Simulating X-ray Observations with Python

John A. ZuHone[‖*], Veronica Biffi[¶], Eric J. Hallman[§], Scott W. Randall[‡], Adam R. Foster[‡], Christian Schmid[**]

http://www.youtube.com/watch?v=fUMq6rmNshc

Abstract—X-ray astronomy is an important tool in the astrophysicist's toolkit to investigate high-energy astrophysical phenomena. Theoretical numerical simulations of astrophysical sources are fully three-dimensional representations of physical quantities such as density, temperature, and pressure, whereas astronomical observations are two-dimensional projections of the emission generated via mechanisms dependent on these quantities. To bridge the gap between simulations and observations, algorithms for generating synthetic observations of simulated data have been developed. We present an implementation of such an algorithm in the `yt` analysis software package. We describe the underlying model for generating the X-ray photons, the important role that `yt` and other Python packages play in its implementation, and present a detailed workable example of the creation of simulated X-ray observations.

Index Terms—astronomical observations, astrophysics simulations, visualization

Introduction

In the early 21st century, astronomy is truly a multi-wavelength enterprise. Ground and space-based instruments across the electromagnetic spectrum, from radio waves to gamma rays, provide the most complete picture of the various physical processes governing the evolution of astrophysical sources. In particular, X-ray astronomy probes high-energy processes in astrophysics, including high-temperature thermal plasmas (e.g., the solar wind, the intracluster medium) and relativistic cosmic rays (e.g., from active galactic nuclei). X-ray astronomy has a long and successful pedigree, with a number of observatories. These include *Einstein*, *ROSAT*, *Chandra*, *XMM-Newton*, *Suzaku*, and *NuSTAR*, as well as upcoming missions such as *Astro-H* and *Athena*.

An important distinguishing feature of X-ray astronomy from that of studies at longer wavelengths is that it is inherently *discrete*, e.g., the numbers of photons per second that reach the detectors are small enough that the continuum approximation, valid for longer-wavelength photons such as those in the visible light, infrared, microwave, and radio bands, is invalid. Instead of images, the fundamental data products of X-ray astronomy are tables of individual photon positions, energies, and arrival times.

Due to modeling uncertainties, projection effects, and contaminating backgrounds, combining the insights from observations and numerical simulations is not necessarily straightforward. In contrast to simulations, where all of the physical quantities in 3 dimensions are completely known to the precision of the simulation algorithm, astronomical observations are by definition 2-D projections of 3-D sources along a given sight line, and the observed spectrum of emission is a complicated function of the fundamental physical properties (e.g., density, temperature, composition) of the source.

Such difficulties in bridging these two worlds have given rise to efforts to close the gap in the direction of the creation of synthetic observations from simulated data (see, e.g., [Gardini04], [Nagai06], [ZuHone09], and [Heinz11] for recent examples). This involves the determination of the spectrum of the emission from the properties of the source, the projection of this emission along the chosen line of sight, and, in the case of X-ray (and γ-ray) astronomy, the generation of synthetic photon samples. These photons are then convolved with the instrumental responses and (if necessary) background effects are added. One implementation of such a procedure, `PHOX`, was described in [Biffi12] and [Biffi13], and used for the analysis of simulated galaxy clusters from smoothed-particle hydrodynamics (SPH) cosmological simulations. `PHOX` was originally implemented in C using outputs from the `Gadget` SPH code. `PHOX` takes the inputs of density, temperature, velocity, and metallicity from a 3D `Gadget` simulation, using them as inputs to create synthetic spectra (using spectral models from the X-ray spectral fitting package `XSPEC`). Finally, `PHOX` uses these synthetic spectra convolved with instrument response functions to simulate samples of observed photons.

In this work, we describe an extension of this algorithm to outputs from other simulation codes. We developed the module `photon_simulator`, an implementation of `PHOX` within the Python-based `yt` simulation analysis package. We outline the design of the `PHOX` algorithm, the specific advantages to implementing it in Python and `yt`, and provide a workable example of the generation of a synthetic X-ray observation from a simulation dataset.

Model

The overall model that underlies the `PHOX` algorithm may be split up into roughly three steps: first, constructing an original large sample of simulated photons for a given source, second, choosing

* *Corresponding author: jzuhone@milkyway.gsfc.nasa.gov*
‖ *Astrophysics Science Division, Laboratory for High Energy Astrophysics, Code 662, NASA/Goddard Space Flight Center, Greenbelt, MD 20771*
¶ *SISSA - Scuola Internazionale Superiore di Studi Avanzati, Via Bonomea 265, 34136 Trieste, Italy*
§ *Center for Astrophysics and Space Astronomy, Department of Astrophysical & Planetary Science, University of Colorado, Boulder, CO 80309*
‡ *Harvard-Smithsonian Center for Astrophysics, 60 Garden Street, Cambridge, MA 02138*
** *Dr. Karl Remeis-Sternwarte & ECAP, Sternwartstr. 7, 96049 Bamberg, Germany*

a subset of these photons corresponding to parameters appropriate to an actual observation and projecting them onto the sky plane, and finally, applying instrumental responses for a given detector. We briefly describe each of these in turn.

Step 1: Generating the Original Photon Sample

In the first step of the PHOX algorithm, we generate a large sample of photons in three dimensions, with energies in the rest frame of the source. These photons will serve as a "Monte-Carlo" sample from which we may draw subsets to construct realistic observations.

First, to determine the energies of the photons, a spectral model for the photon emissivity must be specified. In general, the normalization of the photon emissivity for a given volume element will be set by the number density of emitting particles, and the shape of the spectrum will be set by the energetics of the same particles.

As a specific and highly relevant example, one of the most common sources of X-ray emission is that from a low-density, high-temperature, thermal plasma, such as that found in the solar corona, supernova remnants, "early-type" galaxies, galaxy groups, and galaxy clusters. The specific photon count emissivity associated with a given density, temperature T, and metallicity Z of such a plasma is given by

$$\varepsilon_E^{\gamma} = n_e n_H \Lambda_E(T,Z) \text{ photons s}^{-1} \text{ cm}^{-3} \text{ keV}^{-1} \quad (1)$$

where the superscript γ refers to the fact that this is a photon count emissivity, E is the photon energy in keV, n_e and n_H are the electron and proton number densities in cm^{-3}, and $\Lambda_E(T,Z)$ is the spectral model in units of photons s^{-1} cm^3 keV^{-1}. The dominant contributions to Λ_E for an optically-thin, fully-ionized plasma are bremmstrahlung ("free-free") emission and collisional line excitation. A number of models for the emissivity of such a plasma have been developed, including Raymond-Smith [Raymond77], MeKaL [Mewe95], and APEC [Smith01]. These models (and others) are all built into the XSPEC package, which includes a Python interface, PyXspec, which is a package we will use to supply the input spectral models to generate the photon energies.

The original PHOX algorithm only allowed for emission from variants of the APEC model for a thermal plasma. However, astrophysical X-ray emission arises from a variety of physical processes and sources, and in some cases multiple sources may be emitting from within the same volume. For example, cosmic-ray electrons in galaxy clusters produce a power-law spectrum of X-ray emission at high energies via inverse-Compton scattering of the cosmic microwave background. Recently, the detection of previously unidentified line emission, potentially from decaying sterile neutrinos, was made in stacked spectra of galaxy clusters [Bulbul14]. The flexibility of our approach allows us to implement one or several models for the X-ray emission arising from a variety of physical processes as the situation requires.

Given a spectral model, for a given volume element i with volume ΔV_i (which may be grid cells or Lagrangian particles), a spectrum of photons may be generated. The *total number* of photons that are generated in our initial sample per volume element i is determined by other factors. We determine the number of photons for each volume element by artificially inflating the parameters that determine the number of photons received by an observer to values that are large compared to more realistic values. The inflated Monte-Carlo sample should be large enough that realistic sized subsets from it are statistically representative. In the description that follows, parameters with subscript "0" indicate those with "inflated" values, whereas we will drop the subscripts in the second step when choosing more realistic values.

To begin with, the bolometric flux of photons received by the observer from the volume element i is

$$F_i^{\gamma} = \frac{n_{e,i} n_{H,i} \Lambda(T_i,Z_i) \Delta V_i}{4\pi D_{A,0}^2 (1+z_0)^2} \text{ photons s}^{-1} \text{ cm}^{-2} \quad (2)$$

where z_0 is the cosmological redshift and $D_{A,0}$ is the angular diameter distance to the source (if the source is nearby, $z_0 \approx 0$ and $D_{A,0}$ is simply the distance to the source). The physical quantities of interest are constant across the volume element. The total number of photons associated with this flux for an instrument with a collecting area $A_{\rm det,0}$ and an observation with exposure time $t_{\rm exp,0}$ is given by

$$N_{\rm phot} = t_{\rm exp,0} A_{\rm det,0} \sum_i F_i^{\gamma} \quad (3)$$

By setting $t_{\rm exp,0}$ and $A_{\rm det,0}$ to values that are much larger than those associated with typical exposure times and actual detector areas, and setting z_0 to a value that corresponds to a nearby source (thus ensuring $D_{A,0}$ is similarly small), we ensure that we create suitably large Monte-Carlo sample to draw subsets of photons for more realistic observational parameters. Figure 1 shows a schematic representation of this model for a roughly spherical source of X-ray photons, such as a galaxy cluster.

Step 2: Projecting Photons to Create Specific Observations

The second step in the PHOX algorithm involves using this large 3-D sample of photons to create 2-D projections of simulated events, where a subsample of photons from the original Monte-Carlo sample is selected.

First, we choose a line-of-sight vector $\hat{\mathbf{n}}$ to define the primed coordinate system from which the photon sky positions (x',y') in the observer's coordinate system $\mathcal{O}'$ are determined (c.f. Figure 1). The total emission from any extended object as a function of position on the sky is a projection of the total emission along the line of sight, minus the emission that has been either absorbed or scattered out of the sight-line along the way. In the current state of our implementation, we assume that the source is optically thin to the photons, so they pass essentially unimpeded from the source to the observer (with the caveat that some photons are absorbed by Galactic foreground gas). This is appropriate for most X-ray sources of interest.

Next, we must take into account processes that affect on the photon energies. The first, occurring at the source itself, is Doppler shifting and broadening of spectral lines, which arises from bulk motion of the gas and turbulence. Each volume element has a velocity $\mathbf{v}_i$ in $\mathcal{O}$, and the component $v_{i,z'}$ of this velocity along the line of sight results in a Doppler shift of each photon's energy of

$$E_1 = E_0 \sqrt{\frac{c+v_{z'}}{c-v_{z'}}} \quad (4)$$

where E_1 and E_0 are the Doppler-shifted and rest-frame energies of the photon, respectively, and c is the speed of light in vacuum. Second, since many X-ray sources are at cosmological distances, each photon is cosmologically redshifted, reducing its energy further by a factor of $1/(1+z)$ before being received in the observer's frame.

Since we are now simulating an actual observation, we choose more realistic values for the exposure time $t_{\rm exp}$ and detector area

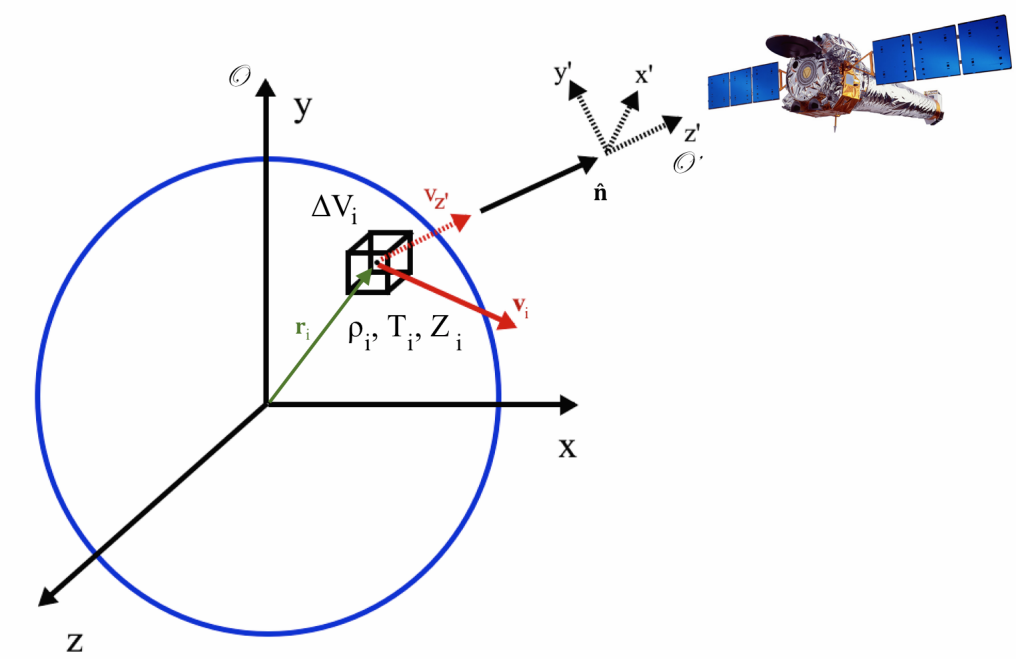

Fig. 1: *Schematic representation of a roughly spherical X-ray emitting object, such as a galaxy cluster. The volume element ΔV_i at position $\mathbf{r}_i$ in the coordinate system $\mathcal{O}$ of the source has a velocity $\mathbf{v}_i$. Photons emitted along the direction given by $\hat{\mathbf{n}}$ will be received in the observer's frame in the coordinate system $\mathcal{O}'$, and will be Doppler-shifted by the line-of-sight velocity component $v_{i,z'}$.* Chandra *telescope image credit: NASA/CXC.*

$A_{\rm det}$ than we did in the first step to determine the number of photons to use from the original Monte-Carlo sample. Similarly, we may also specify alternative values for the angular diameter distance D_A and the cosmological redshift z, if desired. The fraction f of the photons that will be used in the actual observation is then given by

$$f = \frac{t_{\rm exp}}{t_{\rm exp,0}} \frac{A_{\rm det}}{A_{\rm det,0}} \frac{D_{A,0}^2}{D_A^2} \frac{(1+z_0)^3}{(1+z)^3} \tag{5}$$

where $f \leq 1$.

Before being received by the observer, a number of the photons, particularly on the softer end of the spectrum, are absorbed by foregrounds of hydrogen gas in the Milky Way Galaxy. The last operation that is applied in our implementation of the `PHOX` algorithm is to use a tabulated model for the absorption cross-section as a function of energy (examples include `wabs` [Morrison83], `phabs` [Balucinska-Church92], `tbabs` [Wilms00], all included in `XSPEC`) as an acceptance-rejection criterion for which photons will be retained in the final sample, e.g., which of them are actually received by the observer.

The advantage of the `PHOX` algorithm is that the two steps of generating the photons in the source frame and projecting them along a given line of sight are separated, so that the first step, which is the most computationally expensive, need only be done once for a given source, whereas the typically cheaper second step may be repeated many times for many different lines of sight, different instruments, and different exposure times.

Step 3: Modeling Instrumental Effects

Unfortunately, the data products of X-ray observations do not simply consist of the original sky positions and energies of the received photons. Spatially, the positions of the received photons on the detector are affected by a number of instrumental factors. These include vignetting, the layout of the CCD chips, and a typically spatially dependent point-spread function. Similarly, the photon energies are binned up by the detectors into a set of discrete energy channels, and there is typically not a simple one-to-one mapping between which channel a given photon ends up in and its original energy, but is instead represented by a non-diagonal response matrix. Finally, the "effective" collecting area of the telescope is also energy-dependent, and also varies with position on the detector. When performing analysis of X-ray data, the mapping between the detector channel and the photon energy is generally encapsulated in a redistribution matrix file (RMF), and the effective area curve as a function of energy is encapsulated in an ancillary response file (ARF).

In our framework, we provide two ways of convolving the detected photons with instrumental responses, depending on the level of sophistication required. The first is a "bare-bones" approach, where the photon positions are convolved with a user-specified point-spread function, and the photon energies are convolved with a user-input energy response functions. This will result in photon distributions that are similar enough to the final data products of real observations to be sufficient for most purposes.

However, some users may require a full simulation of a given telescope or may wish to compare observations of the same simulated system by multiple instruments. Several software packages exist for this purpose. The venerable `MARX` software performs detailed ray-trace simulations of how *Chandra* responds to a variety of astrophysical sources, and produces standard event data files in the same FITS formats as standard *Chandra* data products. `SIMX` and `Sixte` are similar packages that simulate most of the effects of the instrumental responses for a variety of current and planned X-ray missions. We provide convenient output

formats for the synthetic photons in order that they may be easily imported into these packages.

Implementation

The model described here has been implemented as the analysis module `photon_simulator` in `yt` [Turk11], a Python-based visualization and analysis toolkit for volumetric data. `yt` has a number of strengths that make it an ideal package for implementing our algorithm.

The first is that `yt` has support for analyzing data from a large number of astrophysical simulation codes (e.g., `FLASH`, `Enzo`, `Gadget`, `Athena`), which simulate the formation and evolution of astrophysical systems using models for the relevant physics, including magnetohydrodynamics, gravity, dark matter, plasmas, etc. The simulation-specific code is contained within various "frontend" implementations, and the user-facing API to perform the analysis on the data is the same regardless of the type of simulation being analyzed. This enables the same function calls to easily generate photons from models produced by any of these simulation codes making it possible to use the `PHOX` algorithm beyond the original application to `Gadget` simulations only. In fact, most previous approaches to simulating X-ray observations were limited to datasets from particular simulation codes.

The second strength is related, in that by largely abstracting out the simulation-specific concepts of "cells", "grids", "particles", "smoothing lengths", etc., `yt` provides a window on to the data defined primarily in terms of physically motivated volumetric region objects. These include spheres, disks, rectangular regions, regions defined on particular cuts on fields, etc. Arbitrary combinations of these region types are also possible. These volumetric region objects serve as natural starting points for generating X-ray photons from not only physically relevant regions within a complex hydrodynamical simulation, but also from simple "toy" models which have been constructed from scratch, when complex, expensive simulations are not necessary.

The third major strength is that implementing our model in `yt` makes it possible to easily make use of the wide variety of useful libraries available within the scientific Python ecosystem. Our implementation uses `SciPy` for integration, `AstroPy` for handling celestial coordinate systems and FITS I/O, and `PyXspec` for generating X-ray spectral models. Tools for analyzing astrophysical X-ray data are also implemented in Python (e.g., `CIAO`'s `Sherpa` package, [Refsdal09]), enabling an easy comparison between models and observations.

Example

Here we present a workable example of creating simulated X-ray events using `yt`'s `photon_simulator` analysis module. We implemented the module in `yt` v. 3.0 as `yt.analysis_modules.photon_simulator`. `yt` v. 3.0 can be downloaded from http://yt-project.org. The example code here is available as an IPython notebook. This is not meant to be an exhaustive explanation of all of the `photon_simulator`'s features and options--for these the reader is encouraged to visit the yt documentation.

As our input dataset, we will use an `Athena` simulation of a galaxy cluster core, which can be downloaded from the `yt` website at http://yt-project.org/data/MHDSloshing.tar.gz. You will also need to download a version of `APEC` from http://www.atomdb.org. Finally, the absorption cross section table used here and the *Chandra* response files may be downloaded from http://yt-project.org/data/xray_data.tar.gz.

First, we must import the necessary modules:

```
import yt
from yt.analysis_modules.photon_simulator.api \
    import TableApecModel, ThermalPhotonModel, \
    PhotonList, TableAbsorbModel
from yt.utilities.cosmology import Cosmology
```

Next, we load the dataset `ds`, which comes from a set of simulations presented in [ZuHone14]. `Athena` datasets require a `parameters` dictionary to be supplied to provide unit conversions to Gaussian units; for most datasets generated by other simulation codes that can be read by `yt`, this is not necessary.

```
parameters={"time_unit":(1.0,"Myr"),
            "length_unit":(1.0,"Mpc"),
            "mass_unit":(1.0e14,"Msun")}

ds = yt.load("MHDSloshing/virgo_low_res.0054.vtk",
             parameters=parameters)
```

Slices through the density and temperature of the simulation dataset are shown in Figure 2. The luminosity and temperature of our model galaxy cluster roughly match that of Virgo. The photons will be created from a spherical region centered on the domain center, with a radius of 250 kpc:

```
sp = ds.sphere("c", (250., "kpc"))
```

This will serve as our `data_source` that we will use later. Now, we are ready to use the `photon_simulator` analysis module to create synthetic X-ray photons from this dataset.

Step 1: Generating the Original Photon Sample

First, we need to create the `SpectralModel` instance that will determine how the data in the grid cells will generate photons. A number of options are available, but we will use the `TableApecModel`, which allows one to use the `APEC` data tables:

```
atomdb_path = "/Users/jzuhone/Data/atomdb"

apec_model = TableApecModel(atomdb_path,
                            0.01, 10.0, 2000,
                            apec_vers="2.0.2",
                            thermal_broad=False)
```

where the first argument specifies the path to the `APEC` files, the next three specify the bounds in keV of the energy spectrum and the number of bins in the table, and the remaining arguments specify the `APEC` version to use and whether or not to apply thermal broadening to the spectral lines.

Now that we have our `SpectralModel`, we need to connect this model to a `PhotonModel` that will connect the field data in the `data_source` to the spectral model to and generate the photons which will serve as the sample distribution for observations. For thermal spectra, we have a special `PhotonModel` called `ThermalPhotonModel`:

```
thermal_model = ThermalPhotonModel(apec_model,
                                   X_H=0.75,
                                   Zmet=0.3)
```

Where we pass in the `SpectralModel`, and can optionally set values for the hydrogen mass fraction `X_H` and metallicity `Z_met`, the latter of which may be a single floating-point value or the name of the `yt` field representing the spatially-dependent metallicity.

Next, we need to specify "fiducial" values for the telescope collecting area in cm^2, exposure time in seconds, and cosmological

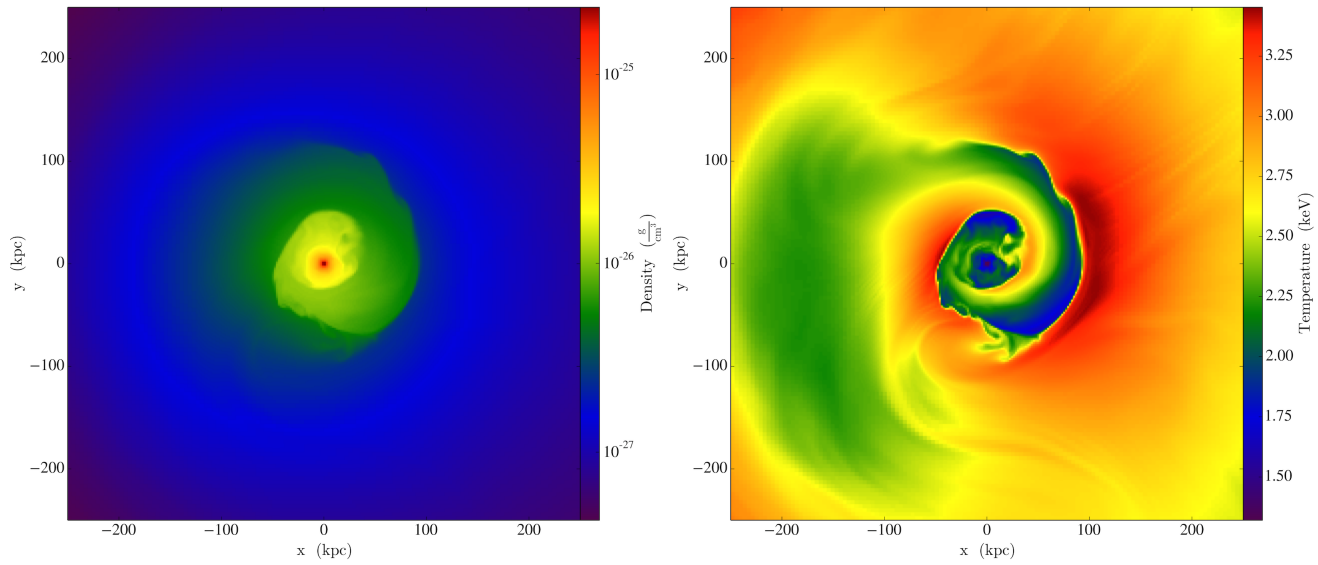

Fig. 2: *Slices of density (left) and temperature (right) of an* `Athena` *dataset of a galaxy cluster core.*

redshift, choosing generous values so that there will be a large number of photons in the Monte-Carlo sample. We also construct a `Cosmology` object, which will be used to determine the source distance from its redshift.

```
A = 6000. # must be in cm**2!
exp_time = 4.0e5 # must be in seconds!
redshift = 0.05
cosmo = Cosmology()
```

By default the `Cosmology` object uses the WMAP7 cosmological parameters from [Komatsu11], but others may be supplied, such as the [Planck13] parameters:

```
cosmo = Cosmology(hubble_constant = 0.67,
                  omega_matter = 0.32,
                  omega_lambda = 0.68)
```

Now, we finally combine everything together and create a `PhotonList` instance, which contains the photon samples:

```
photons = PhotonList.from_scratch(sp, redshift, A,
                                  exp_time,
                                  thermal_model,
                                  center="c",
                                  cosmology=cosmo)
```

where we have used all of the parameters defined above, and `center` defines the reference coordinate which will become the origin of the photon coordinates, which in this case is `"c"`, the center of the simulation domain. This object contains the positions and velocities of the originating volume elements of the photons, as well as their rest-frame energies.

Generating this Monte-Carlo sample is the most computationally intensive part of the PHOX algorithm. Once a sample has been generated it can be saved to disk and loaded as needed rather than needing to be regenerated for different observational scenarios (instruments, redshifts, etc). The photons object can be saved to disk in the HDF5 format with the following method:

```
photons.write_h5_file("my_photons.h5")
```

To load these photons at a later time, we use the `from_file` method:

```
photons = PhotonList.from_file("my_photons.h5")
```

Step 2: Projecting Photons to Create Specific Observations

At this point the photons can be projected along a line of sight to create a specific synthetic observation. First, it is necessary to set up a spectral model for the Galactic absorption cross-section, similar to the spectral model for the emitted photons set up previously. Here again, there are multiple options, but for the current example we use `TableAbsorbModel`, which allows one to use an absorption cross section vs. energy table written in HDF5 format (available in the xray_data.tar.gz file mentioned previously). This method also takes the column density `N_H` in units of 10^{22} cm^{-2} as an additional argument.

```
N_H = 0.1
a_mod = TableAbsorbModel("tbabs_table.h5", N_H)
```

We next set a line-of-sight vector `L`:

```
L = [0.0, 0.0, 1.0]
```

which corresponds to the direction within the simulation domain along which the photons will be projected. The exposure time, telescope area, and source redshift may also be optionally set to more appropriate values for a particular observation:

```
texp = 1.0e5
z = 0.07
```

If any of them are not set, those parameters will be set to the original values used when creating the `photons` object.

Finally, an `events` object is created using the line-of-sight vector, modified observation parameters, and the absorption model:

```
events = photons.project_photons(L,
                                 exp_time_new=texp,
                                 redshift_new=z,
                                 absorb_model=a_mod)
```

`project_photons` draws events uniformly from the `photons` sample, orients their positions in the coordinate frame defined by `L`, and applies the Doppler and cosmological energy shifts, and removes a number of events corresponding to the supplied Galactic absorption model.

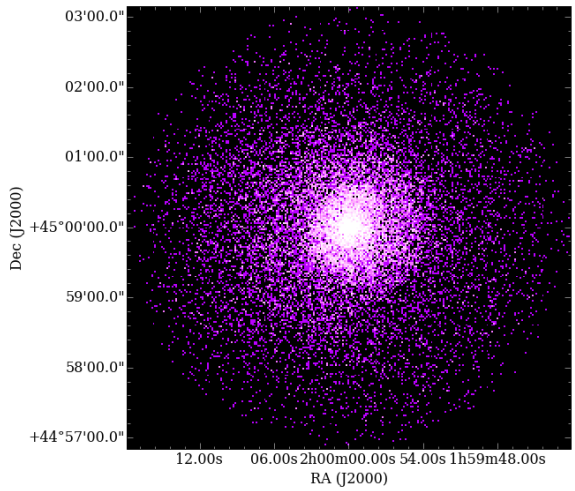

Fig. 3: *100 ks exposure of our simulated galaxy cluster, from a FITS image plotted with* `APLpy`.

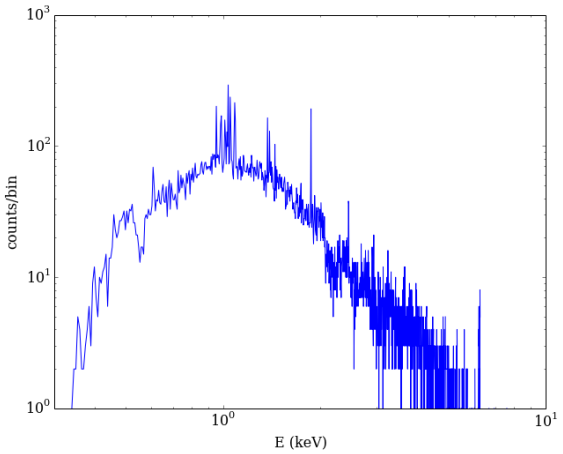

Fig. 4: *Spectral energy distribution of our simulated observation.*

Step 3: Modeling Instrumental Effects

If desired, instrumental response functions may be supplied to convolve the photons with a particular instrumental model. The files containing these functions are defined and put in a single list `resp`:

```
ARF = "chandra_ACIS-S3_onaxis_arf.fits"
RMF = "chandra_ACIS-S3_onaxis_rmf.fits"
resp = [ARF,RMF]
```

In this case, we would replace our previous call to `project_photons` with one that supplies `resp` as the `responses` argument:

```
events = photons.project_photons(L,
                                 exp_time_new=texp,
                                 redshift_new=z,
                                 absorb_model=a_mod,
                                 responses=resp)
```

Supplying instrumental responses is optional. If they are provided, `project_photons` performs 2 additional calculations. If an ARF is provided, the maximum value of the effective area curve will serve as the `area_new` parameter, and after the absorption step a number of events are further removed using the effective area curve as the acceptance/rejection criterion. If an RMF is provided, it will be convolved with the event energies to produce a new array with the resulting spectral channels.

However, if a more accurate simulation of a particular X-ray instrument is needed, or if one wishes to simulate multiple instruments, there are a couple of options for outputting our simulated events to be used by other software that performs such simulations. Since these external packages apply instrument response functions to the events list, the original `events` object generated from the `project_photons` method must not be convolved with instrument responses (e.g., the ARF and RMF) in that step. For input to `MARX`, we provide an implementation of a `MARX` "user source" at http://bitbucket.org/jzuhone/yt_marx_source, which takes as input an HDF5 file. The events list can be written in the HDF5 file format with the following method:

```
events.write_h5_file("my_events.h5")
```

Input to `SIMX` and `Sixte` is handled via `SIMPUT`, a file format designed specifically for the output of simulated X-ray data. The events list can be written in SIMPUT file format with the following method:

```
events.write_simput_file("my_events",
                         clobber=True,
                         emin=0.1, emax=10.0)
```

where `emin` and `emax` are the energy range in keV of the outputted events. Figure 5 shows several examples of the generated photons passed through various instrument simulations. `SIMX` and `MARX` produce FITS event files that are the same format as the data products of the individual telescope pipelines, so they can be analyzed by the same tools as real observations (e.g., `XSPEC`, `CIAO`).

Examining the Data

The `events` may be binned into an image and written to a FITS file:

```
events.write_fits_image("my_image.fits",
                        clobber=True,
                        emin=0.5, emax=7.0)
```

where `emin` and `emax` specify the energy range for the image. Figure 3 shows the resulting FITS image plotted using `APLpy`.

We can also create a spectral energy distribution (SED) by binning the spectrum into energy bins. The resulting SED can be saved as a FITS binary table using the `write_spectrum` method. In this example we bin up the spectrum according to the original photon energy, before it was convolved with the instrumental responses:

```
events.write_spectrum("my_spec.fits",
                      energy_bins=True,
                      emin=0.1, emax=10.0,
                      nchan=2000, clobber=True)
```

here `energy_bins` specifies whether we want to bin the events in unconvolved photon energy or convolved photon channel. Figure 4 shows the resulting spectrum.

Summary

We have developed an analysis module within the Python-based volumetric data analysis toolkit `yt` to construct synthetic X-ray observations of astrophysical sources from simulation datasets, based on the `PHOX` algorithm. This algorithm generates a large

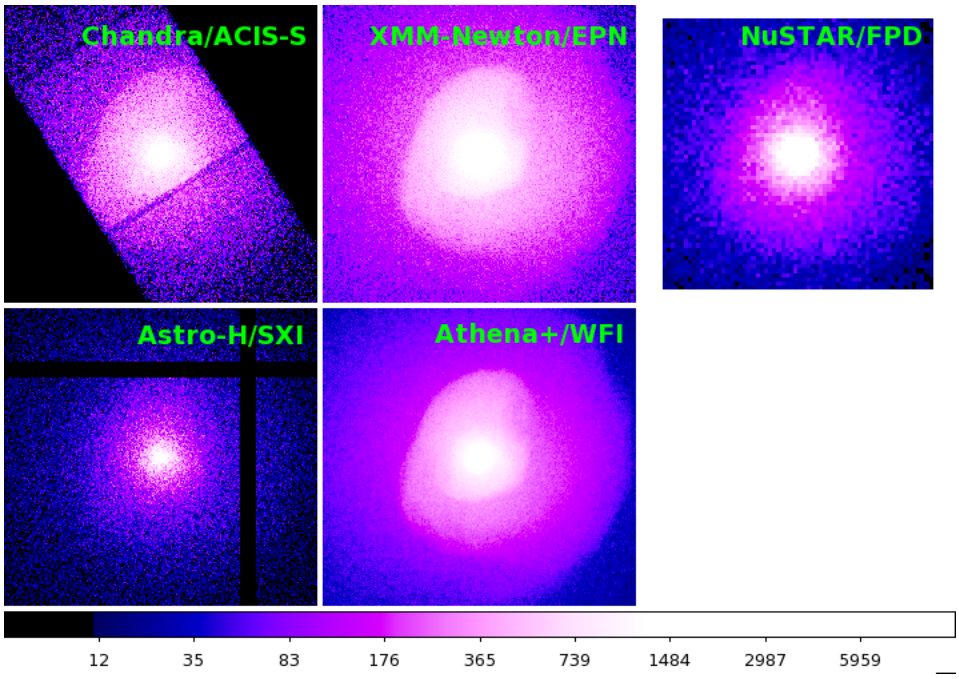

Fig. 5: *100 ks exposures of our simulated galaxy cluster, observed with several different existing and planned X-ray detectors. The Chandra image was made with `MARX`, while the others were made with `SIMX`. All images have the same angular scale.*

sample of X-ray photons in the rest frame of the source from the physical quantities of the simulation dataset, and uses these as a sample from which a smaller number of photons are drawn and projected onto the sky plane, to simulate observations with a real detector. The utility of this algorithm lies in the fact that the most expensive step, namely that of generating the photons from the source, need only be done once, and these may be used as a Monte Carlo sample from which to generate as many simulated observations along as many projections and with as many instrument models as desired.

We implement PHOX in Python, using `yt` as an interface to the underlying simulation dataset. Our implementation takes advantage of the full range of capabilities of `yt`, especially its focus on physically motivated representations of simulation data and its support for a wide variety of simulation codes as well as generic `NumPy` array data generated on-the-fly. We also benefit from the object-oriented capabilities of Python as well as the ability to interface with existing astronomical and scientific Python packages.

Our module provides a crucial link between observations of astronomical sources and the simulations designed to represent the objects that are detected via their electromagnetic radiation, enabling some of the most direct testing of these simulations. Also, it is useful as a proposer's tool, allowing observers to generate simulated observations of astrophysical systems, to precisely quantify and motivate the needs of a proposal for observing time on a particular instrument. Our software also serves as a model for how similar modules in other wavebands may be designed, particularly in its use of several important Python packages for astronomy.

REFERENCES

[Balucinska-Church92] Balucinska-Church, M., & McCammon, D. 1992, ApJ, 400, 699

[Biffi12] Biffi, V., Dolag, K., Böhringer, H., & Lemson, G. 2012, MNRAS, 420, 3545

[Biffi13] Biffi, V., Dolag, K., Böhringer, H. 2013, MNRAS, 428, 1395

[Bulbul14] Bulbul, E., Markevitch, M., Foster, A., et al. 2014, ApJ, 789, 13

[Gardini04] Gardini, A., Rasia, E., Mazzotta, P., Tormen, G., De Grandi, S., & Moscardini, L. 2004, MNRAS, 351, 505

[Heinz11] Heinz, S., Brüggen, M., & Friedman, S. 2011, ApJS, 194, 21

[Komatsu11] Komatsu, E., Smith, K. M., Dunkley, J., et al. 2011, ApJS, 192, 18

[Mewe95] Mewe, R., Kaastra, J. S., & Liedahl, D. A. 1995, Legacy, 6, 16

[Morrison83] Morrison, R. & McCammon, D. 1983, ApJ, 270, 119

[Nagai06] Nagai, D., Vikhlinin, A., & Kravtsov, A. V. 2007, ApJ, 655, 98

[Planck13] Planck Collaboration, Ade, P. A. R., Aghanim, N., et al. 2013, arXiv:1303.5076

[Raymond77] Raymond, J. C., & Smith, B. W. 1977, ApJS, 35, 419

[Refsdal09] B. Refsdal et al. *Sherpa: 1D/2D modeling and fitting in Python*. Proceedings of the 8th Python in Science conference (SciPy 2009), G Varoquaux, S van der Walt, J Millman (Eds.), pp. 51-57

[Smith01] Smith, R. K., Brickhouse, N. S., Liedahl, D. A., & Raymond, J. C. 2001, ApJL, 556, L91

[Turk11] Turk, M. J., Smith, B. D., Oishi, J. S., Skory, S., Skillman, S. W., Abel, T., & Norman, M. L. 2011, ApJS, 192, 9

[Wilms00] Wilms, J., Allen, A., & McCray, R. 2000, ApJ, 542, 914

[ZuHone09] ZuHone, J. A., Ricker, P. M., Lamb, D. Q., & Karen Yang, H.-Y. 2009, ApJ, 699, 1004

[ZuHone14] ZuHone, J. A., Kunz, M. W., Markevitch, M., Stone, J. M., & Biffi, V. 2014, arXiv:1406.4031