OpenFOAM: History, Current Capabilities and Future Perspectives

Hrvoje Jasak

Wikki Ltd. United Kingdom and

Faculty of Mechanical Engineering and Naval Architecture, Uni Zagreb, Croatia

1st Portuguese FOAM User Group Meeting, 10-11 July 2015

Outline

Objective

- Provide an overview of existing capabilities and future directions of OpenFOAM developments

Topics

1. Executive Overview and Historical Note
2. Overview of capabilities
3. Some new developments
   - Coupled solver framework and examples: `pUCoupledFoam`
   - `cfMesh`: native automatic parallel meshing tool
4. Examples of application
   - Modelling of visco-elastic fluid flows
   - Examples of dynamic mesh handling
   - Non-linear structural analysis and fluid-structure interaction
5. Summary

OpenFOAM: Executive Overview

What is OpenFOAM?

- **OpenFOAM** is a free-to-use Open Source numerical simulation software with extensive CFD and multi-physics capabilities
- Free-to-use means using the software without paying for license and support, including **massively parallel computers**: free 1000-CPU CFD license!
- Software under active development, capabilities mirror those of commercial CFD
- Substantial installed user base in industry, academia and research labs
- Possibility of extension to non-traditional, complex or coupled physics: Fluid-Structure Interaction, complex heat/mass transfer, internal combustion engines, nuclear reactor simulations, strongly coupled systems

Main Components

- Discretisation: Polyhedral Finite Volume Method, second order in space and time
- Lagrangian particle tracking, Finite Area Method (2-D FVM on a curved surface)
- Massive parallelism in domain decomposition mode
- Automatic mesh motion (FEM), support for topological changes
- All components implemented in library form for easy re-use
- Physics model implementation through **equation mimicking**

OpenFOAM: Historical Note

FOAM and OpenFOAM, Early Days and Current Status

- Software originally developed in 1990s at Imperial College by Henry Weller and Hrvoje Jasak. Notable contributions from the group and early spread in the community: Chalmers Uni, Uni Zagreb, Poly Milano, UC Dublin
- Nabla Ltd. and FOAM as commercial software: 2000-2004, with bulk of funding provided by Fluent Inc. (snapshot access). Nabla Ltd. wound up in 2006
- Release of FOAM into public domain by agreement of Weller and Jasak in 2004. Resolution of Copyright and authorship claims under way

Open Source Environment and Industrial CFD Work

- Currently using GNU Public License, which allows free exploitation, with some limits on access to source code: your customers have the same rights as you do
- **GPL is NOT a way to separate people from their intellectual property**
- Companies have a choice to keep the code internal and confidential or to engage with the community either before or after project completion

OpenFOAM: Historical Note

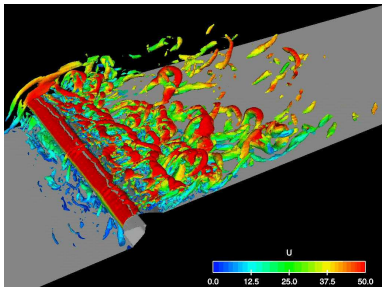
Current Status

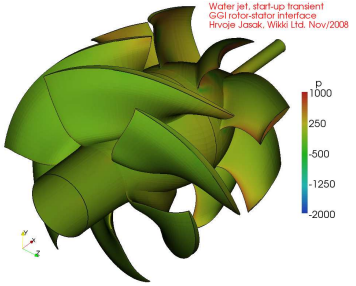
- OpenFOAM Trade Mark currently owned by ESI: supporting commercial customers
- OpenFOAM Foundation claims copyright (disputed): open contributions not allowed and transfer of copyright is demanded
- FOAM-Extend is designed to support **Open Source development framework**: community collaboration and contributions
- Particular emphasis on industrial-academic collaboration and new developments

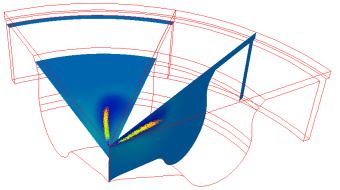
OpenFOAM: Executive Overview

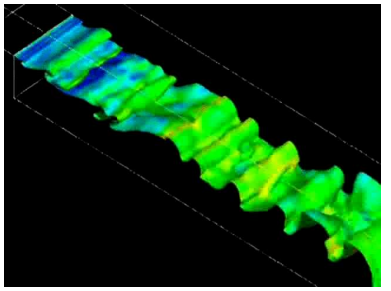
Physical Modelling Capability Highlights

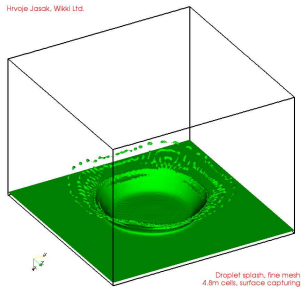
- Basic: Laplace, potential flow, passive scalar/vector/tensor transport
- Incompressible and compressible flow: segregated pressure-based algorithms
- Heat transfer: buoyancy-driven flows, conjugate heat transfer
- Multiphase: Euler-Euler, VOF free surface capturing and surface tracking
- RANS for turbulent flows: 2-equation, RSTM; full LES capability
- Pre-mixed and Diesel combustion, spray and in-cylinder flows
- Stress analysis, fluid-structure interaction, electromagnetics, MHD, etc.

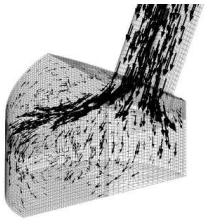


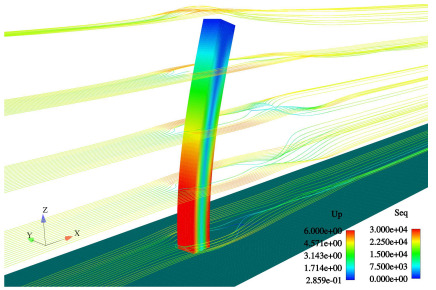


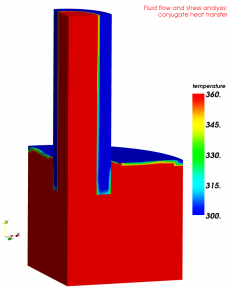


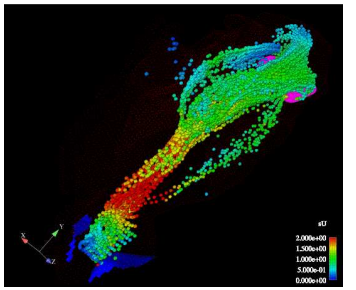








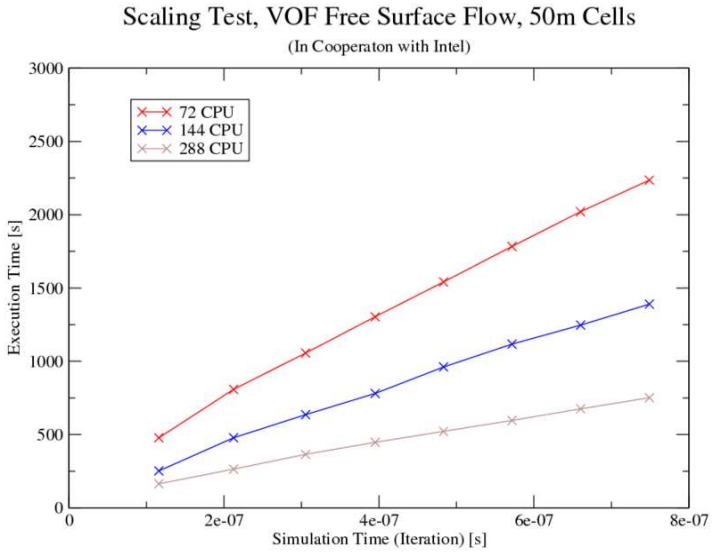


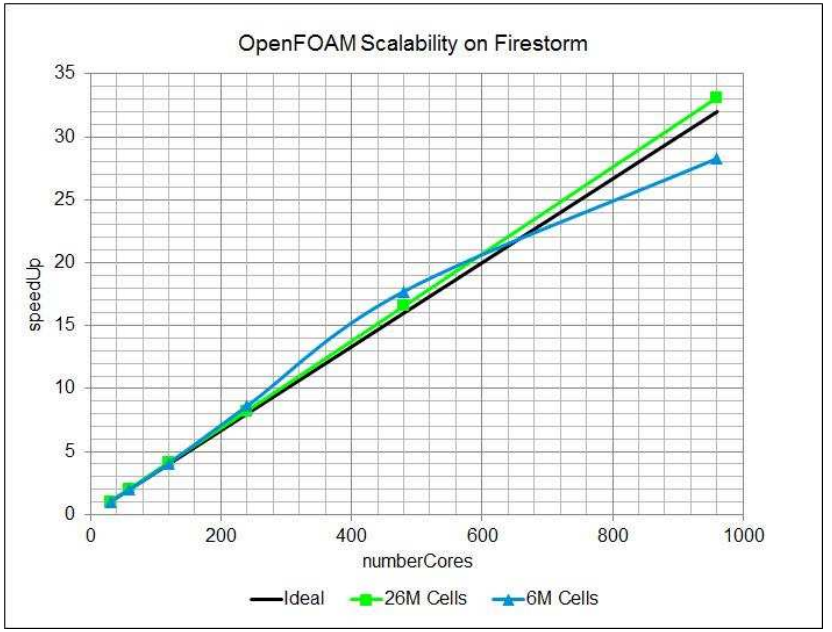


OpenFOAM and HPC

Using OpenFOAM on Large-Scale HPC Platforms

- In reality, FOAM scales to approx 4000 cores without intervention (fantastic!)
- Further fundamental work is required for large-scale machines (eg. MIRA)
  - Loss of parallel performance: need next-generation linear solver tools
  - Better integration with available hardware, eg. hierarchical communication
  - Rewrite of the I/O sub-system is needed to match the hardware layout
- In practical simulations, FOAM scales better than Fluent or Star-CCM+
- We can make FOAM scale for the next 10 years, with a reasonable effort





foam-extend-3.2

Major developments in foam-extend-3.2

- General software and deployment improvements
  - **Global control switches**: no central dictionaries: Martin Beaudoin
  - Automated generation of config files from bashrc: Bernhard Gschaider
  - Native `long long` and `long double` support
  - Intel 15.0 and CLang port
- Numerics update: new features
  - Integrated release of **Immersed Boundary Method**
  - Integrated release of the **Liquid Film Model**: finite area method with volume-to-surface coupling tools
  - Upgrade of segregated compressible turbomachinery solvers
  - Incremental development of the coupled solver and **block p-U solver upgrade**: performance, porous, MRF (compressible = WIP)
  - AMG and block-AMG upgrade: solo cells, robustness and parallel performance
- Utilities, meshing and others
  - Integrated release and feature upgrade for `cfMesh`
  - . . . and many others: see release notes

Coupled p-U Solver

Coupled Implicit p-U Solver: `pUCoupledFoam` and `MRFPorousFoam`

```
fvVectorMatrix UEqn
(
    fvm::div(phi, U)
  + turbulence->divDevReff(U)
);
fvScalarMatrix pEqn
(
   - fvm::laplacian(rUAf, p) == -fvc::div(fvc::grad(p))
);
BlockLduSystem<vector, vector> pInU(fvm::grad(p));
BlockLduSystem<vector, scalar> UInp(fvm::UDiv(U));

fvBlockMatrix<vector4> UpEqn(Up);
UpEqn.insertEquation(0, UEqn);
UpEqn.insertEquation(3, pEqn);
UpEqn.insertBlockCoupling(0, 3, pInU, true);
UpEqn.insertBlockCoupling(3, 0, UInp, false);

UpEqn.solve();
UpEqn.retrieveSolution(0, U.internalField());
UpEqn.retrieveSolution(3, p.internalField());
```

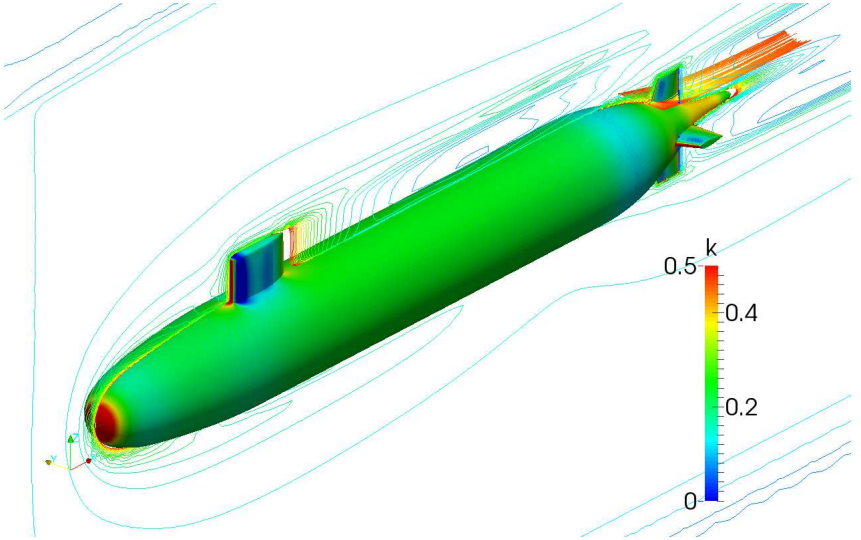
Coupled p-U Solver

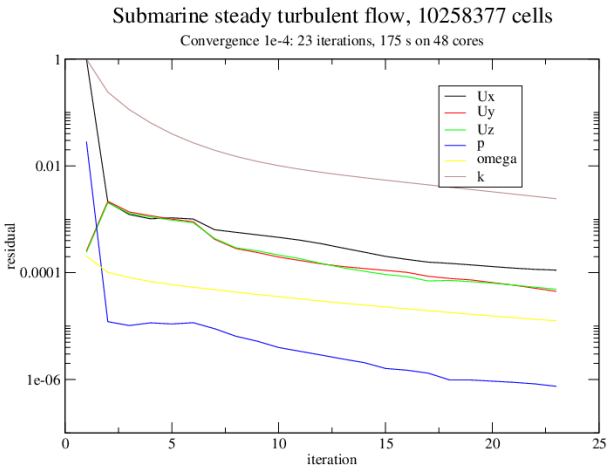
Open Source Implementation of the Coupled Implicit p-U Solver

- Coupled solver reproduces the results of the segregated algorithm
- Full integration with the `finiteVolume` library: re-use the existing and validated discretisation operators and boundary conditions without re-coding

Coupled Solver Performance

- Example: steady turbulent flow around a submarine, 10.2 million cells
- Turbulent $k - \omega$ SST model converges to 1e-4 in 23 iterations, 175 s on 48 cores

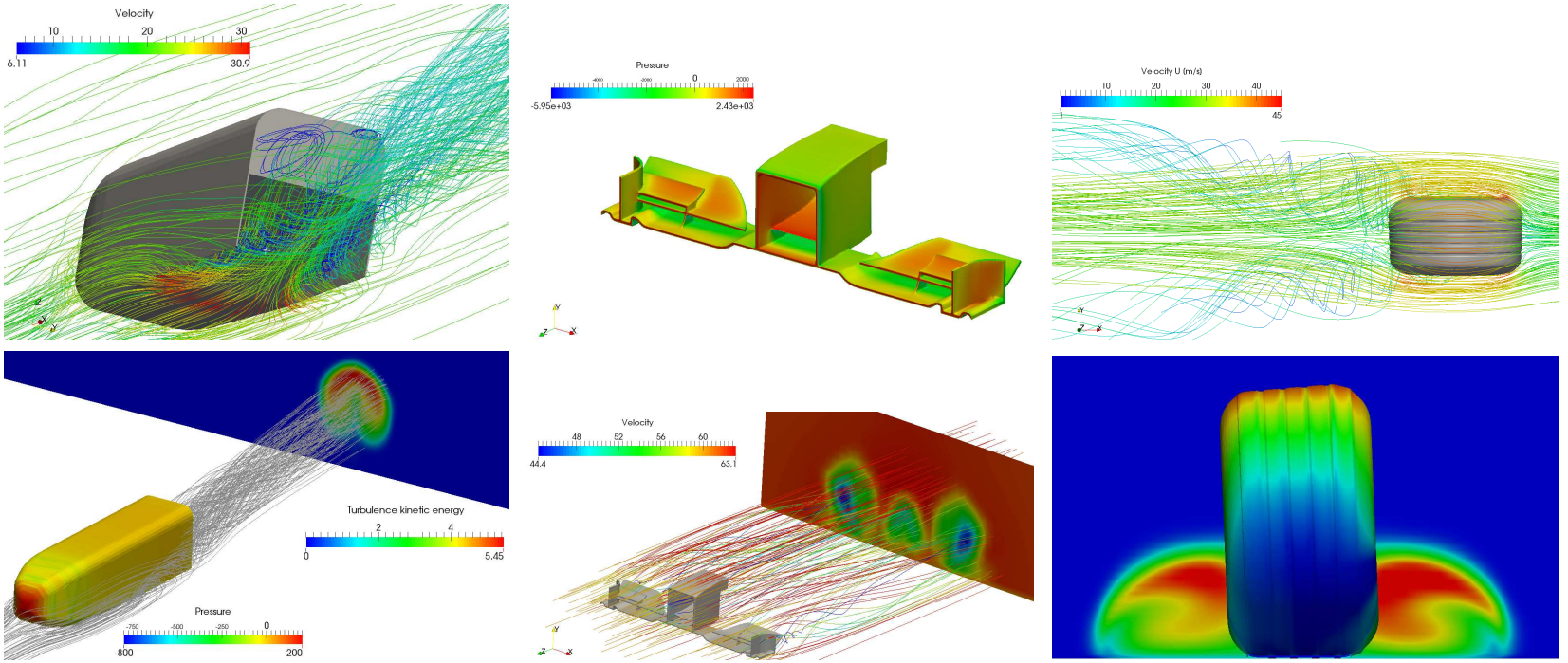




Coupled p-U Solver

Who Needs a Coupled Solver?

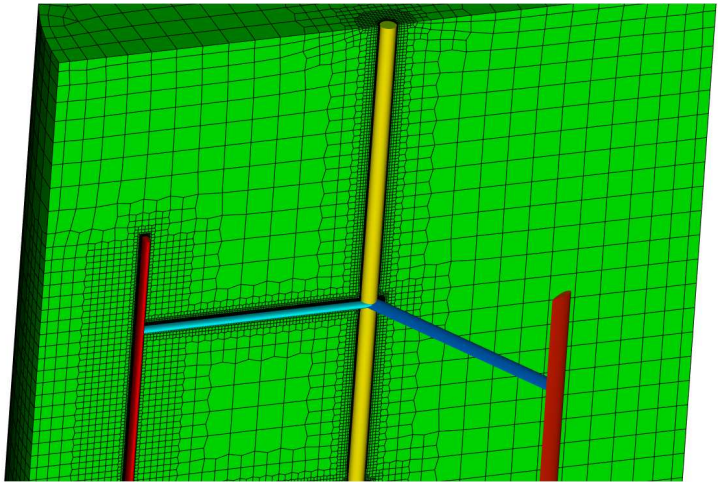
- Improved speed-to-solution and reliability on difficult meshes
- Robust discretisation: fewer settings
- Improved parallel scaling: fewer processor communication calls
- Work-in-progress: coupled turbulence model equations

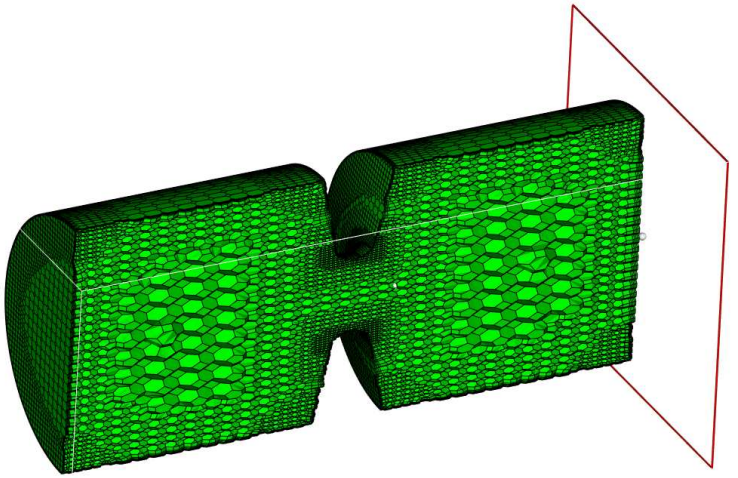


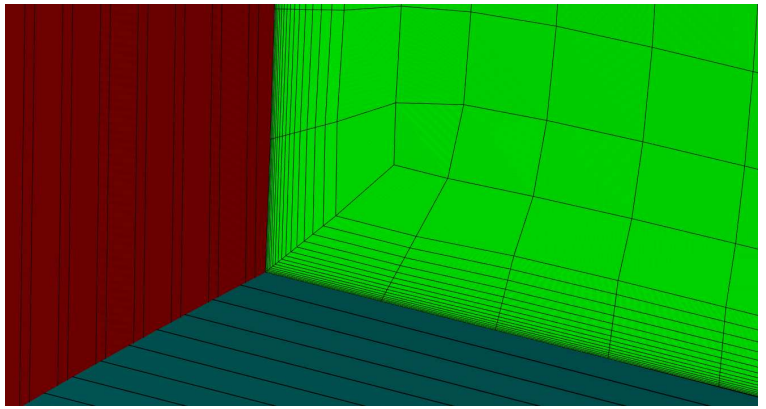
cfMesh **Update**

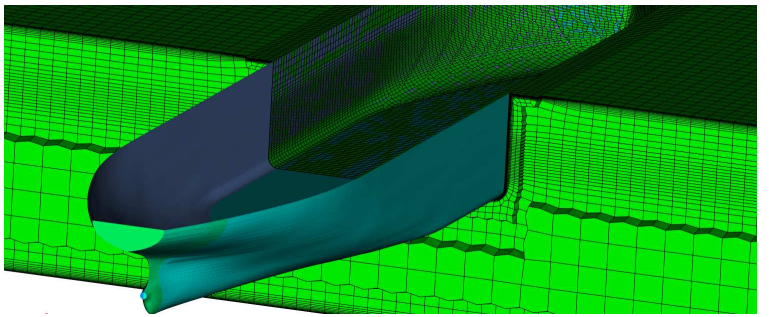
cfMesh Mesher Update by Creative Fields: Some New Features

- Generation of high quality low-Re boundary layers with high aspect ratio
- Anisotropic mesh generation in hex and polyhedral meshes: stretching algorithm is independent of mesh type









Viscoelastic Flow Model

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

- Viscoelastic flow model:

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

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

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

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

  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 implied viscoelastic (tensorial) viscosity:

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

- Split complete stress into implicit and explicit component

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

- Implicit component contains (large) tensorial viscosity, based on equilibrium stress and stabilises the momentum equation and coupling
- Solution algorithm: SIMPLE p-U coupling, with FEM-like handling of stress terms

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 behaviours in each phase
- Support for topological changes: syringe ejection

Dynamic Mesh

Simulations with Time-Varying Geometry in CFD

- In many simulations, shape of computational domain changes during the solution, either in a prescribed manner or as a part of the solution
- In cases of **prescribed motion**, it is possible to pre-define a sequence of meshes or mesh changes which accommodate the motion
- . . . but mesh generation now becomes substantially more complex
- **Solution-dependent motion** implies the shape of computational domain is a part of the solution itself: depends on solution parameters
- Examples of solution-dependent motion
  - Contact stress analysis: shape and location of contact region is unknown
  - Free surface tracking simulation: unknown shape of free surface
  - Fluid-structure interaction

Dynamic Mesh

Definition of Automatic Mesh Motion

- Automatic mesh motion will determine the position of mesh points based on the prescribed boundary motion
- Motion will be obtained by solving a **mesh motion equation**, where boundary motion acts as a boundary condition
- Properties of motion equation: preserve spatial consistency
- The “correct” space-preserving equation is a large deformation formulation of linear elasticity . . . but it is too expensive to solve
- Choices for a simplified mesh motion equation:
  - Spring analogy: insufficiently robust
  - Linear + torsional spring analogy: complex, expensive and non-linear
  - Laplace equation with constant and variable diffusivity

$$\nabla_{\bullet}(k\nabla\mathbf{u}) = 0$$

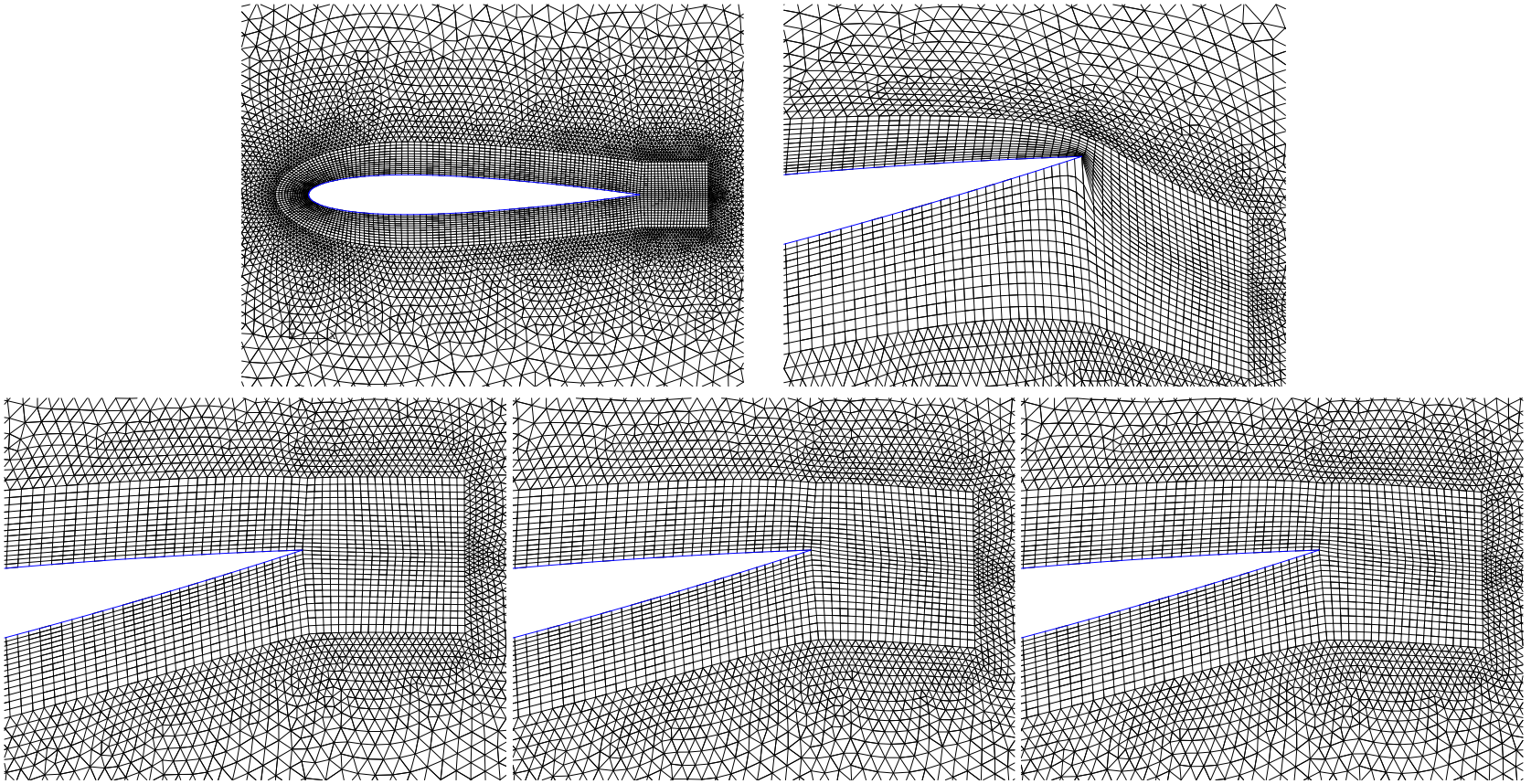
  - Linear pseudo-solid equation for small deformations

$$\nabla_{\bullet}[\mu(\nabla\mathbf{u} + (\nabla\mathbf{u})^T) + \lambda\mathbf{I}\,\nabla_{\bullet}\mathbf{u}] = 0$$

Dynamic Mesh

Effect of Variable Diffusivity: Oscillating Airfoil Simulation

- Distance-based diffusivity $1/l^2$; deformation energy; distortion energy
- Moving mesh and flow solution



Dynamic Mesh

Finite Volume Moving Mesh Support

- Definition of conservation laws will involve a moving volume rather than a stationary one, where $\mathbf{u}_b$ is the “mesh velocity”
- Additional terms relate to the change of cell volume and mesh motion fluxes

$$\frac{d}{dt}\int_V \phi dV + \oint_S d\mathbf{s}\bullet(\mathbf{u}-\mathbf{u}_b)\phi = \oint_S d\mathbf{s}\bullet\mathbf{q}_\phi + \int_V s(\phi)dV$$

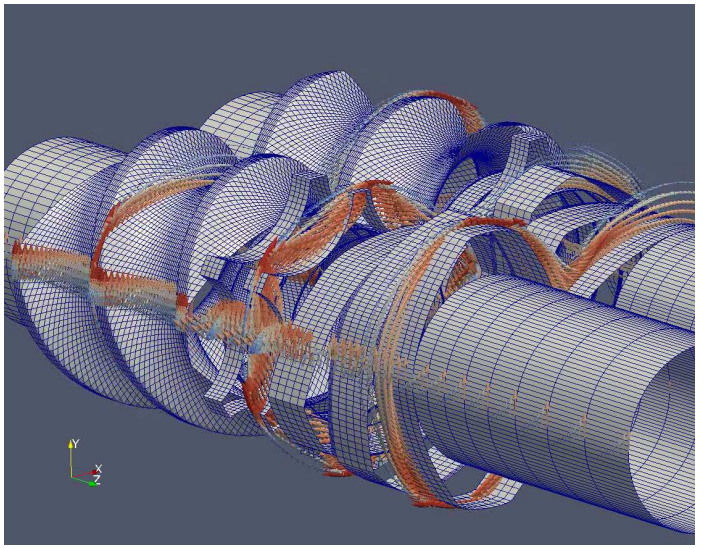
$$\frac{d}{dt}\int_V dV - \oint_S d\mathbf{s}\bullet\mathbf{u}_b = 0$$

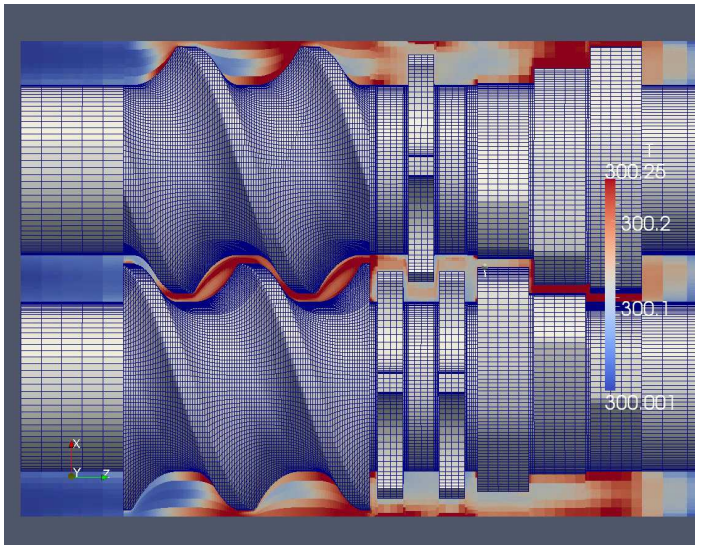
$$\oint_S d\mathbf{s}\bullet\mathbf{u}_b = \sum_f \int_{S_f} d\mathbf{s}\bullet\mathbf{u}_b = F_m$$

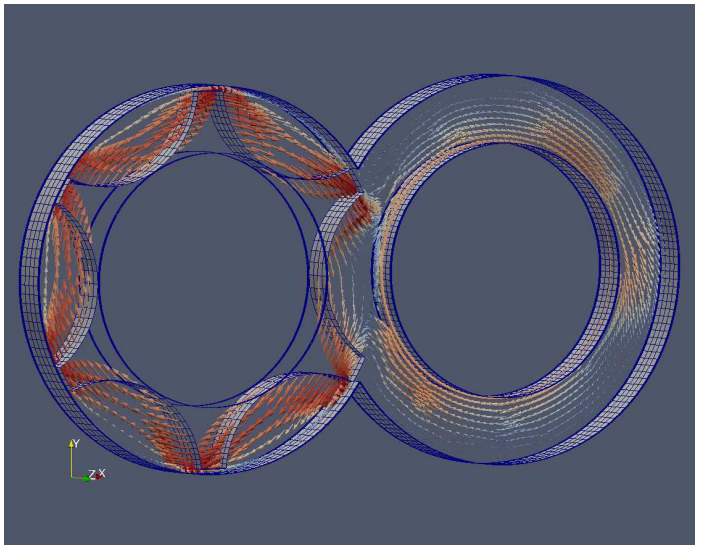
- Volume change appears in the rate-of-change term and is handled automatically
- Mesh motion flux appears in all convection terms and needs to be accounted for algorithmically
- Note: in incompressible flows, there are two possible formulations on the pressure equation, working either with relative or absolute fluxes. As a result, moving mesh solvers are not yet consistently integrated with static mesh solvers (efficiency)

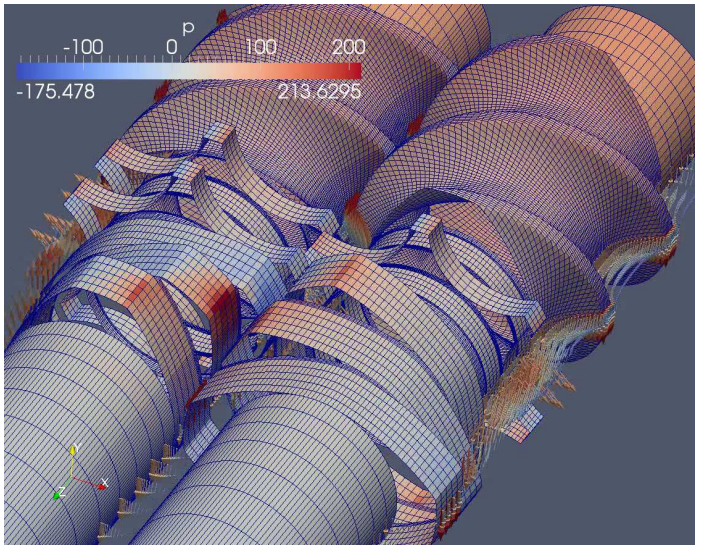
Dynamic Mesh

Dynamic Mesh Examples of Complex Combination of Motion and Sliding



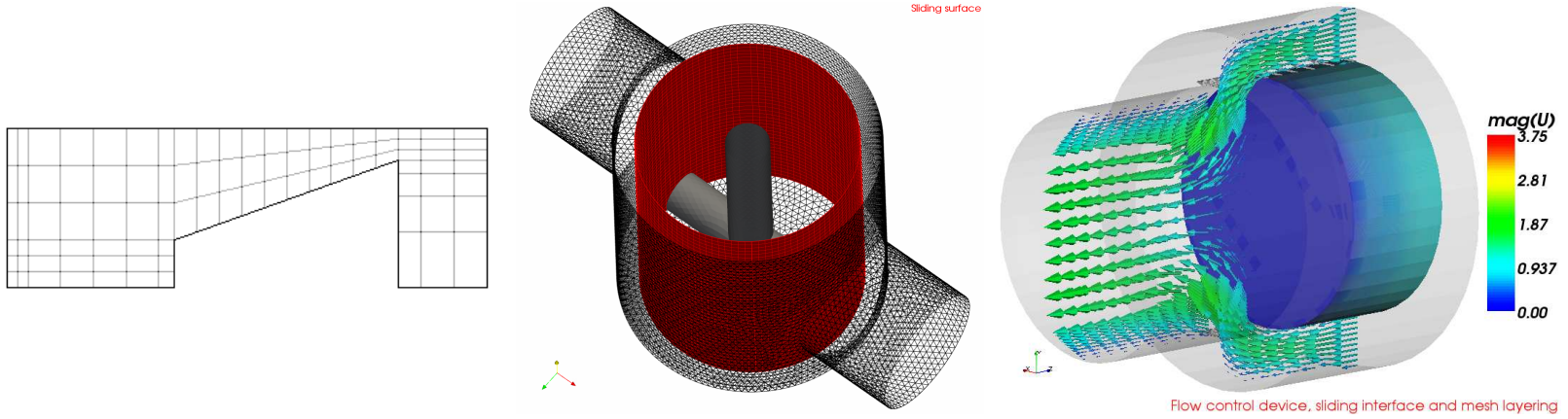






Dynamic Mesh

Topological Changes on Polyhedral Meshes



- For extreme cases of mesh motion, changing point positions is not sufficient to accommodate boundary motion and preserve mesh quality
- Definition of a **topological change**: number or connectivity of points, faces or cells in the mesh changes during the simulation
- Motion can be handled by the FVM with no error (moving volume), while a topological change requires additional algorithmic steps
- Cell insertion and deletion will formally be handled as a combination of mesh motion (collapsing cells and faces to zero volume/area) and a change in connectivity after the face and cell collapse

Dynamic Mesh

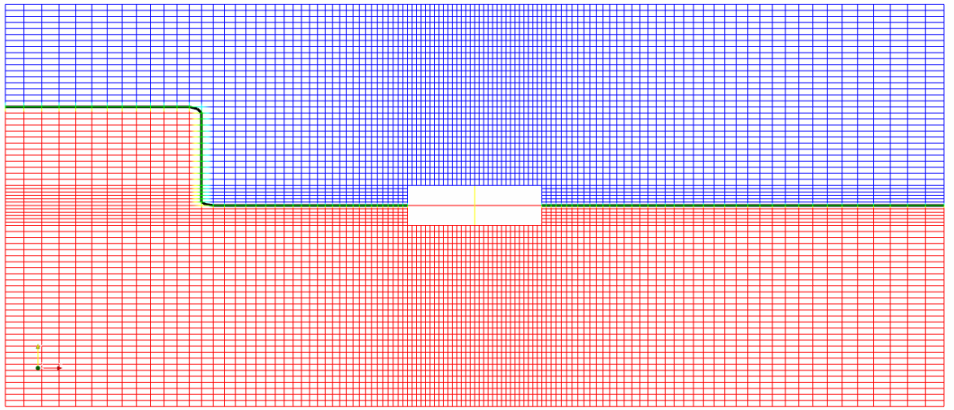
Implementation of Topological Changes in OpenFOAM

- **Primitive mesh operations**
  - Add/modify/remove a point, a face or a cell
  - This is sufficient to describe all cases, even to to build a mesh from scratch
  - . . . but using it directly is very inconvenient
- **Topology modifiers**
  - Typical dynamic mesh operations can be described in terms of primitive operations. Adding a user-friendly definition and triggering logic creates a “topology modifier” class for typical operations
  - Some implemented topology modifiers
    - Attach-detach boundary
    - Cell layer additional-removal interface
    - Sliding interface
    - Error-driven adaptive mesh refinement
- **Dynamic meshes**
  - Combining topology modifiers and user-friendly mesh definition, create dynamic mesh types for typical situations
  - Examples: mixer mesh, 6-DOF motion, IC engine mesh (valves + piston), solution-dependent crack propagation in solid mechanics

Dynamic Mesh

Example: Single Floating Body in Free Surface Flow (VOF)

- Single phase VOF free surface flow model with accurate pressure reconstruction
- 6-DOF force balance for solid body motion: solving an ODE
- Variable diffusivity Laplacian motion solver with 6-DOF boundary motion as the boundary condition condition



Problem Setup

1. Specify mesh, material properties and initial + boundary flow conditions
2. Dynamic mesh type: `sixDofMotion`. Mesh holds `floatingBody` objects
3. A floating body holds 6-DOF parameters: mass, moment of inertia, support, forces
4. Flow solver only sees a `dynamicMesh`: encapsulated motion

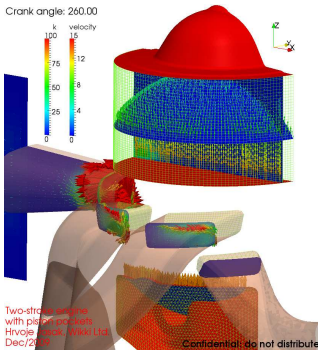
Dynamic Mesh

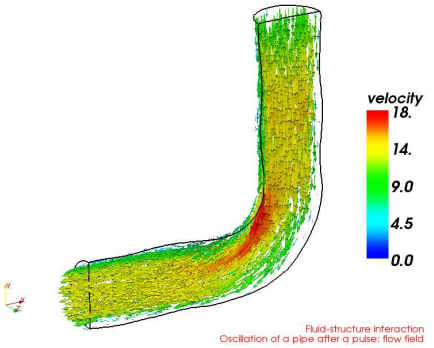
Example: In-Cylinder Flow with Moving Piston and Valves

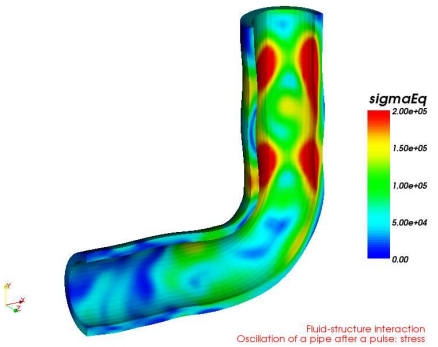
- Exhaust and intake stroke in a 2- and 4-stroke engine
- Moving piston and operating valves using topological changes
- Interacting topological modifiers and mesh motion handled by the mesh class
- Politecnico di Milano: combining mesh deformation, topological changes and re-meshing to achieve optimum resolution and quality (SAE 2007-01-0170)

Example: Fluid-Structure Interaction

- Formally, considerably more complex than floating body cases
- . . . but due to automatic mesh motion only limited changes are needed
- Dynamic mesh class transfers data on surface and uses automatic motion
- (Close coupling: shared matrix format or Reduced Rank Extrapolation)



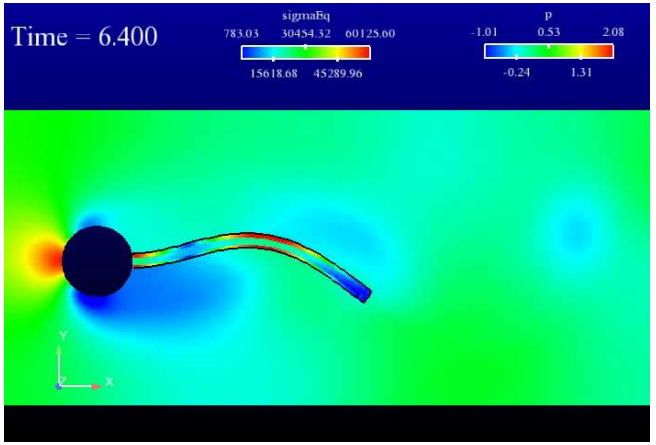


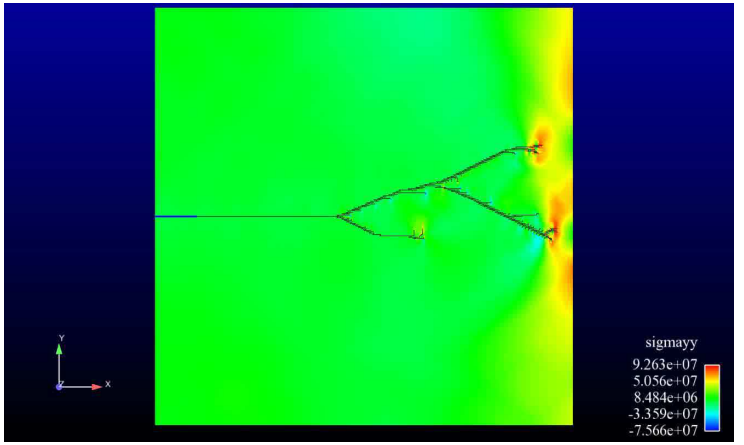


Fluid-Structure Interaction

Highly Non-Linear Structural Analysis and FSI

- As a Continuum Mechanics solver, FOAM 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 FOAM 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





Compressible Free Surface Flow

Compressible Free Surface Gas-Liquid Flow

- Conservation of mass

$$\frac{\partial \rho}{\partial t} + \nabla \bullet (\rho \, \mathbf{u}) = 0$$

- Phase mass conservation equation

$$\frac{\partial (\rho \alpha)}{\partial t} + \nabla \bullet (\rho \mathbf{u} \, \alpha) = 0$$

- Momentum equation

$$\frac{\partial (\rho \mathbf{u})}{\partial t} + \nabla \bullet (\rho \mathbf{u} \, \mathbf{u}) - \nabla \bullet \boldsymbol{\sigma} = \rho \mathbf{g}$$

- Phase compressibility: ideal gas law and constant speed of sound for liquid

$$\rho_G = \frac{p}{RT} = \psi_G \, p, \quad \psi_G = \frac{\partial \rho}{\partial p} = f(p)$$

$$\rho_L = \rho_0 + \psi_L (p - p_0), \quad \psi_L = const.$$

Compressible Free Surface Flow

Derivation of the Compressible Form of Mass Conservation Equations

- Introducing chain rule:

$$\frac{\partial \rho}{\partial t} = \frac{\partial \rho}{\partial p}\frac{\partial p}{\partial t}$$

reorganising the terms, and dividing through by ρ_G

$$\frac{\partial \alpha}{\partial t} + \mathbf{u}\bullet\nabla\alpha = -\frac{\alpha}{\rho_G}\psi_G\left(\frac{\partial p}{\partial t} + \mathbf{u}\bullet\nabla p\right) - \alpha\nabla\bullet\mathbf{u}$$

- **Pressure equation** follows from global mass conservation, by summing up phase mass conservation equations

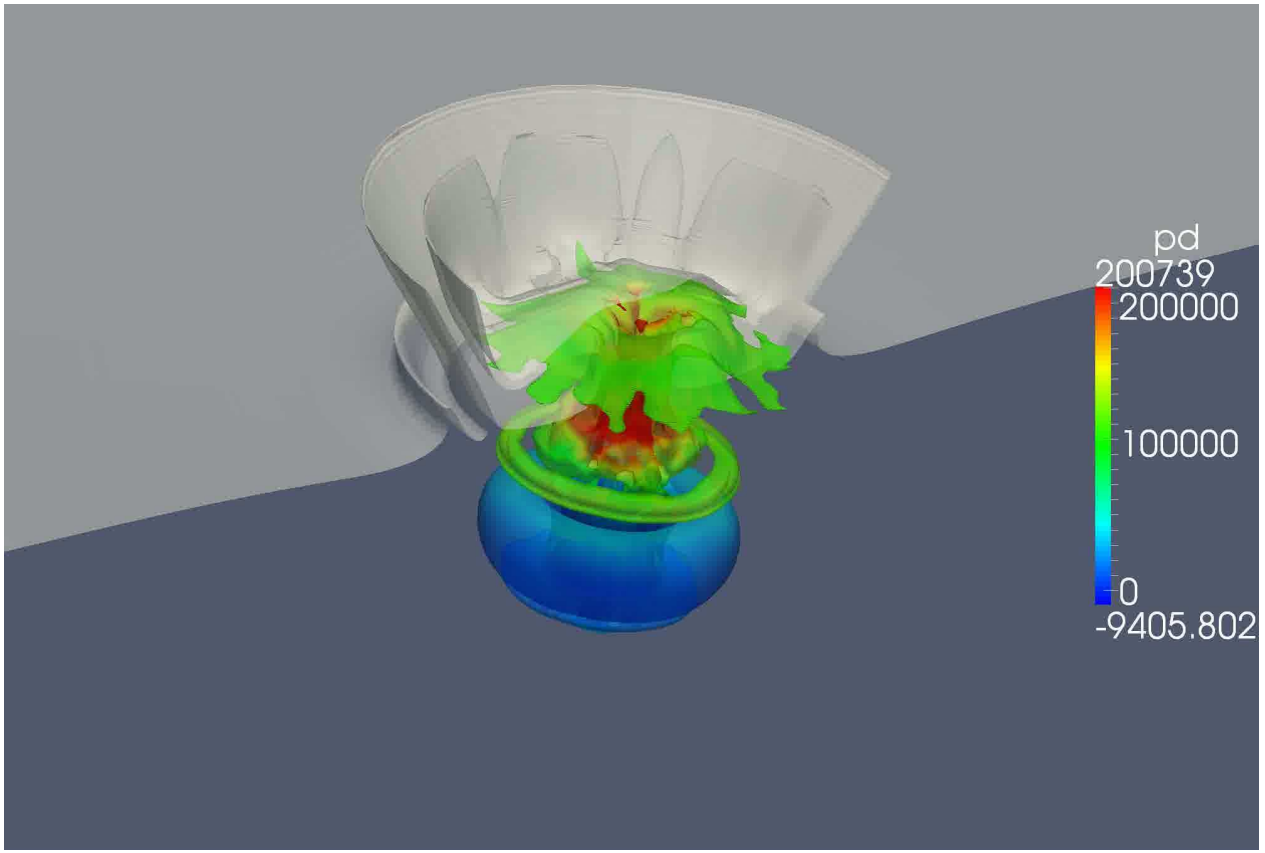
$$\left(\frac{\alpha\psi_1}{\rho_1} + \frac{(1-\alpha)\psi_2}{\rho_2}\right)\left(\frac{\partial p}{\partial t} + \nabla\bullet(\mathbf{u}\,p) - p\nabla\bullet\mathbf{u}\right) + \nabla\bullet\mathbf{u} = 0$$

- Handling of the $\nabla\bullet\mathbf{u}$ term is done in the “incompressible manner”, creating a pressure Laplacian. In the absence of compressibility, the pressure equation reverts to the incompressible formulation

Compressible Free Surface Flow

Under-Water Explosion Under a Free Surface: Eric Paterson, Virginia Tech

- 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



FOAM in the Community

- Benefit of Open Source code is easier sharing of work and collaborative community-based development
- To gain most from FOAM, it is essential to build a strong community
- New developments and collaborative work produces very exciting results
- FOAM is gaining strength in the industrial and academic community: **this is a good time to get involved!**

NUMAP-FOAM Summer School, Uni Zagreb Croatia, 25/Aug-5/Sep/2015

11th OpenFOAM Workshop, Guimarães, Portugal, Summer 2016

Acknowledgement

Acknowledgement

- Credit for p-U coupled solver: Klas Jareteg Chalmers, Vuko Vukčević FSB, University of Zagreb
- Compressible free surface flow solver: Scott Miller, Eric Paterson, David Boger, Ashish Nedungadi, Penn State, Johns Hopkins APL