testing

December 22, 2021

[11]:
```
import numpy as np
import yt
from yt.data_objects.level_sets.api import *

fname = "Gadget2.0090"
#unit_base = {"UnitLength_in_cm": 3.08568e21, "UnitMass_in_g": 1.989e43,␣
 ↪"UnitVelocity_in_cm_per_s": 100000,}
bbox_lim = 70 # pc
bbox = [[-bbox_lim, bbox_lim], [-bbox_lim, bbox_lim], [-bbox_lim, bbox_lim]]

#ds = yt.load(fname, unit_base=unit_base, bounding_box=bbox)

ds = yt.load(fname, bounding_box=bbox)
ds.index
sorted(ds.field_list)

print(ds.field_list)

obj = ds.arbitrary_grid([-10, -10, -10], [10, 10, 10], dims=[128, 128, 128])

print(obj)


#density = obj["deposit", "all_density"]
#center position : ([-2.13004994,  1.08833396,  0.87476391])
#print(obj["deposit", "all_density"])
#print(obj["deposit", "all_mass"])


'''
for i in range(obj['Gas', 'Coordinates'].size):
  print(
        "(%f,  %f,  %f)"
        % (
            obj['Gas', 'Coordinates'][i][0],
            obj['Gas', 'Coordinates'][i][1],
            obj['Gas', 'Coordinates'][i][2],
```

```
            #obj["deposit", "all_density"][i],
        )
  )
'''

'''
for i in range(obj["gas", "density"].size):
  print(
            i,
            obj["gas", "x"][i],
            obj["gas", "y"][i],
            obj["gas", "z"][i],
            obj["gas", "density"][i],
        )

'''
'''
Rho_min = obj["deposit", "all_density"].min()
Rho_max = obj["deposit", "all_density"].max()
Mass_min = obj["deposit", "all_mass"].min()
Mass_max = obj["deposit", "all_mass"].max()
print(Rho_min)
print(Rho_max)
print(Mass_min)
print(Mass_max)
'''

#obj.index
#sorted(obj.field_list)

ds2 = obj.save_as_dataset(fields=[("deposit", "all_density"), ("deposit",␣
↪"all_mass")])

#ds2 = obj.save_as_dataset(fields=[("deposit", "all_density"), ("deposit",␣
↪"all_mass"), ("gas", "density")])
print(ds2)


################################################################################
# sphere_ds = yt.load(ds2)
# sphere_ds.index
# sorted(sphere_ds.field_list)

# print(sphere_ds.data["grid", "all_density"])
# print(sphere_ds.data["gas", "density"])
# print(sphere_ds.data["deposit", "all_density"])
```

```
# data_source = sphere_ds.disk([0.5, 0.5, 0.5], [0.0, 0.0, 1.0], (8, "kpc"),␣
↪(1, "kpc"))

# master_clump = Clump(data_source, ("grid", "all_density"))
# master_clump.add_validator("min_cells", 20)

# master_clump.add_info_item("center_of_mass")

# c_min = data_source["grid", "all_density"].min()
# c_max = data_source["grid", "all_density"].max()

# print(c_min)
# print(c_max)
# step = 2.0
# find_clumps(master_clump, c_min, c_max, step)
```

```
yt : [INFO     ] 2021-12-22 11:37:53,274 Omega Lambda is 0.0, so we are turning
off Cosmology.
yt : [INFO     ] 2021-12-22 11:37:53,277 Assuming length units are in kpc
(physical)
yt : [INFO     ] 2021-12-22 11:37:53,331 Parameters: current_time              =
10.68
yt : [INFO     ] 2021-12-22 11:37:53,332 Parameters: domain_dimensions         =
[1 1 1]
yt : [INFO     ] 2021-12-22 11:37:53,334 Parameters: domain_left_edge          =
[-70. -70. -70.]
yt : [INFO     ] 2021-12-22 11:37:53,337 Parameters: domain_right_edge         =
[70. 70. 70.]
yt : [INFO     ] 2021-12-22 11:37:53,339 Parameters: cosmological_simulation   =
0
yt : [INFO     ] 2021-12-22 11:37:53,352 Allocating for 2.198e+06 particles
Initializing coarse index : 100%|        | 9/9 [00:00<00:00, 45.00it/s]
yt : [INFO     ] 2021-12-22 11:37:53,558 Updating index_order2 from 2 to 2
Initializing refined index: 100%|        | 9/9 [00:01<00:00,  4.80it/s]
yt : [INFO     ] 2021-12-22 11:37:55,458 Allocating for 2.198e+06 particles
Initializing coarse index : 100%|        | 9/9 [00:00<00:00, 41.92it/s]
yt : [INFO     ] 2021-12-22 11:37:55,683 Updating index_order2 from 2 to 2
Initializing refined index: 100%|        | 9/9 [00:01<00:00,  4.90it/s]
```

```
[('Gas', 'Coordinates'), ('Gas', 'Density'), ('Gas', 'InternalEnergy'), ('Gas',
'Mass'), ('Gas', 'ParticleIDs'), ('Gas', 'SmoothingLength'), ('Gas',
'Velocities'), ('Stars', 'Coordinates'), ('Stars', 'Mass'), ('Stars',
'ParticleIDs'), ('Stars', 'Velocities'), ('all', 'Coordinates'), ('all',
'Mass'), ('all', 'ParticleIDs'), ('all', 'Velocities'), ('nbody',
'Coordinates'), ('nbody', 'Mass'), ('nbody', 'ParticleIDs'), ('nbody',
'Velocities')]
YTArbitraryGrid (Gadget2): , left_edge=[-3.08567758e+22 -3.08567758e+22
-3.08567758e+22] cm cm, right_edge=[3.08567758e+22 3.08567758e+22
```

```
3.08567758e+22] cm cm, ActiveDimensions=[128 128 128]

yt : [INFO     ] 2021-12-22 11:38:00,143 Saving field data to yt dataset:
Gadget2_arbitrary_grid.h5.

Gadget2_arbitrary_grid.h5
```

[17]:
```
import yt

fname2 = 'Gadget2_arbitrary_grid.h5'
sphere_ds = yt.load(fname2)
#sphere_ds.index
#sorted(sphere_ds.field_list)

'''
for i in range(obj["deposit", "all_density"].size):
  print(
            i,
            obj["index", "x"][i],
            obj["index", "y"][i],
            obj["index", "z"][i],
            obj["deposit", "all_density"][i],
        )
'''

rho_max = obj["deposit", "all_density"].max()
rho_min = obj["deposit", "all_density"].min()
print(rho_max)
print(rho_min)
```

```
yt : [INFO     ] 2021-12-22 11:44:48,494 Parameters: current_time              =
10.68 code_time
yt : [INFO     ] 2021-12-22 11:44:48,496 Parameters: domain_dimensions         =
[128 128 128]
yt : [INFO     ] 2021-12-22 11:44:48,497 Parameters: domain_left_edge          =
[-10. -10. -10.] code_length
yt : [INFO     ] 2021-12-22 11:44:48,498 Parameters: domain_right_edge         =
[10. 10. 10.] code_length
yt : [INFO     ] 2021-12-22 11:44:48,500 Parameters: cosmological_simulation   =
0

1.1768608839697469e-15 g/cm**3
0.0 g/cm**3
```

[20]:
```
import numpy as np
import yt
from yt.data_objects.level_sets.api import *
```

```
fname2 = 'Gadget2_arbitrary_grid.h5'
ds = yt.load(fname2)

#sphere_ds.index
#sorted(sphere_ds.field_list)
#print(sphere_ds.data["grid", "all_density"])
#print(sphere_ds.data["gas", "density"])
#print(sphere_ds.data["deposit", "all_density"])

data_source = ds.disk([0.5, 0.5, 0.5], [0.0, 0.0, 1.0], (8, "pc"), (1, "pc"))
#master_clump = Clump(data_source, ("grid", "all_density"))
#master_clump = Clump(data_source, ("gas", "cell_density"))

master_clump = Clump(data_source, ("grid", "all_density"))

master_clump.add_validator("min_cells", 20)
master_clump.add_info_item("center_of_mass")

c_min = data_source["grid", "all_density"].min()
c_max = data_source["grid", "all_density"].max()

print(c_min)
print(c_max)
step = 2.0
find_clumps(master_clump, c_min, c_max, step)
```

```
yt : [INFO     ] 2021-12-22 11:55:52,040 Parameters: current_time              =
10.68 code_time
yt : [INFO     ] 2021-12-22 11:55:52,041 Parameters: domain_dimensions         =
[128 128 128]
yt : [INFO     ] 2021-12-22 11:55:52,043 Parameters: domain_left_edge          =
[-10. -10. -10.] code_length
yt : [INFO     ] 2021-12-22 11:55:52,044 Parameters: domain_right_edge         =
[10. 10. 10.] code_length
yt : [INFO     ] 2021-12-22 11:55:52,046 Parameters: cosmological_simulation   =
0
```

```
---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
/tmp/ipykernel_4421/577330062.py in <module>
     17 #master_clump = Clump(data_source, ("gas", "cell_density"))
     18
---> 19 master_clump = Clump(data_source, ("grid", "all_density"))
     20
     21 master_clump.add_validator("min_cells", 20)
```

```
~/anaconda3/envs/yt/lib/python3.10/site-packages/yt/data_objects/level_sets/
 ↪clump_handling.py in __init__(self, data, field, parent, clump_info,␣
 ↪validators, base, contour_key, contour_id)
     50         self.parent = parent
     51         self.quantities = data.quantities
---> 52         self.min_val = self.data[field].min()
     53         self.max_val = self.data[field].max()
     54         self.info = {}

~/anaconda3/envs/yt/lib/python3.10/site-packages/numpy/core/_methods.py in␣
 ↪_amin(a, axis, out, keepdims, initial, where)
     42 def _amin(a, axis=None, out=None, keepdims=False,
     43           initial=_NoValue, where=True):
---> 44     return umr_minimum(a, axis, None, out, keepdims, initial, where)
     45
     46 def _sum(a, axis=None, dtype=None, out=None, keepdims=False,

~/anaconda3/envs/yt/lib/python3.10/site-packages/unyt/array.py in␣
 ↪__array_ufunc__(self, ufunc, method, *inputs, **kwargs)
   1672                 inp = inp.in_units("radian").v
   1673             # evaluate the ufunc
-> 1674             out_arr = func(np.asarray(inp), out=out_func, **kwargs)
   1675             if ufunc in (multiply, divide) and method == "reduce":
   1676                 # a reduction of a multiply or divide corresponds to

ValueError: zero-size array to reduction operation minimum which has no identit
```

[ ]:

[ ]: