr.save()` method will be available to xarray *Datasets*. To save an existing *Dataset* to a NeXus file simply type:

```
>>> ds = xarray.Dataset()
>>> ds.nxr.save('path/to/file.nx')
```

To load a NeXus file into an xarray *Dataset* use the `nxarray.load()` function:

```
>>> ds = nxarray.load('path/to/file.nx')
```

The default *NXentry* in the NeXus file will be loaded into the *Dataset*, with all its subgroups (*NXdata*, *NXinstrument*, *NXsample*...).

Note that just a single *NXentry* at once can be loaded into a *Dataset*. To load a different *NXentry*, specify it using the `entry=` argument:

```
>>> ds = nxarray.load('path/to/file.nx', entry="myentry")
```

Upon loading, the fields in the *NXdata* groups within the *NXentry* are loaded into *data variable* and *coordinates* of the dataset, with their relevant attributes:

```
>>> ds
```

The NeXus tree of the *NXentry* with all the subgroups (*NXinstrument*, *NXsample*...) is stored in the `NXtree` attribute of the *Dataset* (TAB completion can be used on `NXtree`).

```
>>> ds.NXtree
    data:NXdata
      @axes = 'energy'
      @energy_indices = 0
```

(continues on next page)

(continued from previous page)

```
      @signal = 'absorbed_beam'
    instrument:NXinstrument
      source:NXsource
      current = 308.52
        @units = 'mA'
```

```
>>> ds.NXtree.instrument
    NXinstrument('instrument')
```

All xarray methods and attributes are accesible as usual. E.g. to plot the default signal:

```
>>> ds.absorbed_beam.plot()
```

For more info on the resulting *Dataset* structure and the architecture of `nxarray` look at the *Design section*.

2.1 Examples

Let's start by importing:

```
import numpy as np
import xarray as xr
import nxarray as nxr
```

and creating a dataset `ds`:

```
ds = xr.Dataset()
data = xr.DataArray(np.random.randn(2, 3),
                    dims=('x', 'y'),
                    coords={'x': [10, 20], 'y': [1,2,3]},
                    name='some_data')
ds['MyData'] = data
```

The `ds` *Dataset* can be saved to a NeXus file to disk simply with:

```
ds.nxr.save('ds.nxs')
```

You can load it back, let's say to another *Dataset* `ds2` with:

```
ds2 = nxarray.load('ds.nxs')
```

and you can check that the whole structure of your *Dataset* is preserved. Additionally, the `NXtree` attribute is present (in this example containing zero objects).

2.2 Naming conventions

Note that the `nxr` accessor for xarray objects will always be available with this naming, independently of the shorthand used when import nxarray.

CHAPTER 3

Design

The architecture of a NeXus file resembles the structure of an xarray *Dataset*, with some important differences. In the following it is assumed the reader is familiar with the nomenclature of xarray and NeXus NXdata.

The following table summarize the correspondence brought by nxarray between NeXus and xarray objects and definitions.

NeXus	xarray
`NXentry`	`Dataset.NXtree` (*)
`NXdata.entries`	`Dataset.data_vars, Dataset.coords` (**)
signal	*data variable*
`NXdata.nxaxes`	`Dataset.dims`
axes	*dimensions*

(*) The complete structure of the NXentry is loaded into the `NXtree` Dataset attribute, with the exception of the entries in `NXdata.entries` which are loaded into the Dataset *data variables* and *coordinates* as DataArrays (see below).

(**) The entries in `NXdata.entries` are loaded into the Dataset *data variables* and *coordinates* as DataArrays, provided the attributes `@signal` and `@axes` are present in the *NXdata* group. *NXlinks* are resolved transparently and are kept when saving back to NeXus. The entry attributes are assigned to the correspondent DataArray. Additionally, the `nxgroup` attribute is added to each DatArray and its value is set to the name of the *NXdata* group (`NXdata.nxname`).

The identification of an entry as *data variable* or *coordinate* is performed as follows:

- An entry referred by the `@signal` attribute of *NXdata* is considered a Dataset *data variable*.
- An entry is considered a *coordinate* if:
  - it is listed in the `@axes` attribute of *NXdata* or
  - an attribute `AXIS_indices` is present in the *NXdata* group
- Any other entry:
  - is considered a *data variable* if its shape matches the `@signal` field shape

- is disregarded otherwise.

CHAPTER 4

Motivations

xarray is the most used Python package for labeled multi-dimensional data handling, providing convenient data structures (namely *DataArrays* and *Datasets*) and including a large library of functions for advanced analytics and visualization with these data structures.

Despite xarray supports natively import/export of HDF5 (a file format designed to efficiently store and organize large amount of data), it does not provide an integrated interface to the NeXus file format, the standard *de facto* for scientific data storage, based on HDF5 and increasingly adopted in laboratories and large-scale facilities all over the world.

With this respect, the nxarray package comes into play, bridging xarray with the NeXus format. This package actually extends xarray, providing convenient loading and saving methods for NeXus files, directly to *Datasets* objects. The architecture of a NeXus file resembles the structure of an xarray *Dataset*, and indeed both of them are specifically designed for handling scientific data with its relevant metadata.

`nxarray` is part of the reScipy project.

CHAPTER 5

Feedback

Please report any feedback, bugs, or feature requests by opening an issue on the issue tracker of the code repository. You should provide as much information as possible to reproduce the problem, and details of your desiderata.