CSCS
Centro Svizzero di Calcolo Scientifico
Swiss National Supercomputing Centre

ETH
Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

ParaView's Comparative Viewing, XY Plot, Spreadsheet View, Matrix View

Dublin, March 2013

Jean M. Favre, CSCS

Motivational movie
Supercomputing 2011 Movie Gallery

CSCS
Centro Svizzero di Calcolo Scientifico
Swiss National Supercomputing Centre

Agenda

- **9:30 – 11:00**

  Start ParaView and show some demos. Do some exercises

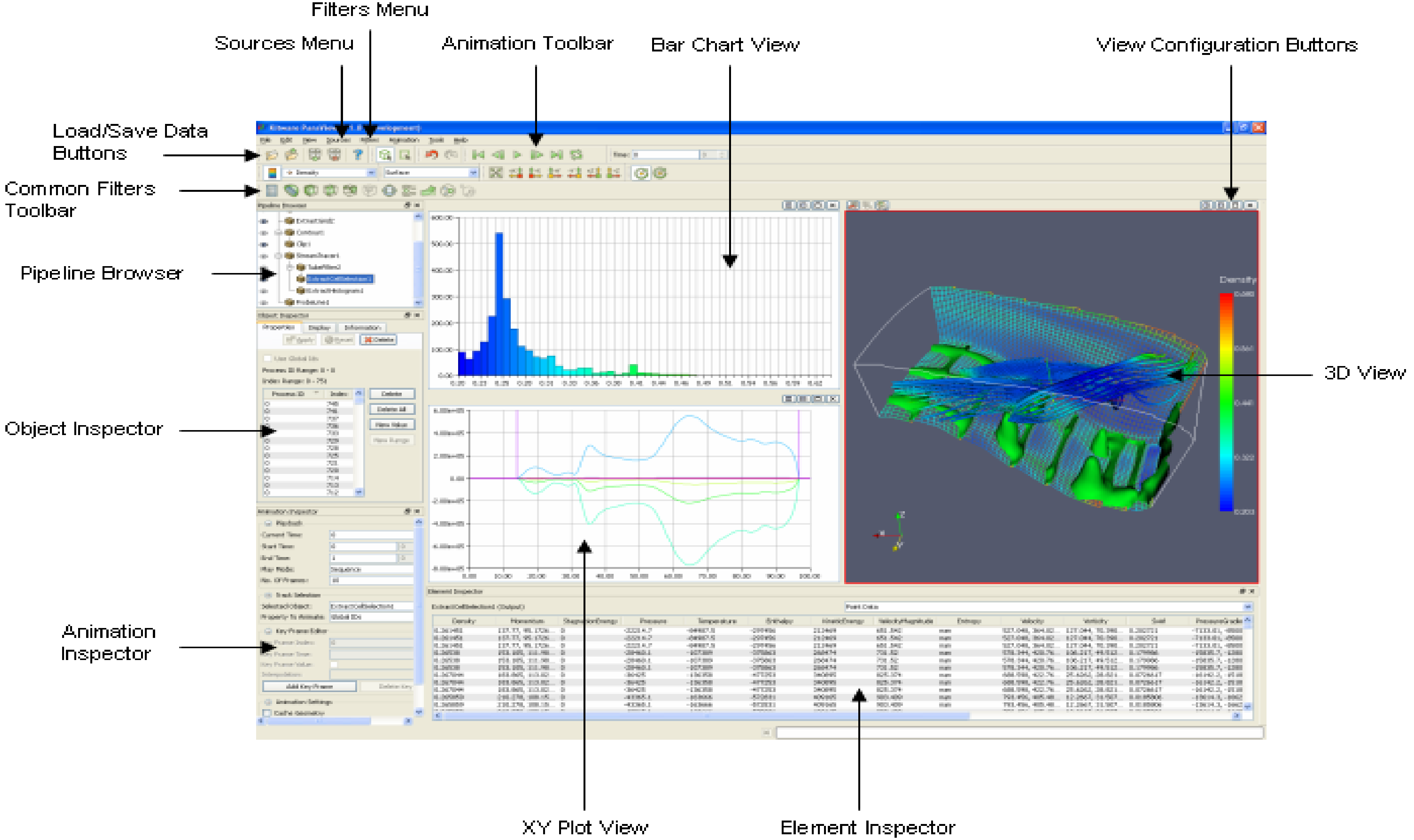
- **11:30 – 13:00**

  Parallel and python usage. More exercises

http://www.paraview.org/files/v3.98/ParaViewData-3.98.1.zip
http://www.paraview.org/files/v3.98/ParaViewData-3.98.1.tar.gz

CSCS
Centro Svizzero di Calcolo Scientifico
Swiss National Supercomputing Centre

Quantitative and qualitative data



Plot Over a Line

- Position the line end-points on the extremities of the dataset
- User can move them back with the “p” stroke (two times)
- Plot will open a “Line Chart View”
- Use panning, zooming and reset camera buttons
- ValidPointMask array set to 0 if data is missing
- Can be done interactively, in “real-time” with the Auto-Accept button (View->Settings)
- Browse with the mouse over the line
- Select which fields to make visibile/invisible

- Exercise with “naca.bin.case”

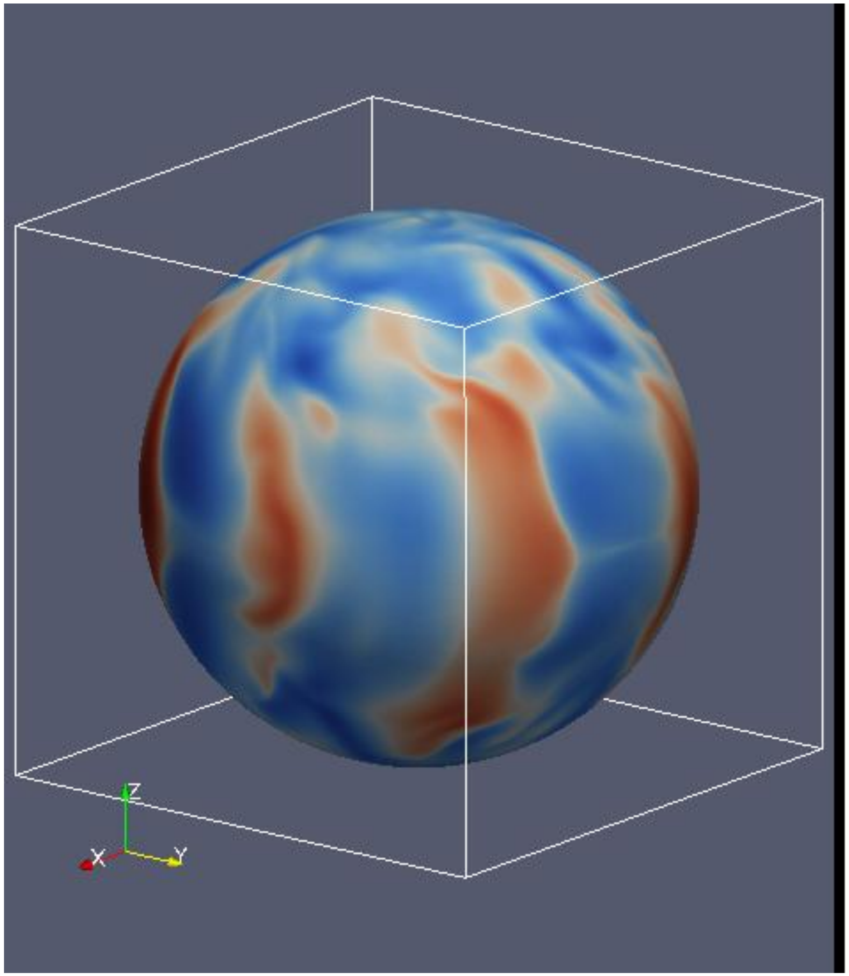
CSCS
Centro Svizzero di Calcolo Scientifico
Swiss National Supercomputing Centre

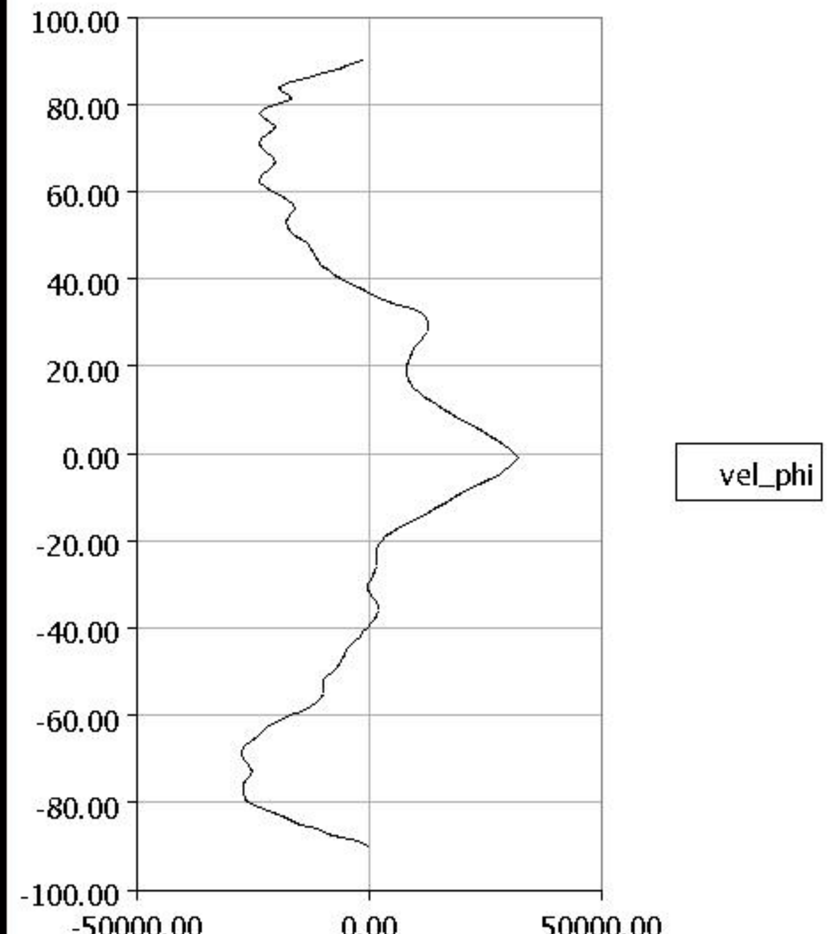
Bart Chart

- Histogram (a vtkTable) will open a “Bar Chart View”
- Use panning, zooming and reset camera buttons
- Can be saved as vtkTable, or as CSV file
- Use SpreadSheet View to look at “RowData”

CSCS
Centro Svizzero di Calcolo Scientifico
Swiss National Supercomputing Centre

GeoPhysics example: Longitudinal average





Spreadsheet View

- Any dataset can be viewed in a “Spreadsheet View”
- Allows display of node-, cell-, field- and row-data
- Allows linked-selection
- Can be exported as CSV file
- Display can be reduced to “Show only selected elements”
- Allows sorting by column

Exercise 1: Source->Wavelet
Filters->PointData to CellData

Select all cells above 230

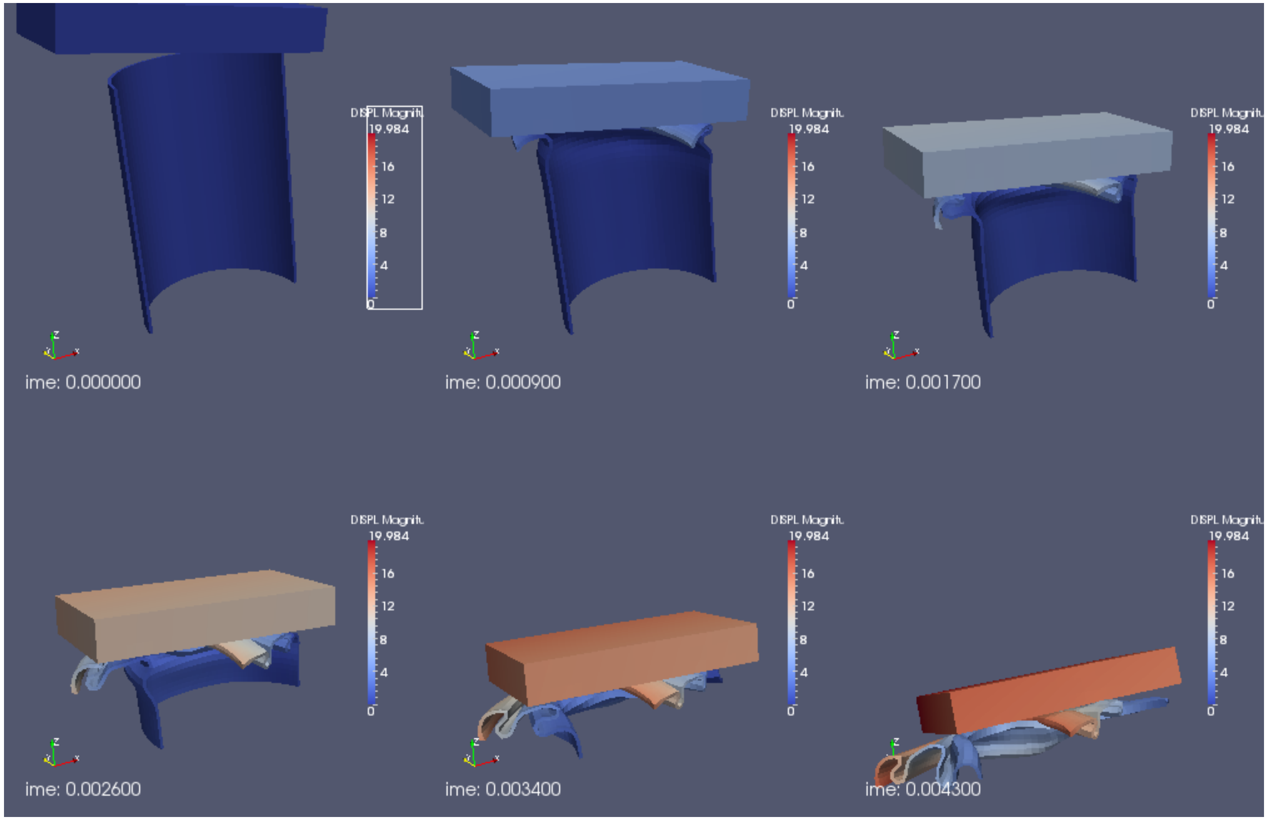
Exercise 2: Use Edit->Find Data to do the same search (Manual page 108)

Comparative Viewing

Compare, side-by-side, multiple visualization pipelines

- Open the 3D View (Comparative) Inspector

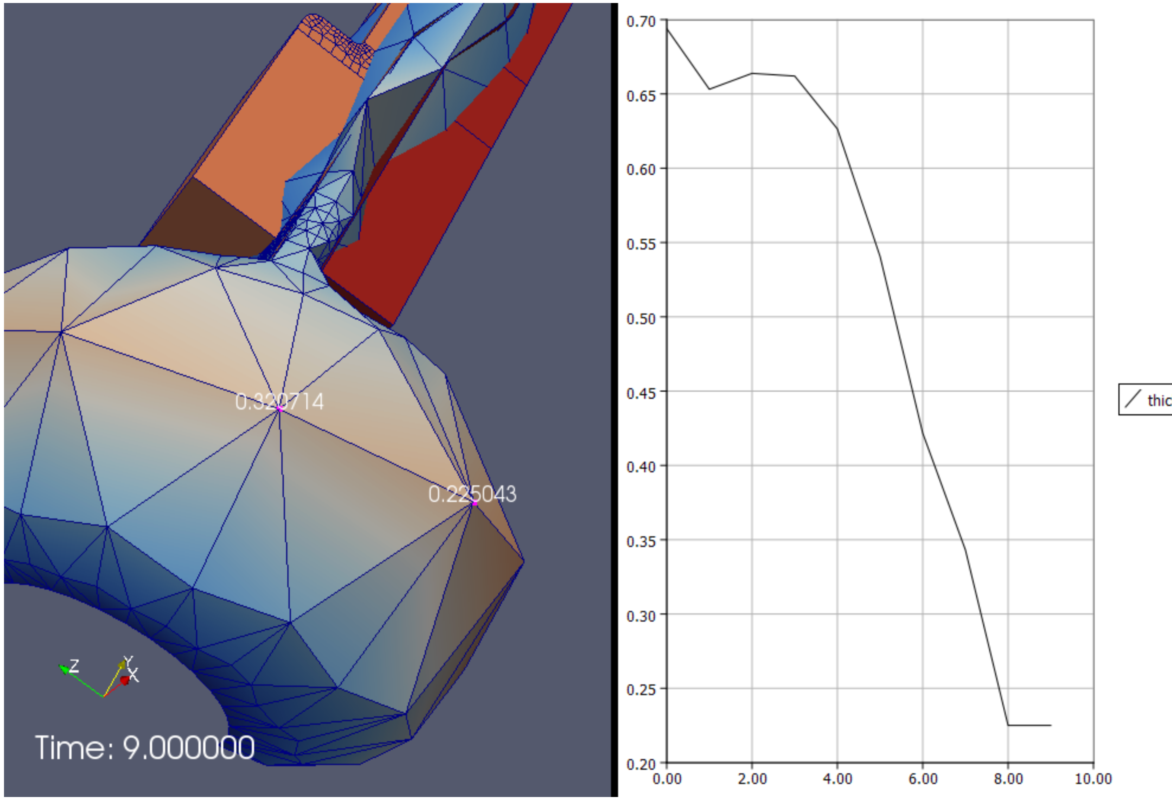
- Load file can.ex2



Plot Over Time

- Multiple points can be tracked over time (based on their ID)
  - Make a selection
  - Copy the Active Selection
  - Apply
  - Plotting is allowed for multiple points

- Produces a multi-block dataset

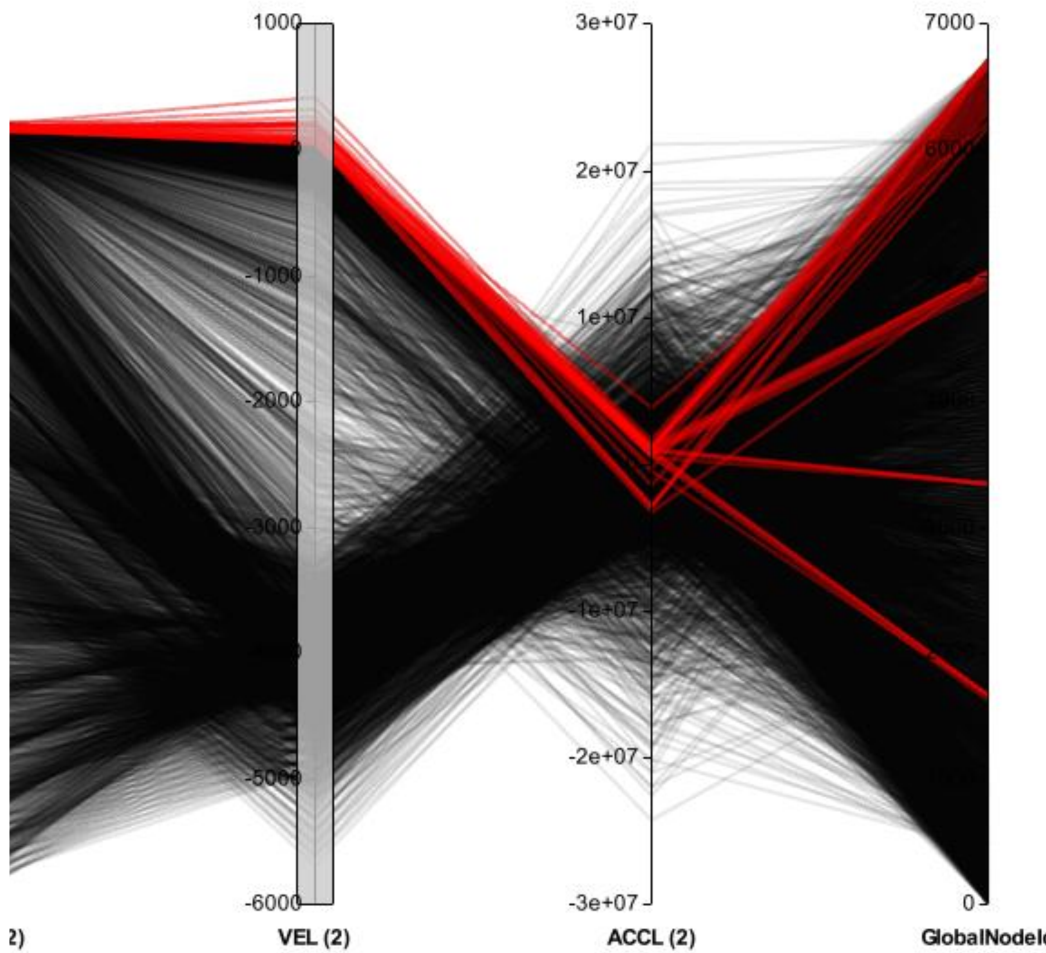


Parallel Coordinates View

- Points are shown in n-dimensional space
- Each vertical column allows subset selection

http://en.wikipedia.org/wiki/Parallel_coordinates

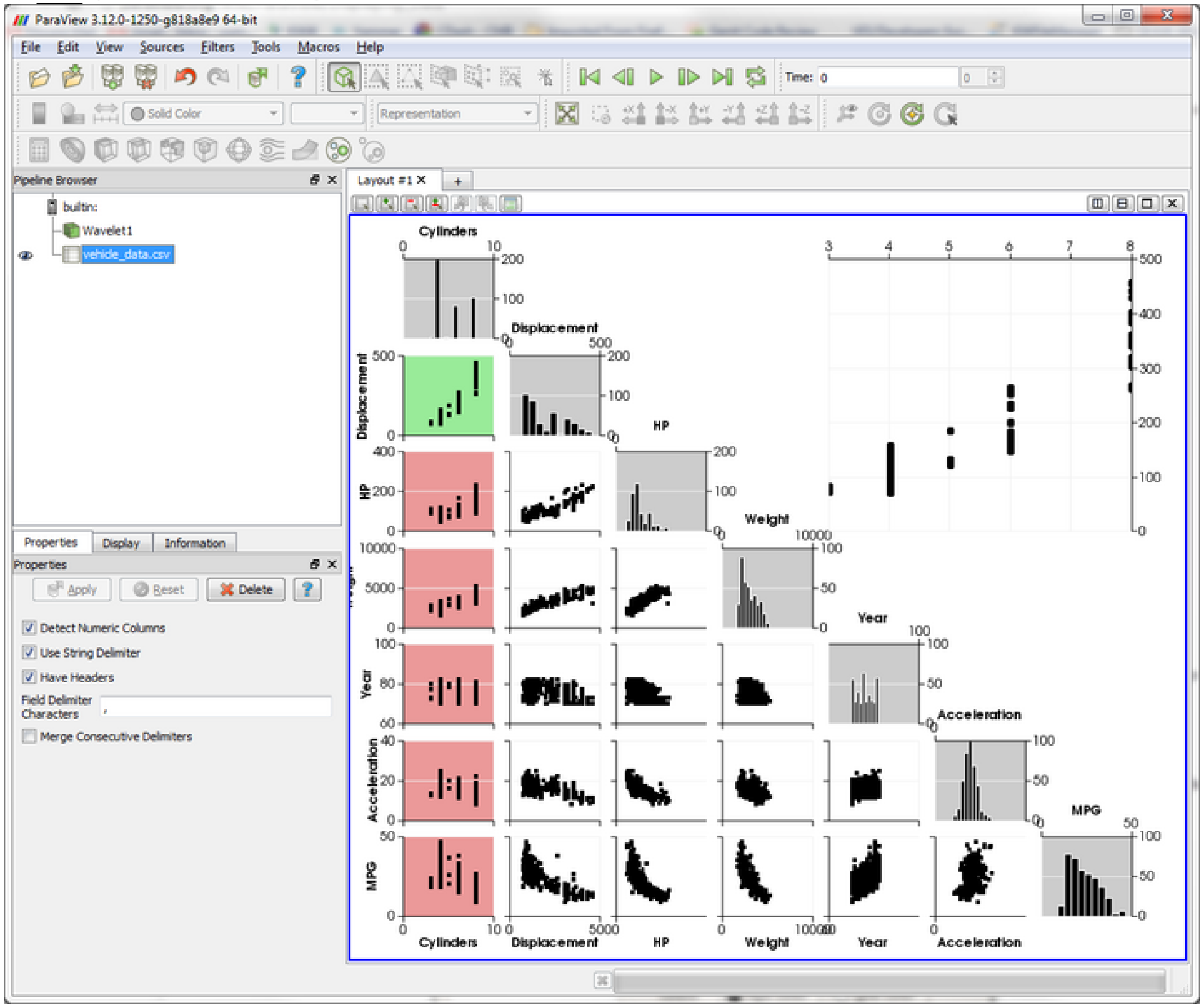
Load "vehicle_data.csv"



CSCS
Centro Svizzero di Calcolo Scientifico
Swiss National Supercomputing Centre

Plot-Matrix View

- Open vehicle_data.csv

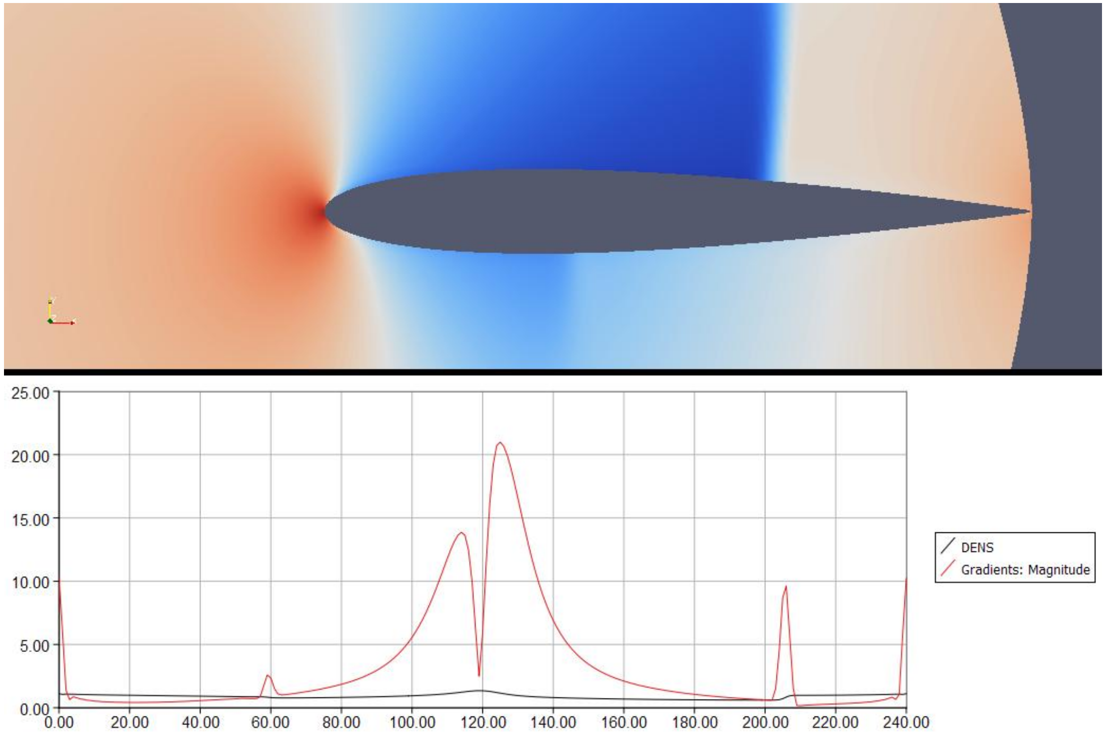


Summary

- http://paraview.org/Wiki/ParaView/Displaying_Data
- Plotting and Charting will use vtkTables
- Comparative viewing is to be done with caution (or low-resolution data)
- Idem for "plot over time"
- Both are ideal candidates for batch-mode processing

Exercise: Naca dataset

- **Load ${PARAVIEW_DATA_ROOT}/Data/naca.bin.case**
- **Plot density and gradient along the curvilinear contour of the airfoil**
- **Export plot as PDF**



CSCS
Centro Svizzero di Calcolo Scientifico
Swiss National Supercomputing Centre

ETH
Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

ParaView Python Tools

Dublin, March 2013

Jean M. Favre, CSCS

Outline

1. Tools, application scripting, python traces
   - pvpython, pvserver
   - parallel execution

2. Quantitative Analysis
   - programmable filters, python calculator

ParaView tools

- paraview, pvbatch can run in a single or multi-cpu session
- pvpython can connect to a parallel server

The “standard” version called paraview, will run interactively, i.e. with a graphics OpenGL window. This is intended to do exploratory visualization, and to prepare a visualization script.

To keep interaction live, you might want to use lower-resolution data

Important:

http://paraview.org/Wiki/ParaView/EnvironmentSetup

pvbatch

The “batch-oriented” tool called pvbatch, will run without user’s interaction.

pvbatch will be used to repeat the same visualization for:

- many time-steps in a transient simulation
- different input datasets
- to customize an animation

pvbatch can execute a hand-written python script, or reload a script generated with paraview, and save images to disk.

Reloading a state file

paraview can reload a state file with the option –state=filename.pvsm

paraview can reload a state with the command File->Load State

pvbatch can reload the same state file with the commands:

```
from paraview.simple import *
Connect()
servermanager.LoadState('/users/jfavre/state.pvsm')
```

CSCS
Centro Svizzero di Calcolo Scientifico
Swiss National Supercomputing Centre

Reloading a python script

paraview can reload a python script with the option --script=filename.py

paraview can reload a python script with the command Tools->Python Shell->Run Script

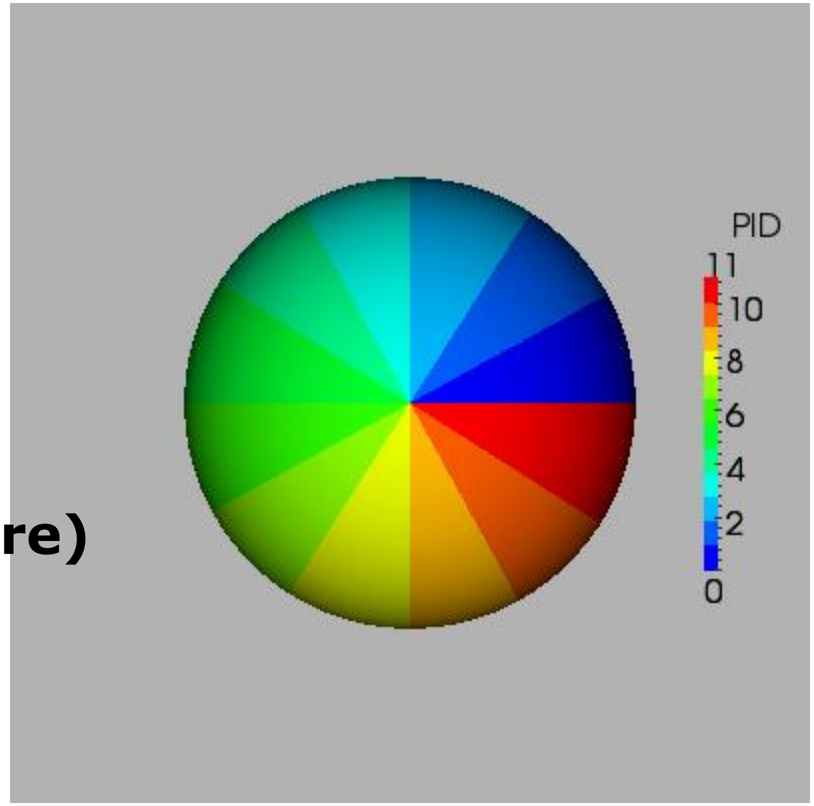
Try reloading lib/paraview-3.98/site-packages/paraview/demos/demo1.py

```
sph = Sphere()
shr = Shrink()
rep = Show()
Render()
```

CSCS
Centro Svizzero di Calcolo Scientifico
Swiss National Supercomputing Centre

ColoredSphere (parallel) example

```
from paraview.simple import *

view = GetRenderView()

sphere = Sphere()
sphere.PhiResolution = 100

pidscal = ProcessIdScalars(sphere)

rep = Show(pidscal)

nbprocs =
  servermanager.ActiveConnection.GetNumberOfDataParti
  tions()
drange = [0, nbprocs-1]
```

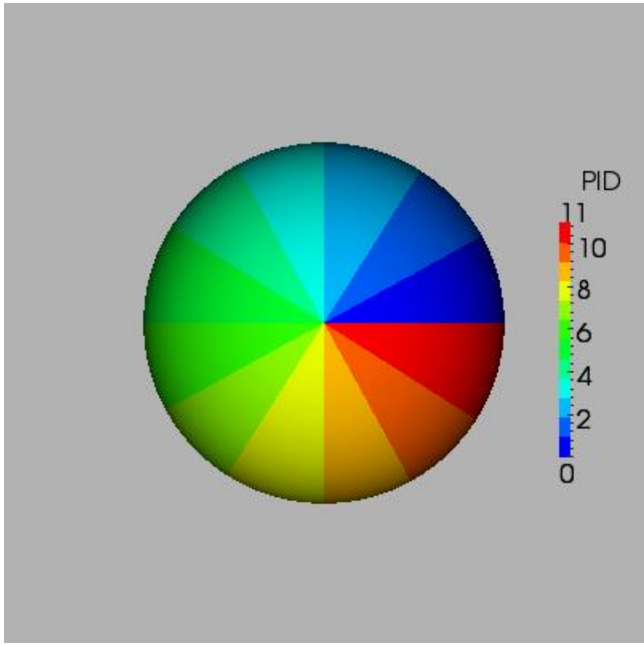


ColoredSphere (parallel) example

```
lt = MakeBlueToRedLT(drange[0], drange[1])
lt.NumberOfTableValues = nbprocs

rep.LookupTable = lt
rep.ColorAttributeType = 'POINT_DATA'
rep.ColorArrayName = "ProcessId"

bar = CreateScalarBar(LookupTable=lt, Title="PID")
bar.TitleColor = [0,0,0]
bar.LabelColor = [0,0,0]
bar.NumberOfLabels = 6

view.Representations.append(bar)
```

running the example with pvbatch

```
view.ResetCamera()
view.Background = [.7, .7, .7]
view.CameraViewUp = [0, 1, 0]
view.StillRender()
WriteImage("coloredSphere.png", view=view,
  Writer="vtkPNGWriter")


# Execute with MPI

mpirun –n12 `which pvbatch` \
         --use-offscreen-rendering \
         coloredSphere.py
```



How to get started with Python commands?

- http://paraview.org/Wiki/ParaView/Python_Scripting

- Utilities/VTKPythonWrapping/servermanager.py
- Utilities/VTKPythonWrapping/simple.py

- Use Python Shell -> Trace

- Start trace, trace state, show/edit/save trace

- The traces are very verbose. Editing is recommended.

Look at data fields stored in the grid

```
r = OpenDataFile("/ParaViewData/Data/bluntfin.vts")
r.UpdatePipeline()

pd = r.PointData

for n in range(pd.GetNumberOfArrays()):
  print pd.GetArray(n).GetName(), ' ',
  pd.GetArray(n).GetRange()

for n in range(pd.NumberOfArrays):
  print pd[n].Name, ' ', pd[n].GetRange()

for k, v in pd.iteritems():     # pd is a python dictionary
  print k, v.GetRange()
```

Execute a script for multiple timesteps

AnimateReader() (from simple.py) is a macro that takes a time-aware data source, a view, and a filename

AnimateReader(reader, GetRenderView(), "/tmp/foo.png")

It will step through all timesteps. The execution is run on-demand by the view

Execute for some timesteps

AnimateReader() starts at the beginning and runs to the end with a fixed increment. You can change that and do your own start, end, and time increment.

```
tsteps = reader.TimestepValues
start = 2
incr = 3
end = 7
for i in tsteps[start:end:incr]:
    view.ViewTime = tsteps[i]
    view.StillRender()
    imgfile = “image.%03d.png” % (start+i*incr)
    view.WriteImage(imgfile, “vtkPNGWriter”, 1)
```

Exercise with a file can.ex2

reader = FindSource("can.ex2")

view = GetRenderView()

tsteps = reader.TimestepValues
start = 0
incr = 1
end = len(tsteps) -1
for i in tsteps[start:end:incr]:
   view.ViewTime = tsteps[i]
   view.StillRender()

AnimateReader(reader, view,
"c:/Users/jfavre/foo.png")

Execute a script for multiple files

While running paraview, get the python interface. Find all files

Tools-> Python Shell

```
import glob, string

files = glob.glob("/scratch/user/file*.dat")
files.sort()
```

Execute a script for multiple files

How do we update the pipeline objects?

must find the names of the objects to be modified

```
view = GetRenderView()
#you created a pipeline and read a file “file.000.dat”
#using the GUI Open menu
# the object called 'file.000.dat' shows in the pipeline viewer

reader  = FindSource('file.000.dat')
# reader can now be updated
reader.Filename = files[i]
view.StillRender()
```

CSCS
Centro Svizzero di Calcolo Scientifico
Swiss National Supercomputing Centre

Quantitative Analysis

- Calculator (page 90)
- Python Calculator (page 96)
- Programmable Source/Filter (page 87)

Ref. ParaView 3.98 User Manual

Calculator filter

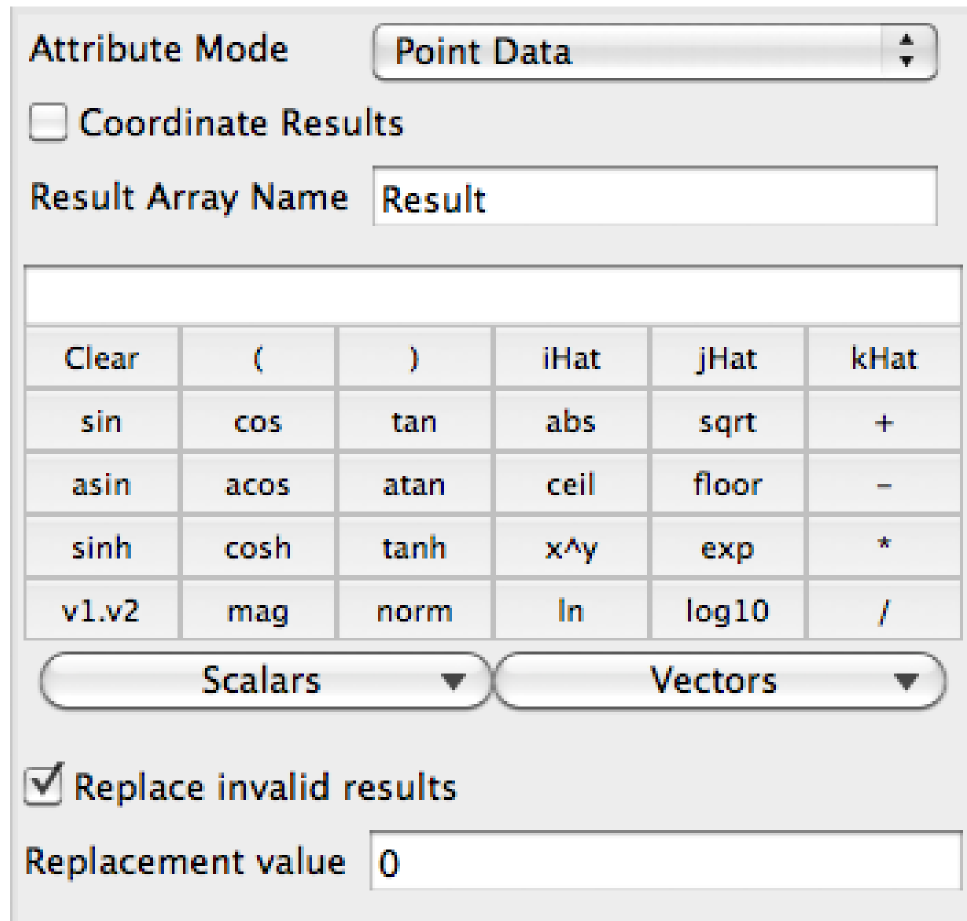
Calculate derived quantities from existing attributes

Use a free-form text expression

Example:

5 * RTData

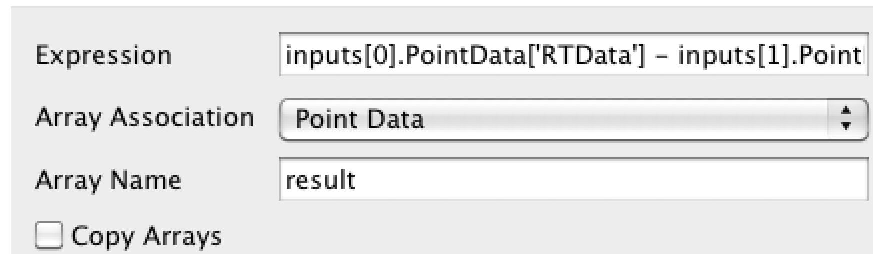
if(condition,true_expression,false_expression)



Python Calculator filter

Uses python and numpy

- Accepts multiple inputs. inputs[0], inputs[1], …
- Can access the point or cell data using the .PointData or .CellData qualifiers.
- Can access the coordinates array using the .Points qualifier:




```
inputs[0].PointData[‘Normals']
inputs[0].Points[:,0]
```

Python Calculator filter

Examples:

```
Normals + 5
Normals + [1,2,3]
velocity[:, 0]
hstack([velocity_x, velocity_y, velocity_z])
```

- When the calculation is more involved and trying to do it in one expression may be difficult…. When you need access to a program flow construct such as if or for…
- When you need to change the type of the mesh…
- => use the programmable filter

Python Programmable Source/Filters

Creates and transforms VTK grids

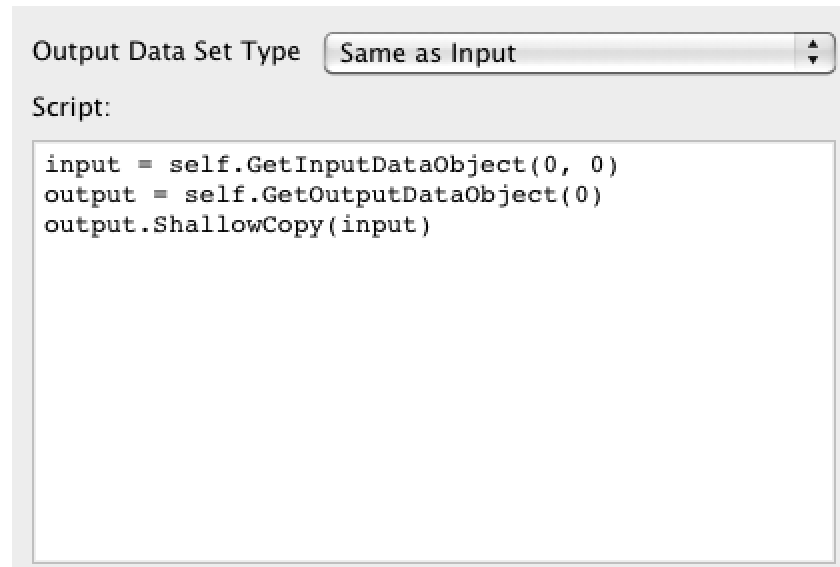
Examples:

Have a Python code to read data, and you may re-use it instead of writing a C++ reader.

Prototype a filter, without a GUI

Import one of many python packages…

Extract the data arrays of a ‘Grid’ and show them as a ‘Table’

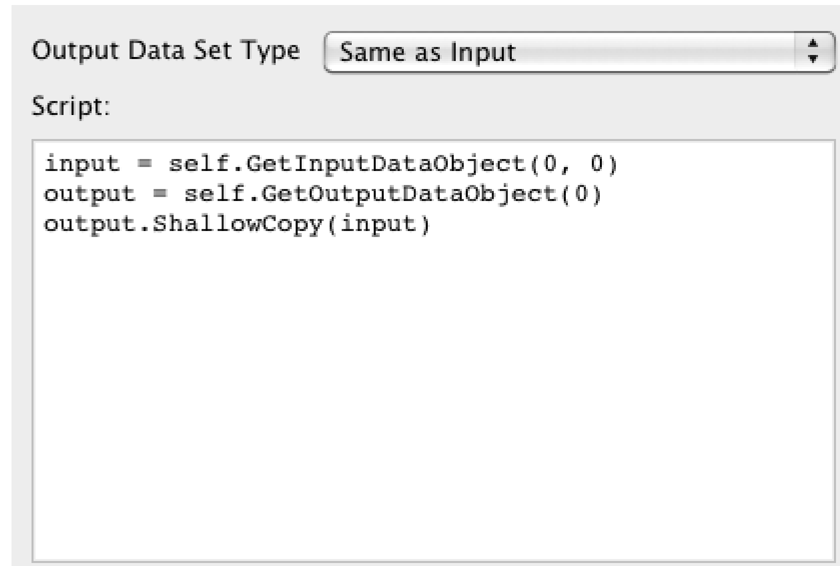


Python Programmable Source/Filters

In it simplest form, the input is copied to the output.

It is a pass-thru filter

With the "Copy Arrays" option, the output will have all of the input arrays



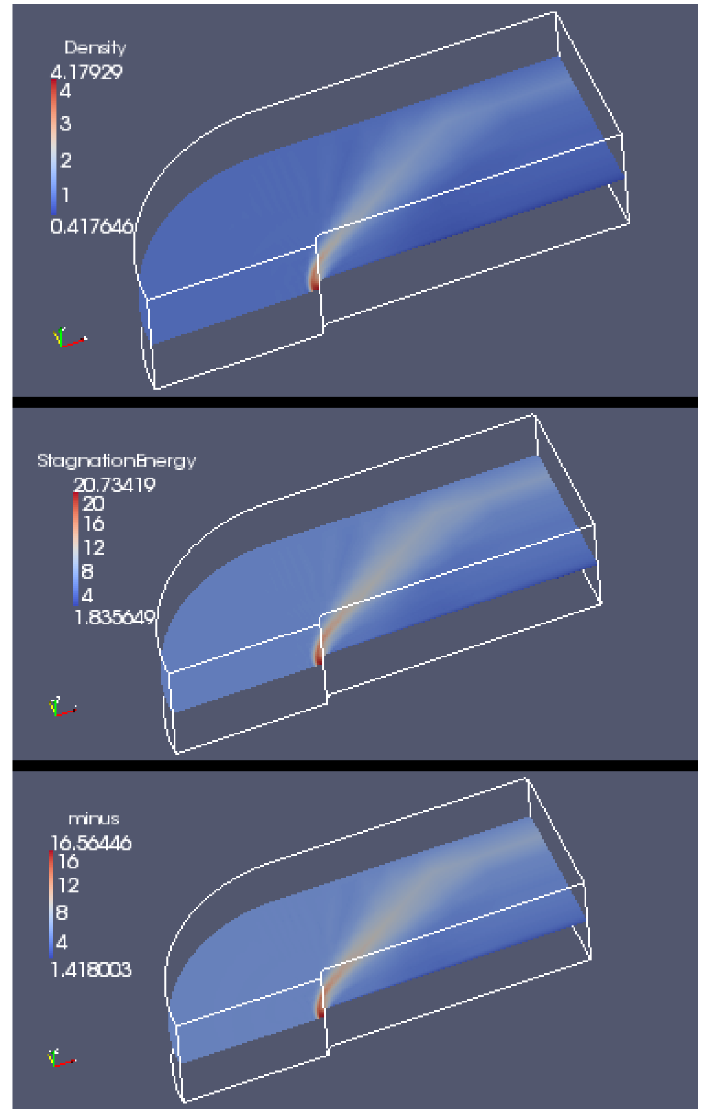
Example:

create a Sphere Source and add

```
normals = inputs[0].PointData['Normals']
output.PointData.append(normals[:,0], "Normals_x")
```

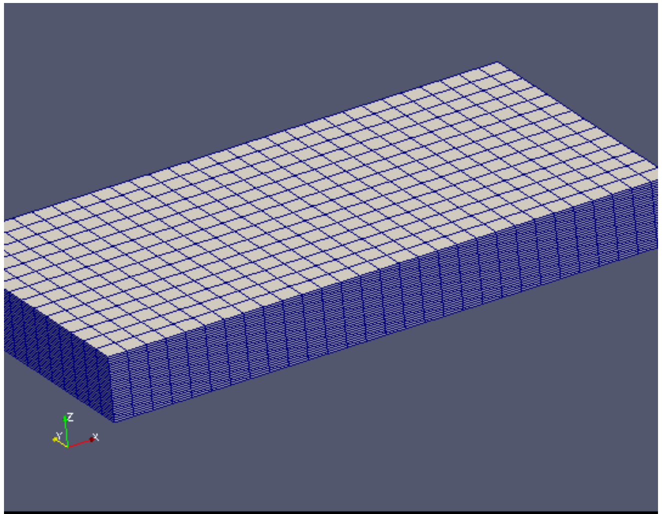
Example of the use of numpy

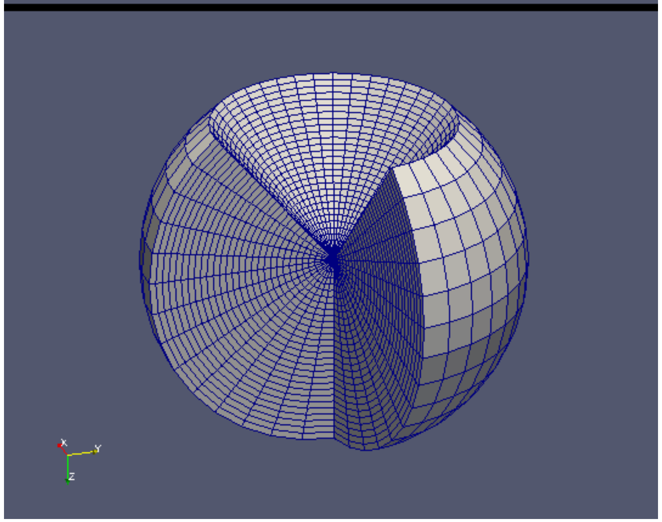
```
#get VTK objects
pdi = self.GetInputDataObject(0,0)
pdo = self.GetOutputDataObject(0)
pdo.ShallowCopy(pdi)

#manipulate Python objects
data0 = inputs[0].PointData['Density']
data1 = inputs[0].PointData['Energy']
output.PointData.append(data1-data0,
   'minus')
```



CSCS
Centro Svizzero di Calcolo Scientifico
Swiss National Supercomputing Centre

Example : create a grid and remap it to spherical space





Python Prog. Source

Python Prog. Filter

Source code here

http://www.paraview.org/pipermail/paraview/2010-August/018495.html

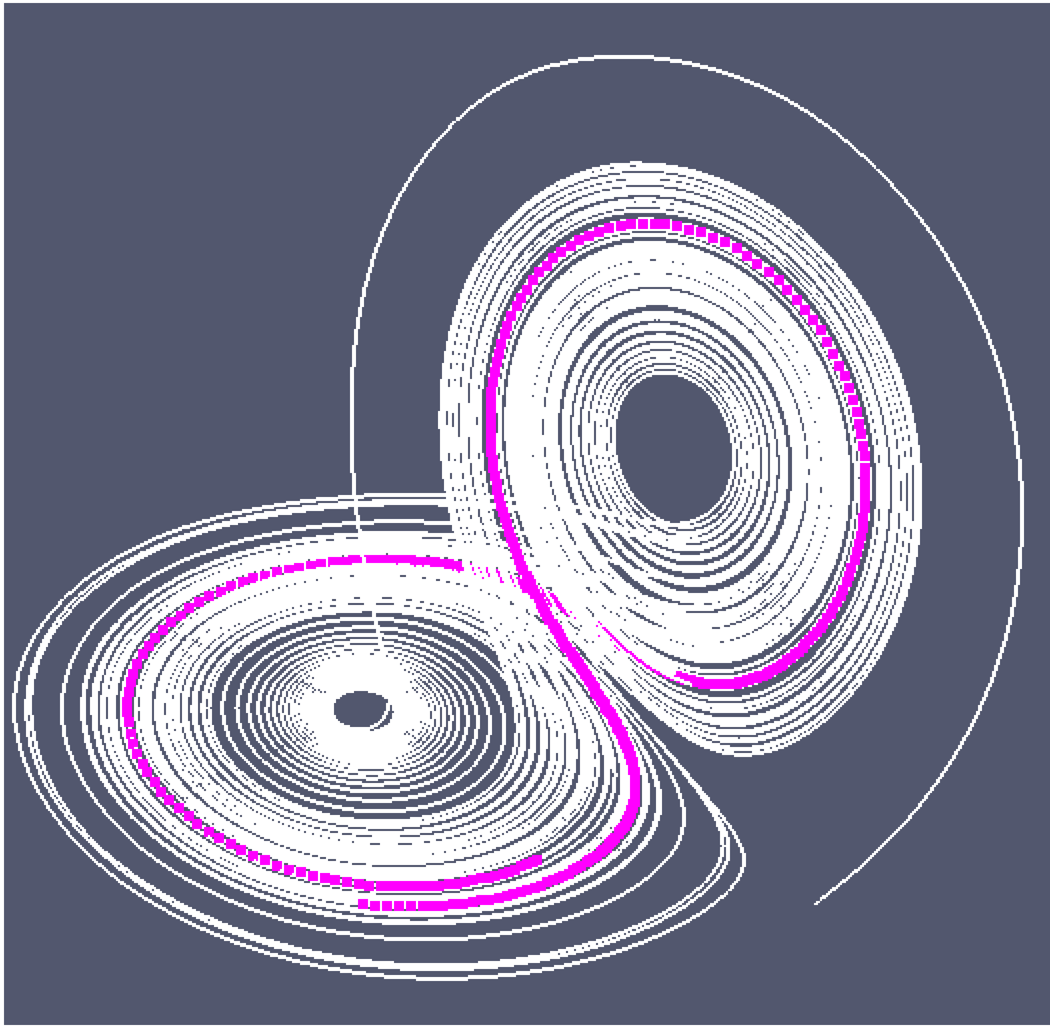
```
from paraview.util import SetOutputWholeExtent
SetOutputWholeExtent(self, [0, 29, 0, 19, 0, 19])
```

CSCS
Centro Svizzero di Calcolo Scientifico
Swiss National Supercomputing Centre

Example of the use of a python package

import scipy
from scipy import integrate

And create a vtkPolyData object to view



- Google for source code with
- “lorenz python laprise”

Grid to Table translation

```
# Get a Programmable filter
# set the output type to "vtkTable"

table = self.GetTableOutput()
pd = self.GetInput().GetPointData()

for i in range(pd.GetNumberOfArrays()):
  table.AddColumn(pd.GetArray(i))
```

Summary

- Python scripts are a much better representation of the pipeline than the older state files (*.pvsm)

- I recommend you learn at least the basics, to reload a given configuration, in a more portable manner

- Python is the only interface to run ParaView in batch mode