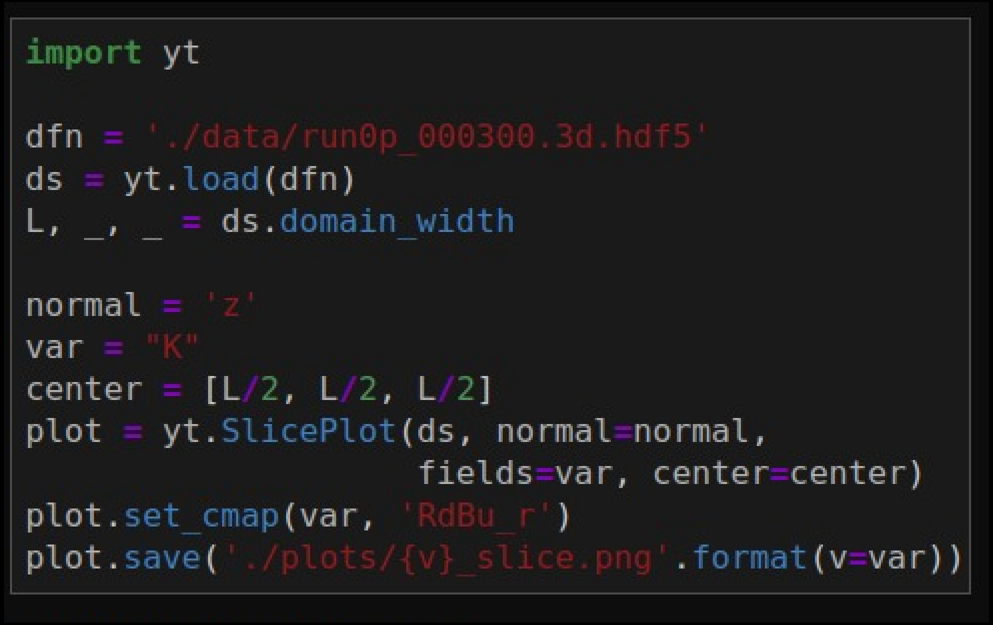
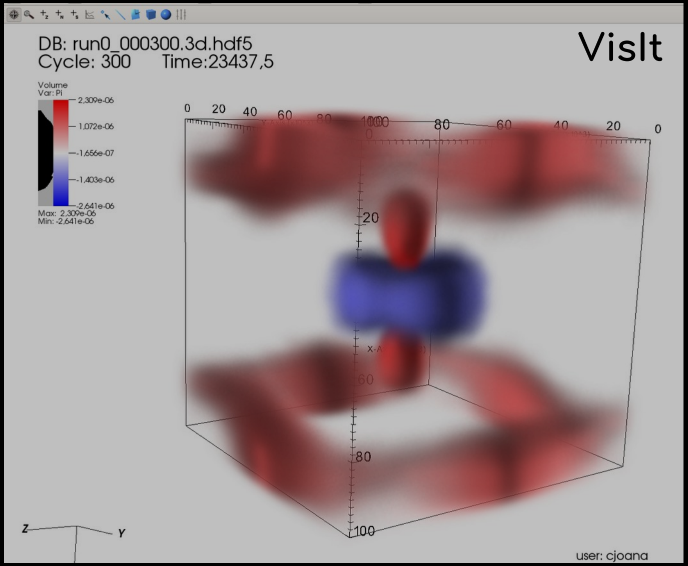
Visualitsation tools for GRChombo



```
import yt

dfn = './data/run0p_000300.3d.hdf5'
ds = yt.load(dfn)
L, _, _ = ds.domain_width

normal = 'z'
var = "K"
center = [L/2, L/2, L/2]
plot = yt.SlicePlot(ds, normal=normal,
                    fields=var, center=center)
plot.set_cmap(var, 'RdBu_r')
plot.save('./plots/{v}_slice.png'.format(v=var))
```



Cristian Joana - UCLouvain (CURL)

GRChombo Meeting 30/03/2022

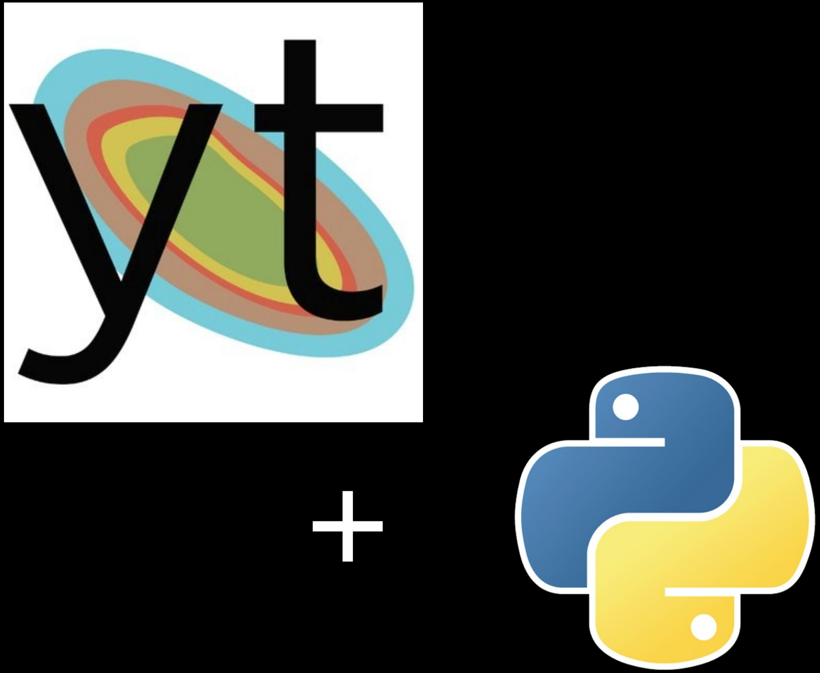
Visualitsation tools for GRChombo

- Using YT (python)
- Using Visit (GUI)

For movies → Josu slides 2019

- Installation
- Utilities
- Examples

YT - project



https://yt-project.org/docs/dev/ (documentation)

https://yt-project.org/community.html (mailing lsit, Slack, etc.)

YT - Installation

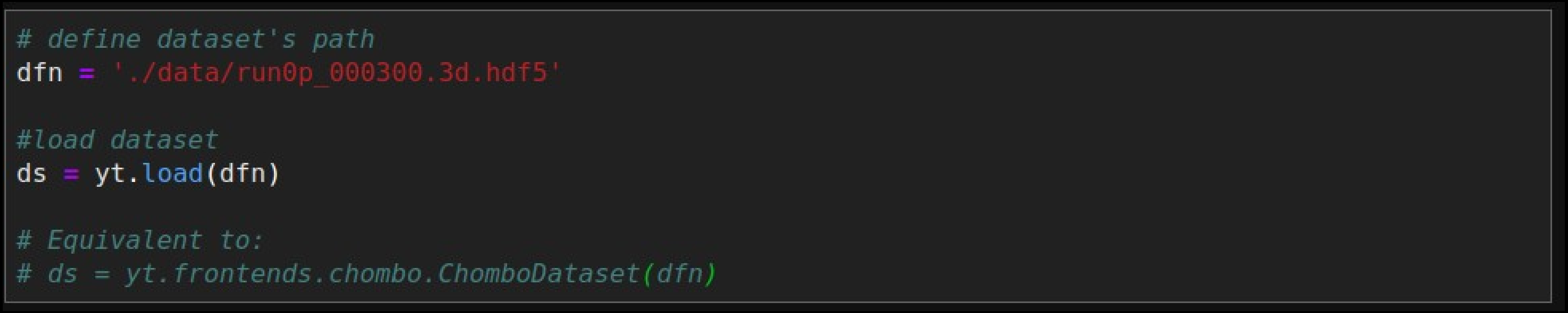
- Via conda:
```
$ conda install -c conda-forge yt
```

- Via PiP:
```
$ pip install yt
```

- Via github repository:
```
$ git clone https://github.com/yt-project/yt
$ cd yt && python setup.py install
```

YT – Loading files

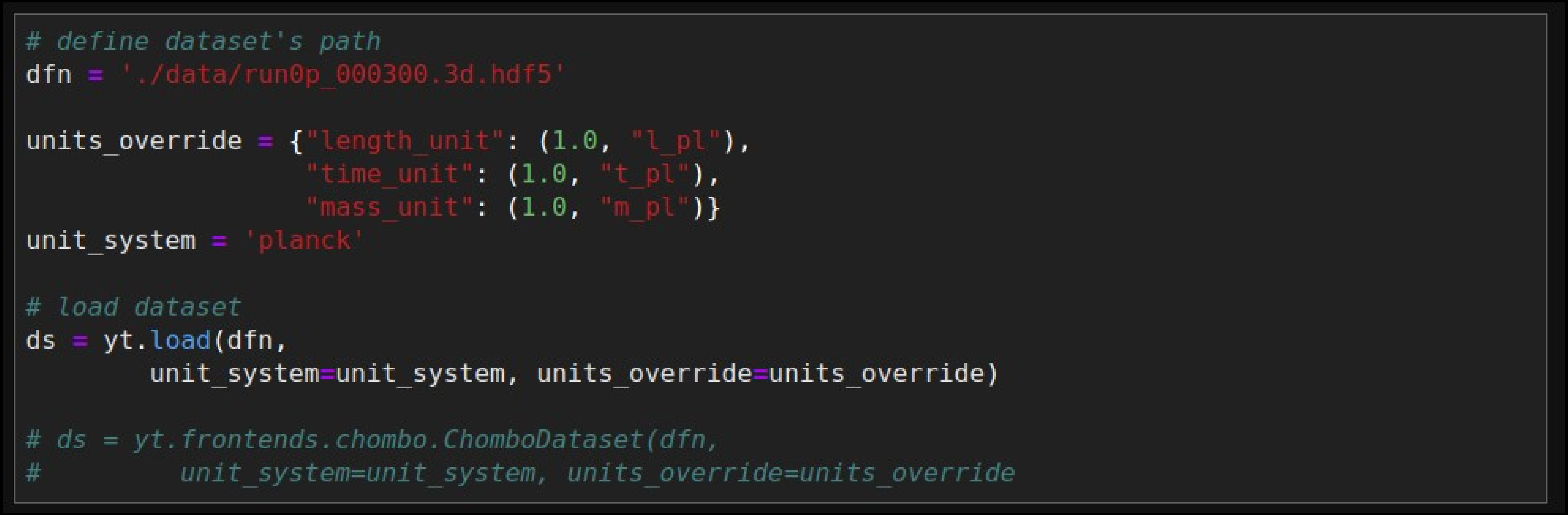
Loading hdf5 file



```
# define dataset's path
dfn = './data/run0p_000300.3d.hdf5'

#load dataset
ds = yt.load(dfn)

# Equivalent to:
# ds = yt.frontends.chombo.ChomboDataset(dfn)
```

YT – Loading files

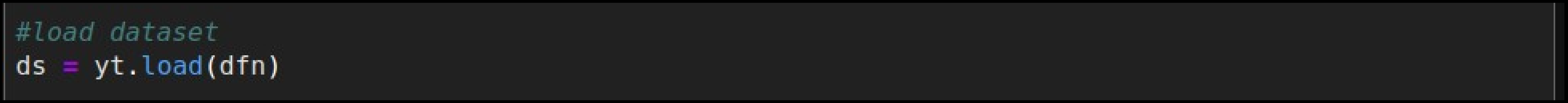
Loading hdf5 file



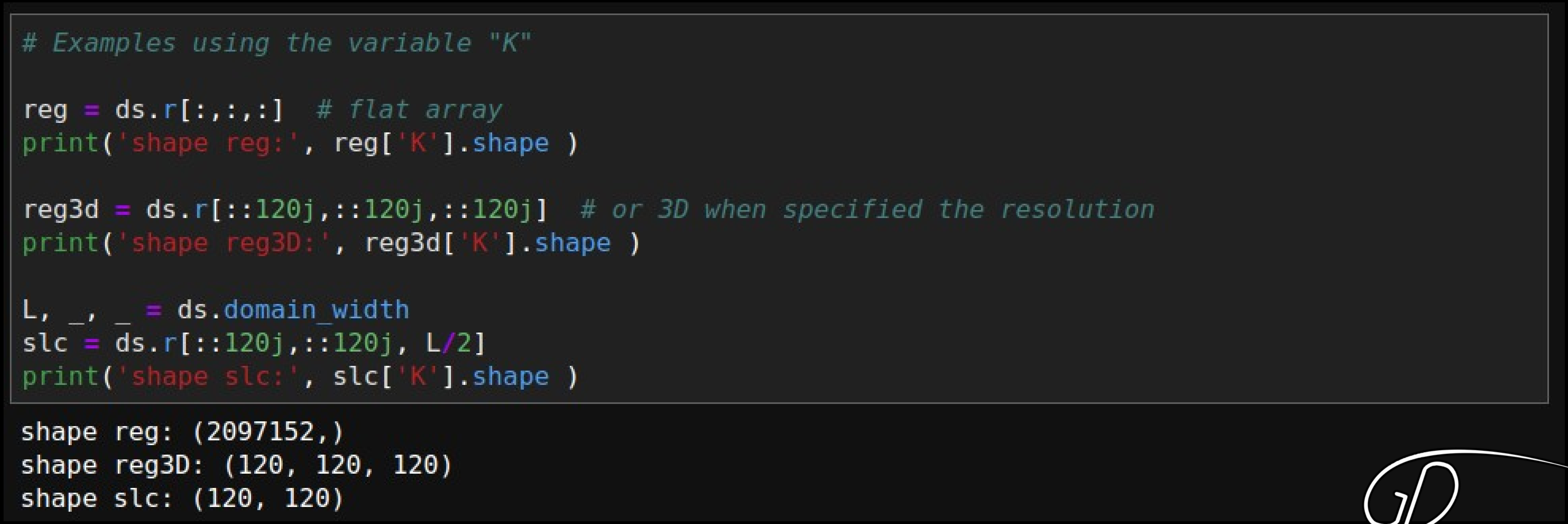
```python
# define dataset's path
dfn = './data/run0p_000300.3d.hdf5'

units_override = {"length_unit": (1.0, "l_pl"),
                  "time_unit": (1.0, "t_pl"),
                  "mass_unit": (1.0, "m_pl")}
unit_system = 'planck'

# load dataset
ds = yt.load(dfn,
        unit_system=unit_system, units_override=units_override)

# ds = yt.frontends.chombo.ChomboDataset(dfn,
#          unit_system=unit_system, units_override=units_override
```

YT – Handeling data

Loading data variables



```
#load dataset
ds = yt.load(dfn)
```

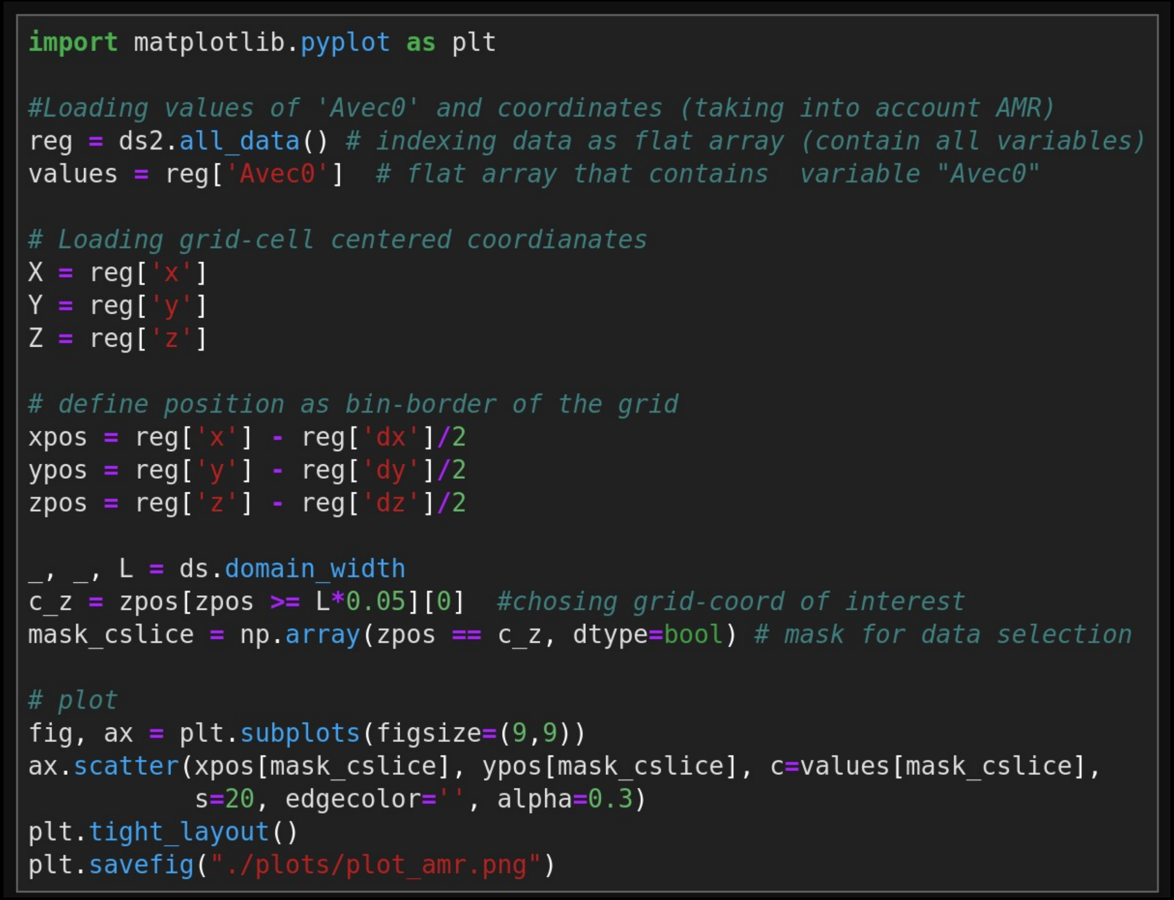


```
# Examples using the variable "K"

reg = ds.r[:,:,:]  # flat array
print('shape reg:', reg['K'].shape )

reg3d = ds.r[::120j,::120j,::120j]  # or 3D when specified the resolution
print('shape reg3D:', reg3d['K'].shape )

L, _, _ = ds.domain_width
slc = ds.r[::120j,::120j, L/2]
print('shape slc:', slc['K'].shape )
```

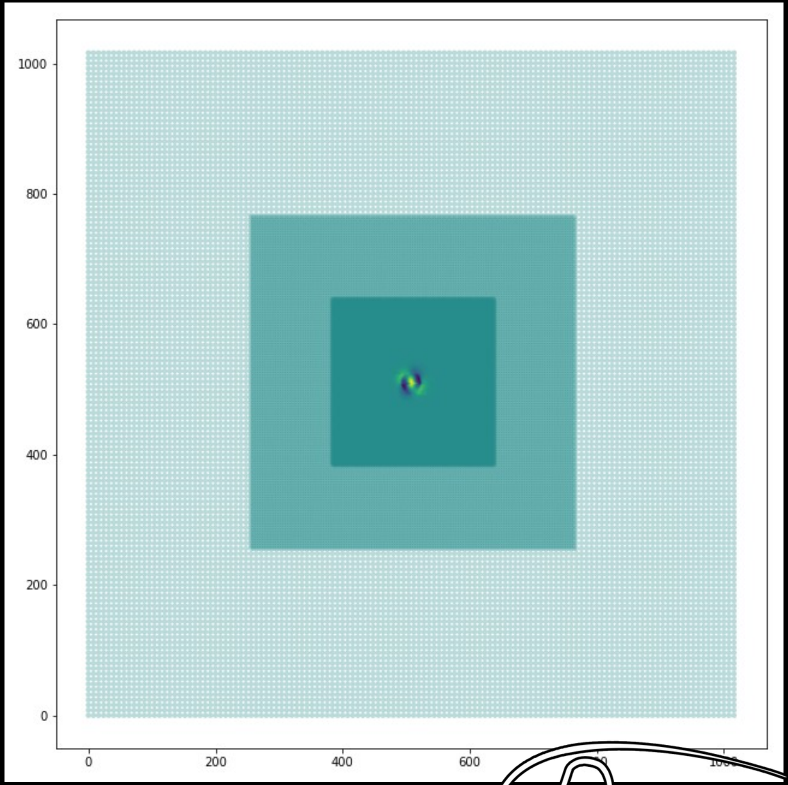
```
shape reg: (2097152,)
shape reg3D: (120, 120, 120)
shape slc: (120, 120)
```

YT – Handeling data

Example: extraction of data & AMR cordinates

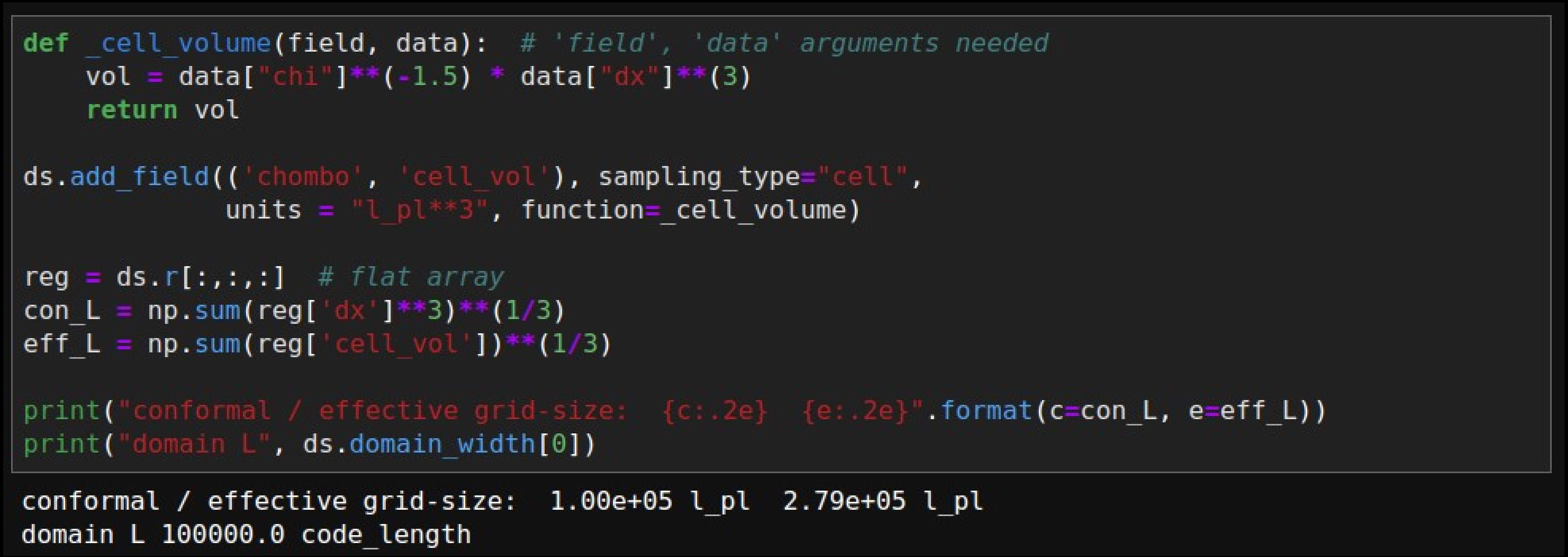


```
import matplotlib.pyplot as plt

#Loading values of 'Avec0' and coordinates (taking into account AMR)
reg = ds2.all_data() # indexing data as flat array (contain all variables)
values = reg['Avec0']  # flat array that contains  variable "Avec0"

# Loading grid-cell centered coordianates
X = reg['x']
Y = reg['y']
Z = reg['z']

# define position as bin-border of the grid
xpos = reg['x'] - reg['dx']/2
ypos = reg['y'] - reg['dy']/2
zpos = reg['z'] - reg['dz']/2

_, _, L = ds.domain_width
c_z = zpos[zpos >= L*0.05][0]  #chosing grid-coord of interest
mask_cslice = np.array(zpos == c_z, dtype=bool) # mask for data selection

# plot
fig, ax = plt.subplots(figsize=(9,9))
ax.scatter(xpos[mask_cslice], ypos[mask_cslice], c=values[mask_cslice],
           s=20, edgecolor='', alpha=0.3)
plt.tight_layout()
plt.savefig("./plots/plot_amr.png")
```



YT – Handeling data

Setting up derived variables



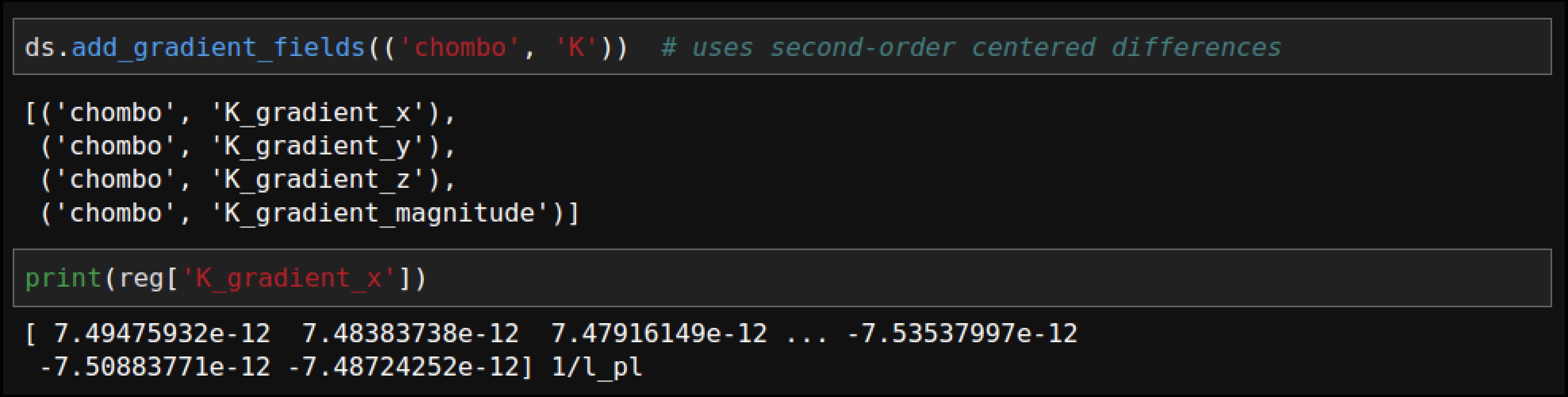
```
def _cell_volume(field, data):  # 'field', 'data' arguments needed
    vol = data["chi"]**(-1.5) * data["dx"]**(3)
    return vol

ds.add_field(('chombo', 'cell_vol'), sampling_type="cell",
             units = "l_pl**3", function=_cell_volume)

reg = ds.r[:,:,:]  # flat array
con_L = np.sum(reg['dx']**3)**(1/3)
eff_L = np.sum(reg['cell_vol'])**(1/3)

print("conformal / effective grid-size:  {c:.2e}  {e:.2e}".format(c=con_L, e=eff_L))
print("domain L", ds.domain_width[0])
```

```
conformal / effective grid-size:  1.00e+05 l_pl  2.79e+05 l_pl
domain L 100000.0 code_length
```

NB: `dx`, `dy`,.. & `x`, `y`... are automatically yt-generated grid variables.

YT – Handeling data

Setting new fields : gradients



```
ds.add_gradient_fields(('chombo', 'K'))  # uses second-order centered differences
```

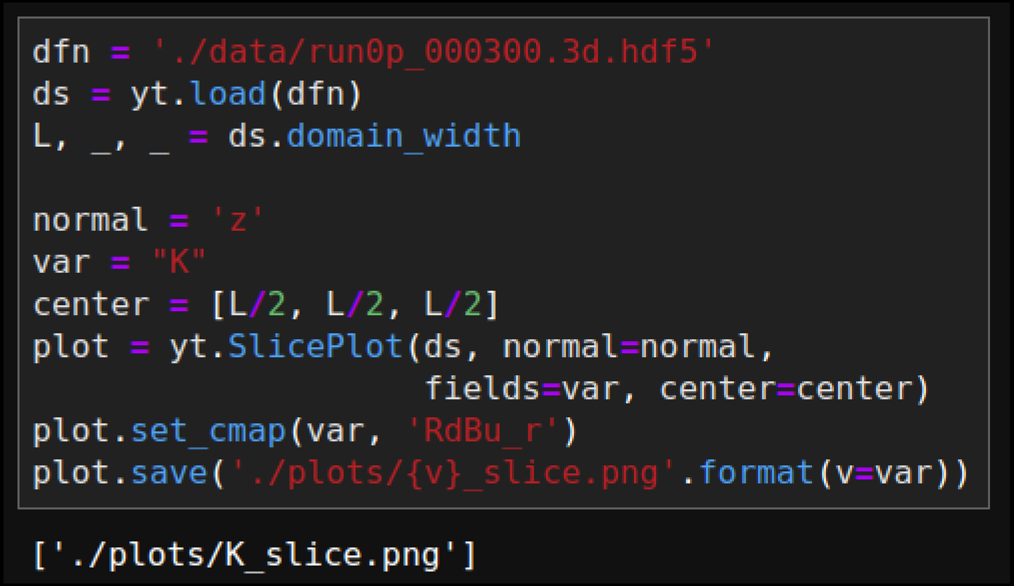
```
[('chombo', 'K_gradient_x'),
 ('chombo', 'K_gradient_y'),
 ('chombo', 'K_gradient_z'),
 ('chombo', 'K_gradient_magnitude')]
```

```
print(reg['K_gradient_x'])
```

```
[ 7.49475932e-12  7.48383738e-12  7.47916149e-12 ... -7.53537997e-12
 -7.50883771e-12 -7.48724252e-12] 1/l_pl
```

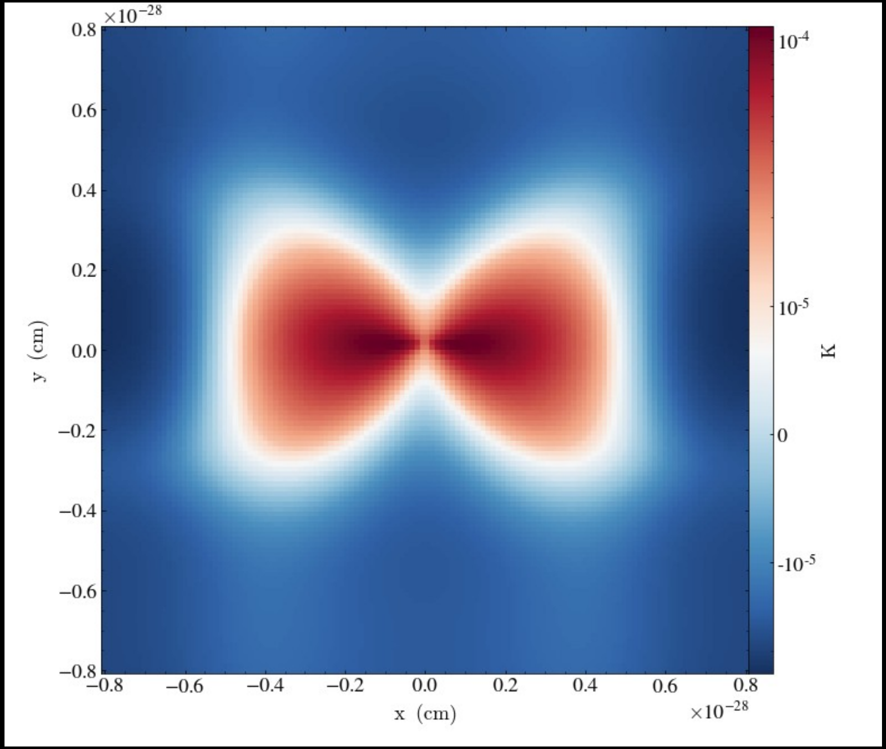
YT – Plotting utilites

yt.SlicePlot()



```
dfn = './data/run0p_000300.3d.hdf5'
ds = yt.load(dfn)
L, _, _ = ds.domain_width

normal = 'z'
var = "K"
center = [L/2, L/2, L/2]
plot = yt.SlicePlot(ds, normal=normal,
                    fields=var, center=center)
plot.set_cmap(var, 'RdBu_r')
plot.save('./plots/{v}_slice.png'.format(v=var))
```

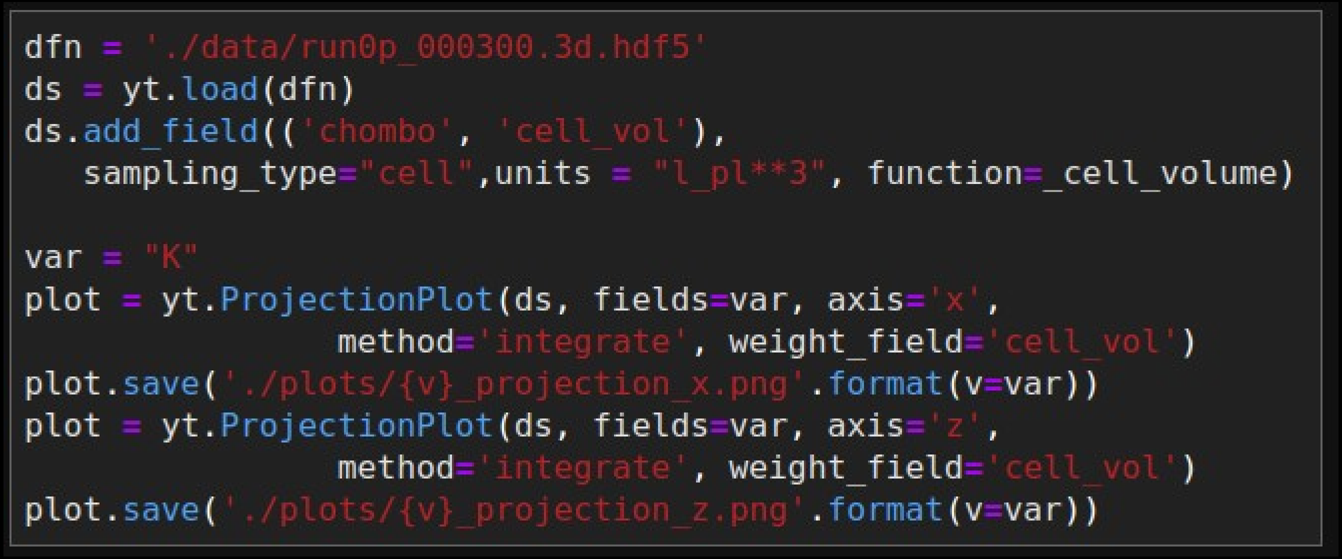
```
['./plots/K_slice.png']
```



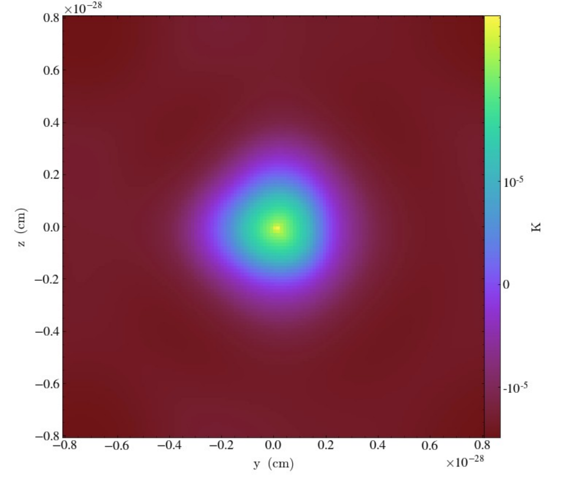
→ https://yt-project.org/doc/visualizing/plots.html

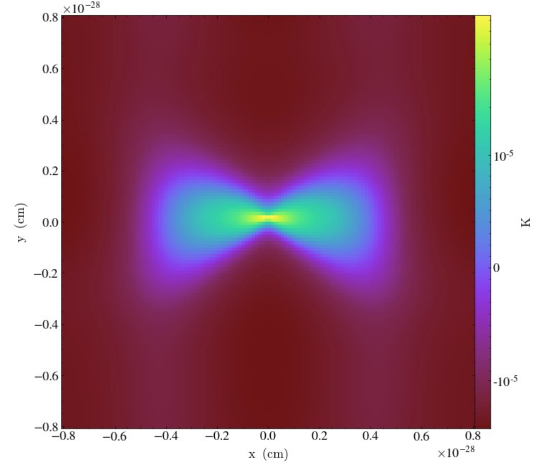
YT – Plotting utilites

yt.ProjectionPlot()



```
dfn = './data/run0p_000300.3d.hdf5'
ds = yt.load(dfn)
ds.add_field(('chombo', 'cell_vol'),
   sampling_type="cell",units = "l_pl**3", function=_cell_volume)

var = "K"
plot = yt.ProjectionPlot(ds, fields=var, axis='x',
                method='integrate', weight_field='cell_vol')
plot.save('./plots/{v}_projection_x.png'.format(v=var))
plot = yt.ProjectionPlot(ds, fields=var, axis='z',
                method='integrate', weight_field='cell_vol')
plot.save('./plots/{v}_projection_z.png'.format(v=var))
```

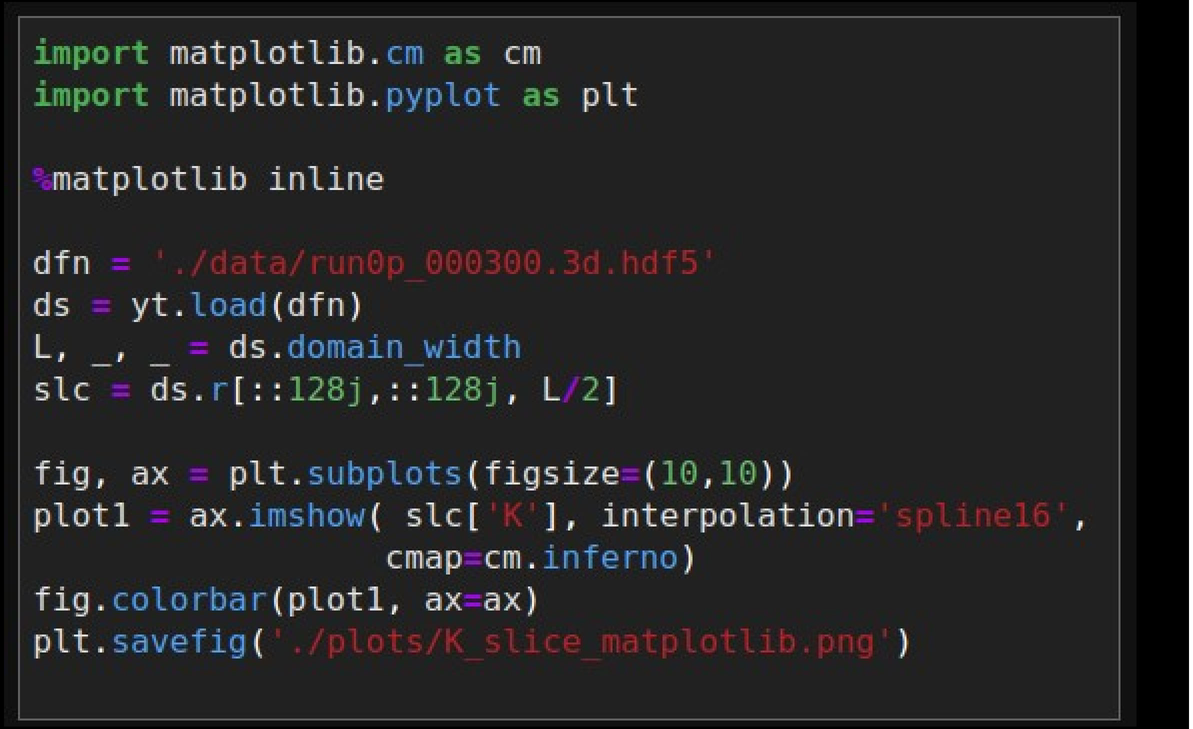




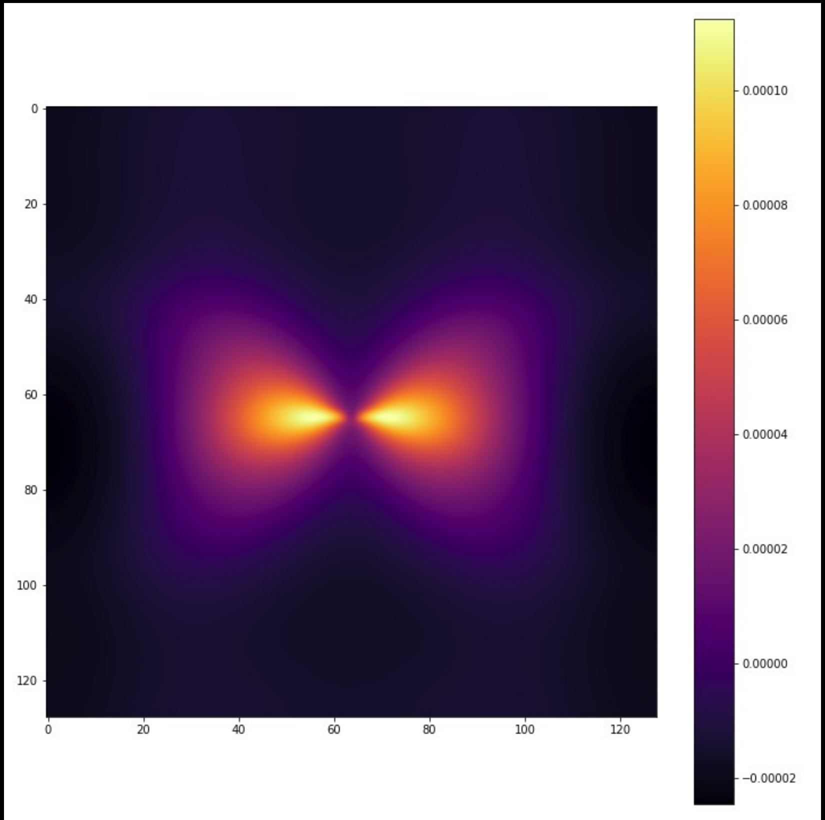
→ https://yt-project.org/doc/visualizing/plots.html

(non) YT – Plotting

Via matplotlib



```
import matplotlib.cm as cm
import matplotlib.pyplot as plt

%matplotlib inline

dfn = './data/run0p_000300.3d.hdf5'
ds = yt.load(dfn)
L, _, _ = ds.domain_width
slc = ds.r[::128j,::128j, L/2]

fig, ax = plt.subplots(figsize=(10,10))
plot1 = ax.imshow( slc['K'], interpolation='spline16',
                  cmap=cm.inferno)
fig.colorbar(plot1, ax=ax)
plt.savefig('./plots/K_slice_matplotlib.png')
```



Visit

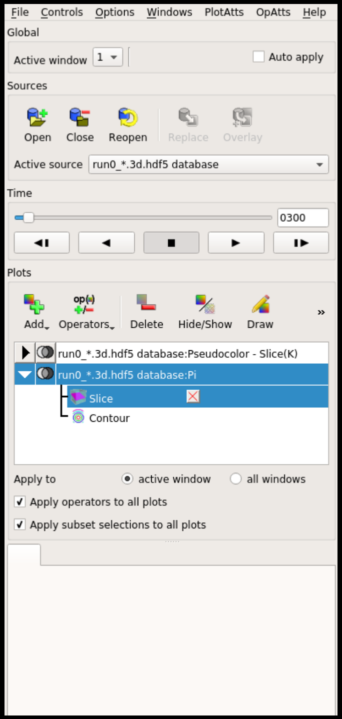
VisIt 2.6 Patch 2

(c) 2000-2015 LLNS. All Rights Reserved.

VisIt 2.6.2, svn revision Unknown

March 2013

Processing config file...



Active window 1 ☐ Auto apply

Active source run0_*.3d.hdf5 database

Time: 0300

run0_*.3d.hdf5 database:Pseudocolor - Slice(K)

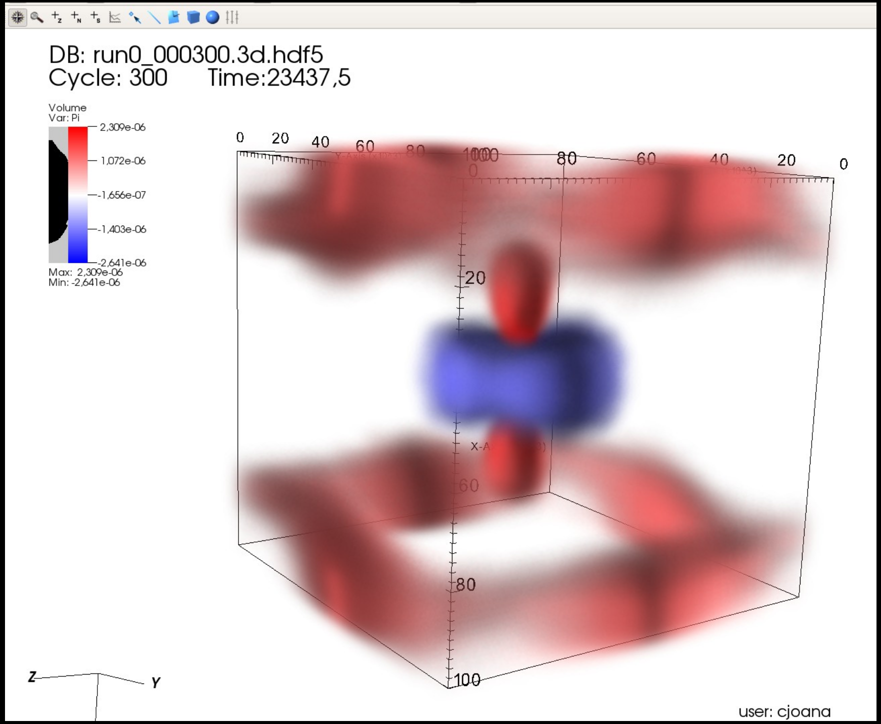
run0_*.3d.hdf5 database:Pi

- Slice
- Contour

Apply to ● active window ○ all windows

☑ Apply operators to all plots

☑ Apply subset selections to all plots



DB: run0_000300.3d.hdf5

Cycle: 300 Time:23437,5

user: cjoana

Download & documentation:

https://wci.llnl.gov/simulation/computer-codes/visit/downloads

https://visit-sphinx-github-user-manual.readthedocs.io/en/develop/gui_manual/

Visit - Installation

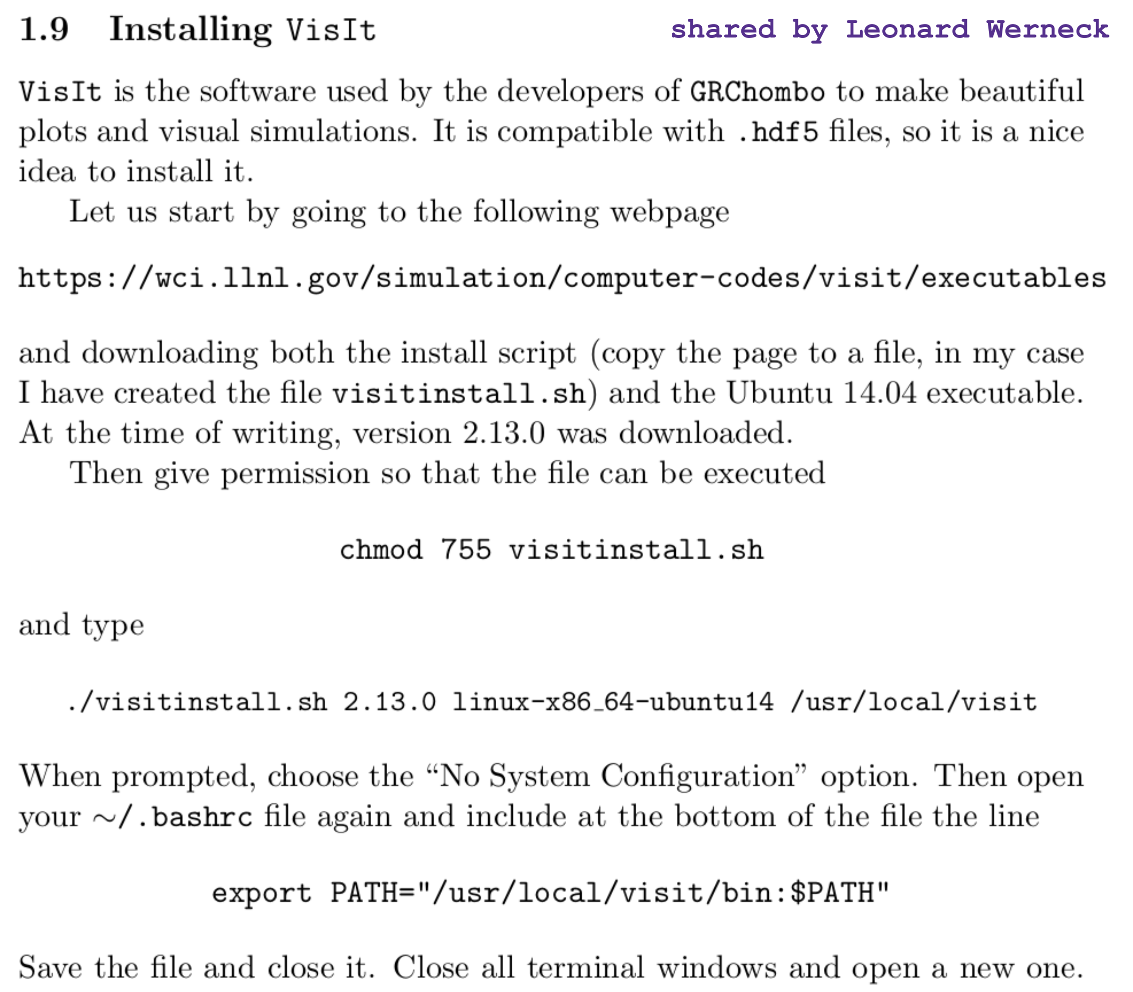
You probably already have it installed. If not, **don't worry**, you will manage :)

+ info:

https://github.com/GRChombo/GRChombo/wiki/Visualising-outputs

https://github.com/GRChombo/GRChombo/wiki/files/grchombo_ubuntu.pdf

shared by Leonard Werneck →



1.9 Installing VisIt

shared by Leonard Werneck

VisIt is the software used by the developers of GRChombo to make beautiful plots and visual simulations. It is compatible with .hdf5 files, so it is a nice idea to install it.

Let us start by going to the following webpage

https://wci.llnl.gov/simulation/computer-codes/visit/executables

and downloading both the install script (copy the page to a file, in my case I have created the file visitinstall.sh) and the Ubuntu 14.04 executable. At the time of writing, version 2.13.0 was downloaded.

Then give permission so that the file can be executed

```
chmod 755 visitinstall.sh
```

and type

```
./visitinstall.sh 2.13.0 linux-x86_64-ubuntu14 /usr/local/visit
```

When prompted, choose the "No System Configuration" option. Then open your ~/.bashrc file again and include at the bottom of the file the line

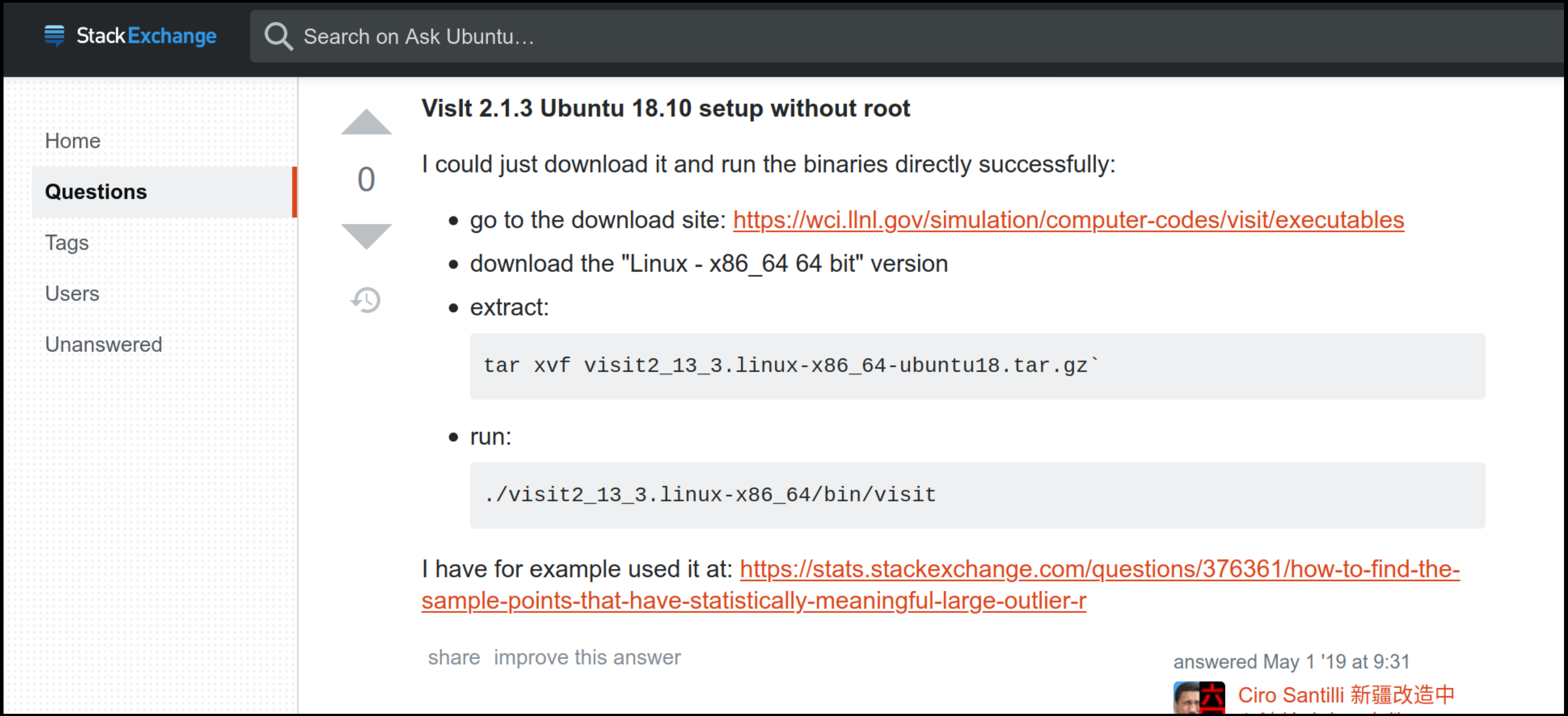
```
export PATH="/usr/local/visit/bin:$PATH"
```

Save the file and close it. Close all terminal windows and open a new one.

Visit - Installation

In Ubuntu: (not tested)

Last version avail. : VisIt 3.1.1 (Feb 2020)



VisIt 2.1.3 Ubuntu 18.10 setup without root

I could just download it and run the binaries directly successfully:

- go to the download site: https://wci.llnl.gov/simulation/computer-codes/visit/executables
- download the "Linux - x86_64 64 bit" version
- extract:

```
tar xvf visit2_13_3.linux-x86_64-ubuntu18.tar.gz`
```

- run:

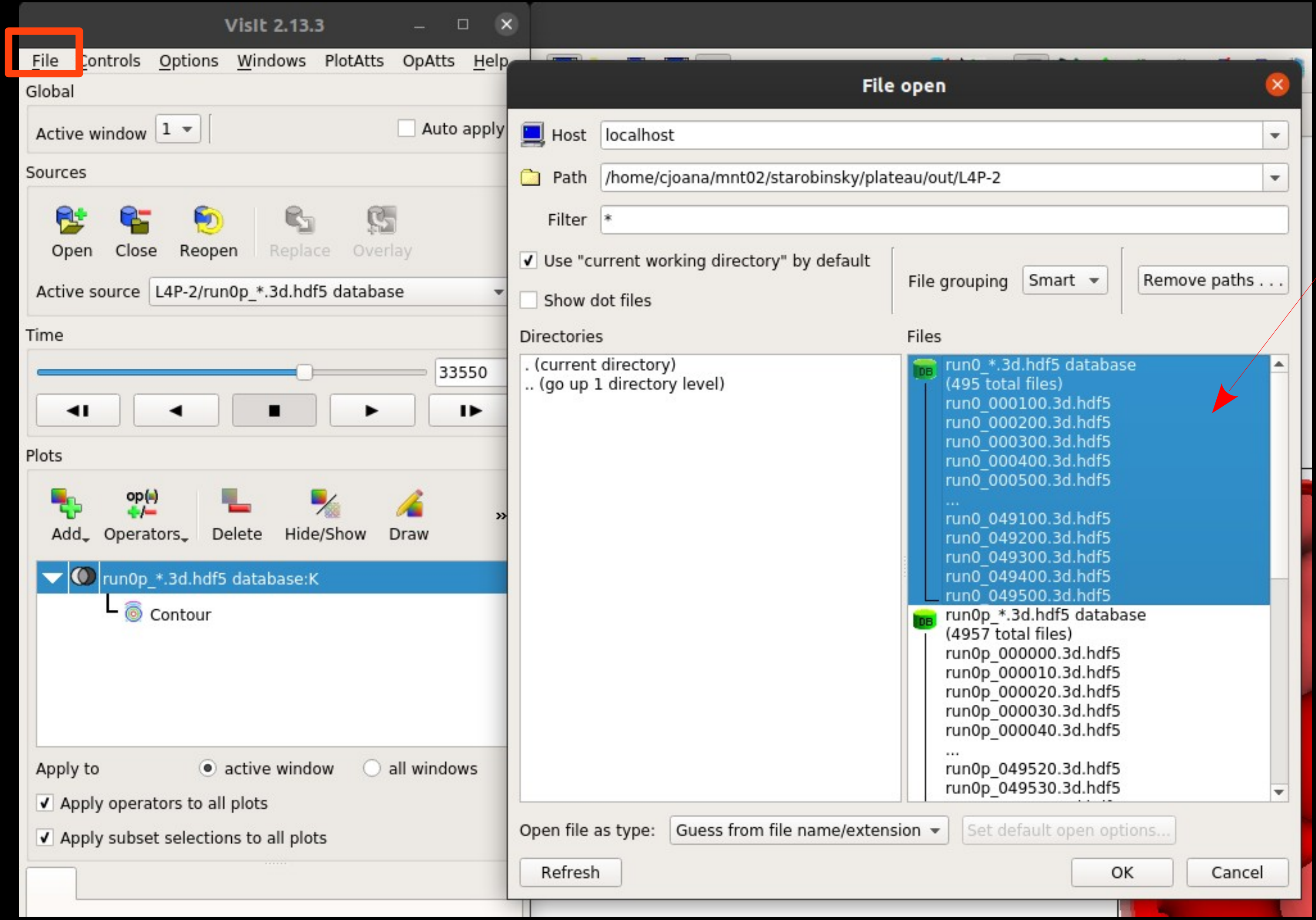
```
./visit2_13_3.linux-x86_64/bin/visit
```

I have for example used it at: https://stats.stackexchange.com/questions/376361/how-to-find-the-sample-points-that-have-statistically-meaningful-large-outlier-r

answered May 1 '19 at 9:31

Ciro Santilli 新疆改造中

Visit – GUI commands



Active window 1

□ Auto apply

Active source L4P-2/run0p_*.3d.hdf5 database

Time 33550

run0p_*.3d.hdf5 database:K

Apply to ● active window ○ all windows

☑ Apply operators to all plots

☑ Apply subset selections to all plots

Host localhost

Path /home/cjoana/mnt02/starobinsky/plateau/out/L4P-2

Filter *

☑ Use "current working directory" by default

□ Show dot files

File grouping Smart

Directories:

```
. (current directory)
.. (go up 1 directory level)
```

Files:

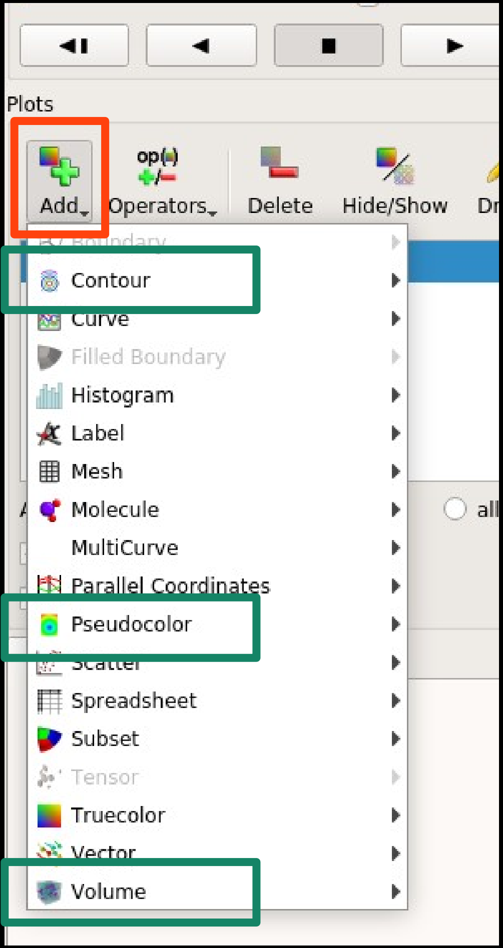
```
run0_*.3d.hdf5 database
(495 total files)
run0_000100.3d.hdf5
run0_000200.3d.hdf5
run0_000300.3d.hdf5
run0_000400.3d.hdf5
run0_000500.3d.hdf5
...
run0_049100.3d.hdf5
run0_049200.3d.hdf5
run0_049300.3d.hdf5
run0_049400.3d.hdf5
run0_049500.3d.hdf5
run0p_*.3d.hdf5 database
(4957 total files)
run0p_000000.3d.hdf5
run0p_000010.3d.hdf5
run0p_000020.3d.hdf5
run0p_000030.3d.hdf5
run0p_000040.3d.hdf5
...
run0p_049520.3d.hdf5
run0p_049530.3d.hdf5
```

Open file as type: Guess from file name/extension

Select multiple files with the same prefix

./<prefix>_000000.hdf5

Visit – GUI commands

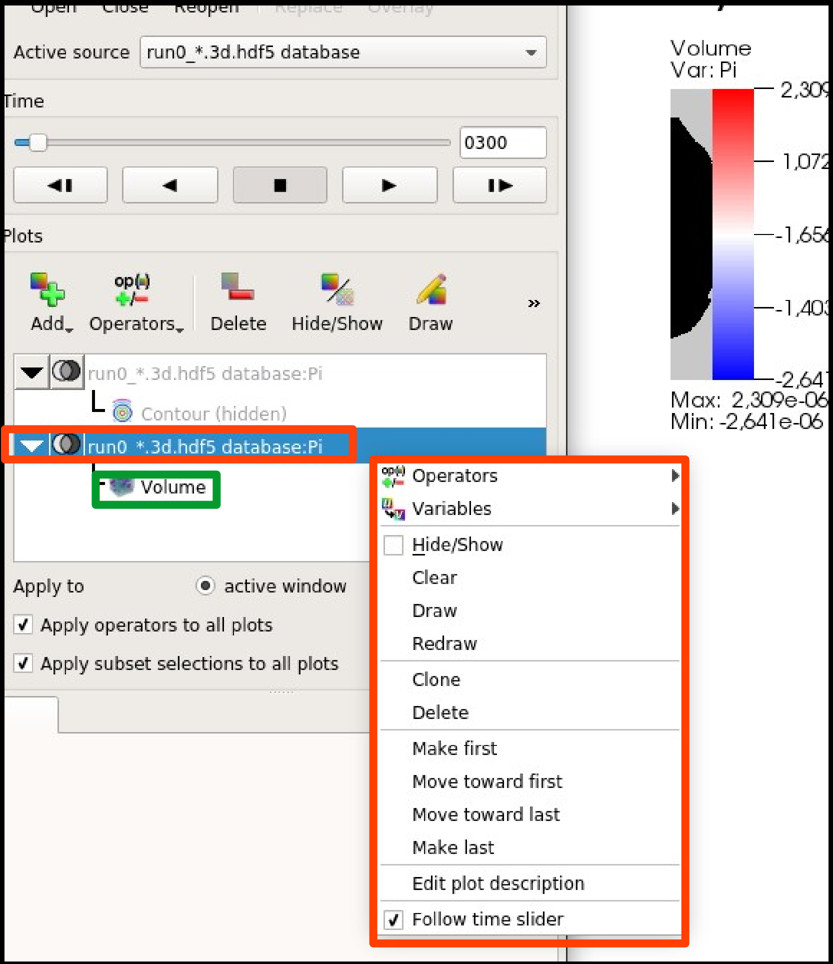


Different plotting functions:

- Contour, for 2D & 3D plots
- Pseudocolor, mainly for 2D plots.
- Volume, mainly for 3D plots

Feel free to play with the other options!

Visit – GUI commands



Active source: run0_*.3d.hdf5 database

Time: 0300

Plots:
- run0_*.3d.hdf5 database:Pi
  - Contour (hidden)
- run0_*.3d.hdf5 database:Pi
  - Volume

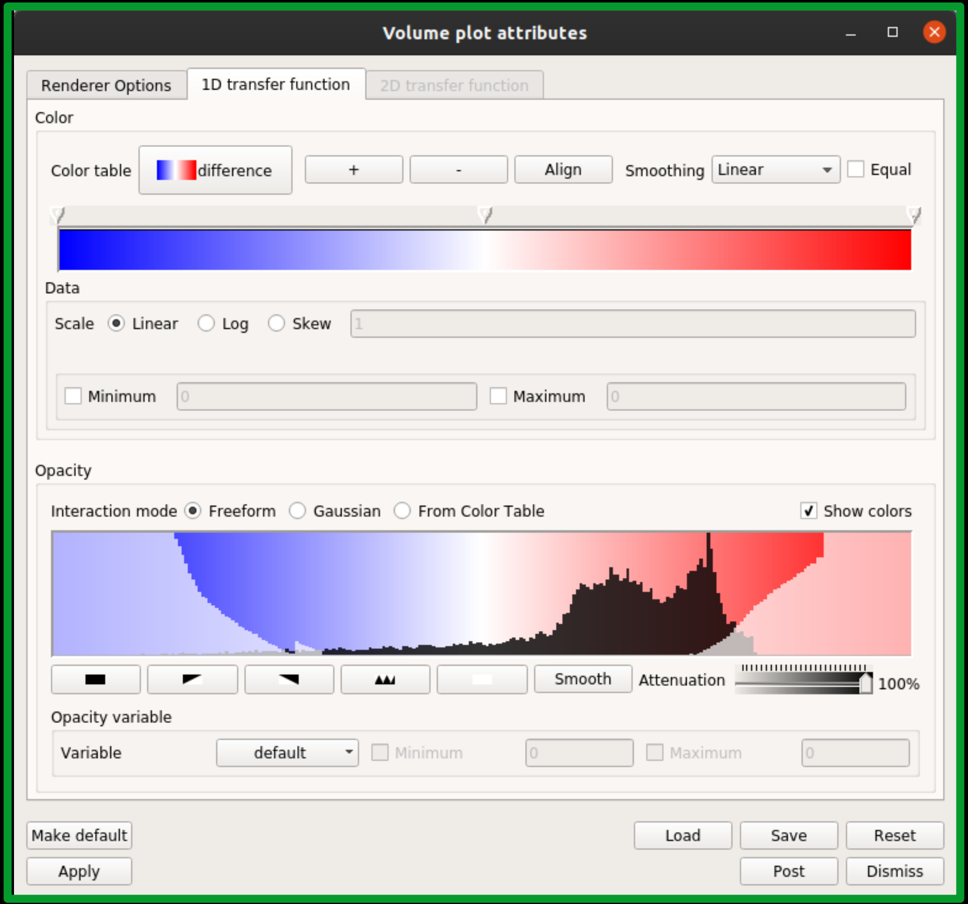
Apply to ● active window

☑ Apply operators to all plots

☑ Apply subset selections to all plots

Volume
Var: Pi

Max: 2,309e-06
Min: -2,641e-06



Volume plot attributes

Color table: difference
Smoothing: Linear
□ Equal

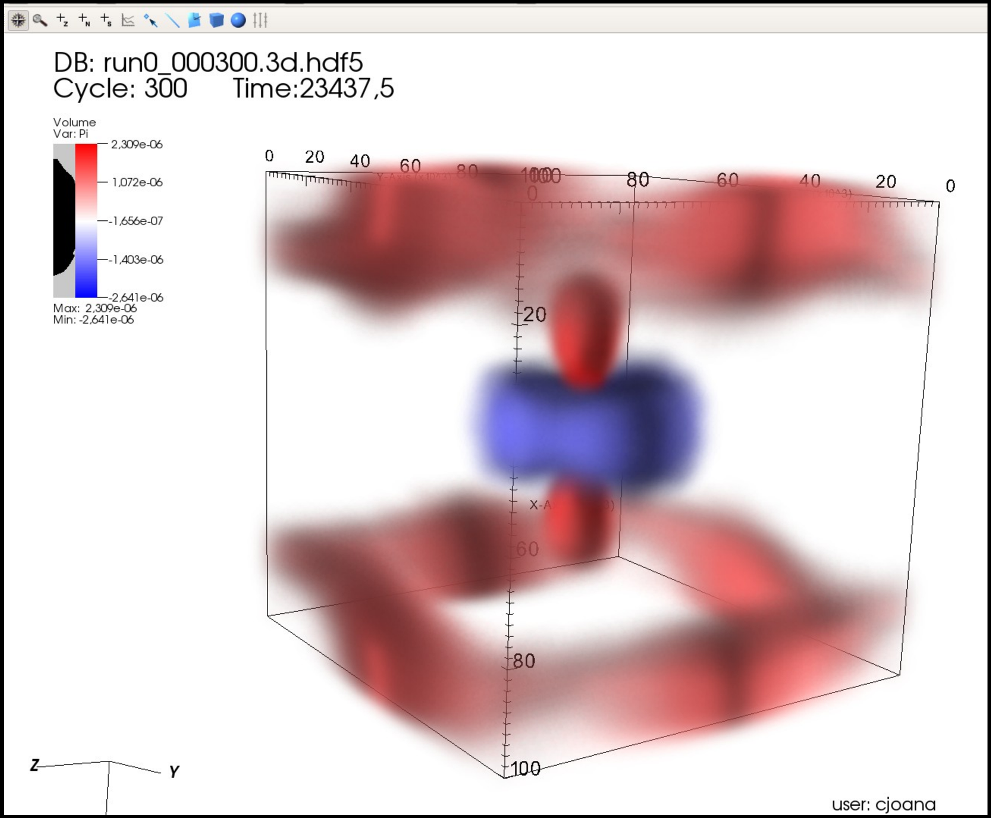
Data
Scale ● Linear ○ Log ○ Skew — 1
□ Minimum 0
□ Maximum 0

Opacity
Interaction mode ● Freeform ○ Gaussian ○ From Color Table
☑ Show colors
Attenuation 100%

Opacity variable
Variable: default
□ Minimum 0
□ Maximum 0

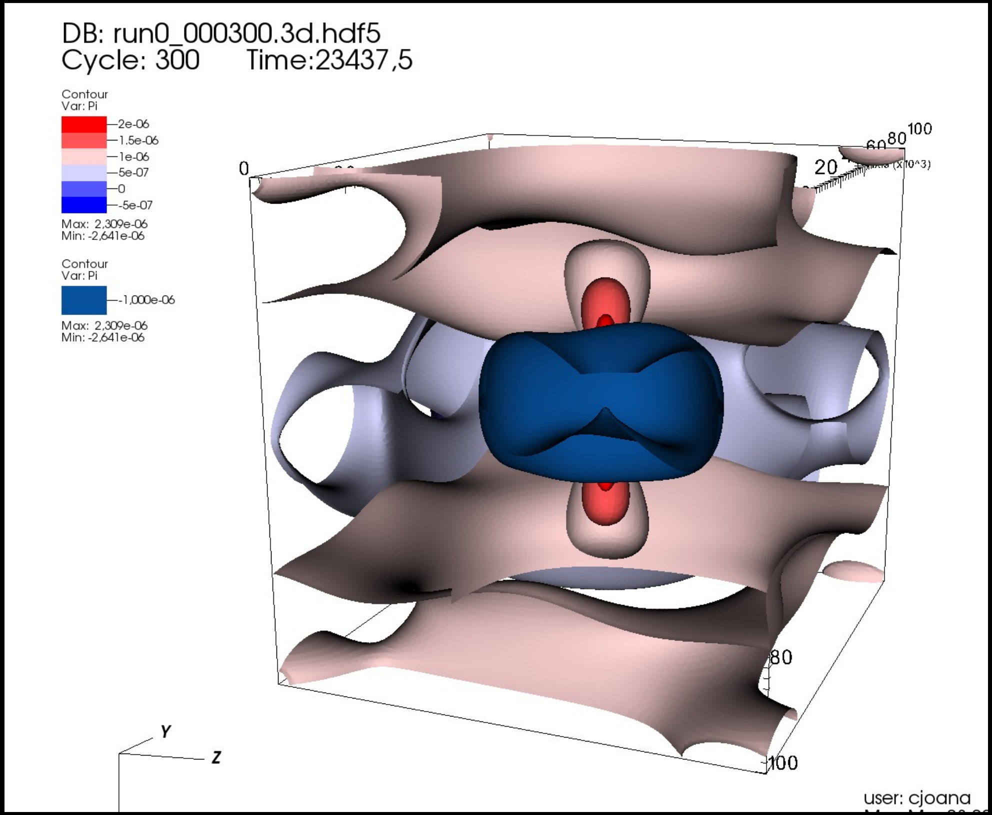
Visit – GUI commands

Volume (3D)



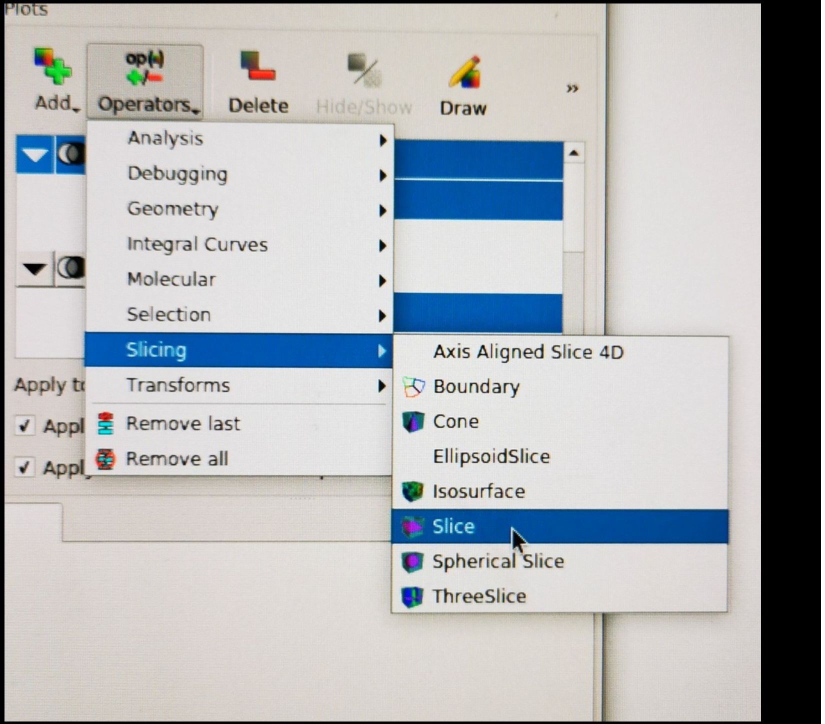
DB: run0_000300.3d.hdf5
Cycle: 300 Time:23437,5

Contour (3D)

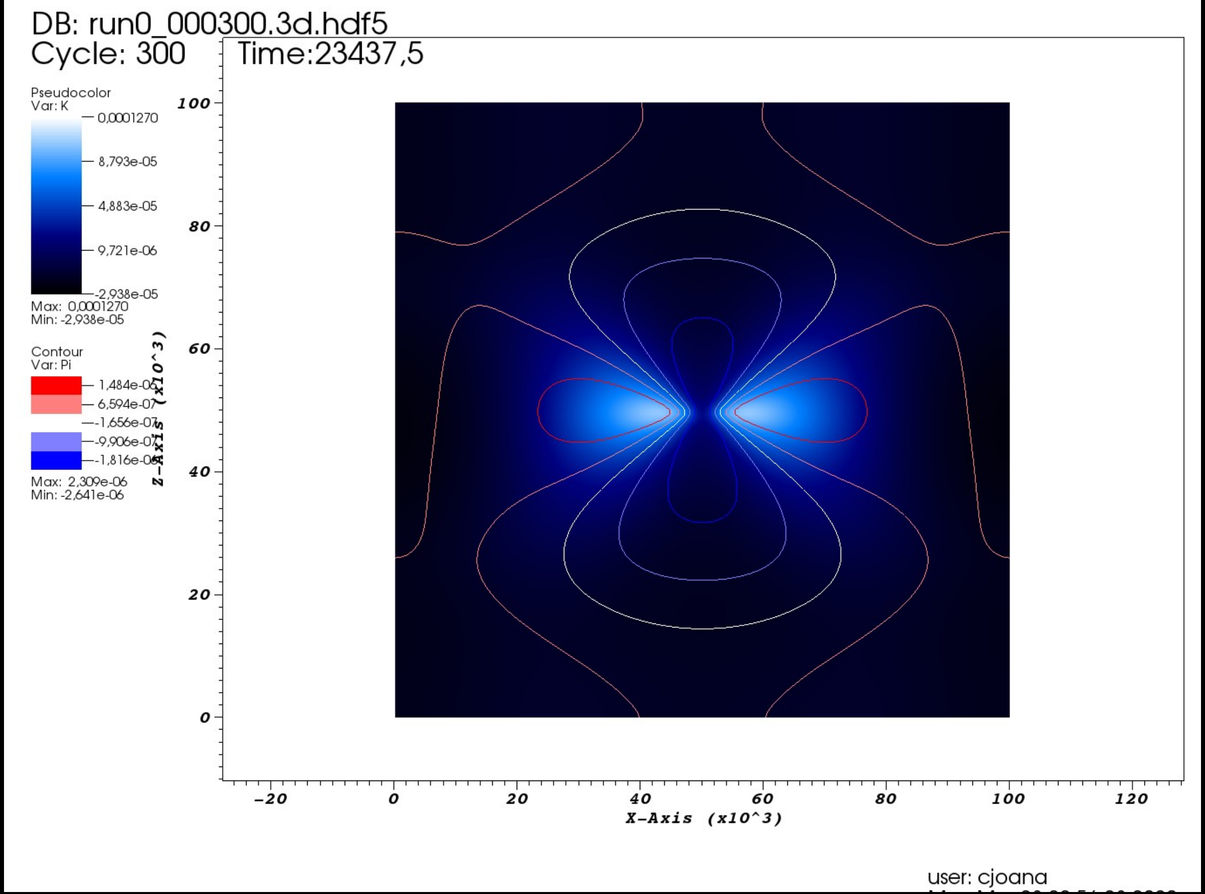


DB: run0_000300.3d.hdf5
Cycle: 300 Time:23437,5

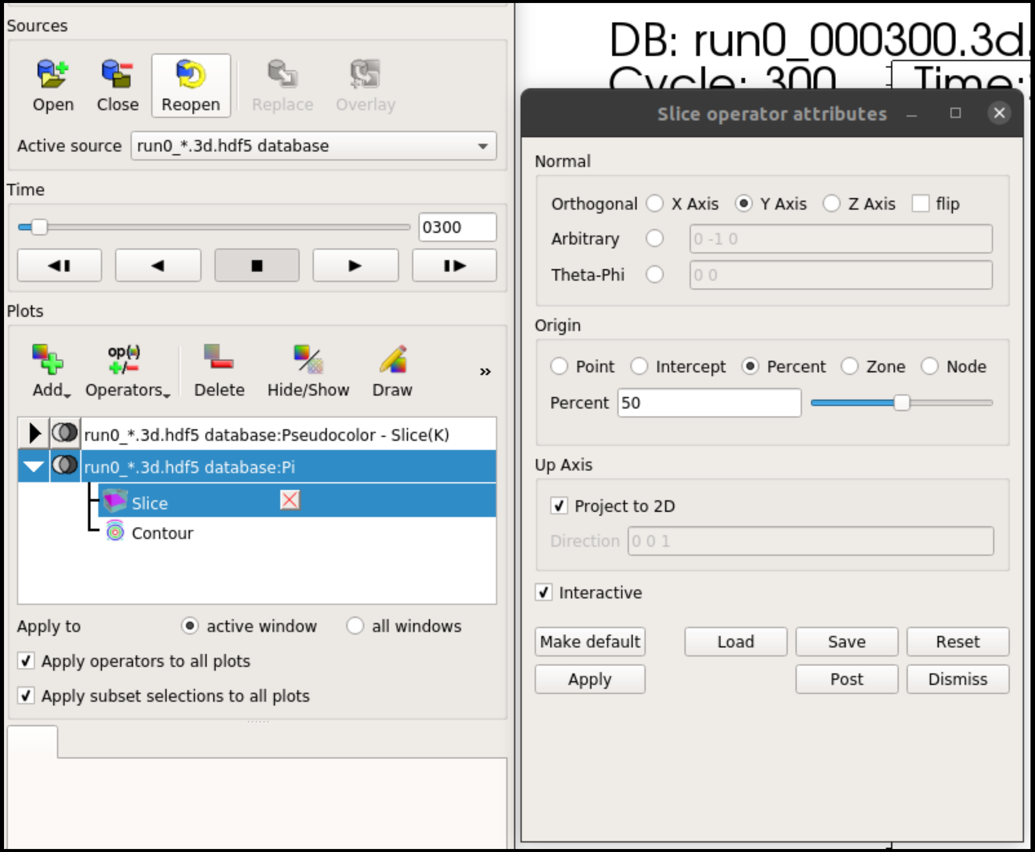
Visit – GUI commands



Pseudocolor + Contour (2D)



Visit – GUI commands



DB: run0_000300.3d

Cycle: 300 Time:

Sources

Active source run0_*.3d.hdf5 database

Time

0300

Plots

run0_*.3d.hdf5 database:Pseudocolor - Slice(K)

run0_*.3d.hdf5 database:Pi

Slice

Contour

Apply to ● active window ○ all windows

☑ Apply operators to all plots

☑ Apply subset selections to all plots

Slice operator attributes

Normal

Orthogonal ○ X Axis ● Y Axis ○ Z Axis □ flip

Arbitrary ○ 0 -1 0

Theta-Phi ○ 0 0

Origin

○ Point ○ Intercept ● Percent ○ Zone ○ Node

Percent 50

Up Axis

☑ Project to 2D

Direction 0 0 1

☑ Interactive

Suggestion:

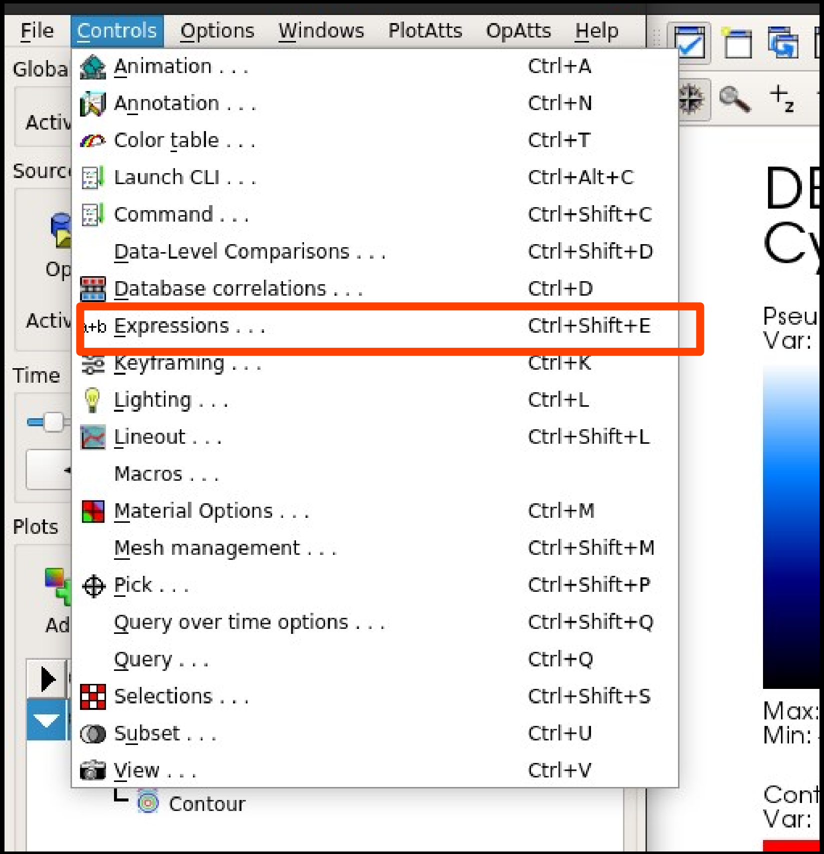
1) Go to Slice settings

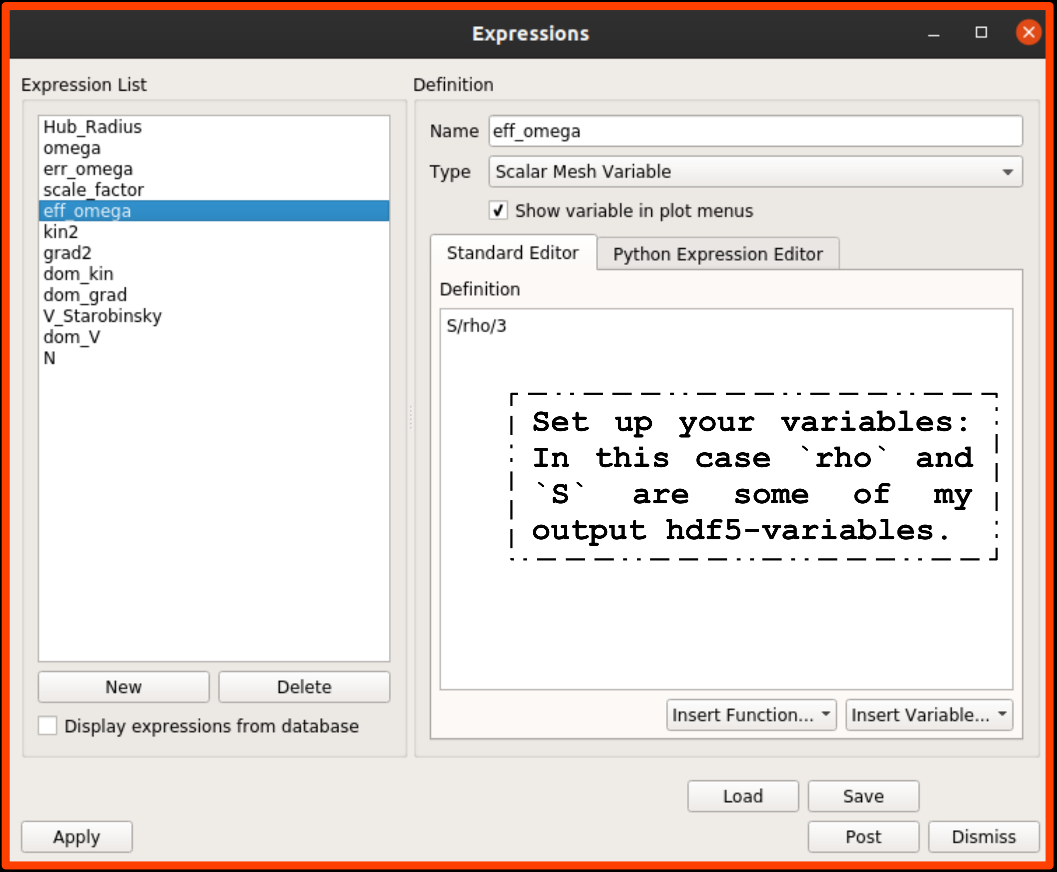
2) Select Ortogonal axis (e.g. Y)

3) Chose `Percent` to select the cord.

Visit – GUI commands

Add derived variable:


File
Controls
Options
Windows
PlotAtts
OpAtts
Help
Animation . . . Ctrl+A
Annotation . . . Ctrl+N
Color table . . . Ctrl+T
Launch CLI . . . Ctrl+Alt+C
Command . . . Ctrl+Shift+C
Data-Level Comparisons . . . Ctrl+Shift+D
Database correlations . . . Ctrl+D
a+b Expressions . . . Ctrl+Shift+E
Keyframing . . . Ctrl+K
Lighting . . . Ctrl+L
Lineout . . . Ctrl+Shift+L
Macros . . .
Material Options . . . Ctrl+M
Mesh management . . . Ctrl+Shift+M
Pick . . . Ctrl+Shift+P
Query over time options . . . Ctrl+Shift+Q
Query . . . Ctrl+Q
Selections . . . Ctrl+Shift+S
Subset . . . Ctrl+U
View . . . Ctrl+V
Contour



Expressions

Expression List

- Hub_Radius
- omega
- err_omega
- scale_factor
- eff_omega
- kin2
- grad2
- dom_kin
- dom_grad
- V_Starobinsky
- dom_V
- N

Definition

Name: eff_omega

Type: Scalar Mesh Variable

☑ Show variable in plot menus

Standard Editor | Python Expression Editor

Definition

```
S/rho/3
```

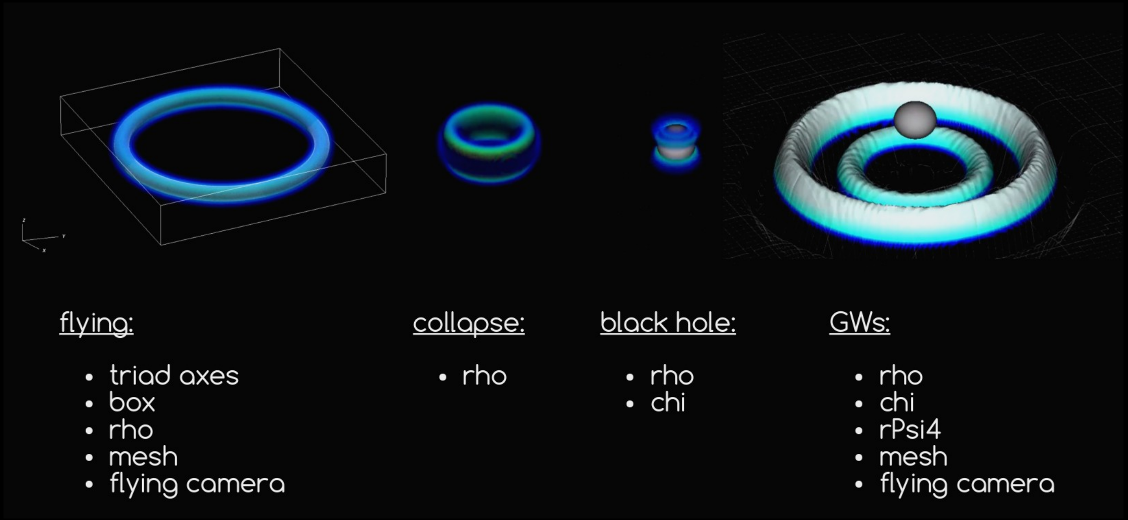
☐ Display expressions from database

Set up your variables: In this case `rho` and `S` are some of my output hdf5-variables.

Visit – Script mode

- VisIt can also be used in `script` mode. But this is <u>not covered</u> on these slides.

- In `script` mode, beautiful animation can be made:



<u>flying:</u>

- triad axes
- box
- rho
- mesh
- flying camera

<u>collapse:</u>

- rho

<u>black hole:</u>

- rho
- chi

<u>GWs:</u>

- rho
- chi
- rPsi4
- mesh
- flying camera

→ See Josu’s slides from 2019, or ask him ;)

Questions ?

VisIt 2.6 Patch 2

(c) 2000-2015 LLNS. All Rights Reserved.
VisIt 2.6.2, svn revision Unknown
March 2013

Processing config file...