Imperial College London

Fluidity Training

Running Fluidity and Visualising the Results

Simon Mouradian

11-13 November 2015

Imperial College London

Outline

Running

Output

- Filetypes and tools
- The stat file
- Paraview
- Python

Running Fluidity

```
$ <trunk>/bin/fluidity my.flml
```

If fluidity is installed on the system:

```
$ fluidity my.flml
```

Common options:

-l (Create log file for each process)

-v N (Verbosity level, default level 0, N can be 0,1 or 2)

-h (Help)

Example:

```
$ fluidity -l -v2 my.flml
```

Output Filetypes

Today we will look at two types:

- Stat file (.stat)
- Unstructured VTK file (.vtu or .pvtu)

Fluidity also has a third filetype:

- Detectors (.detectors)

You may also have log files:

- fluidity.log-*
- fluidity.err-*

Tools

- Statplot
- Paraview
- Python
  - fluidity.statparser
  - vtktools

Copy “Top Hat” example

We’re going to copy and run a ready–made example:

First, to prepare the files...

```
cp -rv /scratch/examples/top_hat .
cd top_hat
make clean
make preprocess
```

Run "Top Hat" example

Type "`ls`" to see the `.flml` and mesh `line.msh` files. There are three separate set-up files and one mesh.

```
$ fluidity -l -v2 top_hat_cg.flml &
```

After running Fluidity, the `.vtu` files contain the results. Each `.vtu` contains the output at a single time–level.

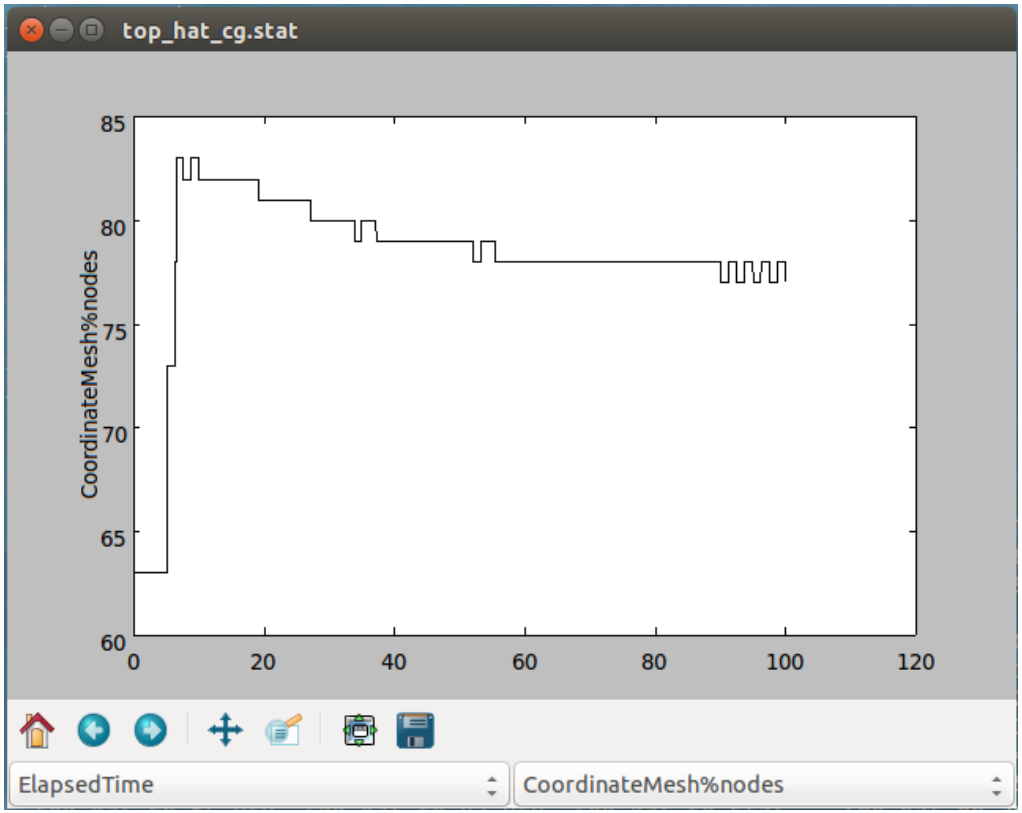
Imperial College London

The stat file

- Bespoke data file type
- Various tools to read and process these data
- Either ASCII or binary

Imperial College London

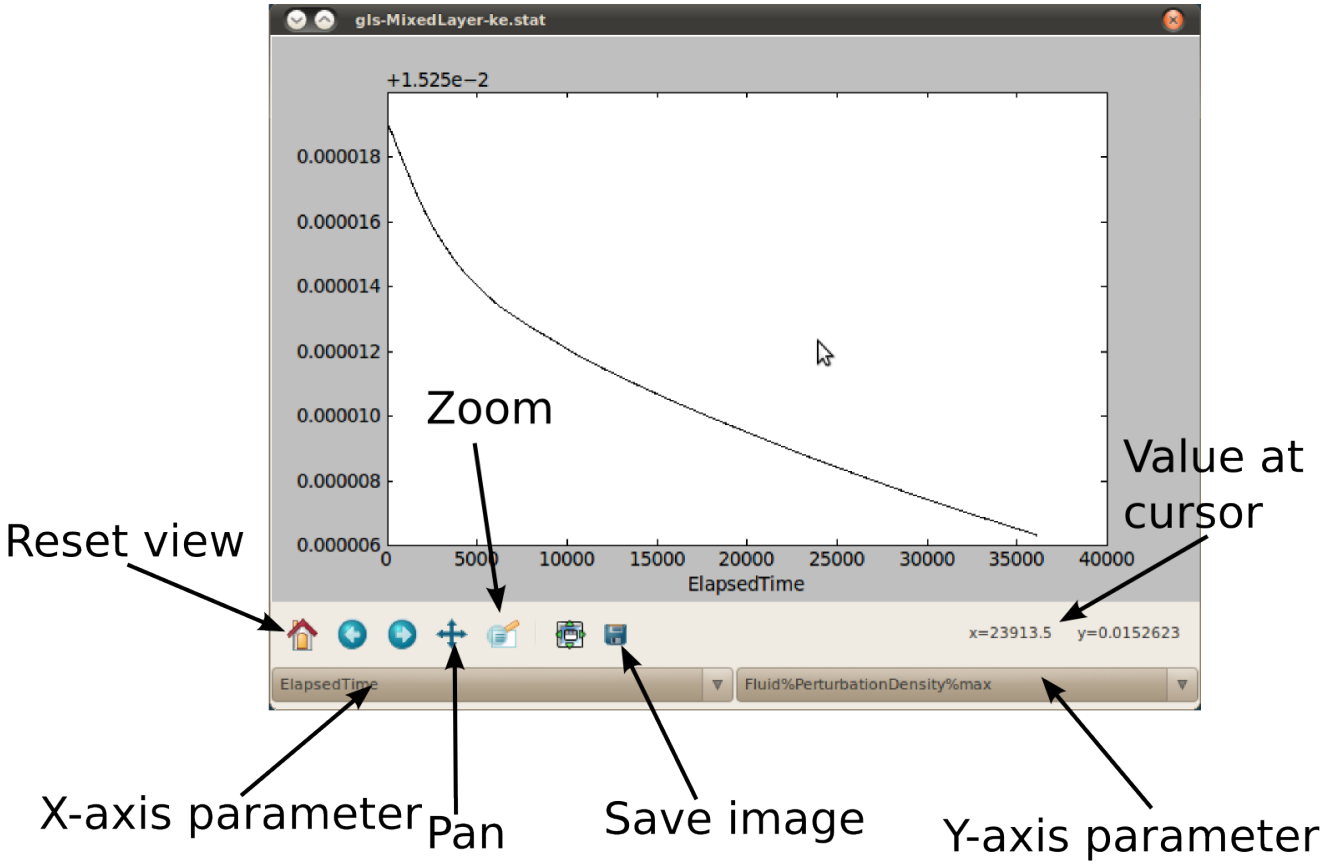
Statplot




```
$ statplot top_hat_cg.stat
```

Imperial College London

Statplot



Statplot keys

- s - scatter plot
- l - line plot
- r - refresh data
- R - refresh data, but keep current bounds
- x - switch x-axis from linear to log or vice versa
- y - switch y-axis from linear to log or vice versa
- q - quit (note: **no warnings!**)

Statplot example

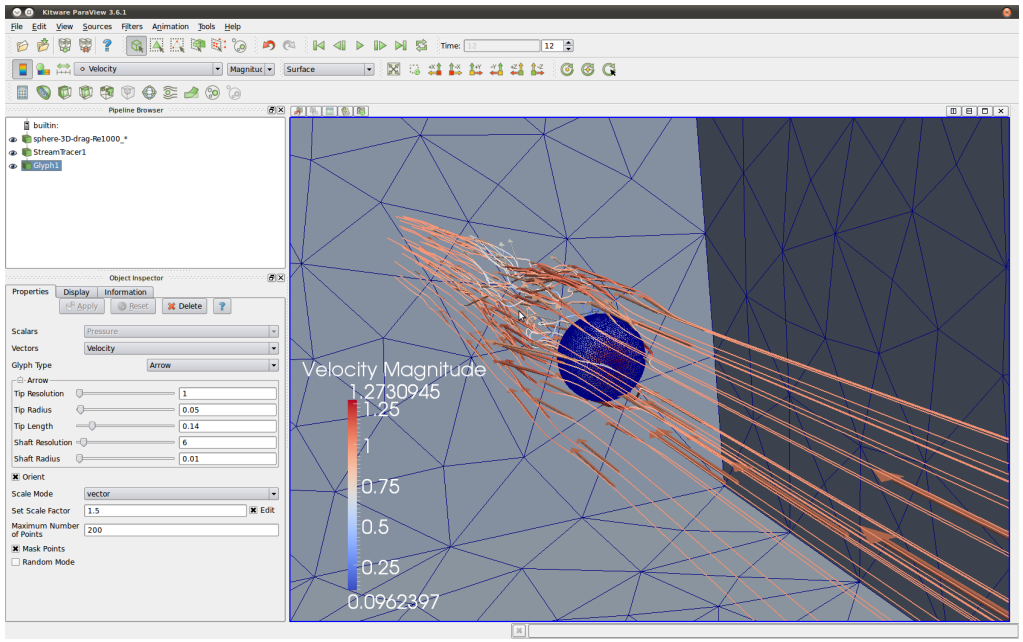
Open the stat file at from your advection problem
Things to try:

- Switch between scatter plot and line plot views
- Change the graph to show the number of elements through the run
- Plot tracer maximum against time
- Zoom in and save a small part of the plot to file

Imperial College London

Paraview

Open–source scientific visualisation software from KitView



This can be obtained on your own machine using:

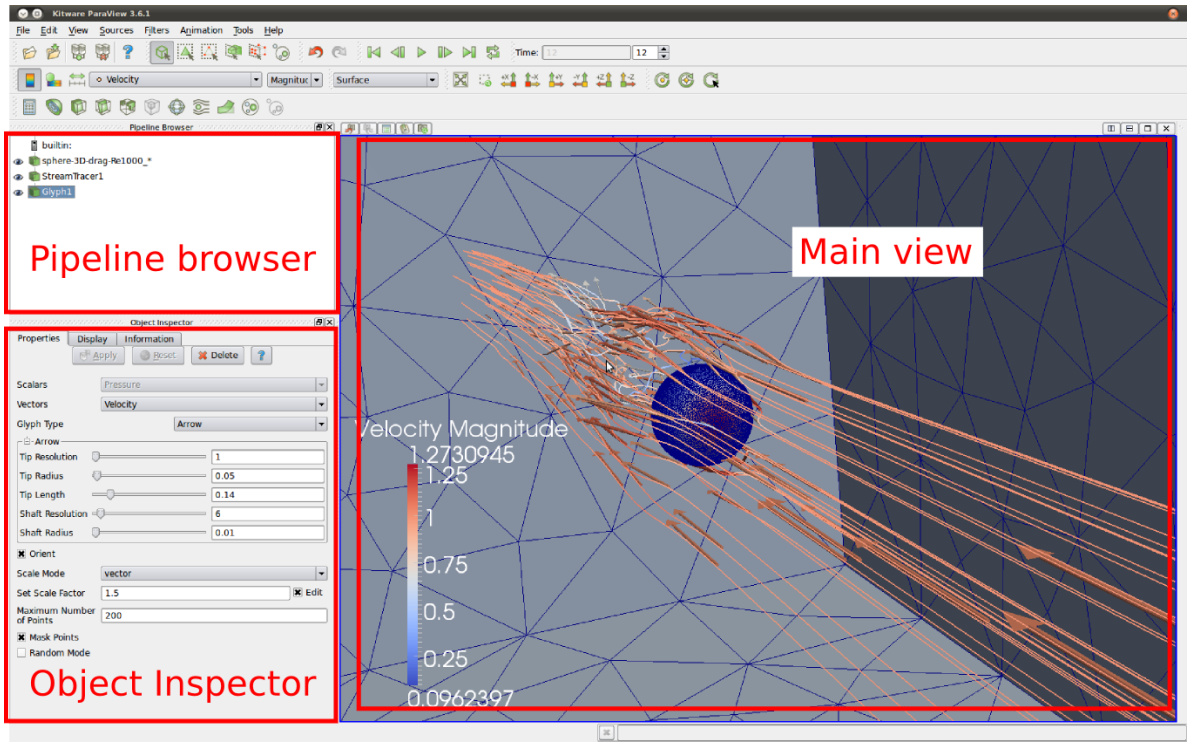
`$ sudo apt-get install paraview` ← But not now!

AMCG

Imperial College London

Paraview: main window

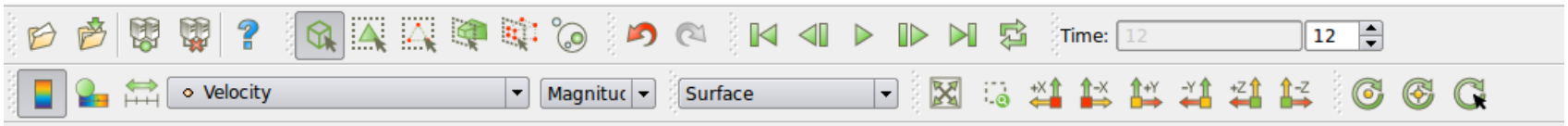
Launch by simply typing "`paraview &`"



AMCG

Imperial College London

Paraview: main window



AMCG

Imperial College London

Paraview: Copy data for visualisation

So we're going to copy across some ready-made data again to play around with.

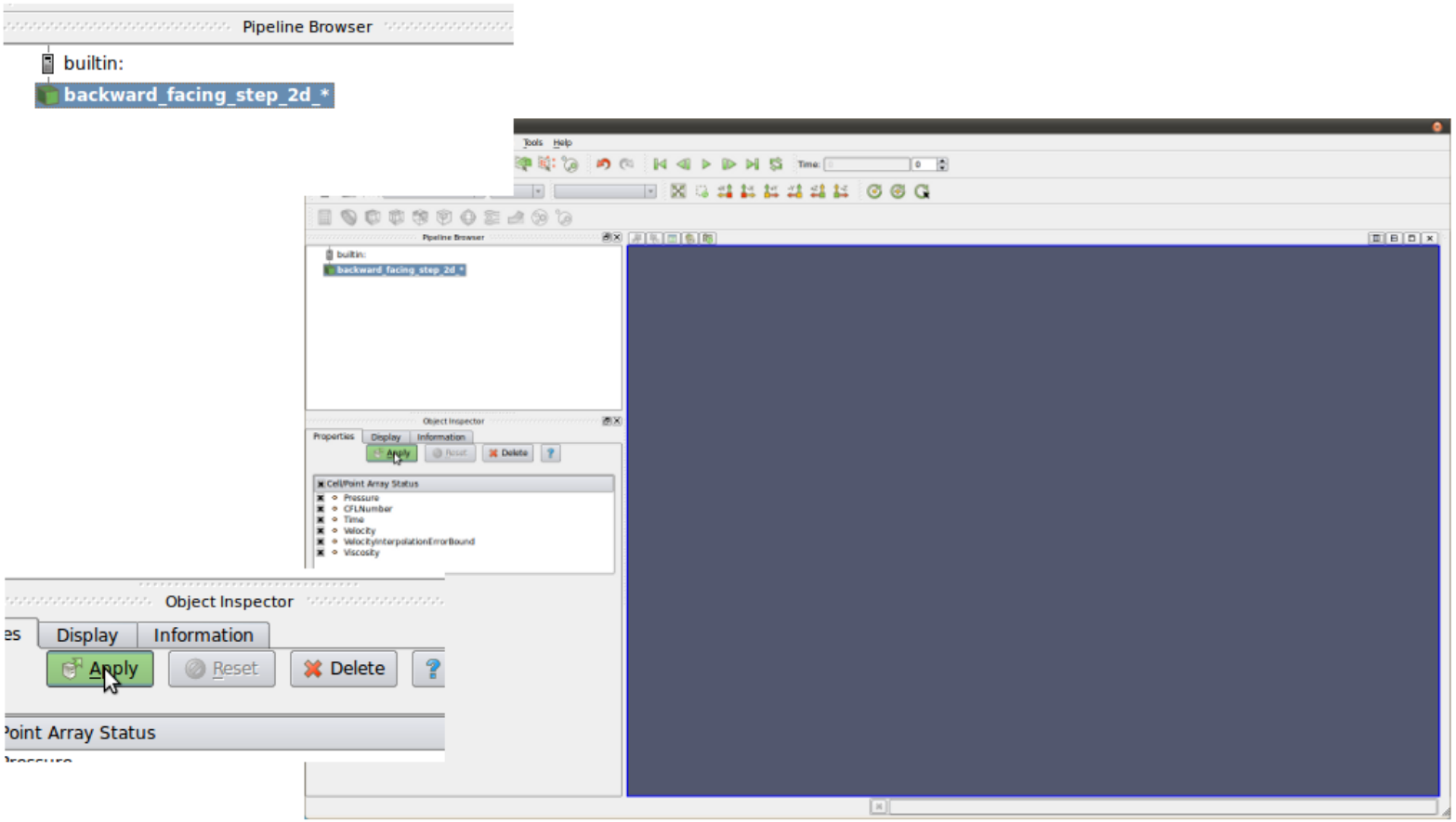
```
cd ..
ls
cp -rv /scratch/examples/backward_facing_step_2d .
cd backward_facing_step_2d
ls
```

Paraview: Open files

- Click on “OPEN” icon
- Navigate into “backward_facing_step_2d” and then “kepsilon”
- Scroll to the bottom
- Double-click on “...pvtu” file

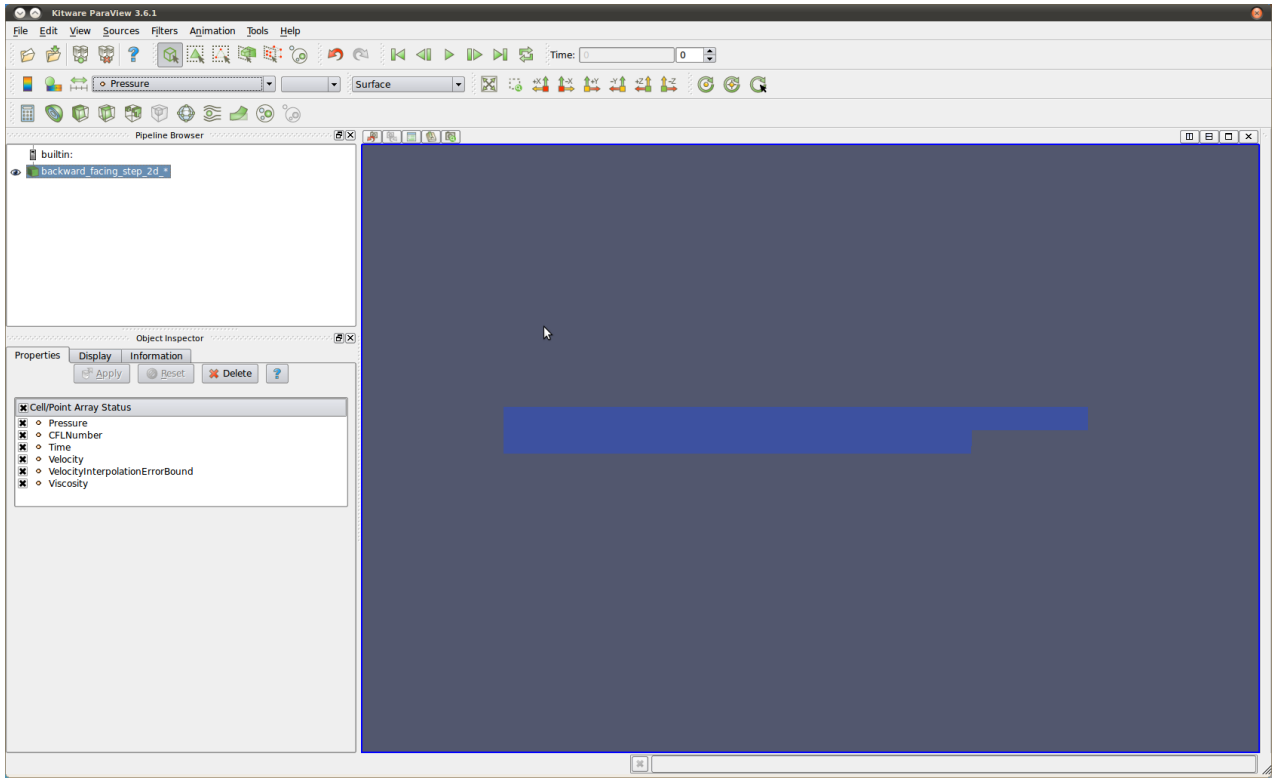
Imperial College London

Paraview



Imperial College London

Paraview



AMCG

Paraview

- Right click: Zoom-in and out
- Left-click: rotate
- Middle-button: move
- Use the drop-down menu to change field
- Use the green arrow keys to change the timestep

You are likely to use Paraview a lot for visualisation of data and learning to use it well is useful. Feel free to consult the Paraview tutorials for further practice and worked examples:

`www.paraview.org/Wiki/The_ParaView_Tutorial`

Python tools

- vtktools - read vtu files
- statparser - read stat files

Reading .stat file in Python

```
import fluidity_tools
stat = fluidity_tools.stat_parser(filename)
```

The stat object is a hierarchy of dictionaries!

```
stat["Fluid"].keys()
```

More info in the Fluidity manual section 9.4.2

Imperial College London

Reading .vtu files in Python

```
import vtktools
data = vtktools.vtu(example.vtu)
p = data.GetScalarField(Pressure)
```

p is now an array of the scalar field 'Pressure'

More info in the Fluidity manual section 9.3.1

Imperial College London

Post-processing Python script

We have some read-made post-processing script to analyse the data we've copied locally.

To view the script, type "`gedit postprocessor_2d.py &`"

If you don't know Python very well, don't worry, it's a very popular and easy language to pick-up.

To run the script, type:

```
make postprocess TYPE=reference
```