Dancing Meshes: Past and Present

Dynamic Mesh Support in OpenFOAM

Hrvoje Jasak

Wikki Ltd, United Kingdom, Germany and Brazil

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

TOBB ETU, Ankara, 23 October 2019

Outline

Objective

- Present my work on dynamic mesh support in OpenFOAM, 1994 to present

Topics

1. Introduction
2. `polyMesh`: Polyhedral mesh support
3. Mesh conversion and manipulation
4. Deforming meshes
5. Topological change support
6. Complex dynamic mesh simulations
7. Native overset mesh
8. Immersed Boundary Surface
9. Summary

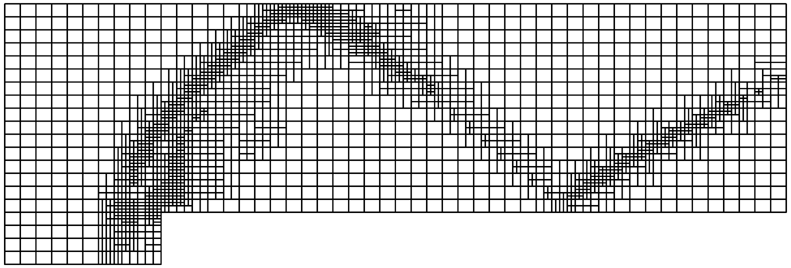
Introduction

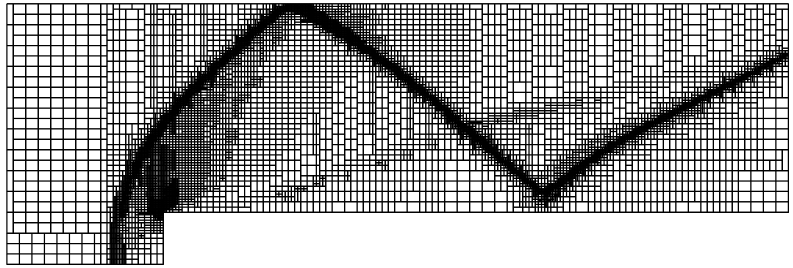
Background: Early Days

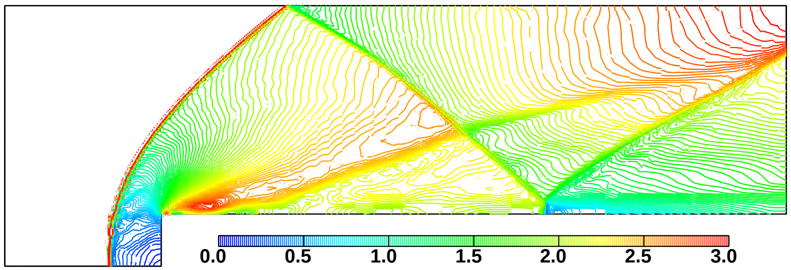
- Ideas on the solver structure, programming language, equation mimicking and discretisation / linear solver looks were established from the onset
- . . . but mesh support in early version was “very traditional”
- **World-class solver requires ultimate meshing flexibility**
- and we did not even have a basic mesh generator
- This was the starting point for my work in 1993:
  PhD on adaptive mesh refinement

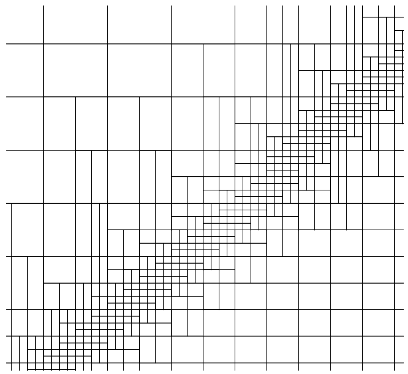
Adaptive Mesh Refinement

Mesh Adaptivity on Shocked Flows

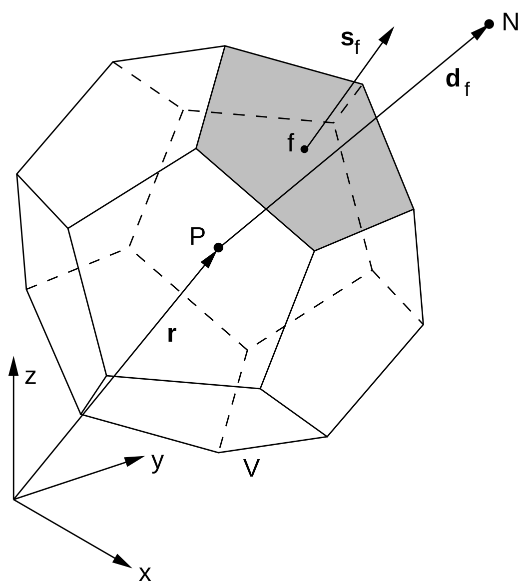








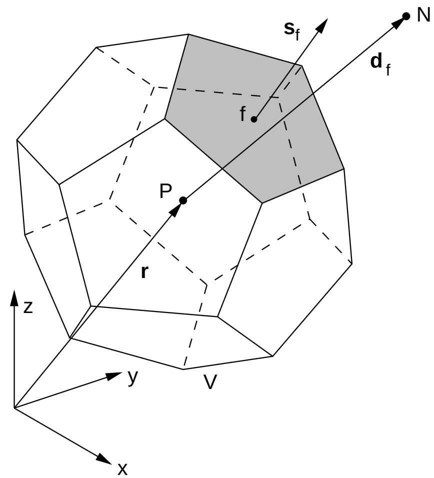
Polyhedral Mesh Support

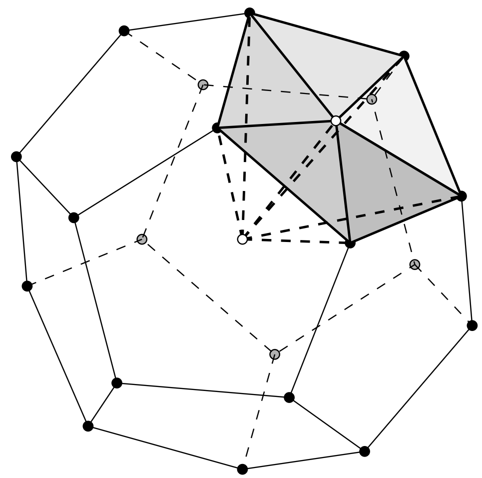


Polyhedral Mesh Support

Flexibility in Meshing: Polyhedral Cells

- Historically, CFD meshes use **shape-based support**: hexahedron, pyramid, prism, wedge, tetrahedron etc, defined in terms of vertices
- ... but the FOAM solver is written using face addressing
- Objective: rewrite mesh classes using **polyhedral mesh**
  - Points list: `(x y z)` coordinates
  - Polygonal face: ordered list of point labels
  - Polyhedral cell: list of face labels: changed to owner/neighbour addressing
  - Boundary patches with slicing of face list
- Mesh metrics calculation using polyhedral decomposition into pyramids/tets





Polyhedral Mesh Support

Rationale

- A polyhedron is a generic form covering all cell types: consistency in discretisation across the board
- Finite Volume Method (FVM) naturally applies to polyhedral cells: cell shape is irrelevant (unlike FEM)
- Mesh generation is still a bottleneck: polyhedral support simplifies the problem
- New mesh checking and consistency checks need to be developed and implemented

Consequences: What Have We Done?

- All algorithms must be fully unstructured. Structured mesh implementation possible where desired (*e.g.* aero-acoustics) but implies separate mesh classes and work on discretisation code re-use
- In 1990s, fully unstructured FVM was a challenge: now resolved
- No problems with imported mesh formats: polyhedral cell covers it all!
- Issues with “old-fashioned software” compatibility with no polyhedral support, *e.g.* post-processors. **On-the-fly cell decomposition**

Mesh Conversion and Manipulation

Basic Mesh Generation and Conversion

- Basic mesh generation tool: `blockMesh`
  - Block-structured mesher with curved edges and flexible grading

Mesh Converters

- `starToFoam`, `sammToFoam`
- `fluentMeshToFoam`
- `gambitToFoam`
- `cfx4ToFoam`
- `ideasUnvToFoam`
- `ansysToFoam`

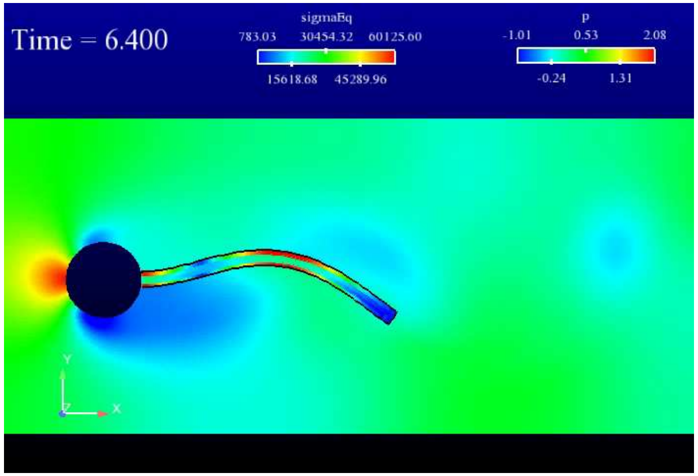
And Reverse Converters

- `foamToStarMesh`
- `foamMeshToFluent`
- `foamDataToFluent`
- `foamMeshToAbaqus`

Mesh Manipulation Tools

- `transformPoints`
- `mergeMeshes`
- `mirrorMesh`
- `subsetMesh`
- `zipUpMesh`
- `checkMesh`

Moving Mesh Simulations



Moving Mesh Simulations

Moving Mesh Simulations

- Definition of a moving mesh problem: the number of points, faces and cells in the mesh and their connectivity remains the same but the point position changes
- Sufficient for most cases where shape of domain changes in time
- FVM naturally extends to moving meshes: need to re-calculate cell volume and area swept by a face in motion
- Moving mesh support built into mesh classes and discretisation operators
- In some places, algorithmic changes required in top-level solver code
- Problem: how to specify point motion for every point in every time-step?

Moving Mesh Simulations

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

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

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

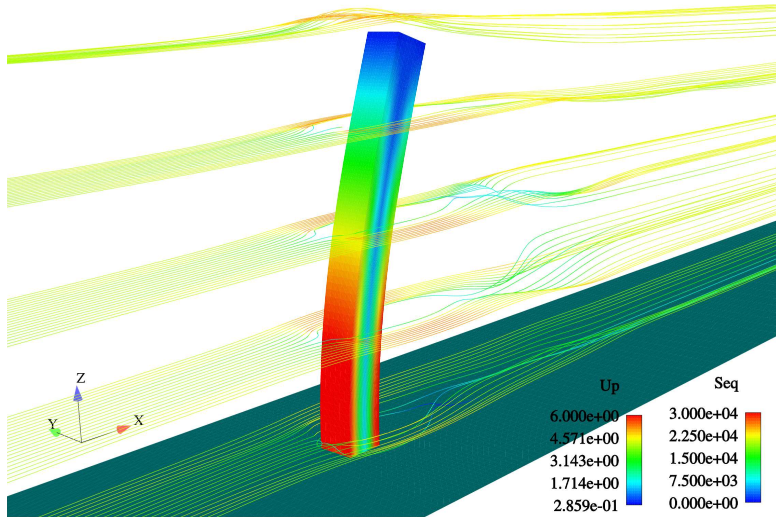
$$\oint_S d\mathbf{s}\bullet\mathbf{u}_b = \sum_f \int_{S_f} d\mathbf{s}\bullet\mathbf{u}_b = F_m$$

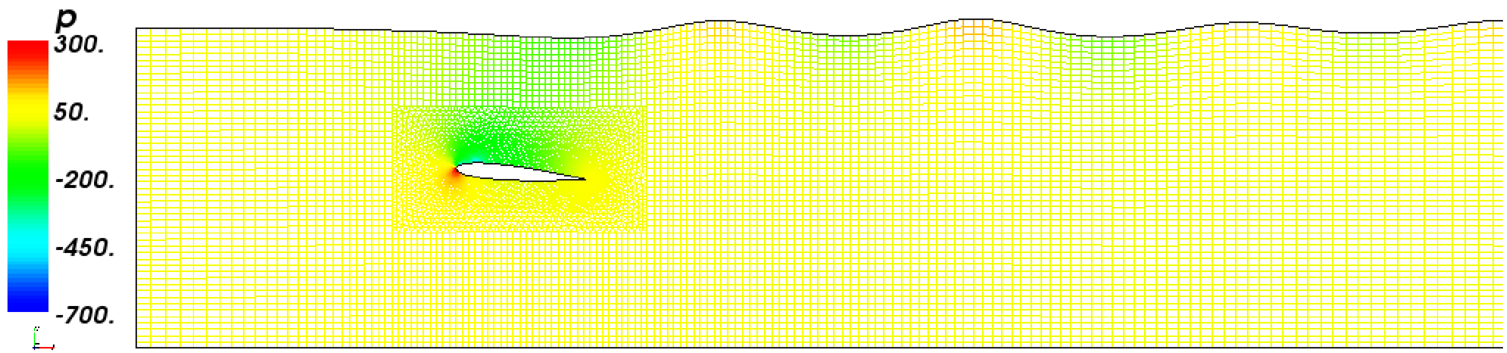
- Volume change appears in the rate-of-change term and is handled automatically
- Mesh motion flux appears in all convection terms and needs to be accounted for algorithmically
- Note: in incompressible flows, there are two possible formulations on the pressure equation, working either with relative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

Automatic Mesh Motion

Automatic Mesh 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, only **motion of the boundary** is known or calculated
- Automatic mesh motion determines the position of internal points based on boundary motion





Surface tracking: hydrofoil

Automatic Mesh Motion

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
- 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: `fvm` or `tetFem`
  - Laplace equation with constant and variable diffusivity

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

  - Linear pseudo-solid equation for small deformations

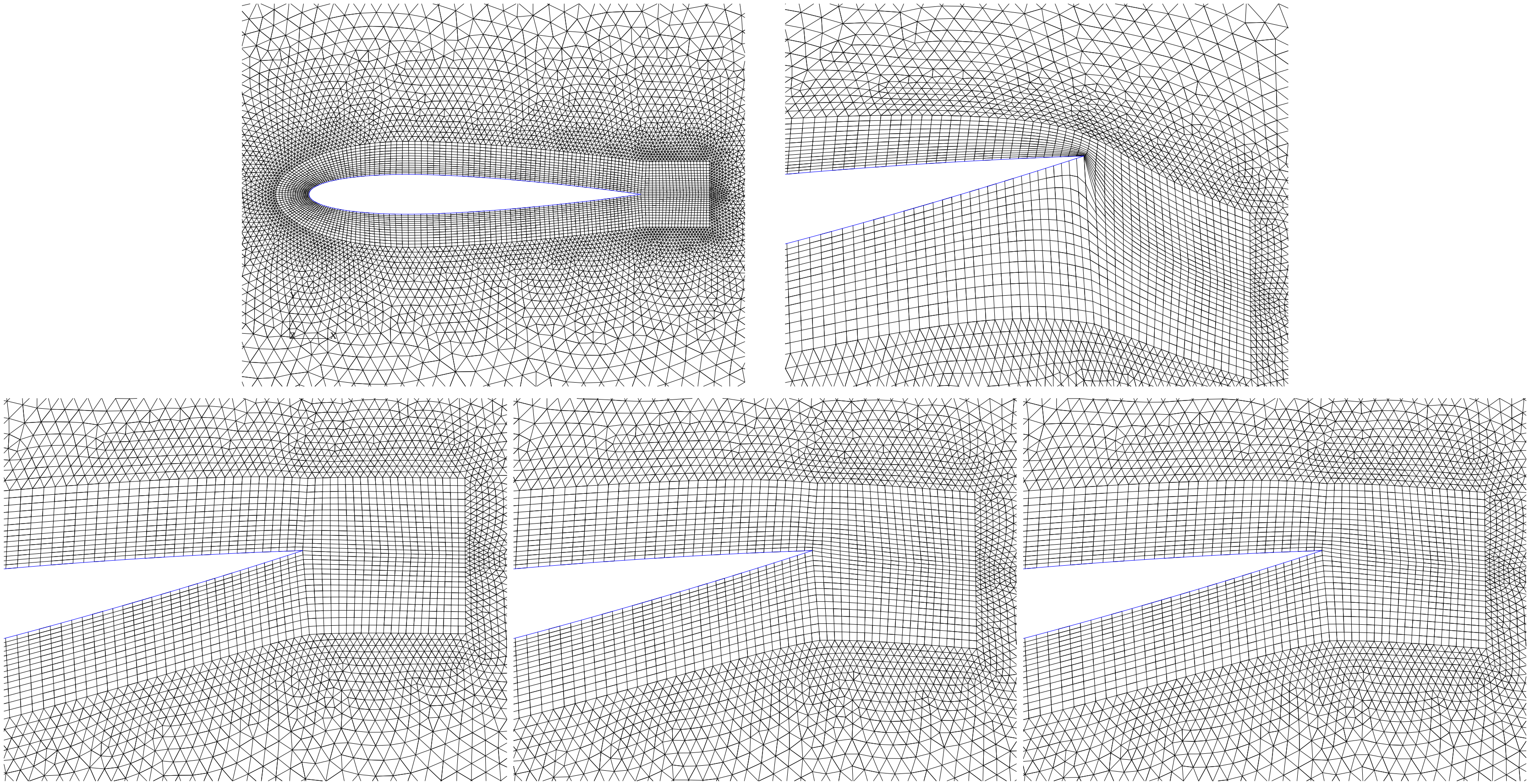
$$\nabla \bullet [\mu (\nabla \mathbf{u} + (\nabla \mathbf{u})^T) + \lambda \mathbf{I} \, \nabla \bullet \mathbf{u}] = 0$$

- Mesh spacing and quality control by **variable diffusivity** (Tuković, 2005)
- Changing diffusivity re-distributes the boundary motion through the volume

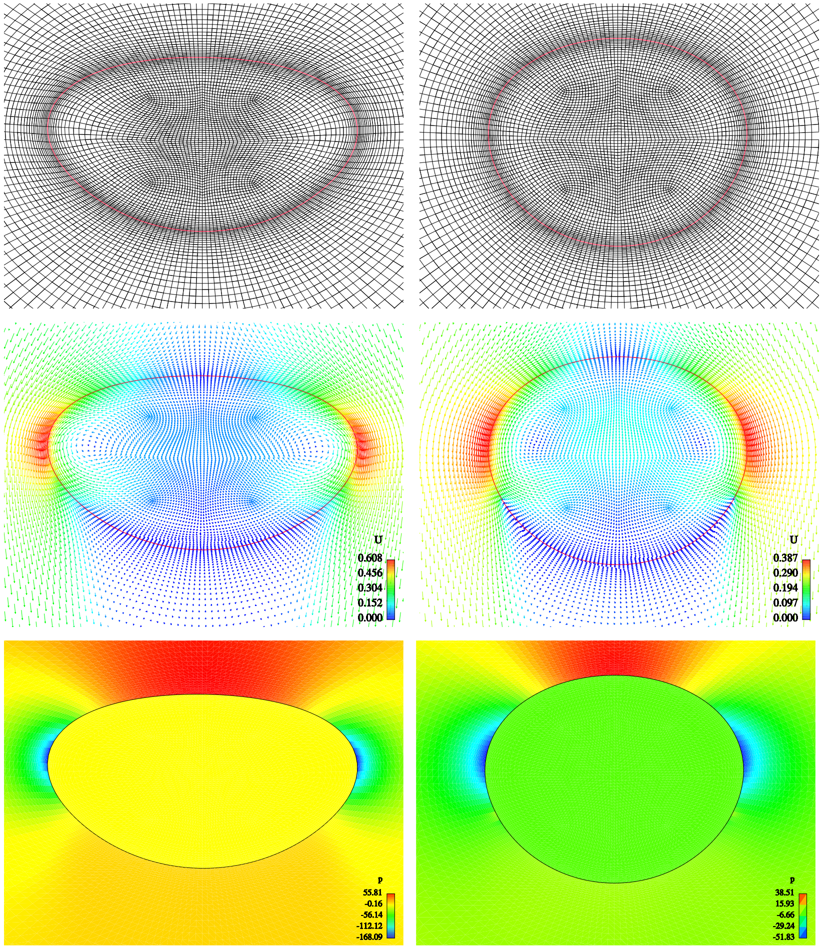
Automatic Mesh Motion

Effect of Variable Diffusivity: Oscillating Airfoil Simulation

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



DNS of Rising Bubbles



Multi-Phase Free Surface Tracking

- Two meshes coupled on free surface: perfect capturing of the interface and curvature evaluation
- Coupling conditions on the interface include stress continuity and surface tension pressure jump

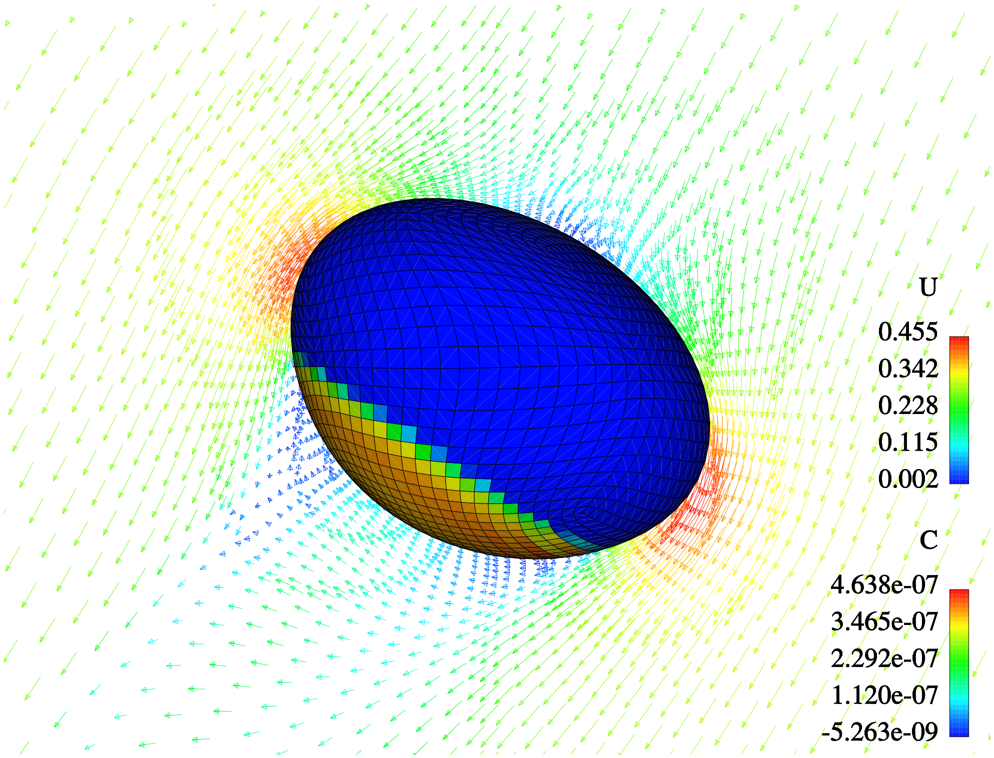
Free Rising Air Bubbles

- Simulation particularly sensitive on accurate handling of surface curvature and surface tension
- Full density and viscosity ratio
- Locally varying surface tension coefficient as a function of surfactant concentration
- Coupling to volumetric surfactant transport: boundary conditions

DNS of Rising Bubbles

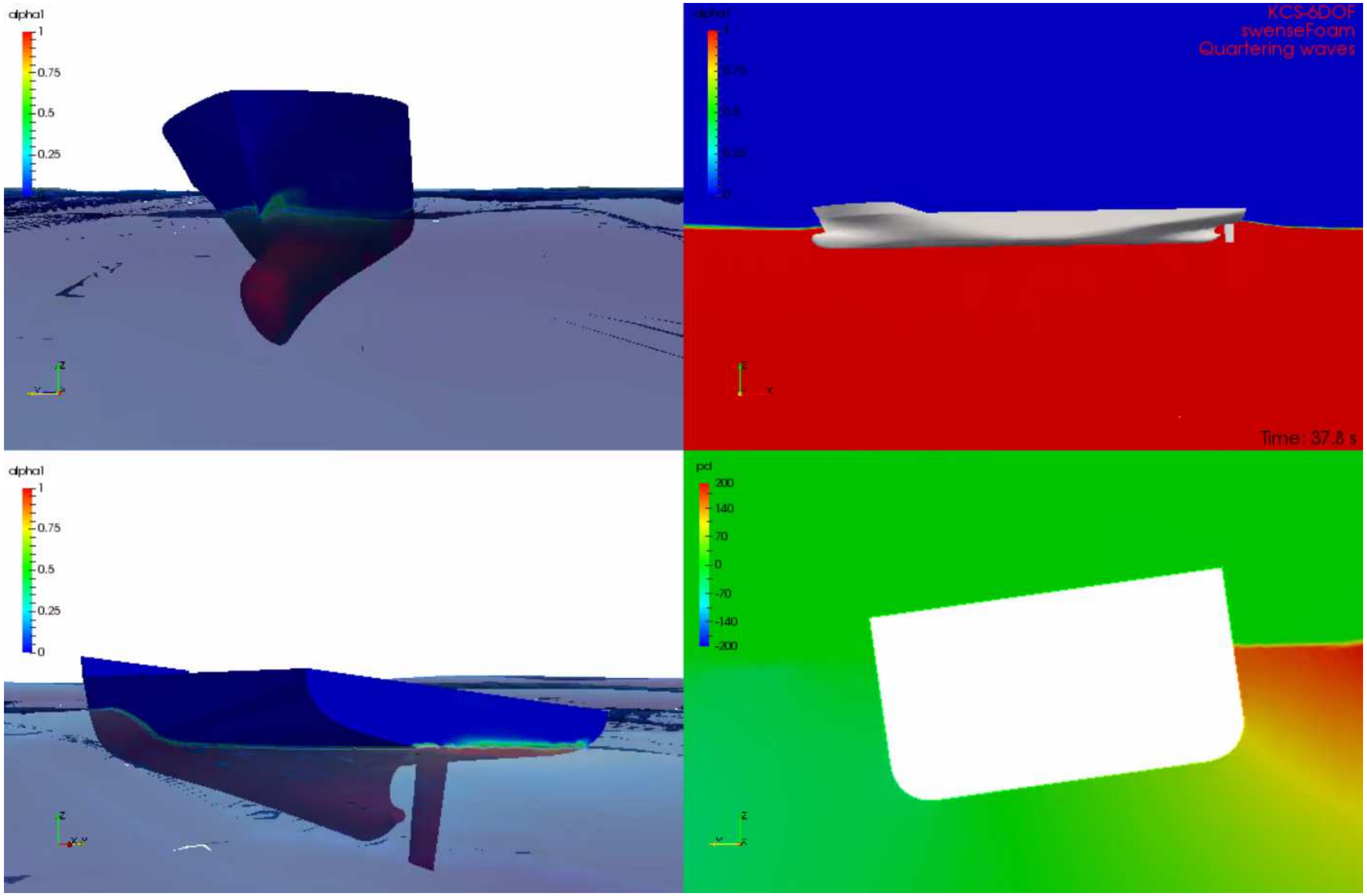
Complex Coupling in a Single Solver: 3-D Rising Bubble: Željko Tuković PhD, 2005

- FVM flow solver: incompressible $p - \mathbf{u}$ coupling
- FEM automatic mesh motion: variable diffusivity Laplacian
- FAM for surfactant transport: convection-diffusion on surface, coupled to 3-D
- Non-inertial frame of reference, attached to bubble centroid



Naval Hydrodynamics

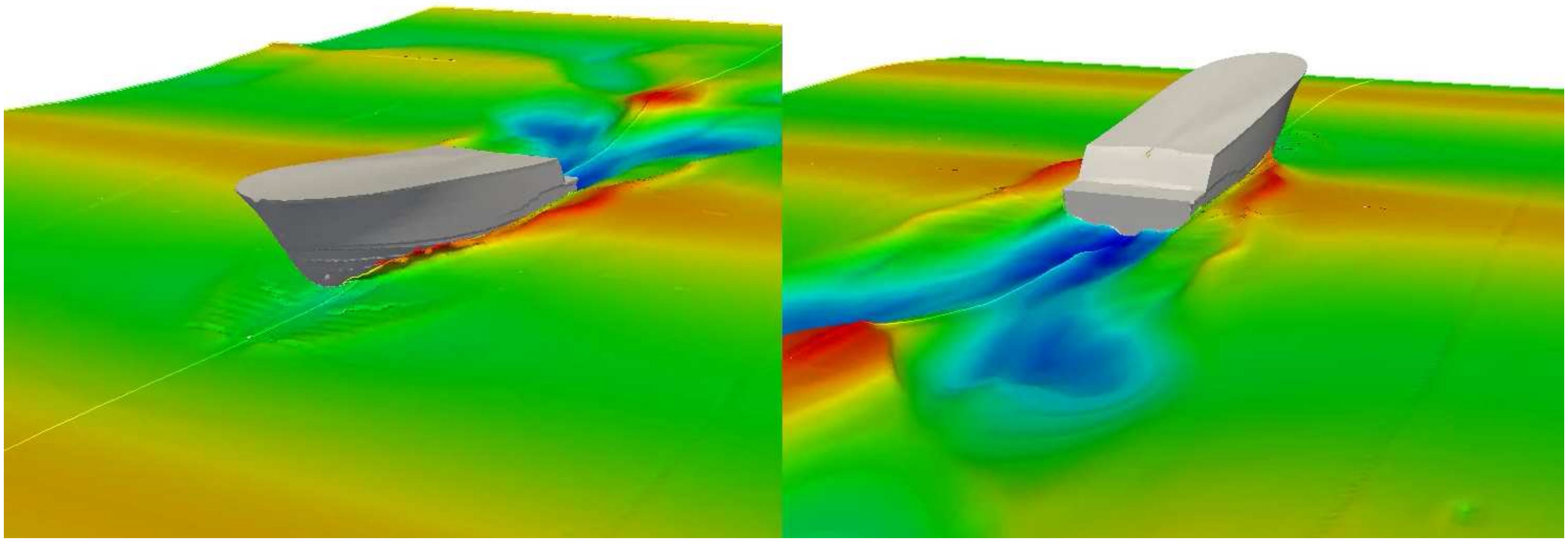
Tokyo 2015 Code Certification Workshop for Naval Hydrodynamics CFD



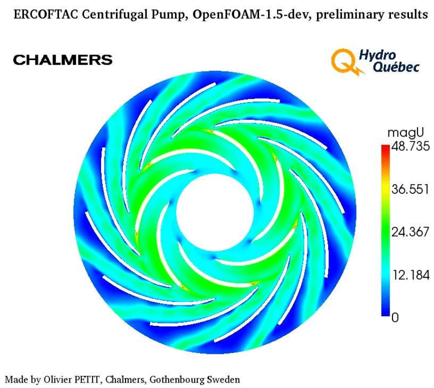
Naval Hydrodynamics

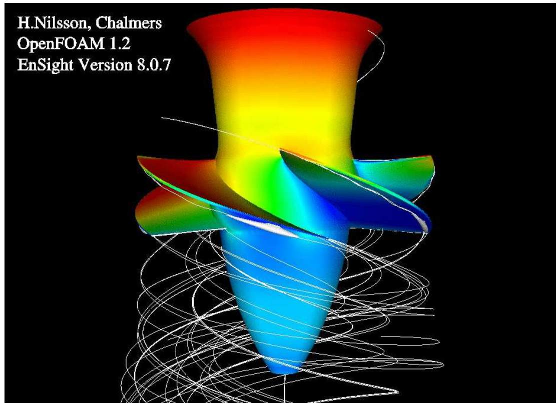
SOPHYA Project: Sea-Keeping for Fast Hulls

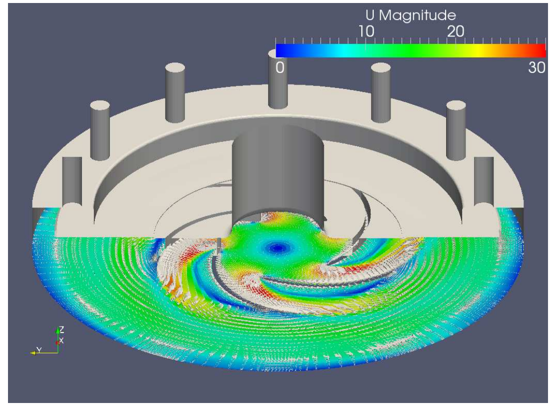
- Modelling, towing tank experiments and **full-scale sea trials** for a fast hull in calm water and in waves
- Combination of model-scale and full-scale CFD simulations
- Collaboration with Uni Trieste and Monte Carlo Yachts

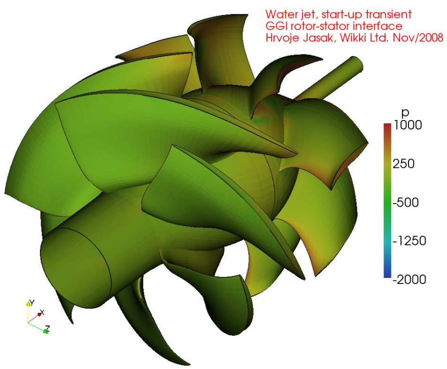


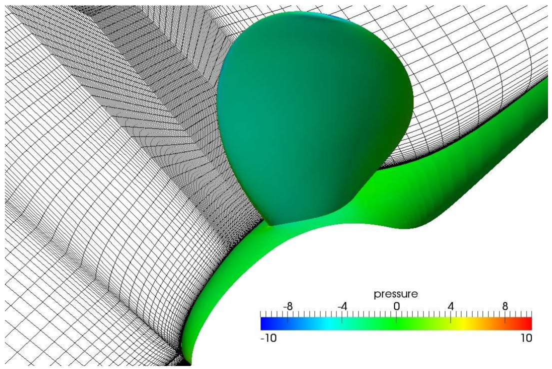
Turbomachinery

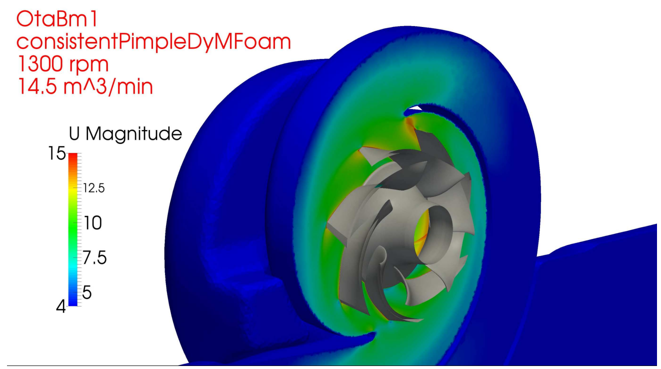








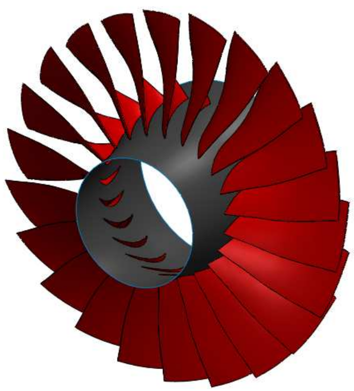


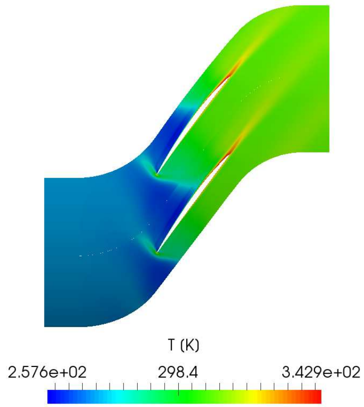


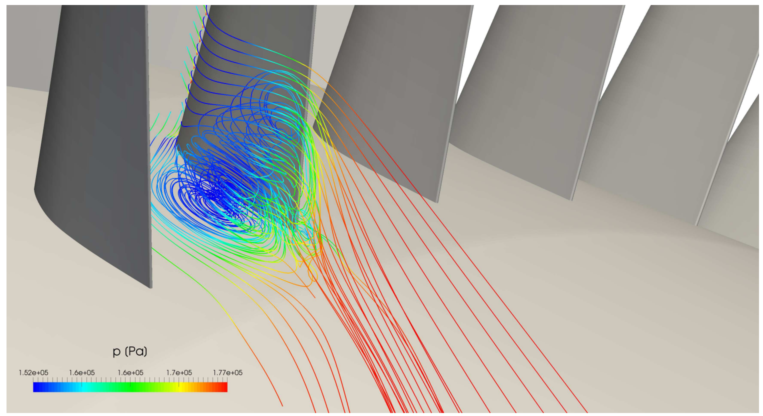
Turbomachinery

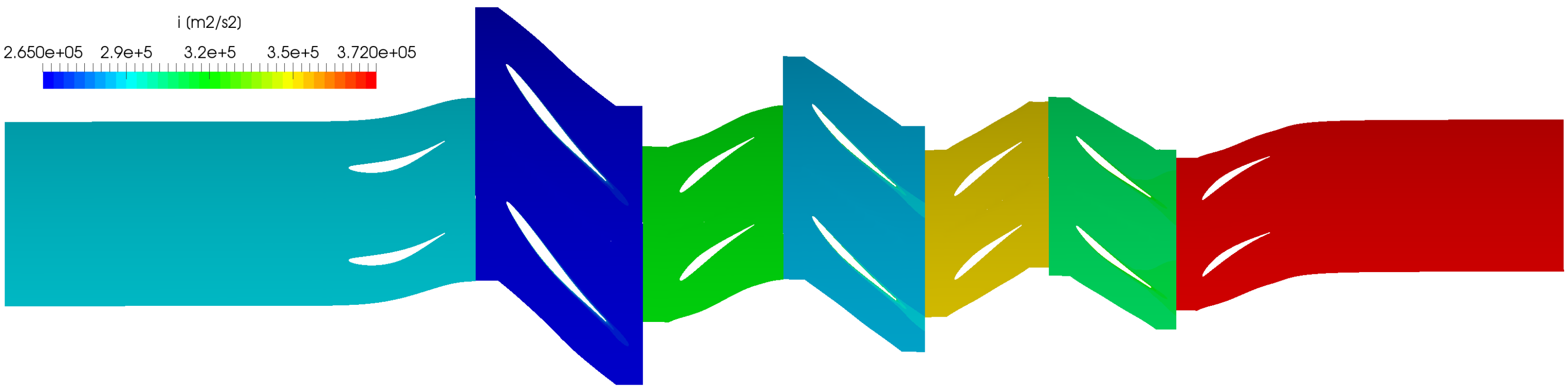
Detailed Validation of Compressible Turbomachinery Solvers

- Rothalpy formulation and rothalpy jump conditions at rotor-stator interfaces
- Compressible harmonic balance

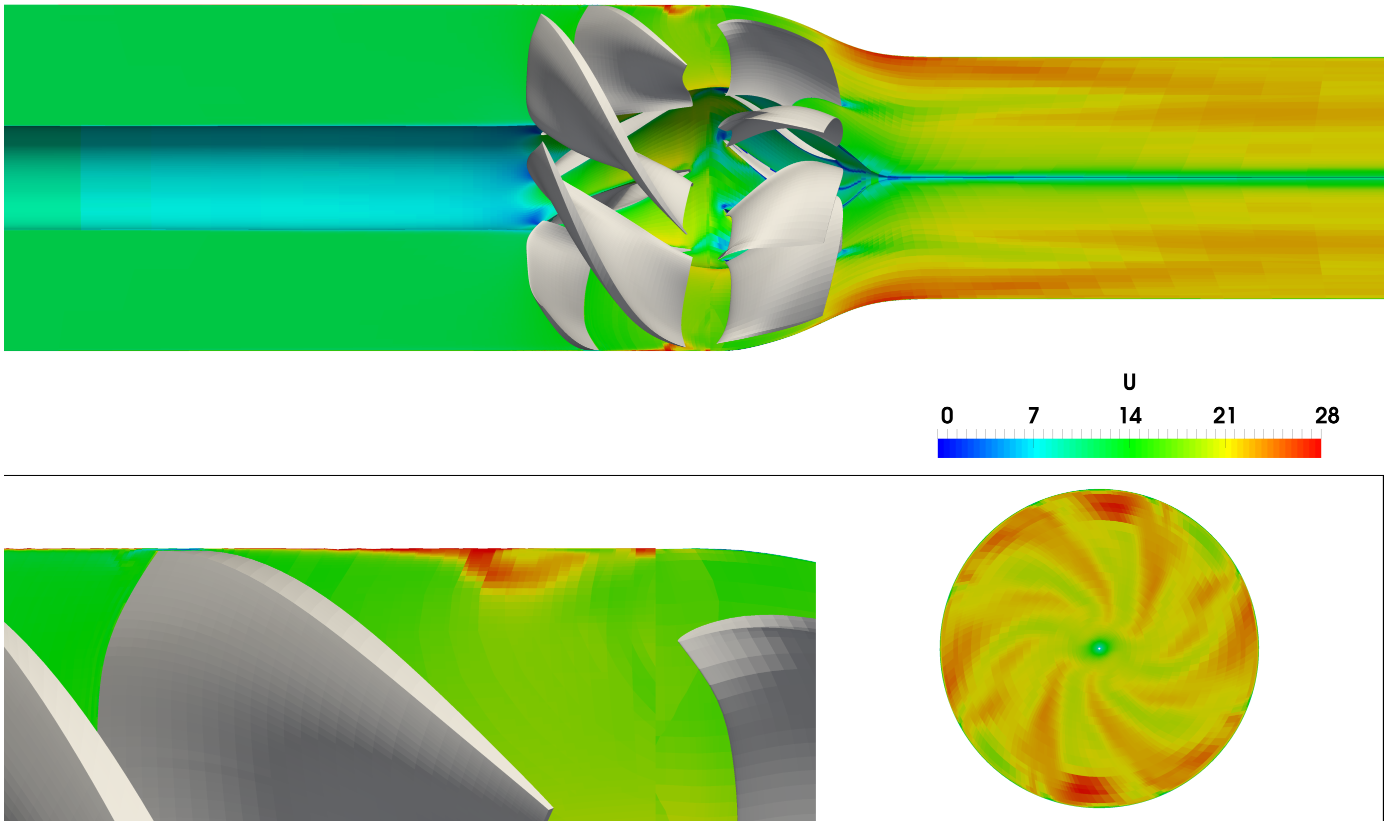








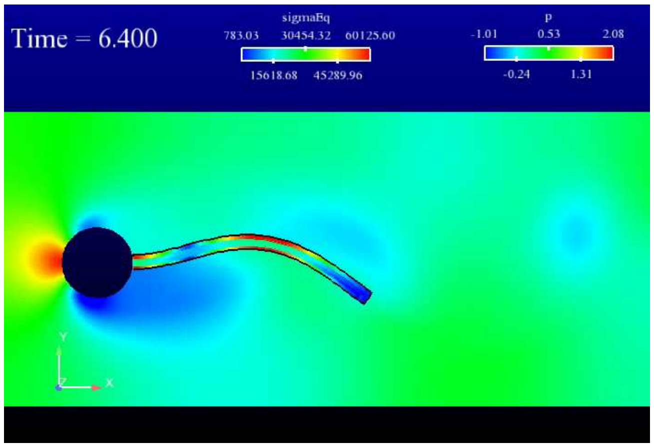
Turbomachinery

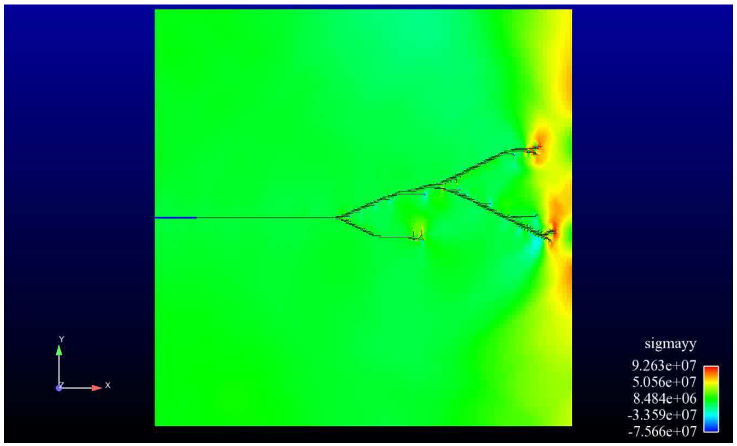


Fluid-Structure Interaction

Fluid-Structure Coupling Capabilities in OpenFOAM

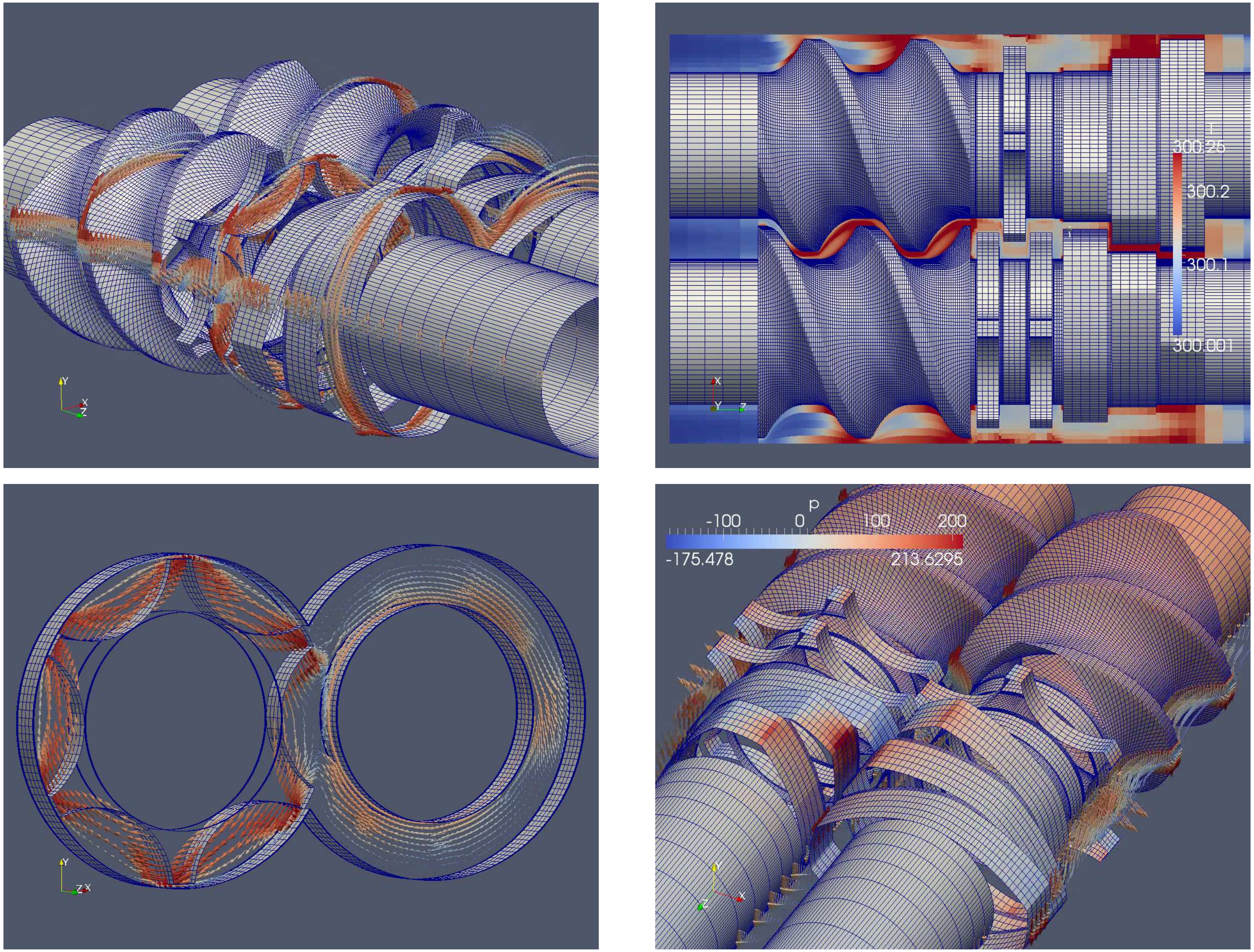
- As a Continuum Mechanics solver, OpenFOAM can deal with both fluid and structure components: easier setup of coupling
- (Parallelised) surface coupling tools implemented in library form: facilitate coupling to external solvers without “coupling libraries” using proxy surface mesh
- Structural mechanics 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



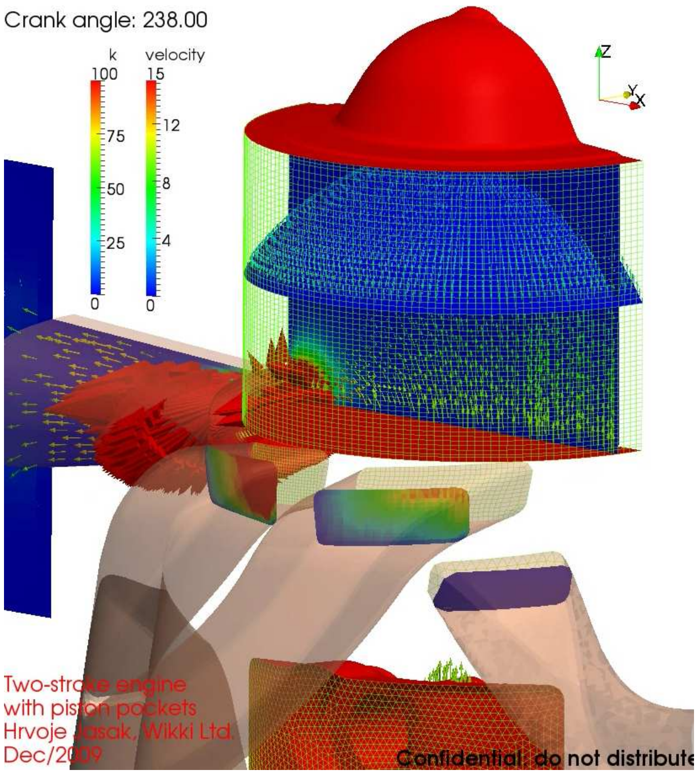


Complex Mesh Motion

Dynamic Mesh Examples of Complex Combination of Motion and Sliding



Topological Changes



Topological Changes

Topological Changes: Mesh Morphing

- 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
- Topological changes need to be automated and paired with (complex, dynamic) point motion, eg. layering or sliding

Topological Changes

WIKKI

Mesh Morphing Engine Implementation of Topological Changes in OpenFOAM

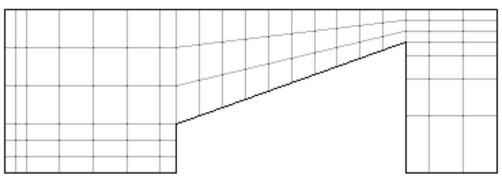
- **Primitive mesh operations**
  - Add/modify/remove a point, a face or a cell
  - This is sufficient to describe all cases, even to to build a mesh from scratch
  - . . . but using it directly is inconvenient
- **Topology modifiers**
  - Typical dynamic mesh operations can be described in terms of primitive operations. Adding a user-friendly definition and triggering logic creates a “topology modifier” class for typical operations
    - 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, 6-DOF motion, IC engine mesh (valves + piston)

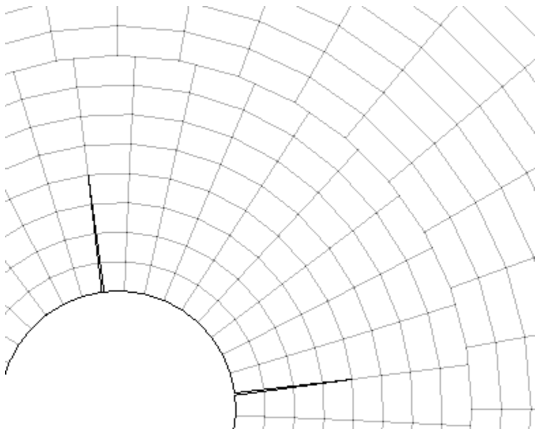
FSB

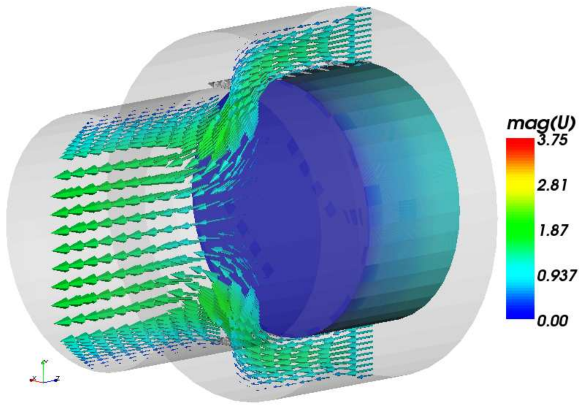
Topological Changes

Mesh Morphing Engine

- Each mesh modifier is self-contained, including the triggering criterion
- Complex cases contain combinations of modifiers working together, mesh motion and multi-step topological changes
- Polyhedral mesh support makes topological changes easier to handle: solver is always presented with a valid mesh







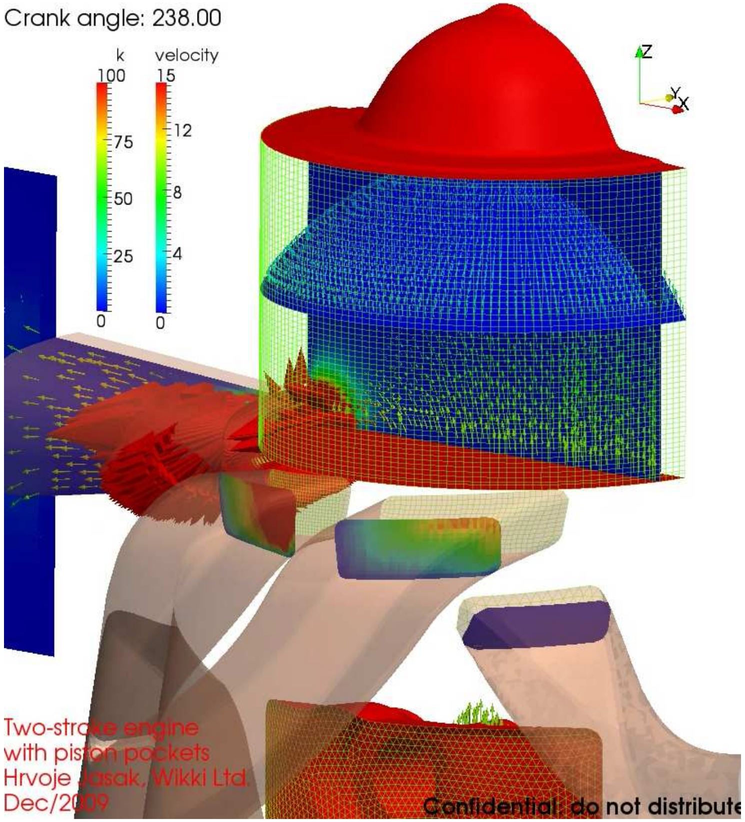
Flow control device, sliding interface and mesh layering

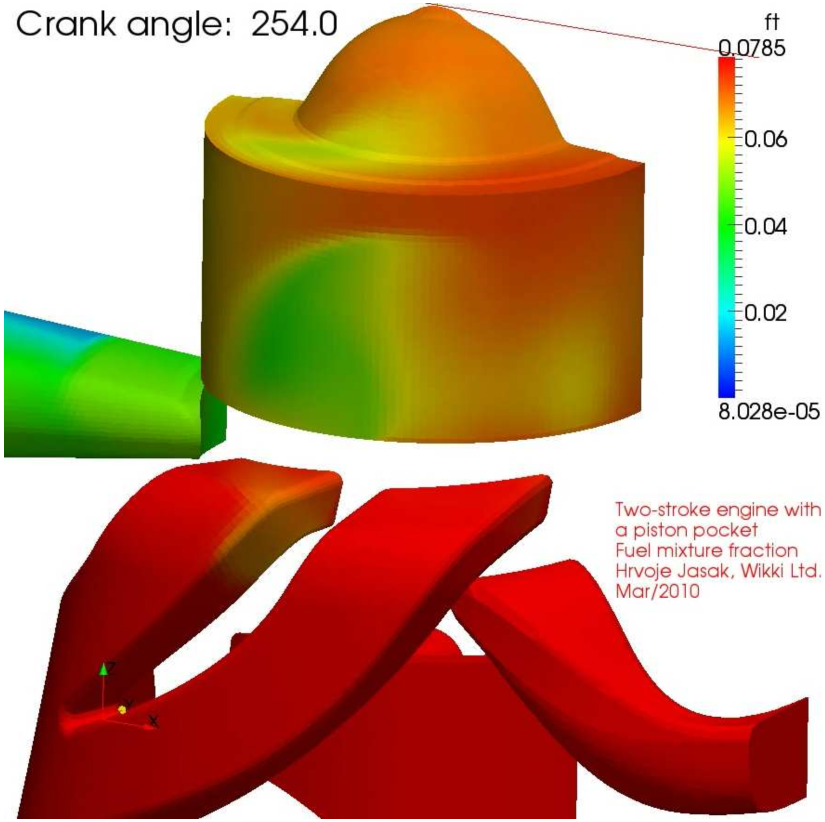
- Topological changes incorporate automatic data renumbering
- Conservation of local and global properties executed by special mesh motion steps: **no data mapping**
- Faces and cells are inflated from zero area/volume before insertion and removed at zero area/volume: mapping is replaced by mesh motion

Topological Changes

Simulating Cold Flow and EGR: Mixture Preparation

- Mixture preparation in a 2-stroke engine: mesh sliding and layering



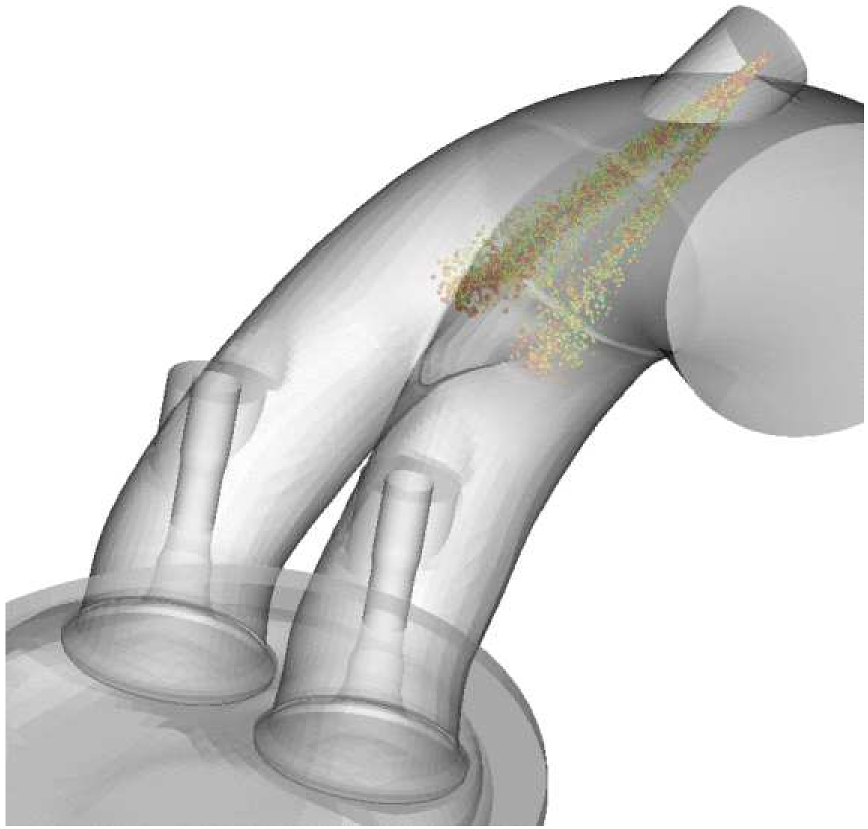


Topological Changes

WIKKI

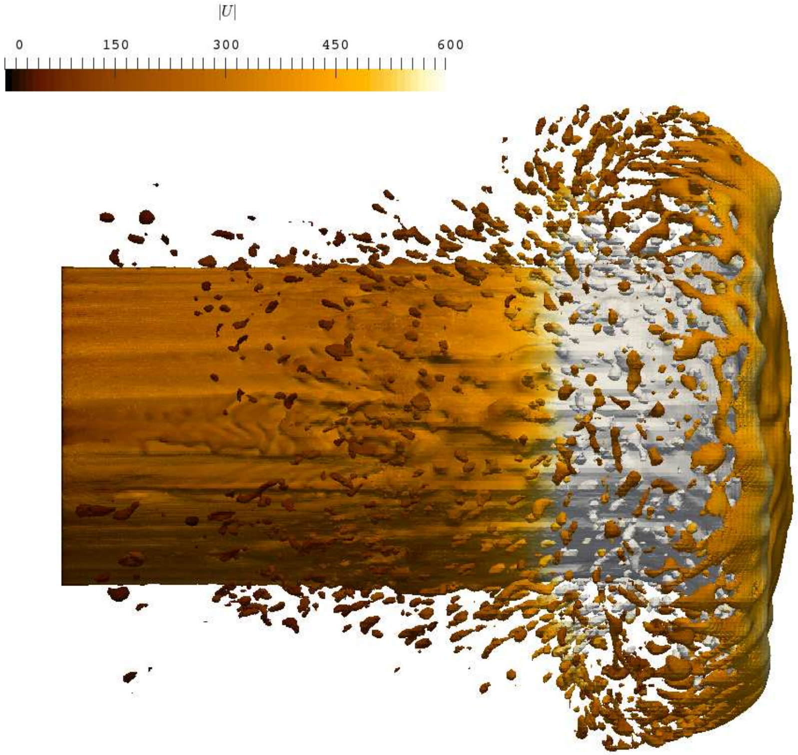
Volume-Surface-Lagrangian Simulation

- Main coupling challenge is to implement all components side-by-side and control their interaction
- Lagrangian tracking uses an ODE solver: block coupling at matrix level is not needed or cannot be used as before
- Close coupling is achieved by sub-cycling or iterations over the block system for each time-step
- If the model-to-model coupling fails, options on improving the stability are considerably limited
- Engine wall film simulation: courtesy of Politecnico di Milano



FSB

Polyhedral AMR With Load Balance



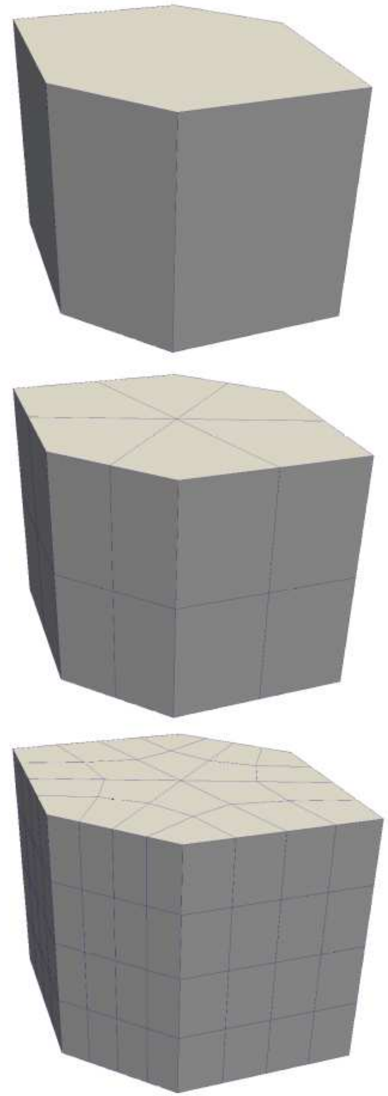
Polyhedral AMR With Load Balance

Polyhedral Adaptive Mesh Refinement

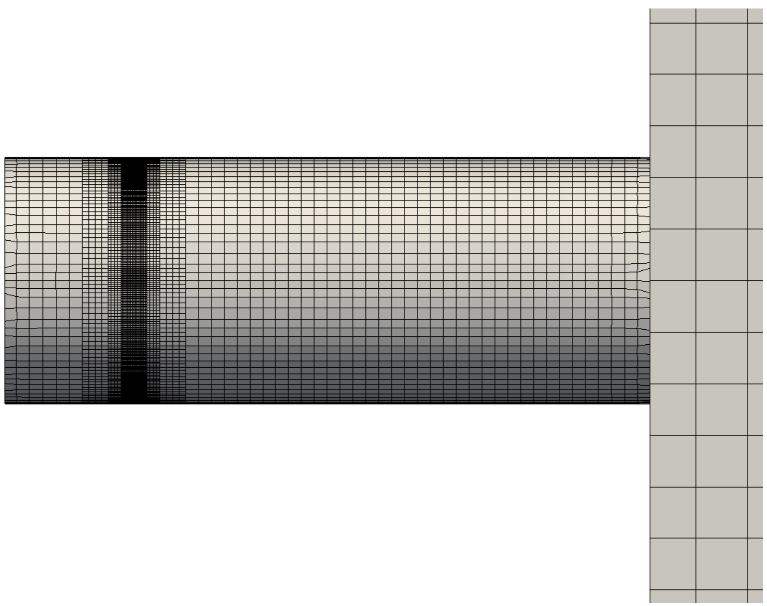
- Jasak PhD (1994): Shape-based refinement: `splitHex`
- Janssens (2003): `hexRef8` class
- Neither of above is great: no directional refinement, no 2-D adaptivity, no control of grading
- The rest of FOAM is polyhedral: refinement isn't!
- **Polyhedral cell adaptivity**: Jasak, Vukčević (2018)

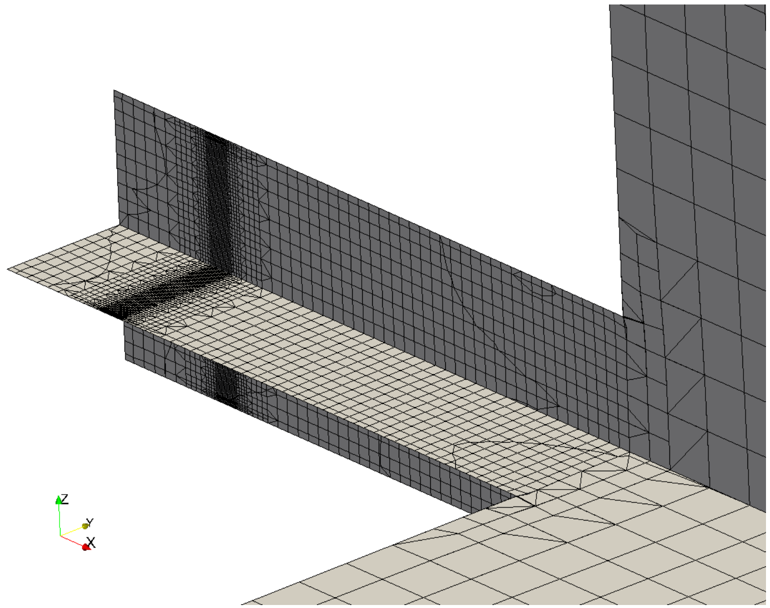
Dynamic Load Balancing

- **New implementation**: load balancing using decompose-reconstruct tools and `Pstream` communication
- Load balancing is now a **basic function** of `topoChangerFvMesh`



Polyhedral AMR With Load Balance



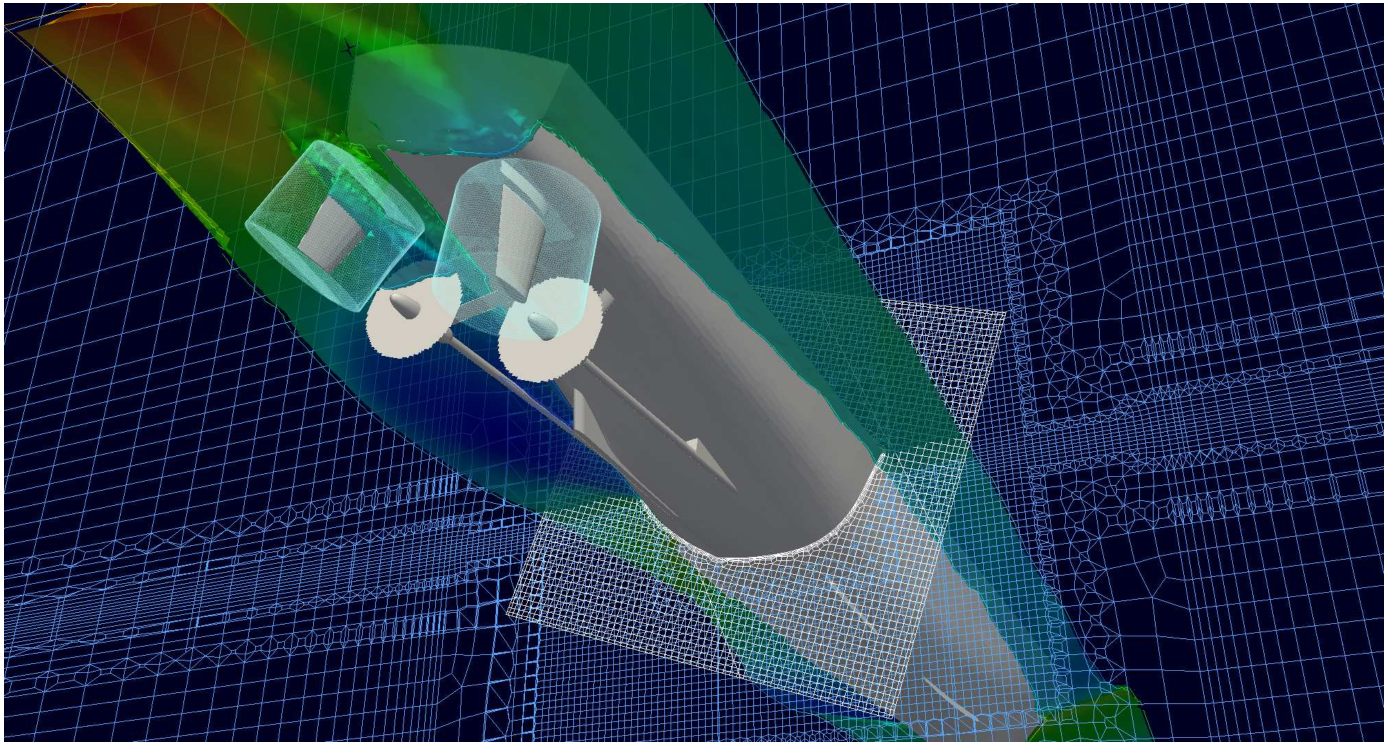


$Time: \ 2 \ \mu s$

$Time: \ 9 \ \mu s$

Polyhedral AMR With Load Balance

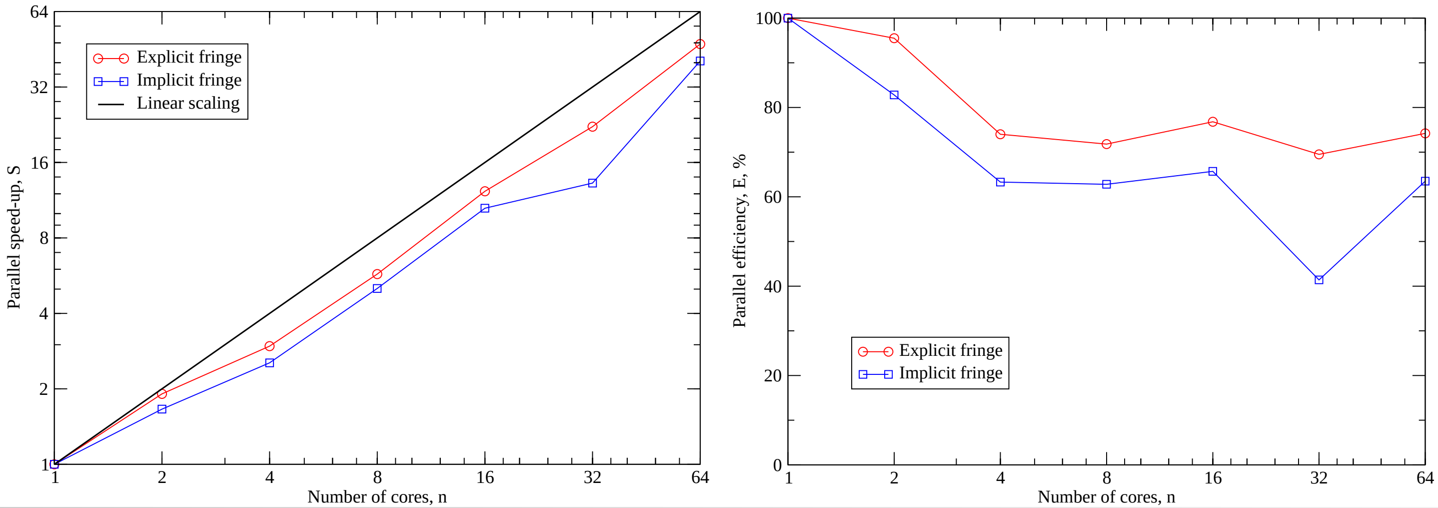
Overset Mesh



Overset Mesh

Parallel Efficiency of the Overset Mesh

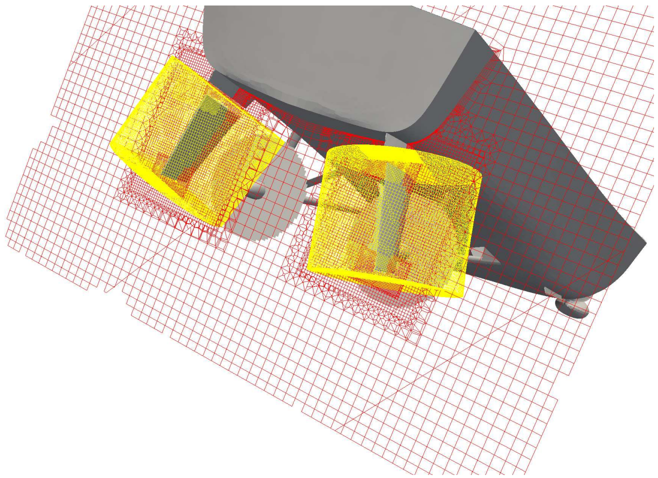
- Implementation of Overset interpolation performed similar to GGI
  - Interpolation performed in out-of-core multiplication with parallel comms
  - Parallelised using `mapDistribute` tool
- Parallel scaling test case
  - Scaling test performed on 20M cells submarine mesh
  - Approximately 40K donor/acceptor cells (0.4% of total cell count)
  - Performed 20 iterations with explicit and implicit Overset fringe
- Parallel speed-up on 64 cores: 41 (implicit) and 46 (explicit)
- **Parallel efficiency on 64 cores: 64% (implicit) and 74% (explicit)**

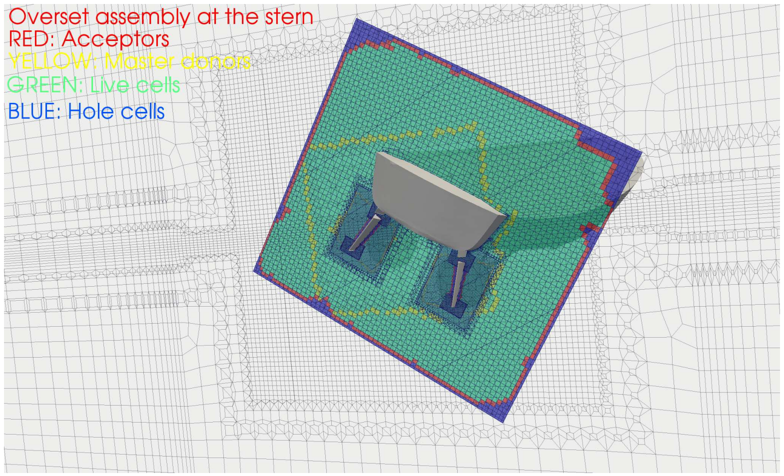


Overset Mesh: DTMB 5415

Single overset region for the ship grid communicates with:

- Background grid
- Two rudder grids (starboard and port-side)

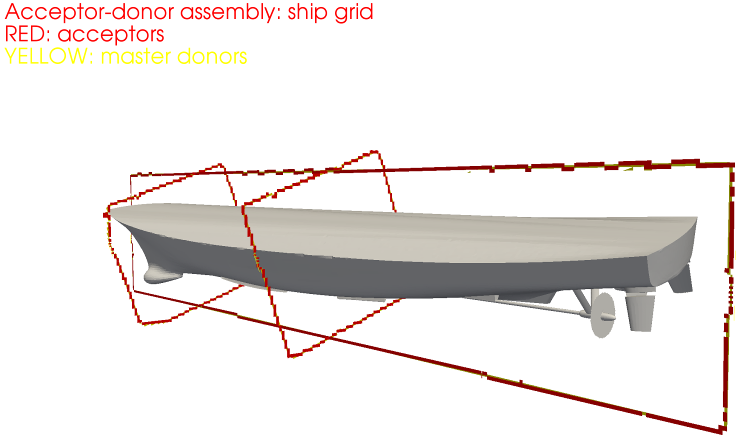


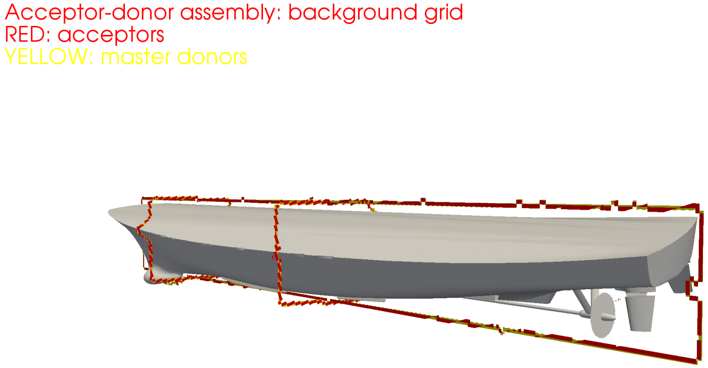


Overset Mesh: DTMB 5415

Adaptive Overlap Assembly has a robust **fallback mechanism**:

- Search for the best overlap according to user–specified criteria
- The search is stopped when the best overlap has been found

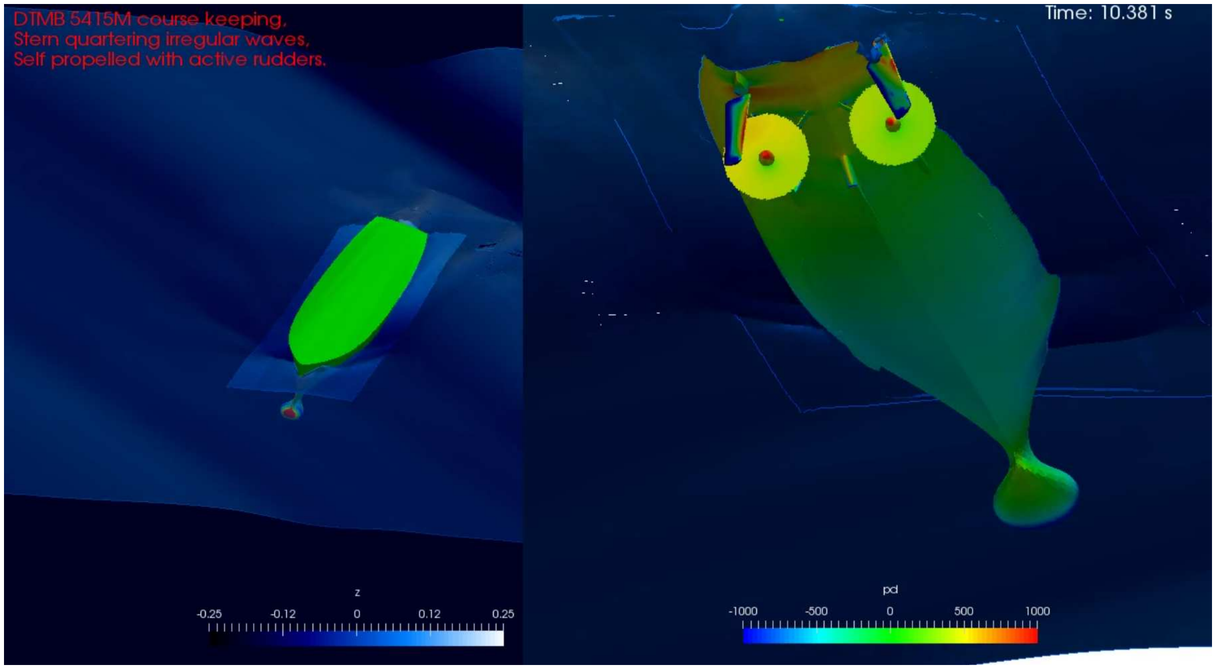




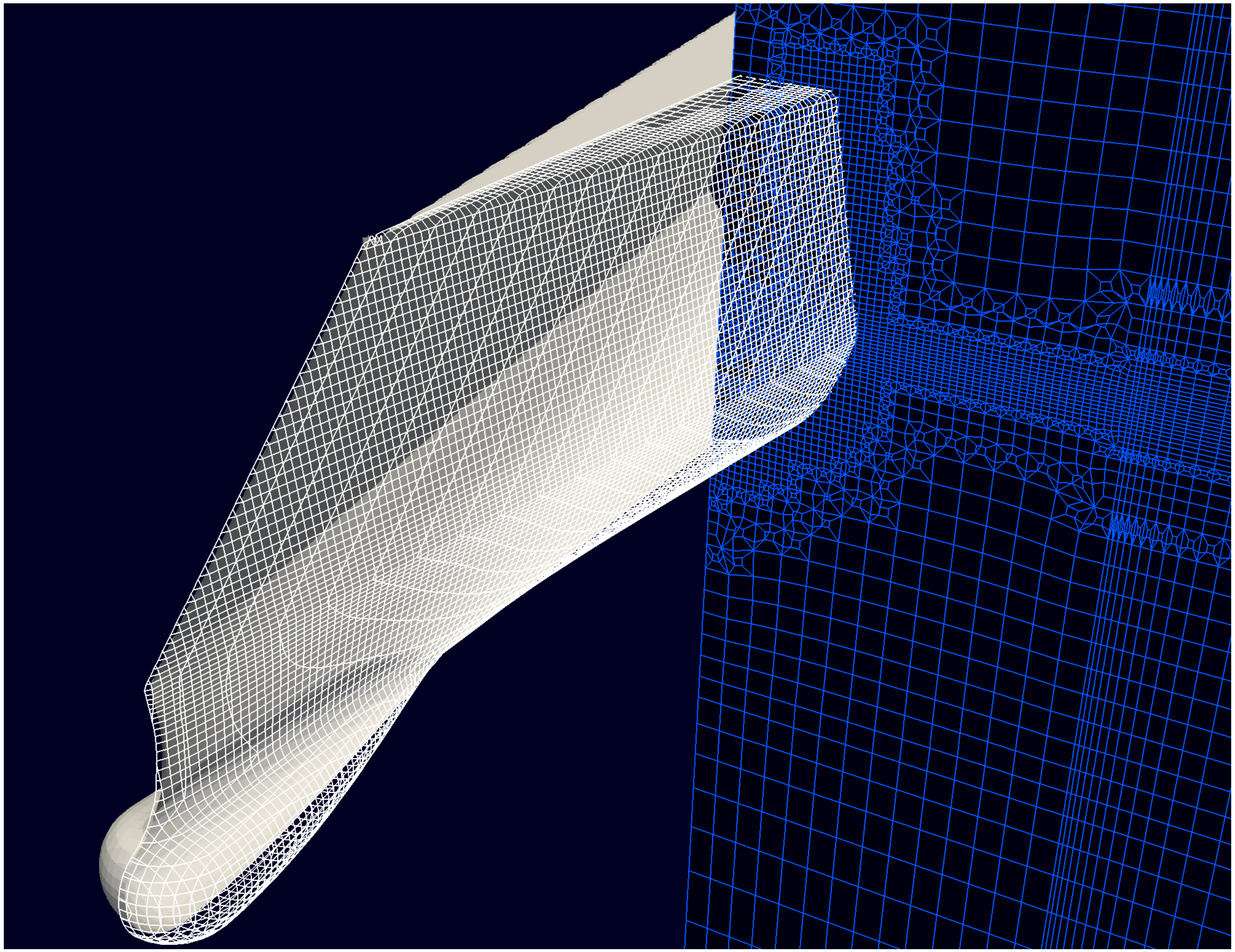
Overset Mesh: DTMB 5415

DTMB 5415 simulation

- 5.5 million cells with 4 overset regions: background, near hull and two rudders
- Irregular, **stern–quartering phase–focused waves**
- **Self–propelled** with two actuator discs
- **Two rudders with PID controllers for course–keeping**
- Running on 104 cores in parallel: roughly 10 peak periods in few days



Immersed Boundary



Immersed Boundary

Immersed Boundary Surface

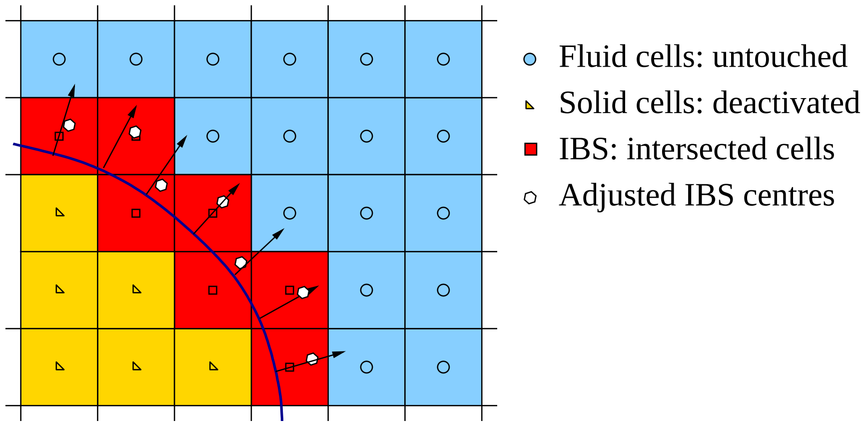
- IB implementation relies on the imposition of the boundary condition in the bulk of the mesh: this is built into the discretisation matrix
- **Objective**: implement the influence of the presence of a boundary within the mesh as if the mesh consists of **polyhedral body-fitted cells**:
  - Introduce the “new” IB face in the cut cell
  - Account for the partial cell volume without loss of accuracy
  - Account for partial face areas without loss of accuracy
  - Calculate face and cell centre consistent with cell cut
- **. . . without changing the geometric mesh at all!**

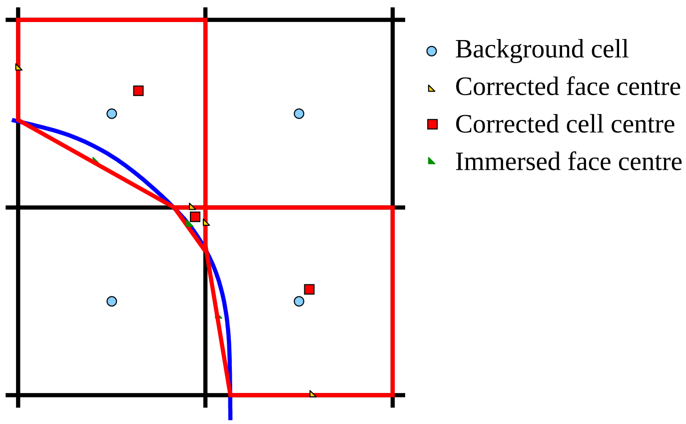
Advantages and Disadvantages

- IBS can eliminate volume mesh generation altogether
- Possible combination of body-fitted mesh and IB appendages or moving parts
- Due to wall functions, turbulent viscous force is (slightly) less accurate with IB

Immersed Boundary

Immersed Boundary Surface: Methodology

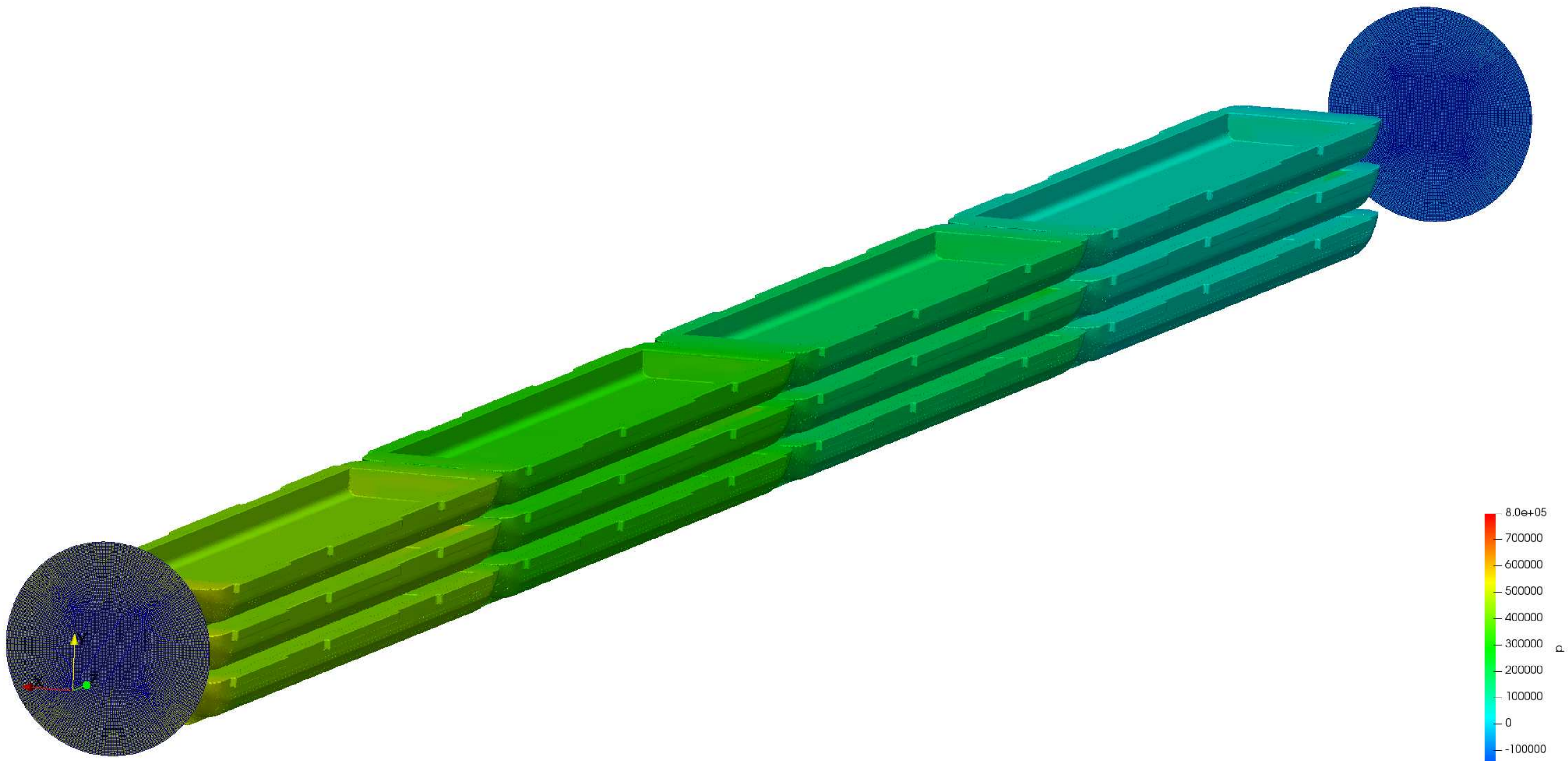




- Immersed boundary patch is included into the mesh via the distance function: **all cells that straddle the immersed boundary remain active**
- STL resolution or quality is not important: only using nearest distance
- Immersed intersection calculated based on point distance
  - All faces and cells are cut by a distance plane
  - Simple planar cutting provides robustness: no feature edges

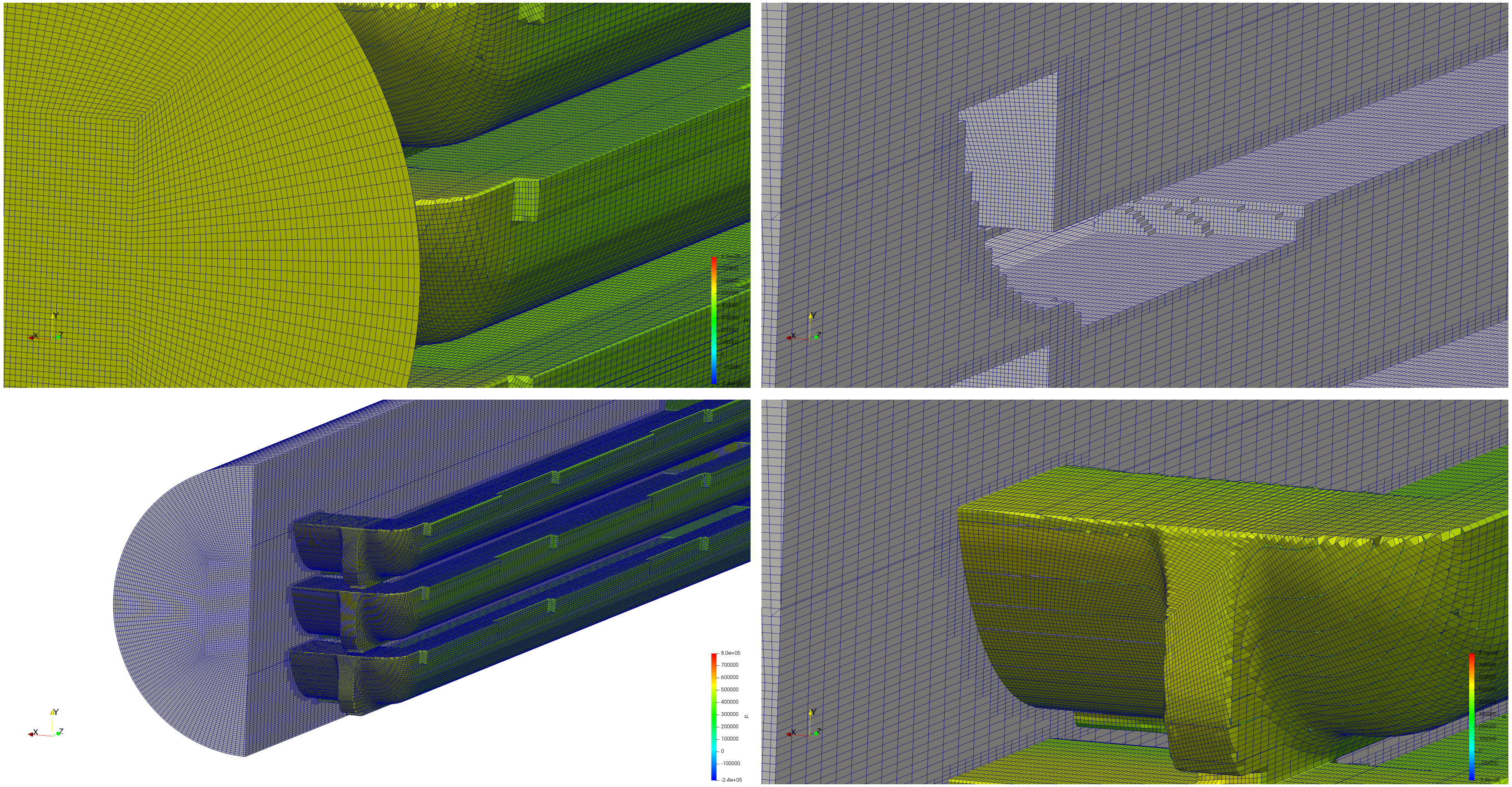
Immersed Boundary

Combined Immersed Boundary and Body-Fitted Mesh



Immersed Boundary

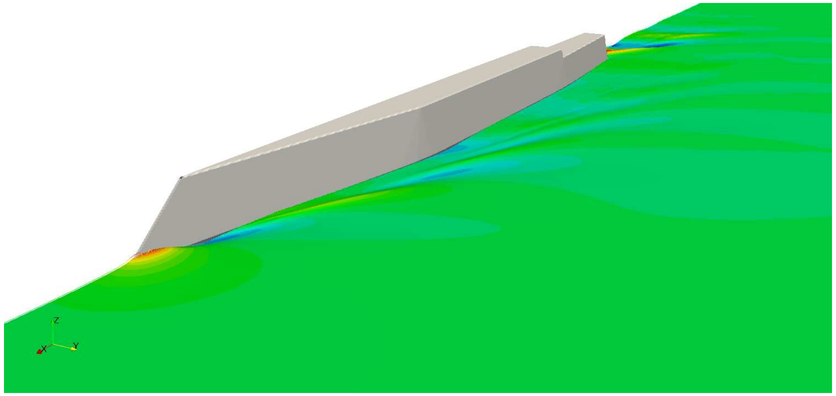
Combined Immersed Boundary and Body-Fitted Mesh

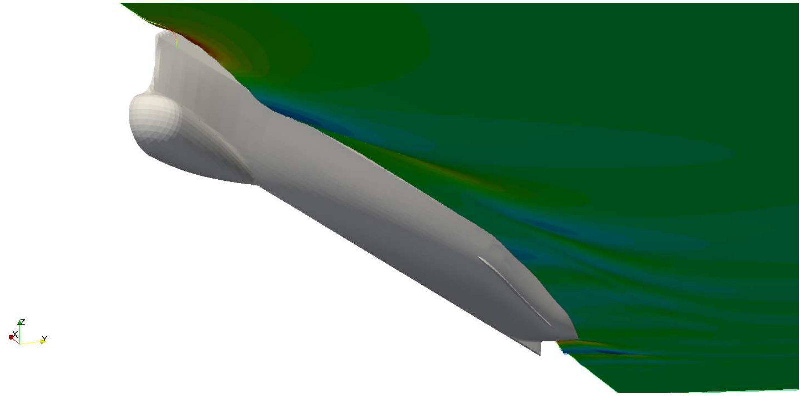


Immersed Boundary

ONR Tumblehome Ship Hull: Body-Fitted vs Immersed Boundary

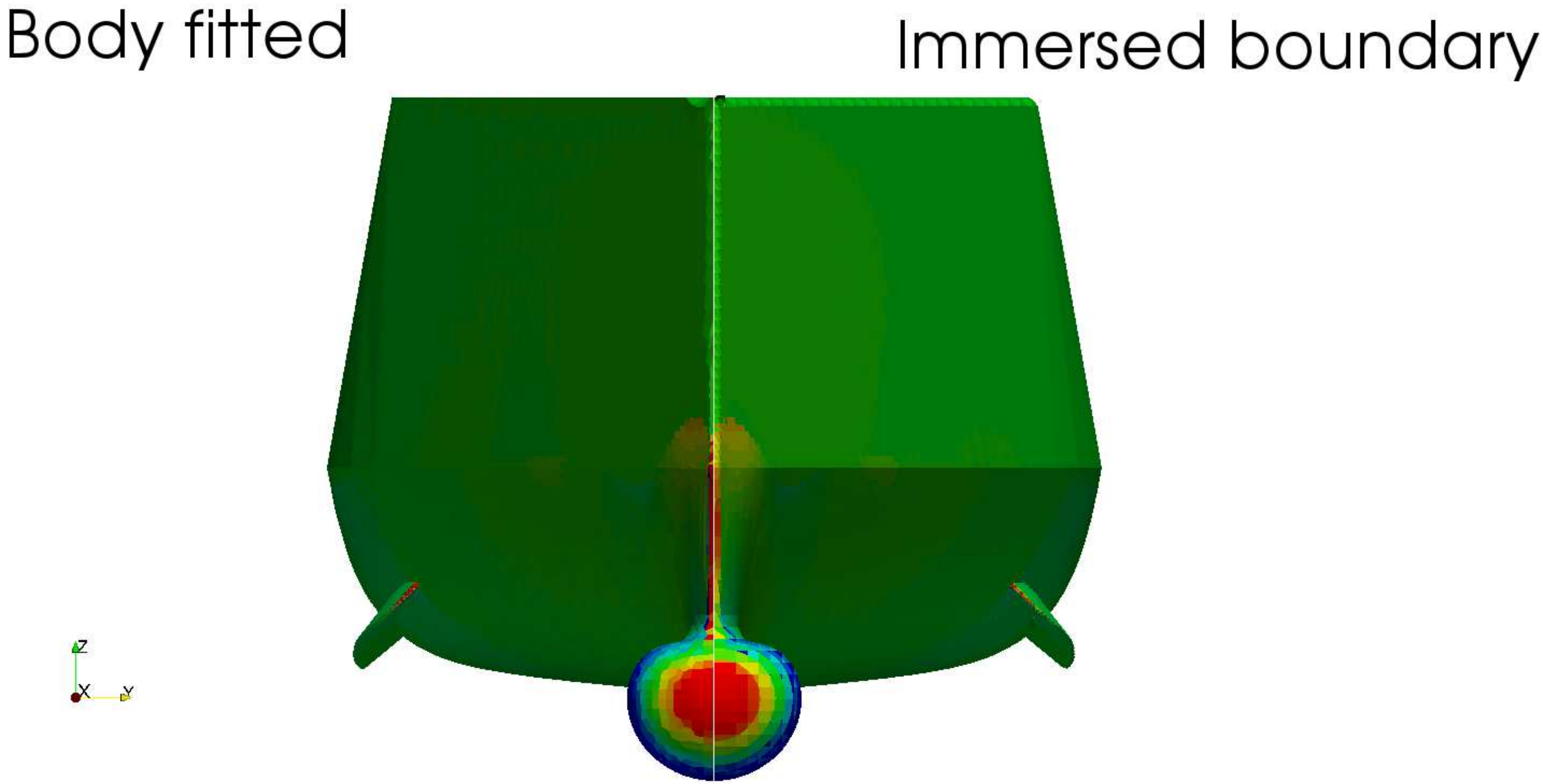
- Complete appended hull using Immersed Boundary: viscous drag test





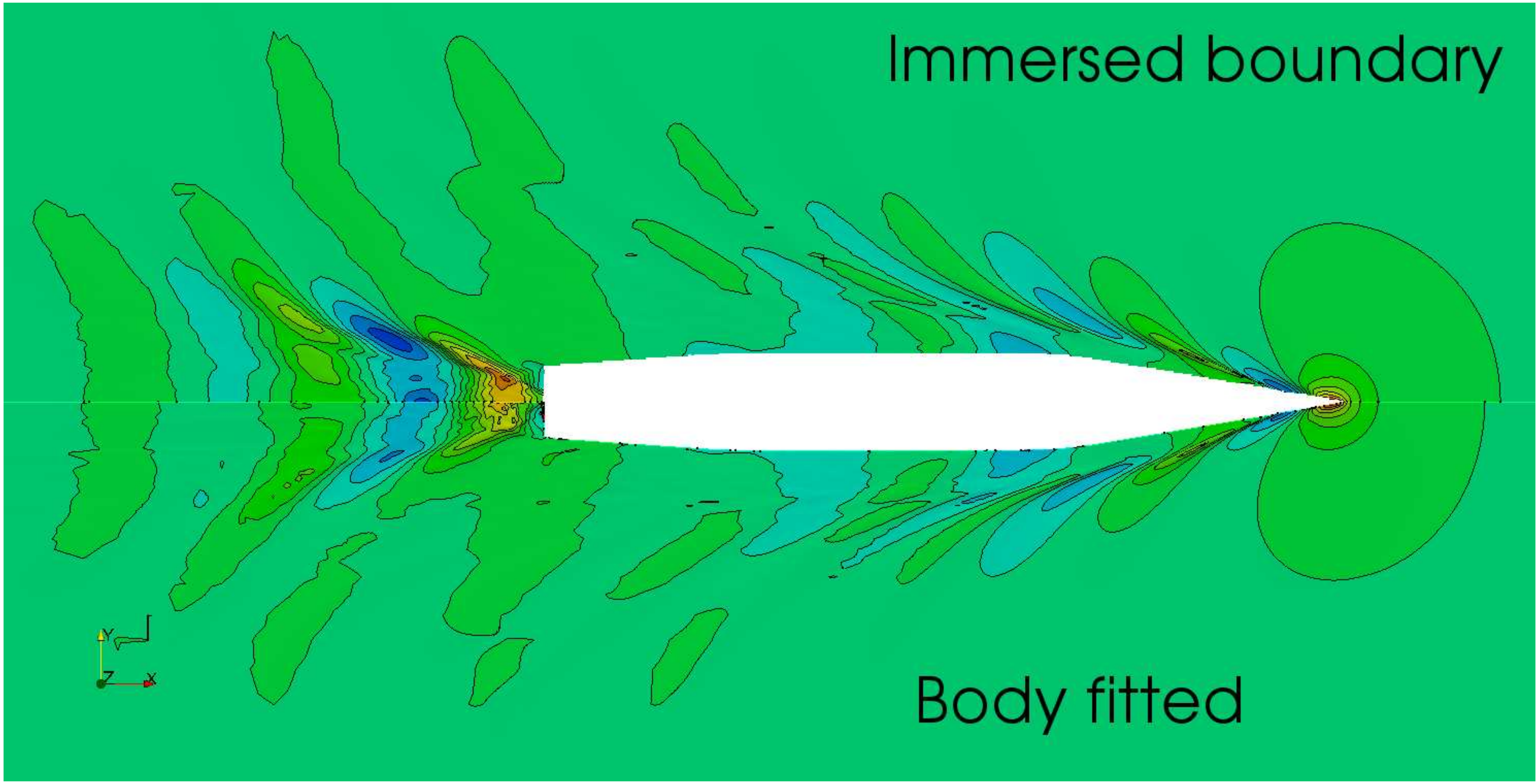
Immersed Boundary

ONR Tumblehome Ship Hull: Body-Fitted vs Immersed Boundary



Immersed Boundary

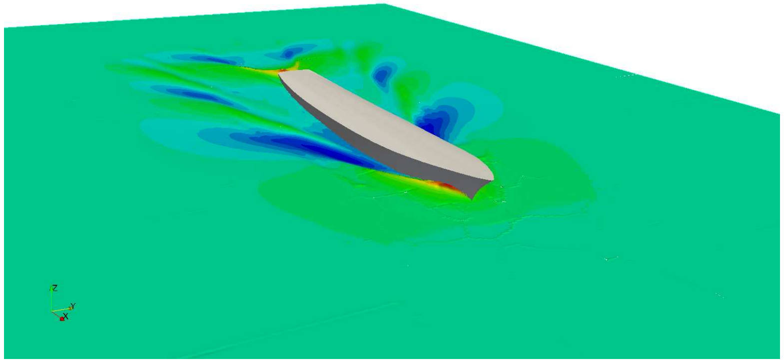
ONR Tumblehome Ship Hull: Body-Fitted vs Immersed Boundary

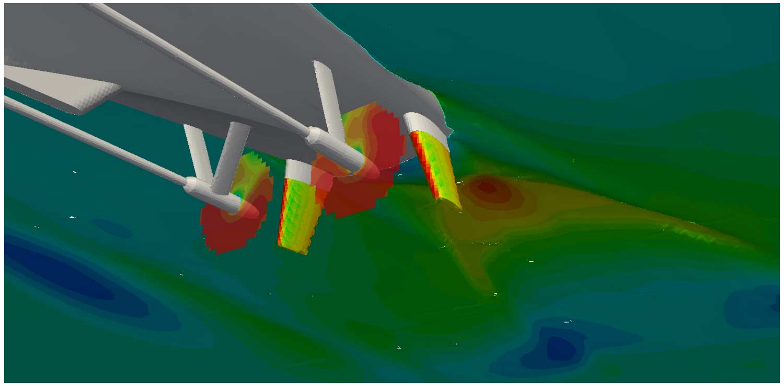


Immersed Boundary

Self-Propulsion in Calm Water (Preliminary Study)

- Self–propulsion in calm water
- PID controller for propeller rotation rate to achieve the desired ship speed
- Two propellers modelled with patch–type actuator disk model
- Static rudders modelled with Immersed Boundary
- Hull and static appendages are body fitted

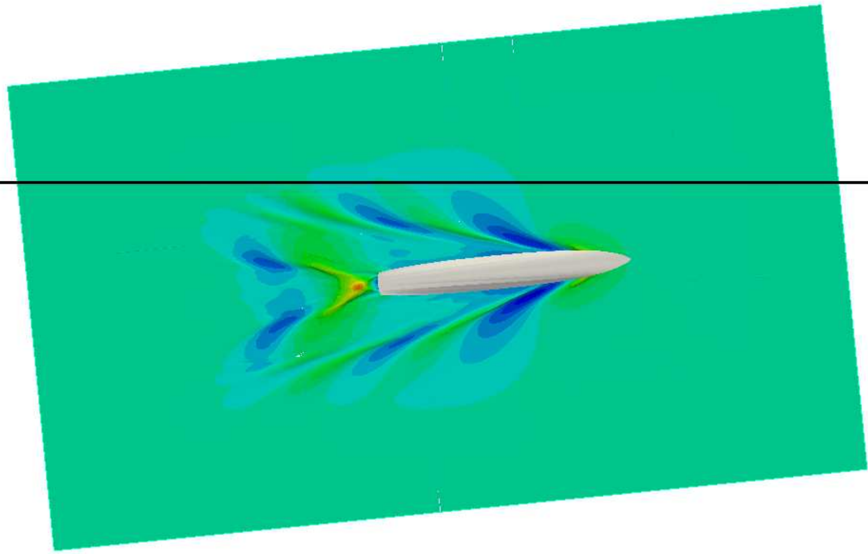


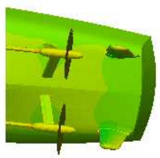


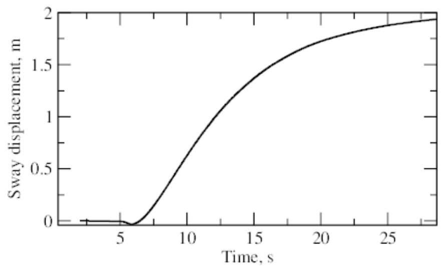
Immersed Boundary

Course-Keeping Test: Combined Body-Fitted and Immersed Boundary Mesh

- Free–running model with propellers at constant rotation rate
- Path offset at time zero to test the rudder controllers and the immersed boundary

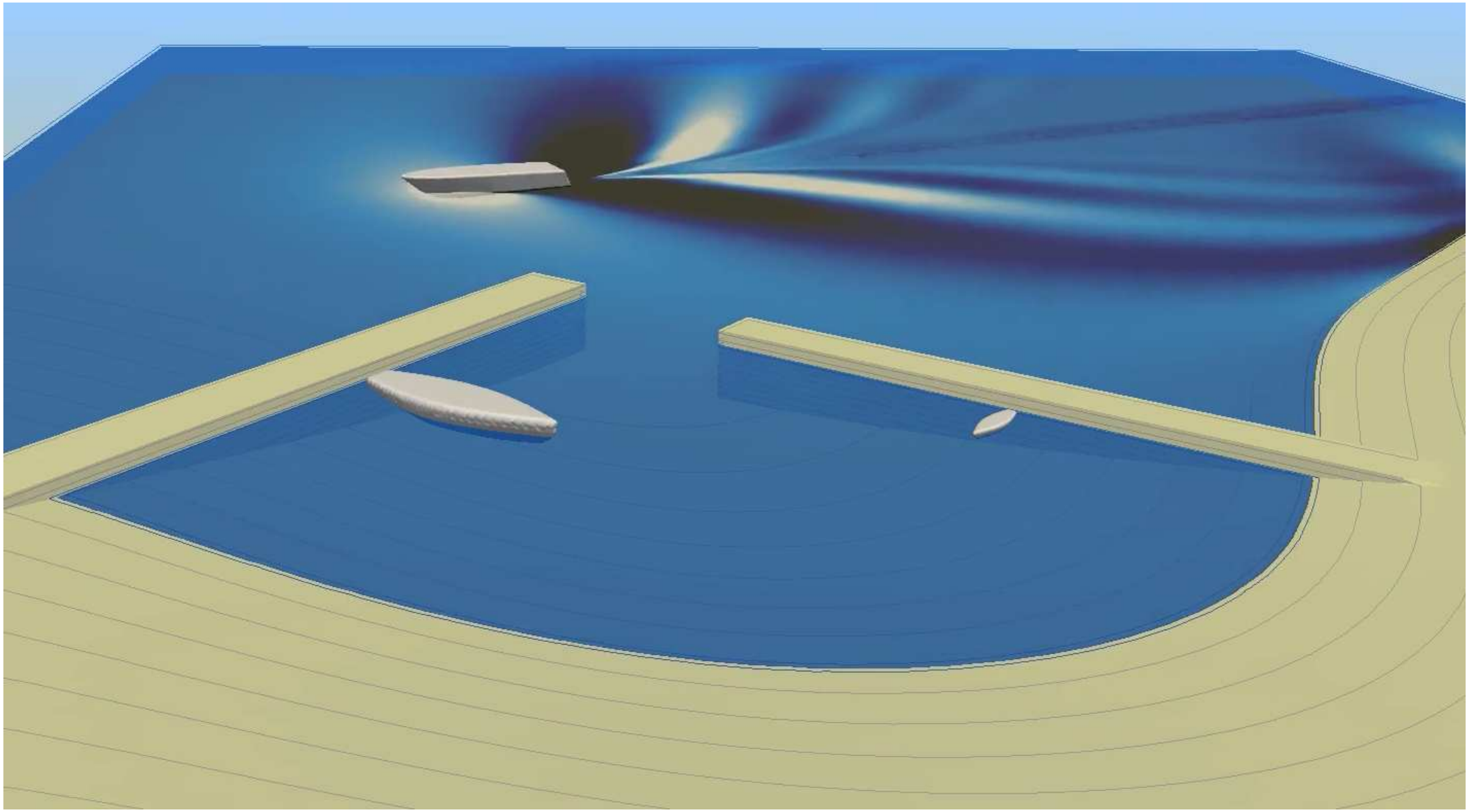






Immersed Boundary

Combined Immersed Boundary and Body-Fitted Mesh: Induced Wave Load



Summary

Summary

- OpenFOAM probably has world-leading dynamic mesh capability today
- . . . but most of it is only used by experts
- Library design allows multiple dynamic mesh techniques to be used together
- Traditional methods of automatic motion and topo changes look quite dated
- Overset mesh and immersed boundary are world-leading!
- Training, validation and verification may help

About Me

Role in OpenFOAM Development

- **One of two original developers of OpenFOAM software**, starting from 1993
- FVM discretisation, polyhedral mesh handling, linear solvers: **Jasak PhD 1996**
- Error estimation, adaptive mesh refinement, dynamic mesh, automatic mesh motion, topological changes: (sliding, layering); engine CFD
- Parallelism and HPC support: decomposition/reconstruction, comms
- Mesh generation, conversion, manipulation; pre- and post-processing tools
- Turbulence modelling, LES, free surface flows, solid mechanics, visco-elastic
- Finite Element motion solver, finite area method, ODE solvers
- POD, reduced order modelling
- Geometric parametrisation and automatic optimisation

Mesh Ordering in OpenFOAM

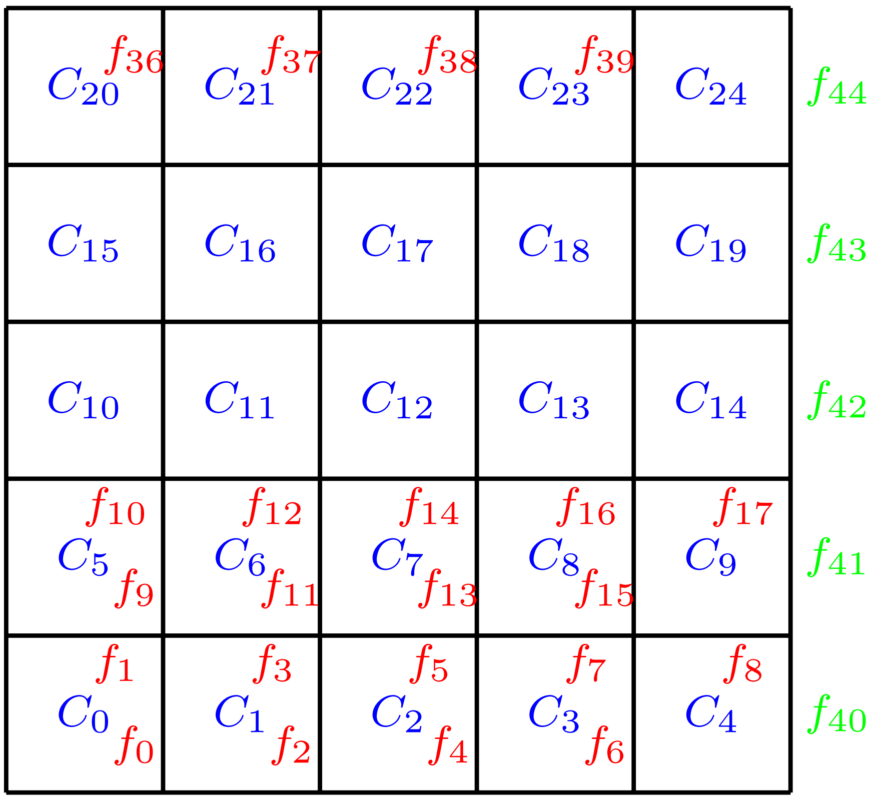
Strong Ordering Requirement

- Polyhedral mesh definition
  - List of vertices. Vertex position in the list determines its label
  - List of faces, defined in terms of vertex labels
  - List of cells, defined in terms of face labels
  - List of boundary patches
- **All indices start from zero**: C-style numbering (no discussion please)
- OpenFOAM uniquely orders mesh faces for easier manipulation
  - All internal faces are first in the list, ordered by the cell index they belong to. Lower index defines **owner cell** (P); face normal points out of the owner cell
  - Faces of a single cell are ordered with increasing neighbour label, *i.e.* face between cell 5 and 7 comes before face between 5 and 12
  - Boundary faces are ordered in patch order. All face normals point outwards of the domain
- With the above ordering, patches are defined by their type, start face in the face list and number of faces
- Above ordering allows use of List slices for geometrical information

Mesh Ordering in OpenFOAM

Strong Ordering Requirement

1. Number points and cells arbitrarily: band compression improves smoother performance
2. Insert all internal faces **based on cell ordering**: upper triangle
3. Add boundary face patch by patch (as ordered by the mesher)



Mesh Ordering in OpenFOAM

Strong Ordering Requirement: Face Addressing Format

- Face owner list: size of **all cells**
- Face neighbour list: size of **internal cell**
- Boundary patch: defined by **size and start face**

face	owner	neighbour
0	0	1
1	0	5
2	1	2
3	1	6
4	2	3
5	2	7
6	3	4
7	3	8
8	4	9
9	5	6
10	5	10
11	6	7
12	6	11

```
2
(
    Wing
    {
        type            wall;
        nFaces          154;
        startFace       23579;
    }
    Inlet
    {
        type            patch;
        nFaces          74;
        startFace       23733; <- 23579 + 154
    }
)
```

Zones and Sets in OpenFOAM

Operating on Sub-Spaces in the Mesh

- Zones and sets allow sub-setting of mesh elements
- Note: discretisation and matrix will always be associated with the complete mesh!
- Zones: points, faces and cells
  - Define partition of mesh elements. Each point/face/cell may belong to a maximum of one zone.
  - Fast two-directional query: what zone does this point belong to?
  - Used extensively in topological mesh changes

Definition of Zones

- A **Zone** is a collection of points/faces/cells which represent a mesh feature **within a contiguous numbering space**
- A zone remains invariant
  - In parallel execution (points/faces/cells are locally numbered)
  - Under topological changes (layering, sliding, refinement)

Examples of Use

- Definition of a rotating "space" for MRF
- Integrate flow measures in an "internal surface" within the mesh, which consists of **oriented collection of faces**: `flipMap` in a face zone

Zones and Sets in OpenFOAM

Definition of Sets

- Arbitrary grouping of points/faces/cells for manipulation
- Single cell may belong to multiple sets
- Sets used to create other sets: data manipulation with `setSet` tool
- Examples

```
faceSet f0 new patchToFace movingWall
faceSet f0 add labelToFace (0 1 2)
pointSet p0 new faceToPoint f0 all
cellSet c0 new faceToCell f0 any
cellSet c0 add pointToCell p0 any
```

- On completion, sets can be converted into zones: `setsToZones` utility

Multiple Equation Sets in OpenFOAM

Code Organisation

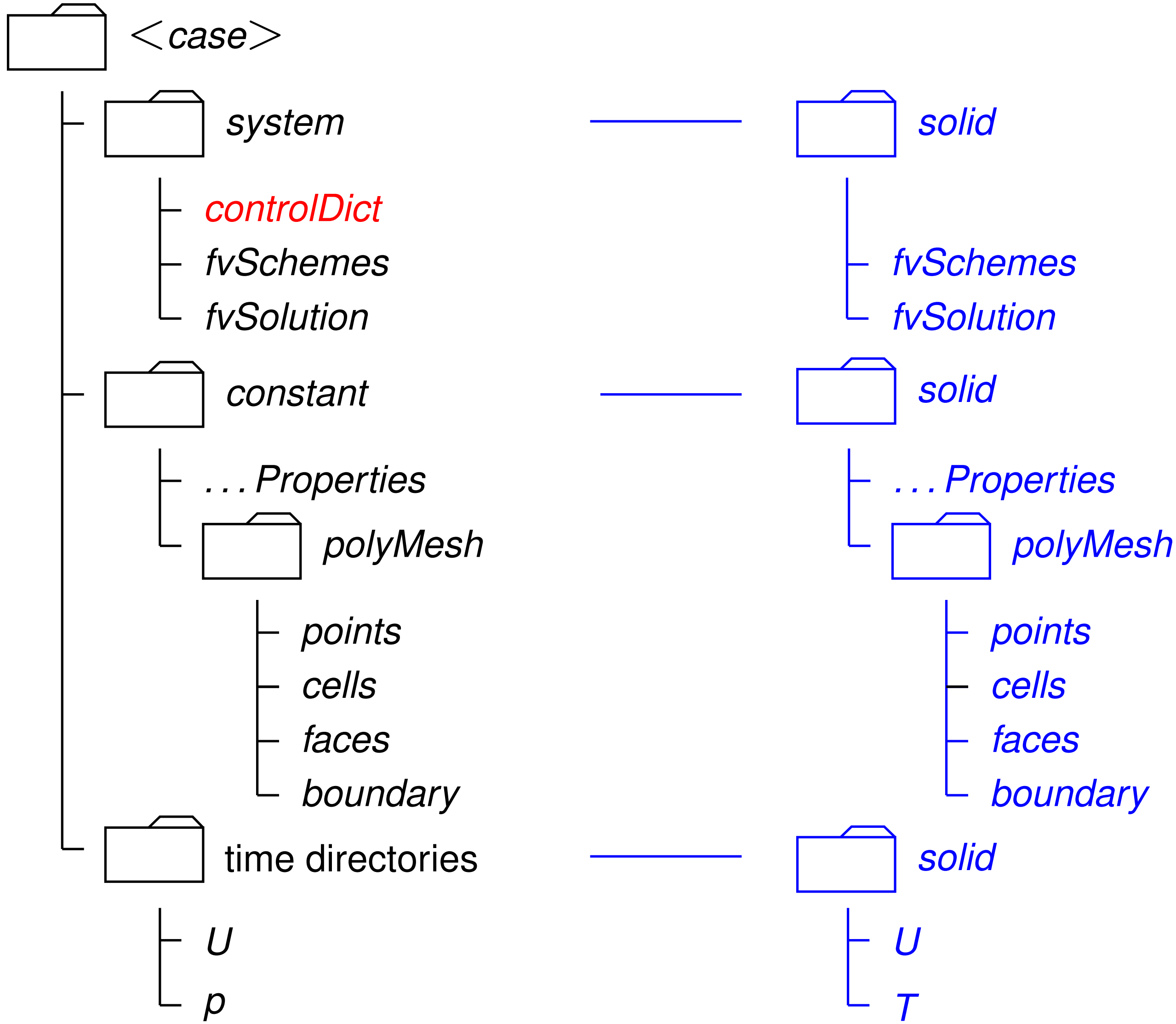
- Every individual mesh **region** represents a **single addressing space**, with its own internal faces and boundaries. Operations on various face types are consistent: consequences for conjugate heat transfer type of coupling
- Combining variables or addressing spaces into implicit coupling requires special practices and tools

Multiple Domains in a Single Simulation

- Original class-based design allows for multiple object of the same type in a single simulation, e.g. meshes and fields
  - Multiple named mesh databases within a single simulation:
    1 mesh = 1 domain, with separate fields and physics
  - Fields, material properties and solution controls separate for each mesh

Multiple Equation Sets in OpenFOAM

Case Organisation for Multiple Meshes: “Main Mesh” and `solid`



Multiple Equation Sets in OpenFOAM

Example: Conjugate Heat Transfer

- T-equation spans multiple meshes
- Conjugate solid wall is present as a boundary condition on `T` and `Tsolid`

```
coupledFvScalarMatrix TEqns(2);
TEqns.set
(
    0,
    fvm::ddt(T) + fvm::div(phi, T)
  - fvm::laplacian(DT, T)
);
TEqns.set
(
    1,
    fvm::ddt(Tsolid) - fvm::laplacian(DTsolid, Tsolid)
);

TEqns.solve();
```

- Coupled solver handles multiple matrices together in internal solver sweeps
- . . . and the linear equation solver sees a “single addressing space”