Scientific Visualization

Tools & Techniques

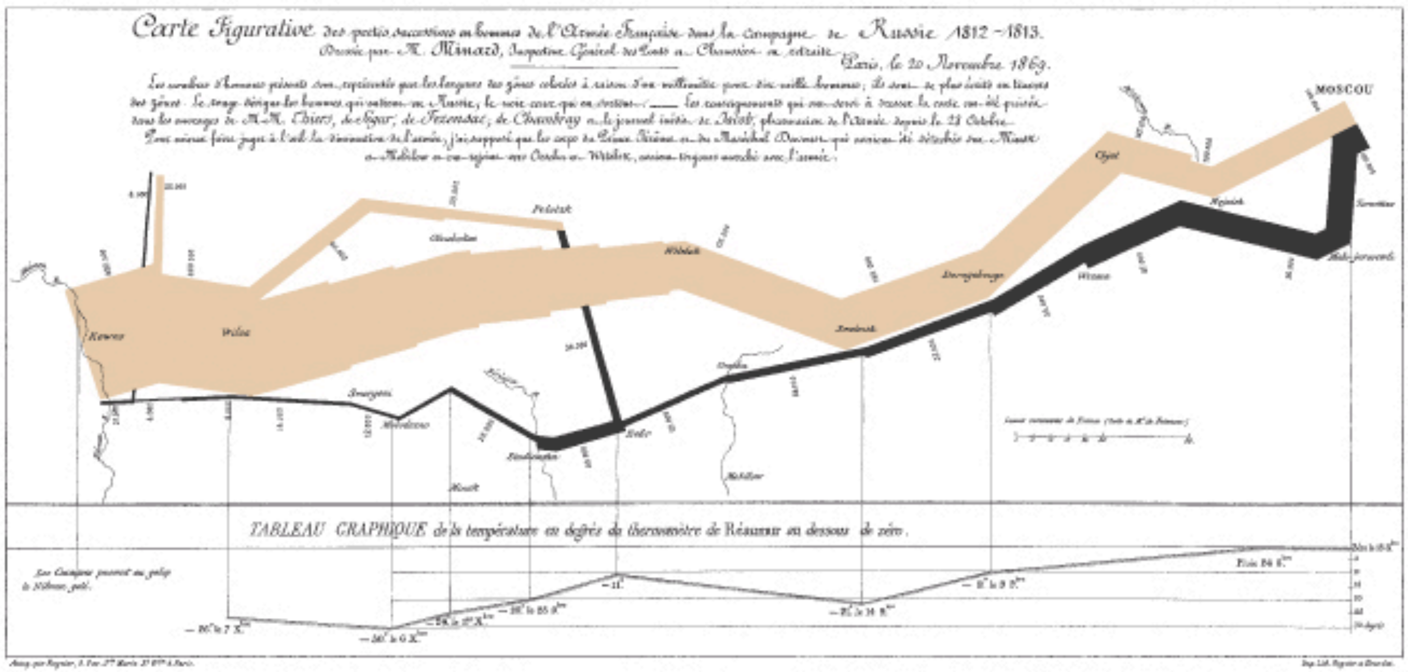
Eliot Feibush

Research Computing Boot Camp

November 1, 2018

Early Multivariate Data Vis

Not so good for Napoleon in 1812



Napoleon's March to Moscow   The War of 1812

Charles Joseph Minard

This classic of Charles Joseph Minard (1781–1870), the French engineer, shows the terrible fate of Napoleon's army in Russia. Described by E. J. Marey as seeming to defy the pen of the historian by its brutal eloquence, this combination of data map and time-series, drawn in 1869, portrays the devastating losses suffered in Napoleon's Russian campaign of 1812. Beginning at the left on the Polish-Russian border near the Niemen River, the thick band shows the size of the army (422,000 men) as it invaded Russia in June 1812. The width of the band indicates the size of the army at each place on the map. In September, the army reached Moscow, which was by then sacked and deserted, with 100,000 men. The path of Napoleon's retreat from Moscow is depicted by the darker, lower band, which is linked to a temperature scale and dates at the bottom of the chart. It was a bitterly cold winter, and many froze on the march out of Russia. As the graphic shows, the crossing of the Berezina River was a disaster, and the army finally struggled back into Poland with only 10,000 men remaining. Also shown are the movements of auxiliary troops, as they sought to protect the rear and the flank of the advancing army. Minard's graphic tells a rich, coherent story with its multivariate data, far more enlightening than just a single number bouncing along over time. Six variables are plotted: the size of the army, its location on a two-dimensional surface, direction of the army's movement, and temperature on various dates during the retreat from Moscow. It may well be the best statistical graphic ever drawn.

Edward R. Tufte, *The Visual Display of Quantitative Information*   Graphics Press Box 430 Cheshire, Connecticut 06410

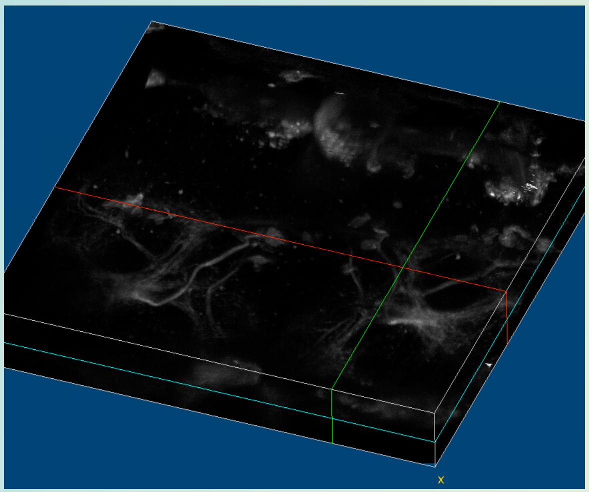
Credit: Edward Tufte via Charles Minard 1869

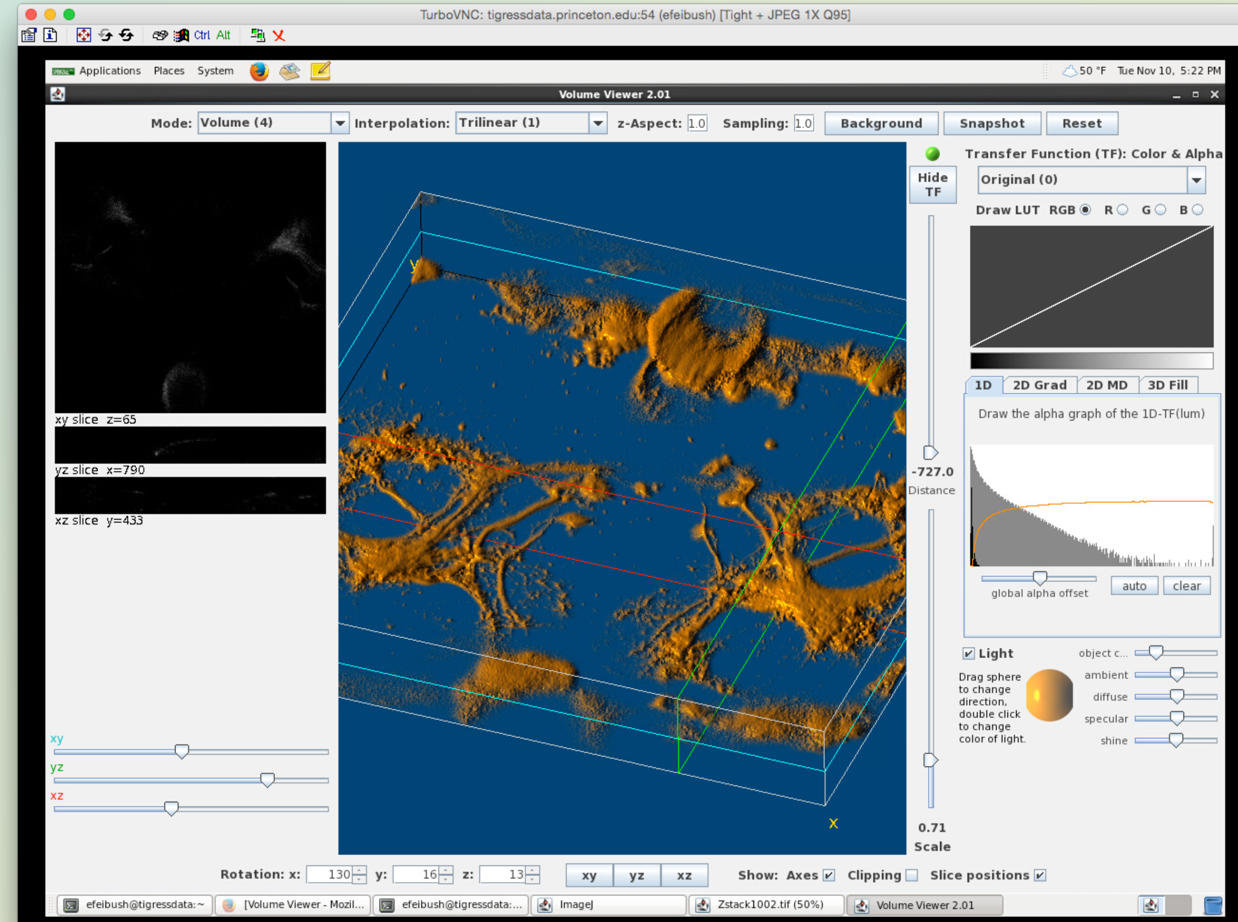
Modern Scientific Data Vis

Simulate

or

Acquire





Analyse

Visualize

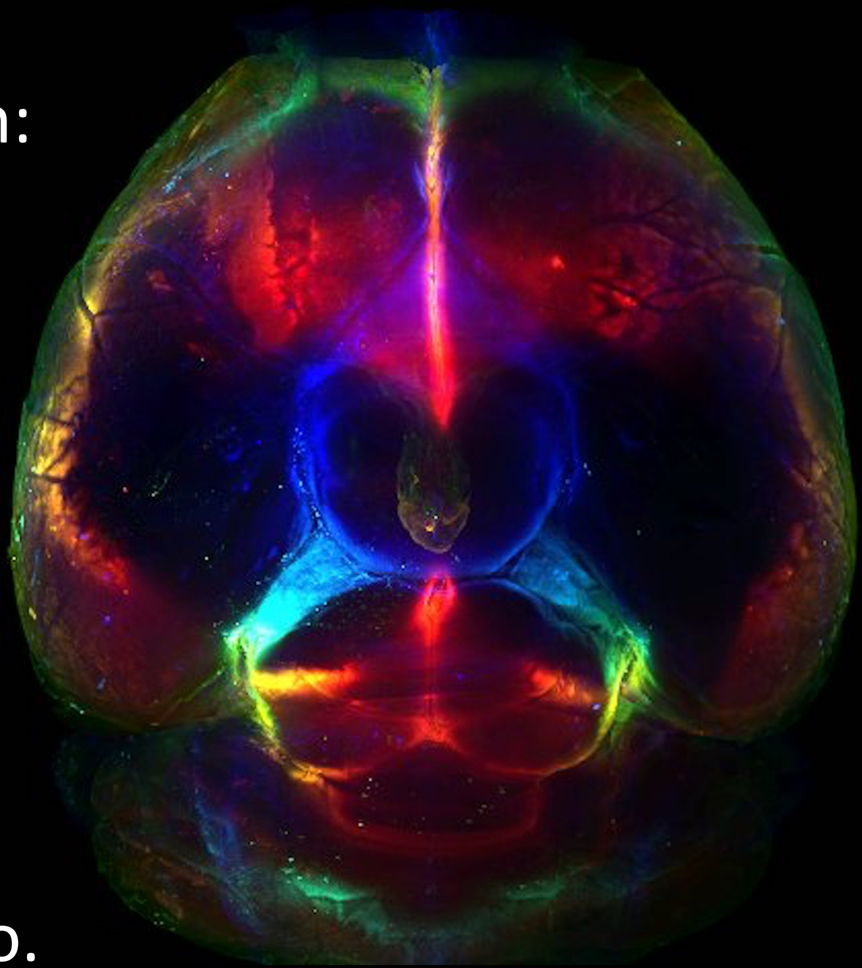
Visual ization

Human visual perception system:

Very high spatial resolution.
Very fast temporal resolution.

Detect anomalies.
Perceive structure.
Identify motion.

Pathway for large amount of info.



Light sheet microscopy, color-coded by depth - Thomas Pisano - PNI

Scientific Visualization

Simulations generate data.

Acquire data from experiments.

GTS

XGC

M3D-C1

MDSplus

...

Explore

Communicate

Based on computer graphics

- points
- lines
- polygons, surface mesh
  - 3D transformations
  - hidden surface removal
  - shading
  - lighting

Vis Plot Types
(Based on graphics primitives)

Points

Lines

Vectors

Contour lines & isosurfaces

Polygons, mesh

Volume

Designing a Visualization

I want a visualization of my climate model.

Researcher

Map your data to a plot type.

Vis Guy

2-D/3-D Compute grid:

scalar or vector

per point, per cell

Selection + Operators

Getting to Know Your Data

Geometric range

Numerical domain (min, max)

- Histogram
- Outliers
- Features
- Local / Global (steps)

Presentation

Dimensionality of Data

$f(x)$

$f(x, time)$ f(x, i)

$f(x, y)$

$f(x, y, time)$

$f(x, y, z)$

$f(x, y, z, time)$

Understanding Complexity !

Time dependent data is a good candidate for animation.

Python

Very convenient for converting data to vis formats.

Analyze & extract.

Good economy for programmer/user.

matplotlib

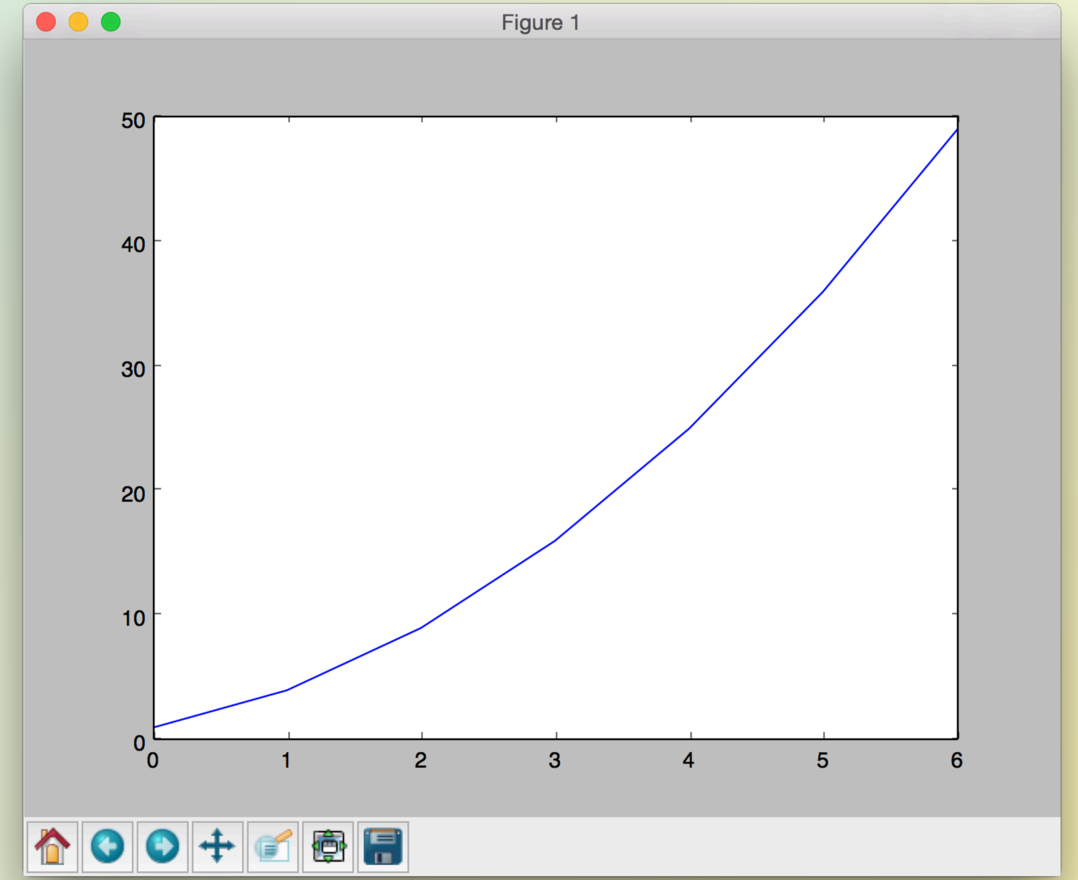
  Examples: *f(x), f(x,y), f(x,y,t)*

numpy, scipy, PIL

tkinter

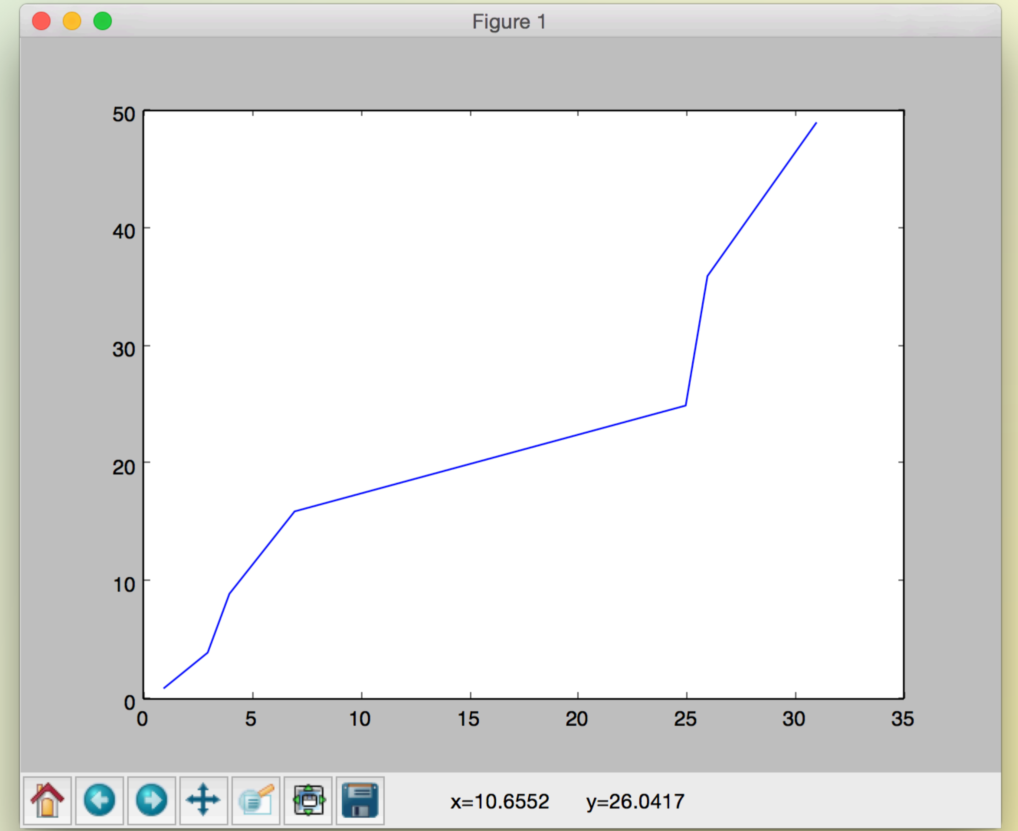
matplotlib *f(x)*

```
import matplotlib.pyplot as plt
y = [1, 4, 9, 16, 25, 36, 49]
plt.plot(y)
plt.show()
```



matplotlib *f(x)*

```
import matplotlib.pyplot as plt
y = [1, 4, 9, 16, 25, 36, 49]
x = [1, 3, 4, 7, 25, 26, 31]
plt.plot(x,y)
plt.show()
```



---

matplotlib.org

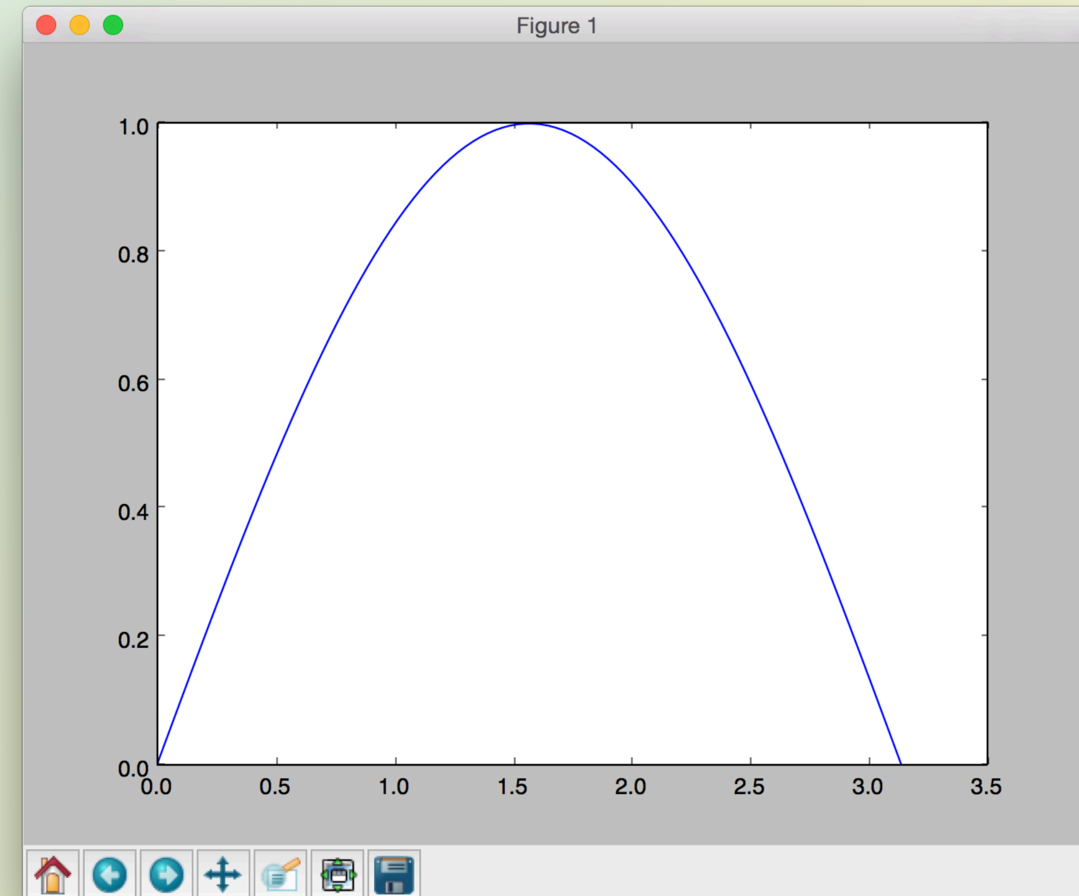
- docs, API
- Examples, Tutorials

matplotlib + numpy *f(x)*

```
import matplotlib.pyplot as plt
import numpy
x = numpy.arange(0., numpy.pi, .01)    # loop with floats
y = numpy.sin(x)
plt.plot(x,y)
plt.show()
```

x and y are
1D numpy.ndarray

arange takes floats



Anaconda Python

Very good distribution of Python.

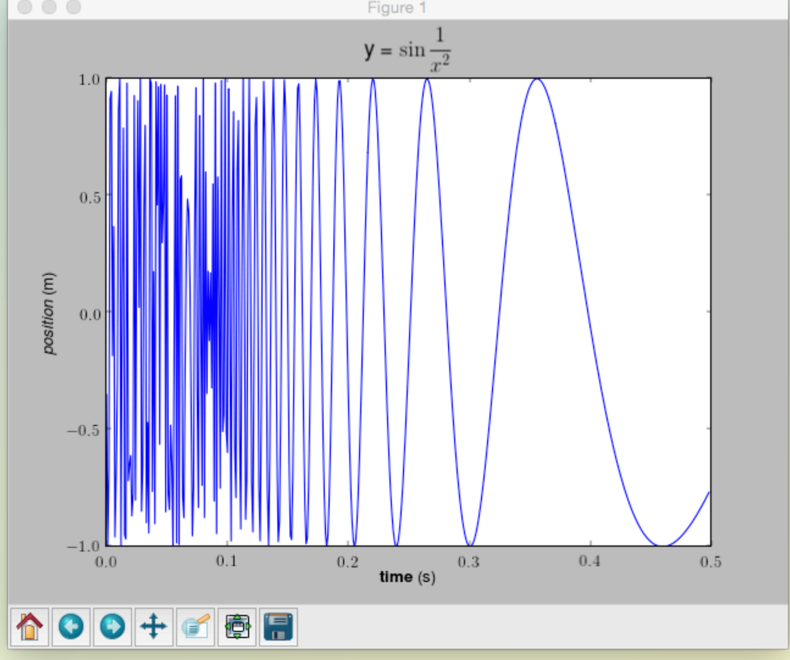
Integration of packages: numpy, scipy, PIL, many others

- Free
- Mac, Windows, Linux
  - /usr/pppl - module avail anaconda

Mac – do not change your original python

- *Used by OS, Safari, etc.*

Latex in Graphs



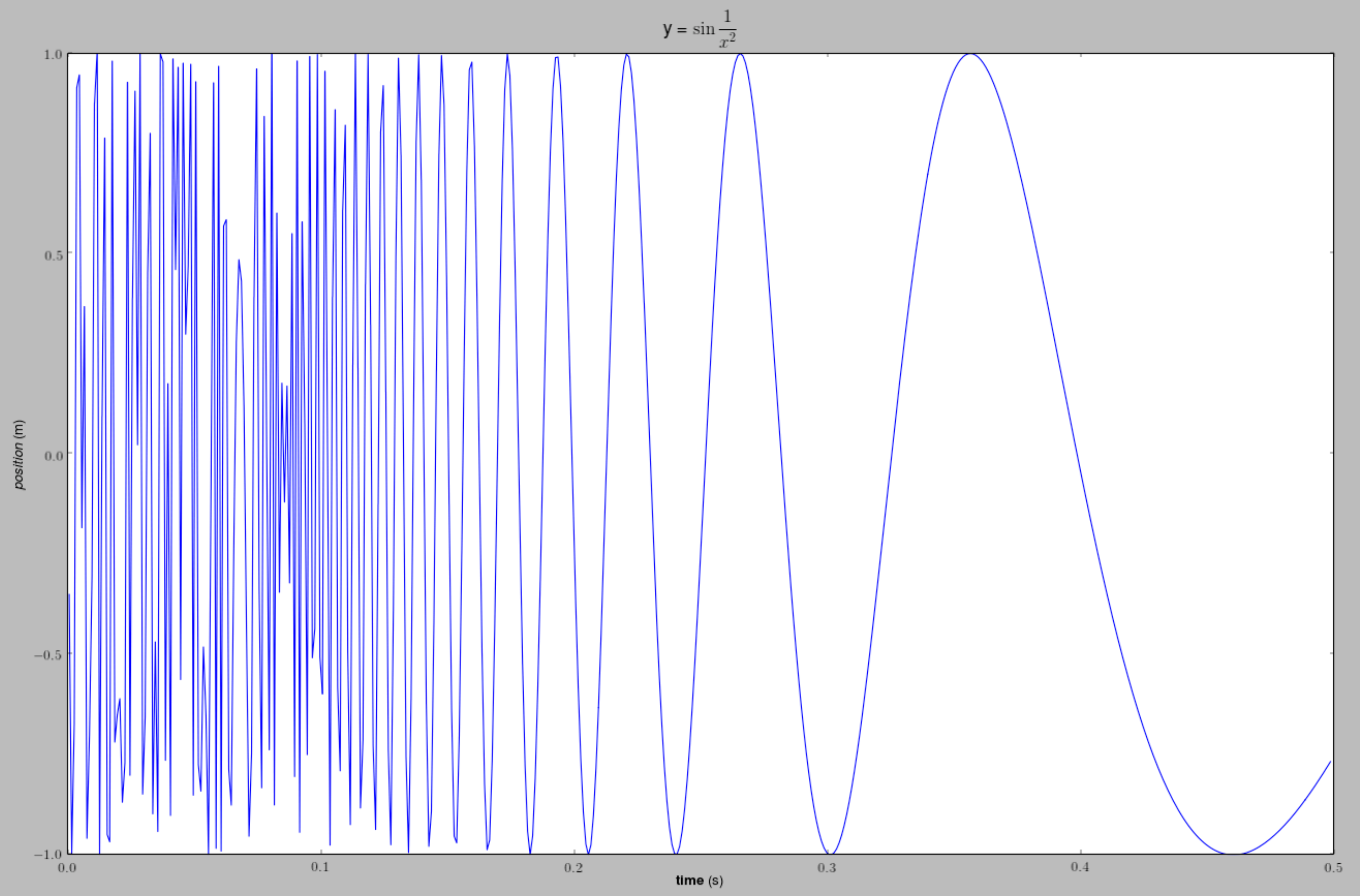
export PATH=$PATH:/Library/TeX/texbin

```
plt.rc(“text”, usetex=True)
plt.plot(x,y)
plt.xlabel(r"\textbf{time} (s)")
plt.ylabel(r"\textit{position} (m)")
```

raw string r before quotes *escapes* tex format instead of \t becoming a tab character

```
plt.title(r"y = "
   r"$\displaystyle\sin\frac{1}{x^2}$",
   fontsize=16)

plt.show()
```

Python system call to /usr/bin/latex on portal is OK.



y = $\sin\frac{1}{x^2}$
1.0
0.5
0.0
−0.5
−1.0
position (m)
0.0
0.1
0.2
0.3
0.4
0.5
time (s)

matplotlib – Save to PDF File

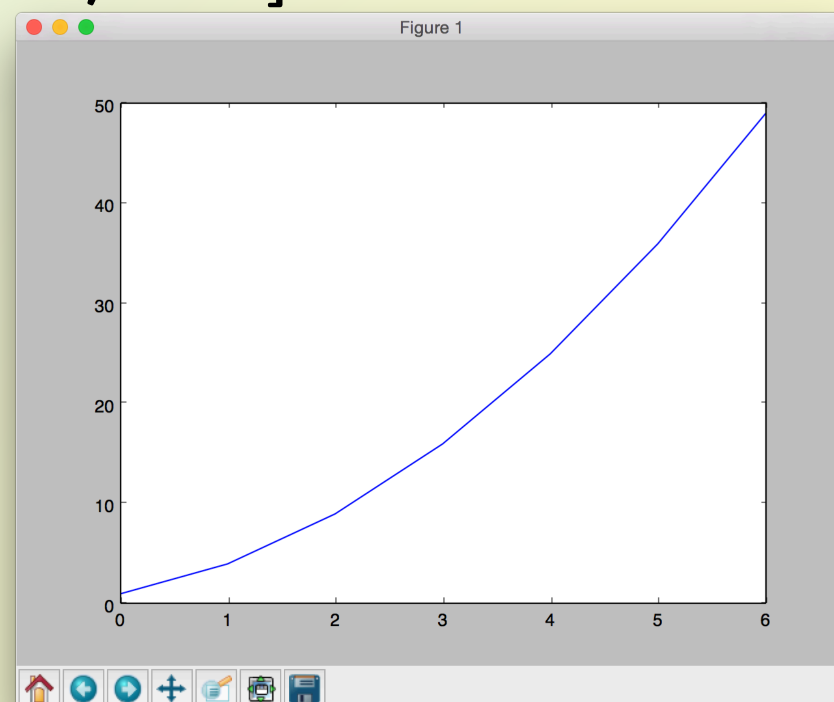
```
import matplotlib.pyplot as plt
```

```
from matplotlib.backends.backend_pdf import PdfPages
pdfp = PdfPages('multipage.pdf')    # starts pdf file, 1 plot per page
```

```
y = [1, 4, 9, 16, 25, 36, 49]
plt.plot(y)
```

```
plt.savefig(pdfp, format='pdf')
        # save fig to file before showing
pdfp.close()     # after last save fig
```

```
plt.show()
```

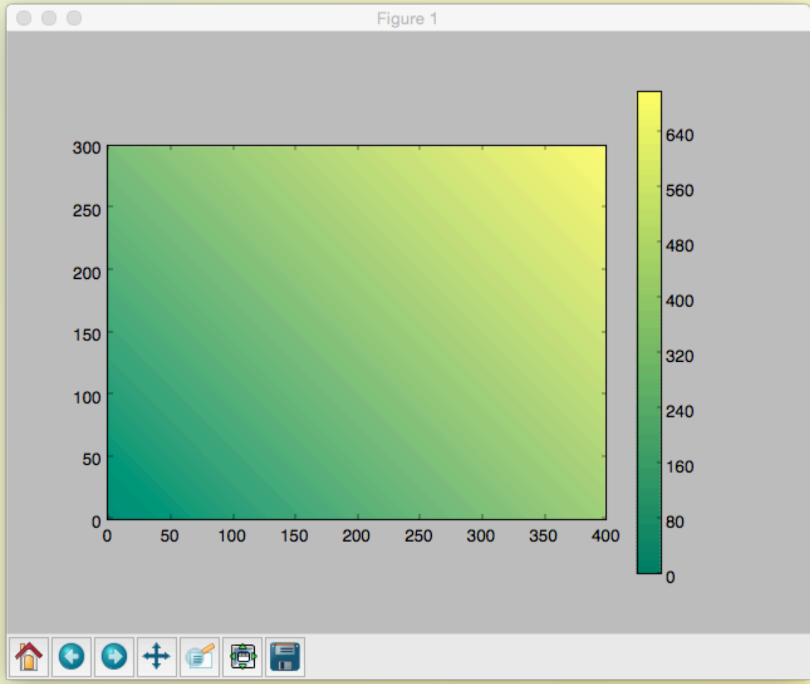


Save Multiple Graphs to 1 PDF File

```
import matplotlib.pyplot as plt
from matplotlib.backends.backend_pdf import PdfPages
pdfp = PdfPages('multipage.pdf')    # starts pdf file, 1 plot per page

plt.plot( … )      # Figure 1, page 1
plt.savefig(pdfp, format='pdf')
plt.show()

plt.plot( … )      # Figure 2, page 2
plt.savefig(pdfp, format='pdf')
plt.show()

plt.plot( … )      # Figure 3, page 3
plt.savefig(pdfp, format='pdf')
plt.show()

pdfp.close()    # after last save fig
```

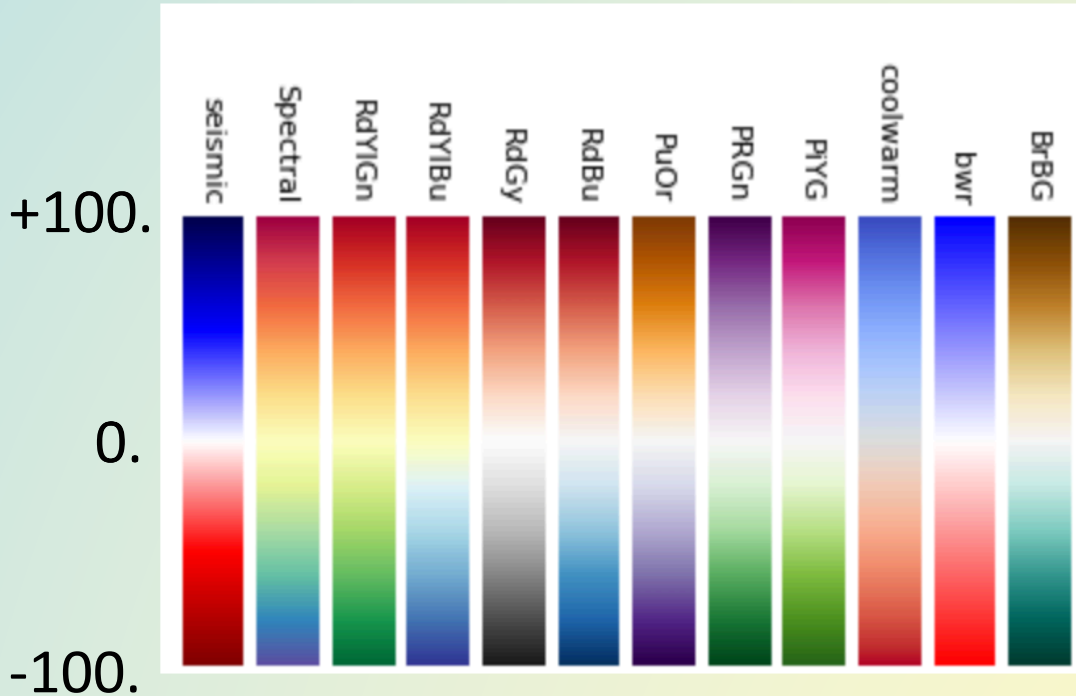
matplotlib + numpy *f(x,y)*

```
import matplotlib.pyplot as plt
import numpy
a = numpy.zeros( (300,400) )      # 2D ndarray

for i in range(300):
    for j in range(400):
        a[i,j] = i + j

plt.imshow(a, origin="lower", map=plt.cm.summer)
plt.colorbar()
plt.show()

# show names of colormaps
dir(plt.cm)

  # summer is a "sequential" colormap

  # summer_r to reverse order
```



Color Maps

Divergent color maps

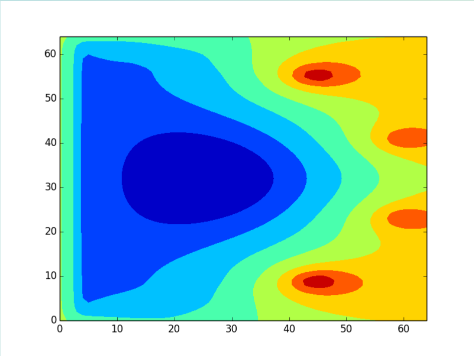


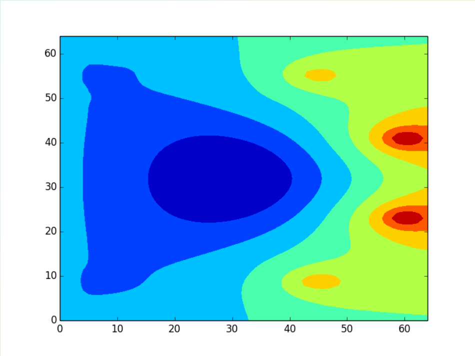
See also: Sequential & Qualitative colormaps discussion

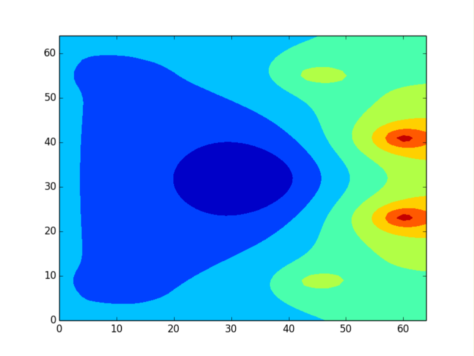
matplotlib.org/users/colormaps.html

f(x,y,t)

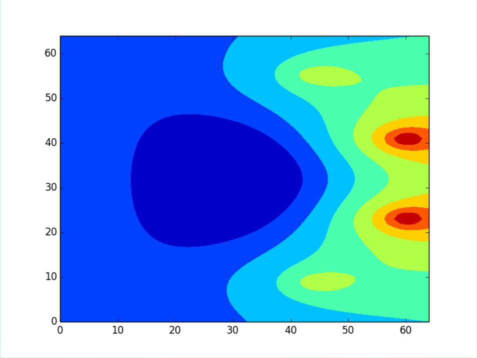
2D Magnetic Field for 84 time steps

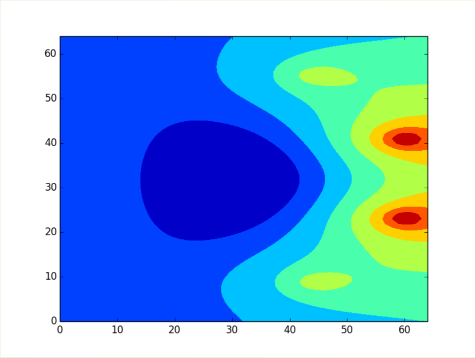


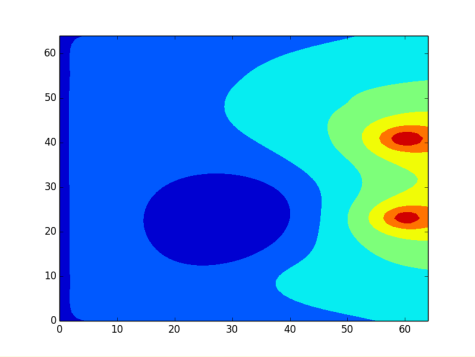




84 Image Files ...







Combine with ffmpeg into .mov or .mp4 movie file

http://www.princeton.edu/~efeibush/movies/rz.mov

matplotlib + numpy *f(x,y,t)* + netCDF

from netCDF4 import Dataset

```
import matplotlib.pyplot as plt
from scipy.io import netcdf

f = netcdf.netcdf_file("psiRZ.cdf", "r")
for v in f.variables:                  # f.variables is a dictionary
    print v                            # print keys, name of variable

rz = f.variables["pout_psiRZ"].data      # get the data
shape = rz.shape      # (84, 65, 65) tuple:  84 steps,  65 x 65 magnetic field
for i in range(shape[0]):              # for each time step
    rzt = rz[i][:][:]                  # get the time slice
    plt.contour(rzt)                   # make a contour plot
    plt.draw()                         # draw on screen for .1 second
    plt.pause(.1)
    fname = "figs/" + str(i).zfill(3) + ".png"  # 001.png ...
    plt.savefig(fname)                 # save each plot to a PNG file
```

Movie Maker Program

`ffmpeg`

- Most comprehensive
- Downloads for Mac & Windows; build for Linux*
- Command line Linux - module load ffmpeg/4.0.1

```
ffmpeg
   –y –f image2
   –framerate 8
   –pattern_type glob –I ‘*.png’
   –b:v 4000k
   –pix_fmt yuv420p
   psiRZ.mov
```

ImageMagick - Resize all images in directory

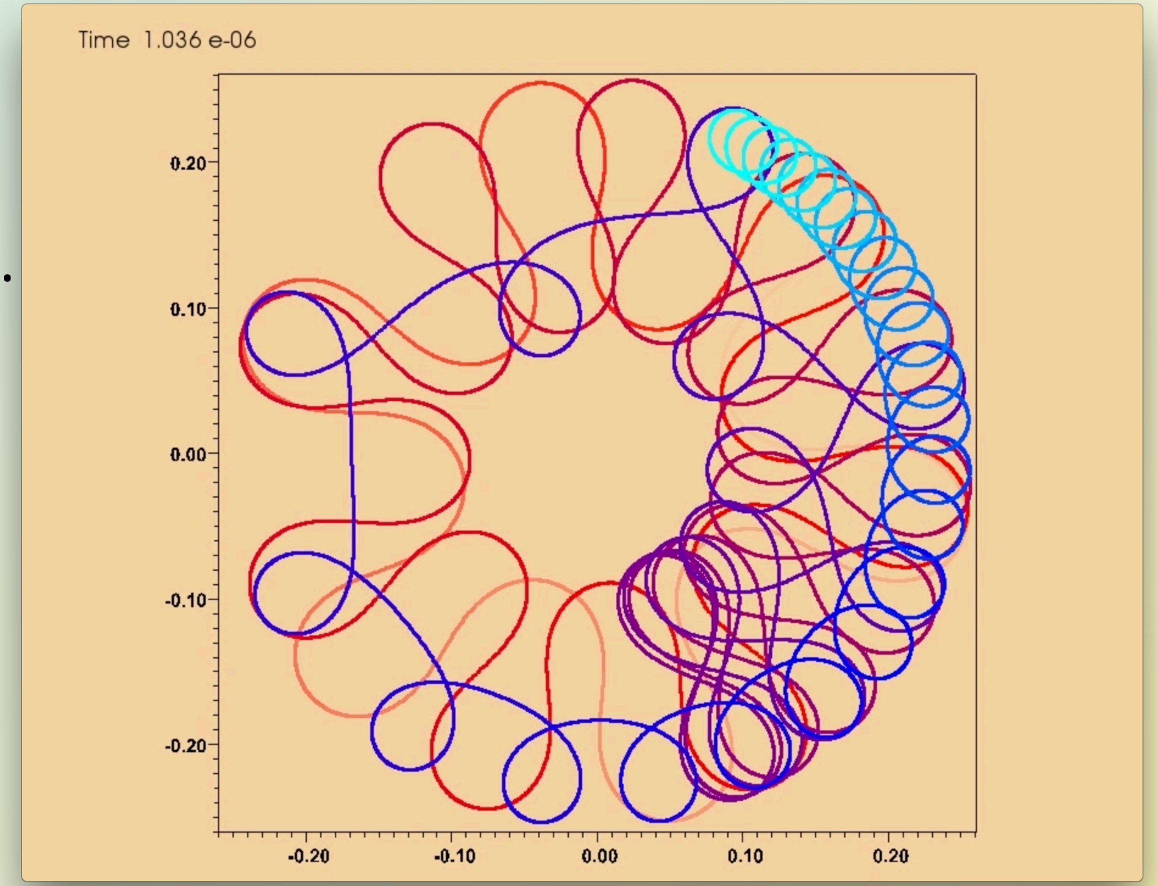
mogrify -resize 90% *.jpeg

Implementation

- Vis GUI – VisIt, ParaView
- VTK – Visualization ToolKit
- Graphics Primitives
- Pixels

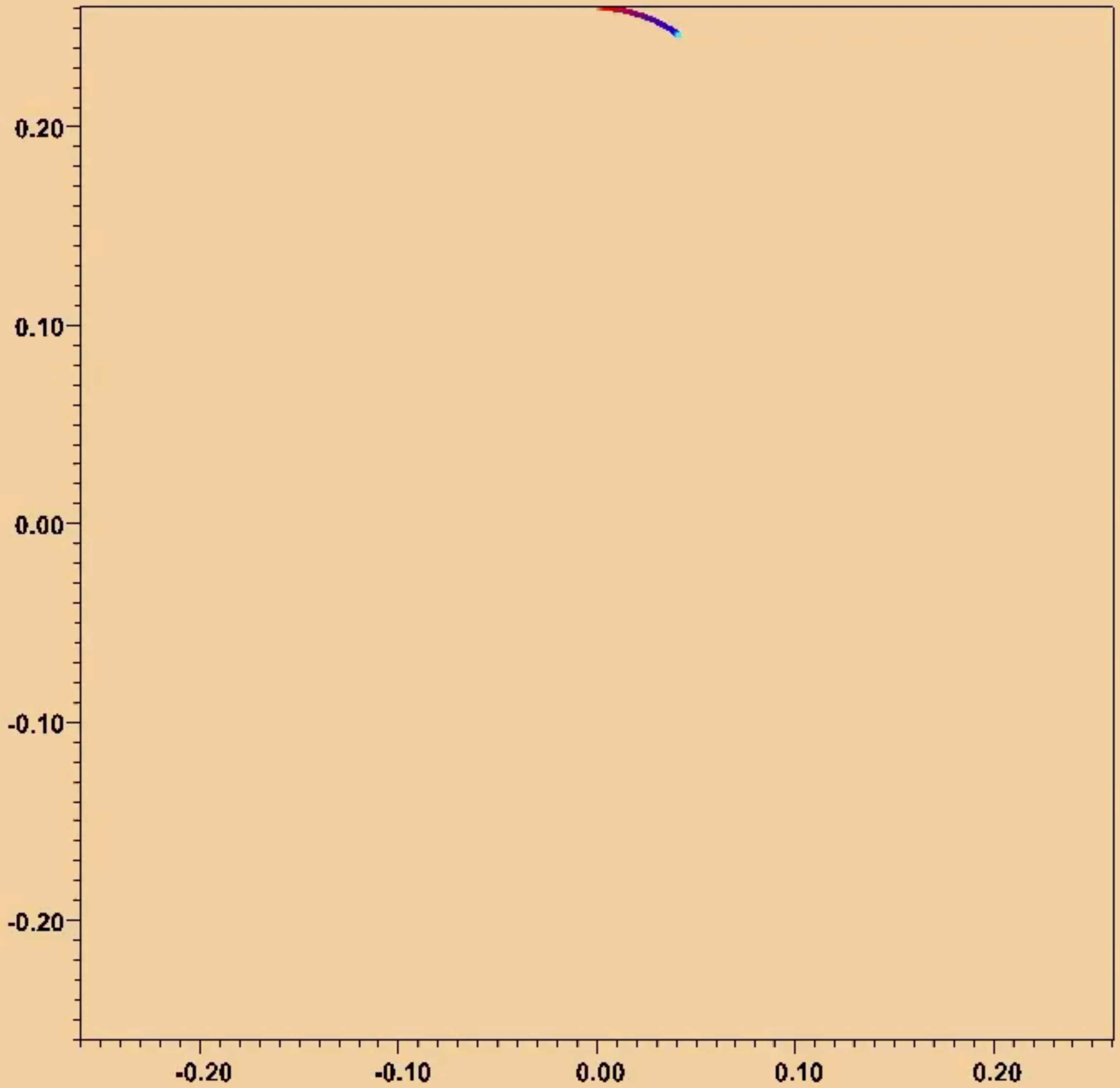
Visualization of 1 Particle Can Be Interesting: *Simulation of Ion Path as Energy Decreases*

Trajectory starts as betatron.
Transitions to Figure 8.
Finally becomes cyclotron.

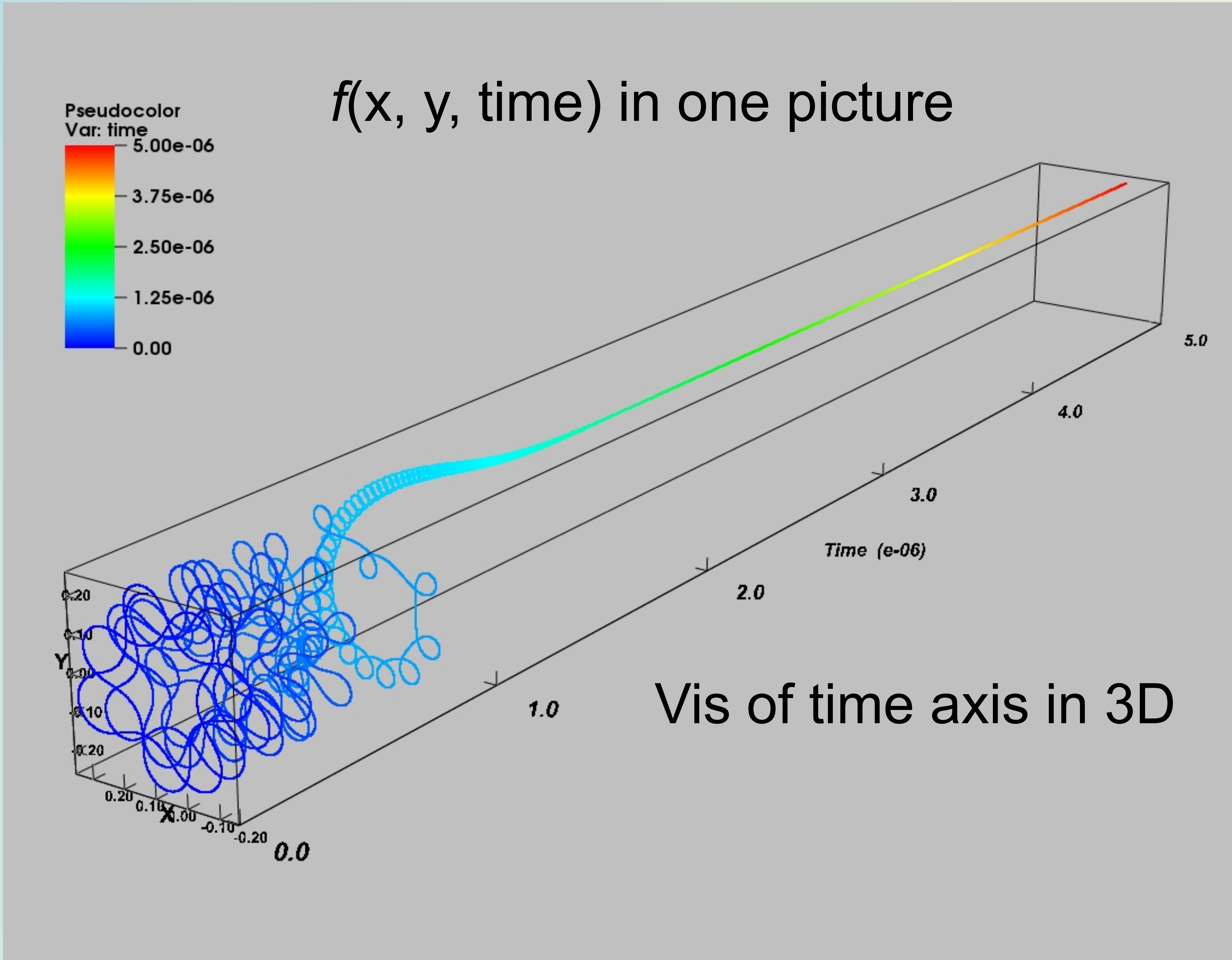


http://w3.pppl.gov/~efeibush/movies/m3_720.mov

Time 8.240 e-10



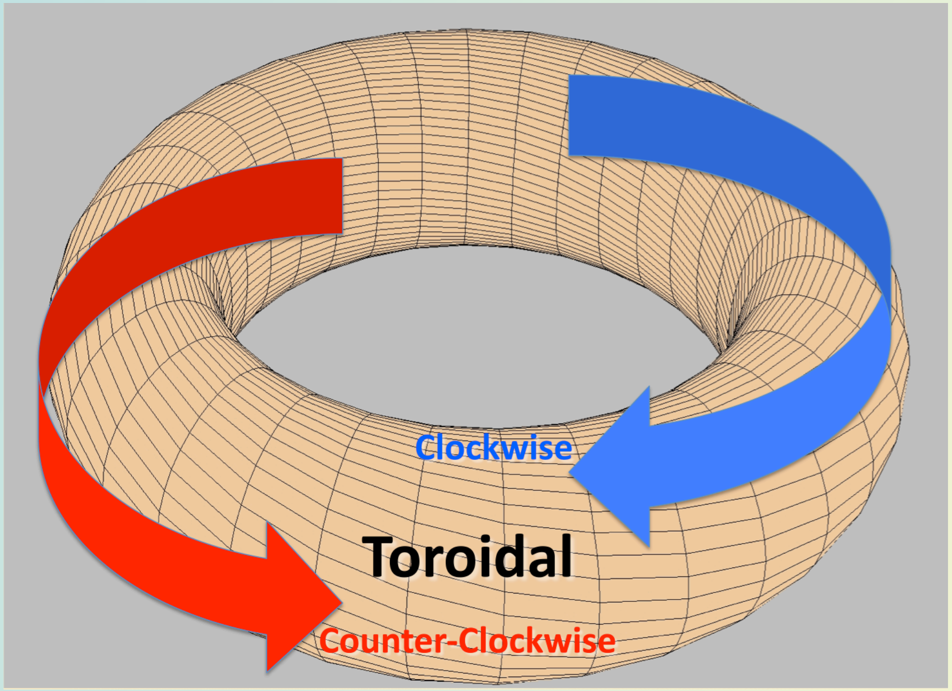
Visualization in 1 Picture

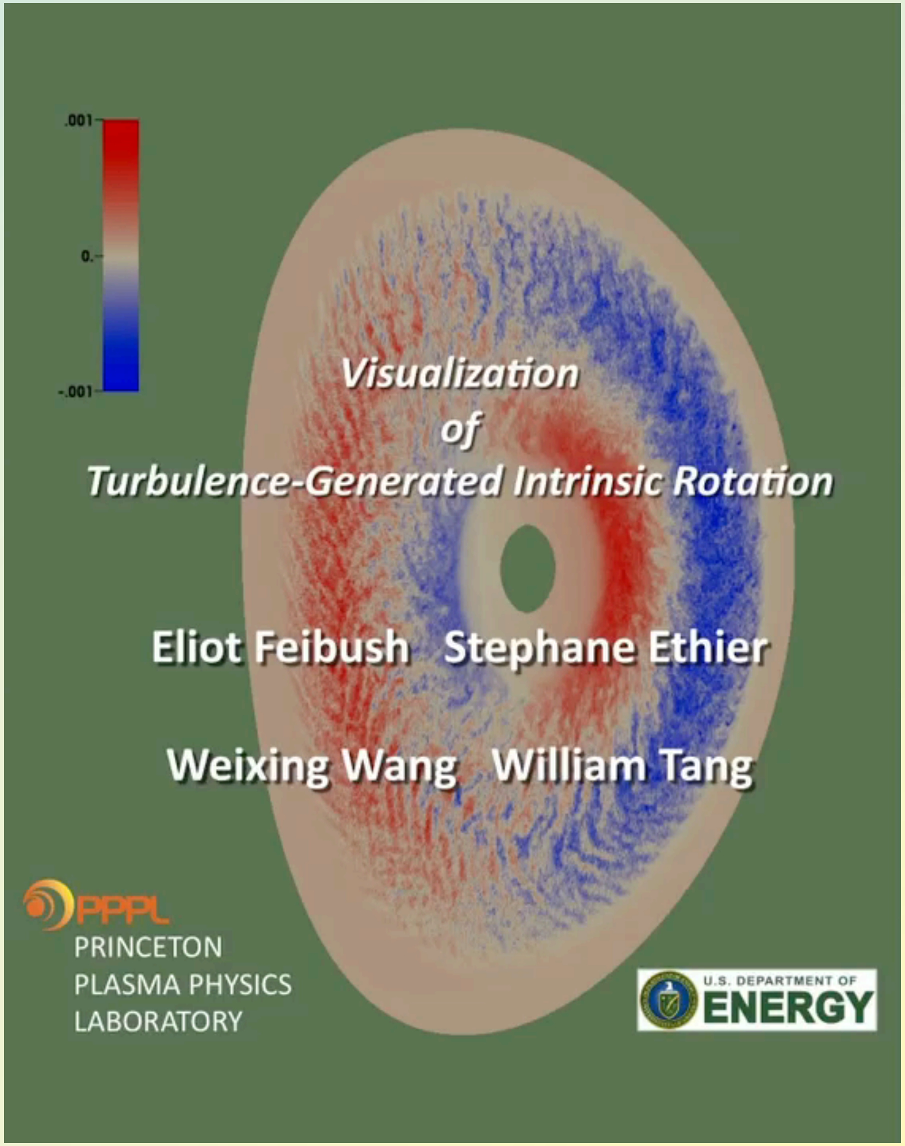


Time Step Simulation

GTS

Plasma Flow around torus.





Plasma Flow through 1 poloidal plane

http://w3.pppl.gov/~efeibush/gts/intrinsicvphi720.mov

Time Step Simulation

XGC

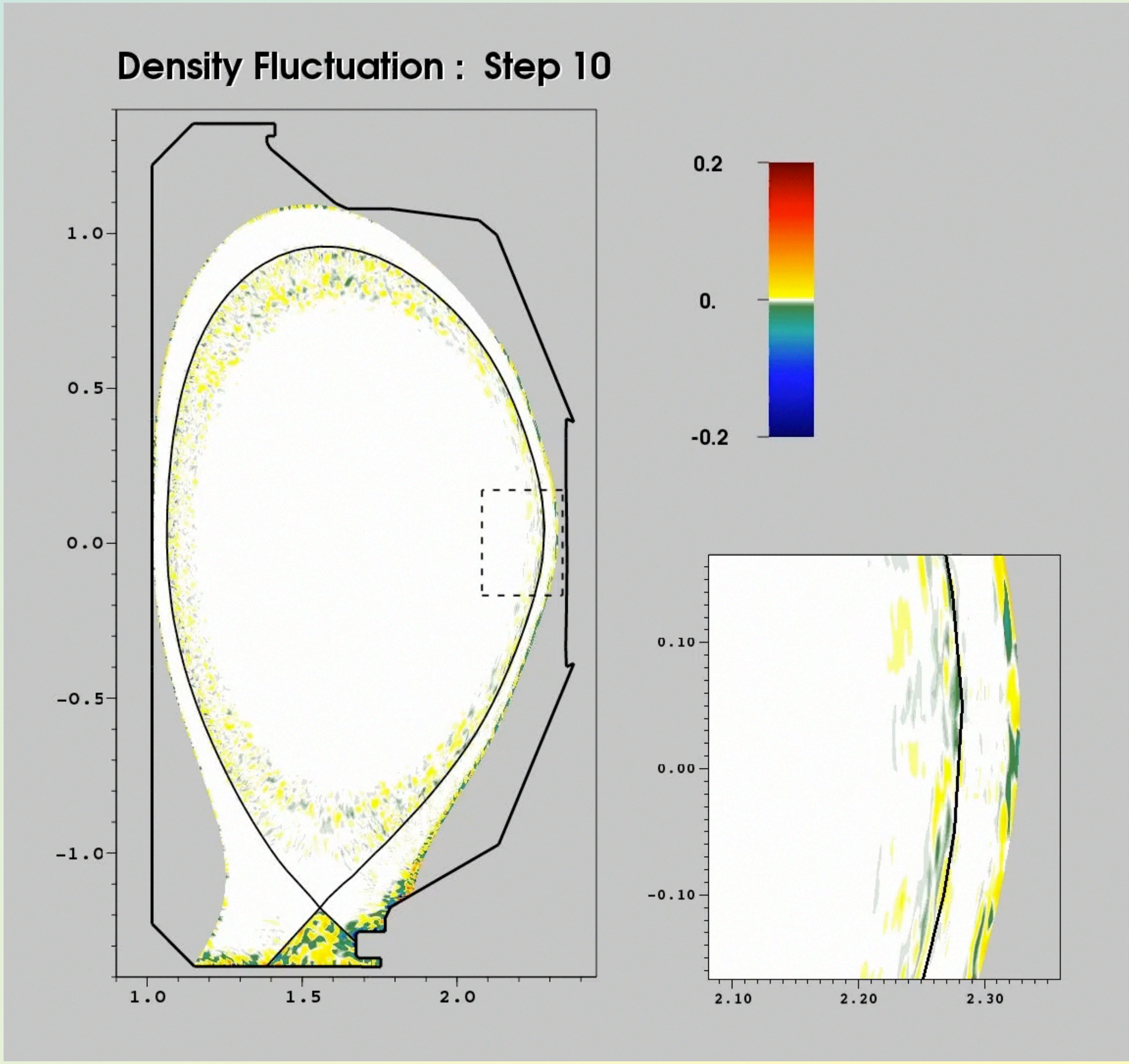
Render overview

+

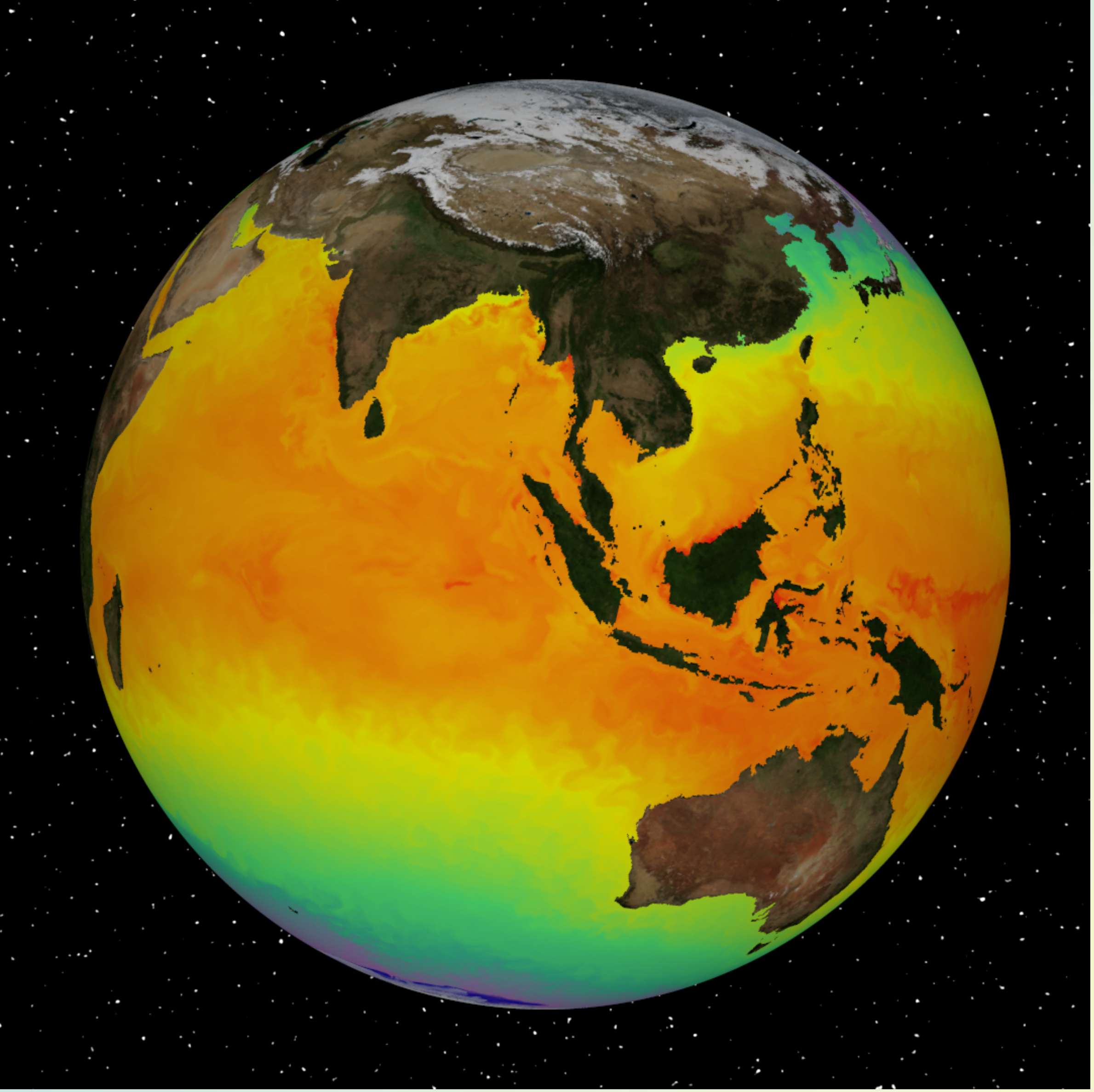
Region of Interest.

Merge images.

Custom color map.



http://w3.pppl.gov/~efeibush/movies/deninsetb1080.mov



GFDL & AOS

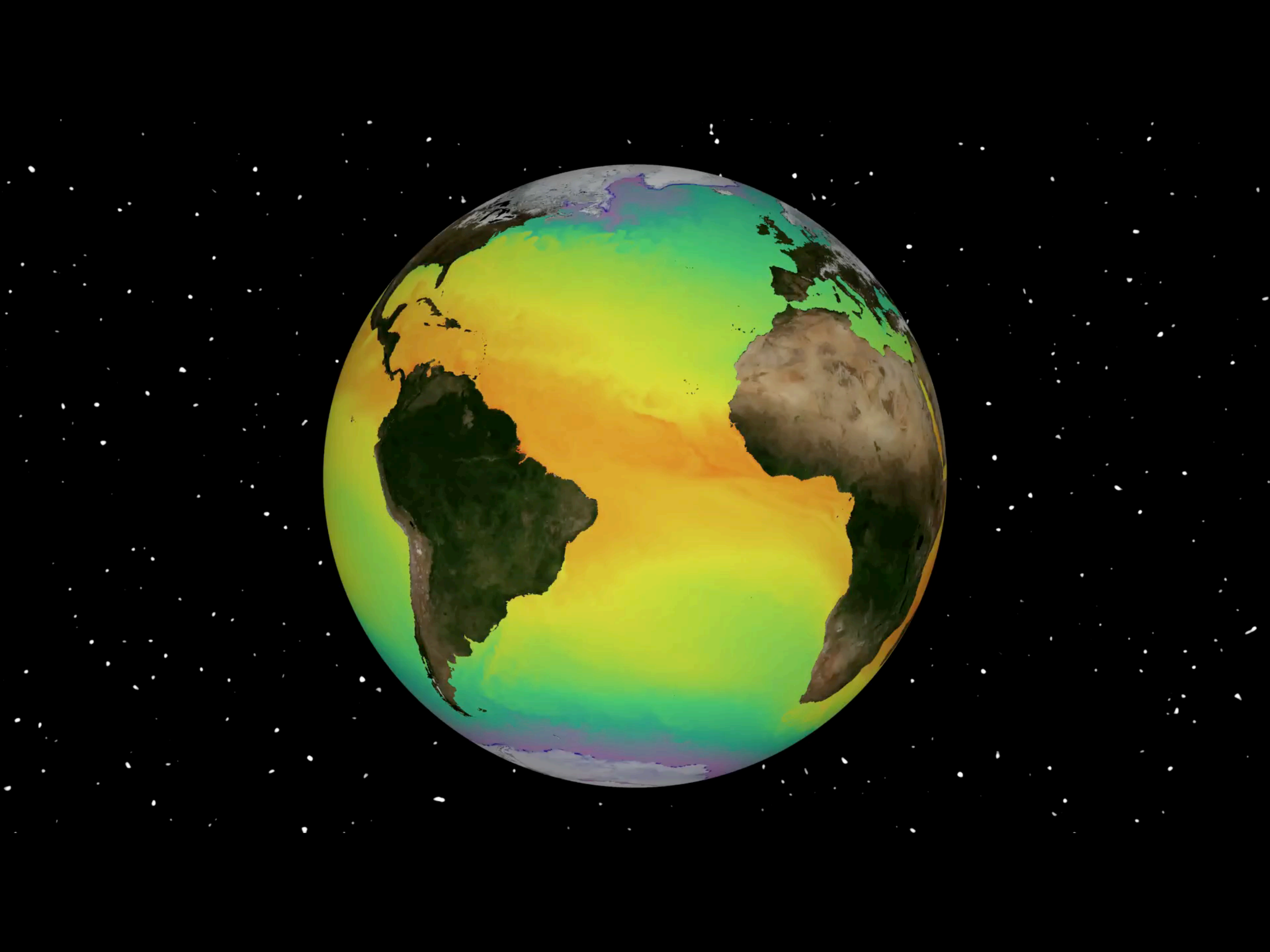
Sea Surface:

Temperature

Salinity

Terrain

http://www.princeton.edu/~efeibush/cm/earth.mov



VisIt Can Read Data Files

- Silo
- Chombo, AMR
- GTC
- M3D, M3DC1
- H5Nimrod
- POINT3D
- S3D
- OpenFOAM
- ITAPS
- XDMF
- Adios
- FLASH
- EnSight
- VTK **VTK is Internal Format**
- NetCDF
- CGNS
- NASTRAN, ANSYS
- TecPlot
- Protein Databank (PDB)
- Plot3D
- GIS (ESRI Shapefile, DEM, many more)
- Image formats

Variable types

- Scalar
- Vector
- Tensor
- Arrays
- Label
- Material
- Species
- X,Y pairs

Database reader plug-ins can be developed for new formats

Getting Data into VisIt

Discrete Point Data

Define and display data at specific points in 3D.

Each point is a unique, independent sample.

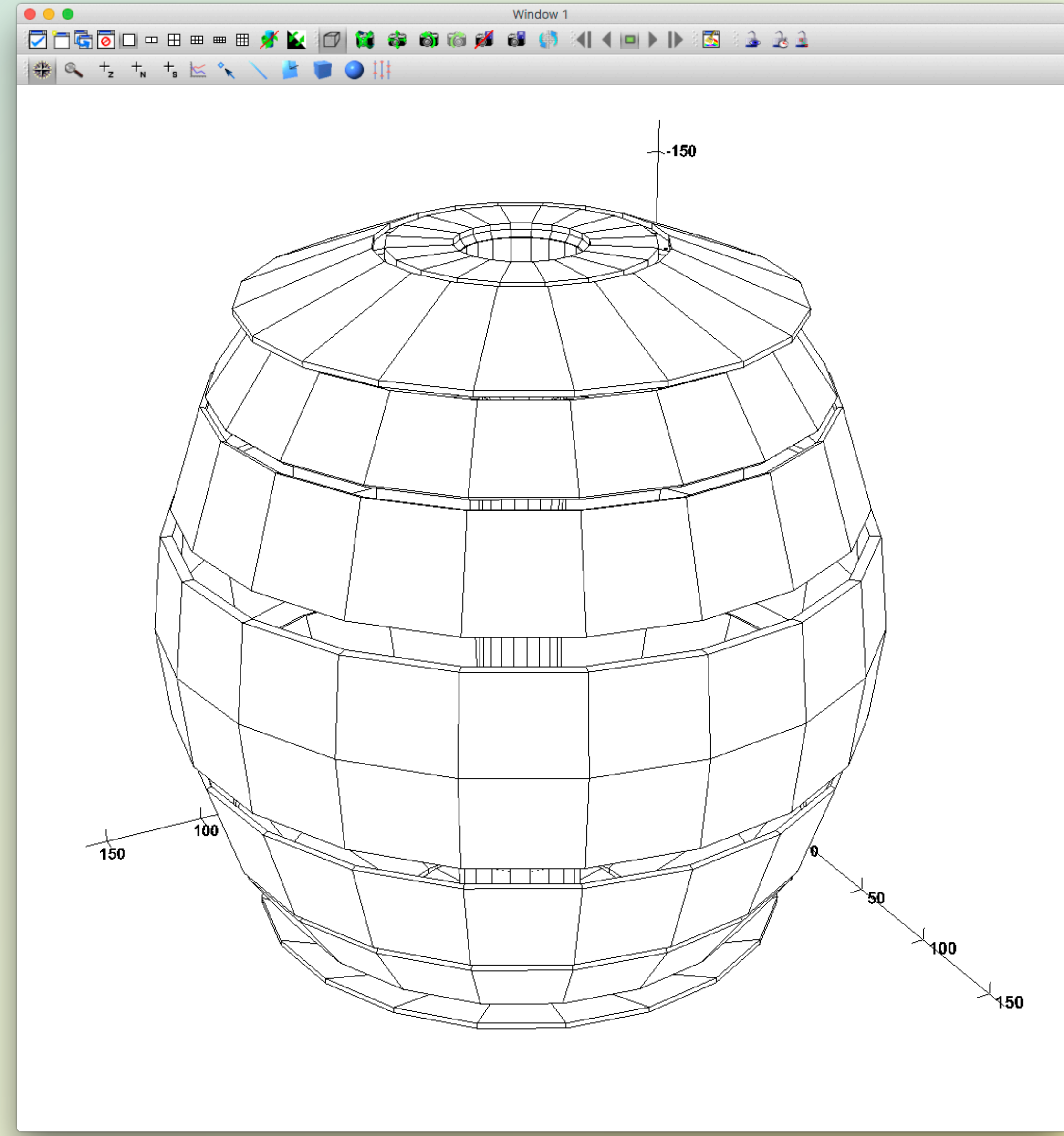
Taken from simulation grid (perhaps).

Look at Point3D data file:

```
x y z density
2.5  0.5  -0.1  .003
...
```

Connecting Points → Polygons

```
DATASET POLYDATA
POINTS 48
90. -140. 100.
80. -140. 100.
...
POLYGONS 24
4 0 1 24 25
4 1 2 25 26
...
```



Structured 3D Grids

3D volume of data – sampled at grid points

f(x,y,z)

VisIt interpolates among grid points in all 3 directions for continous display.

Specify data at grid locations.

Apply Operators to explore & examine data.

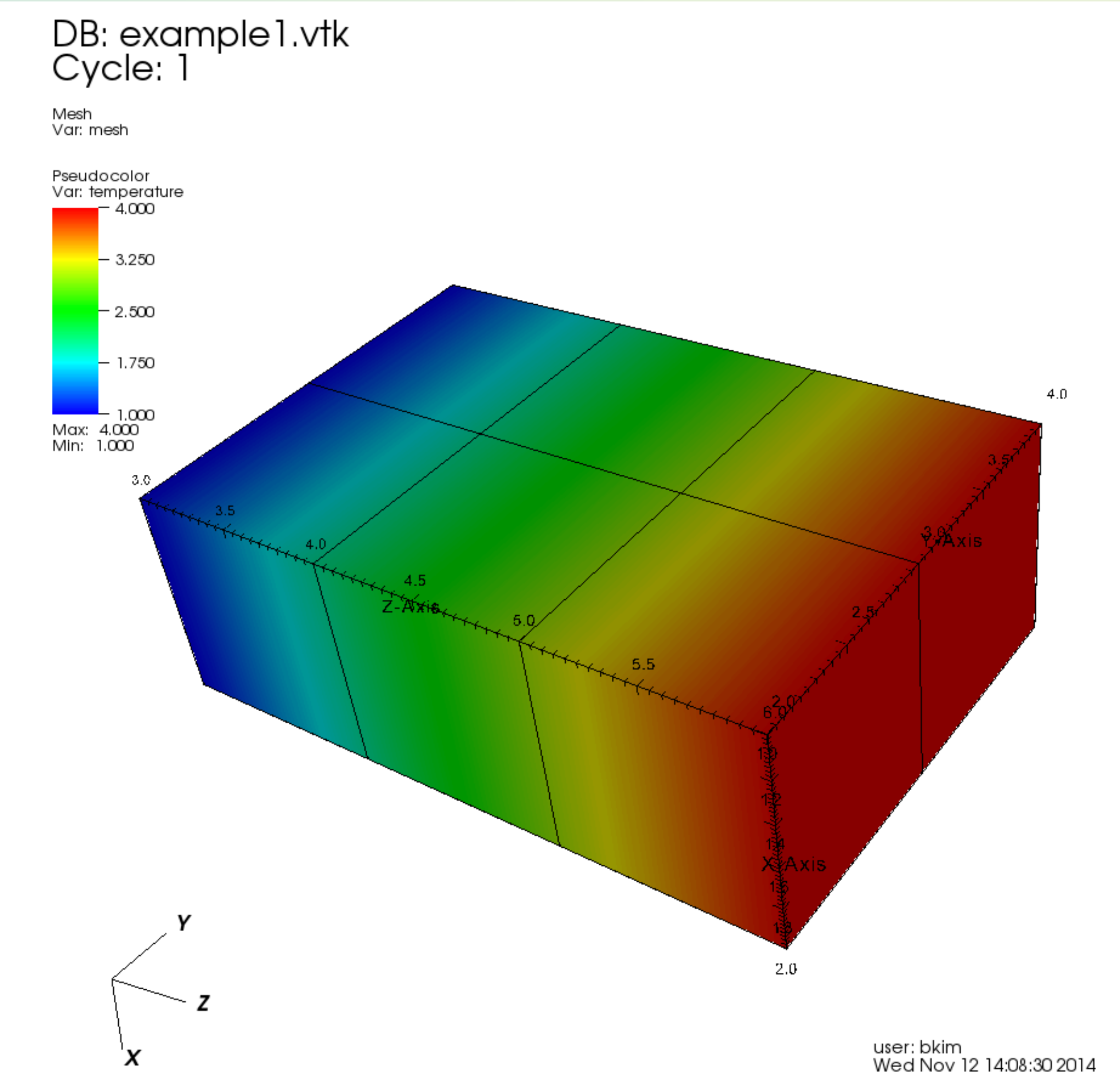
Structured Points *f(x,y,z)*

```
# vtk DataFile Version 3.0
VTK format
ASCII
DATASET STRUCTURED_POINTS
DIMENSIONS 2 3 4
ORIGIN 1. 2. 3.
SPACING 1. 1. 1.
POINT_DATA 24
SCALARS temperature int
LOOKUP_TABLE default
1
1
1
1
1
1
2
2
2
2
2
2
3
3
3
3
3
3
4
4
```

Uniform spacing per axis.

Value at each point.

Interpolates into continuous volume of data.

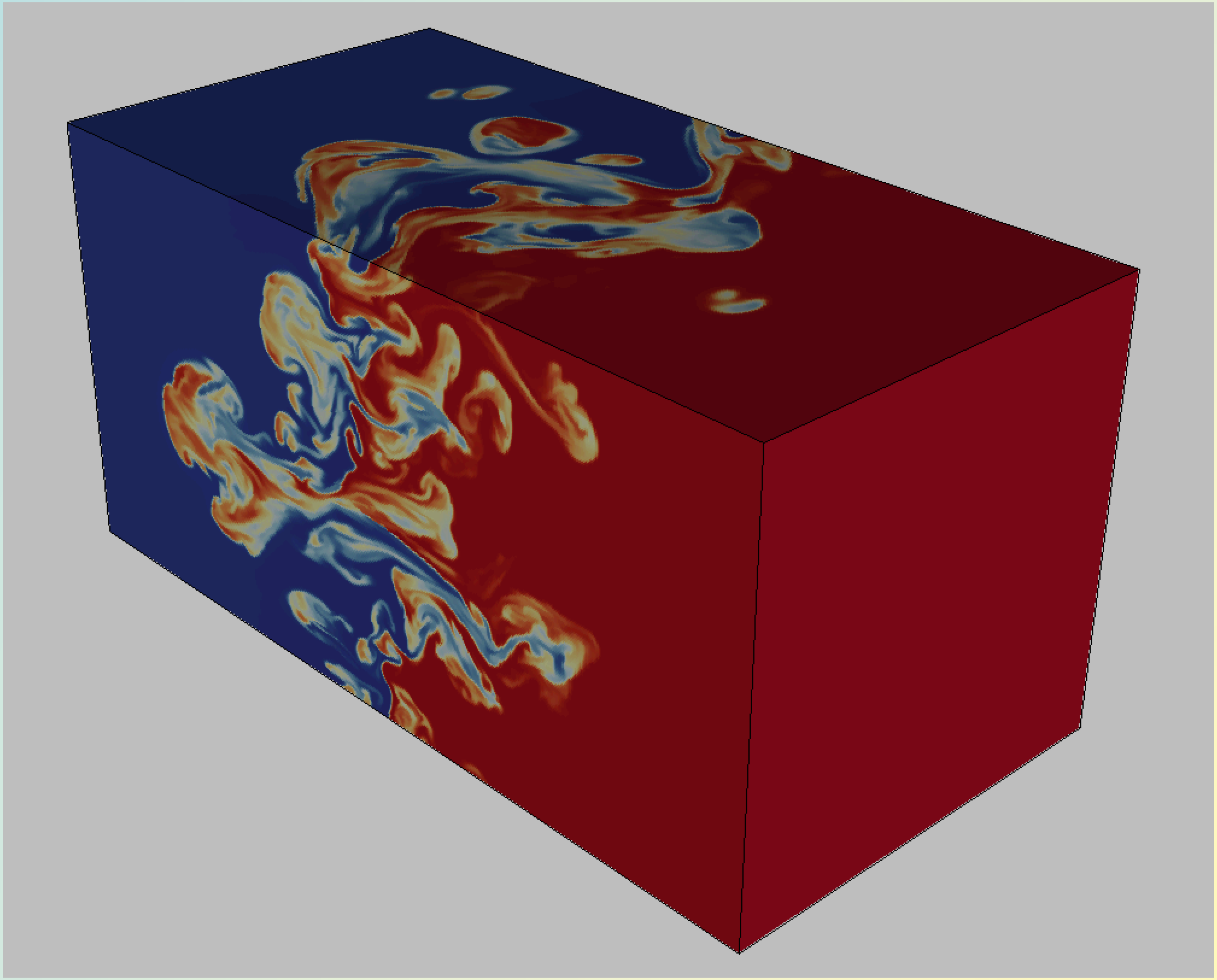


Structured Points Ordering

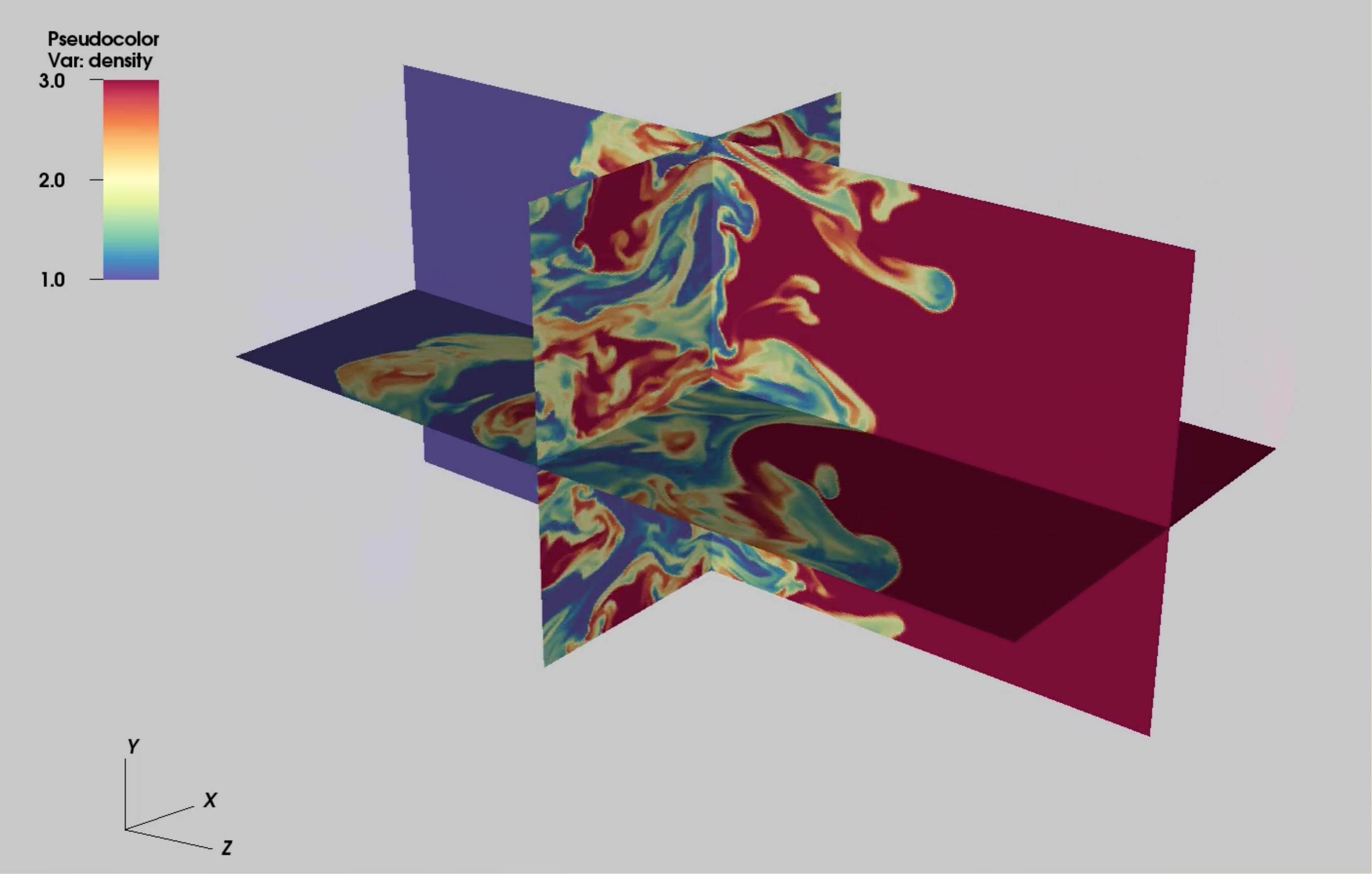
```
# Example python loop to write values to vtk file

for z in range(4):
    for y in range(3):
        for x in range(2):
            # write f(x,y,z) value to file
```

VTK - text or binary (byte representation of ASCII file)
(large text files are OK)

Structured Points $f(x,y,z)$



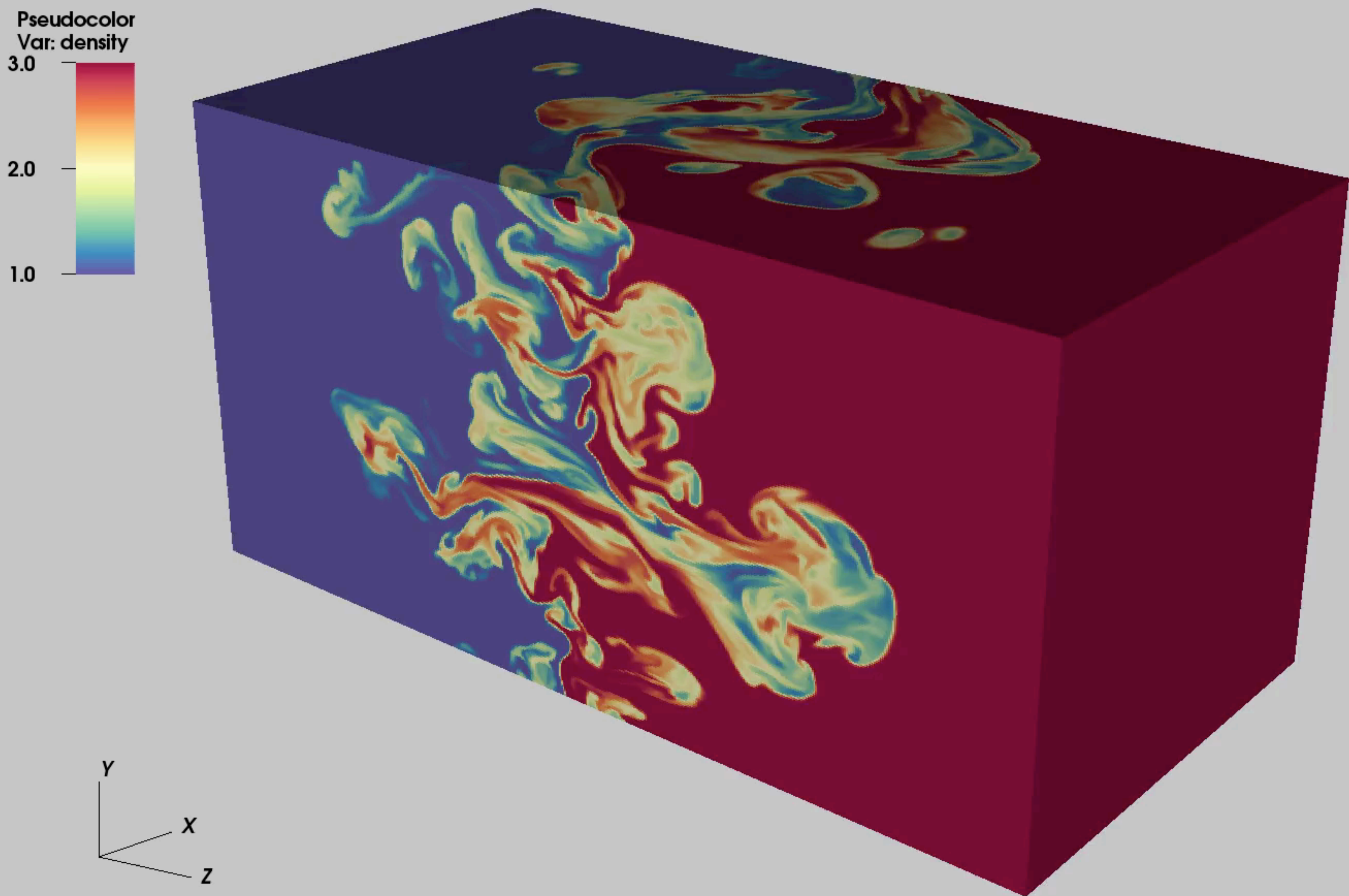
Continuous Volume of Data: Slicable

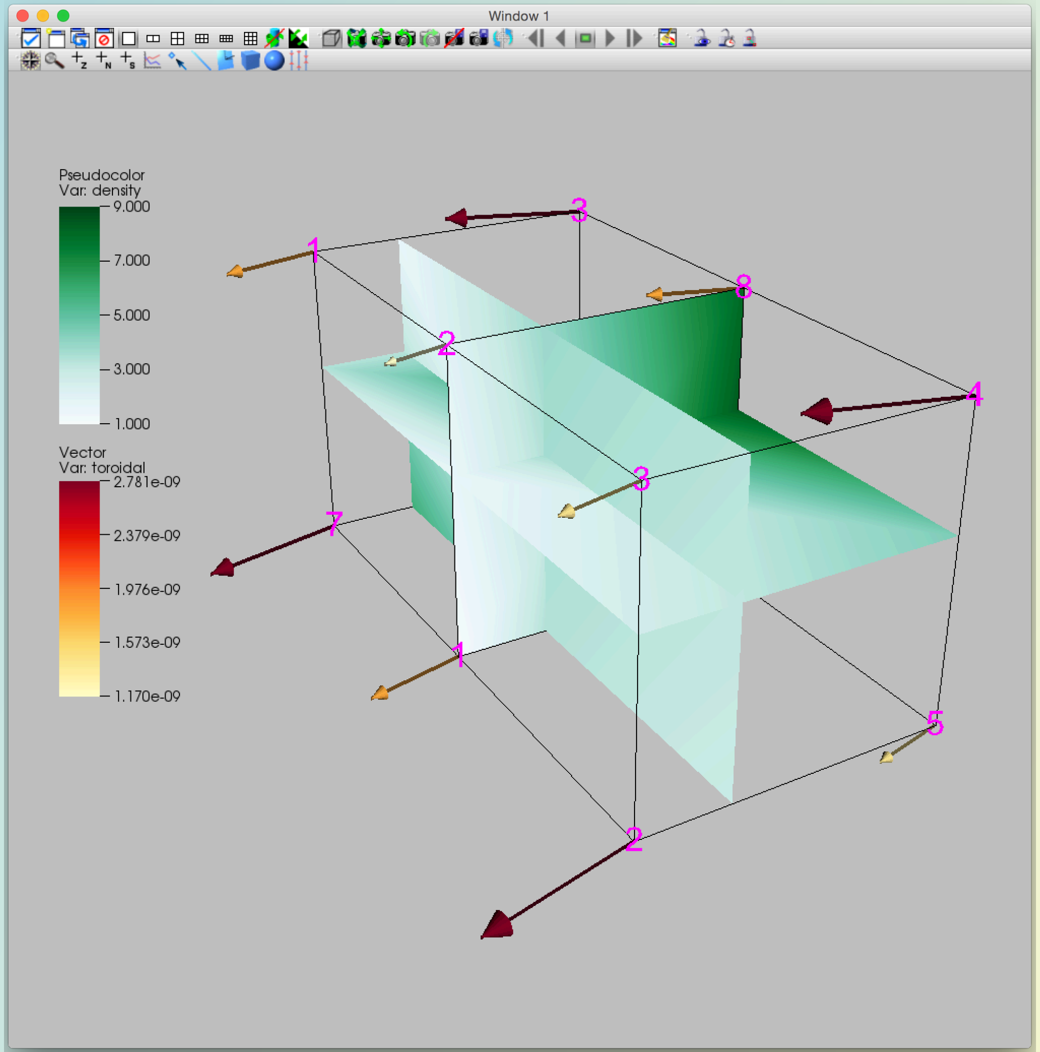


Geometric Operator: 3-Slice

Examine data throughout the volume …

http://www.princeton.edu/~efeibush/movies/rt.mp4


Pseudocolor
Var: density
3.0
2.0
1.0
Y
X
Z

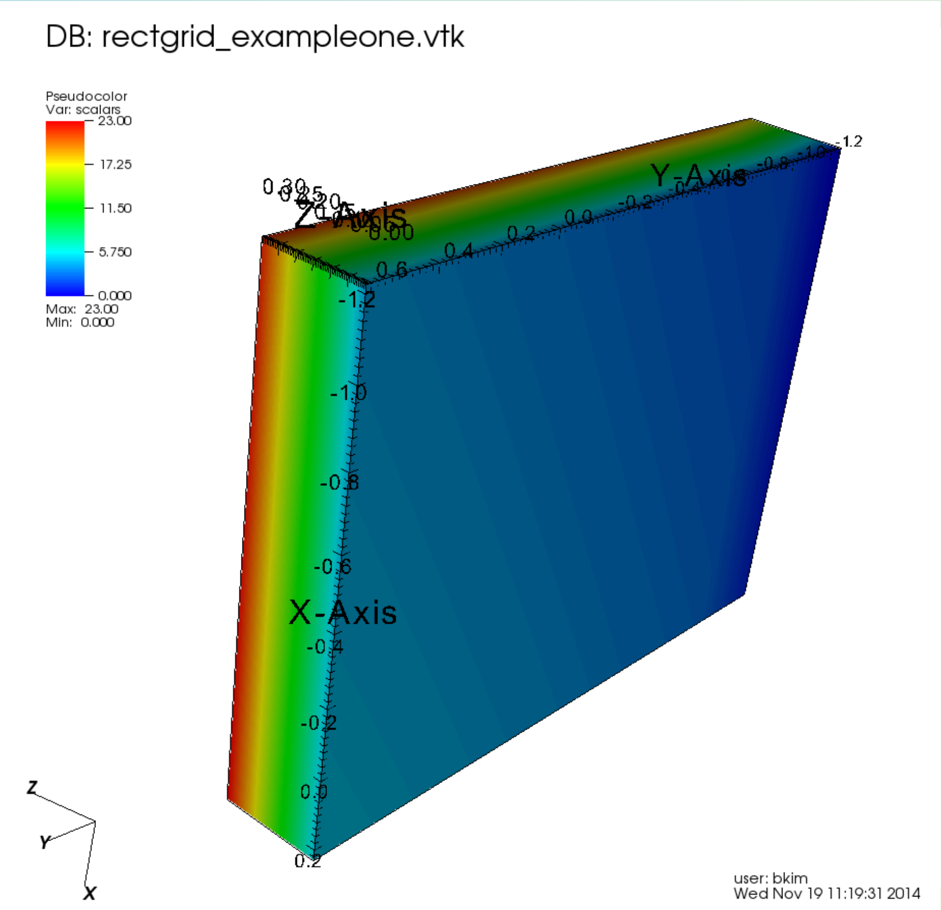


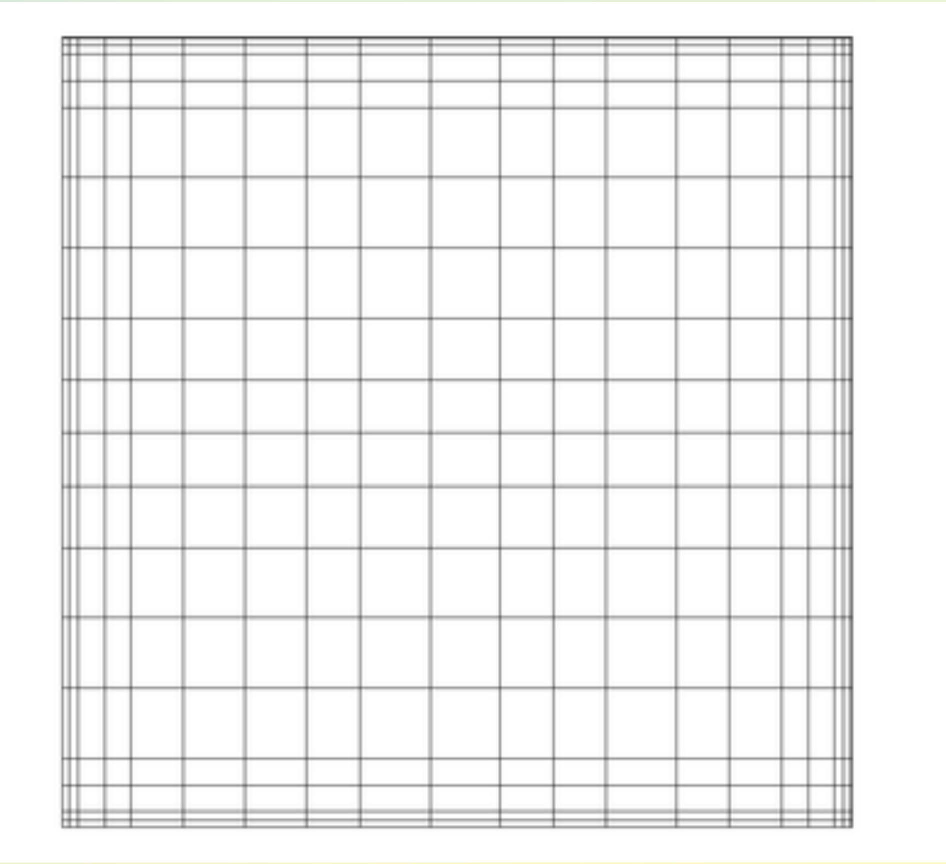
Different color maps for different variables.

- Scalars
- Vectors

Density labels at grid points.

Rectilinear Grid

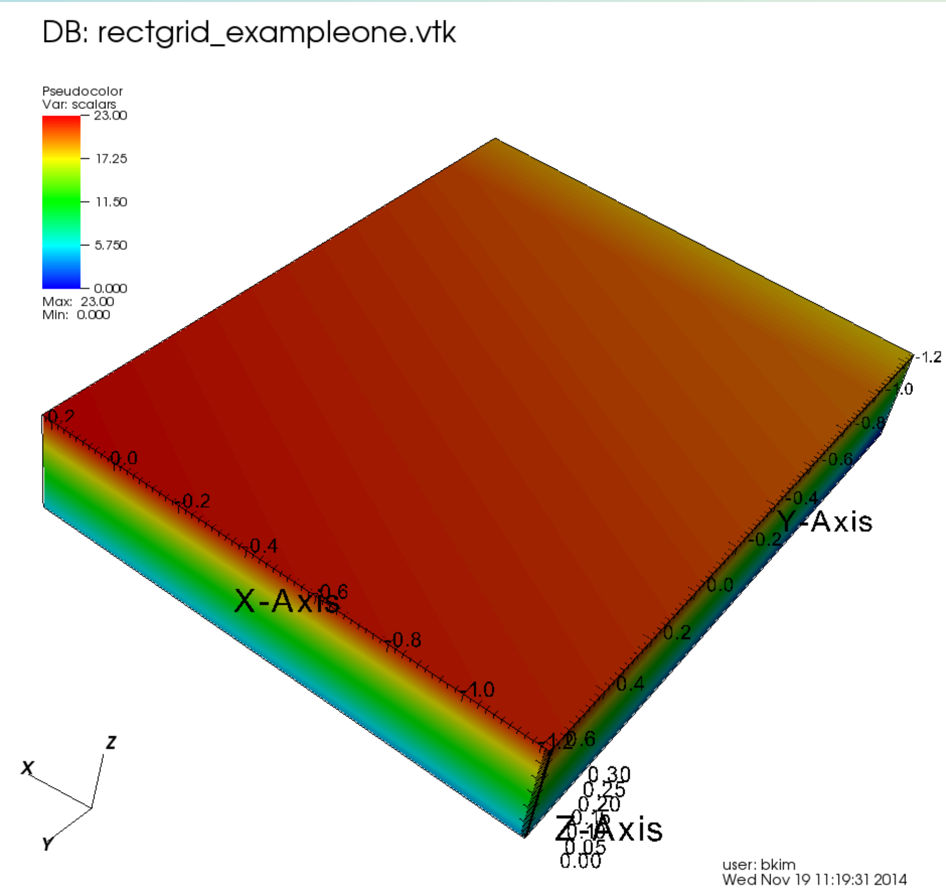




Continuous volume of data defined at specific points.

Non-Uniform spacing per axis.

Rectilinear Grid

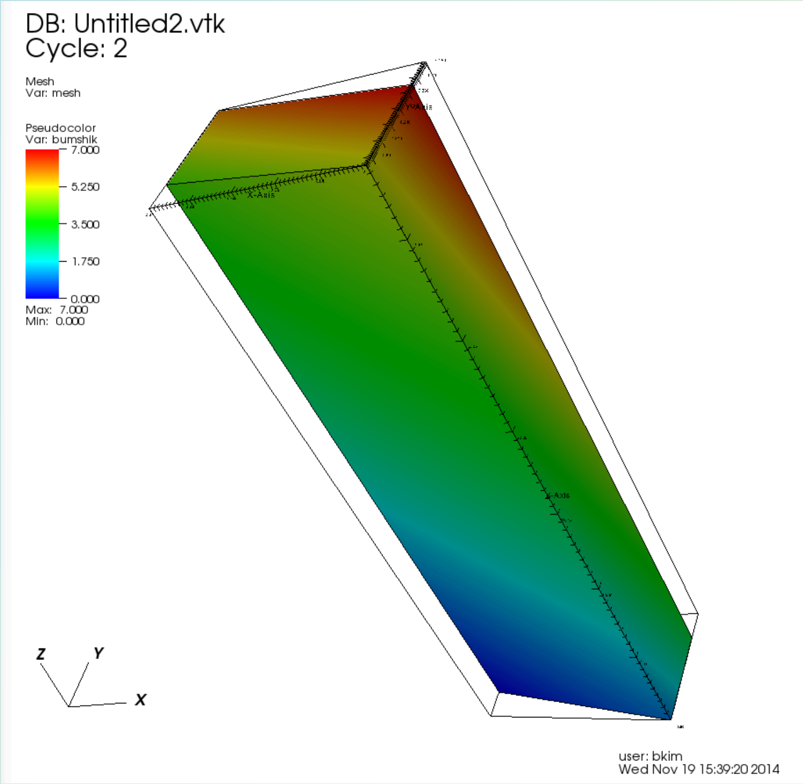


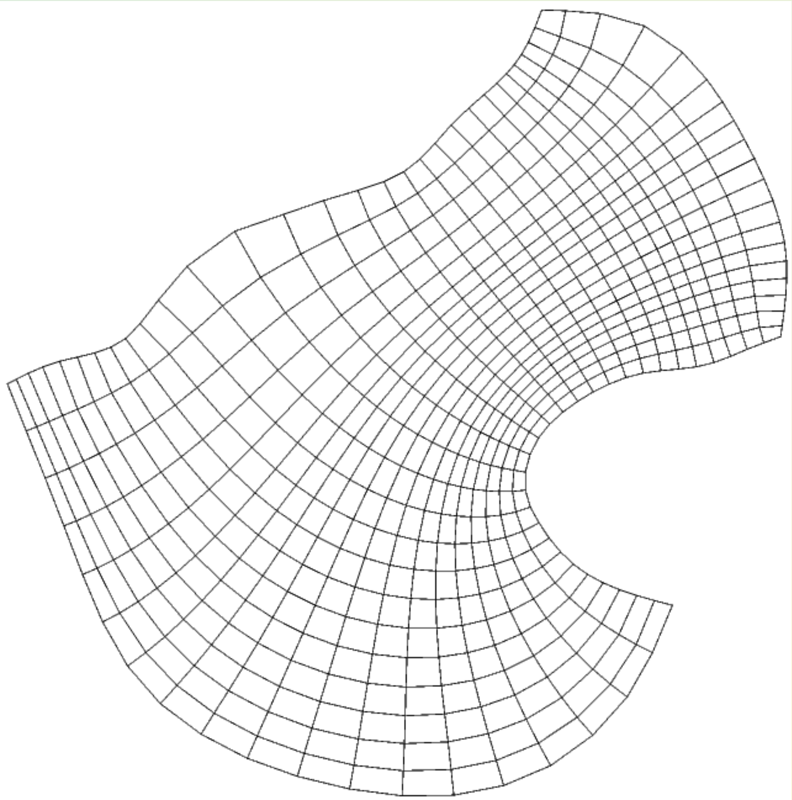
rectgrid_exampleone

```
# vtk DataFile Version 3.0
VTK format
ASCII
DATASET RECTILINEAR_GRID
DIMENSIONS 2 3 4
X_COORDINATES 2 float
-1.22 0.23
Y_COORDINATES 3 float
-1.25 -1.01 0.6125
Z_COORDINATES 4 float
0 0.1 0.2 0.3
POINT_DATA 24
SCALARS scalars float
LOOKUP_TABLE default
0 1 2 3 4 5 6 7 8 9 10
11 12 13 14 15 16 17 18 19
20 21 22 23
```

Non-Uniform Axis Spacing

Structured Grid





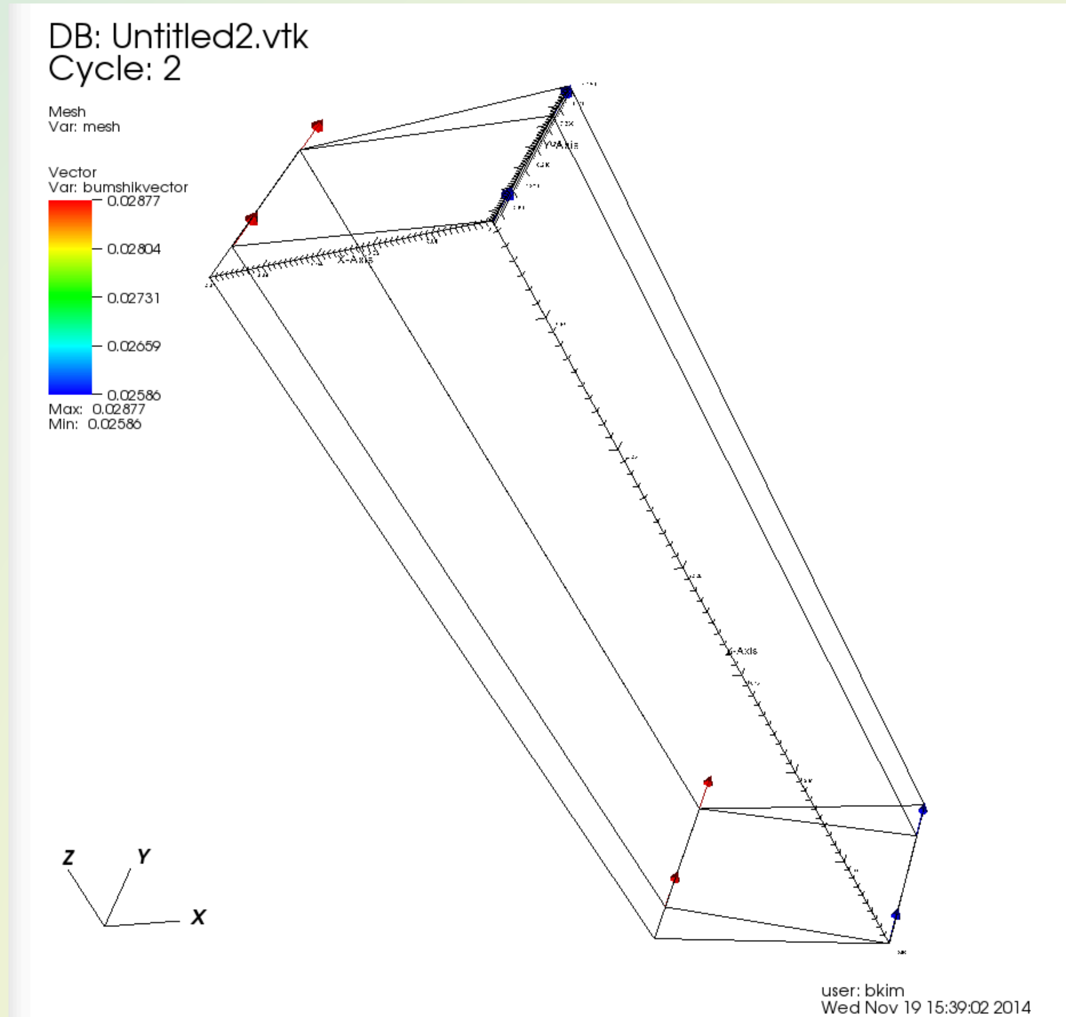
Continuous volume (or surface) of data defined at specific points.

Regular topology, non-orthogonal, specify each point location.
Quadrilateral cell faces.

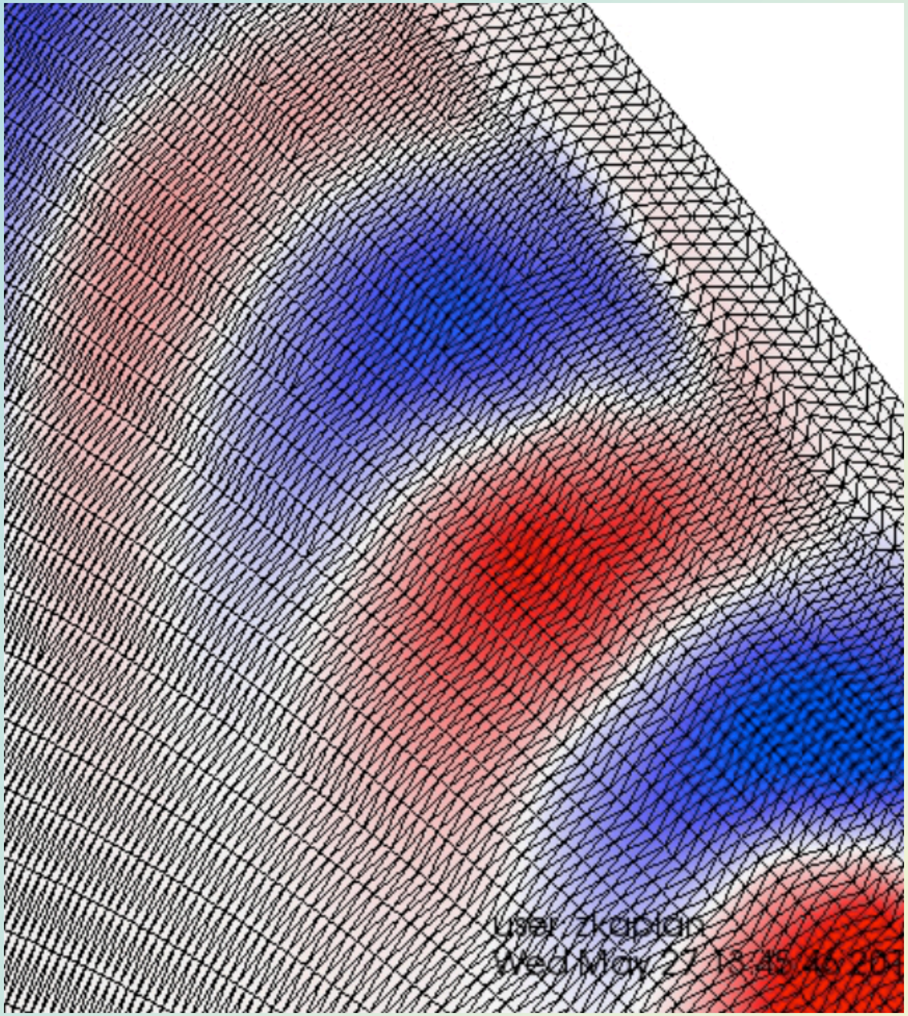
Structured Grids + Vectors

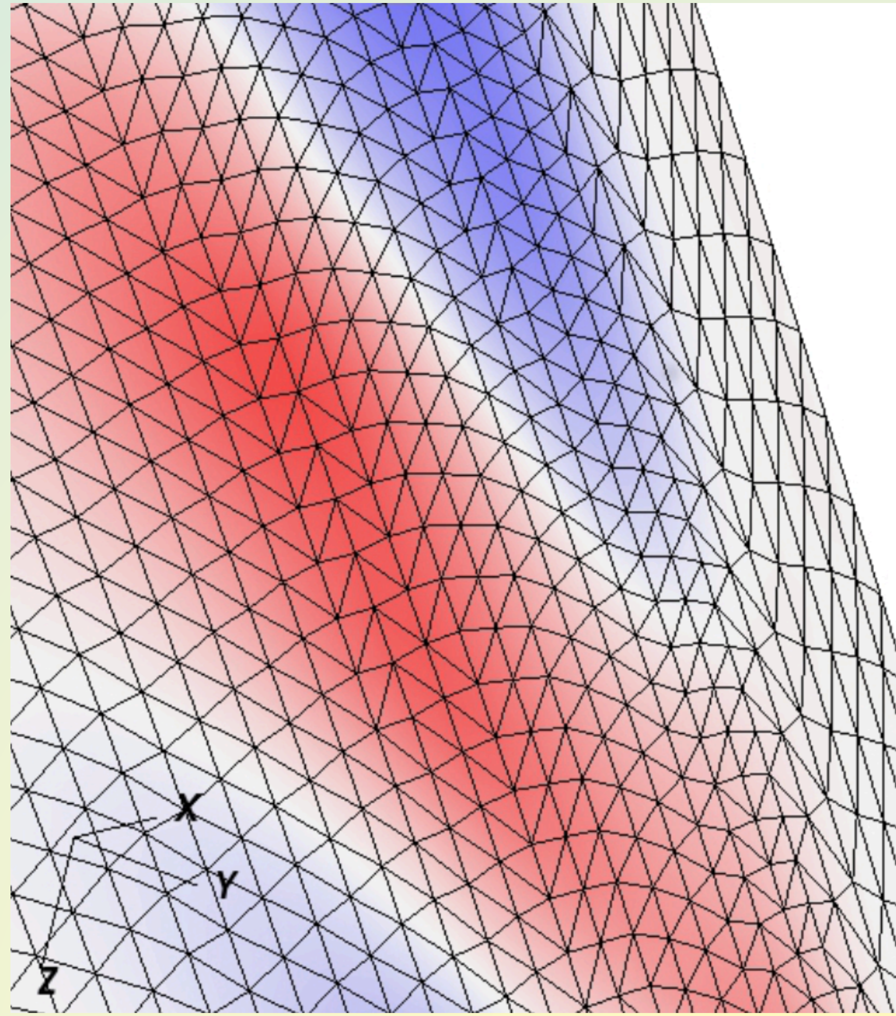
Untitled2.vtk

```
# vtk DataFile Version 3.0
vtk output
ASCII
DATASET STRUCTURED_GRID
DIMENSIONS 2 2 2
POINTS 8 float
0 0.2 0 0.1 0.184843 0 0 0.25 0
0.1 0.234843 0 0 0.2 0.333333 0.1 0.184843 0.333333
0 0.25 0.333333 0.1 0.234843 0.333333

POINT_DATA 8
SCALARS bumshik float
LOOKUP_TABLE default
0 1 2 3 4 5 6 7
VECTORS bumshikvector float
0 0.0287671 0 0 0.0258604 0 0 0.0287671 0
0 0.0258604 0 0 0.0287671 0 0 0.0258604 0
0 0.0287671 0 0 0.0258604 0
```



VTK *Unstructured* Grid

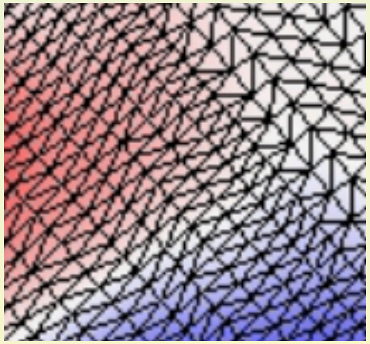




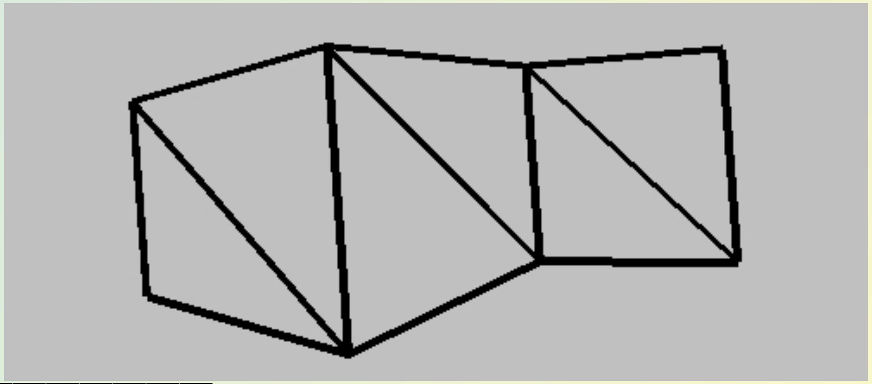
GTS Complex Mesh – Poloidal Rings
Delaunay Triangulation Algorithm

VTK Grid Summary

Structured Points – uniform spacing, orthogonal
Rectilinear Grid – non-uniform spacing, orthogonal
Structured Grid – non-orthogonal quads
Unstructured Grid – any combination of polygons:



Triangle Strip



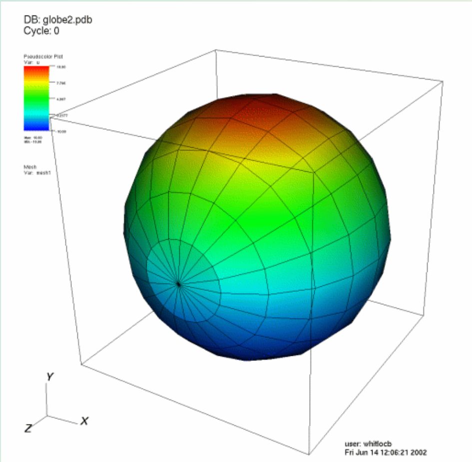
____

Paraview wiki: Users Guide VTK_Data_Model

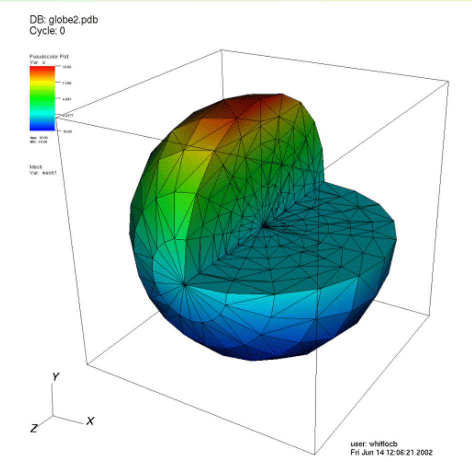
Geometric Selection - *Clip* Operator

- The Clip operator clips 2D or 3D plots against planes or a sphere to remove sections of the plots
- Use this operator when you want to see a cross section of a 3D plot, while still leaving the plot in 3D

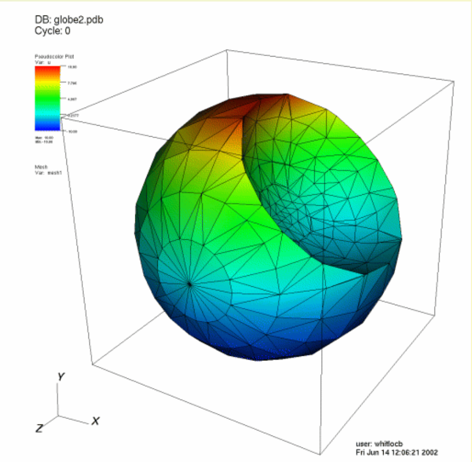
Original plots



Plots clipped with 2 planes



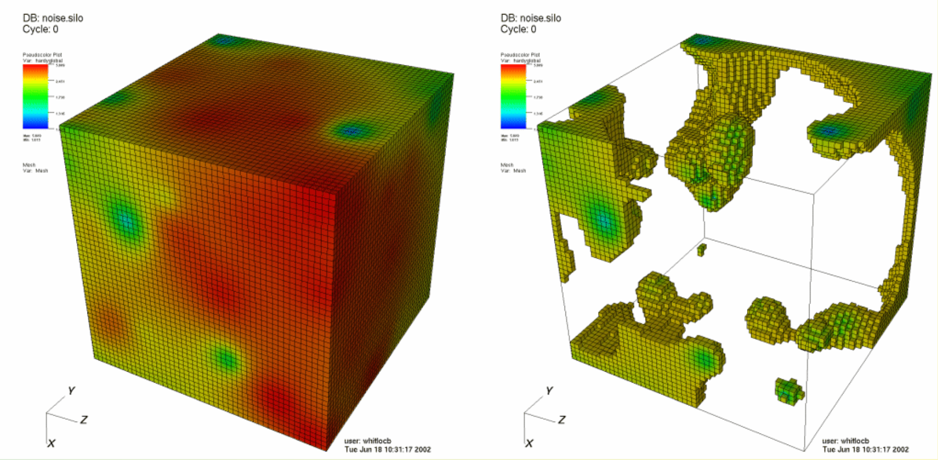
Plots clipped with a sphere



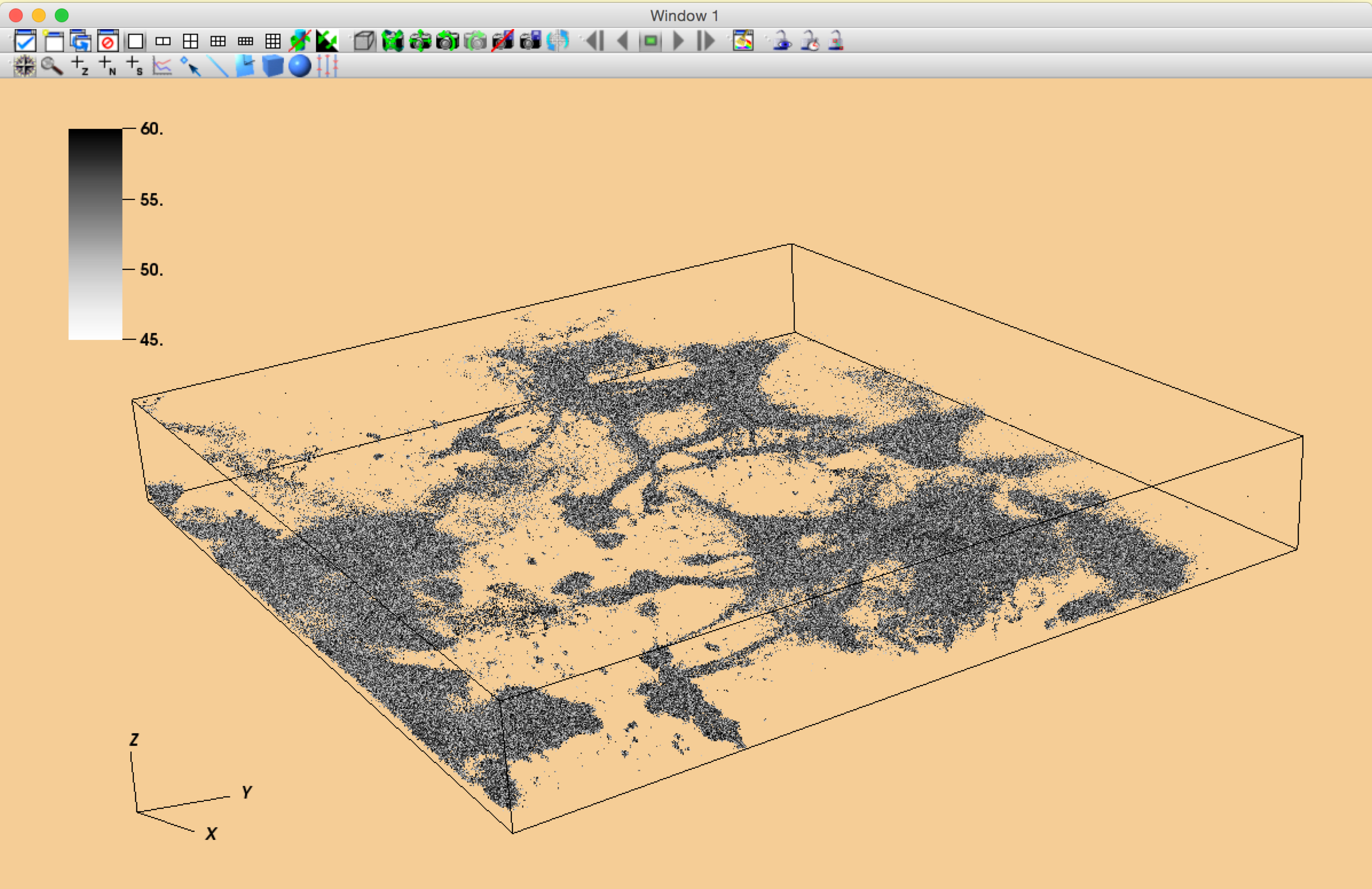
Data Value Selection - *Threshold* Operator

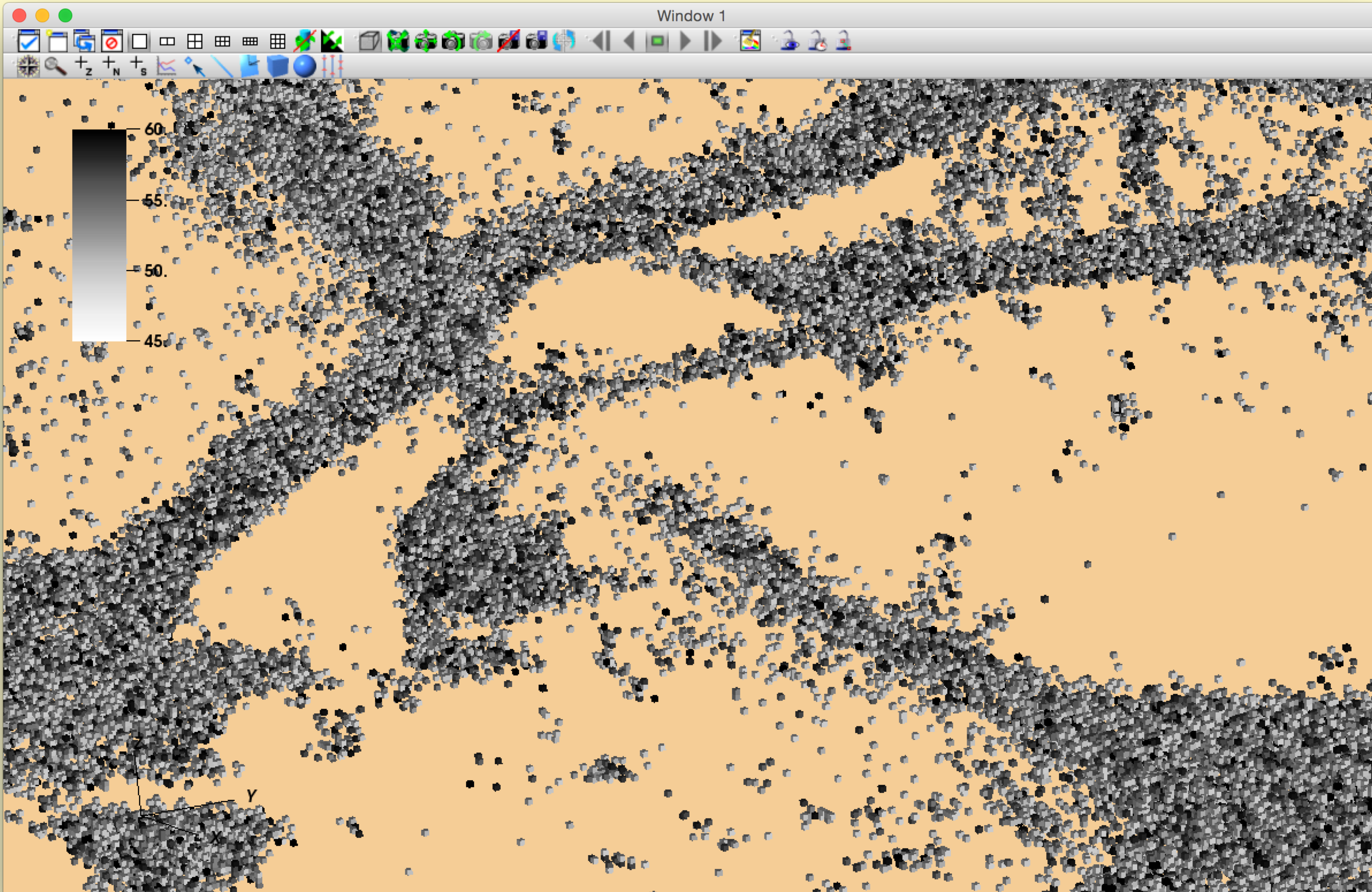
Use this operator to look at cells that have values within a numerical range.

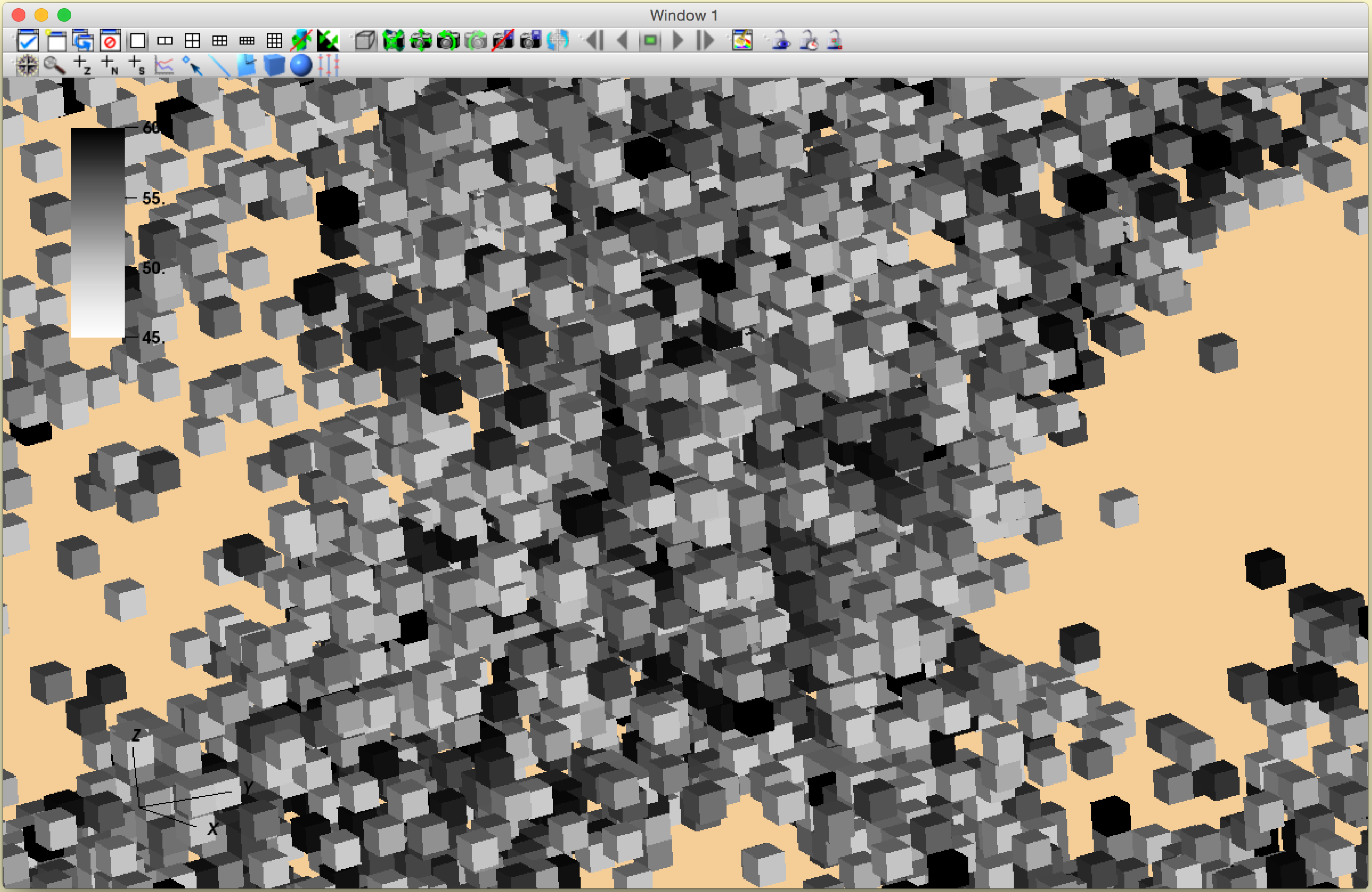
Removes cells whose value is not in the specified range.



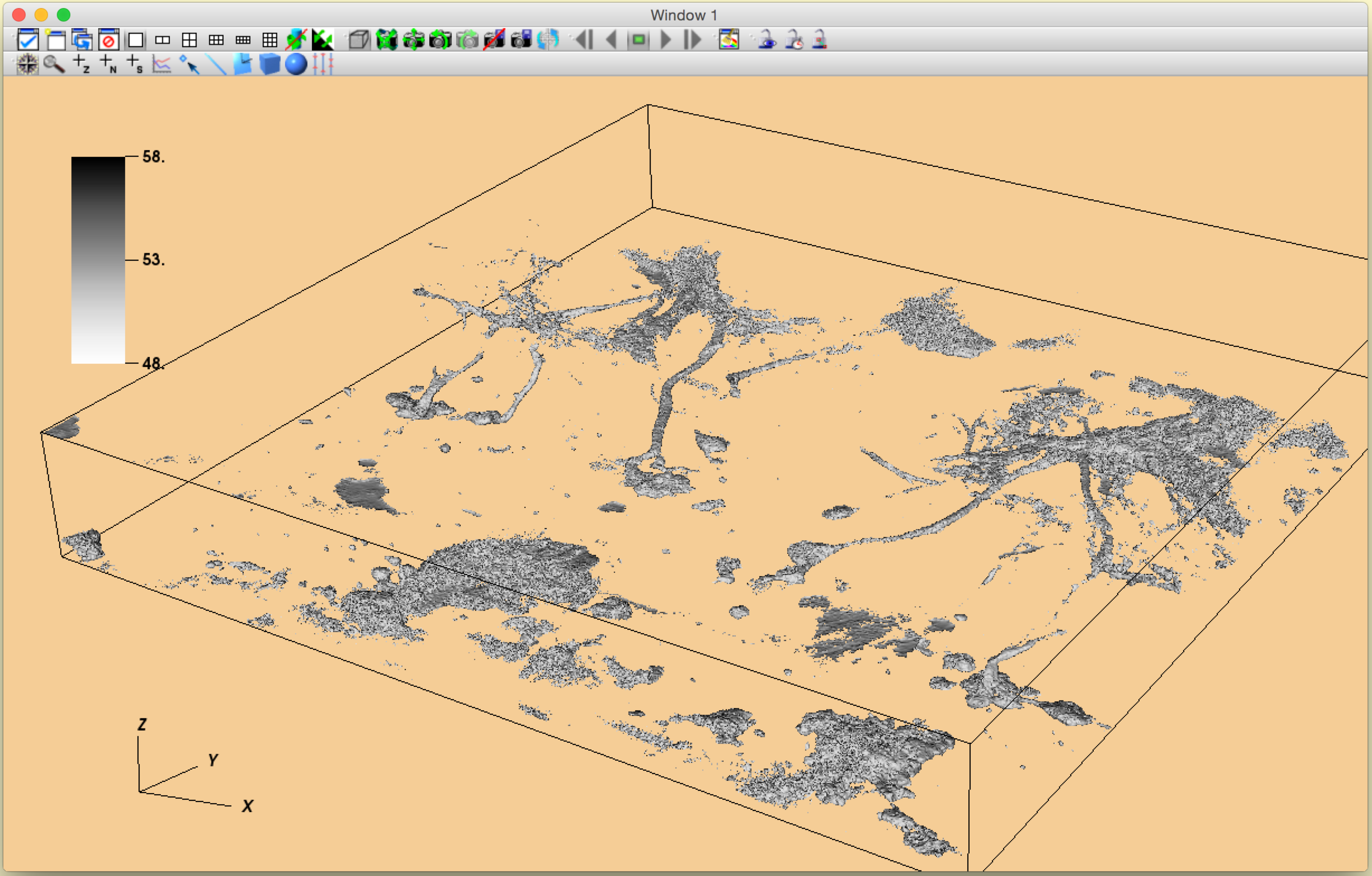
Threshold Operator



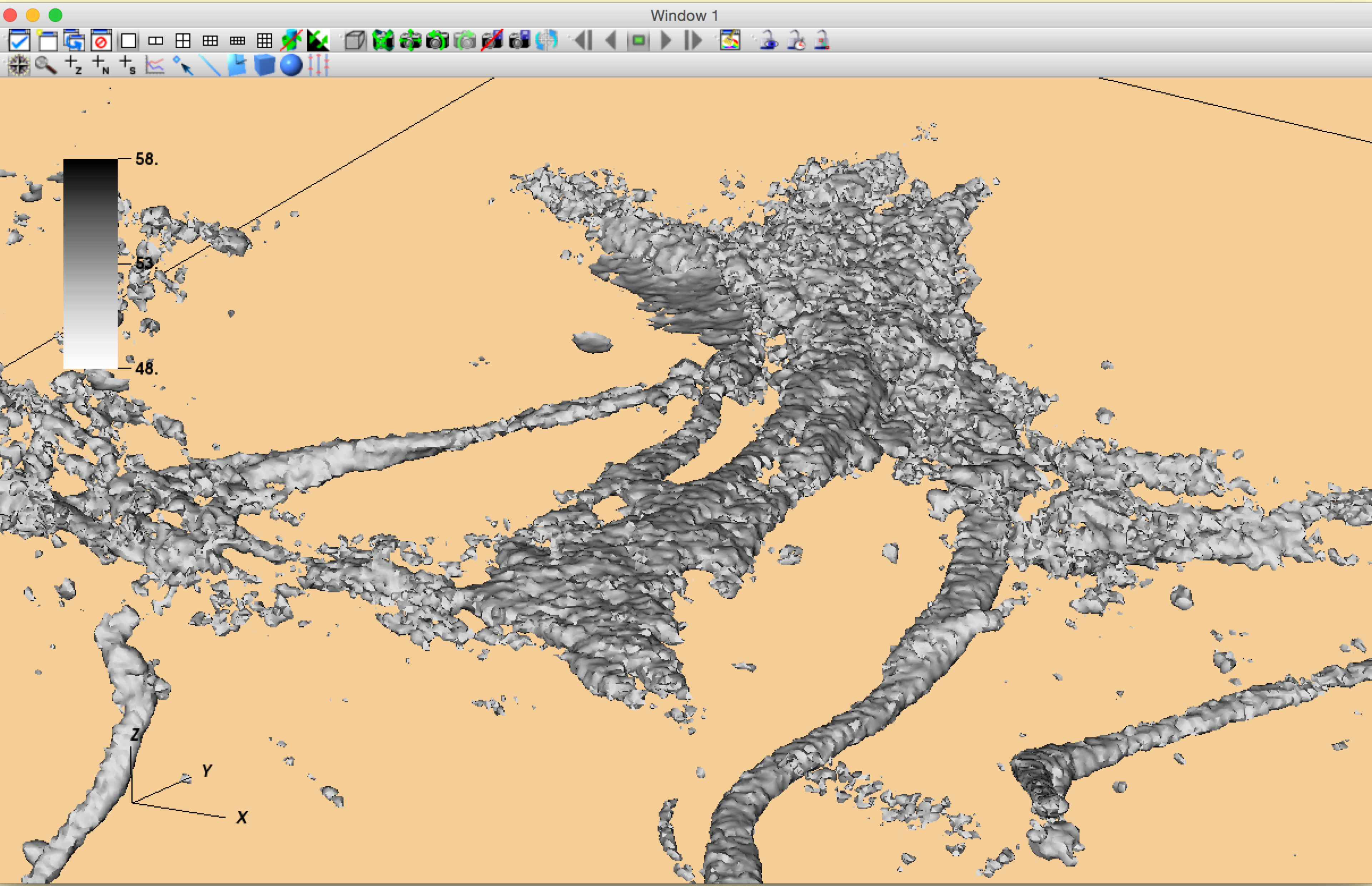

Window 1
60.
55.
50.
45.
Y
X


Window 1
60.
55.
50.
45.
Z
Y
X

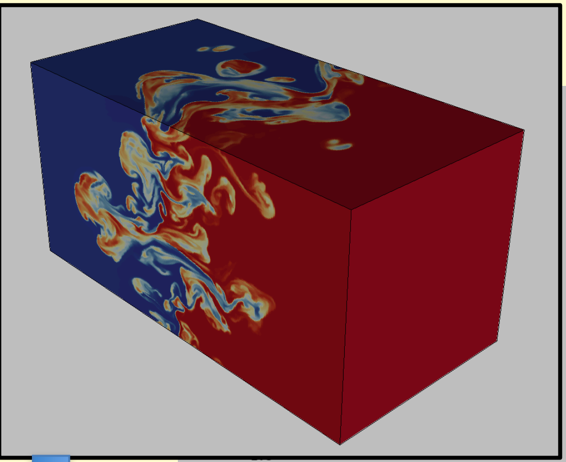
Isovolume Operator

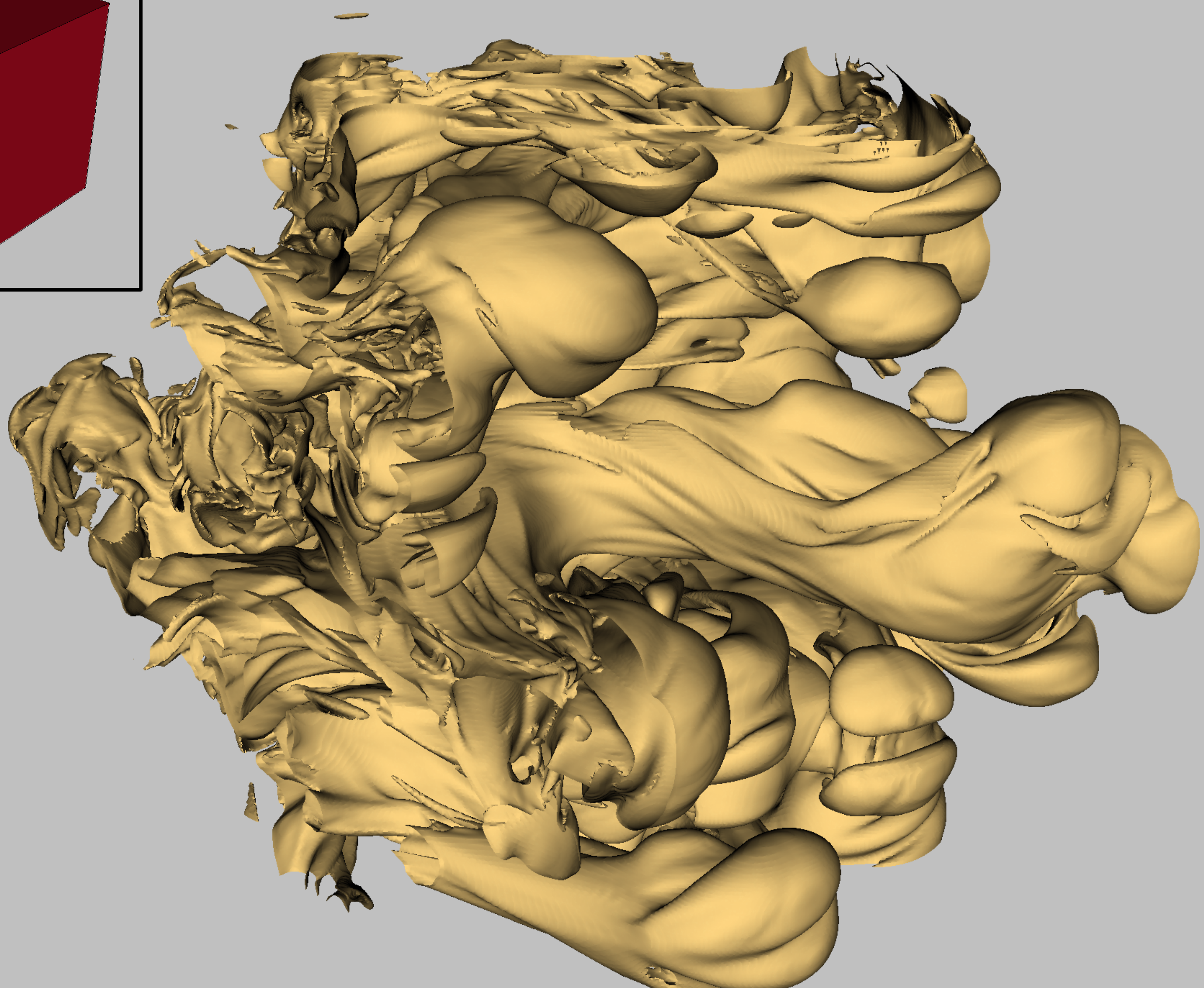


Isosurface Operator

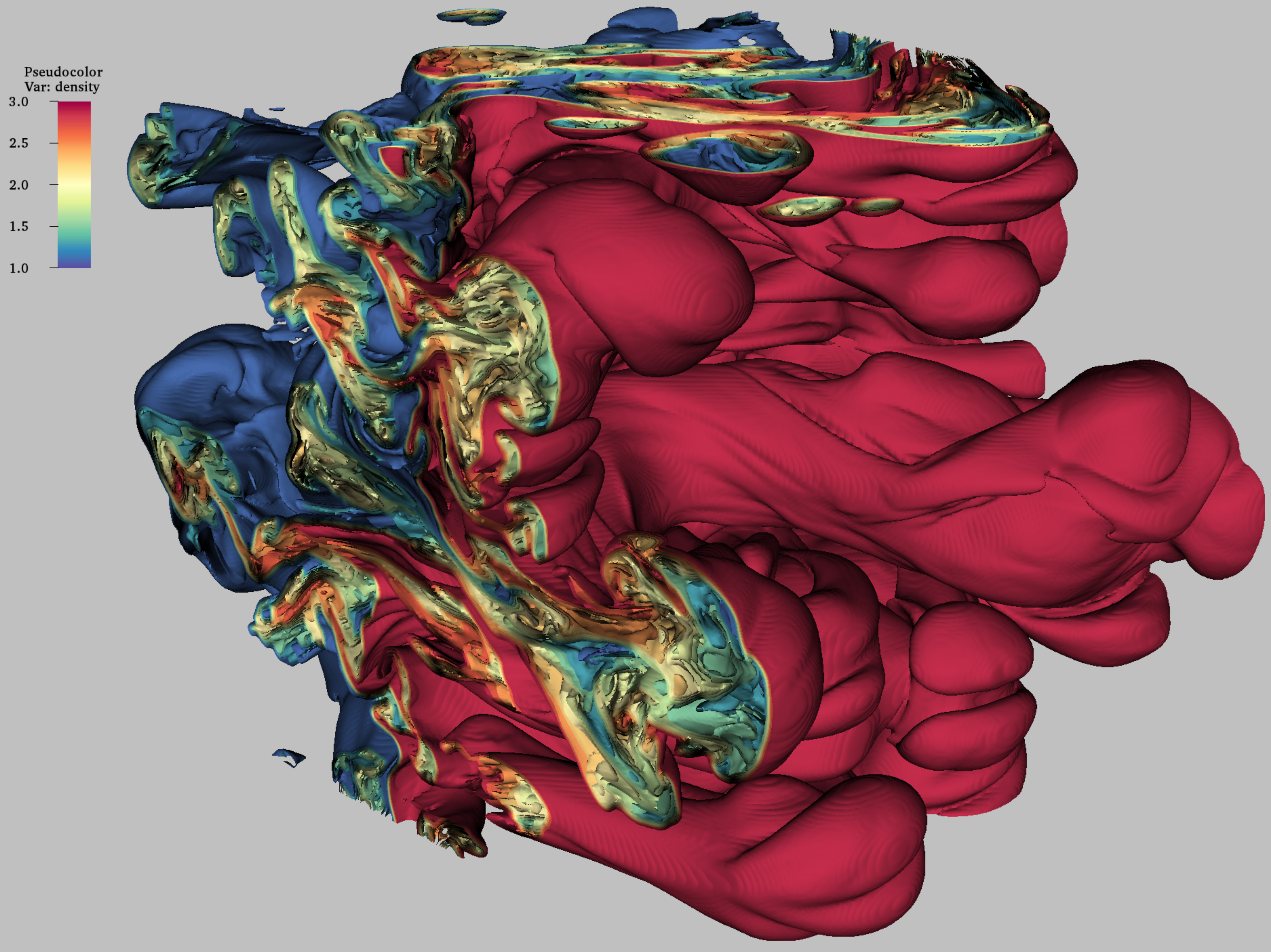


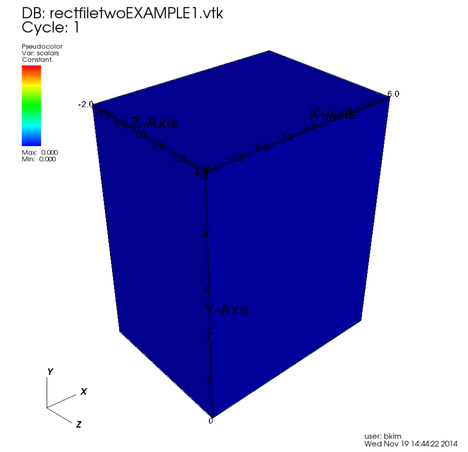
Data Value Selection – *Isosurface* Operator





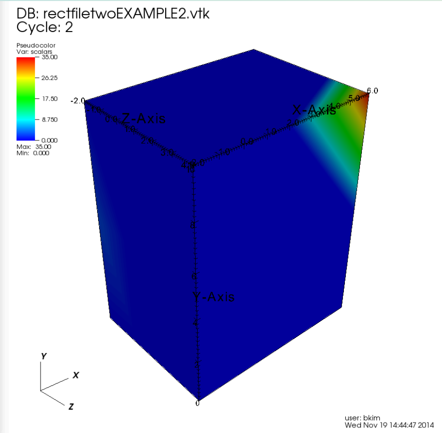
10 Isosurfaces Between Data min - max

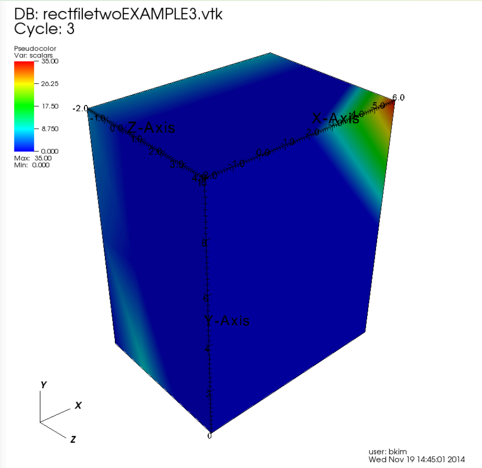


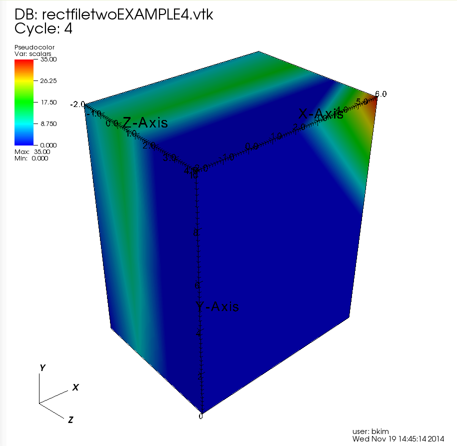


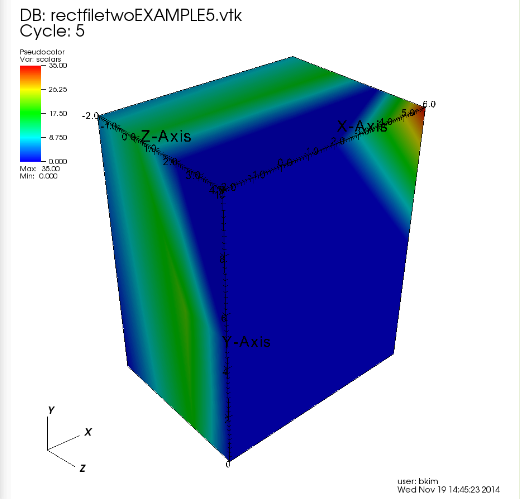
Time Steps *f(x,y,z,t)*

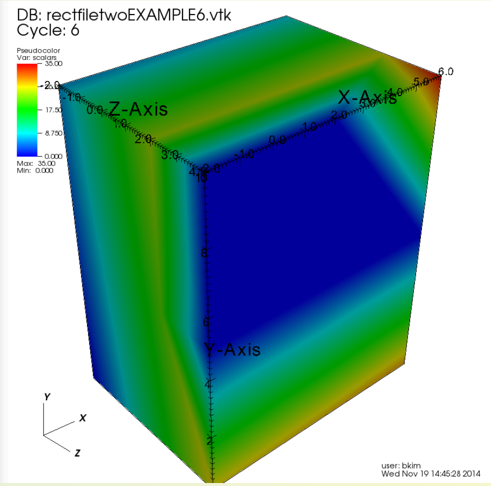
VisIt automatically reads files named in numerical order for time step visualization.

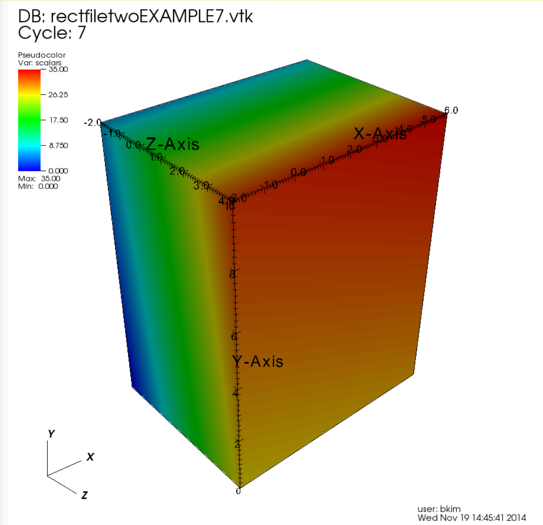


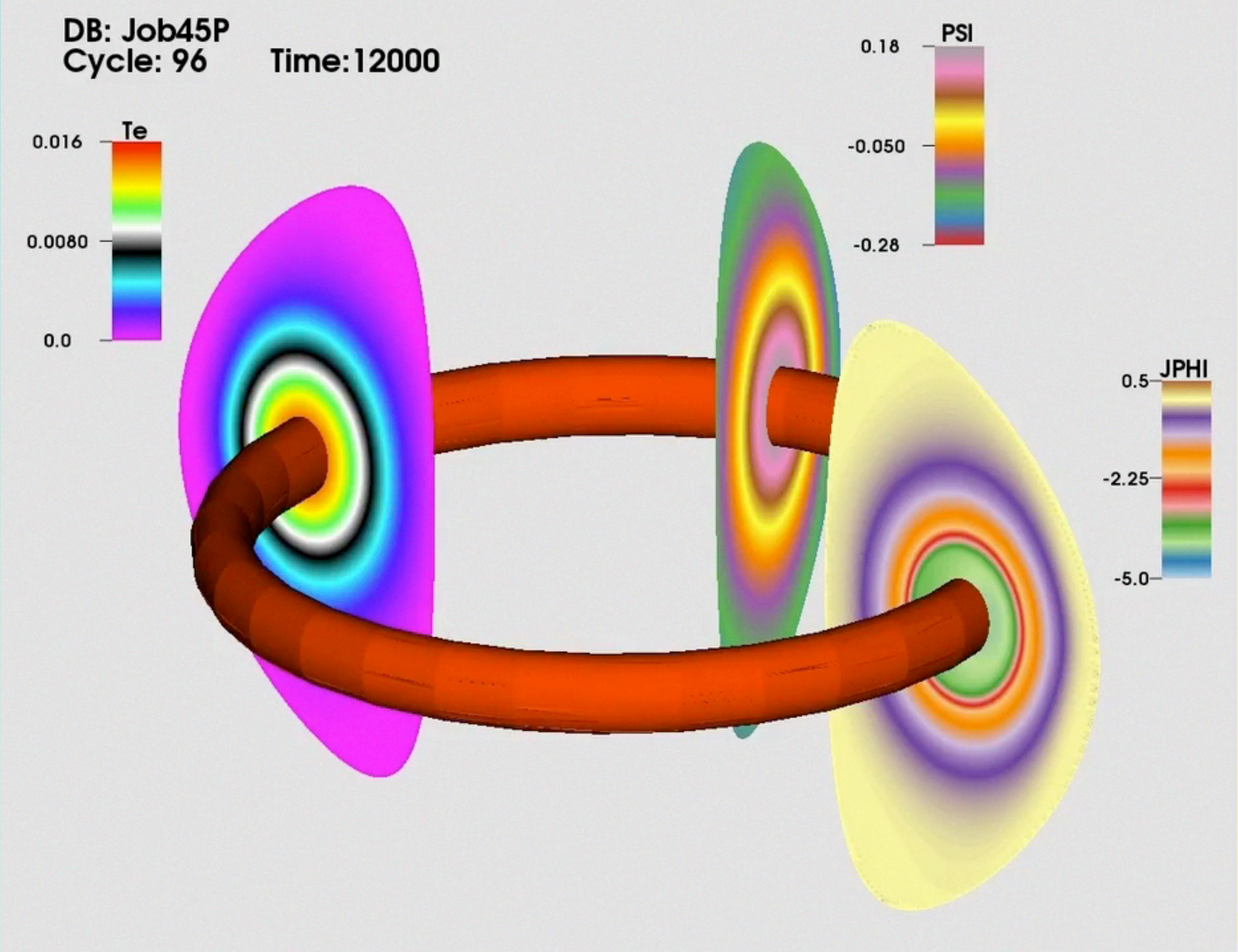










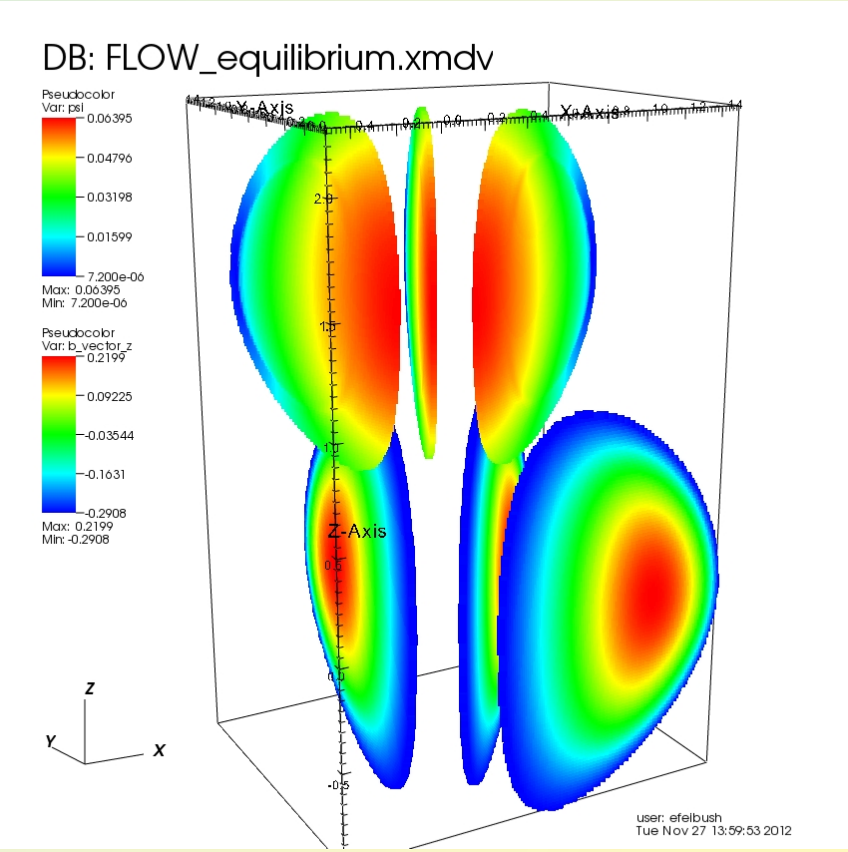


Isosurface of Te = 0.015 at each time step.
Shows Te, PSI, and JPHI concurrently.
http://w3.pppl.gov/~efeibush/movies/teiso015.mov

Transforms

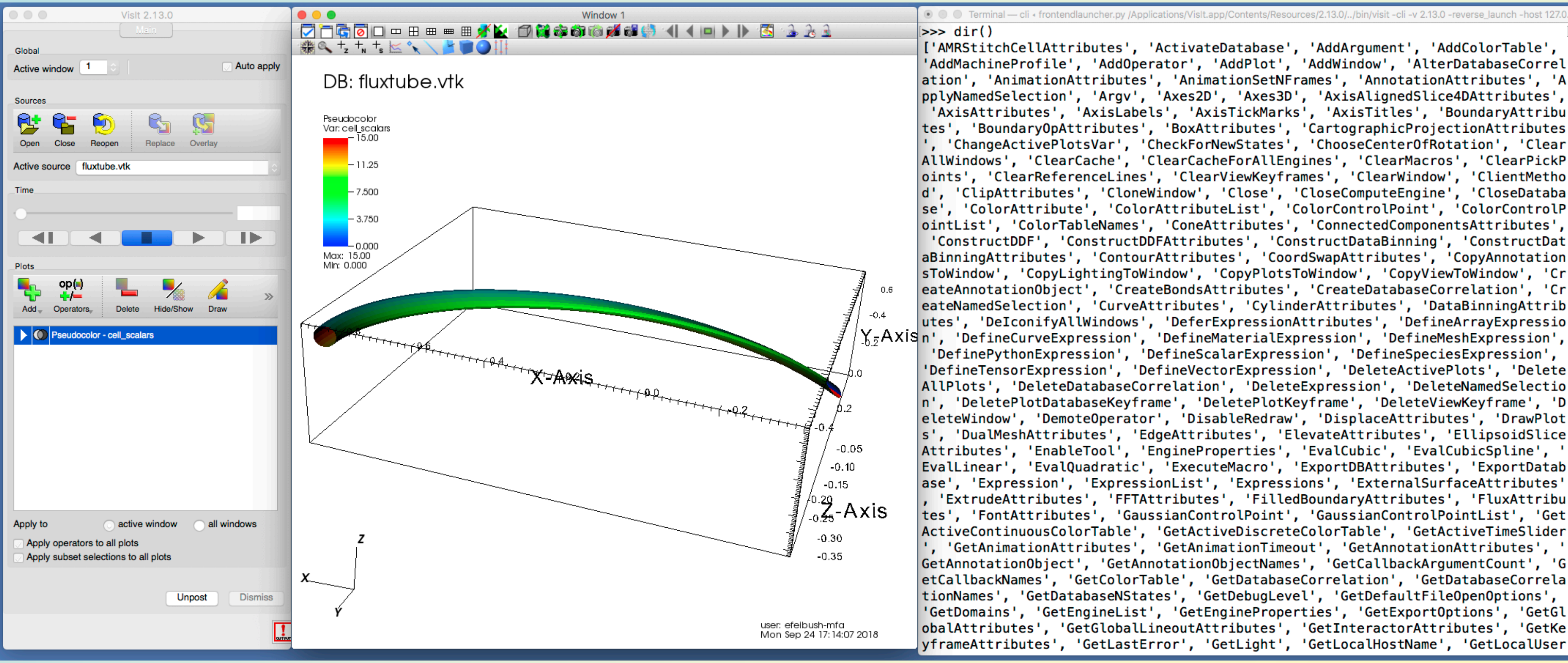
Relocate geometry

- Translate
- Rotate
- Scale



VisIt Python Interpreter

Anything from the GUI + loops, file I/O, numpy …



VisIt Python Interpreter

Anything from the GUI + loops, file I/O, numpy …

```
for k in range(3000):                    # time steps
    filename = "densityData/den" + str(k) + ".Point3D"
    print "opening database file:  " + filename
    OpenDatabase(filename, 0, "Point3D”)
    AddPlot("Pseudocolor", "value")
    DrawPlots()
    SaveWindow()        # auto increments image file name number
```

```
SetPlotOptions                  # range,  log,  geometry
   colorTableName = “CS_Chang”
Legend.numTicks = 3             # also position,  title,  font
SaveWindowAttributes()          # width,  height,  filename
```

Paraview also has a python interpreter!

Summary of Features in VisIt

Plots + Attributes
- Mesh
  - Points, Lines
- Pseudocolor
  - Polygons
  - Color Maps
- Vectors

Data Files
- VTK - grids
- Point3D

Transform Operators
- Scale, Rotate, Translate

Selection Operators
- Clip (geometric)
- Box
- Threshold (data)

Slicing Operators
- Slice, ThreeSlice
- Isosurface

Viewing

Lighting, Shadow, Depth-Cue

Annotation

Animation
- Simple Time Slider movie
- Python scripting

Images to QuickTime movie

Animation

Time step

Variable index

Geometry change

View

Operators (slice, clip, etc.)

Simple VTK file time steps

or

Complex python scripting

jpeg, png files → QuickTime .mov or .mp4

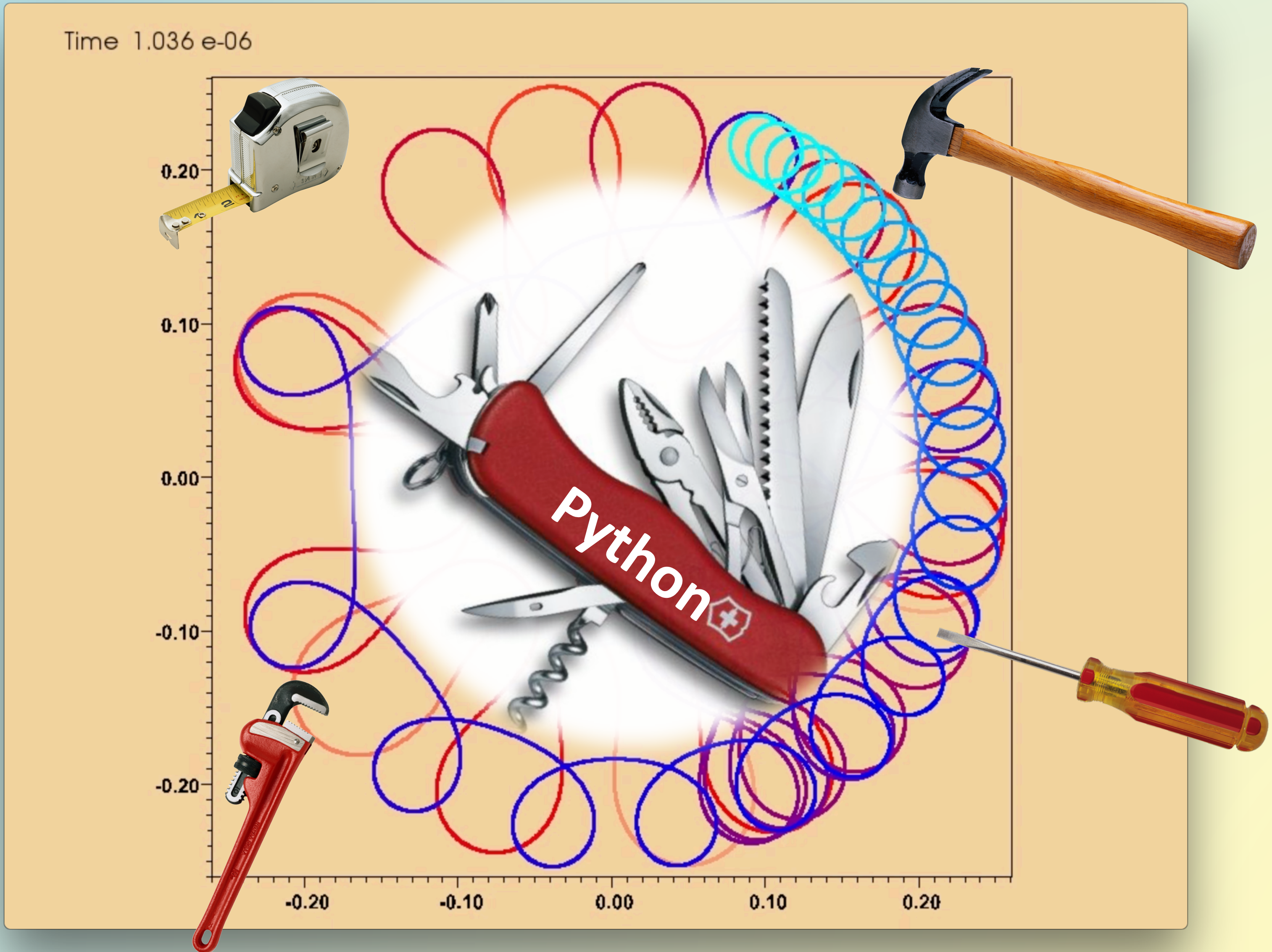
Python interpreter -

```
import myscript
```

[edit, retry]

```
reload(myscript)
```

Include Vis in Workflow



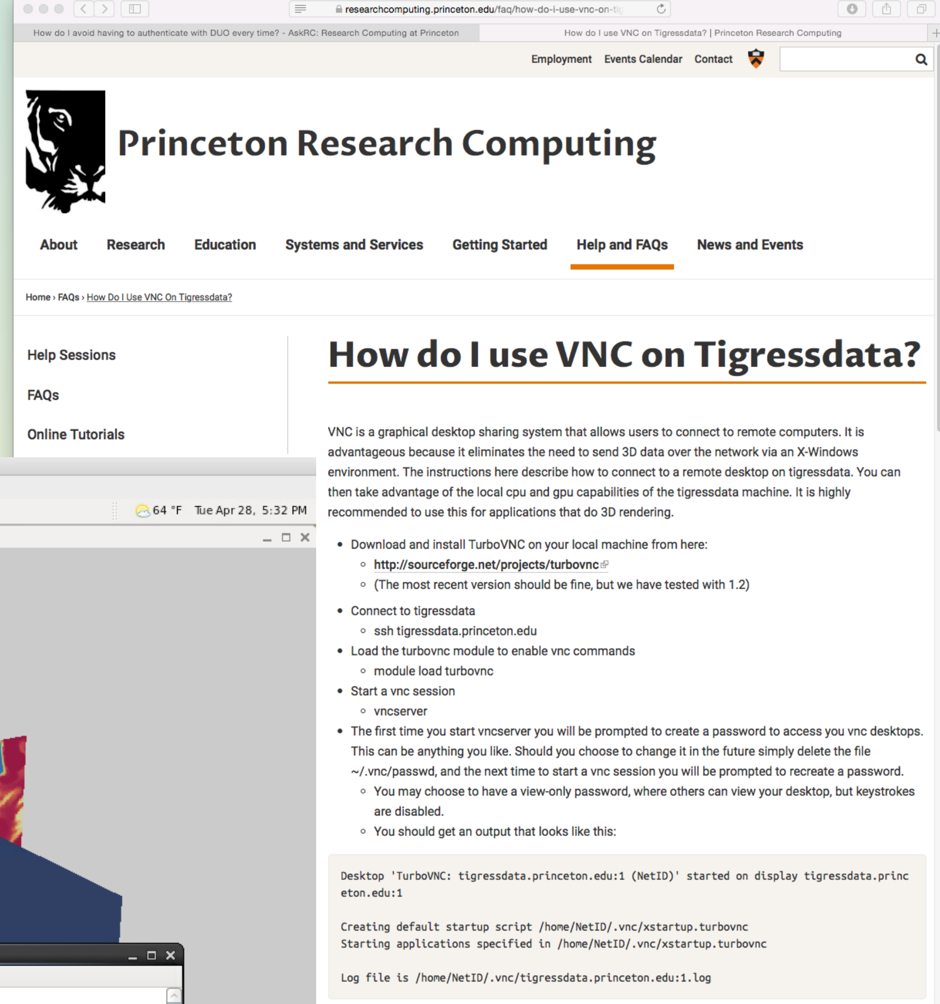
matplotlib
numpy
scipy
Imaging

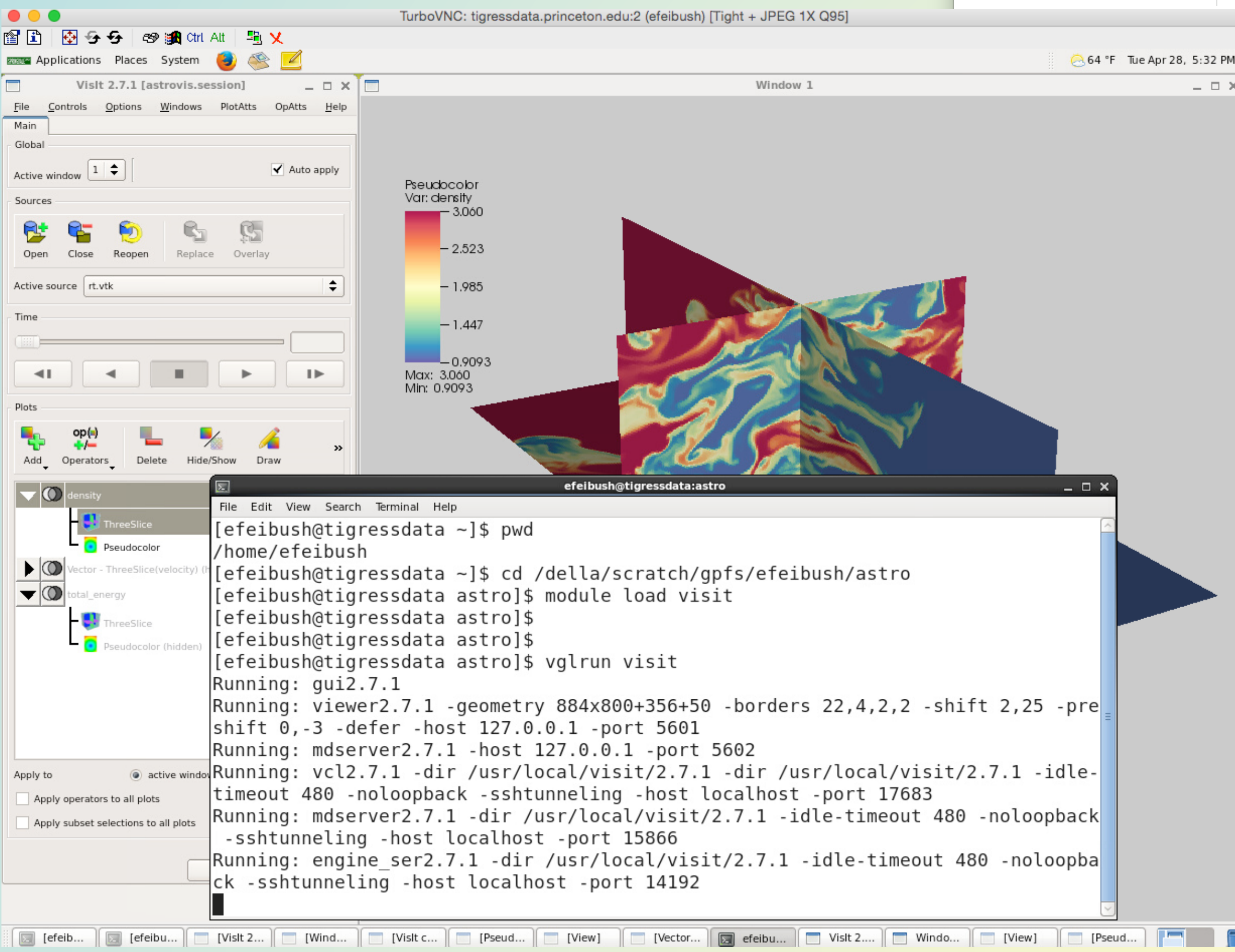
VisIt
Paraview
VTK
ffmpeg

netCDF

Remote Vis at Princeton

turboVNC





File system:
- /tigress
- /projects
- /della/scratch/gpfs
- /perseus/scratch/gpfs
- /tiger/scratch/gpfs
- /home

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

Downloads

Just search for: “visit visualization”

visitusers.org search …

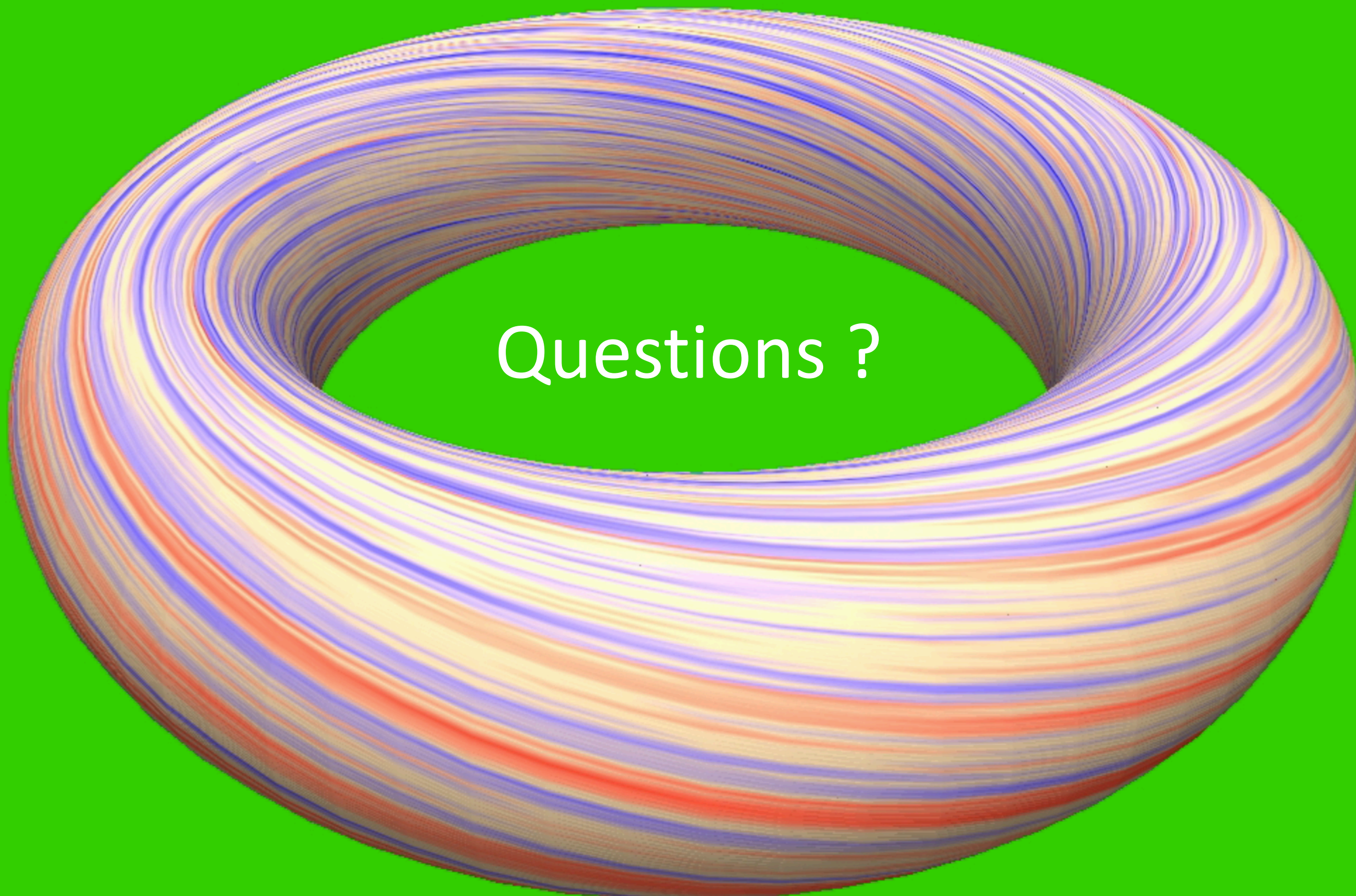
paraview.org

Getting Data Into VisIt - document (& your goal)

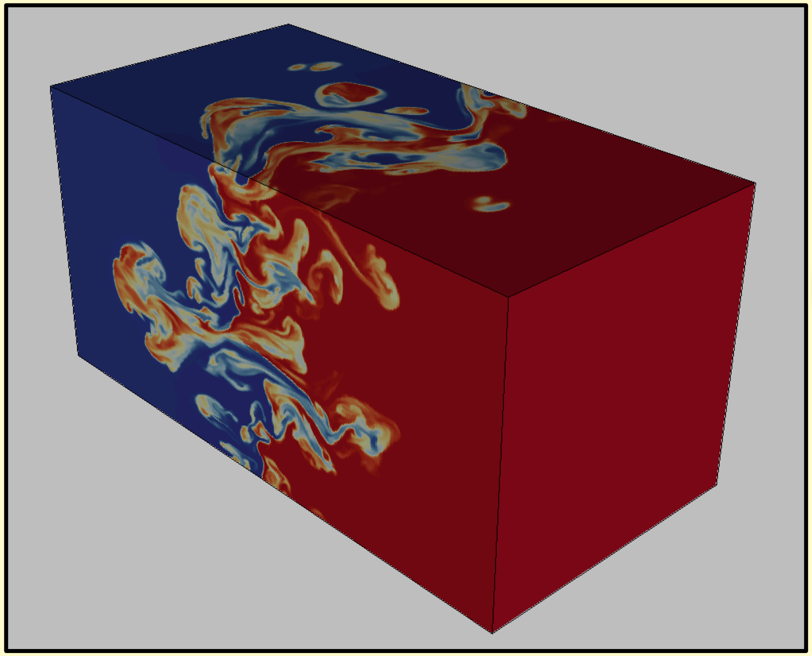
VTK File Formats - vtk.pdf on my website
www.princeton.edu/~efeibush

Python Programming Techniques 11/16

Visualization with VisIt mini-course 11/30

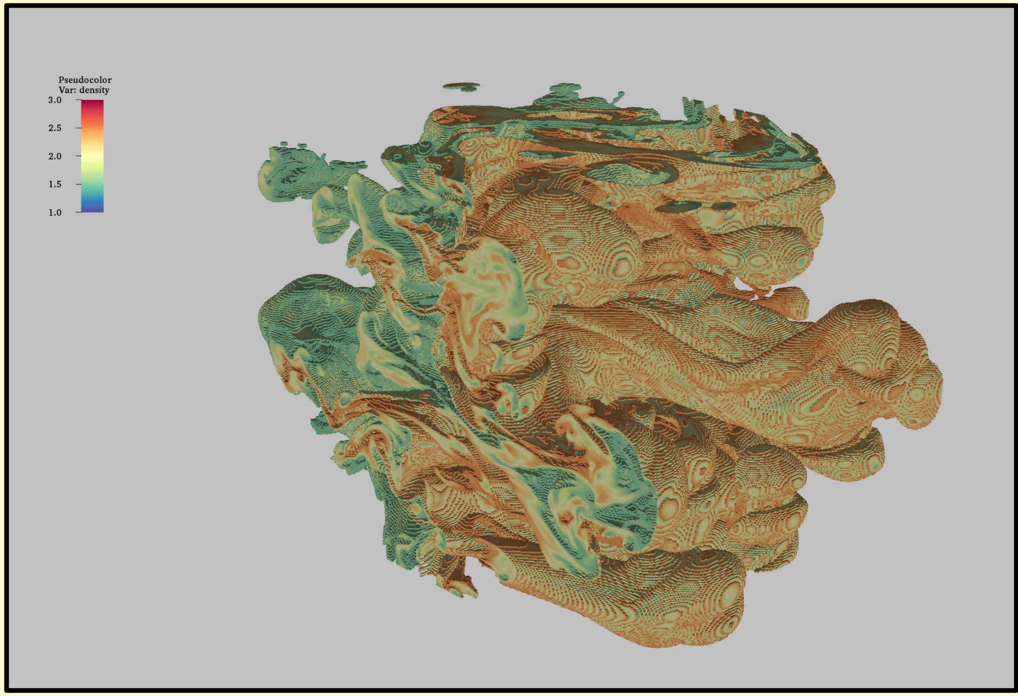

Questions ?

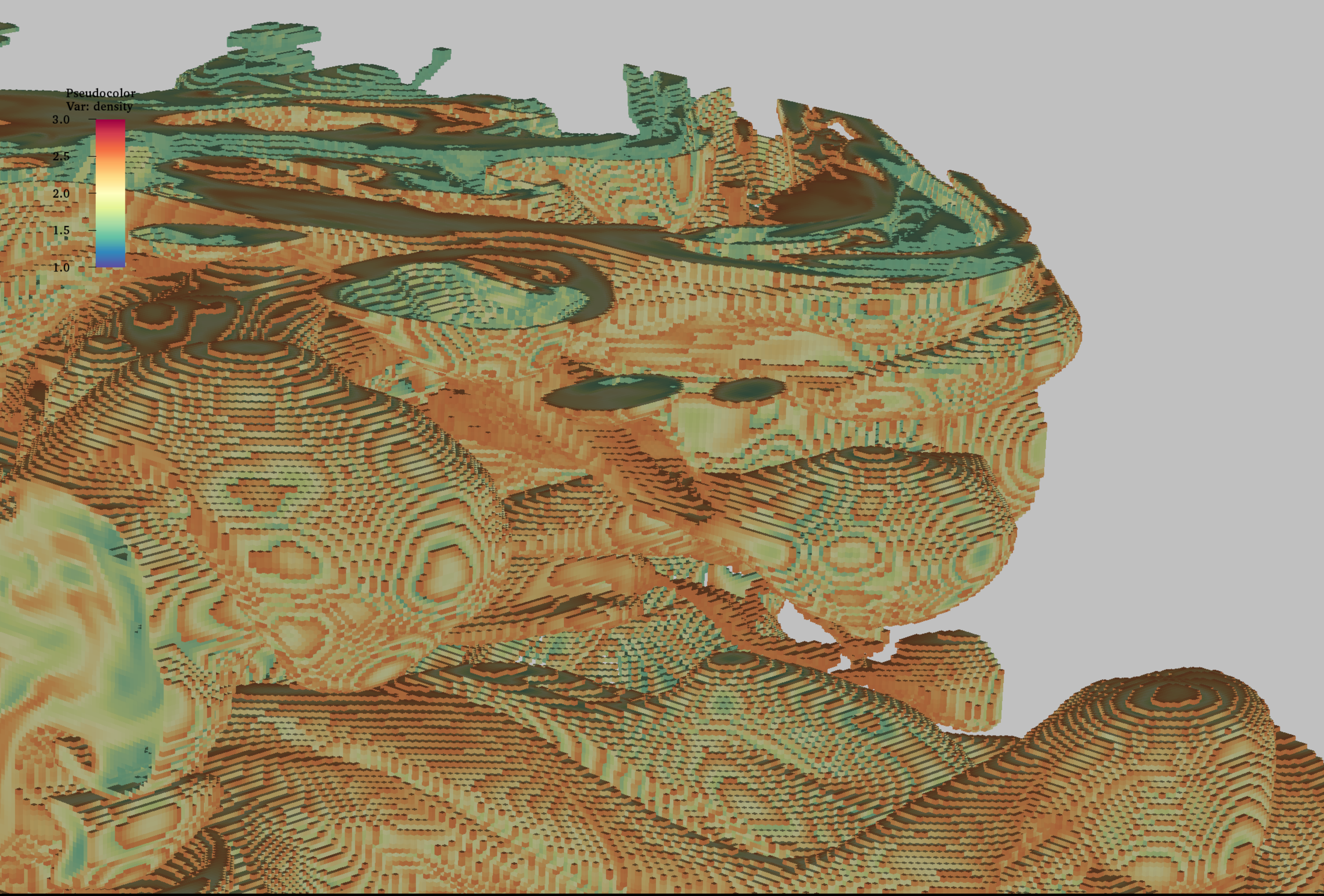
Data Value Selection



257 x 257 x 51 grid
Scalar value per grid point

Threshold Operator
Range 1.5 to 2.5




Pseudocolor
Var: density
3.0
2.5
2.0
1.5
1.0

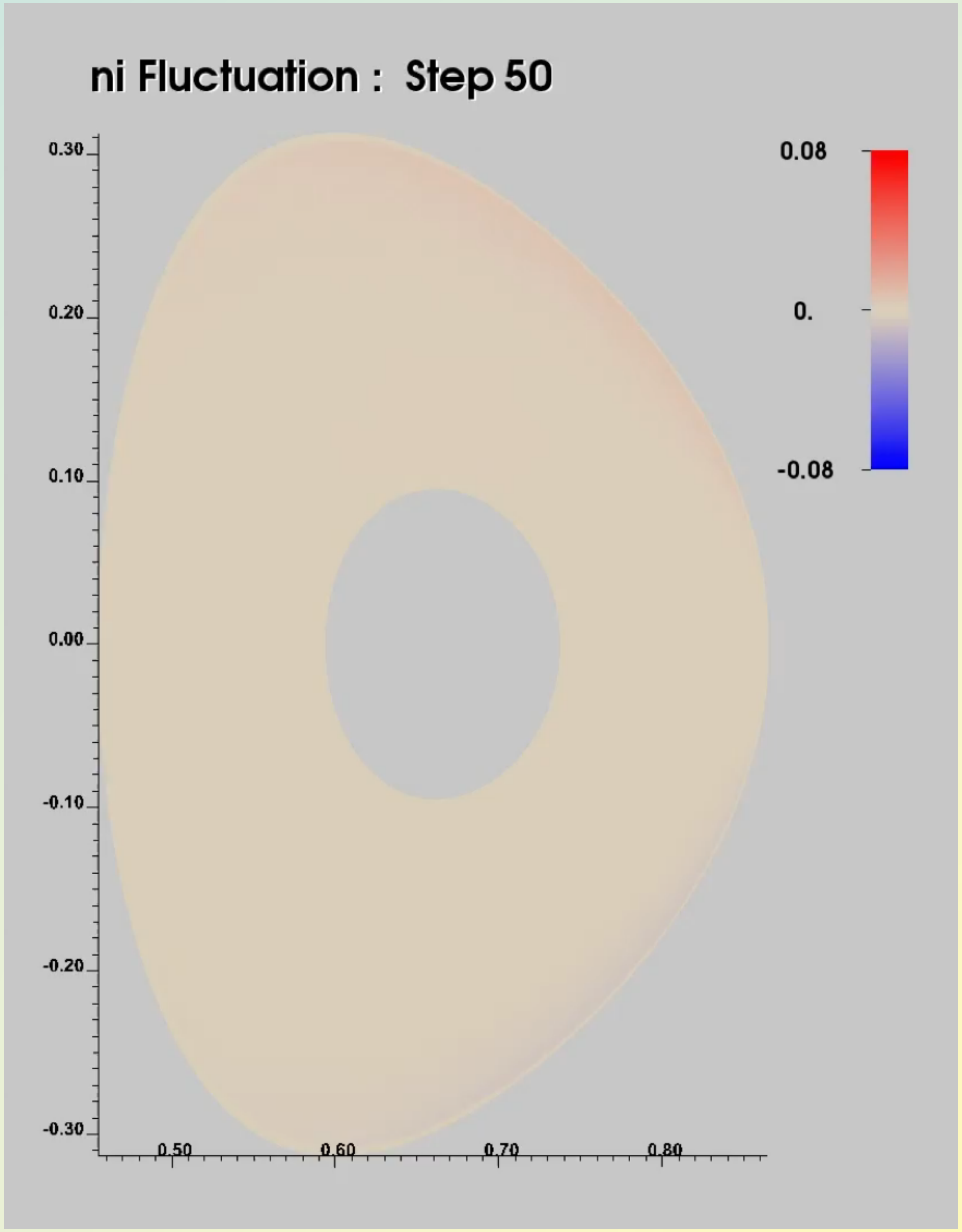
Time Step Simulation

GTS

f(x,y,t)
Ion Density Fluctuation

Direction
Magnitude
Structure

Divergent color map.



http://www.princeton.edu/~efeibush/movies/ni1080.mov