Introduction to open source CFD

Gijsbert Wierink

Christian – Doppler Laboratory on Particulate Flow Modelling
Johannes Kepler University | Linz | Austria

Industrial Dust Recycling
Outline

- Introduction
- Meshing and running
- More details on running cases
- Validation test case
- Summary

Disclaimer

This course is offered non-commercially as part of course work at the Department of Particulate Flow Modelling (PFM) at the Johannes Kepler University (JKU) in Linz, Austria. This offereing is not approved or endorsed by OpenCFD Limited, the producer of the OpenFOAM software and owner of the OPENFOAM® and OpenCFD ® trade marks.

INTRODUCTION

Introduction
Open source CFD

- Open source software allows the user to modify and further develop the code

- We will be using **OpenFOAM**
  - Free and open source (GPL)
  - Produced by OpenCFD Ltd. (ESI)
  - Modular C++ framework to solve PDEs (FVM)

- Other open source options, see e.g.

  www.cfd-online.com/Wiki/Codes#Free_codes

Aim of the course

- Aim: get you up to speed with OpenFOAM
  - OpenFOAM confidence
  - Linux confidence
  - C++ confidence
- Plan
  - Get used to the OpenFOAM and linux environment
  - Learn to modify OpenFOAM code
  - Learn to create your own models
  - Learn about post-processing

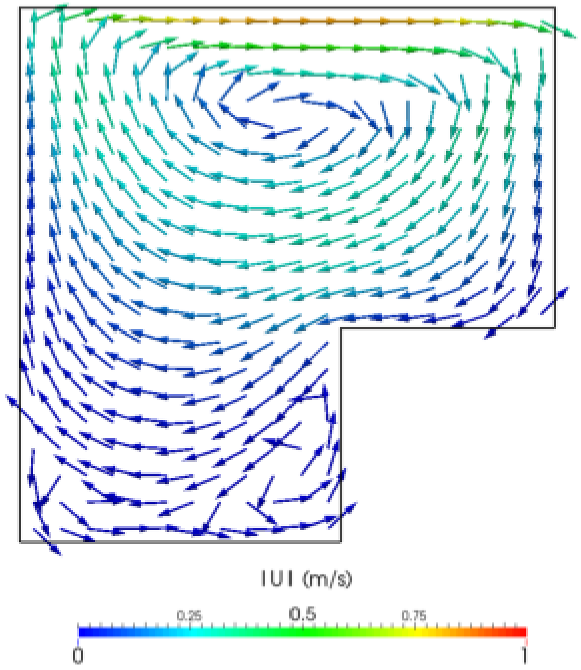
MESHING AND RUNNING

Lid-driven cavity
A basic example case

Aims:

- generate a mesh using `blockMesh`
- run a solver
- view the results in ParaView

- Incompressible, transient solver `icoFoam`
- Lid driven cavity in 2D

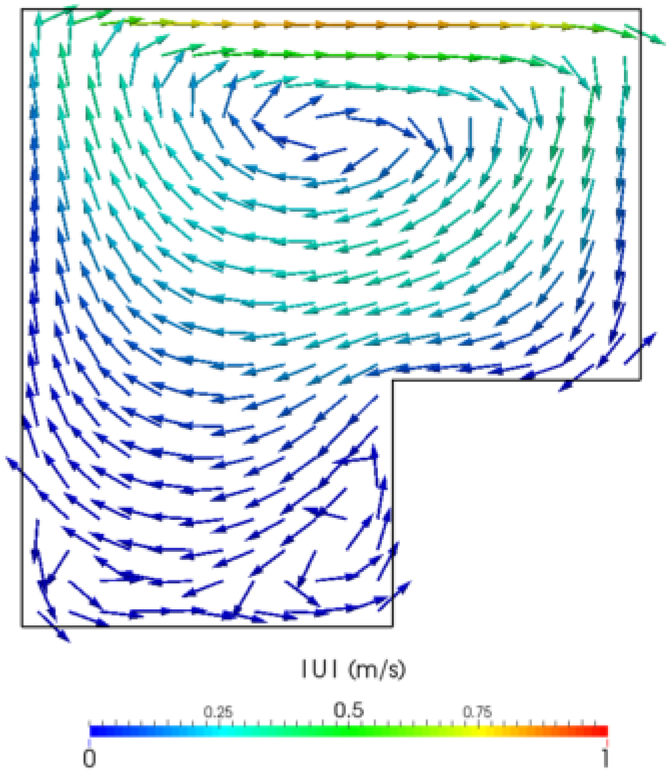


```
> run
> cp -r $FOAM_TUTORIALS/incompressible/icoFoam/cavityClipped .
> cd cavityClipped
```

Lid-driven cavity
Case structure

- Case structure

```
.
├── 0
│   ├── p
│   └── U
├── constant
│   ├── polyMesh
│   │   └── blockMeshDict
│   └── transportProperties
└── system
    ├── controlDict
    ├── fvSchemes
    ├── fvSolution
    └── mapFieldsDict
```

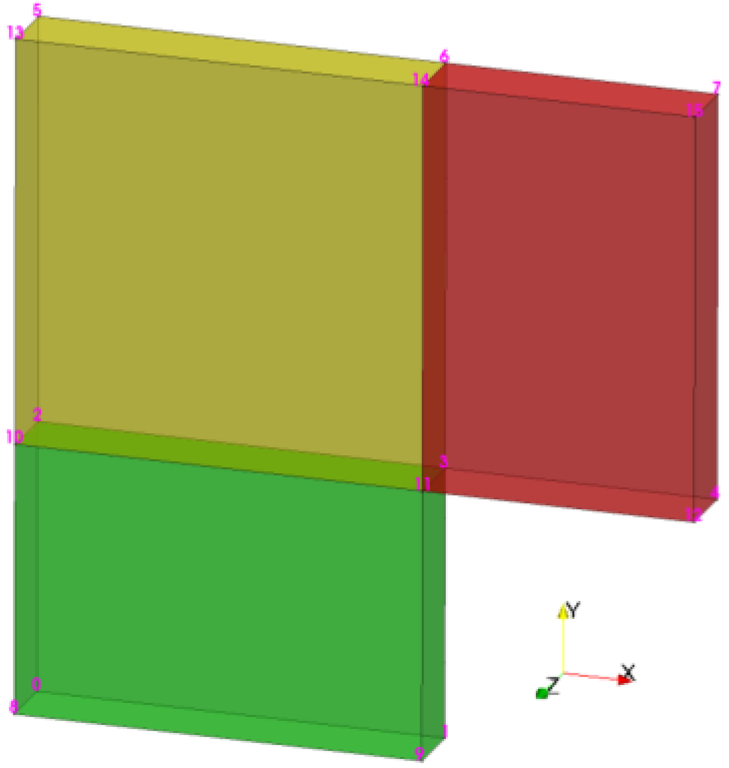


- Open `blockMeshDict`

```
> gedit constant/polyMesh/blockMeshDict &
```

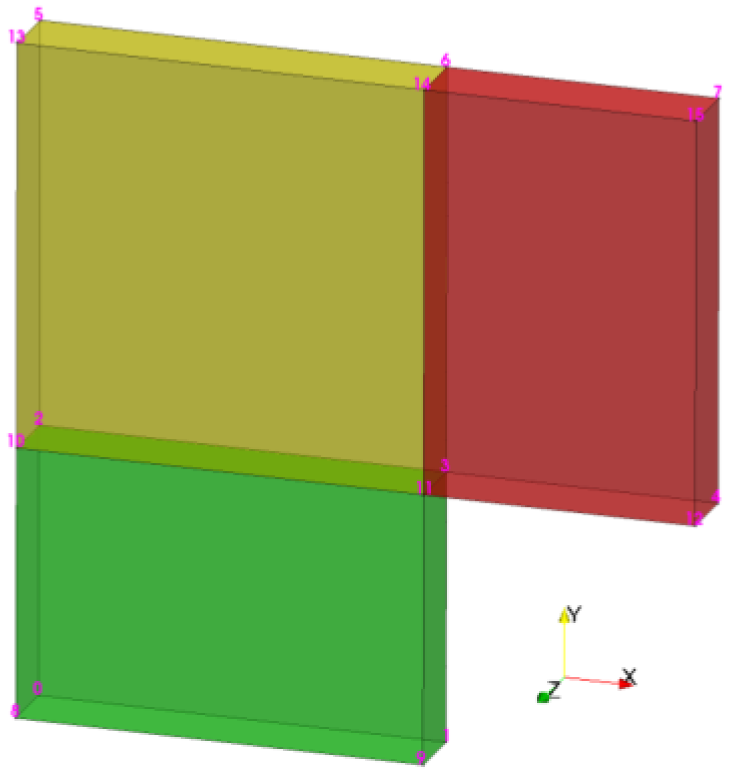
Lid-driven cavity
blockMeshDict

```
// Conversion factor to m
convertToMeters 0.1;

// List of vertices
vertices
(
    (0   0 0  ) //  0
    (0.6 0 0  ) //  1
    ...
    (0.6 1 0.1) // 14
    (1   1 0.1) // 15

);

// Block definition
blocks
(
    hex (0 1 3 2 8 9 11 10) // shape and vertices
        (12 8 1)            // nr cells in x-y-z
    simpleGrading (1 1 1)   // grading in x-y-z
    ...
);
```



Lid-driven cavity
blockMeshDict

```
...
// List of edges for curves
edges
(
);

// Boundary definition
boundary
(
    lid                    // Patch name
    {
        type wall;         // Patch type
        faces
        (
            (5 13 14 6) // Face from vertices
            (6 14 15 7)
        );
    }
    ...
);

// Merge duplicate face pairs
mergePatchPairs
(
);
```



Lid-driven cavity
Mesh and view results

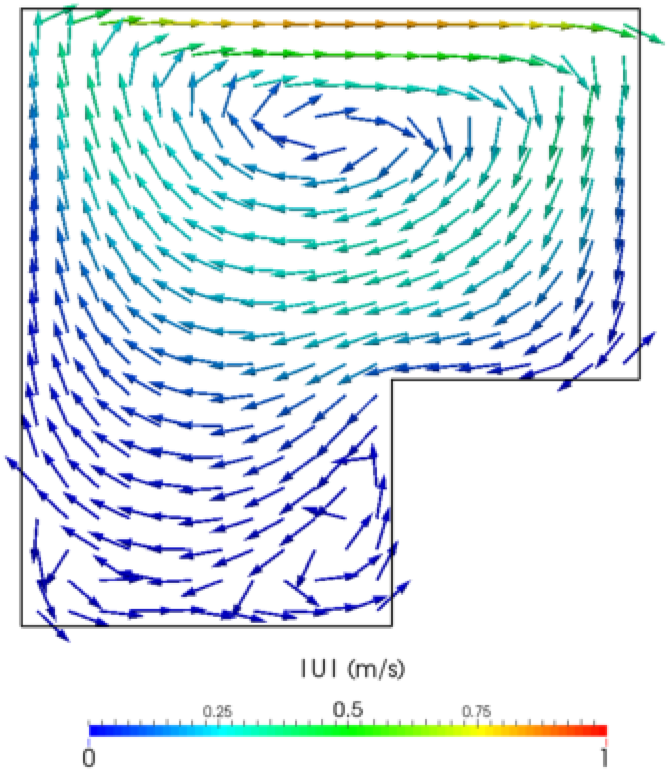
- Create the mesh and run the solver

```
> blockMesh
> icoFoam
```

- View the results with ParaView

```
> paraFoam
```

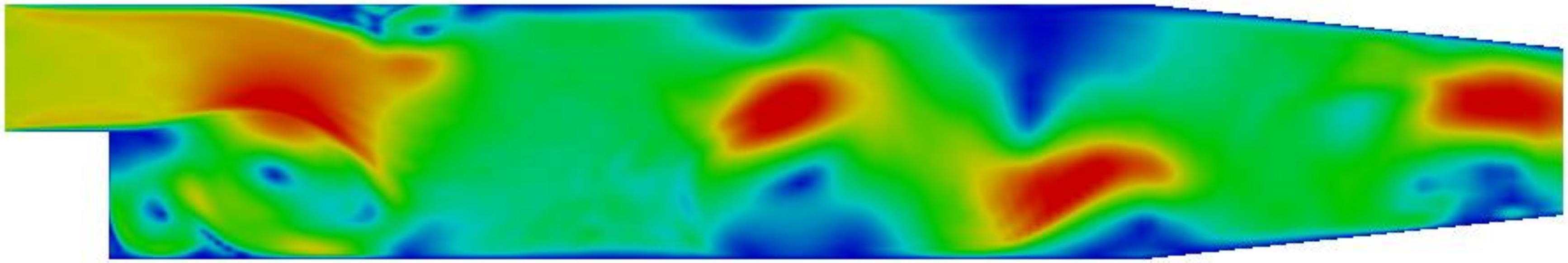
- Create a vector plot using ParaView's filters



MORE CASE RUNNING

More case details
LES pitzDaily case

- Run the incompressible LES pitzDaily case



```
> run

> cp -r $FOAM_TUTORIALS/incompressible/pisoFoam/les/pitzDaily .

> cd pitzDaily

> blockMesh

> pisoFoam |& tee log.pisoFoam
```

More case details
Solver output

```
Time = 0.03185Courant Number mean: 0.055869 max: 0.271596

smoothSolver:  Solving for Ux, Initial residual = 0.00205101, Final residual
    = 1.03065e-06, No Iterations 2
smoothSolver:  Solving for Uy, Initial residual = 0.00195554, Final residual
    = 1.10399e-06, No Iterations 2
GAMG:  Solving for p, Initial residual = 0.0342705, Final residual = 0.00260759,
    No Iterations 16
time step continuity errors : sum local = 1.57233e-07, global = 2.05609e-08,
    cumulative = -1.0282e-06
GAMG:  Solving for p, Initial residual = 0.00943142, Final residual =
    9.37652e-07, No Iterations 14
time step continuity errors : sum local = 7.25275e-11, global = 5.25955e-12,
    cumulative = -1.0282e-06
smoothSolver:  Solving for k, Initial residual = 0.000743129, Final residual =
    2.69896e-07, No Iterations 2
bounding k, min: 0 max: 9.78355 average: 0.142138

ExecutionTime = 262.76 s ClockTime = 266 s

fieldAverage fieldAverage1 output:    Calculating averages
```

More case details
Solver output

```
Time = 0.03185Courant Number mean: 0.055869 max: 0.271596

smoothSolver:  Solving for Ux, Initial residual = 0.00205101, Final residual
    = 1.03065e-06, No Iterations 2
```

- Controlled by the U entry in system/fvSolution

```
"(U|k|B|nuTilda)„
{
    solver          smoothSolver;
    smoother        GaussSeidel;
    tolerance       1e-05;
    relTol          0;
}
```

More case details
Solver output

```
GAMG:  Solving for p, Initial residual = 0.0342705, Final residual = 0.00260759,
    No Iterations 16
time step continuity errors : sum local = 1.57233e-07, global = 2.05609e-08,
    cumulative = -1.0282e-06
```

- Controlled by the p entry in system/fvSolution

```
P
{
    solver          GAMG;
    tolerance       1e-06;
    relTol          0.1;
    ...
}
```

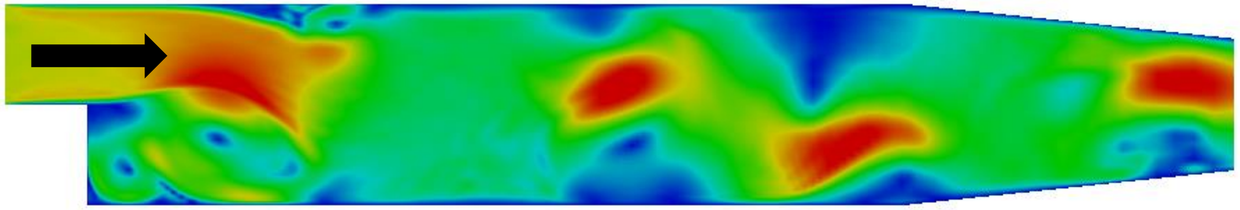
More case details
Solver output

```
Time = 0.03185Courant Number mean: 0.055869 max: 0.271596

smoothSolver:  Solving for Ux, Initial residual = 0.00205101, Final residual
    = 1.03065e-06, No Iterations 2
smoothSolver:  Solving for Uy, Initial residual = 0.00195554, Final residual
    = 1.10399e-06, No Iterations 2
GAMG:  Solving for p, Initial residual = 0.0342705, Final residual = 0.00260759,
    No Iterations 16
time step continuity errors : sum local = 1.57233e-07, global = 2.05609e-08,
    cumulative = -1.0282e-06
GAMG:  Solving for p, Initial residual = 0.00943142, Final residual =
    9.37652e-07, No Iterations 14
time step continuity errors : sum local = 7.25275e-11, global = 5.25955e-12,
    cumulative = -1.0282e-06
smoothSolver:  Solving for k, Initial residual = 0.000743129, Final residual =
    2.69896e-07, No Iterations 2
bounding k, min: 0 max: 9.78355 average: 0.142138

ExecutionTime = 262.76 s ClockTime = 266 s

fieldAverage fieldAverage1 output:    Calculating averages
```

More case details
A quick look at the boundary conditions

- The boundary and initial conditions can be found in 0/<fieldName>
- In essence the boundary conditions are

$U = 10$
$\nabla p = 0$



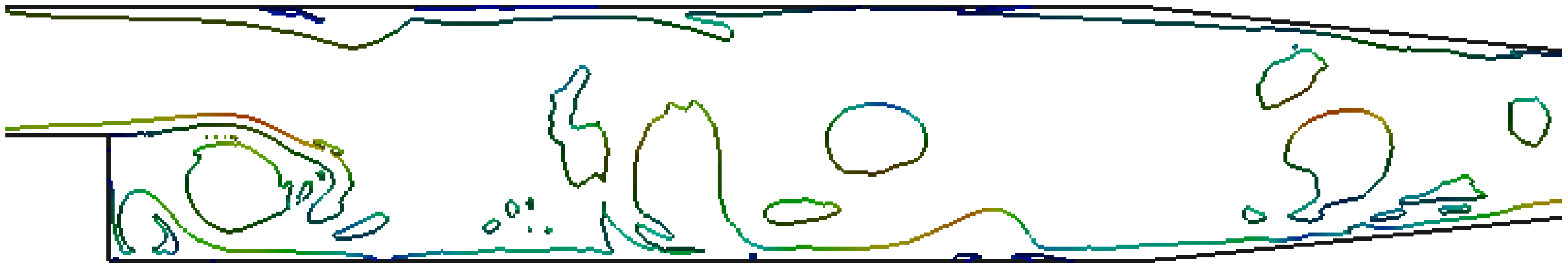
$\nabla U = 0$
$p = 0$

... with some sofistication ...

More case details
Post-processing

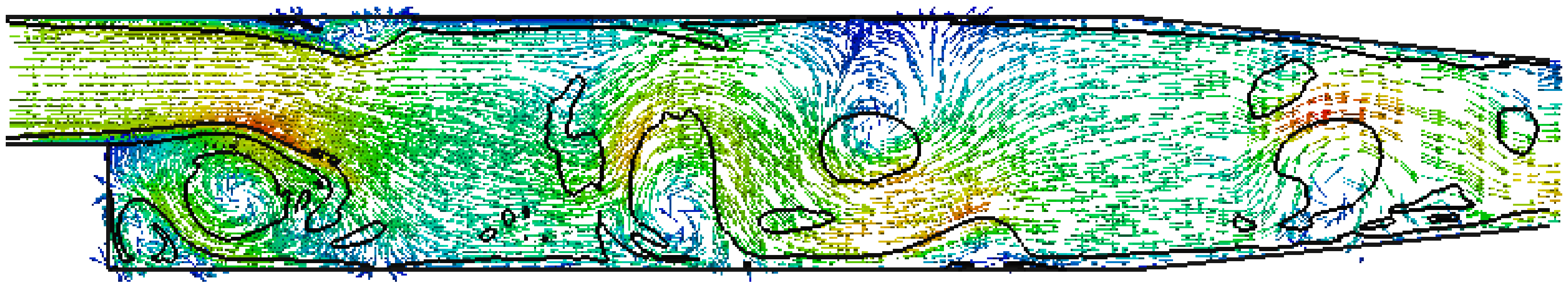
- OpenFOAM provides a ton of utilities, see $FOAM_UTILITIES
- For example, visualize vortex shedding using the vorticity and/or Q utilities and ParaView‘s Countour filter

```
> vorticity
> Q
```



More case details
Post-processing

- Use the Extract Edges filter on the Countour filter results to show the edges of vorticity contours in 2D
- Optionally, show flow vectors using the Glyph filter



BUICE-EATON DIFFUSER

Buice-Eaton diffuser
Introduction

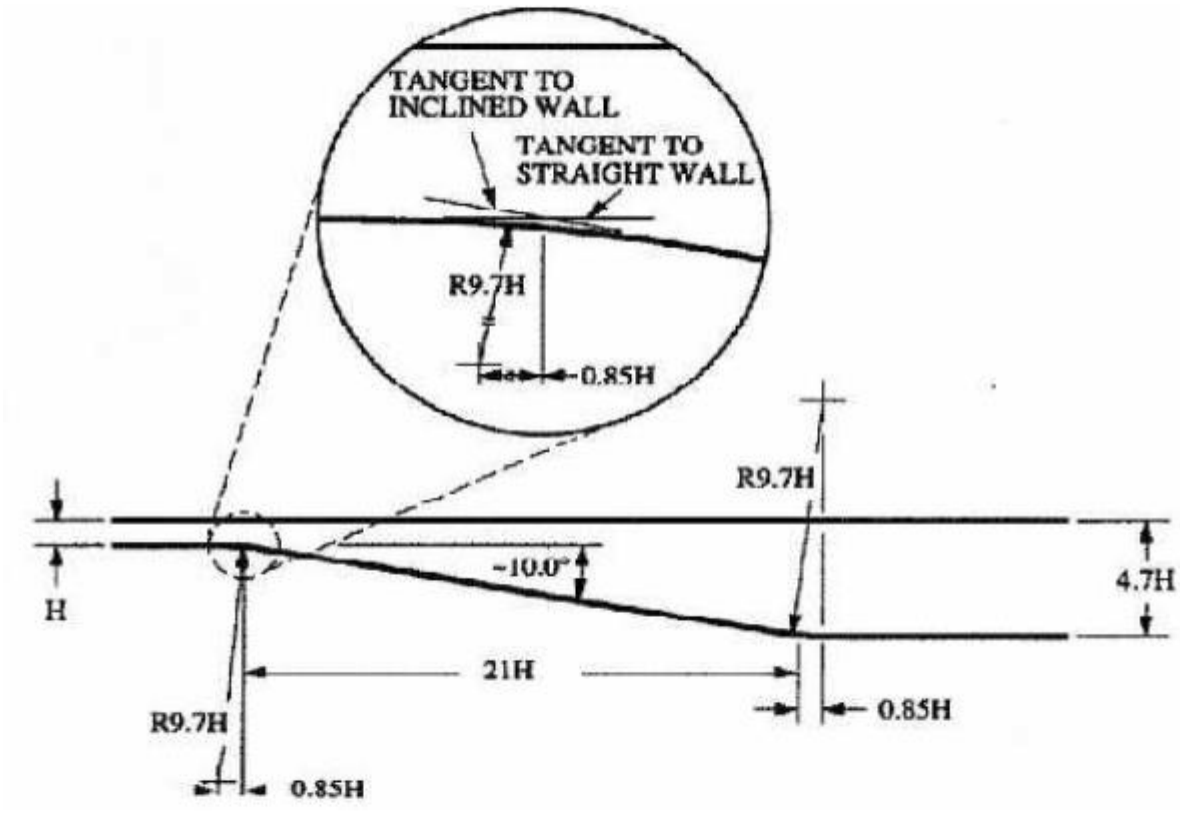
- Buice-Eaton[1] diffuser is a 2D validation case to study predicition of flow separation
  - behaviour of wall functions and turbulennce models
  - numerical settings

- NASA[2] published the mesh, case set-up, experimental data and an analysis of results

[1] Buice, C.U. & Eaton, J.K., "Experimental Investigation of Flow Through an Asymmetric Plane Diffuser", *Journal of Fluids Engineering*, Vol. 122, pp. 433-435, June 2000.

[2] http://www.grc.nasa.gov/WWW/wind/valid/buice/buice01/buice01.html

Buice-Eaton diffuser
Geometry

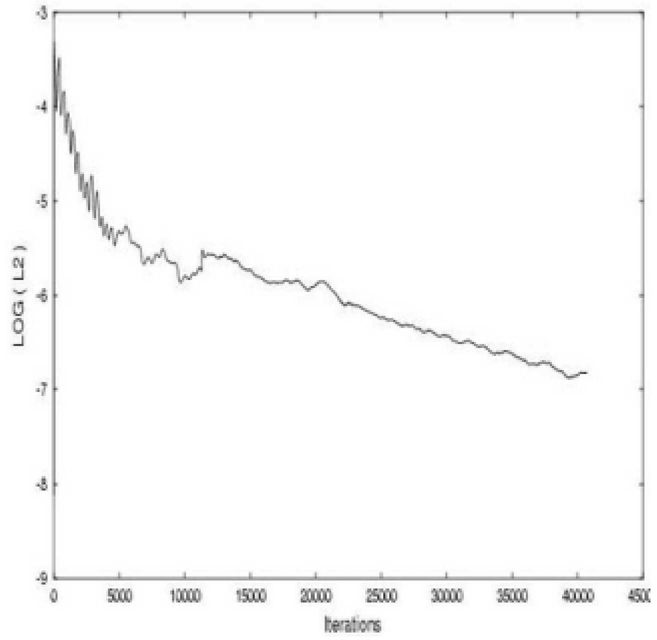


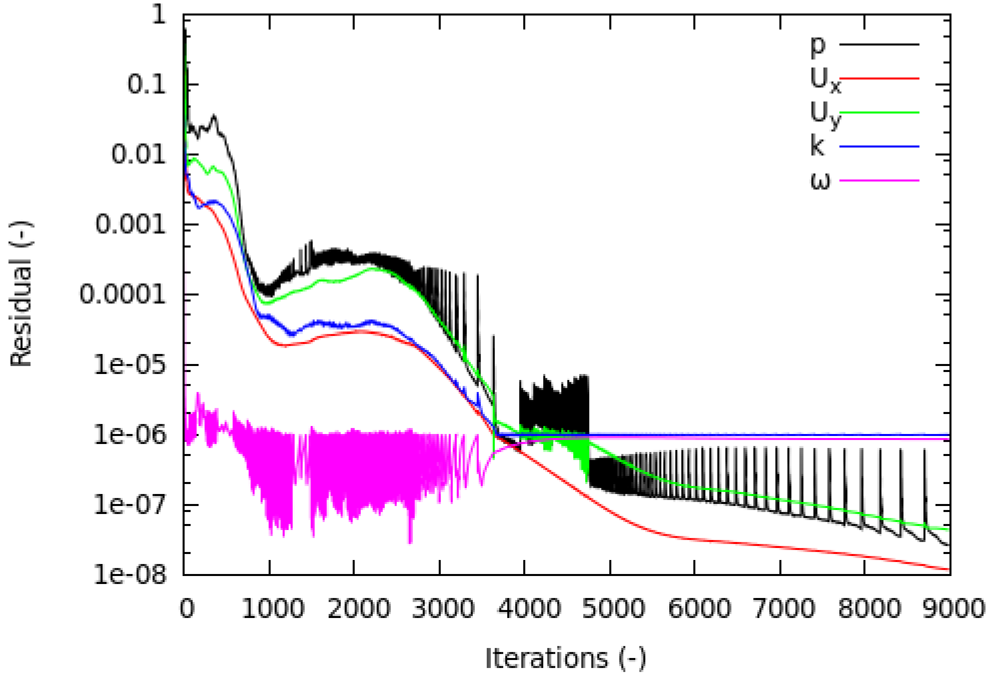
http://www.grc.nasa.gov/WWW/wind/valid/buice/buice01/buice01.html

Buice-Eaton diffuser
Residuals and convergence

NASA results[1] (40,000 iterations)

This course results (9,000 iterations)





[1] http://www.grc.nasa.gov/WWW/wind/valid/buice/buice01/buice01.html

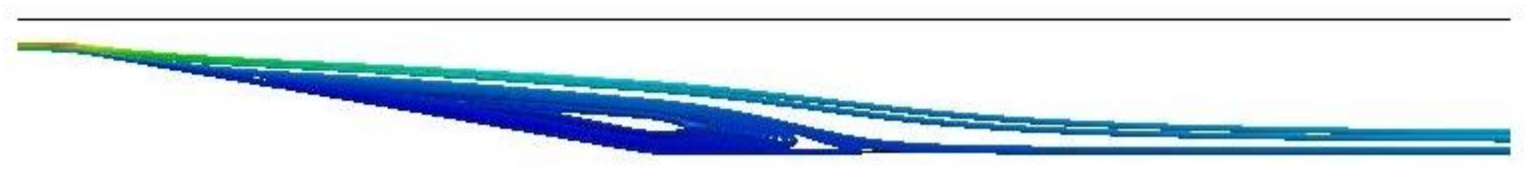
Buice-Eaton diffuser
Residuals and convergence

- Use OpenFOAM‘s foamJob and foamLog utilities to monitor residuals
- Use gnuplot to plot the residuals

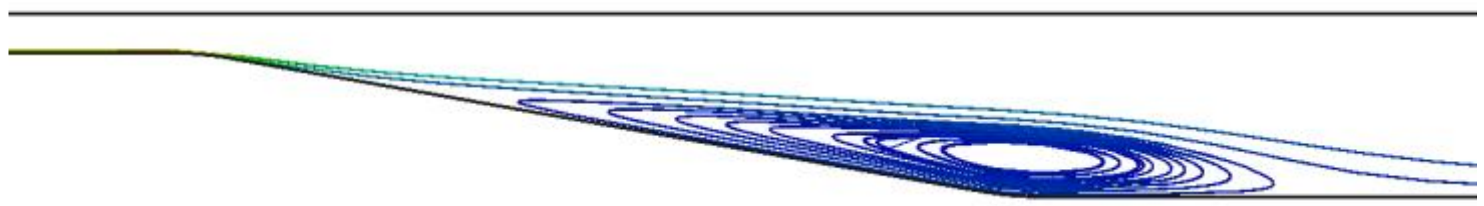
```
> foamJob simpleFoam

> foamLog

> gnuplot

gnuplot > plot 'logs/p_0' using 1:2 with lines notitle
```

Buice-Eaton diffuser
Separation point

NASA results[1] (20,000 iterations)



This course results (9,000 iterations)

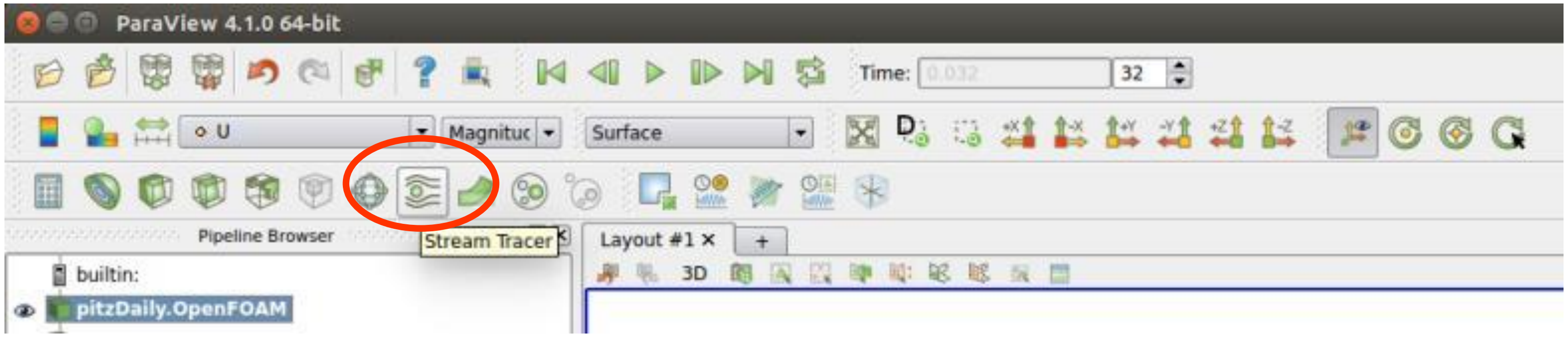


- Use ParaView‘s streamline and tube filters to generate streamlines

[1] http://www.grc.nasa.gov/WWW/wind/valid/buice/buice01/buice01.html

Buice-Eaton diffuser
Separation point

- Use ParaView‘s streamline and tube filters to generate streamlines



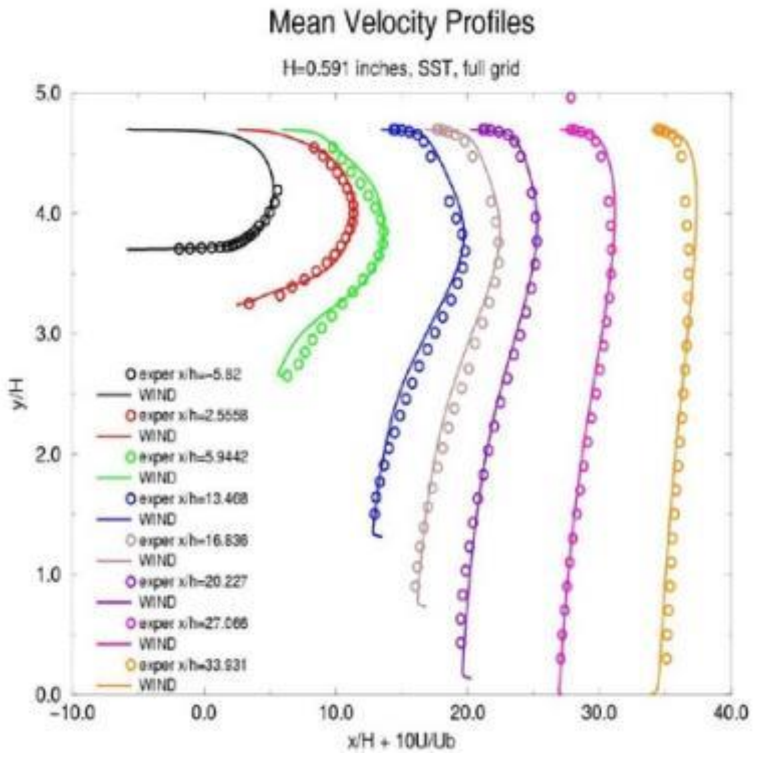
- ... or use the Filters menu:

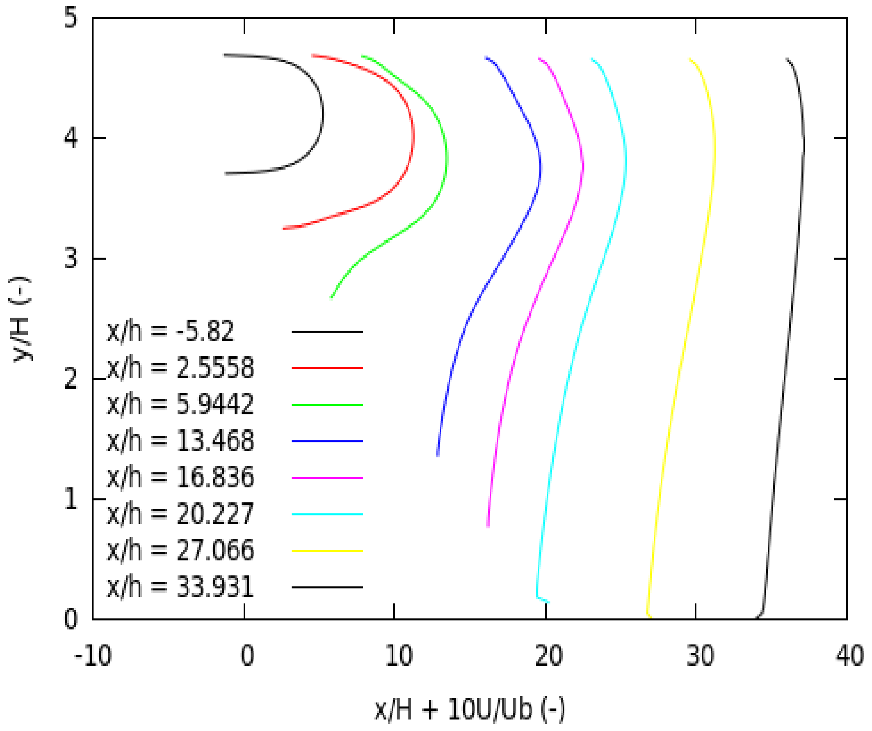
Filters > Alphabetical > Stream Tracers

Buice-Eaton diffuser
Sampling velocity

NASA results[1] (20,000 iterations)

This course results (9,000 iterations)





[1] http://www.grc.nasa.gov/WWW/wind/valid/buice/buice01/buice01.html

Buice-Eaton diffuser
Sampling velocity

- Create a sampleDict file in the system directory with the following content

```
interpolationScheme cellPointFace;

setFormat raw;

sets
(
    sampleLine
    {
        type face;
        axis y;
        start   ( 0.2022 0.0199 0 );
        end     ( 0.2022 0.0706 0 );
        nPoints 100;
    }
);

fields ( U );
```

- Plot using e.g. gnuplot

```
plot 'postProcessing/sets/<time>/sampleLine_U.xy' using 2:1 with lines notitle
```

Summary
Recap

- Basic meshing strategy
- Case structure
- Running of cases and viewing results
- Basic post-processing methods

Summary
Up next

- More advanced meshing techniques
- Creating our own post-processing utility
- Creating our own libraries
    - Boundary conditions
    - Physics library
- Using macros, expansion and `codeStream`
- Bit deeper look at C++ and linux
- Overview of physics
- Parallel computation