Review of Recent and Ongoing Developments of the OpenFOAM library

Henrik Rusche

henrik.rusche@wikki-gmbh.de

Wikki GmbH, Germany

Multiphysical Modelling in OpenFOAM Workshop, Riga, 20 October 2011

Outline

Topics

- Introduction
- Metallurgical Applications
- Automotive Applications
- Viscoelastic Flow Model
- Flash-Boiling Simulations
- Block Matrix in Use: Multi-Phase VOF
- Fluid-Structure Interaction
- Improvements in dynamic mesh
  - Overset Grid, Immersed Boundary Method, Tetrahedral Edge Swapping
  - Radial Basis Function
  - Parallel dynamic mesh simulations
  - Turbo Tools: mixing plane interface
- Shape Optimisation and Adjoint
- Improvements in compressible flows
  - Multi-phase compressibility: under-water explosions
  - New density-based solver: `dbnsFoam`
- Summary and Outlook

Examples of Simulation

Example of Capabilities of OpenFOAM in Complex Physics and Industrial CFD

- This is only a part of the OpenFOAM capabilities!
- Chosen for relevance and illustration of the range of capabilities rather than exhaustive illustration of range of capabilities
- In some cases, simplified geometry is used due to confidentiality
- Regularly, the work resulted in a new solver; in many cases, it is developed as an extension or combination of existing capabilities

Description of Simulation and Setup

- Physics and numerical method setup
- Standard or customised solver; details of mesh resolution and customisation

Metallurgical Applications

Assembling a Matrix for Conjugate Heat Transfer Problems

- OpenFOAM supports multi-region simulations, with possibility of separate addressing and physics for each mesh: multiple meshes, with local fields
- Some equations present only locally, while others span multiple meshes

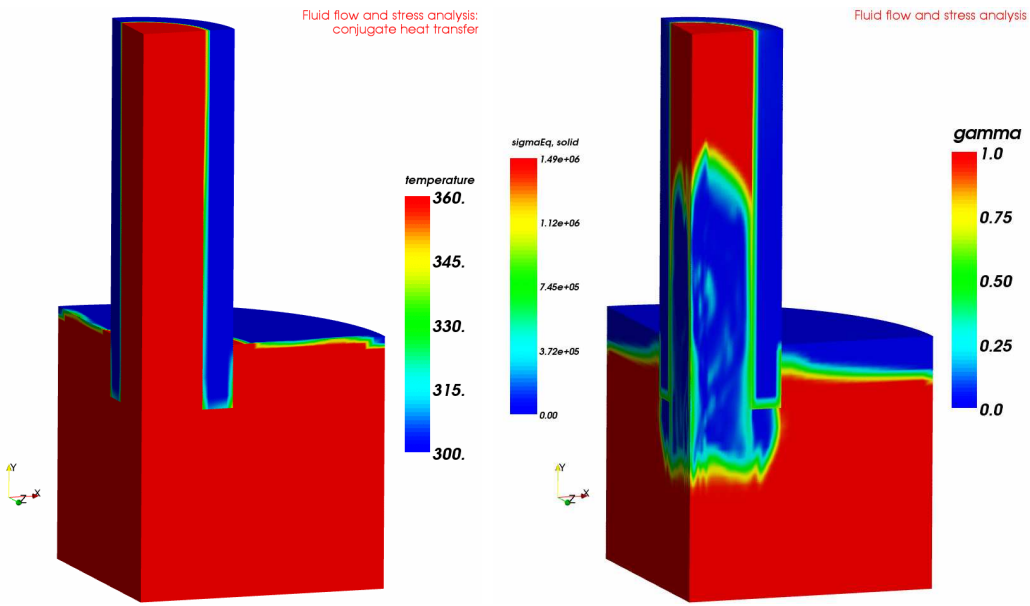
```
coupledFvScalarMatrix TEqns(2);

TEqns.hook
(
    fvm::ddt(T) + fvm::div(phi, T)
  - fvm::laplacian(DT, T)
);

TEqns.hook
(
    fvm::ddt(Tsolid) - fvm::laplacian(DTsolid, Tsolid)
);

TEqns.solve();
```

- Coupled solver handles multiple matrices together in internal solver sweeps: arbitrary matrix-to-matrix and domain-to-domain coupling

Metallurgical Applications

Conjugate Heat Transfer and Thermal Shock

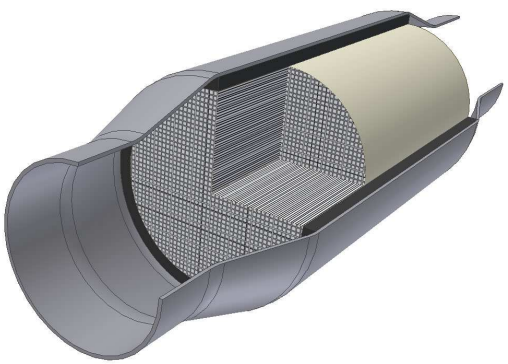
- Coupling may be established geometrically: adjacent surface pairs
- Each variable is stored only on a mesh where it is active: (U, p, T)
- Choice of conjugate variables is completely arbitrary: e.g. catalytic reactions
- Coupling is established only per-variable: handling a general coupled complex physics problem rather than conjugate heat transfer problem specifically
- Allows additional models to be solved on each region without overhead: structural stress analysis, turbulence or LES

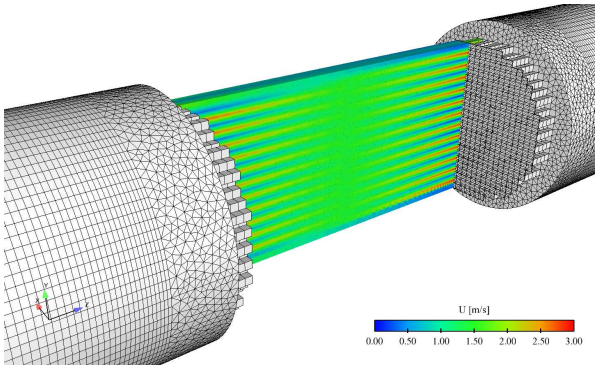


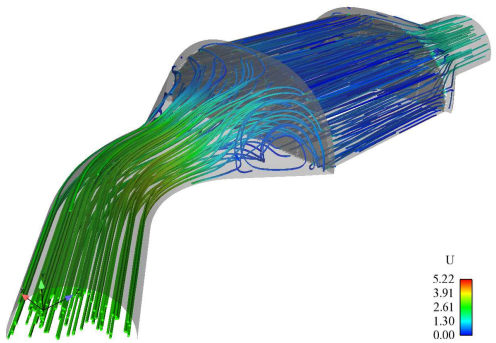
Automotive Applications

Modelling Diesel Particular Filters: **Federico Piscaglia, Politecnico di Milano**

- Steady-state compressible flow through thin porous layers
- Detailed 3-D meshing of channels expensive: mesh resolution requirements
- . . . but due to flow non-uniformity, channel-scale simulations cannot provide the answer: not all channels are equally loaded and energy equation is solved globally
- Solution: **multi-scale filter model**
  - Each channel is one cell thick: (1-D) simulation
  - Porosity is a face property; flow friction is a volumetric sink
  - Porous faces are assigned time-dependent filtration efficiency due to soot deposition, affecting the flow and species distribution
- Automatic meshing tool for the monolith, unstructured mesh for inlet and outlet



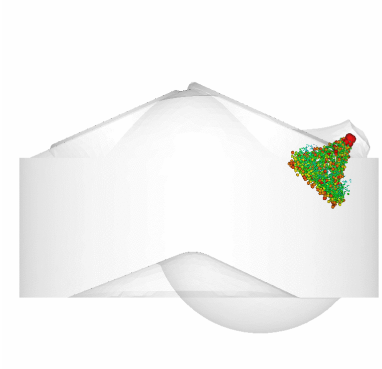
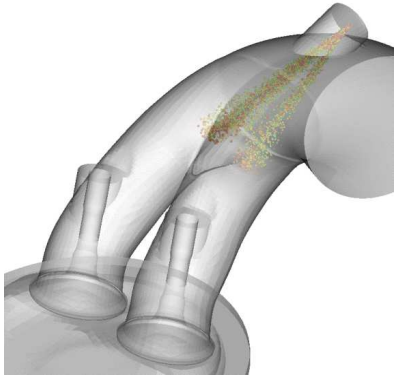


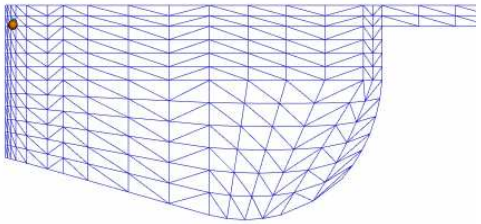


Automotive Applications

Spray, Wall Film and Combustion Simulations in Internal Combustion Engines

- Complete simulation of spray injection, evaporation, wall film and combustion in a GDI engine. Mesh motion and topological changes as shown before
- Basic flow solver, **automatic mesh motion**, topological changes used in standard form. Simulation includes intake stroke (moving piston + valves): **reverse tumble**
- Full suite of Diesel spray modelling using Lagrangian modelling framework
- Implementation of wall film and spray-film interaction: Željko Tuković, FSB
- Mesh sensitivity of **spray penetration**: solved with adaptive refinement!
- Authors of engine simulations: Tommaso Lucchini, Gianluca D'Errico, Daniele Ettore, Politecnico di Milano and Dr. Federico Brusiani, University of Bologna



Viscoelastic Flow Model

MSc Thesis: Jovani Favero, Universidade Federal de Rio Grande del Sul, Brazil

- Viscoelastic flow model:

$$\nabla\bullet\mathbf{u} = 0$$

$$\frac{\partial(\rho\,\mathbf{u})}{\partial t} + \nabla\bullet(\rho\,\mathbf{u}\mathbf{u}) = -\nabla p + \nabla\bullet\boldsymbol{\tau}_\mathbf{s} + \nabla\bullet\boldsymbol{\tau}_\mathbf{p}$$

  where $\boldsymbol{\tau}_\mathbf{s} = 2\eta_s\mathbf{D}$ is the solvent stress contribution and $\boldsymbol{\tau}_\mathbf{p}$ is the **polymeric part of the stress**, non-Newtonian in nature

- Depending on the model, $\boldsymbol{\tau}_\mathbf{p}$ is solved for: saddle-point problem
- Models introduce “upper”, “lower” or Gordon-Schowalter derivatives, but we shall consider a general form: standard transport equation in relaxation form

$$\frac{\partial\boldsymbol{\tau}_\mathbf{p}}{\partial t} + \nabla\bullet(\mathbf{u}\boldsymbol{\tau}_\mathbf{p}) = \frac{\boldsymbol{\tau}^* - \boldsymbol{\tau}_\mathbf{p}}{\delta}$$

  where δ is the relaxation time-scale

- Problem: $\boldsymbol{\tau}_\mathbf{p}$ dominates the behaviour and is explicit in the momentum equation

Viscoelastic Flow Model

Model Implementation Recipe

- Recognise $\boldsymbol{\tau}^*$ as the equilibrium stress value: make it implicit!

$$\nabla\bullet\boldsymbol{\tau}^* = \nabla\bullet\left[\boldsymbol{\kappa}\bullet\frac{1}{2}\left(\nabla\mathbf{u} + (\nabla\mathbf{u})^T\right)\right]$$

- Calculate implicit viscoelastic viscosity:

$$\boldsymbol{\kappa} = \boldsymbol{\tau}^*\bullet\left[\frac{1}{2}(\nabla\mathbf{u} + (\nabla\mathbf{u})^T)\right]^{-1}$$

- Split complete stress into implicit and explicit component

$$\begin{aligned}\nabla\bullet\boldsymbol{\tau}_{\mathbf{p}} &= \nabla\bullet\boldsymbol{\tau}^* + \nabla\bullet\boldsymbol{\tau}_{\mathbf{corr}} \\ &= \nabla\bullet(\boldsymbol{\kappa}\bullet\nabla\mathbf{u}) + \nabla\bullet\boldsymbol{\tau}_{\mathbf{corr}}\end{aligned}$$

Viscoelastic Flow Model

Implemented Viscoelastic Models

- **Kinetic Theory Models**: Maxwell linear; UCM and Oldroyd-B; White-Metzner; Larson; Cross; Carreau-Yasuda; Giesekus; FENE-P; FENE-CR
- **Network Theory of Concentrated Solutions and Melts Models**: Phan-Thien-Tanner linear (LPTT); Phan-Thien-Tanner exponential (EPTT); Feta-PTT
- **Reptation Theory / Tube Models**: Pom-Pom model; Double-equation eXtended Pom-Pom (DXPP); Single-equation eXtended Pom-Pom (SXPP); Double Convected Pom-Pom (DCPP)
- **Multi-Mode Form**: The value of $\boldsymbol{\tau}_{\mathbf{p}}$ is obtained by the sum of the K modes

$$\boldsymbol{\tau}_{\mathbf{p}} = \sum_{K=1}^{n} \boldsymbol{\tau}_{\mathbf{p}\,K}$$

Flow Solvers Implemented by Jovani Favero: **Example Simulation**

- Single-phase non-Newtonian solver based on transient SIMPLE
- Multi-phase free surface VOF solver: viscoelastic in each phase
- Support for topological changes: syringe ejection

Flash-Boiling Simulations

Flash-Boiling Flows: **Shiva Gopalakrishnan, David P. Schmidt, UMass Amherst**

- The fundamental difference between flash boiling and cavitation is that the process has a higher saturation pressure and temperature: higher density
- Enthalpy required for phase change is provided by inter−phase heat transfer
- **Jakob number**: ratio of sensible heat available to amount of energy required for phase change

$$Ja = \frac{\rho_l c_p \Delta T}{\rho_v h_{fg}}$$

- Equilibrium models are successful for cavitation since Ja is large and timescale of heat transfer is small. Flash boiling represents a finite rate heat transfer process: **Homogeneous Relaxation Model (HRM)**

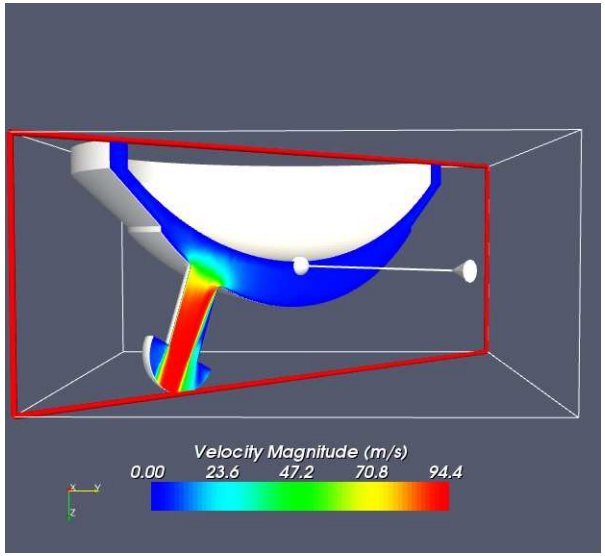
$$\frac{Dx}{Dt} = \frac{\bar{x} - x}{\Theta}; \qquad \Theta = \Theta_0 \epsilon^{-0.54} \phi^{1.76}$$

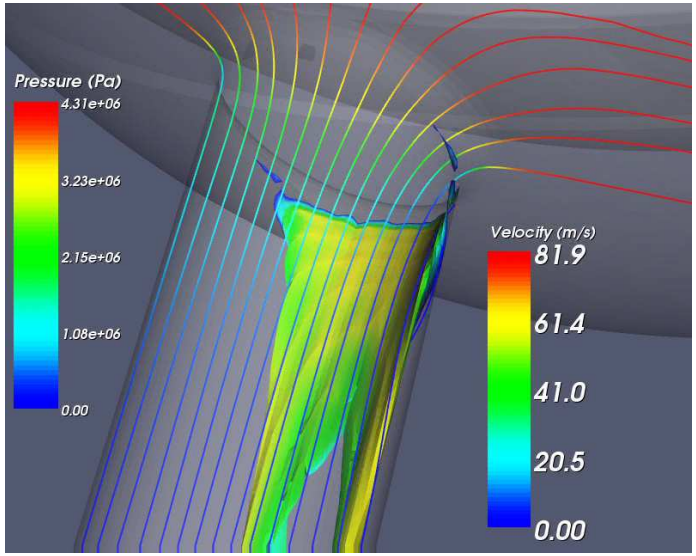
x is the quality (mass fraction), relaxing to the equilibrium $\bar{x}$ over a time scale Θ

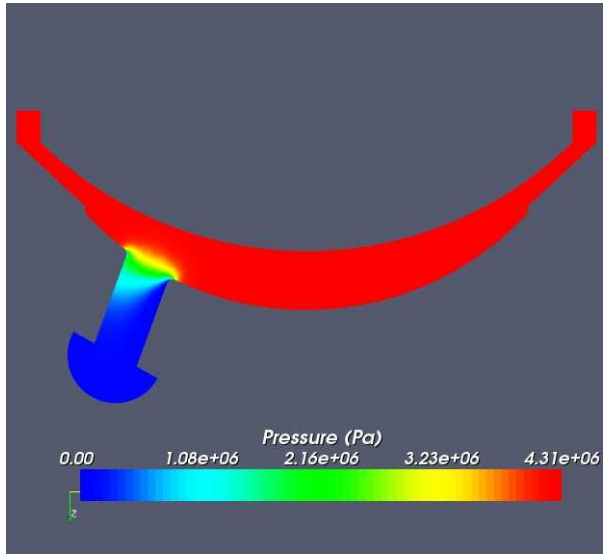
- The timescale Θ is obtained from empirical relationship: Downar–Zapolski [1996]. ϵ is the void fraction and ϕ is the non−dimensional pressure.

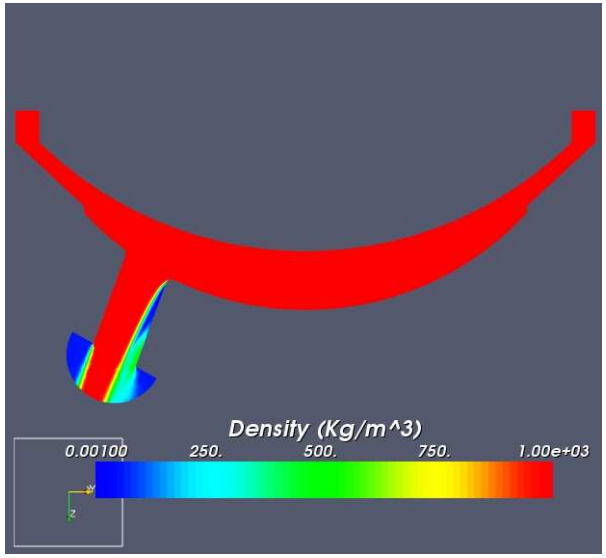
Flash-Boiling Simulations

Asymmetric Fuel Injector Nozzle-Design from Bosch GmbH.









Block Matrix in Use: Multi-Phase VOF

Multi-Phase Volume-of-Fluid Solver

- System of equations contains multiple VOF equations and global continuity handled by a pressure equation in standard form
- The phase for which VOF is solved first dominates the other phases: this is not acceptable; flipping the order of solution moves the problem around
- Aim: **Coupled pressure based approach to achieve physical behaviour**
- Solution strategy: solve α_i transport equations, volumetric continuity and closure equation in a strongly coupled manner, making the coupling terms implicit!
- To make this run, we shall use the block matrix and block solver (Jasak and Clifford, 2009)
- Credit goes to **Kathrin Kissling and Julia Springer, NUMAP-FOAM 2009**

Block Matrix in Use: Multi-Phase VOF

Multi-Phase Volume-of-Fluid Solver: Equation Set and Coupling

- Volume fraction transport equation, with separated pressure-driven flux terms

$$\left[\frac{\partial\left[\alpha_i\right]}{\partial t}\right] + \left[\nabla\bullet\left(\left(\left(\frac{A_H}{A_D}\right)_f \bullet \mathbf{s}_f + \frac{1}{A_D}(\sigma\kappa)_f \nabla_f^{\perp}\alpha + \frac{1}{A_D}(\mathbf{g}\bullet\mathbf{x})_f |\mathbf{s}_f| \nabla_f^{\perp}\rho\right)\left[\alpha_i\right]\right)\right]$$
$$- \left[\nabla\bullet\left(\left(\frac{1}{A_D}\right)\nabla p^*\left[\alpha_i\right]\right)\right] + \left[\nabla\bullet\left(\left[\alpha_i\right]\sum_{k=1,k\neq i}^{N}\alpha_k\,\phi_{r,ij}\right)\right] = 0$$

- Pressure equation, with separated phase fluxes (coupling terms)

$$\left[\nabla\bullet\left[\left(\frac{1}{A_D}\right)_f \nabla\left[p^*\right]\right]\right] = \nabla\bullet\left(\sum_{i=1}^{N}\alpha_i\phi^*\right)$$

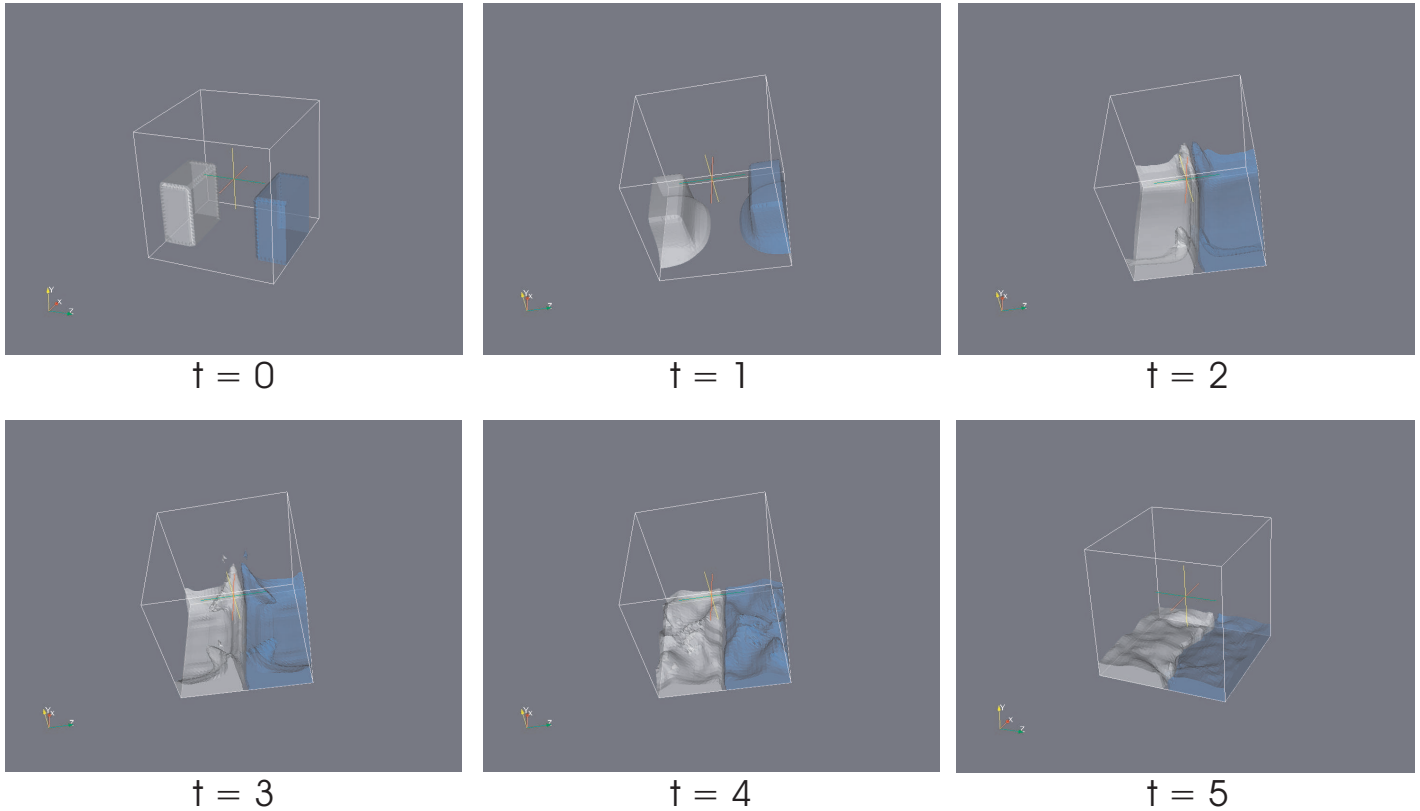
- Closure equation and definition of fluxes

$$\sum_i \alpha_i = 1 \quad \phi^* = \sum_i \alpha_i^N \phi^*$$

Block Matrix in Use: Multi-Phase VOF

Multi-Phase Volume-of-Fluid Solver: Solution Strategy

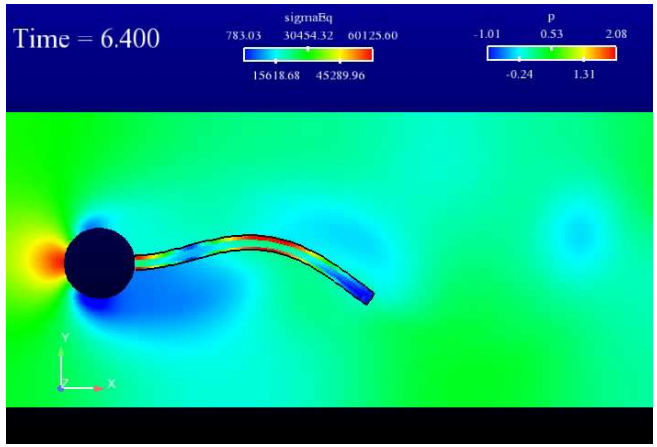
- Above equations are dumped into a block matrix format, with coefficient size N + 1: (p^*, α_i) and solved in a block-coupled manner
- Result: strong coupling between αs and p: no dominant phase

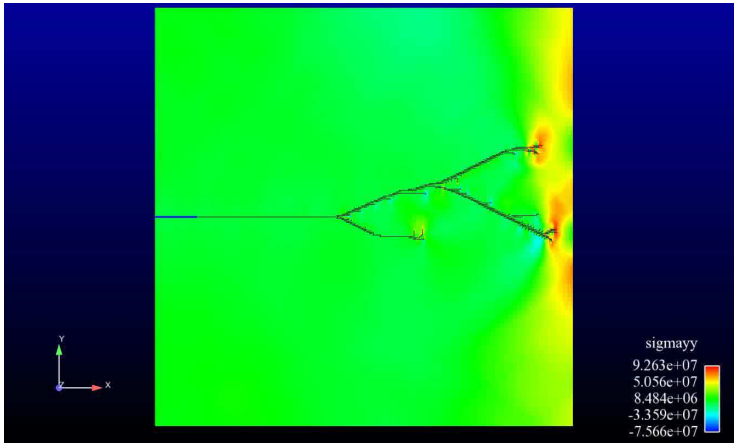


Fluid-Structure Interaction

Fluid-Structure Coupling Capabilities in OpenFOAM

- As a Continuum Mechanics solver, OpenFOAM can deal with both fluid and structure components: easier setup of coupling
- (Parallelised) surface coupling tools implemented in library form: facilitate coupling to external solvers without “coupling libraries” using proxy surface mesh
- Structural mechanics in OpenFOAM targeted to non-linear phenomena: consider best combination of tools
  - Large deformation formulation in absolute Lagrangian formulation
  - Independent parallelisation in the fluid and solid domain
  - Parallelised data transfer in FSI coupling
- Dynamic mesh tools and boundary handling used to manipulate the fluid mesh

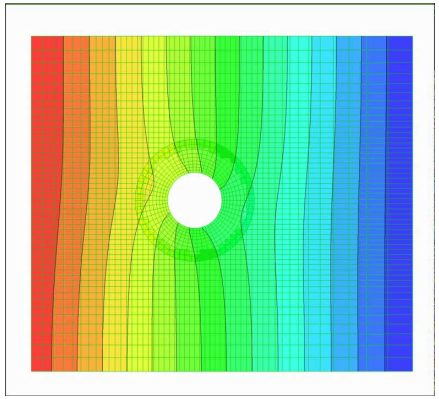


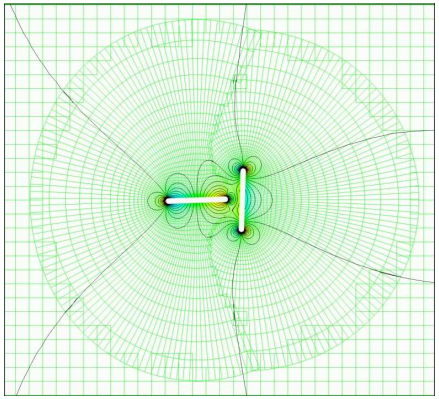


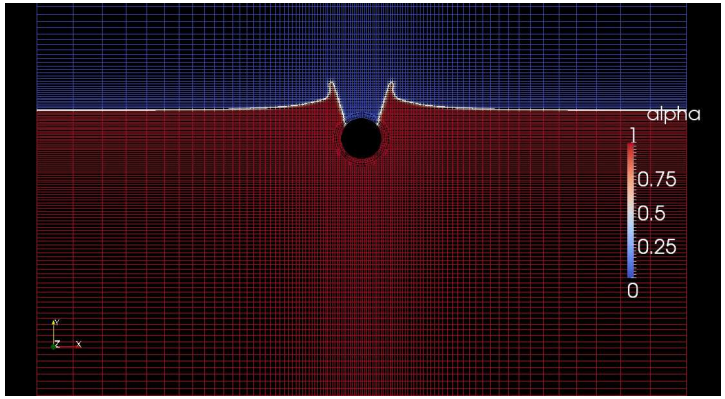
Overset Grid in OpenFOAM

`foamedOver`: Overset Grid Technology in OpenFOAM

- Work by David Boger, Penn State University using SUGGAR and DirtLib libraries developed by Ralph Noack, Penn State (must mention Eric Paterson!)
- Overset Grid Technology
  - Multiple components meshed individually, with overlap
  - Hole cutting algorithm to remove excess overlap cells
  - Mesh-to-mesh interpolation with implicit updates built into patch field updates and linear solver out-of-core operations
- Body-fitted component meshes: preserving quality and near-wall resolution
- Simple mesh motion and geometrical studies (replacing individual components)
- Overset grid is physics-neutral! Currently testing for free surface flows



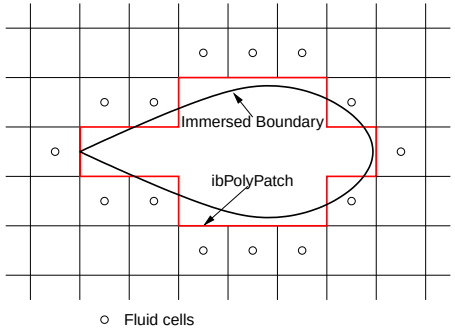


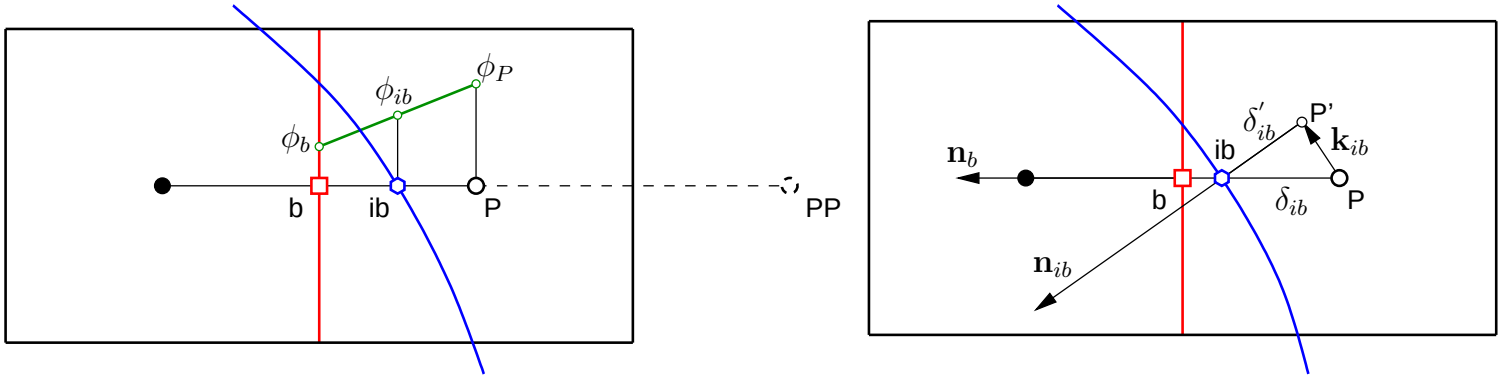


Immersed Boundary Method

Immersed Boundary Method

- Handling of moving obstacles in the flow domain whose size is larger than mesh resolution: covering multiple cells
- Mesh topology and connectivity does not change: **immersed STL surface**
- Presence of boundary implicitly accounted for in discretisation, with appropriate handling of boundary conditions

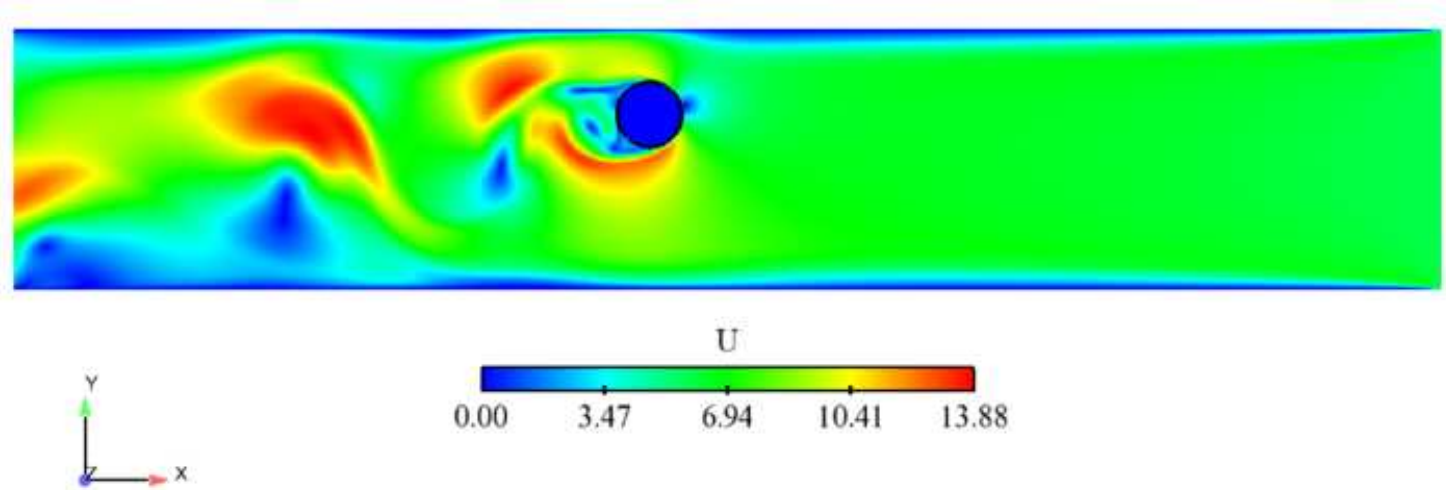




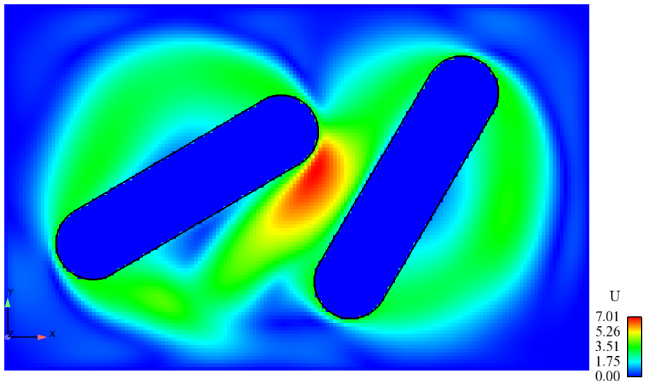
Immersed Boundary Method

Immersed Boundary Method: Examples

- Laminar flow around a 2-D moving circular cylinder in a channel



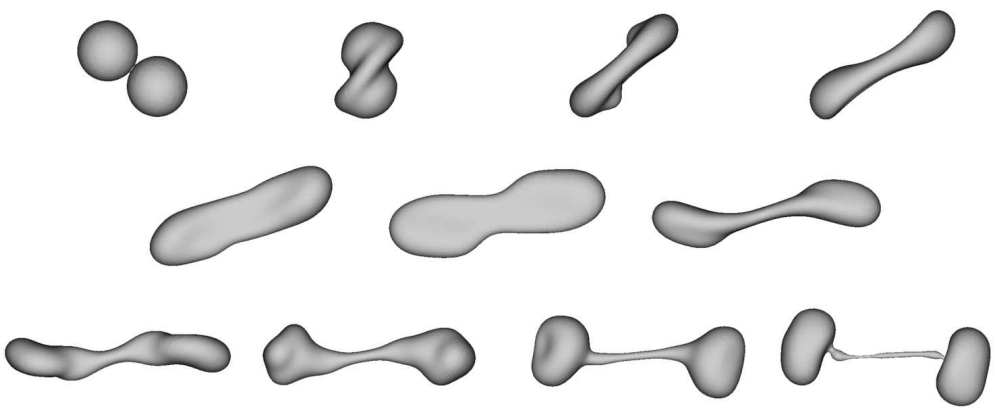
- Laminar flow around two counter-rotating elements in a cavity

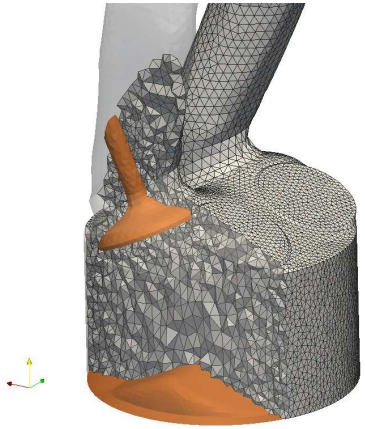


Tetrahedral Edge Swapping

Re-Meshing with Tetrahedral Edge Swapping

- In cases where mesh motion involves topological change at the boundary or unpredictable mesh deformation, topological change machinery is impractical: cannot decide a-priori where to place topology modifiers
- **Dynamic remeshing using tetrahedral edge swapping**
  - Motion is prescribed on external boundaries
  - Tetrahedral cell quality examined continuously: bad cells trigger automatic remeshing without user interaction: answers to `dynamicMesh` interface
  - Implemented by **Sandeep Menon, UMass Amherst** as a ready-to-use library
- Example: viscoelastic droplet collision using free surface tracking
- Can be used for all dynamic mesh cases: ultimate ease of mesh setup!

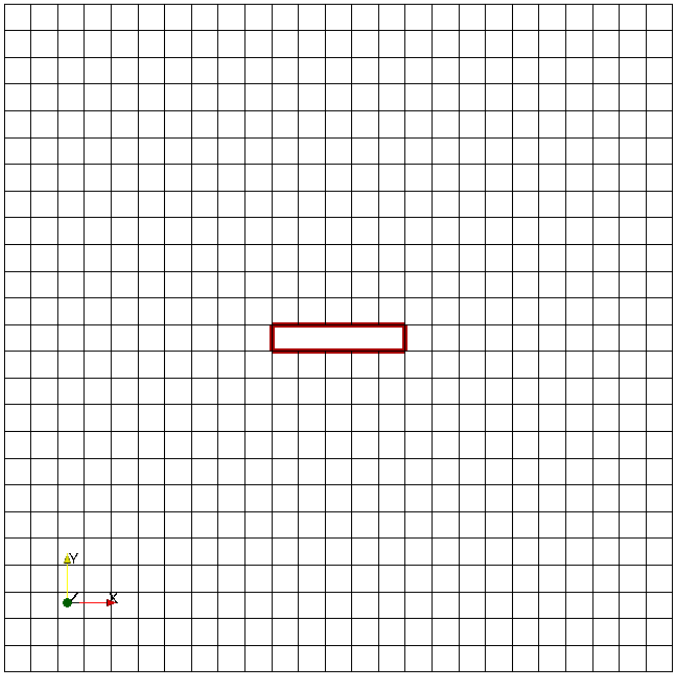


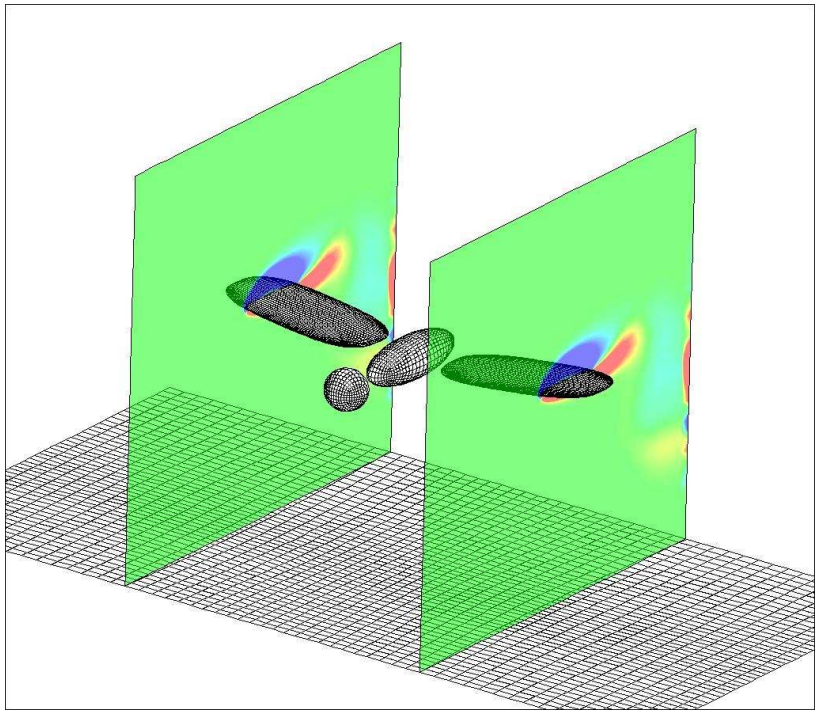


Radial Basis Function

Radial Basis Function Interpolation

- General interpolation for clouds of points
- Mathematical tool which allows data interpolation from a small set of control points to space **with smoothness criteria** built into the derivation
- Used for mesh motion in cases of large deformation: no inverted faces or cells
- Implemented by Frank Bos, TU Delft and Dubravko Matijašević, FSB Zagreb





Radial Basis Function

RBF Mesh Morphing Object

- RBF morphing object defines the parametrisation of geometry (space):
  1. Control points in space, where the parametrised control motion is defined
  2. Static points in space, whose motion is blocked
  3. Range of motion at each control point: $(\mathbf{d}_0, \mathbf{d}_1)$
  4. Set of scalar parameters δ for control points, defining current motion as

$$\mathbf{d}(\delta) = \mathbf{d}_0 + \delta(\mathbf{d}_1 - \mathbf{d}_0), \quad \text{where} \quad 0 \leq \delta \leq 1$$

- For each set of δ parameters, mesh deformation is achieved by interpolating motion of control points $\mathbf{d}$ over all vertices of the mesh: new deformed state of the geometry
- Mesh in motion remains valid since RBF satisfies smoothness criteria

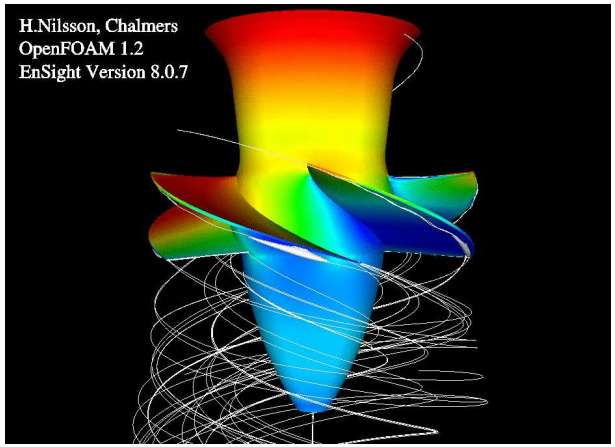
Using RBF in Optimisation

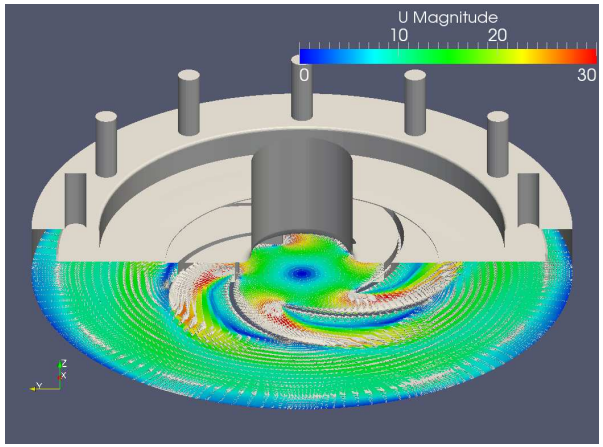
- Control points may be moved individually or share δ values: further reduction in dimension of parametrisation of space
- Mesh morphing state is defined in terms of δ parameters: to be controlled by the optimisation algorithm

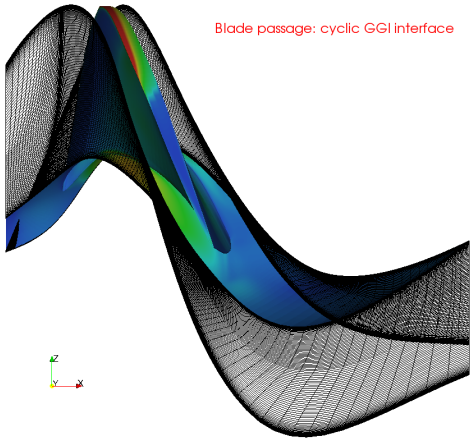
OpenFOAM Turbo Tools

General Grid Interface

- Turbomachinery CFD requires additional features: implemented in library form
- **General Grid Interface (GGI)** and its derived forms
  - Cyclic GGI
  - Partial overlap GGI
  - Mixing plane interface (under testing)
- Implementation and parallelisation is complete: currently running validation cases in collaboration with commercial clients and Turbomachinery Working Group
- Other turbo-related components in pipeline: harmonic balance solver
- Library-level implementation allows re-use of GGI beyond turbomachinery







Parallelisation of Topological Changes

Topologically Changing Meshes in Parallel Simulations

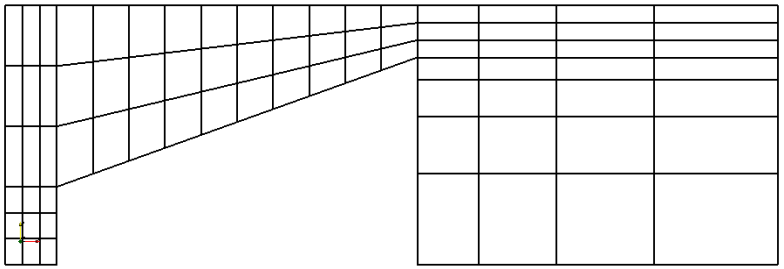
- Topology engine did not account for parallelisation: synchronisation and triggering of topological changes
- For good scaling with parallel execution
  - Mesh change must be localised on a single processor
  - . . . but topology update most be synchronised!
- Different types of topology modifiers behave in a different manner
- **Domain reconstruction completely broken**: no addressing data
- Issue of dynamic load balancing on topologically changing meshes becomes acute

Layer Addition-Removal

Parallelisation of Layer Addition-Removal Mesh Modifier

- `layerAdditionRemoval` mesh modifier removes cell layers when the mesh is compressed and adds cells when the mesh is expanding. Definition:
  - Oriented face zone, defining an internal surface
  - Minimum and maximum layer thickness in front of the surface
  - Both internal and patch faces are allowed

```
right
{
    type layerAdditionRemoval;
    faceZoneName rightExtFaces;
    minLayerThickness 0.0002;
    maxLayerThickness 0.0005;
    active on;
}
```



Tasks

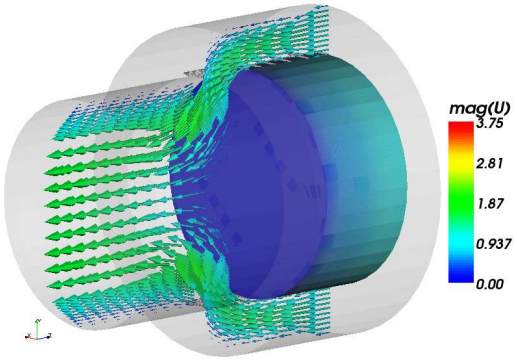
- Allow face zone to be distributed over multiple processors
- Synchronise logic for addition and removal: interface is formally present on all processors, even if it locally has zero faces
- Check and validate decomposition normal to face zone: mesh motion issues

Sliding Interface

Parallelisation of Sliding Interface Mesh Modifier

- `slidingInterface` allows for relative sliding of components. Definition:
  - A master and slave patch, originally external to the mesh
  - Allows uncovered master and slave faces to remain as boundaries

```
mixerSlider
{
    type slidingInterface;
    masterPatchName outsideSlider;
    slavePatchName insideSlider;
    projection visible;
    active on;
}
```



Flow control device, sliding interface and mesh layering

Tasks

- Sliding surfaces are defined as a patch pair and allow partial coverage
- No global logic: sliding pair may be local on a processor mesh
- Protection in parallel decomposition is required for points: this is a task for domain decomposition tool

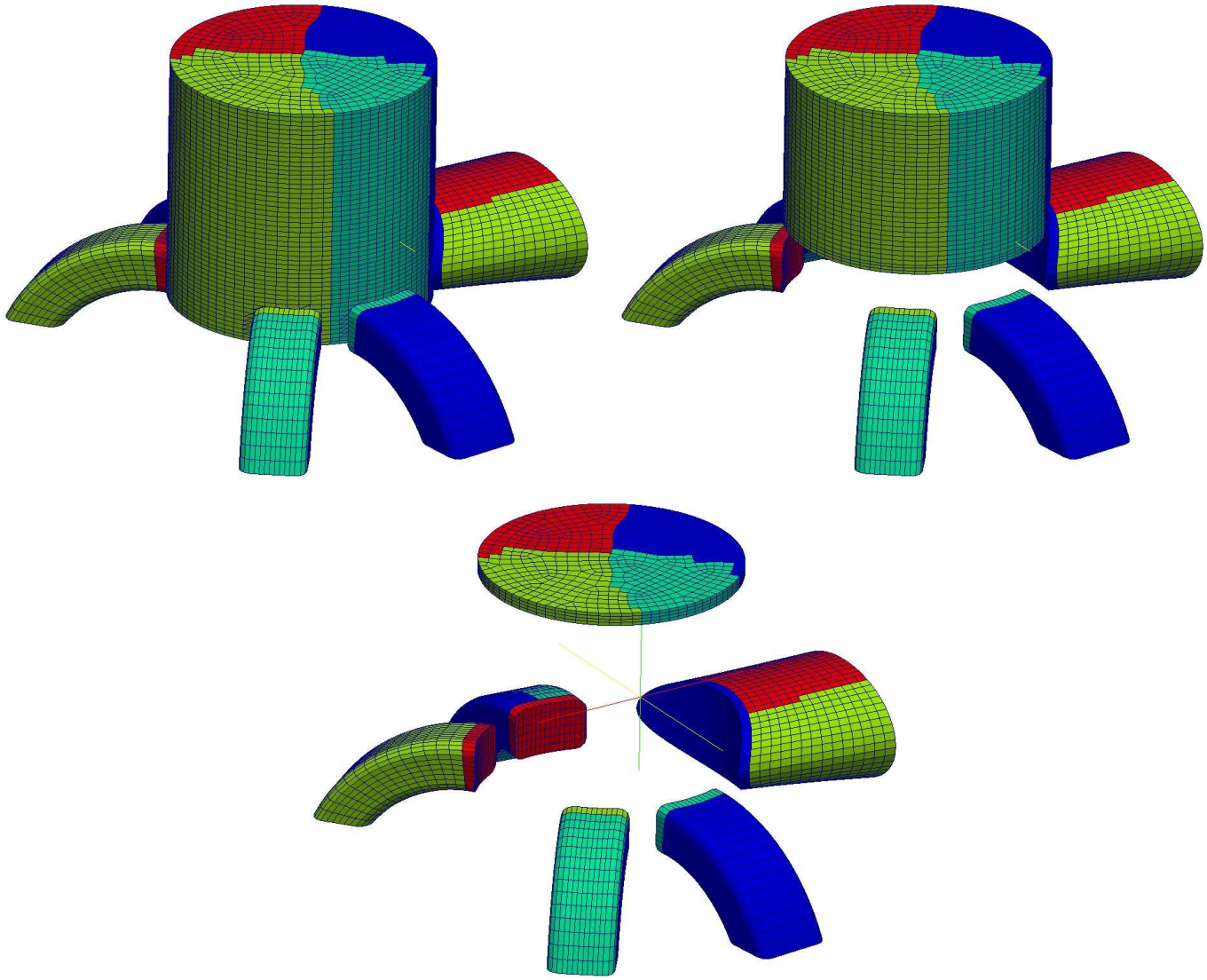
Parallelised Topological Changes

Low-Level Work in Parallelised Topological Changes

- New effects to be accounted for
  - Topological change happening on some processors but not on others
  - Re-calculation of local and global mesh data involves communication which need to be synchronised when topological change is local
  - Synchronisation required for processor boundary faces that undergo a topological change: identical operation on both processors
- Data mapping on neighbouring processor may require local update for parallel synchronisation
- Currently addressing issues of load balancing on a problem-by-problem basis

Parallel Decomposition for Engines

Example: `engineScotch` Domain Decomposition Method in Motion



Reconstruction for Topological Changes

Domain Reconstruction Tool for Topologically Changing meshes

- Standard decomposition tools uses point/face/cell/boundary maps between a single CPU and processor mesh. Maps are created on a static mesh with decomposition
- With parallelised topological changes, this breaks down completely: global mesh and numbering does not exist and cannot be implied
- Solution: use processor meshes to build a global mesh from scratch, by adding processor meshes in order, merging shared points and faces

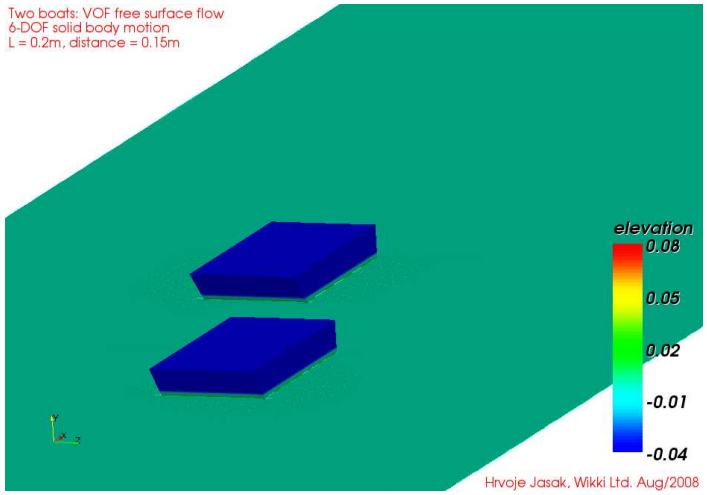
`reconstructParMesh` Utility

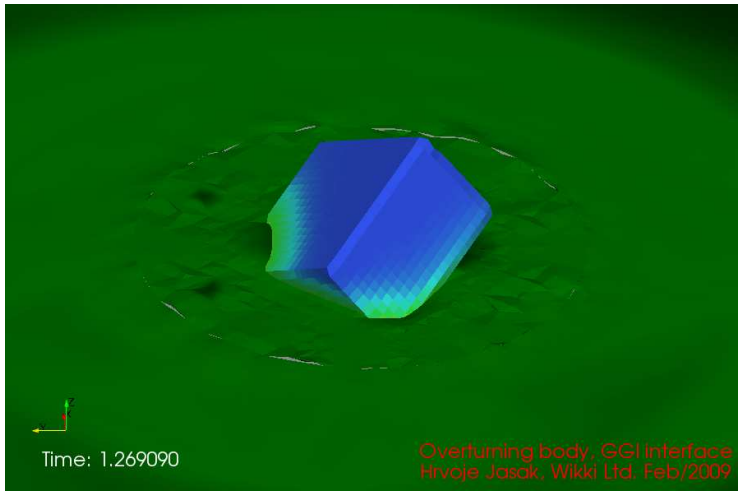
- In presence of maps (no topological changes) use standard method
- Upon topological changes build and merge the mesh, adding cells in order of processor index and assemble mapping data
- Fields on reconstructed mesh can be assembled or decomposed as before
- `-cellDist` option can be used to visualise domain decomposition: this time it is assembled on reconstruction!

Examples of Complex Mesh Motion

6-DOF Floating Body in Free Surface Flow

- **Flow solver**: turbulent VOF free surface, with moving mesh support
- **Mesh motion** depends on the forces on the hull: 6-DOF solver
- **6-DOF solver**: `ODE + ODESolver` energy-conserving numerics implemented using quaternions, with optional elastic/damped support
- Variable diffusivity Laplacian motion solver with 6-DOF boundary motion as the boundary condition for the mesh motion equation
- Topological changes to preserve mesh quality on capsize
- Coupled transient solution of flow equations and 6-DOF motion, force calculation and automatic mesh motion: custom solver is built from library components





Shape Optimisation and Adjoint

Types of Optimisation

- **Deterministic Methods**: all observables and controls assumed deterministic
- **Stochastic Optimisation**: Probability density function associated with all inputs and controls

Types of Analyses: Gradient or Non-Gradient Based Analysis

- Sensitivity analysis
- Uncertainty analysis: robust design
- Parameter estimation
- Geometric shape optimisation; Topology optimisation

Software Toolkit

1. **Low-degree shape parametrisation**: RBF morphing, (free form deformation)
2. **Direct mesh representation and deformation**: automatic mesh motion solver
3. **CFD solver and evaluation of objective**: OpenFOAM solvers
4. **Optimisation algorithm**: native or external optimiser
   - Non-gradient based: native Simplex Nelder-Mead
   - Gradient-based: Newton-type, using tangent and adjoint (limited capability with scope for further work)

Shape Optimisation and Adjoint

Gradient-Based Optimisation in OpenFOAM

- Options on formulation of derivative: **Tangent method**, **Adjoint solver**
- Options on evaluation: **discrete or continuous method**
- Currently implemented only a laminar continuous adjoint solver. This is not publicly available and requires rewrite of numerics, using block coupling
- Prototype implementation for **automatic differentiation** (discrete method): `Fad0ne` class and operator overloading

Continuous Adjoint Optimisation for Turbulent Flows in OpenFOAM

- Solving adjoint incompressible Navier-Stokes system with a complete turbulence model with near-wall treatment
  - Adjoint momentum and continuity equation
  - Adjoint $k - \epsilon$ turbulence model with adjoint wall treatment
- This is a specific tool for specific problems: derivation for internal and external flows is different (eg. surface forces vs. pressure loss)
- Substantial mathematical effort required: boundary conditions are derived specifically for the chosen objective

Automatic Differentiation

Background on Forward Derivatives

- In CFD we specify a number of input variables (mesh, inlet velocity, attack angle), perform a series of mathematical operations to get some output (pressure distribution, lift coefficient).
- Clearly, all output variables depend on the input and mathematics is tractable
- Therefore, forward derivatives, propagation of uncertainty etc. simply describes how the output data depends on the input: we wish to recover this dependence not only in value but also in derived properties

Implementation

- In all cases, implementation provides a discrete forward derivative and discrete adjoint: full consistency with the physical model and discretisation
- Operation on derivatives (or other attached data) performed as side-effects on basic operations: templating on `scalar` type
- Currently propagated to matrix level, excluding the mesh module
- For **shape optimisation**, parametric geometry and mesh is required
- Aim: pull through the sensitivity analysis machinery through complete OpenFOAM using templating trickery and meta-programming

Geometric Shape Optimisation

Geometric Shape Optimisation with Parametrised Geometry

- Specify a desired object of optimisation and use the parametrisation of geometry to explore the allowed solution space in order to find the **minimum of the optimisation objective**

$$objective = f(\mathbf{shape})$$

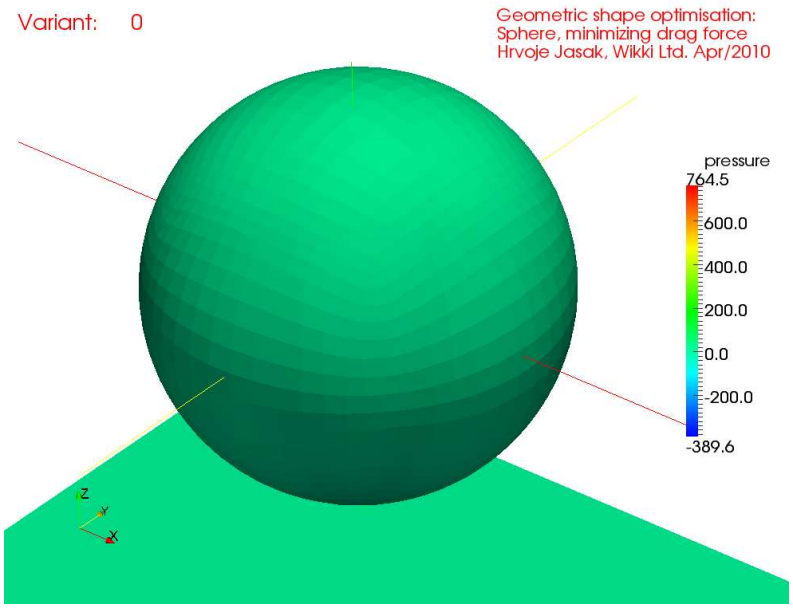
1. **Parametrisation of Geometry**
   - Computational geometry is complex and usually available as the computational mesh: a large amount of data
   - Parametrisation tool: **RBF mesh morphing**, defining deformation at a small number of mesh-independent points in space
2. **CFD Flow Solver** is used to provide the flow solution on the current geometry, in preparation for objective evaluation
3. **Evaluation of Objective**: usually a derived property of the flow solution
4. **Optimiser Algorithm**: explores the solution space by providing sets of $\mathbf{shape}$ coordinates and receiving the value of $objective$. The search algorithm iteratively limits the space of solutions in search of a minimum value of $objective$

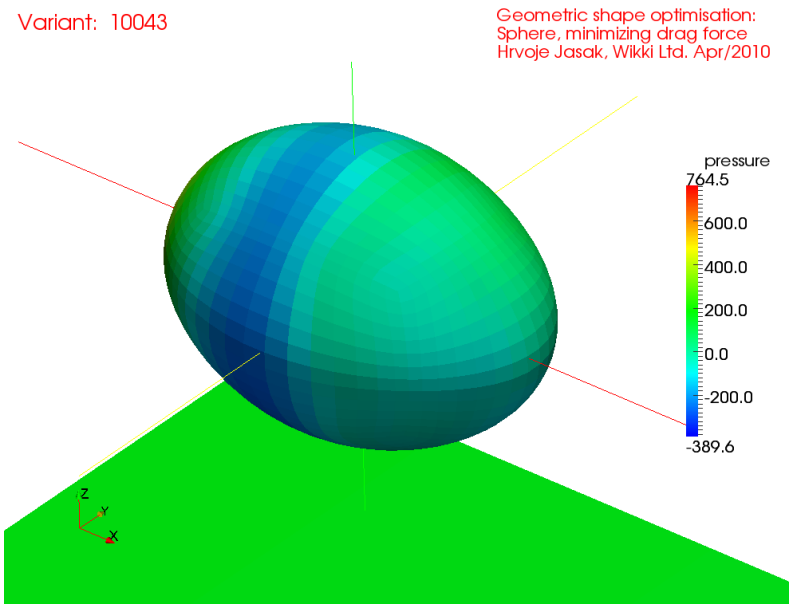
Geometric Shape Optimisation

3-D Sphere: Minimising Drag Force

- Using 9 control points in motion, with symmetry constraints: 4 points in front square, radial motion; 4 points in back square, radial and axial motion; 1 tail point, axial motion only
- Optimisation is performed with 3 parameters:

```
iter = 1 pos = (0.2 0.7 0.2) v = 147.96 size = 0.2997
iter = 5 pos = (0.06111 0.7092 0.7092) v = 106.26 size = 0.2153
iter = 12 pos = (0.03727 0.9354 0.3830) v = 77.934 size = 0.0793
iter = 22 pos = (0.04095 0.9458 0.3413) v = 75.821 size = 0.006610
```



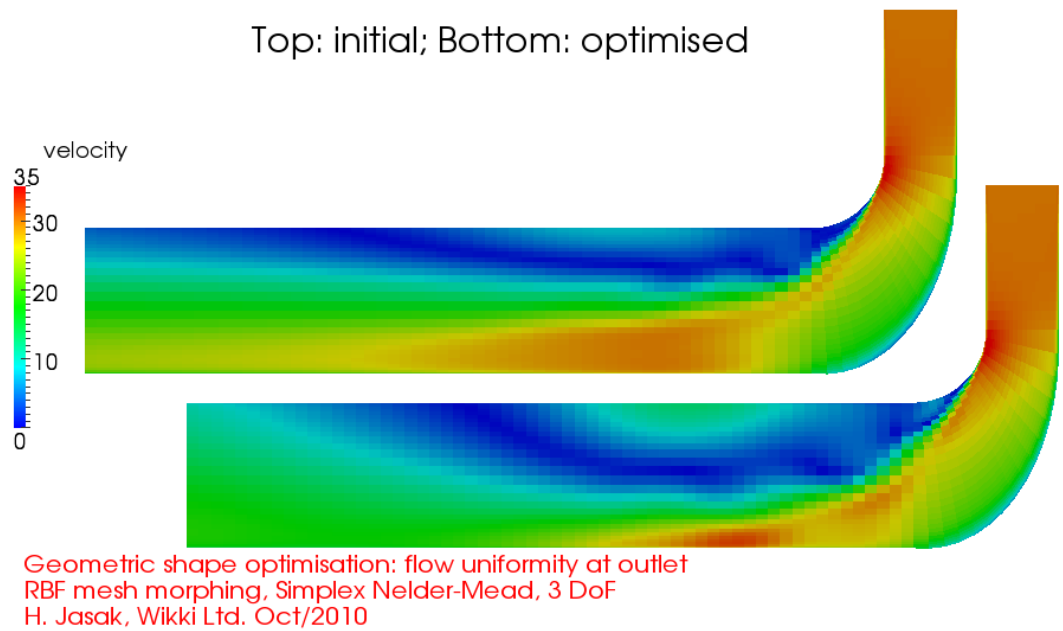


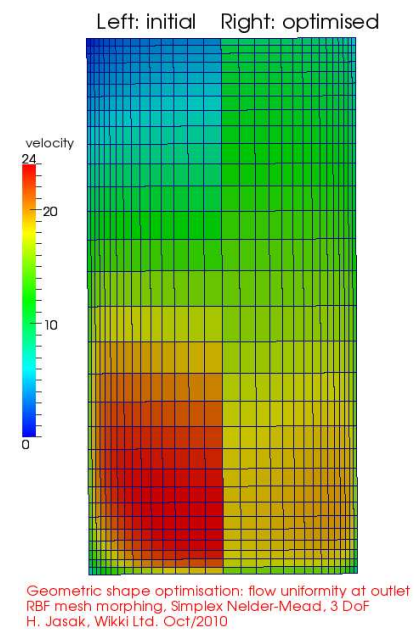
Geometric Shape Optimisation

HVAC 90 deg Bend: Flow Uniformity at Outlet

- Flow solver: incompressible steady-turbulent flow, RANS $k - \epsilon$ model; coarse mesh: 40 000 cells; 87 evaluations of objective with CFD restart
- RBF morphing: 3 control points in motion, symmetry constraints; 34 in total
- Objective: flow uniformity at outlet plane

```
iter = 0 pos = (0.9 0.1 0.1) v = 22.914 size = 0.69282
iter = 5 pos = (0.1 0.1 0.1) v = 23.0088 size = 0.584096
iter = 61 pos = ((0.990164 0.992598 0.996147) v = 13.5433 size = 0.00095
```





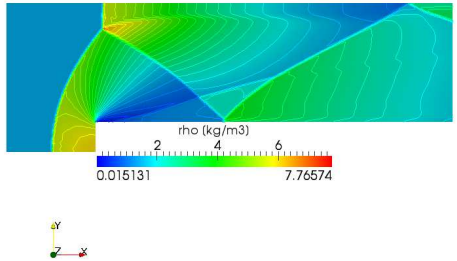
Compressible Solver: `dbnsFoam`

New Compressible Flow Solver: `dbnsFoam`

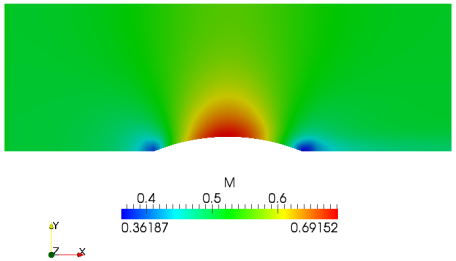
- Currently, OpenFOAM is not competitive for compressible flow with strong compressibility effects: high speed, shocks. **Block-solution is required!**
- `dbnsFoam` solver utilises **flux vector splitting** (HLLC and derivatives), flux difference splitting (Roe) and central (Local Lax-Friedrichs) schemes; multidimensional limiters are available
- Time stepping schemes: multistage and pseudo-time stepping algorithms
- Fully integrated GGI and MRF capabilities: interfacing with Turbo Tools
- FAS multigrid under testing; Implicit solution planned in the near term
- Currently focused on turbomachinery and external aerodynamics - planning for reactive flows and frequency-based methods

Examples of Use Compressible Solver: dbnsFoam

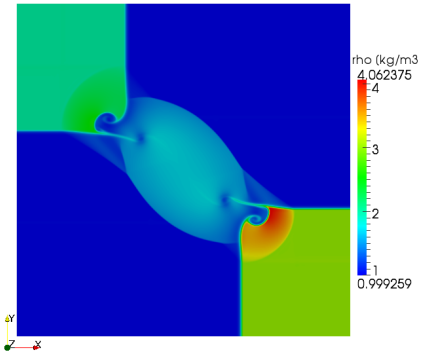
Forward facing step



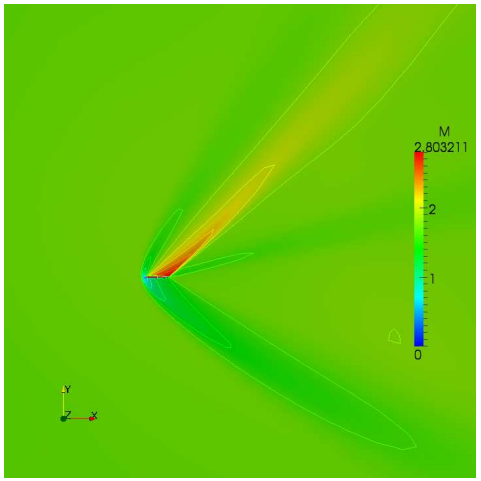
2D Bump



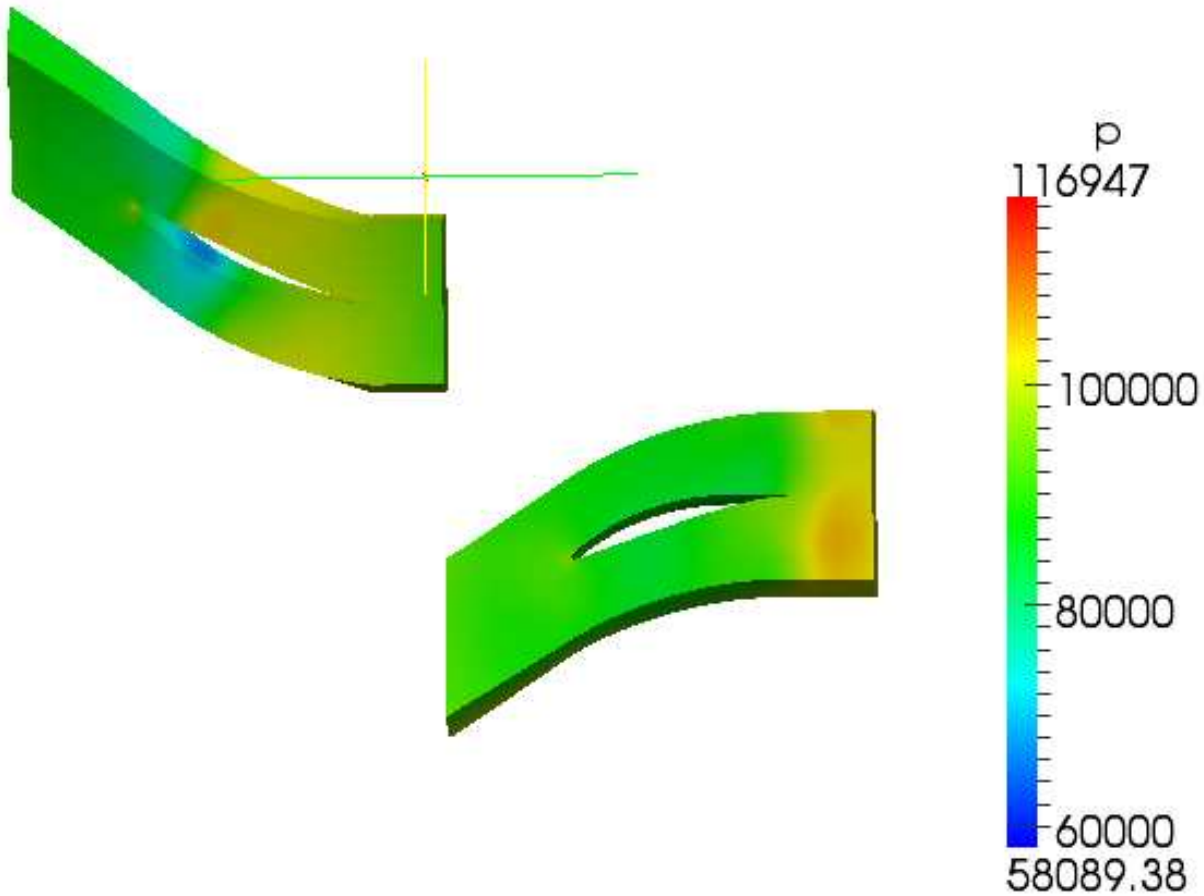
2D Riemann problem (800×800 mesh)



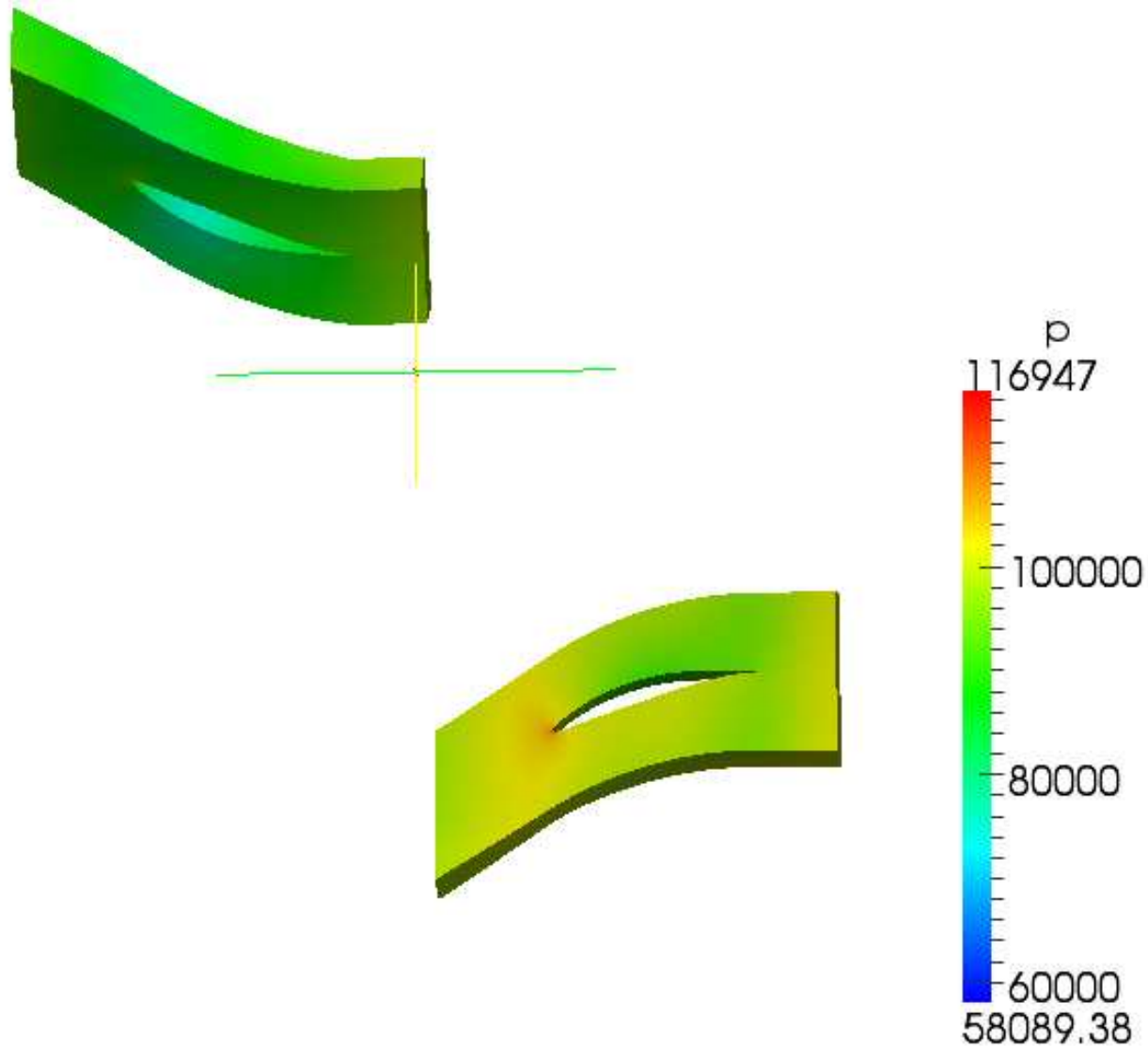
$M = 2.5$ turbulent flow over NACA0012



DBNS & Dynamic mesh



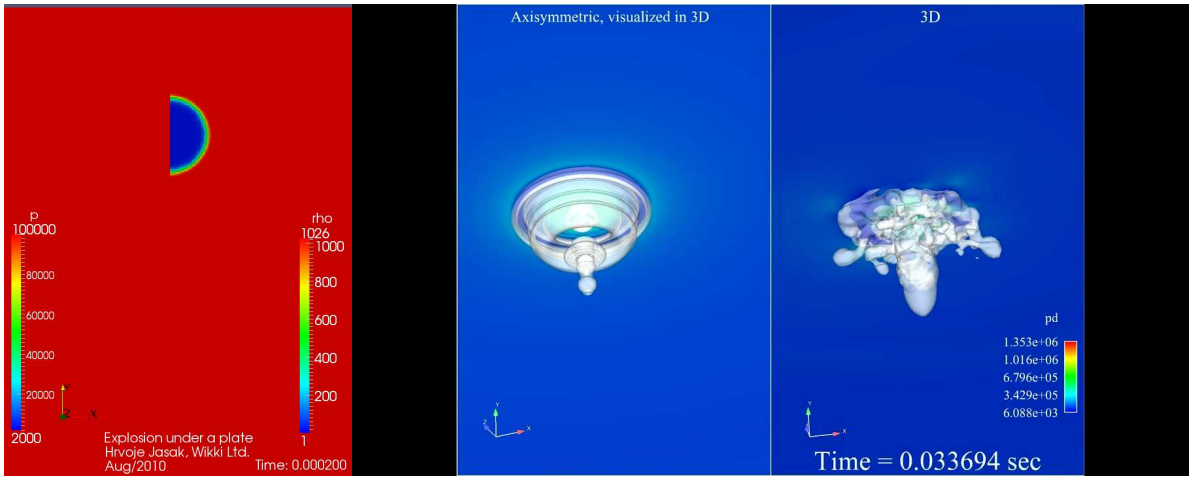
DBNS & Dynamic mesh



Multi-Phase Compressibility

Simulation of Under-Water Explosions

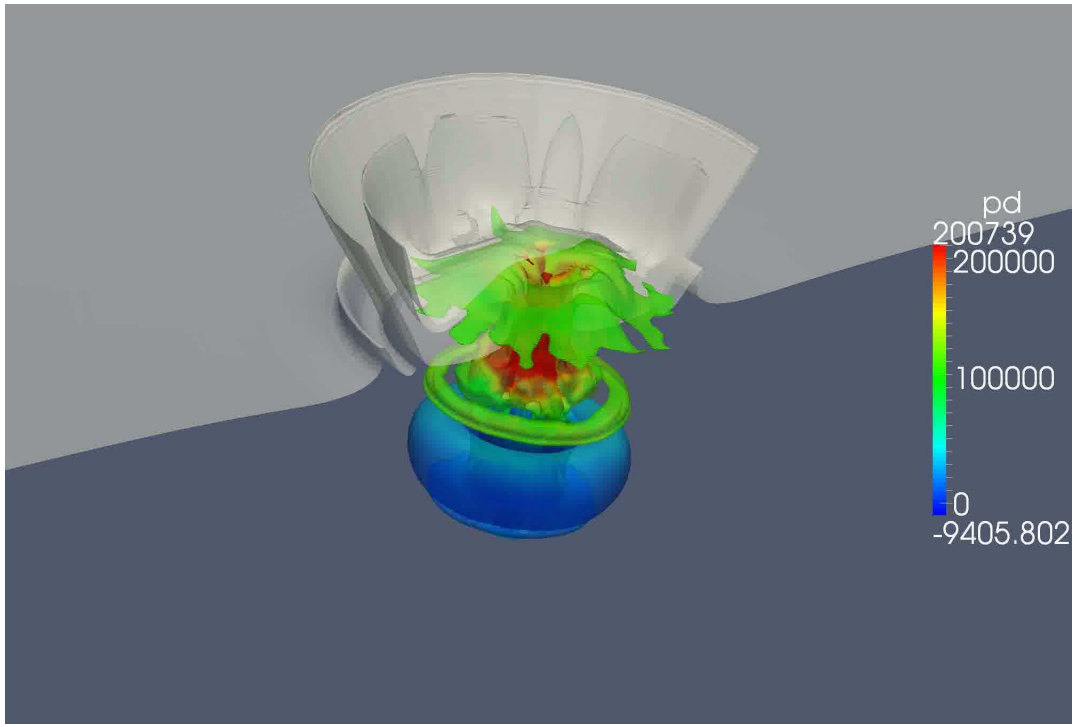
- This is ongoing collaboration with Johns Hopkins APL, Penn State University and Wikki: working hard for almost a year
- Dominating effects of compressibility in air and water: massive change in density, with propagating pressure waves
- Pressure ranges from 500 bar to 20 Pa
- Stiff numerics: collapse of over-expanded bubble due to combined compressibility of both phases are the basis of the phenomenon
- Test cases: Rayleigh-Plesset oscillation, undex under a plate, explosion
- Eric Paterson, Scott Miller, David Boger, Penn State; Ashish Nedungadi, JHU-APL



Multi-Phase Compressibility

First Simulations of Under-Water Explosions: Eric Paterson, Penn State

- Bubble of high initial pressure expands after explosion
- Initial pressure pulse is very fast - with little effect
- Bubble collapse creates re-entrant jer which pierces the free surface
- Stability problems resolved in segregated solver
- Next phase: **block-coupled** $p - \alpha$ **solution algorithm**



Project Status Summary

- OpenFOAM is a free software, available to all at no charge: GNU Public License
- Object-oriented approach facilitates model implementation
- Extensive capabilities already implemented; open design for easy customisation
- Solvers are validated in detail and match the efficiency of commercial codes
- Project work-load is impressive and shows a new way forward: customers organise in consortia to share development cost and experiences in migration
- Open Source model dramatically changes the industrial CFD landscape: new business model
- Biggest problems quoted by the community
  - **Steep learning curve for new users**
  - **Insufficient documentation**: capability is there but we do not know how to set up cases and use it to the full potential
  - Need to make **validation examples publicly accessible**
  - Concerted effort on open source **graphical user interface**

Closing

- We need to **grow the expert-level knowledge in the community**
  - 16-session OpenFOAM training at OFW-6!
  - NUMAP-FOAM 2011, 4th edition of Summer School
  - Other hosted events for expert users?
- Help to shape the community!
- **Seventh OpenFOAM Workshop**: Darmstadt University 25-28 June 2011
  `http://www.openfoamworkshop.org`