Pynbody: Where's the point?

Author: Matthew Prem
Date: 5/12/2025

[1] The pynbody package is a python packaged creaed to allow for easy visualization of and interaction with the output files of a n-body program. This packages doesn not run n-body simulations itself, just allows for easy visualization of snapshots generated by the n-body simulators.

[2] I selected this package because I am rather intersted in n-body simulations, having attempted (unsuccessfuly, after writing a buggy kernel to dynaically resize the simulation domain) to create an interactive GPU n-body simulator while in high school. While I wanted to review a professional GPU acceerated N-body code, I wasn't able to find any 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 definitally 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 parallized 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 purely a visualization library. I considered switching to Pnbody, but decided I wanted to stick with the package which 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 intructions 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 relavent 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 seting everything up for the main package. Pynbody also has a very thurough series of tutorials which are really helpful for getting started and providing "boilerplate" code which is easier to modifiy 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). There is no way to use Pynbody with a Mac except for a virtual machine. 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" extention, 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 easilly 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 neede 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, open another powershell instance at this file location, and type 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 completely. 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 jupyterlab 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 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 Jupyter notebook for running 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 use that to build up a command that will then be execulted by bash. If you wanted to render just the gas particles of that same data set with this method, that can be achived 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`