HW#6: Getting Started with GIZMO

In this problem set you will get up to speed with the public, multi-physics parallel astrophysics code GIZMO. You can checkout the code here: `https://bitbucket.org/phopkins/gizmo-public/src/master/`). The documentation is here: `http://www.tapir.caltech.edu/~phopkins/Site/GIZMO_files/gizmo_documentation.html`.

1. **Hello GIZMO :)**

Familiarize yourself with the GIZMO documentation and relevant code method papers (e.g., Hopkins 2015, Hopkins & Raives 2016). Follow the documentation to download and compile GIZMO on Stampede2 (see §6 and §12 "General Super-Computing Questions" in the documentation. I have created a summary document for getting GIZMO running on Stampede2 (but see the TACC and GIZMO documentation for complete details.)

2. **Testing, testing, 1, 2, 3...**

The fastest way to get up to speed with a code is to run already existing problem setups, i.e., test problems. It is also generally a good idea to run test problems to make sure the code is behaving as expected. Run the following test problems and make plots of the results (see documentation, the initial condition files are located here: `http://www.tapir.caltech.edu/~phopkins/sims/`). If the problem has a given analytic or expected numerical result, comment on how the code performs. Record how many processors you used for the test.

For each test, make a plot of the gas density versus position at a few different times (see notes on reading and plotting snapshots below).

a) **Sod Shock Tube.** See left panels in Fig 10 in Hopkins 2015. Compare against the expected analytic solution (the predicted solution is available as a text file). Discuss any features you see in the result.

Phil provides two different initial configuration files – one in which the cells have fixed mass and one in which the cells have fixed volume (you can check this for yourself by opening the ics hdf5 files and inspecting the 'Masses' and 'Coordinates' fields). Note that you can run this problem in either the meshless finite mass (MFM) or meshless finite volume (MFV) mode (as set in the GIZMO Config.sh file). GIZMO MFM will run the fixed-volume ics file .... but it doesn't look as nice.

b) **Optional: MFM vs. MFV.** Run the Sod shock tube in both MFM and MFV modes and compare the two solutions. Which do you think is better and why, i.e., how do the artifacts in the solution differ?

c) **But what about magnetic fields?** Choose and run *one* of the available tests that includes MHD. Make a plot of the resulting magnetic field, e.g., show either the field in one

direction such as 'magnetic_field_x' or a proxy for the rms magnetic field (such as ('gas', 'magnetic_energy'), which is available in *yt*).

d) **Optional: More complexity.** Choose and run one of the following physics tests based upon your research interests: Evrard Test, Keplerian Disk, Santa Barbara Cluster or Galactic Disk without ISM physics.

e) **Problems and Comments.** Did you encounter any problems? If so, how did you solve them? Do you have any questions that were not answered by the documentation?

3. **Sod it! Not again!**

Using *Make_IC.py* in the GIZMO `scripts` directory as a guide (or another of the setup options) generate initial conditions for the Sod shock tube problem we did in HW2. You may either use GIZMO's MFM solver – in which case make the cell mass uniform in your setup – or the MFV solver — make the cell volume uniform (constant dx spacing).

Run the problem with GIZMO and compare with your result from HW2.[1]

Troubleshooting:
• In your *.params file, don't forget to change the name of the initial condition file 'InitCondFile', 'TimeMax' and 'BoxSize.' You may also want to change the 'TimeBetSnapshot'.
• If using *Make_IC.py* make sure you don't add any dust ('PartType3') to your IC file.
• Cell masses should adopt the 1D mass. • You may want to include an initial array of 'SmoothingLength' in your initial conditions.
• To debug problems in your initial problem setup, it is useful to inspect the HDF5 data attributes and plot your initial conditions (and compare them to the Phil's Sod setup HDF5 files).
• To debug problems, it is useful to compare your problem run-time output to the run-time output from Phil's setup. Note that the fiducial problem setup from Phil has some warnings; if these appear in your setup, too, it shouldn't be a problem.

Notes about reading and visualizing GIZMO data

Regarding visualization and analysis, I myself use *yt* (`https://yt-project.org`), a python-based data analysis and visualization package, to plot GIZMO snapshots. *yt* is fairly well documented and has an extensive cookbook of examples. It is powerful (it does volume renderings and streamlines!) but it can be fiddly. Another simpler Python-based package is *Meshoid* (see `https://github.com/mikegrudic/meshoid`). Snapshots can be read by a

[1] If your HW2 implementation was not fully working, you can use my code – see solutions in `https://github.com/soffner/ComputationalAstrophysics`. Note that because of the periodic boundary condition you may need to add 'filler' like the example shock tube, since your solution with a periodic boundary will have 2 shock fronts.

variety of other programs, too, so if you are already familiar with one of these (e.g., TOPCAT, Visit) you may want to use that to make your plots[2].

It is also possible to directly read the data from the HDF5 file and plot it as you would any arrays. For example, the Sod problem is a fundamentally 1D problem and it is run in Gizmo's 1D mode (`BOX_SPATIAL_DIMENSION=1`). In this case, it is straightforward to read and plot it with matplotlib.[3] In Python a snipit that reads the hdf5 file directly looks like:

```
 import h5py
# Open the file
f = h5py.File('snapshot_xxx.hdf5', 'r')

# List the names of the available data arrays
partdata = f['PartType0']
for name in partdata:
print(name)

# Grab the data
den = f['PartType0']['Density'][:]  # Returns a ndarray
x = f['PartType0']['Coordinates'][:]
```

In the case of fully 3D data, it is possible to use 2D histograms to make plots of the density distribution.

If you choose to use *yt* note that to set the units you must specify the unit base when you load the file like so:

```
 # Set units
unit_base = 'UnitMagneticField_in_gauss':  1.0,
'UnitLength_in_cm' :  1.0,
'UnitMass_in_g' :  1.0,
'UnitVelocity_in_cm_per_s' :  1.0
# Load the dataset
ds = yt.load("output/snapshot_010.hdf5", unit_base=unit_base)
```

The unit variables must be the same as they appear in the *.params file of the problem setup.

[2]Note the GIZMO HDF5 file format is the same one used by the Gadget code. The name 'PartType0', the datatype of the main gas variables, is inherited from that code.

[3]I did not have any luck getting yt or Visit to make a 1D line plot of this data.