Imperial College London

Fluidity Training

Running Fluidity and Visualising the Results

Dave Robinson

6th November 2014

AMCG

Outline

Running

Output

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

Running Fluidity

Running source code:

```
[fluidity-directory]/fluidity my.flml
```

Running binary:

```
fluidity my.flml
fluidity -l my.flml
fluidity -l -v3 my.flml
```

fluidity --help

```
Revision: fluidity/4.1
Compile date: Nov 20 2012 10:16:12
OpenMP Support no
Adaptivity support yes
...
FEMDEM support no
Hyperlight support no


Usage: fluidity [options ...] [simulation-file]

Options:
 -h, --help
Help! Prints this message.
 -l, --log
Create log file for each process (useful for non-interactive testing).
 -v <level>, --verbose
Verbose output to stdout, default level 0
 -p, --profile
Print profiling data at end of run
This provides aggregated elapsed time for coarse-level computation
(Turned on automatically if verbosity is at level 2 or above)
 -V, --version
Version
```

Copy “Top Hat” example

We’re going to copy and run a ready-made example to show you what the process looks like.

First, to prepare the files...

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

Run “Top Hat” example

If you type “ls” then you can see three sets of .flml files and three “line.*” files. These are the three separate set-up files and one (triangle) mesh, respectively.

```
fluidity -v3 -l top_hat_cg.flml &
```

This simulation is one-dimensional so will complete almost immediately. If you now type “ls” again you will see a whole string of vtu files containing results data.

Filetypes

There are **two** main filetypes:

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

You may also have log files:

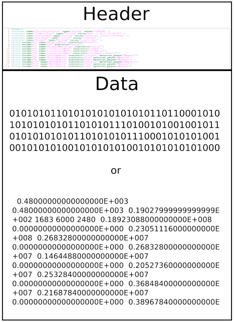
- fluidity.log.*
- fluidity.err.*

Tools

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

The stat file

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



Header

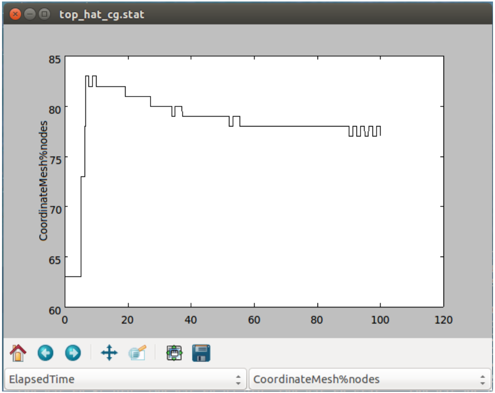
Data

```
0101010110101010101010101010110110001010
1010101010101101010101110100101001001011
010101010101101010101110001010101001
001010101001010101010010101010101000
```

or

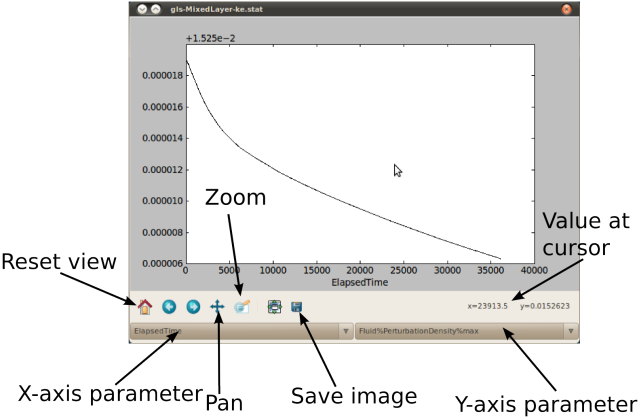
```
  0.48000000000000000E+003
0.48000000000000000E+003  0.19027999999999999E
+002 1683 6000 2480  0.18923088000000000E+008
0.00000000000000000E+000  0.23051116000000000E
+008  0.26832800000000000E+007
0.00000000000000000E+000  0.26832800000000000E
+007  0.14644880000000000E+007
0.00000000000000000E+000  0.20527360000000000E
+007  0.25328400000000000E+007
0.00000000000000000E+000  0.36848400000000000E
+007  0.21687840000000000E+007
0.00000000000000000E+000  0.38967840000000000E
```

Statplot



```
statplot top_hat_cg.stat
```

Statplot



gls-MixedLayer-ke.stat

x=23913.5 y=0.0152623

ElapsedTime

Fluid%PerturbationDensity%max

Zoom

Value at cursor

Reset view

X-axis parameter

Pan

Save image

Y-axis parameter

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

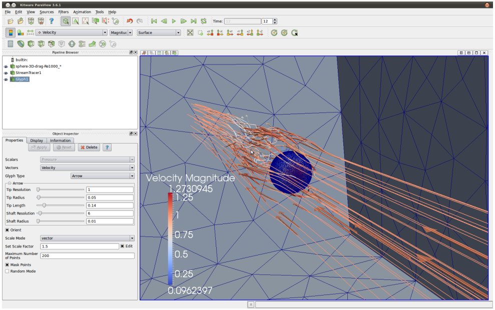
Open the stat file at from your advection problem

Things to try:

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

Paraview

Open-source scientific visualisation software from KitView



builtin:
sphere-3D-drag-Re1000_*
StreamTracer1
Glyph1

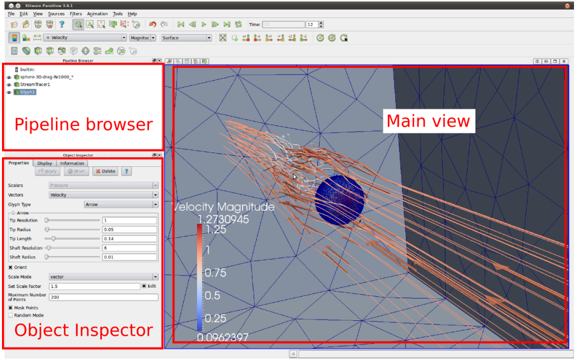
Scalars: Pressure
Vectors: Velocity
Glyph Type: Arrow
Arrow
Tip Resolution: 1
Tip Radius: 0.05
Tip Length: 0.14
Shaft Resolution: 6
Shaft Radius: 0.01
☒ Orient
Scale Mode: vector
Set Scale Factor: 1.5 ☒ Edit
Maximum Number of Points: 200
☒ Mask Points
□ Random Mode

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

Imperial College
London

Paraview: main window

Launch by simply typing “`paraview &`”



Time: 12

builtin:
sphere-3D-drag-Re1000_*
StreamTracer1
Glyph1

Scalars: Pressure
Vectors: Velocity
Glyph Type: Arrow

Arrow
Tip Resolution: 1
Tip Radius: 0.05
Tip Length: 0.14
Shaft Resolution: 6
Shaft Radius: 0.01

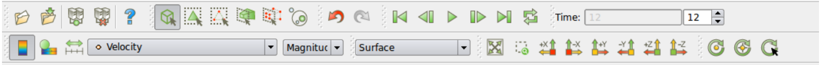
☒ Orient
Scale Mode: vector
Set Scale Factor: 1.5 ☒ Edit
Maximum Number of Points: 200
☒ Mask Points
☐ Random Mode

Pipeline browser

Main view

Object Inspector

Paraview: main window



Paraview: Copy data for visualisation

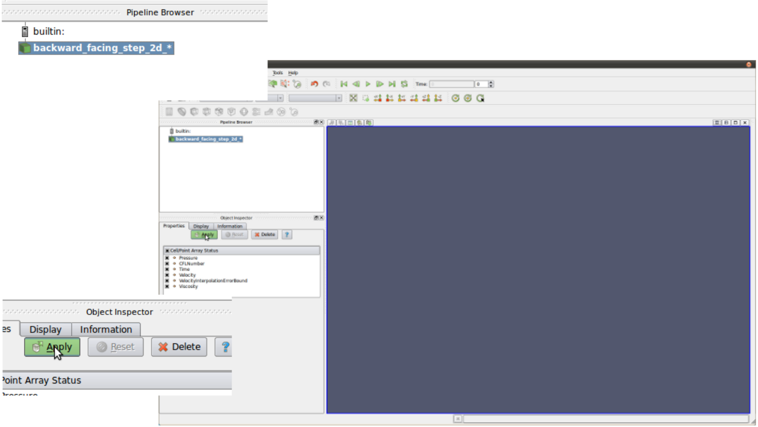
So we're going to copy across some ready-made data again to play around with.

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

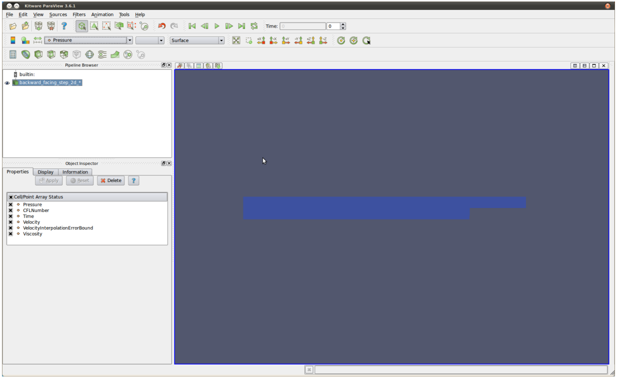
Paraview: Open files

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

Paraview


Pipeline Browser
builtin:
backward_facing_step_2d_*
Tools Help
Time:
Object Inspector
Properties
Display
Information
Apply
Reset
Delete
?
Cell/Point Array Status
Pressure
CFLNumber
Time
Velocity
VelocityInterpolationErrorBound
Viscosity
es
Point Array Status
Pressure

Paraview



Time: 0

Pressure

Surface

builtin:

backward_facing_step_2d_*

☒ Cell/Point Array Status

- ☒ ○ Pressure
- ☒ ○ CFLNumber
- ☒ ○ Time
- ☒ ○ Velocity
- ☒ ○ VelocityInterpolationErrorBound
- ☒ ○ Viscosity

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 a whole training course in itself. Please 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

Useful python modules

- numpy - numerical package, including arrays
- stats - linear regression, etc
- matplotlib - plotting 2- and 3-D

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
```