# OpenFOAM postprocessing and advanced running options

Tommaso Lucchini


Department of Energy
Politecnico di Milano

# The post processing tool: paraFoam

The main post-processing tool provided with OpenFOAM is the reader module to run with ParaView, an open-source, visualization application.

The module is compiled into 2 libraries, PV3FoamReader and vtkFoam, using version 3.3 (or higher) of ParaView supplied with the OpenFOAM release.

ParaView uses the Visualization Toolkit (VTK) as its data processing and rendering engine and can therefore read any data in VTK format.

OpenFOAM includes the foamToVTK utility to convert data from its native format to VTK format, which means that any VTK-based graphics tools can be used to post-process OpenFOAM cases. This provides an alternative means for using ParaView with OpenFOAM.

`paraFoam` is strictly a script that launches ParaView using the reader module supplied with OpenFOAM.

It is executed like any of the OpenFOAM utilities with the root directory path and the case directory name as arguments:

```
paraFoam [-case dir]
```

# Before starting

Before starting with the user should copy two tutorials in his own run folder:

```
cp -r $FOAM_TUTORIALS/simpleFoam/pitzDaily/ $FOAM_RUN/
cp -r $FOAM_TUTORIALS/dieselFoam/aachenBomb $FOAM_RUN/
```

SimpleFoam is an incompressible steady-state flow solver, while dieselFoam is dedicated to combustion process with spray injection.

The two cases should also be executed:

```
run
blockMesh -case pitzDaily
simpleFoam -case pitzDaily
blockMesh -case aachenBomb
dieselFoam -case aachenBomb
```

The user should care about reducing the endTime and the writeInterval to have immediately some calculations to post process for the pitzDaily case.

For the aachenBomb case the user should switch off the chemistry to reduce the computational time.

# Viewing the mesh

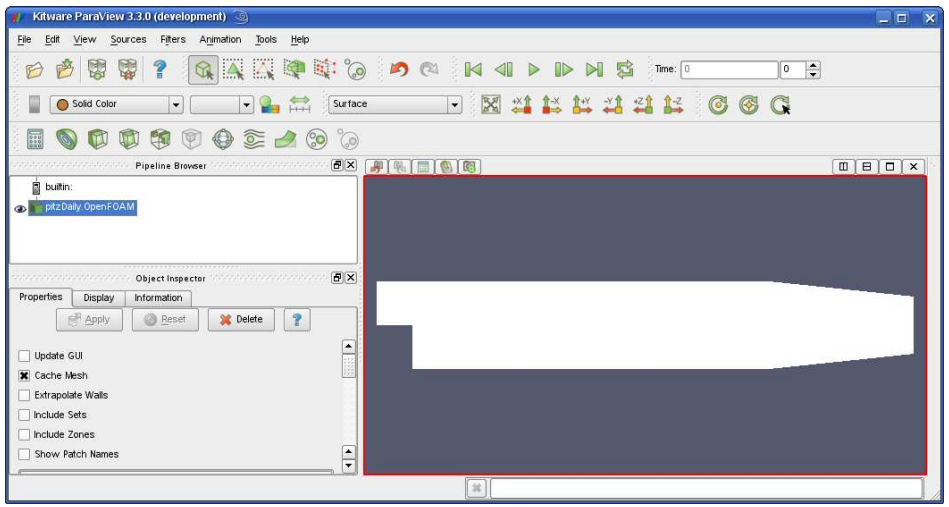
The mesh can be viewed only in paraFoam since there is no pre-processing tool to view the mesh.

We shall start with the pitzDaily case of the simpleFoam tutorial:

```
paraFoam -case pitzDaily
```

You can also run `paraFoam` directly in the `pitzDaily` case directory without any additional argument.

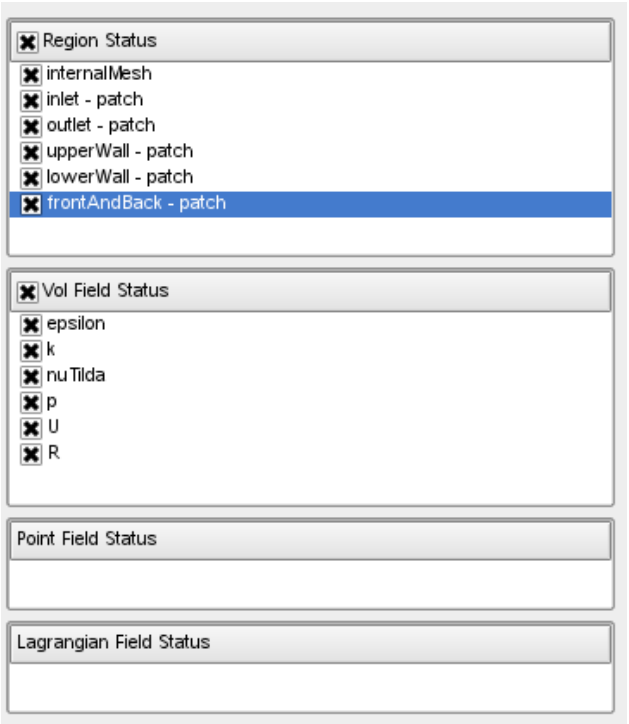
Click the Apply button which will bring up an image of the case geometry in the image display window.



# Panels

In the Properties panel it is possible to choose what you can visualize:

- Region status (internal mesh and patches).
- Vol field status (available geometric fields).
- Point fields.
- Lagrangian fields.



# The post processing tool: paraFoam

When `paraView` is launched, visualization is controlled by:

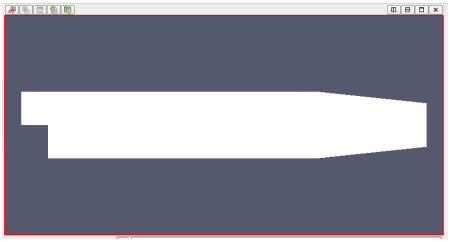
- **Pipeline browser**: lists the modules opened in ParaView, where the selected modules are highlighted in blue and the graphics for the given module can be enabled/disabled by clicking the eye button alongside
- **Object inspector** consisting in three different panels:
  - **Properties panel**: it contains the input selections for the case, such as times, regions and fields.
  - **Display panel** controls the visual representation of the selected module, e.g. colors.
  - **Information panel** gives case statistics such as mesh geometry and size.
- **Current time control panel** allows to selects the simulation time to be visualized.

ParaView operates a tree-based structure in which data can be filtered from the top-level case module to create sets of sub-modules.

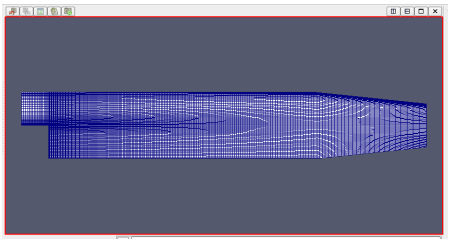
# Viewing the mesh

Go to the **Display** panel and in the *style* window select:

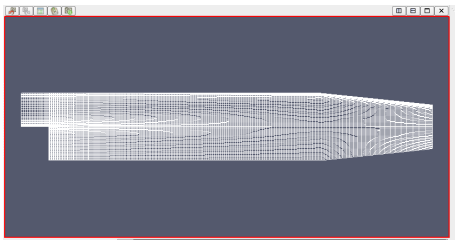
- Surface



- Surface with edges



- Wireframe of surface



To visualize the simulated domain and its surface mesh. Play with the `solid color` and `edge color` menus to select the colors you prefer.

# Viewing fields: display panel

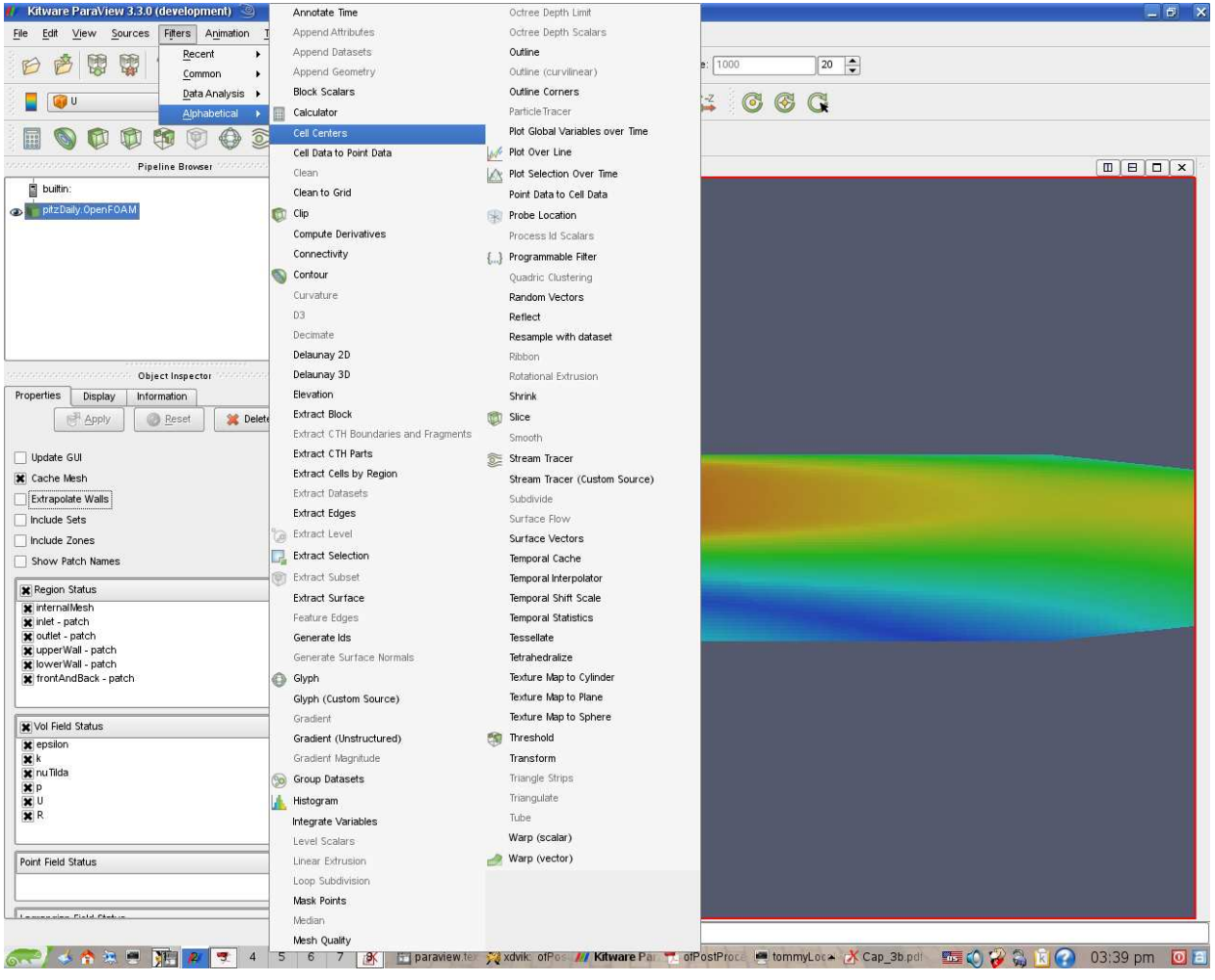
The Display panel contains the settings for visualizing the data/fields for a given case module. You can choose to color the mesh with a constant *solid color* or with a colour range representing the field values. The same operation can be done directly in the **Active variable controls variable** menu located on left-top of the screen.

- Activate the legend visibility button to see the field data range.
- The magnitude or the single components of a vector field can be visualized.
- Data range is not automatically rescaled, so the user should rescale the data to range when necessary.
- The *Edit color map* window makes possible to choose the color range and appropriate legend font colors and size.
- the underlying mesh can be represented by selecting *Surface With Edges* in the *Representation* menu.
- the image can be made translucent by editing the value in Opacity (1 = solid, 0 = invisible).
- the activated field can be translated, scaled, rotated with respect the other ones.

In the color windows it is possible to select the field that the user wants to plot.

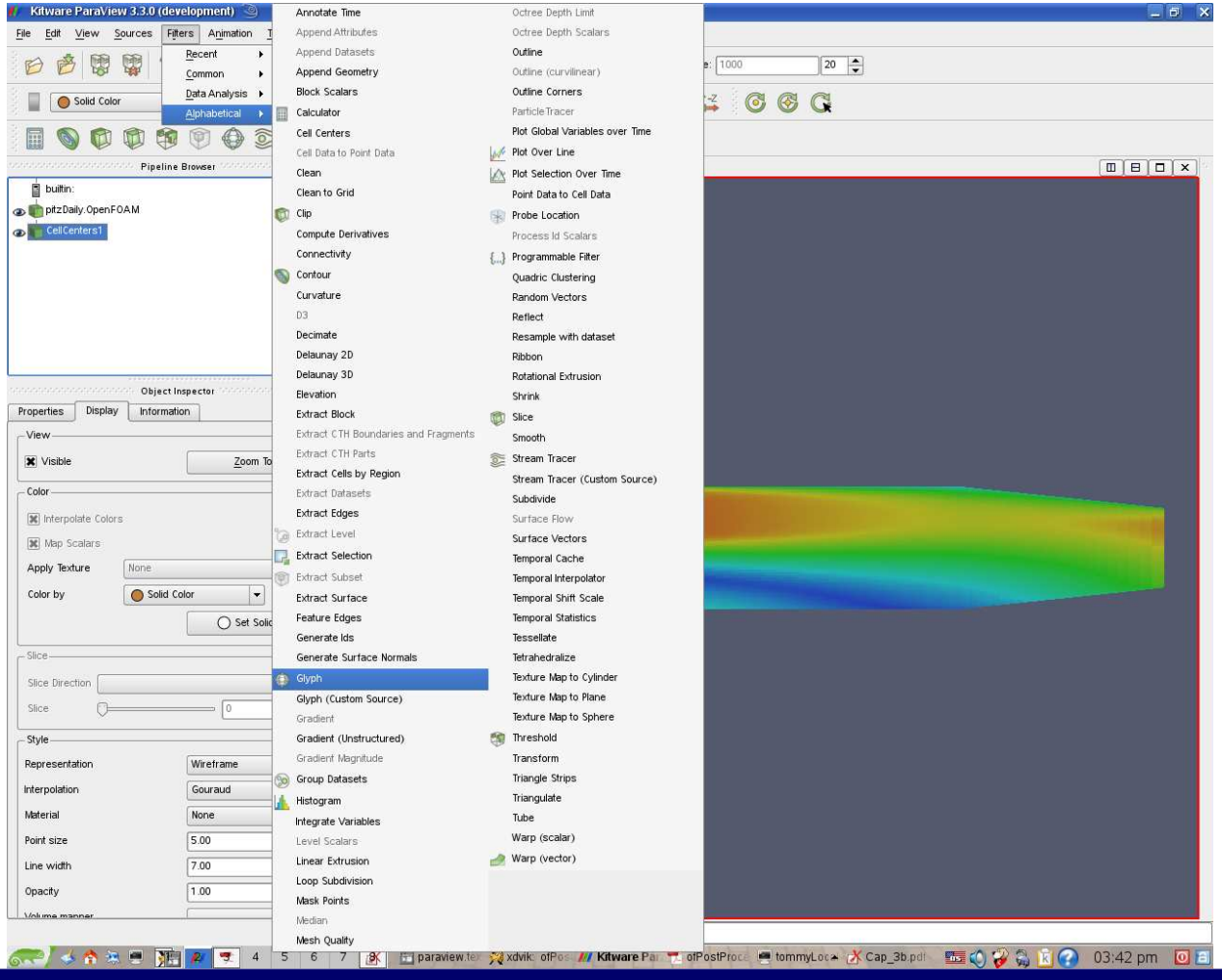
# Viewing fields: vectorFields (U)

The user must first create a filter to select the quantities in the cell centers:



# Viewing fields: vectorFields (U)

Starting from the cell center filtered field, the user must select the `Glyph` filter and then accept:



# Viewing fields: glyphs (U)

The filter reads an Input and offers a range of Glyphs for which the Arrow provides a clear vector plot images.

In the Orient/Scale window, the most common options for Scale Mode are: `vector`, where the glyph length is proportional to the vector magnitude, whereas selecting `off` each glyph has the same length.

The `Set scale Factor` option controls the base length of the glyphs.

It is possible to select the maximum number of Glyphs to be displayed. Putting a number lower than the cell number can speed up the visualization.

Different gliph types can be used and for each one of them different options can be chosen to optimize the visualization.

It is possible to select the Glyph filter directly from the ParaView toolbar:

# Viewing fields: glyphs (U)

Visualization of the velocity magnitude contours:

- Firstly, a field called `magU` has to be created with the `foamCalc` utility:

  ```
  foamCalc mag '(U)' -case pitzDaily -time 1000
  ```

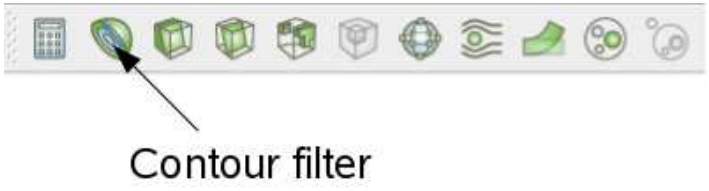
  `magU` will appear in the `1000` directory and it is a `volScalarField` containing the velocity magnitude.

- Refresh the `Vol Fields` window and magU will appear.

- Visualize `magU`.

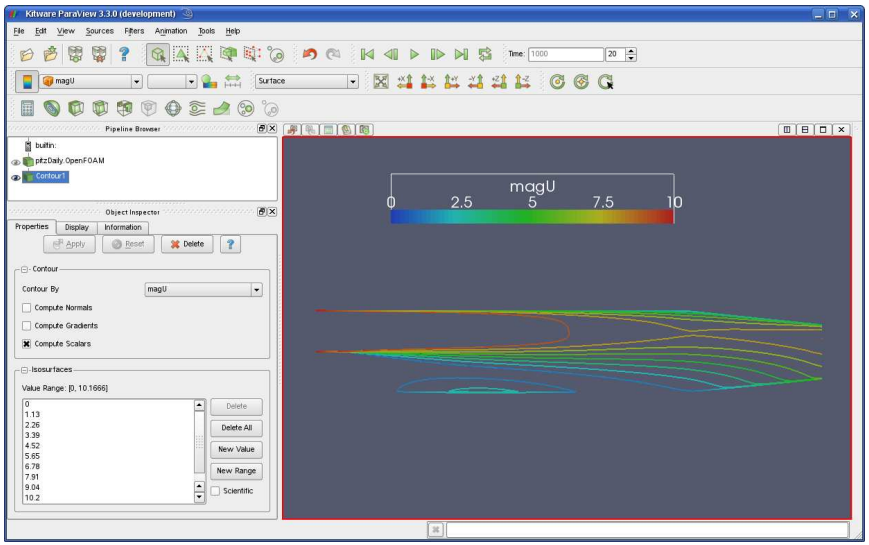
# Viewing fields: contour plots

A contour plot is created by selecting Contour from the Filter menu at the top menu bar.



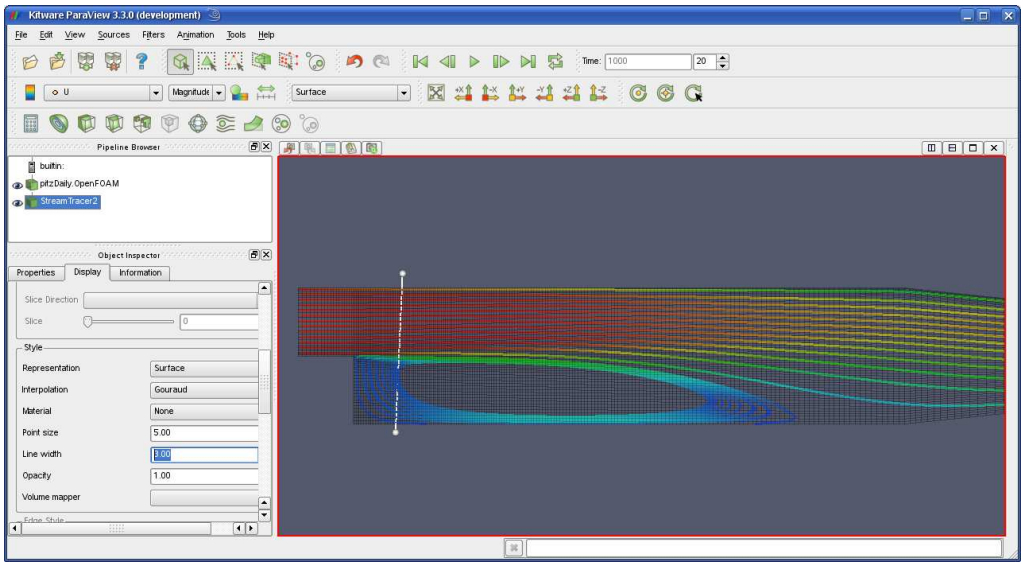

The filter acts on a given module so that, if the module is the 3D case module itself, the contours will be a set of 2D surfaces that represent a constant value, i.e. isosurfaces.

The Parameter panel allows to choose the field to contour and the value range for the isosurfaces.



# Viewing fields: streamlines

- The `Stream Tracer` filter is used to create the streamlines. The tracer Seed window specifies a distribution of tracer points over a Line or Point Cloud.
- The `Stream Tracer` windows provides additional settings on streamlines length and creation (integration step, method, . . . ).

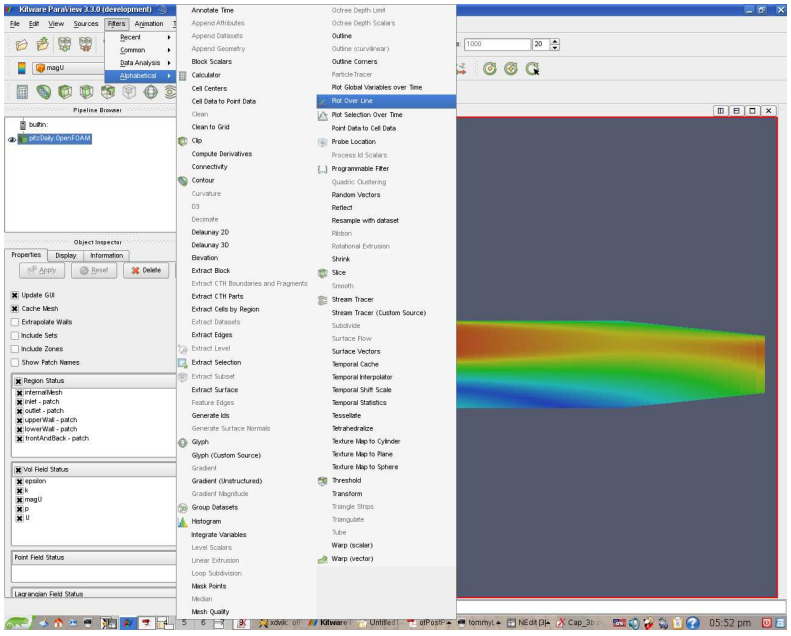


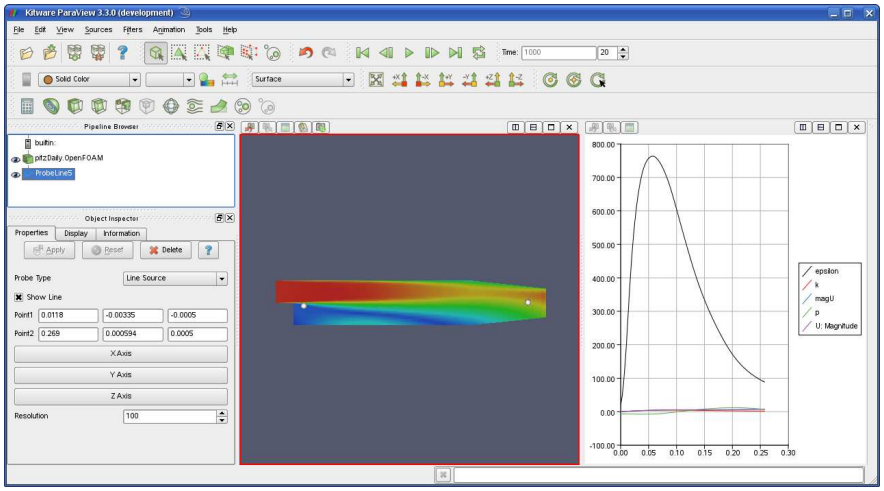
# Viewing fields: probes

To plot a graph in ParaView, the users can select `Plot Over Line` from the Filter menu.

All the fields data will sampled along a line specified in the `Properties` panel. The number of sampling points can be chosen with the `resolution` option.

Fields to be plotted and plot options can be selected in the `Display` panel when the graph is selected.





The graph can be saved onto a csv file by selecting `Save Data` in the `File` menu.

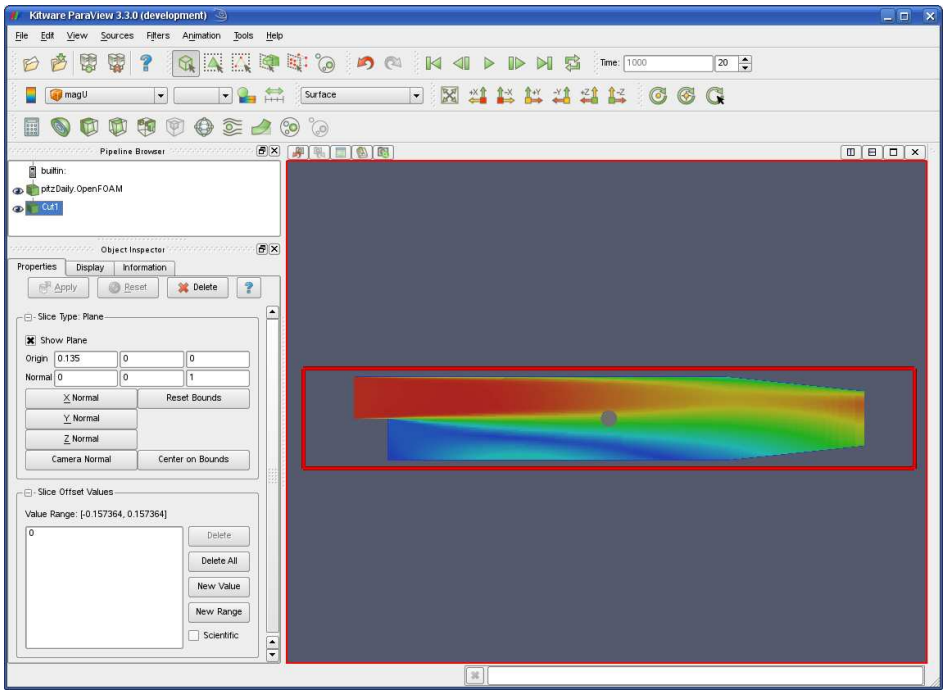
# Cut planes

To create a contour plot across a plane rather than producing isosurfaces, the user must first use the `Slice` filter to create the cutting plane, on which the contours can be plotted.

The `Slice` filter allows the user to specify a cutting Plane in the Properties menu by a center and normal/radius respectively.

The user can then run the Contour filter on the cut plane to generate contour lines.

Multiple cut planes can be generated using the **New value** or the **New range of values** sub panel.



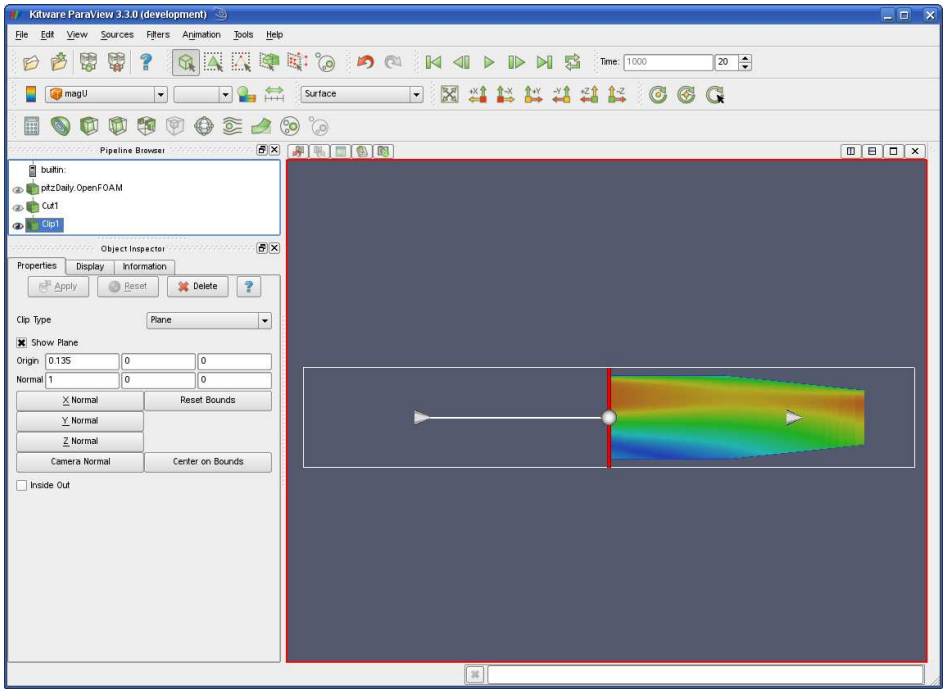
# Clip filter

The `Clip` filter works similarly to the `Cut` one, but keeps the mesh and field information on one side of the cutting plane.

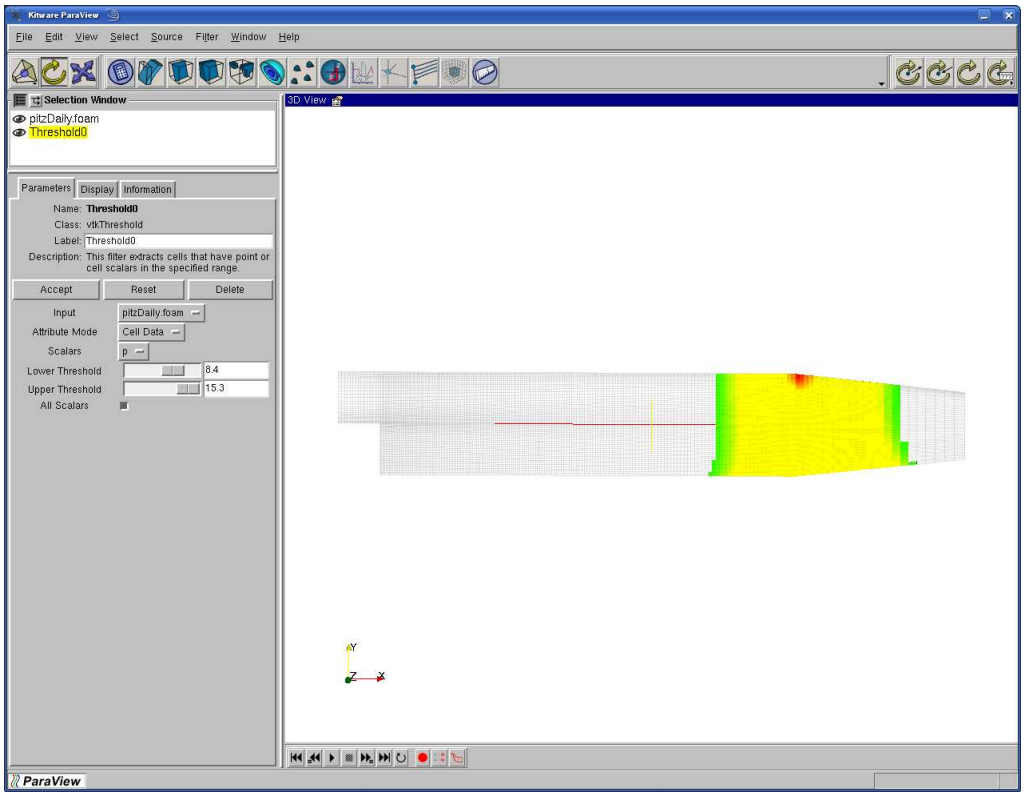
The `Clip` filter allows the user to specify a cutting Plane, Sphere, Box or on the basis of a value for a scalar field.

The clip selection can be inverted by activating the **Inside out** option.

The **Show Plane** option allows the user to activate or deactivate the visibility of the cutting plane, sphere or box.



# Threshold




The `Threshold` filter allows to visualize only the cells having the values of the selected field within a specified range

The range can be specified by means of moving the **Upper threshold** and **lower threshold** bars.

# Visualizing Lagrangian

To visualize the Lagrangian particles the case must be converted into the paraview format.

The user should run the application `foamToVTK` on the case containing the Lagrangian particle tracking (in our case the aachenBomb tutorial for the `dieselFoam` application).

```
foamToVTK -case aachenBomb
```

This command will generate an additional folder named `VTK` inside the case directory.

The user must start paraview typing the following command:

```
paraview
```

The case can be opened clicking on the file menu, open, and selecting the VTK directory. By clicking on the `*.vtk` file the case will be opened.

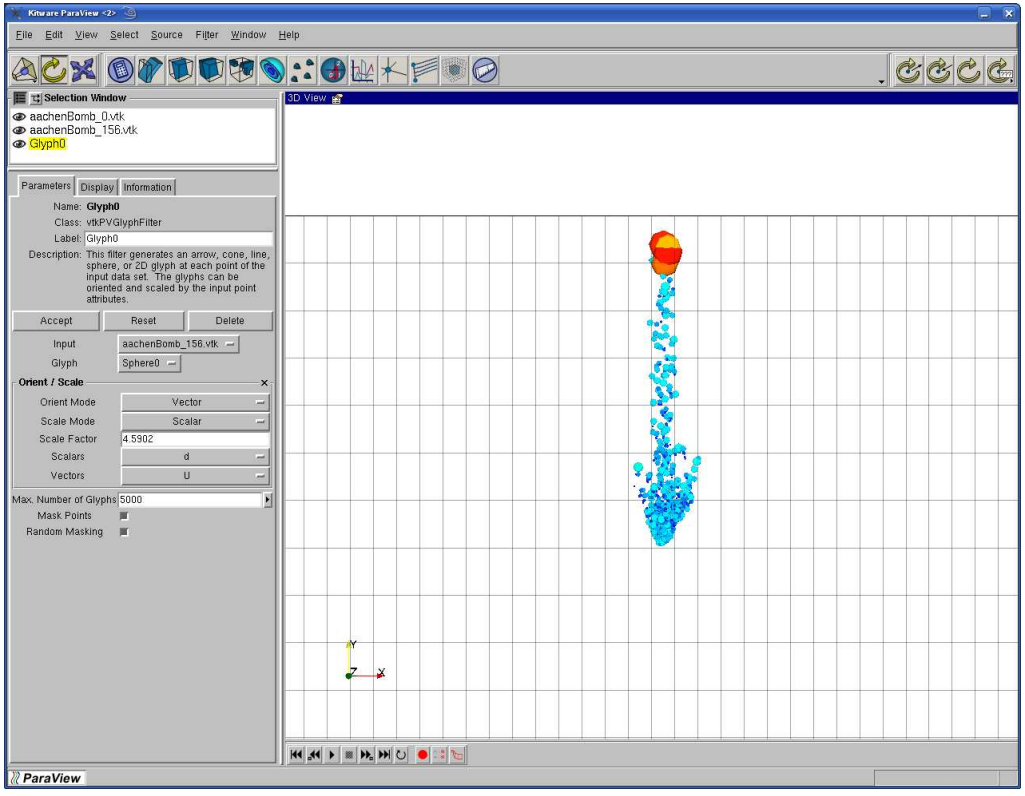
The user may load all the time steps by clicking on the timesteps button and selecting the Add all choice on the new window.

The use must then open the files contained in the **lagrangian** directory inside the VTK folder.

# Visualizing Lagrangian

To display the particles the user must filter the lagrangian field with the `Glyph` filter.

Then the scalar option must be selected in the scale mode window along with the spheres as glyphs



# Viewing sets with paraview

The user may create sets (pintSet, faceSet and cellSet) to perform certain additional operation over the fields, i.e. moving certain points or applying a source term only in a certain area.

A set, i.e. a cellSet, can be created in OpenFOAM using the command: `cellSet`

`$FOAM_APP/utilities/mesh/manipulation/cellSet`

This application creates a list of cells on the base of data read from the `cellSetDict` file.

The user shall try to create a new cellSet in the pitzDaily tutorial case:

```
run
cp -r $FOAM_TUTORIAL/simpleFoam/pitzDaily .
cd pitzDaily/system
cp $FOAM_UTILITIES/mesh/manipulation/cellSet/cellSetDict .
```

# Viewing sets with paraview

The user must modify the cellSetDict file in the following way:

```
// Name of set to operate on
name myCellSet;

// One of clear/new/invert/add/delete|subset/list
action new;

topoSetSources
(
    // Cells with cell centre within box
    boxToCell
    {
       box   (-0.05 -0.25 -0.001) (0.1 0.25 0.001);
    }
);
```

To create the cell set the user must run the cellSet application:

```
cellSet -case pitzDaily
```

# Viewing sets with paraview

Once the cellSet has been created run the `foamToVTK` command on the pitzDaily case using the `cellSet` option:

```
foamToVTK -case pitzDaily -cellSet myCellSet
```

To know all the available option of the `foamToVTK` application type `foamToVTK --help` and press enter. It will be displayed the usage of that application:

```
Usage: foamToVTK [-noZero] [-surfaceFields] [-ascii]
[-region region name]  [-faceSet faceSet name] [-nearCellValue]
[-pointSet pointSet name] [-noLinks] [-case dir]
[-excludePatches patches to exclude] [-allPatches]
[-cellSet cellSet name] [-parallel] [-noFaceZones]
[-fields fields] [-constant] [-noPointValues] [-latestTime]
[-noInternal] [-time time]  [-help] [-doc] [-srcDoc]
```

To visualize the cellSet the user must open the VTK folder (it works also form a already opened paraFoam windows) in the pitzDaily directory and select the file with the name of the cellSet: `myCellSet_*.*`.

# Data manipulation

It possible to rely on data manipulation utilities which OpanFOAM provides inside its utilities folder.

OpenFOAM utilities can also be copied and customized to create user defined utilities.

An example could be the `ptot` utilities which calculates the total pressure starting from the velocity and pressure field:

```
$FOAM_UTILITIES/postProcessing/miscellaneous/ptot
```

It is also possible to export data to be visualized with other post-processing tools:

```
>cd $FOAM_UTILITIES/postProcessing/dataConversion
>ls
foamDataToFluent  foamToEnsightParts  foamToGMV  smapToFoam
foamToEnsight     foamToFieldview9    foamToVTK
```

# The sample utility

OpenFOAM provides the sample utility to sample field data for plotting on graphs.

The sampling locations are specified for a case through a `sampleDict` dictionary located in the case `system` directory.

The data can be written in a range of formats including well-known graphing packages such as Grace/xmgr, gnuplot and jPlot.

The sampling can be executed by running the utility application `sample` according to the application syntax:

```
sample -case  pitzDaily
```

The sampleDict dictionary can be generated by copying a detailed `sampleDict` from the sample utility folder:

```
$FOAM_UTILITIES/postProcessing/miscellaneous/sample
```

# The sampleDict file

The sampleDict contains entries to be set according to the user needs:

```
interpolationScheme cellPoint;
```

The choice for the interpolationScheme can be one of the following:

- cell : use cell-centre value only, constant over cells, no interpolation at all.
- cellPoint : use cell-centre and vertex values.
- cellPointFace : use cell-centre, vertex and face values.

Vertex values are determined from neighboring cell-centre values whereas face values determined using the current face interpolation scheme for the field (linear, gamma, etc.).

```
writeFormat raw;
```

The choice for the write format depends on the application used to plot the series. The user can choose one of the following:

- xmgr;
- jplot;
- gnuplot;
- raw;

# The sampleDict file

The `sampleDict` file contains a `sets` subdictionary which defines where the fields should be sampled:

```
sets
(
    line // name of the set
    {
        type uniform; // type of the set
        axis             x;
        start            (-0.0206 0 0);
        end              (0.29 0 0);
        nPoints          1000;
    }

);
```

In this subdictionary the user can specify the type of sampling method, the name of samples and how to write point coordinate plus other parameters depending on the method chosen.

# The sampleDict file

The user may choose one of the following sampling methods:

- **uniform**: evenly distributed points on line.
- **face**: one point per face intersection.
- **midPoint**: one point per cell, in between two face intersections.
- **midPointAndFace**: combination of face and midPoint.
- **curve**: specified points, not necessary on line, uses tracking.
- **cloud**: specified points, uses findCell.

```
lineX1;
```

The name entry is typically a filename. After running the `sample` application the user will find a new folder in the case directory named `sets`.

Each data file is given a name containing the field name, the sample set name, and an extension relating to the output format, including .xy for raw data, .agr for Grace/xmgr and .dat for jPlot.

```
axis        distance;
```

Specifies how to write point coordinates, options are:

- `x/y/z`: x/y/z coordinate only.
- `xyz`: three columns.
- `distance`: distance from start of sampling line (if uses line) or distance from first specified sampling point.

# The sampleDict file

There are also some type specific options:

- `start` and `end` coordinate for uniform, face, midPoint and midPointAndFace.
- `nPoints` number of sampling points for uniform.
- list of coordinates for curve and cloud.

The fields list contains the fields that the user wishes to sample:

```
fields
(
    p
    magU
    Ux
);
```

# Probes

To plot values of fields as functions of the time the user can rely on the new functionObject class.

This is a modular plugin that evaluates at every time step the specified field at the specified location.

Function objects are created adding them to a function subdict in the controlDict file.

The user may find an example of how to modify the controlDict file in `$FOAM_TUTORIAL/oodles/pitzDaily/` case

This function cannot parse the set of functions as the sample utility does, however in the case of the Ux component extraction the user can easily do that for the U probed field.

# Probes

```
functions
(
    probes1
    {
        type probes; // Type of functionObject
        // Where to load it from (if not already in solver)
        functionObjectLibs ("libsampling.so");
        probeLocations // Locations to be probed. runTime modifiable!
        (
            (0.1778 0.0253 0.0)
        );
        // Fields to be probed. runTime modifiable!
        fields
        (
            p
        );
    }
);
```

During the job run it will be created, inside the case directory, a folder named probes containing the probed values.

# foamLog

There are limitations to monitoring a job by reading the log file, in particular it is difficult to extract trends over a long period of time.

The foamLog script is therefore provided to extract data of residuals, iterations, Courant number etc. from a log file and present it in a set of files that can be plotted graphically.

The user must run the application in the following manner:

```
simpleFoam -case pitzDaily > log
```

Where the log file can be named arbitrarily
The script can be executed in different ways:

```
foamLog - extracts xy files from Foam logs.

Usage: foamLog [-n][-s] <log>
           extracts xy files from log
       foamLog -l <log>
           lists but does not extract
       foamLog -h
           for a help message
```

A `logs` subdirectory containing the xy files is created in the directory where the command is run.

# foamLog

Once the `log` file has been created the `foamLog` must be executed to generate the required statistics:

```
foamLog [-n] [-s] <log>
```

A `logs` subdirectory containing the xy files is created in the directory where the command is run.

Each file has the name `<var>_<subIter>` where `<var>` is the name of the variable specified in the log file and `<subIter>` is the iteration number within the time step.

Those variables that are solved for, the initial residual takes the variable name `<var>` and final residual takes `<var>FinalRes`. By default, the files are presented in two-column format of time and the extracted values:

```
cd logs
ls

contCumulative_0  epsilonIters_0  kIters_0    Time_0       UyFinalRes_0
contGlobal_0      executionTime_0 p_0         Ux_0         UyIters_0
contLocal_0       foamLog.awk     pFinalRes_0 UxFinalRes_0
epsilon_0         k_0             pIters_0    UxIters_0
epsilonFinalRes_0 kFinalRes_0     Separator_0 Uy_0
```

# foamCalc

The `foamCalc` is a powerful post-processing application. It calculates different properties of a `GeometricField`:

- vector magnitude, components.
- gradient magnitude, divergency.

Example of applications:

```
foamCalc mag U -case pitzDaily
foamCalc components U -case pitzDaily
foamCalc magGrad p -case pitzDaily
```

# Acknowledgements

Dr. Gianluca Montenegro from Politecnico di Milano is gratefully acknowledged for its contribution to the slides presented in this work.