Dynamic Mesh Handling in OpenFOAM

Hrvoje Jasak

h.jasak@wikki.co.uk

Wikki Ltd, United Kingdom

Introduction

Objective

- Present the layout and use of dynamic mesh features in OpenFOAM
- Demonstrate how object orientation makes it easy to set up complex interacting motion and topological changes

Topics

- Definition of a mesh motion problem
- Geometry handling in OpenFOAM: polyhedral cell support
- Deforming mesh strategies
- FE-based automatic mesh motion solver for the FVM
- Support in the Finite Volume Discretisation
- Topological mesh changes
- Algorithmic implications
- Dynamic mesh classes with examples: 6-DOF bodies, IC engines, FSI
- Advanced topics: RBF motion and morphing, immersed boundary method, overset mesh
- Summary

Background

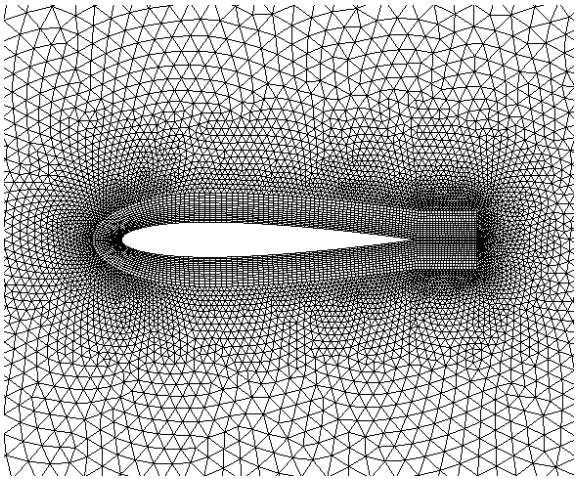
Simulations with Time-Varying Geometry in CFD

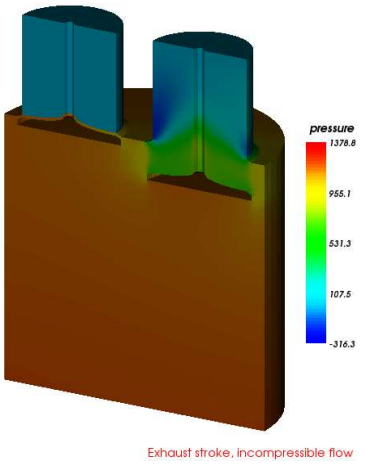
- In many simulations, shape of computational domain changes during the solution, either in a prescribed manner or as a part of the solution
- In cases of **prescribed motion**, it is possible to pre-define a sequence of meshes or mesh changes which accommodate the motion
- . . . but mesh generation 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

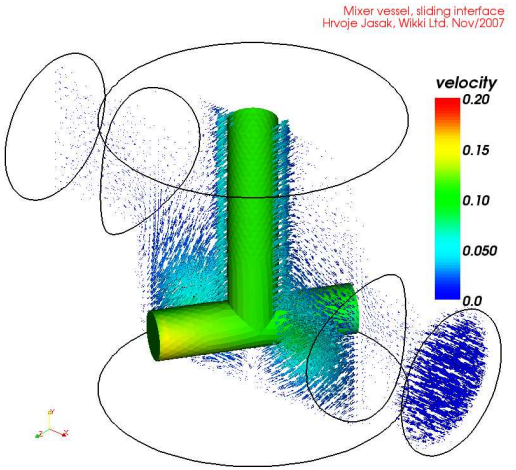
Prescribed Mesh Motion

Example: Prescribed Mesh Motion

- Domain shape is changing during the simulation in a prescribed manner
- Motion is known and independent of the solution
- . . . but it is usually only prescribed at boundaries
- Definition of moving mesh involves point position and mesh connectivity for every point and cell for every time-step of the simulation. This is typically defined with reference to a pre-processor or parametrically in terms of motion parameters (crank angle, valve lift curve, etc.)
- Solution-dependent mesh changes are out of the question: eg. mesh refinement
- Can we simplify mesh generation for dynamic mesh cases?



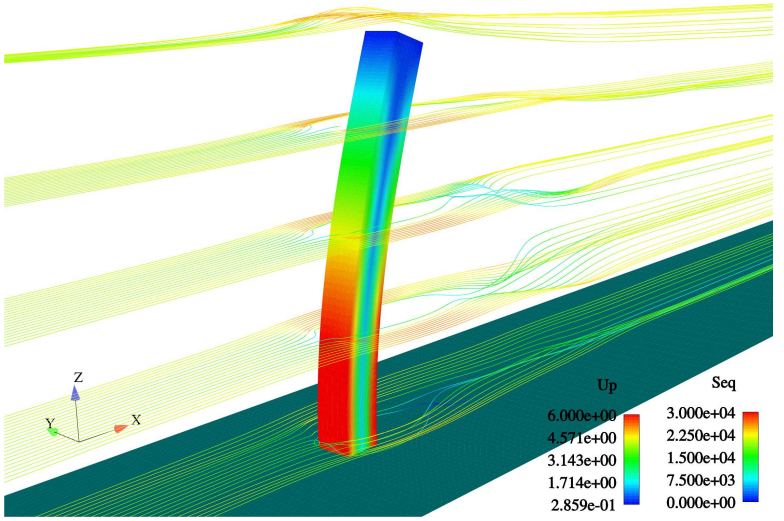


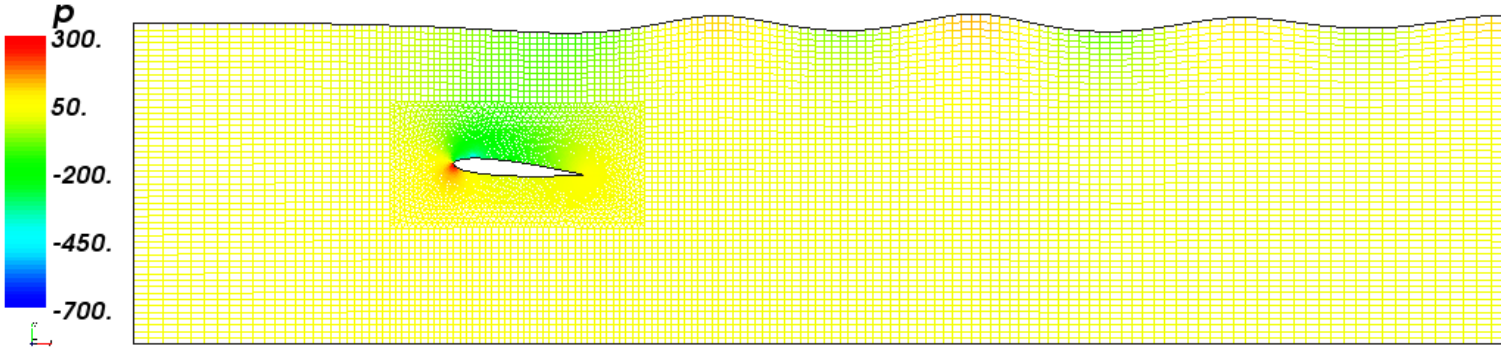


Solution-Dependent Motion

Example: Solution-Dependent Motion

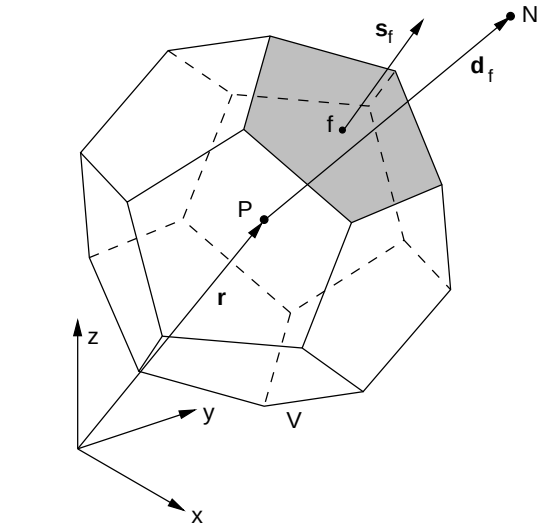
- External shape of the domain is unknown and a part of the solution
- By definition, it is impossible to pre-define mesh motion a-priori
- In all cases, it is the **motion of the boundary** that is known or calculated
- Automatic mesh motion determines the position of internal points based on boundary motion

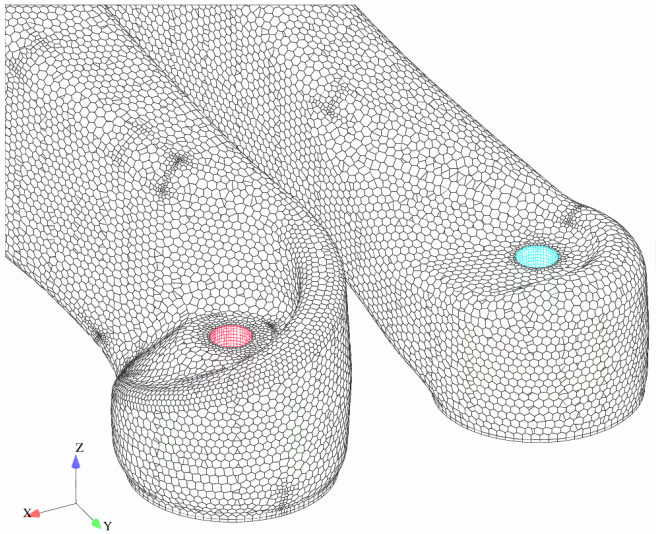




Surface tracking: hydrofoil

Geometry Handling





Handling Complex Geometry in OpenFOAM

- Complex geometry is a rule, not exception
- Polyhedral cell support
  - Polyhedral cell with polygonal faces
  - Consistent handling of all cell types
  - More freedom in mesh generation
- Interfaces to mesh generation packages

Native Automatic Mesh Generation

- Two techniques under active development, based on STL surface geometry description
- Polyhedral dual mesh; surface-adjusted octree hexahedral mesh

Dynamic Mesh Handling

- Supporting cases of deforming geometry using automatic mesh motion solvers
- For extreme mesh deformation, mesh topology is modified during the simulation using the topology change engine

Automatic Mesh Motion Solver

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

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

Automatic Mesh Motion Solver

Control of Mesh Spacing and Discretisation Error

- Mesh spacing and quality control by **variable diffusivity** (Tuković, 2005)
- Changing diffusivity implies re-distribution of the boundary motion through the volume of the mesh: it is not necessarily good to absorb deformation next to the moving boundary
  - Distance-based methods: $1/L$, quadratic or exponential distance from a moving boundary
  - Quality- and distortion-based methods

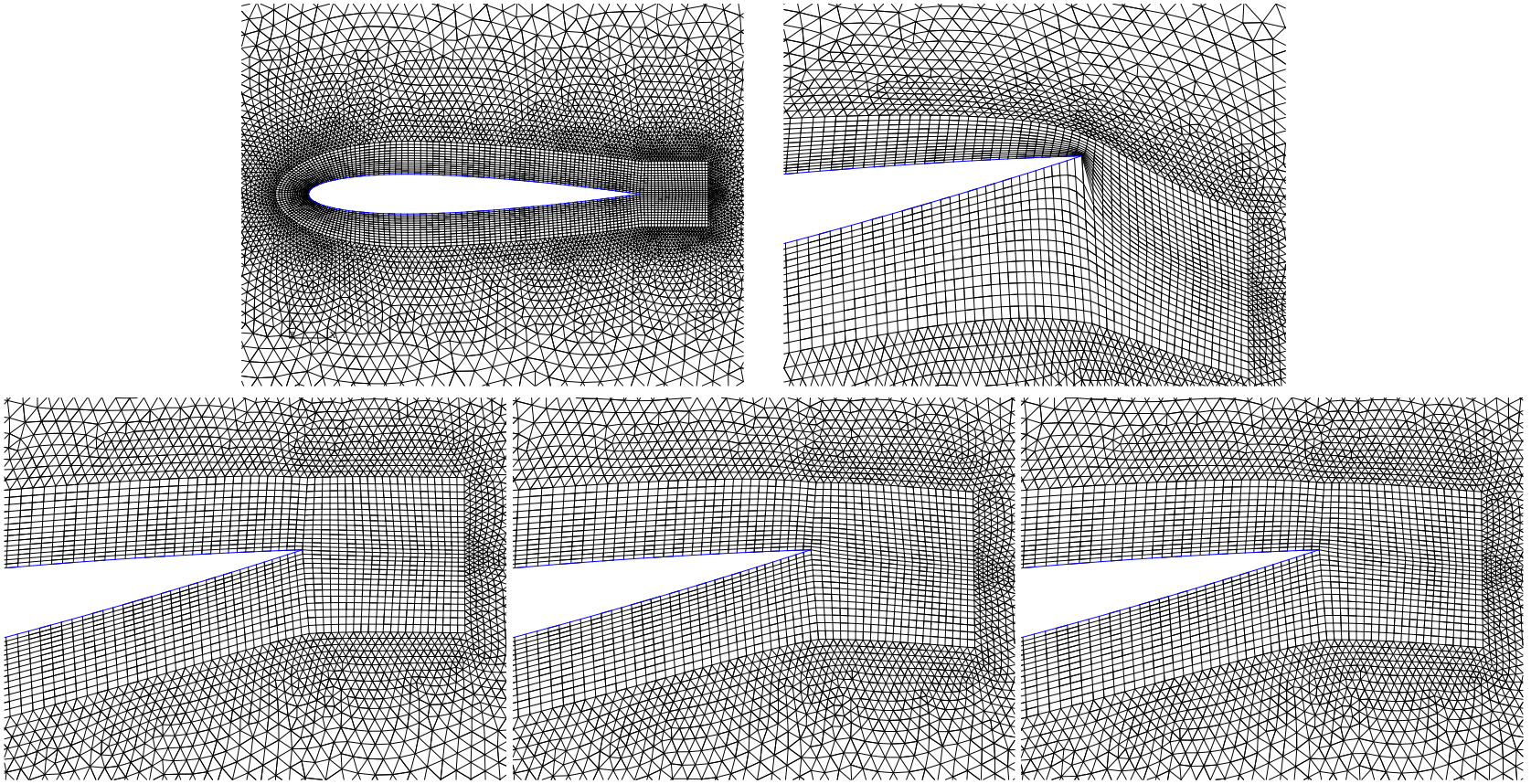
Preserving Mesh Quality

- Definition of valid motion from an initially valid mesh implies that no faces or cells are inverted during motion
- Face area and cell volume on polyhedral meshes is calculated using triangular/tetrahedral decomposition of the cell of face. Motion validity is guaranteed if no triangles or tetrahedra in the cell and face decomposition are inverted during motion
- The rest reduces to controlling the discretisation error in the motion equation

Automatic Mesh Motion

Effect of Variable Diffusivity: Oscillating Airfoil Simulation

- Initial mesh; constant diffusivity
- Distance-based diffusivity $1/l^2$; deformation energy; distortion energy



Choice of Motion Solvers

Motion Solver Equations in OpenFOAM

- Second-order Finite Element method with polyhedral support
  - Vertex-based method: no interpolation required
  - FE shape function does not allow tetrahedral or triangular flip: perfect for large boundary deformation
  - Mini-element technique involves enriching the point set: more equations but lower discretisation error
  - Choice of element decomposition
    - **Cell decomposition**: additional point in every cell centroid
    - **Cell-and-face decomposition**: additional point in cell and face centroid
  - Validated and efficient for large deformation, eg. internal combustion engines
- Cell-based methods
  - Solving motion equation on cell centres, interpolating motion into points
  - Special corrections and extrapolation on corners and patch intersections
  - Smaller equation set, but problems in cell-to-point interpolation

Moving Mesh Finite Volume Method

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

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

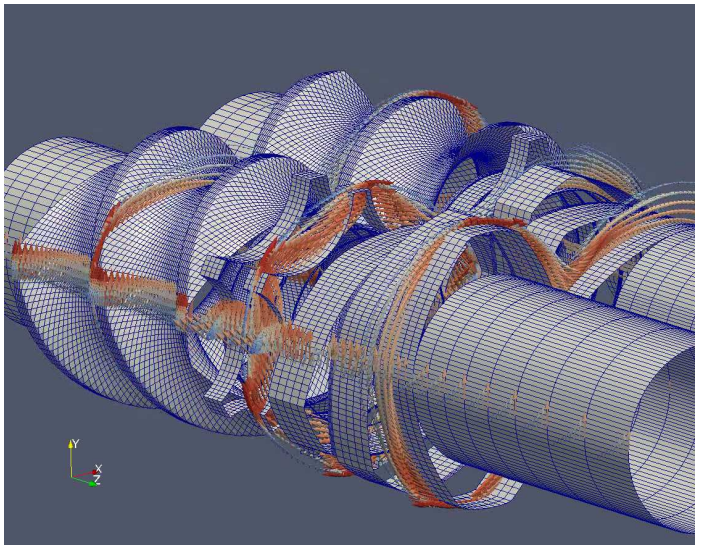
$$\frac{d}{dt}\int_V dV - \oint_S d\mathbf{s}\bullet\mathbf{u}_b = 0$$

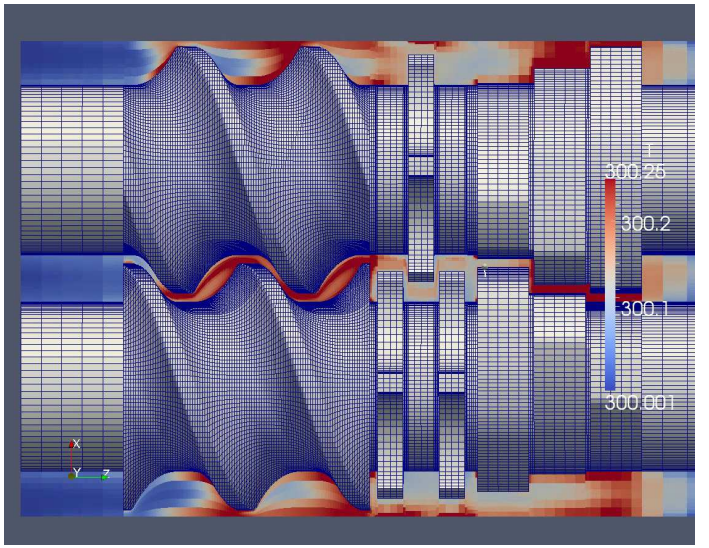
$$\oint_S d\mathbf{s}\bullet\mathbf{u}_b = \sum_f \int_{S_f} d\mathbf{s}\bullet\mathbf{u}_b = F_m$$

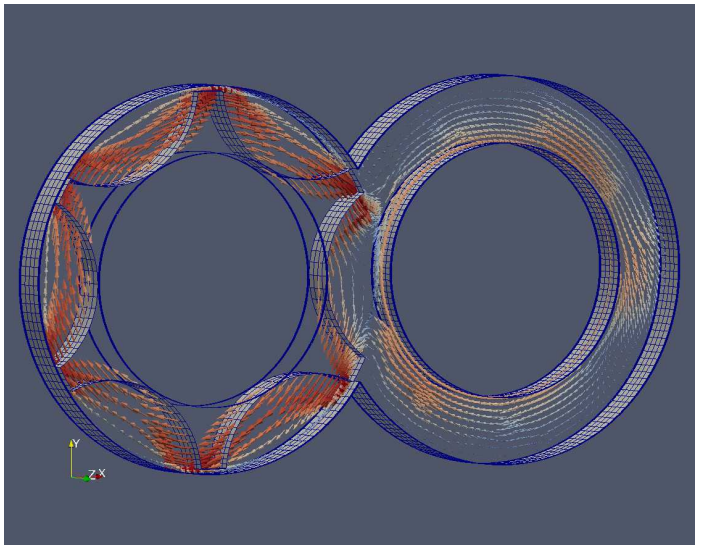
- Volume change appears in the rate-of-change term and is handled automatically
- Mesh motion flux appears in all convection terms and needs to be accounted for algorithmically
- Note: in incompressible flows, there are two possible formulations on the pressure equation, working either with relative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

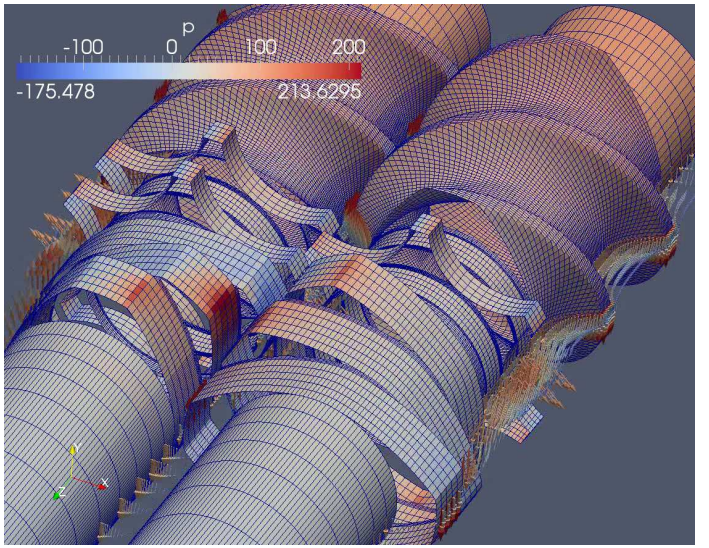
Complex Mesh Motion

Dynamic Mesh Examples of Complex Combination of Motion and Sliding



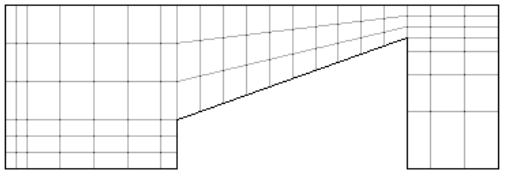


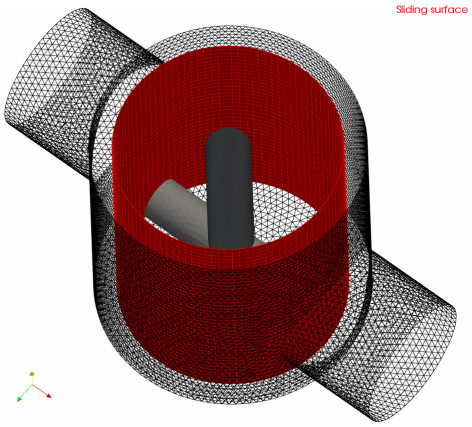


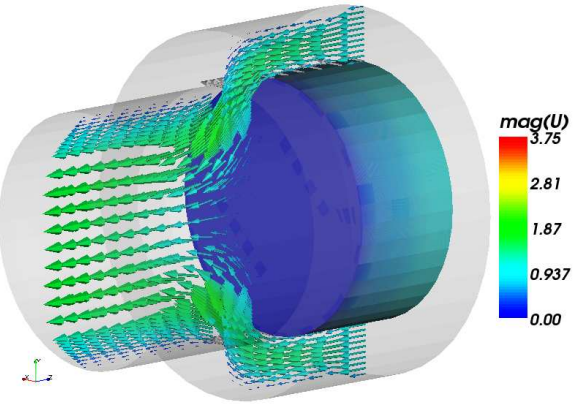


Topological Mesh Changes

Topological Changes on Polyhedral Meshes







Flow control device, sliding interface and mesh layering

- 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

Topological Mesh Changes

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

Topology Modifiers

“Set-and-Forget” Definition of Topology Modifiers

- `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
- `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

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

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

- Even for simple cases, it is easier to speak about problem classes (mixer vessels, engines, 6-DOF bodies) rather than working out individual topology modifiers

Dynamic Mesh Classes

Implementation of Dynamic Mesh Motion

- Definition or details of motion are irrelevant from top-level solver
- . . . but it makes sense to simplify mesh setup: a suite of problem- or algorithm-specific classes with a common interface
- Virtual base class: `dynamicFvMesh`

```
class dynamicFvMesh
:
    public fvMesh
{
    static autoPtr<dynamicFvMesh> New(const IOobject& io);

    // Destructor
    virtual ~dynamicFvMesh();

    //- Update the mesh for both mesh motion and topology change
    virtual bool update() = 0;
};

class topoChangerFvMesh : public dynamicFvMesh
{ ... };
```

Dynamic Mesh Classes

Dynamic Mesh with Automatic Mesh Motion

- `dynamicMotionSolverFvMesh`: dynamic mesh with motion solver

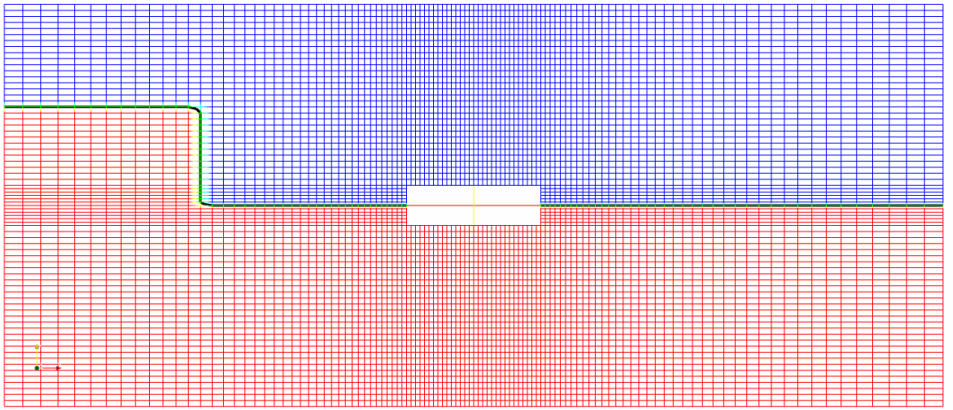
```
class dynamicMotionSolverFvMesh
:
    public dynamicFvMesh
{
    autoPtr<motionSolver> motionPtr_;

    virtual bool update()
    {
         fvMesh::movePoints(motionPtr_->newPoints());

        // Mesh motion only - return false
        return false;
    }
};
```

- Boundary motion is prescribed by setting a boundary condition on the motion equation: from top-level code, moving mesh class, boundary condition etc.
- Example: 6-DOF dynamic mesh = motion solver + 6-DOF solid body motion solver setting up its boundary condition

Dynamic Mesh: Floating Body

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

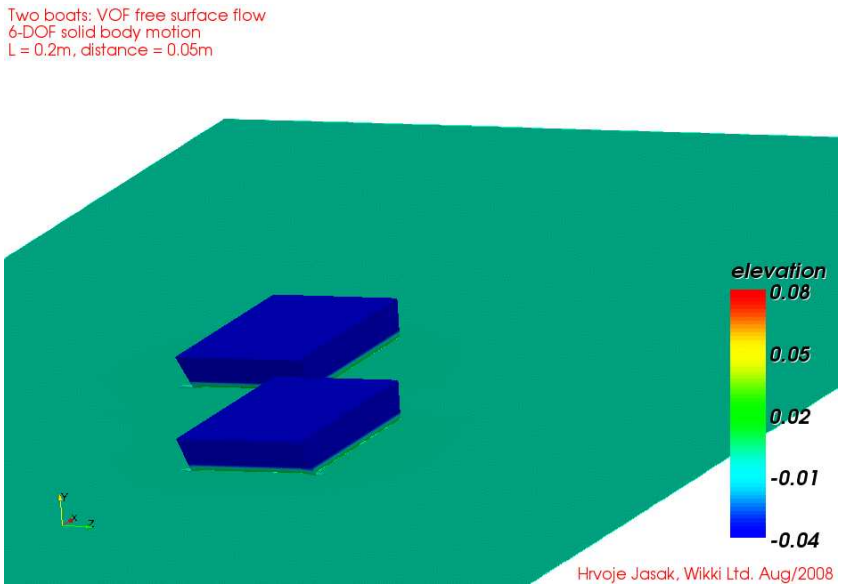
1. Specify mesh, material properties and initial + boundary flow conditions
2. Dynamic mesh type: `sixDofMotion`. Mesh holds `floatingBody` objects
3. A floating body holds 6-DOF parameters: mass, moment of inertia, support, forces
4. Flow solver only sees a `dynamicMesh`: encapsulated motion

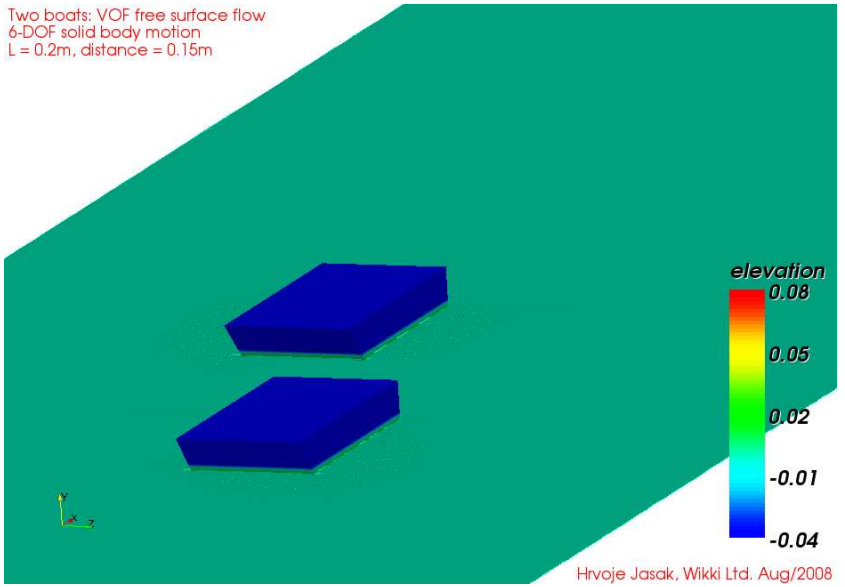
Floating Body Simulations

WIKKI

Multiple Floating Bodies

- Problem setup: as above, but with multiple bodies ⌣
- Example: simulation of two bodies in close proximity with different distance
- Elastic support for each boat in the x-direction with linear spring and damping; minor elastic support in the y-direction
- Automatic mesh motion shows its use: adding constrained components is trivial
- Extensive validation effort under way in collaboration with clients and University research groups

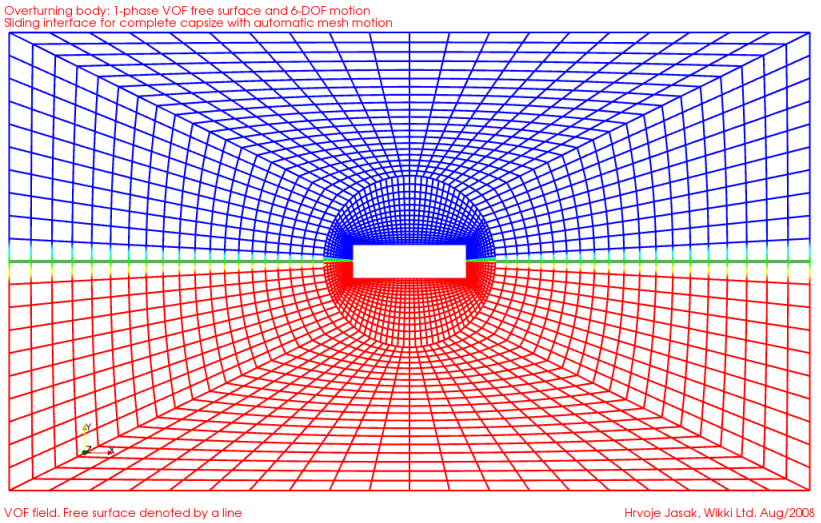


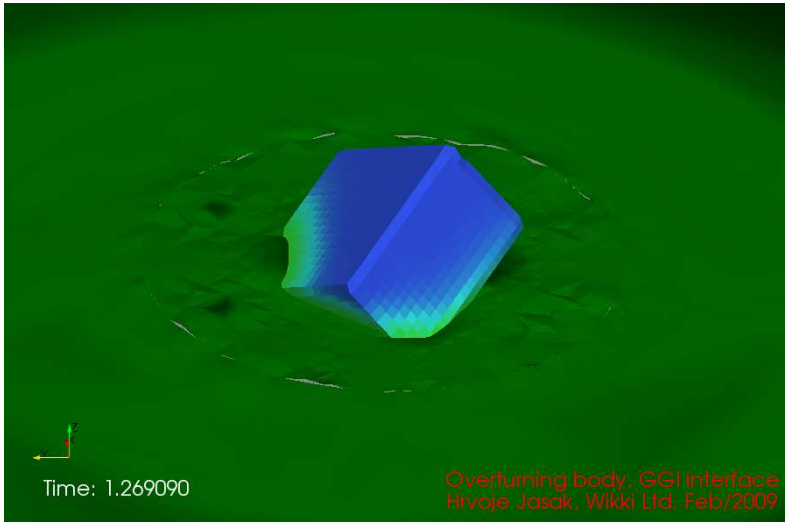


Floating Body Simulations

Capsizing Body with Topological Changes or GGI

- Full capsize of a floating body cannot be handled without topology change
- Mesh motion is decomposed into translational and rotational component
  - External mesh performs only translational motion
  - Rotation on capsize accommodated by a GGI interface
- Automatic motion solver handles the decomposition, based on 6-DOF solution
- Mesh inside of the sphere is preserved: boundary layer resolution
- Precise handling of GGI interface is essential: boundedness and mass conservation for the VOF variable must be preserved

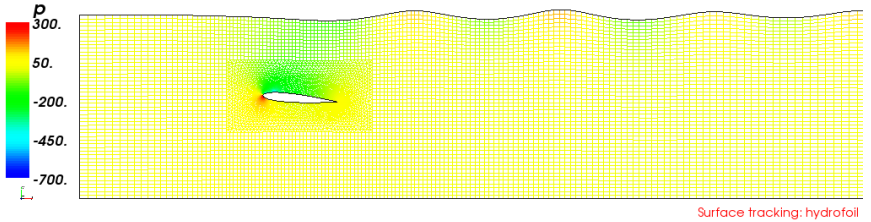




Automatic Motion: Surface Tracking

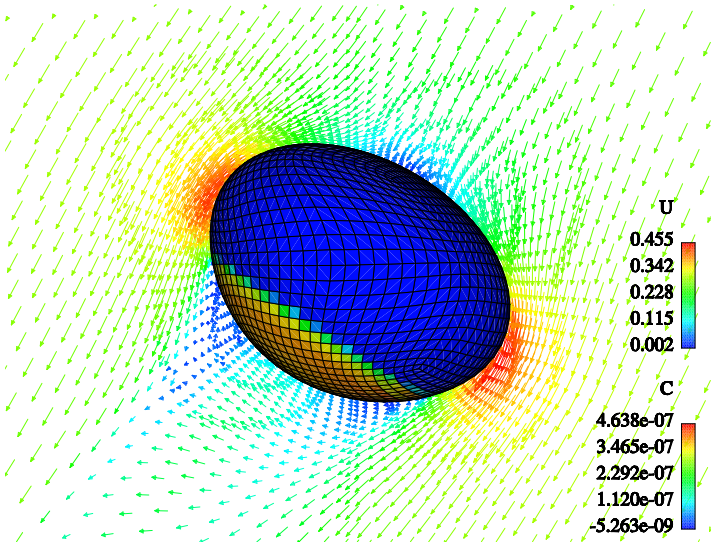
Hydrofoil Under A Free Surface

- Flow solver gives surface displacement
- Mesh adjusted to free surface position



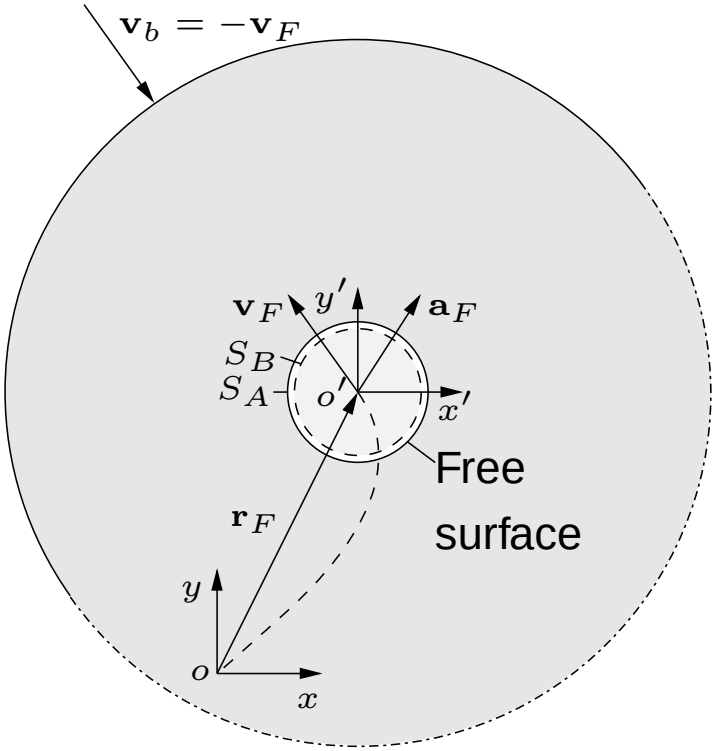
Free-Rising Air Bubble with Surfactants

- Two meshes coupled on free surface



Single Solver, Complex Coupling

- FVM on moving meshes: segregated $p - \mathbf{u}$ solver
- Automatic mesh motion
- Finite Area Method (FAM)



Dynamic Mesh: IC Engine

Example: Multi-Valve Engine Mesh

- For complex topological changes, multiple interacting topology modifiers are used and need to be synchronised and used in unison with mesh motion
- Case setup requires the mesh class to adhere to the “language of the problem”
- Example: Engine valve
  - Definition identifies valve stem, top and bottom surface in geometry
  - Topology modifiers: layer addition-removal on top and bottom surface; sliding interface along valve curtain
  - Valve motion defined in terms of valve lift curves and crank angle degree
  - **Result:** topology modifiers are added automatically for each valve
- Engine mesh components
  - Piston class, with motion defined in terms of crank angle degree and cell layering thickness
  - List of valves, each with own lift curve
  - Identification of intake and exhaust ducts, piston bowl etc.
- The user builds the mesh and associates various surfaces to engine components: easy setup after initial static mesh generation

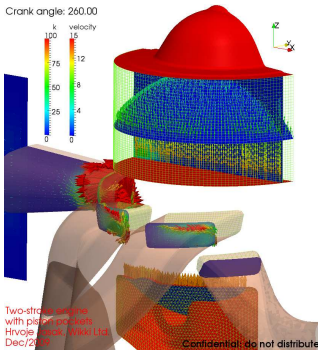
More Dynamic Mesh Examples

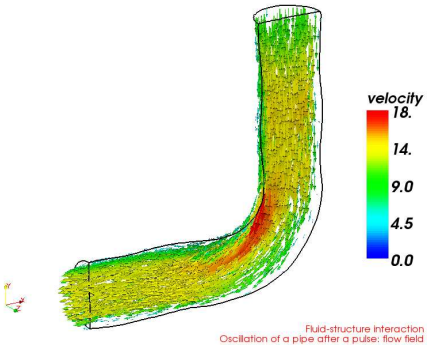
Example: In-Cylinder Flow with Moving Piston and Valves

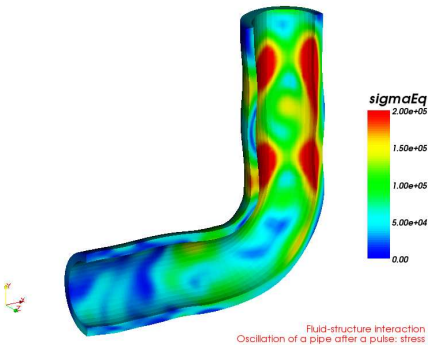
- Exhaust and intake stroke in a 2- and 4-stroke engine
- Moving piston and operating valves using topological changes
- Interacting topological modifiers and mesh motion handled by the mesh class
- Politecnico di Milano: combining mesh deformation, topological changes and re-meshing to achieve optimum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



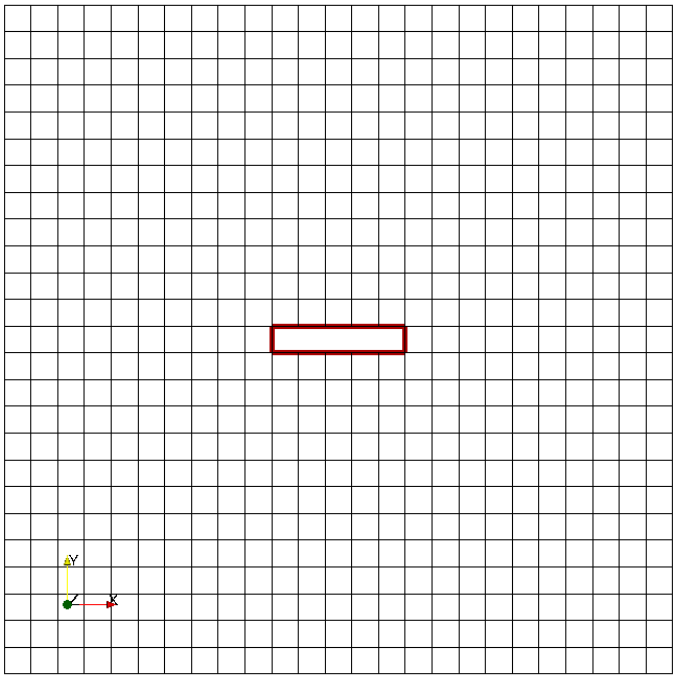




New Interpolation Tools

Radial Basis Function Interpolation

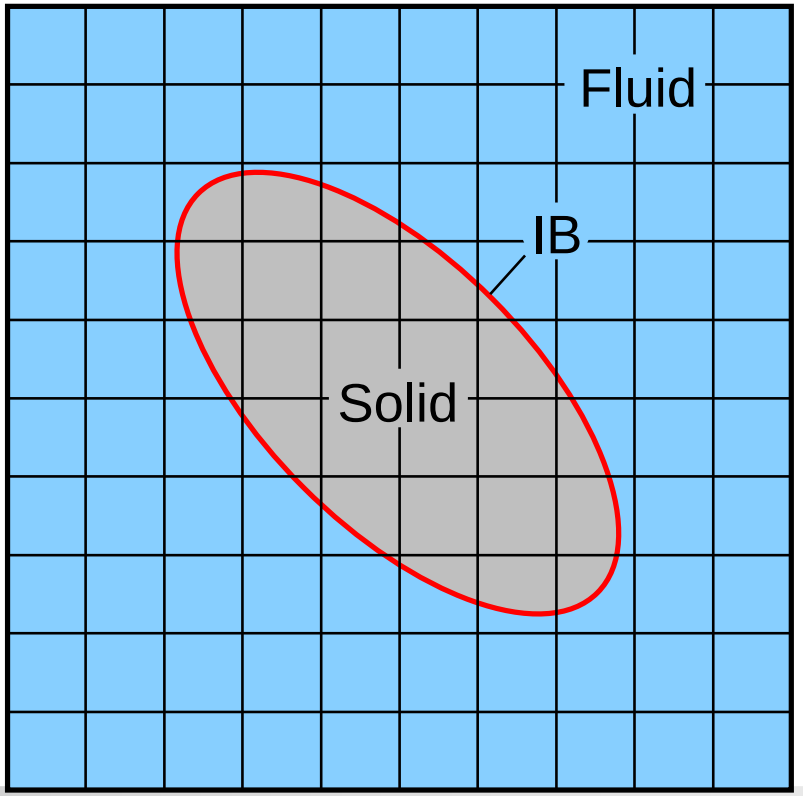
- General interpolation for clouds of points
- Mathematical tool which allows data interpolation from a small set of control points to space **with smoothness criteria**
- Used for mesh motion in cases of large deformation: no cross-over
- Implemented by Frank Bos, TU Delft and Dubravko Matijašević,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



Immersed Boundary Methods (IBM)

Immersed Boundary Method: Non-Conformal Boundary Surfaces

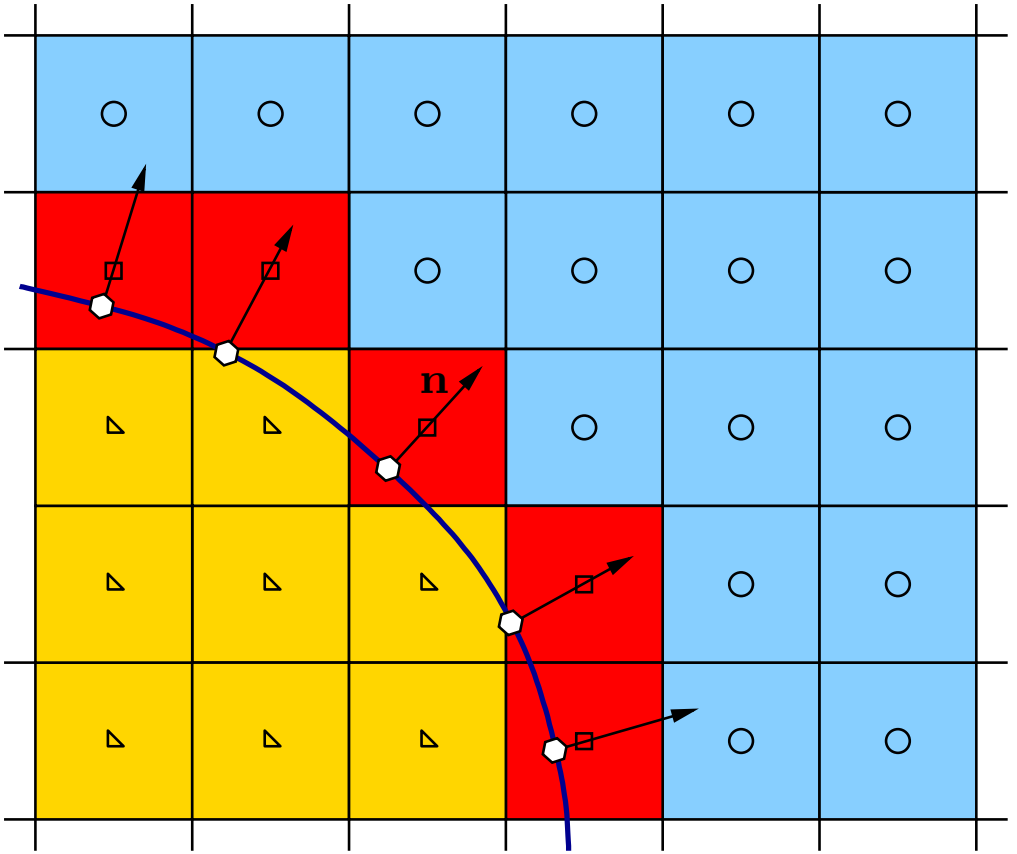
- Simulation of the flow around immersed boundary is carried out on a grid (usually Cartesian) which does not conform to the boundary shape
- Immersed boundary (IB) is represented by surface grid
- IB boundary conditions modify the equations in cells which interact with the immersed boundary



Immersed Boundary Methods (IBM)

Implementation of IBM In OpenFOAM

- Discrete forcing approach with direct imposition of boundary conditions
- Basic principle: Value of dependent variable in the IB cell centres is calculated by interpolation using neighbouring cells values and boundary condition at the corresponding IB point

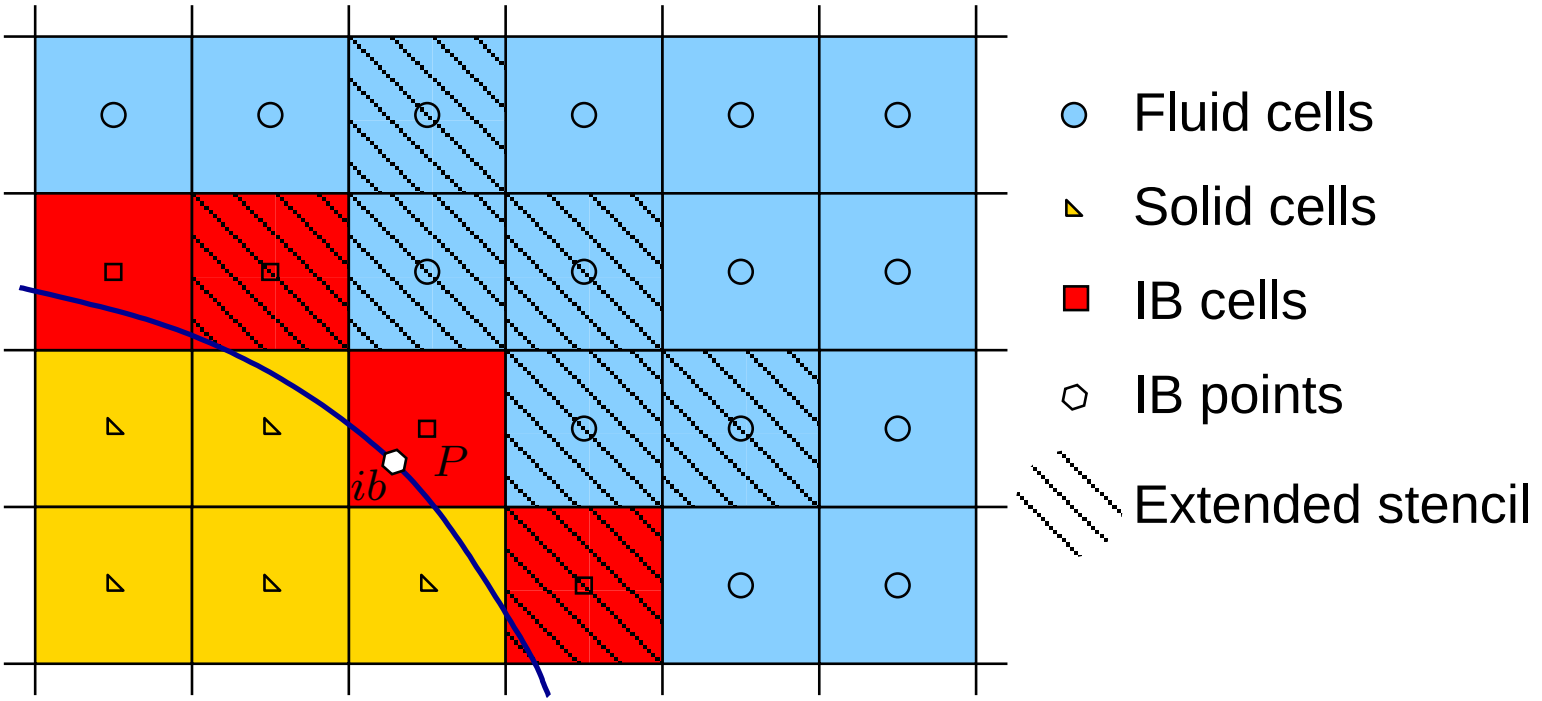


Immersed Boundary Methods (IBM)

Immersed Boundary Condition via Quadratic Interpolation

$$\phi_P = \phi_{ib} + C_0(x_P - x_{ib}) + C_1(y_P - y_{ib})$$
$$+ C_2(x_P - x_{ib})(y_P - y_{ib}) + C_3(x_P - x_{ib})^2 + C_4(y_P - y_{ib})^2$$

Unknown coefficients of quadratic polynomial determined using weighted least square method on extended stencil.



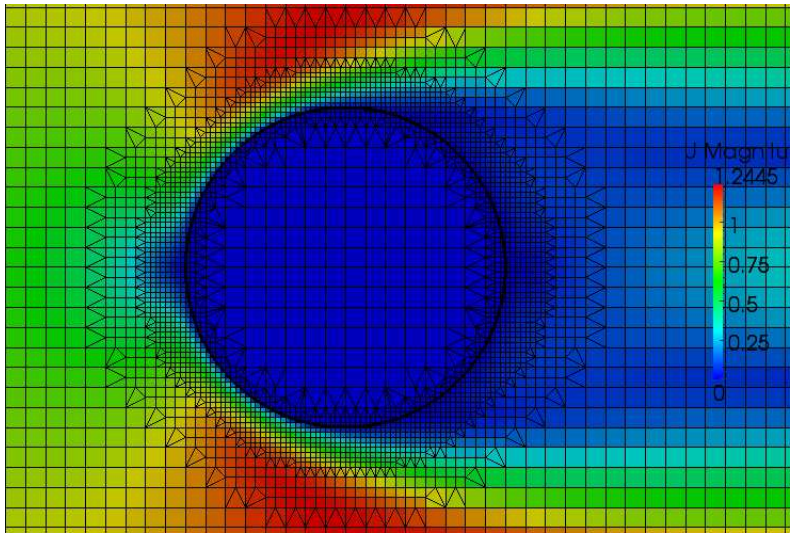
Immersed Boundary Methods (IBM)

Immersed Boundary Implementation in Three Classes:

- `class immersedBoundaryPolyPatch`: Basic mesh support, IB mesh
- `class immersedBoundaryFvPatch`: FV support, with derived Fv properties
  - Cell and face mask fields, live and dead cells indication
  - Calculation of intersection points, normals and distances
  - Calculation of interpolation matrices used in imposition of boundary conditions
  - Parallel communications framework and layout
- `class immersedBoundaryFvPatchField`: field support and evaluation of boundary conditions
  - Patch field evaluation for the IB patch
  - Calculation and interpolation of field data at mesh intersection, fixed value and fixed gradient conditions etc.
  - Handling of boundary updates

Immersed Boundary Methods (IBM)

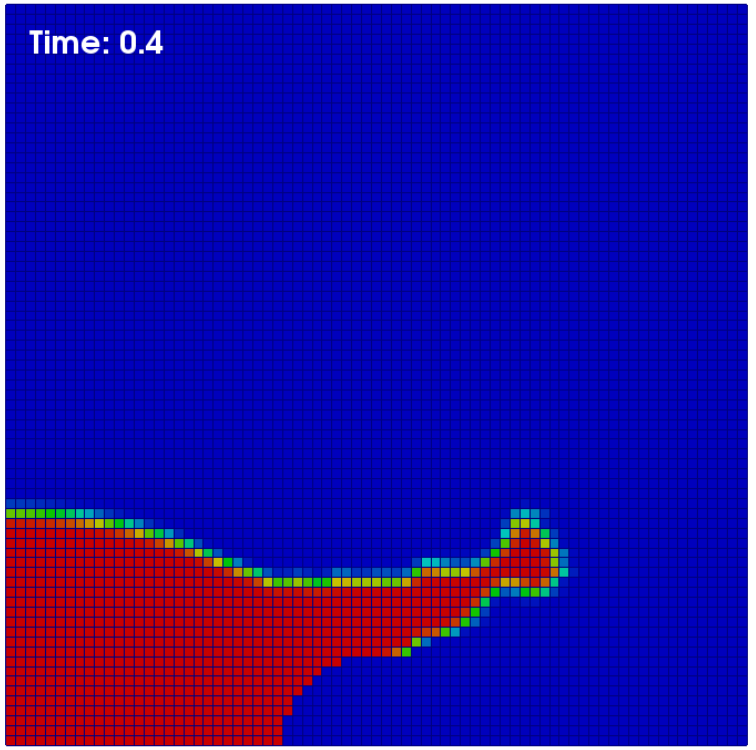
Example Case:: `flowOverCylinder`



- Laminar flow around a circular cylinder in open space
- Case setup data
  - Open space dimensions: 90×90 m
  - Inlet velocity: 1 m/s
  - Cylinder diameter: 1 m
  - Reynolds number: $100 - 10\,000$

- Automatic surface proximity refinement of background mesh
- Single phase laminar/turbulent steady or transient flow
- Wall function treatment on immersed boundary is available for high Re flows!

Immersed Boundary Methods (IBM)



- Dam break VOF interface capturing simulation with circular bump at the bottom boundary
- Case setup data:
  - Domain dimension: 2×2 m
  - Bump diameter: 0.5 m
  - Water-air multi-phase system

- Most of OpenFOAM library models (eg. material properties, turbulence, combustion) are used without change
- Due to explicit flux manipulation, top-level solvers still require minor changes: **code conversion cook-book** is provided with the library
- In specialist applications, eg. Lagrangian particle tracking or surface-to-surface radiation, large changes are required

Overset Mesh in OpenFOAM

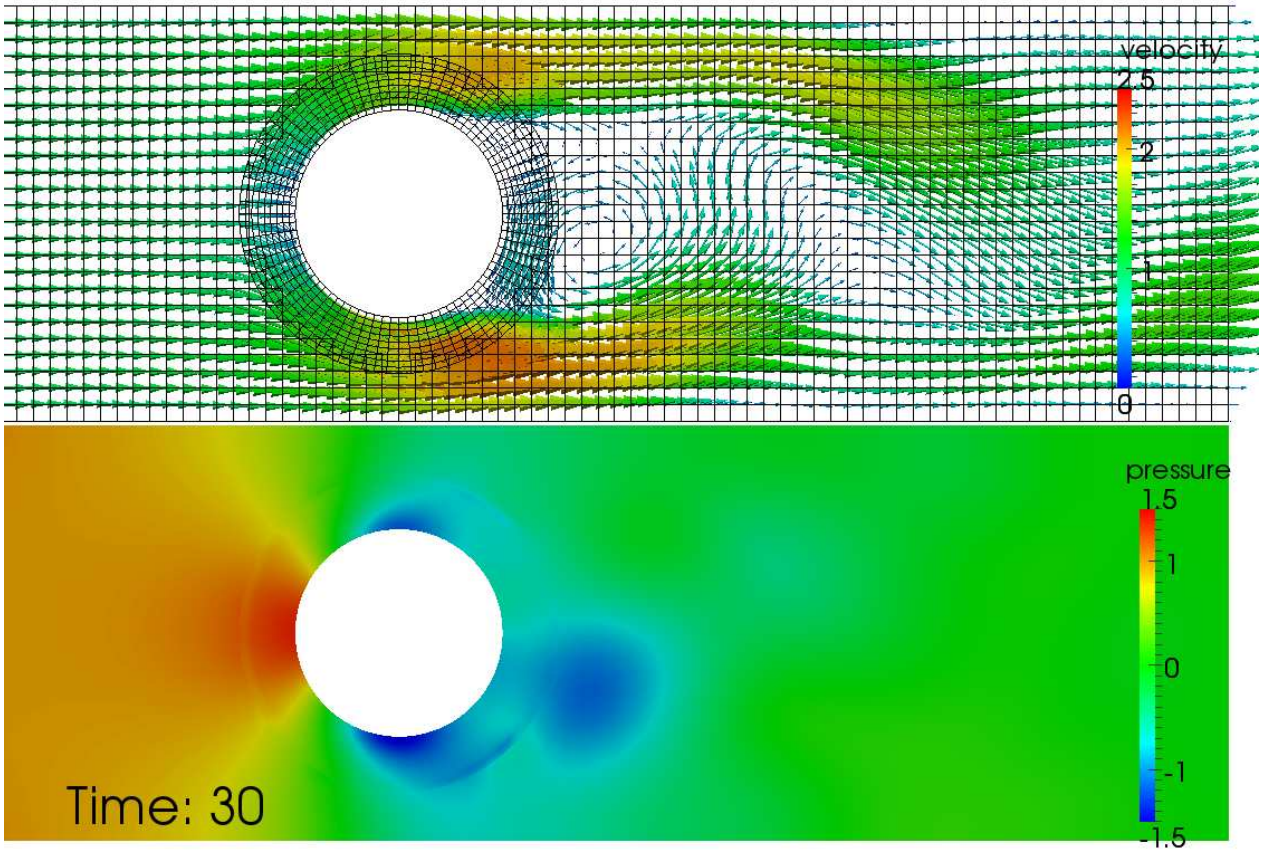
Native Overset Mesh Implementation

- Previous implementation based on SUGGAR and DirtLib is too restrictive and insufficiently flexible: modification of all models, use of external linear algebra package, ITAR restrictions on library components
- **Objective**: native open source, parallel bottom-up implementation of Overset Mesh with minimal impact on existing OpenFOAM capabilities
- Design layout matches the Immersed Boundary Method
  - Overset boundary patch to handle geometrical support
  - For multiple overset patches, the interpolation information must be shared: `oversetMesh` object
  - Overset boundary patch field to handle field updates: like immersed boundary patch, but with implicit updates
- Overset mesh specific tools
  - Overset mesh hierarchy interpolation tools
  - Selection of donor and acceptor cells
  - Parallelisation support for inter-processor interpolation and fringe search

Overset Mesh in OpenFOAM

Native Overset Mesh Implementation

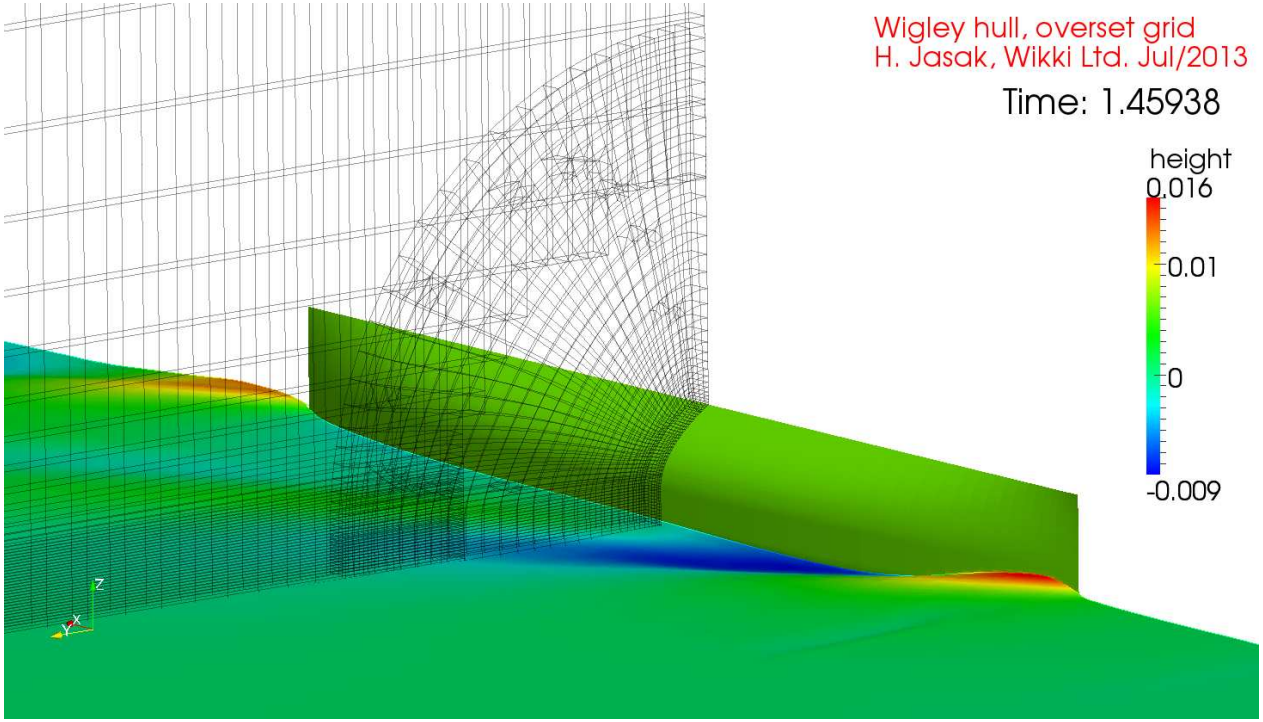
- Example: two-mesh overset with inactive cells blanked out
- Transient laminar flow around a cylinder in a shedding mode



Overset Mesh in OpenFOAM

Overset Mesh in Naval Hydrodynamics

- Free surface flow solver from Wikki Naval Hydro pack used for steady resistance simulation around a Wigley hull geometry



Summary

Summary

- OpenFOAM implements polyhedral mesh handling in library form
- Mesh analysis classes separated from discretisation support classes
- Built-in support for mesh motion and topological changes
- Simple handling of moving boundary problems: automatic mesh motion solver
- Topological changes support
  - Basic operations: add/modify/remove point/face/cell
  - Mesh modifier classes operate in terms of basic operations. Triggering of topological changes is a part of the class (automatic)
  - Pre-defined mesh modifiers available for standard operations
- Dynamic mesh classes encapsulate topology modifiers and (automatic) mesh motion for easy handling of classes of dynamic mesh problems
- Top-level physics solver is separated from dynamic mesh handling for inter-operability of components. Example: 6-DOF moving body in flow
- Quest for a fully automatic polyhedral mesh generator continues: `snappyHexMesh`, `cfMesh`