Plotting large datasets

Colin Leach, 2020

Galaxy simulations, even undergraduate projects, can involve about 10^7 particles. Tough to plot, so what are some options?

I made these notes for my own use but there may (?) be other students who are interested.

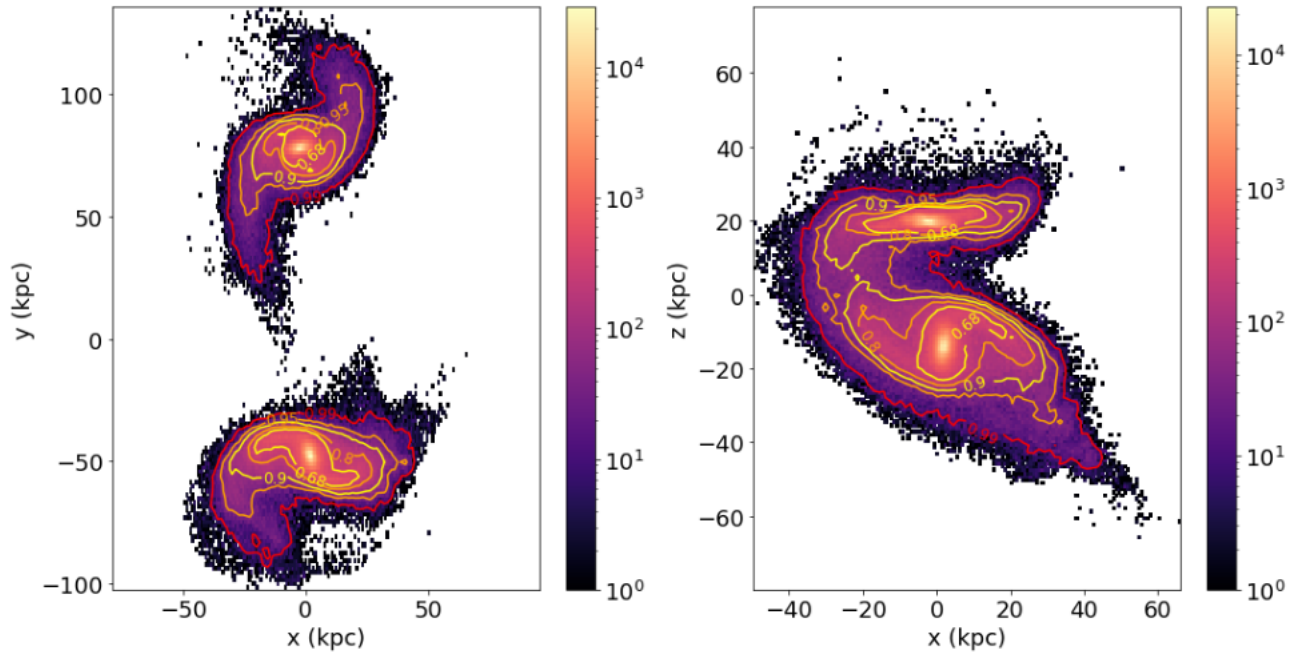
The dataset used is an array with shape $(3, 975000)$, so 975,000 sets of (x, y, z) coordinates, at a time when two galaxies are tidally disrupting one another.

1 Static plots

1.1 Matplotlib hist2d

Covered in Lab 7 of ASTR400B, which also adds density contours. That only dealt with single disks but extending it to collisions is relatively simple. Think centering, orientation and (perhaps hardest!) what x and y limits to set.

The result may look something like this:



1.2 Matplotlib plus mpl_scatter_density

This isn't a new problem, so someone wrote an extension to matplotlib to make density plots easier.

Website: `https://github.com/astrofrog/mpl-scatter-density`

Borrowing heavily from their examples, this is a first attempt:

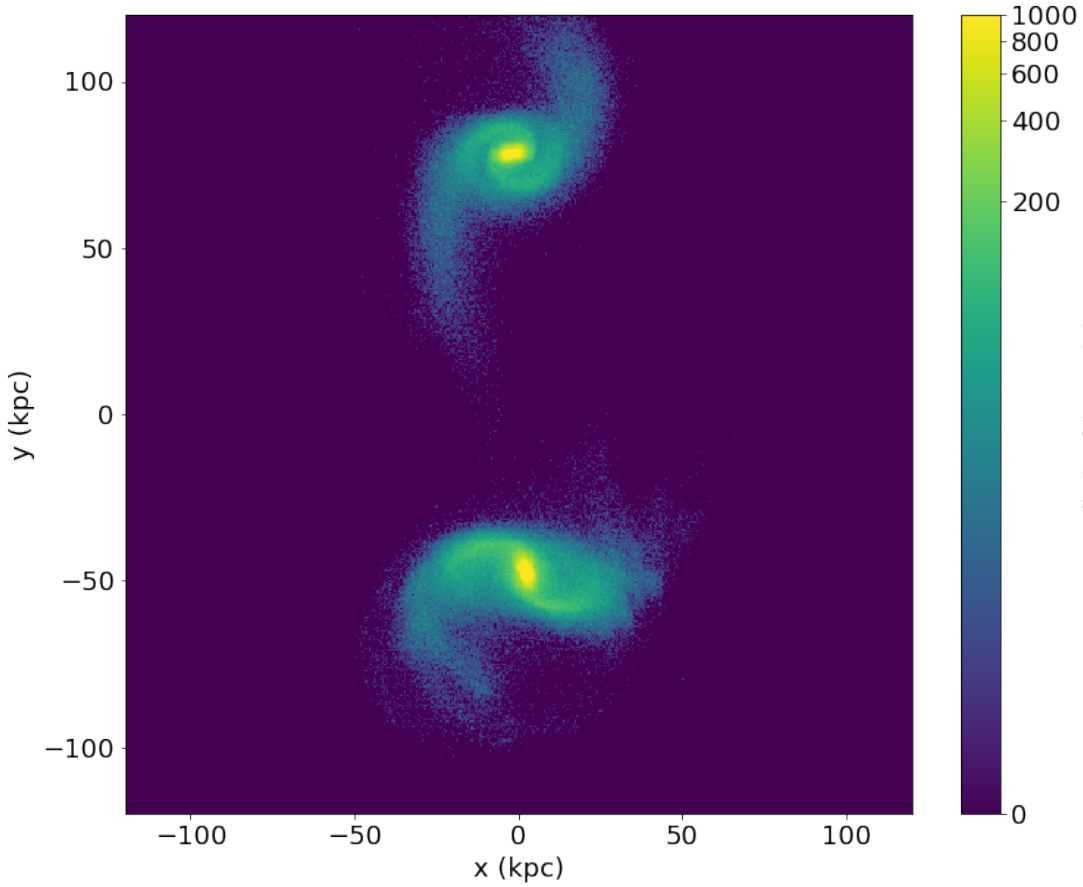
```
# install with 'conda install mpl-scatter-density'
import mpl_scatter_density

# Make the norm object to define the image stretch
from astropy.visualization import LogStretch
from astropy.visualization.mpl_normalize import ImageNormalize
norm = ImageNormalize(vmin=0., vmax=1000, stretch=LogStretch())

# Make the plot in the (almost) usual pyplot way
# Note the projection to invoke mpl_scatter_density
#  and the norm to use LogStretch

fig = plt.figure(figsize=(12,10))
ax = fig.add_subplot(1, 1, 1, projection='scatter_density')
density = ax.scatter_density(disks[0], disks[1], norm=norm)
plt.xlim(-120, 120)
plt.ylim(-120, 120)

# Add axis labels (standard pyplot)
fontsize = 18
plt.xlabel('x (kpc)', fontsize=fontsize)
plt.ylabel('y (kpc)', fontsize=fontsize)
fig.colorbar(density, label='Number of stars per pixel')
```



The colorbar needs some work, but we now get pixel-level resolution. It's a bit easier to see the tidal tails, at least on screen (printouts can be disappointing).

2 Interactive plots

Getting away from matplotlib, there are some newer[1] options which use the power of modern graphics cards to handle large numbers of points. More powerful, steeper learning curve.

2.1 datashader

Websites: `https://github.com/holoviz/datashader` and `https://datashader.org`

This needs the data to be arrange columnwise in a (single) pandas dataframe; fairly typical for modern plotting packages. Used in isolation, datashader will make a 2D image. To add axes, labels, etc, it needs to be hooked into a suitable browser-based package such as Bokeh (usually) or Plotly (under development). This can mean a LOT of extra packages to install from conda or pip and lots of potential conflicts[2]

2.2 ipyvolume

Works natively with 3D data. Only in the browser as it relies on WebGL rendering.

Websites: `https://github.com/maartenbreddels/ipyvolume` and `https://ipyvolume.readthedocs.io/en/latest/`

Spectacular when the developer (Maarten Breddels, a Dutch astronomer) demonstrates it at conferences, not so easy for the rest of us.

2.3 All things OpenGL

Now we're at the heavy end of the options. With pyqt5 + (pyopengl or vispy), an experienced graphics programmer with unlimited time can do virtually anything.

I played with this a bit over spring break. At the current rate of progress, I may have a crude prototype working by the end of the Spring 2021 semester (but no promises).

[1] In other words, typically less finished and poorly documented

[2] The release of Bokeh 2.0 broke datashader 0.10.0. For now, use Bokeh 1.4.0 and hope datashader 0.13 is released soon. The developer says he's working on it while in self-isolation after flying home from Covid-hit Spain.