Pynbody: Where's the point?

Author: Matthew Prem
Date: 5/12/2025

[1] The pynbody package is a Python package created to allow for easy visualization of and interaction with the output files of an N-body program. This package does not run n-body simulations itself, instead, it allows for easy visualization of snapshots generated by the n-body simulators.

[2] I selected this package because I am rather interested in n-body simulations, having attempted (unsuccessfully, after writing a buggy kernel to dynamically resize the simulation domain) to create an interactive GPU n-body simulator while in high school. While I wanted to review a professional GPU-accelerated N-body code, I was not able to find one that was purely Python-based, with the most performant programs written in C++, C, or Fortran. There is a Python version of CUDA, so it is possible to create a GPU-accelerated n-body package purely in Python, but I wasn't able to find any on ASCL. I figured the next closest thing was a highly parallelized CPU n-body code, so I checked out the CONCEPT package, but found that it took a very long time to run, so I started looking at something more lightweight. At that point, I found Pynbody, a Python-centered package designed to visualize snapshots and other output files from more dedicated n-body code. I also learned that Pynbody had an interactive, GPU-accelerated addition called Topsy, which seemed to closely align with my original interests in creating an interactive visualizer, so I decided to do my project on it. I was originally under the impression that Pynbody could also run the simulations itself, but learned rather quickly that it is unable to do so and it is purely a visualization library. I considered switching to Pnbody, but decided I wanted to stick with the package that makes more use of the GPU, in addition to getting the impression when skimming its documentation that it was also merely a visualization tool.

[3] The version of Pynbody that I installed was 2.1.1, which at the time of writing is the most up-to-date version. The first commit to the GitHub of Pynbody was made on October 31, 2014, but Pynbody as a project had already been around for some time at that point as evidenced by its publication date on ADS being May 2013 and the copyright on its documentation covering from 2011 to 2024. I could not find information on whether Pynbody had come out of a previous package, but that doesn't mean it wasn't influenced by previous visualization packages written in other languages. Since Pynbody is still in active development, it doesn't have a clear successor, but it does have the Topsy toy project for interactive visualization. Another popular N-body visualization package, Pnbody appears to have been developed concurrently with Pynbody, as it was originally published in February 2013, just a few months before Pynbody.

[4] Pynbody is still maintained by its original author, Andrew Pontzen, with new commits to the package's GitHub page continuing. As of the time of writing, the most recent commit was made on April 14, 2025, by Andrew Pontzen. If you want to contribute to this package, you can follow the instructions detailed on Pynbody's PyPI page. The two main ways of contributing to the package are to write a tutorial on how to use a specific feature or submit your code to the relevant sub-module and submit a pull request after creating a local form of the Pynbody repository.

[5] The package itself was rather easy to install, with a pip install successfully setting everything up for the main package. Pynbody also has a very thorough series of tutorials which are quite helpful for getting started and providing "boilerplate" code which is easier to modify than typing everything yourself. It should be noted that Pynbody is a Linux-only software, though Windows systems can use it by utilizing the Linux Subsystem

for Windows (WSL). Topsy, on the other hand, was really difficult to work with, though I doubt that is the package's fault, as several things were more difficult than they should be since I had to do everything using WSL. The main complaint that Topsy had was that my graphics cards didn't support the "FLOAT32_FILTERABLE" extension, even though my graphics card (Nvidia RTX 4070 mobile) has the necessary hardware to support it. I was able to get Topsy to work after reinstalling my graphics drivers, but when I came back to the project after a several-day break for other finals, I found that it had decided to stop working even after reinstalling graphics drivers.

[6] Pynbody does install easily with the command `pip install pynbody` once you have Jupyter Lab up and running within a Linux distribution. I found that when using a Jupyter notebook, the command `!pip install pynbody` would always throw an error, but `%pip install pynbody` would succeed. Since this is different behavior than the Jupyter lab installation I have natively on my Windows computer, I suspect it has something to do with there being a link between my overarching Windows OS and WSL, which interact in ways I don't yet have an intuitive grasp of. If you want to locally install the documentation, tutorials, or look through the code for yourself, you can execute the following code cell `!git clone https://github.com/pynbody/pynbody.git`. Installing Topsy is a similar procedure involving `%pip install topsy` and optionally `!git clone https://github.com/pynbody/pynbody.git`, but I found that I needed to reinstall my graphics drivers after to make sure it actually works.
Since I was doing this project in WSL, I needed to run several other commands in order to install Jupyter Lab on WSL. The first step for this was to set up a WSL installation using the PowerShell command `wsl --install`. Once a Linux distribution is installed (the default is Ubuntu), I navigated to the root of its file system, opened another PowerShell instance at this file location, and typed the command `bash` to open a bash command line interface within the WSL installation. I found that if I attempted to use PowerShell from here on out, it would do something, but as soon as I tried to open the installed software, it would fail to open properly. Once in bash, the following commands were run:
`sudo apt-get update`
`sudo apt install python3-pip`
`sudo apt install pipx`
`pipx install jupyterlab`
`pipx ensurepath`
This ensured that an updated version of Python was installed on my WSL distribution and that Jupyter Lab was also installed. I'm not sure if using pipx is considered best practice in this situation, but a regular pip install complained about being in an externally managed environment, and this was the only way I was able to get it to work. I then closed and reopened bash and typed the following commands:
`apt install jupyter-core`
`jupyter-lab --allow-root`
A few seconds after the `jupyter-lab` command was run, a link to a localhost URL was displayed. Once this URL was pasted into a browser (running on the Windows side of my computer), it brought up a Jupyter Lab editor where the notebook in this repository is able to run.

[7] Yes, the source code is available. You can access the source code for Pynbody online at https://github.com/pynbody/pynbody and the source code for Topsy online at https://github.com/pynbody/topsy", or you can download the source code using the `git clone` commands as detailed in part 5.

[8] Yes, Pynbody is indeed used by other packages. One example is the EDGE project for simulating the smallest galaxies in the universe. There are also several packages that reference Pynbody, which were

developed for uses that Pynbody doesn't quite fit. A couple of examples include swiftsimio, which found Pynbody too slow for their application, and Plonk, which made an n-body visualization code more suited to astrophysical n-body simulations rather than the cosmological n-body simulations Pynbody was designed with in mind. I was unable to find any packages that used Topsy.

[9] Pynbody is a Python package, meaning that it is designed to be used either in a Python source file (a .py file) or an implementation of Python such as Jupyter. For this project, I used a Jupyter notebook to run Pynbody. Topsy is designed to be used through the command line, for example by typing this command into bash: `topsy testdata/gasoline_ahf/g15784.lr.01024.gz`. It is possible to use Topsy in Python or a Jupyter notebook in one of two ways:
1: Import the subprocess package and use that to build up a command that will then be executed by bash. If you wanted to render just the gas particles of that same data set with this method, that can be achieved using the following code:

```
import subprocess
subprocess.run(["topsy","testdata/gasoline_ahf/g15784.lr.01024.gz", "-p","gas"]);
```

2: Use the Jupyter-rbf widget to render it within a Jupyter notebook. An equivalent statement to the example of the first method is as follows:

```
import topsy
topsy.load("testdata/gasoline_ahf/g15784.lr.01024.gz", particle="gas",hdr=False);
```

Unfortunately, there are often errors creating the interactive window. When testing this code on my machine, it created the window the very first time I ran this code, but never again. The command line methods will likely be more stable as they create a separate window and don't have to rely on widgets to create a proper rendering context.

[10] Assuming starting from a fresh install and no test data, the following code can be executed to create a figure of the gas in a galaxy in a cosmological simulation:

```
import numpy as np
#download the test data
import pynbody.test_utils
pynbody.test_utils.precache_test_data()

import pynbody
import pylab
s = pynbody.load('testdata/gasoline_ahf/g15784.lr.01024.gz')
print(f"There are {len(s)} particles in this simulation")

#Find halos
h = s.halos()

#Get the halo of the largest galaxy in this cluster
main_halo = h[0]

import matplotlib.pyplot as plt
#create an image
s.physical_units() #turn simulation space into kpc or Mpc
t = pynbody.analysis.center(main_halo) #center on the largest galaxy
image_values = pynbody.plot.image(main_halo.gas, width=100,height=100,cmap="inferno")
#plot the image
plt.title("Halo 1 Galaxy Gas") #give it a title
plt.savefig("Galactic_gas.png") #save it
```

[11] Yes, Pynbody is able to produce figures without any additional installs. Matplotlib is a dependency for Pynbody, which can be installed by pip during the installation of Pynbody; however, while there are plotting functions specific to Pynbody, they ultimately go through Matplotlib. This is actually helpful when it comes to saving figures, as otherwise Pynbody doesn't have a dedicated way to save the figures it produces.

[12] When the code in part [10] is executed, it results in the following figure:

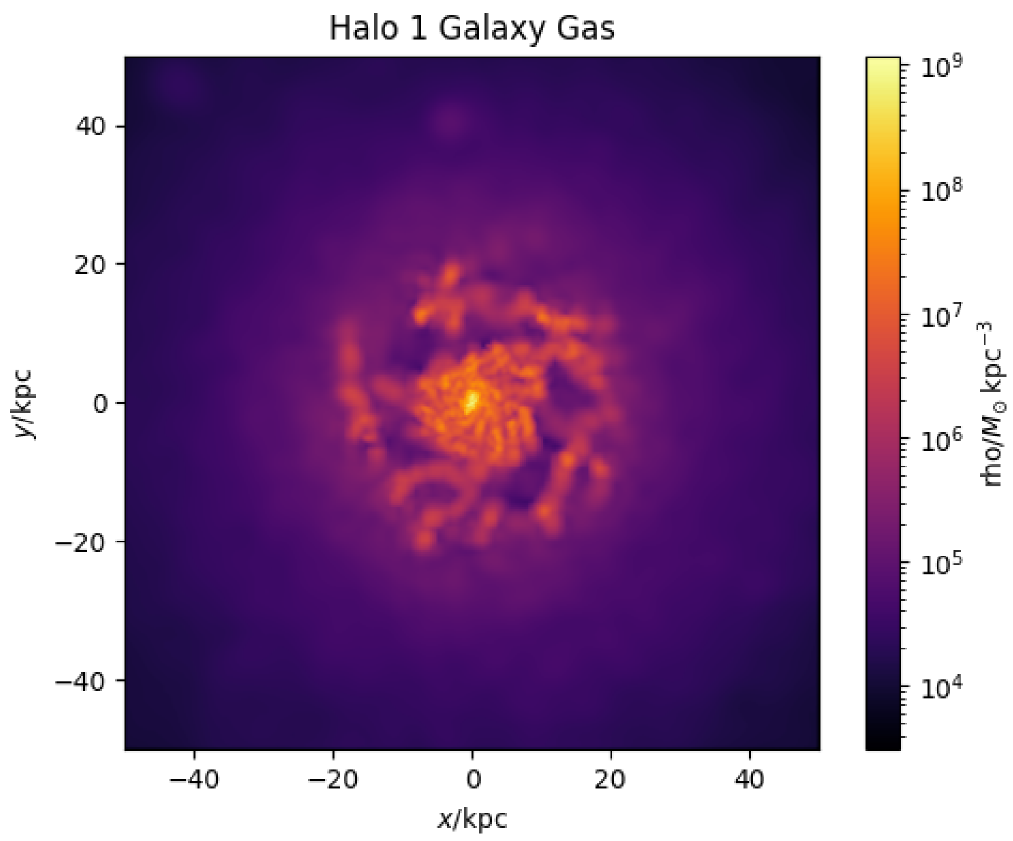

Figure 1: Visualization of the gas density in solar masses per cubic kiloparsec for the largest halo (usually corresponds to the largest galaxy) in the simulation. The galaxy is being viewed from the +z axis.

[13] Pynbody is made primarily in Python, but it does have several components made in C++, particularly in the portions of the code where it implements kdtrees. To combine the C++ and Python files, Cython is used.

[14] The input to Pynbody is a snapshot of an n-body or hydrodynamic astrophysical simulation, such as those produced by PKDGRAV/Gasoline, Gadget, Gadget4/Arepo, N-Chilada, and RAMSES AMR. Along with the raw input data, you can also tell Pynbody how to transform the plot to get the desired viewing angle. For example, you can tell Pynbody to center the plot on the largest halo, and that you want to rotate the coordinate system 10 degrees. For a visual example of the rotations, check out the .mp4 files located in this GitHub, which show the view axes centered on this galaxy being rotated around all three principal axes. These videos were made by putting many snapshots at different angles into the rendering program Blender. For the x and y rotations, portions of the galaxy are clipped by the near and far clipping planes. Additionally, if you have a simulation that contains dark matter particles, gas particles, and star particles, you can tell Pynbody to only render a specific type of particle. Topsy also has arguments to position the view of the loaded snapshot on a particular halo and display a particular type of point, but has many more display settings in the interactive visualizer.

[15] Pynbody can output plots showing visualizations of the spatial positions of the particles (as in the images we have seen so far) as well as profiles of the following properties of halos: density, enclosed mass, circular orbit speed, and surface brightness. Pynbody can also automatically calculate and plot the derivatives of these quantities. In addition to the plotting, Pynbody has the ability to output data on several derived quantities of halos, including their mass, radius, and number of sub-halos, among other quantities. Topsy, as a purely visual tool, can only output screenshots and screen recordings.

[16] The only tests provided by Pynbody are following the tutorials and seeing if they line up with the plots present in the tutorial images. The image in this report is one of these tutorial images under a different colormap. The reference image can be found in the "Making some images" section of the Pynbody Quickstart guide.

[17] To feel confident that the code produces a reliable result, you can go through the tutorials and make sure the images you get match what is in the tutorials. In my case, they did. I also attempted to recreate the same galaxy image with Topsy and saved the result as "Topsy test.png". The results qualitatively appear to match quite well, though the image produced by Pynbody appears somewhat smoother.

[18] Based on the requirements installed by pip during the installation of Pynbody, Pynbody depends on the following Packages: cython, h5py, matplotlib, numpy, scipy, posix-ipc, contourpy, cycler, fonttools, kiwisolver, packaging, pillow, pyparsing, python-dateutil, and six.
Topsy, as a package that uses the GPU, has a longer requirements list: numpy, pynbody, matplotlib, pillow, wgpu, jupyter_rfb, tqdm, opencv-python, Pyside6, superqt, rendercanvas, cffi, sniffio, pycparser, ipywidgets, jupyterlab-widgets, and comm, among others.

[19] The Pynbody documentation can be found at https://pynbody.readthedocs.io/latest/index.html and is rather extensive, more than sufficient for my needs with this project. Topsy has essentially no documentation other than the installation guide in the Readme of its GitHub page. If you are lucky, this will be more than sufficient to get everything installed and running, but if you encounter any errors, especially GPU errors, you are essentially on your own to find a solution.

[20] Yes, the preferred method of citation is to cite its Astrophysics Source Code Library entry using the following BibTex: `@misc{pynbody,`
```
author = {{Pontzen}, A. and {Ro{\v s}kar}, R. and {Stinson}, G.~S. and {Woods},
R. and {Reed}, D.~M. and {Coles}, J. and {Quinn}, T.~R.},
title = "{pynbody: Astrophysics Simulation Analysis for Python}",
note = {Astrophysics Source Code Library, ascl:1305.002},
year = 2013
}
```

[21] I'd like to provide links to some of the technology enabling these visualizations, including Cython for making the Python code of Pynbody run at an acceptable speed and WGPU for making Topsy possible. WGPU, in particular, I find interesting because they are featuring a real-time n-body simulation run on the GPU. WGPU is also based on the WebGPU API, which means it's not that far-fetched to imagine running some of these interactive tools in the browser, which should hopefully increase compatibility or accessibility.

[22] While there are 279 papers that have cited Pynbody, two examples include:

- Synthetic Clones of the Most Distant Galaxies in the Universe: https://ui.adsabs.harvard.edu/abs/2025arXiv250507935K/abstract

- Signatures of Massive Black Hole Merger Host Galaxies from Cosmological Simulations. II. Unique Stellar Kinematics in Integral Field Unit Spectroscopy: https://ui.adsabs.harvard.edu/abs/2024ApJ...977..265B/abstract

[23] For this particular package, I had to learn new Python methods or at least new ways of setting up Python that were not covered in class. Specifically, the use of WSL made this project far more complicated than it would otherwise have been. In terms of Python packages, I also had to learn a little about the subprocess package so I could execute a bash command from a Jupyter notebook. My biggest issue, however, was trying to get my GPU to behave with Topsy. Since this is a highly system and environment-dependent issue, I doubt any level of Python knowledge below that offered by a dedicated Python GPU programming class would have been sufficient.

[24] These packages are all new to me. While I have had some previous experience using Linux and the command line, it wasn't in this specific context. I also worked independently, though I would like to thank Peter Teuben for keeping up with the issues I was having and fielding my questions about the project.