051176 - Computational Techniques for Thermochemical Propulsion

Master of Science in Aeronautical Engineering

Dynamic Mesh Handling in the FV Method

Prof. **Federico Piscaglia**

Dept. of Aerospace Science and Technology (DAER)

POLITECNICO DI MILANO, Italy

federico.piscaglia@polimi.it

Moving Grids

In many application areas the solution domain changes with time due to the movement of boundaries. The movement is determined either by

- external effects (as in piston-driven flows)
- calculation as part of the solution (for example, in free-surface flows).

In either case,

1) the grid has to move to accommodate the changing boundary (`motionSolver`);
2) if the coordinate system remains fixed and the Cartesian velocity components are used, the only change in the conservation equations is the appearance of the relative velocity in convective terms;

Moving Grids - 1D

First consider the one dimensional continuity equation:

$$\frac{\partial \rho}{\partial t} + \frac{\partial \left(\rho v\right)}{\partial x} = 0$$

By integrating this equation over a control volume whose boundaries move with time, from $x_1(t)$ to $x_2(t)$, it follows:

$$\frac{\partial}{\partial t}\int_{x_1(t)}^{x_2(t)} \rho dx + \frac{\partial}{\partial x}\int_{x_1(t)}^{x_2(t)} \left(\rho v\right) dx = 0$$

The time derivative can be calculated by the Leibniz rule (integration by parts) as:

$$\frac{\partial}{\partial t}\int_{x_1(t)}^{x_2(t)} \rho dx = \frac{d}{dt}\int_{x_1(t)}^{x_2(t)} \rho dx - \left[\rho_2 \frac{dx_2}{dt} - \rho_1 \frac{dx_1}{dt}\right]$$

so the governing equation becomes:

$$\boxed{\frac{d\rho}{dt} + \frac{\partial}{\partial x}\left[\rho\left(v - v_b\right)\right] = 0}$$

Leibniz rule - Time Derivative

The Leibniz rule (integration by parts) reads as:

$$\begin{aligned}\int_a^b u(x)v'(x)\,dx &= \Big[u(x)v(x)\Big]_a^b - \int_a^b u'(x)v(x)\,dx \\ &= u(b)v(b) - u(a)v(a) - \int_a^b u'(x)v(x)\,dx\end{aligned}$$

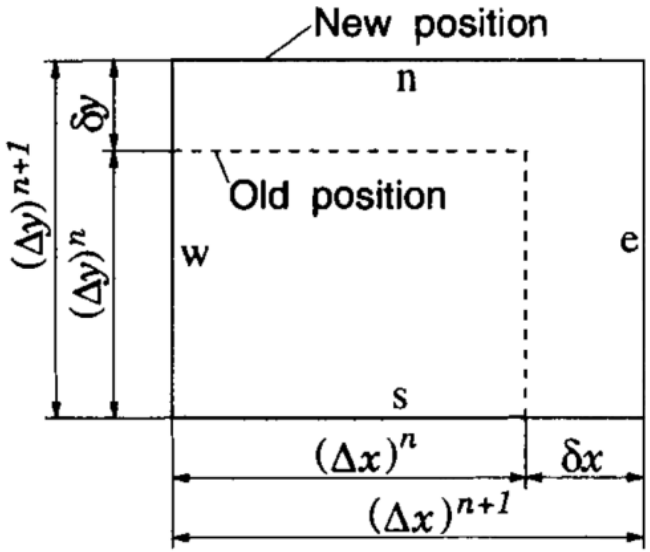
If we apply the Leibniz rule to our case:

$$\int_{x_1(t)}^{x_2(t)} \rho dx = \Big[\rho\, x\Big]_{x_1(t)}^{x_2(t)} - \frac{\partial}{\partial x}\int \rho\, x\, dx$$

so:

$$\begin{aligned}\frac{\partial}{\partial t}\int_{x_1(t)}^{x_2(t)} \rho dx &= \left[\rho_2\frac{dx_2}{dt} - \rho_1\frac{dx_1}{dt}\right] - \frac{\partial}{\partial t}\int \frac{\partial \rho}{\partial x}\, x\, dx \\ &= \left[\rho_2\frac{dx_2}{dt} - \rho_1\frac{dx_1}{dt}\right] \underbrace{-\frac{\partial}{\partial x}\int \rho\,\frac{dx}{dt}\,dx}_{-\nabla\rho\boldsymbol{u}} \\ &= \left[\rho_2\frac{dx_2}{dt} - \rho_1\frac{dx_1}{dt}\right] + \frac{\partial \rho}{\partial t}\end{aligned}$$

Mesh deformation/stretching



Relative sizes of the CV at the old and new time levels. If we assume that the grid lines (CV faces) move with constant, but different velocities, the size of the CV grows with time. The discretized continuity equation for the CV with the implicit Euler scheme reads:

$$\frac{\rho[(\Delta V)^{n+1} - (\Delta V)^n]}{\Delta t} + \rho[(u-u_b)_e - (u-u_b)_w]^{n+1}(\Delta y)^{n+1} + [(u-u_b)_n - (u-u_b)_s]^{n+1}(\Delta x)^{n+1} = 0$$

where u and v are the Cartesian velocity components.

Transport Eqn. with Moving Boundaries

If ϕ is any conserved intensive property (for mass conservation, ϕ=1; for momentum conservation, $\phi = \boldsymbol{v}$ = v; for conservation of a scalar, ϕ represents the conserved property per unit mass), then the corresponding extensive property ϕ can be expressed as:

$$\frac{D}{Dt}\int_V \rho\,\phi dV = \frac{\partial}{\partial t}\int_V \rho\phi dV + \int_S \rho\phi(\vec{v}-\vec{v}_b)\cdot\vec{n}dS$$

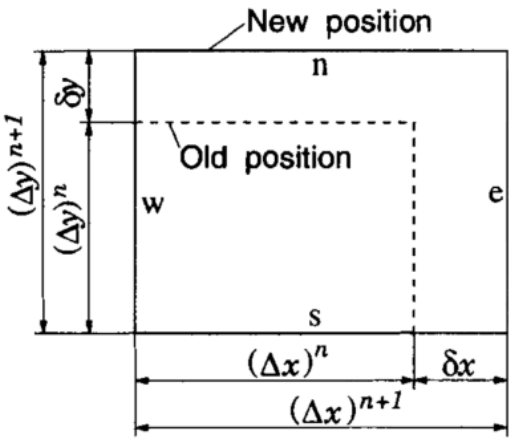
where V is the CV volume, S is the surface enclosing CV, $\boldsymbol{n}$ is the unit vector orthogonal to S and directed outwards, $\boldsymbol{u}$ is the fluid velocity and $\boldsymbol{u}_b$ is the velocity with which the CV surface is moving.

$\rightarrow$ for a fixed CV, which we shall be considering most of the time, $\boldsymbol{u}_b = 0$

SOME NOTES:

- When the boundary moves with fluid velocity, i.e. $\boldsymbol{v}_b = \boldsymbol{v}$, the second integral becomes zero and we have the Lagrangian mass conservation equation, $dm/dt = 0$;
- if the CV does not move, the equations are those dealt with earlier;
- when the cell faces move, conservation of mass (and all other conserved quantities) is not necessarily ensured if the grid velocities are used to calculate the mass fluxes.

Moving Grids - Artificial Mass Source



Since we assume that the fluid moves with a constant velocity in the Control Volume CV, the contribution of fluid velocity in the above equation cancels out and only the difference in grid velocities remains:

$$\frac{\rho}{\Delta t}[(\Delta V)^{n+1} - (\Delta V)^n] - \rho(u_{b,e} - u_{b,w})(\Delta y)^{n+1} - (v_{b,n} - v_{b,s})(\Delta x)^{n+1} = 0$$

Under the assumptions made above, the difference in grid velocities at the opposite CV sides can be expressed as:

$$u_{b,e} - u_{b,w} = \frac{\delta x}{\Delta t}; \qquad v_{b,n} - v_{b,s} = \frac{\delta y}{\Delta t}$$

By substituting these expressions for u_b e v_b into the equation for mass conservation, we find that the discretized mass conservation equation is not satisfied: **there is a mass source**!

$$\delta \dot{m} = \frac{\rho(\delta x \delta y)}{\Delta t} = \rho(u_{b,e} - u_{b,w})(v_{b,n} - v_{b,s})\Delta t$$

Conservation Properties in Moving Grids

Possible issues in mass conservation with moving boundary problems:

1. Artificial mass sources:

$$\delta\dot{m} = \frac{\rho(\delta x \delta y)}{\Delta t} = \rho(u_{b,e} - u_{b,w})(v_{b,n} - v_{b,s})\Delta t$$

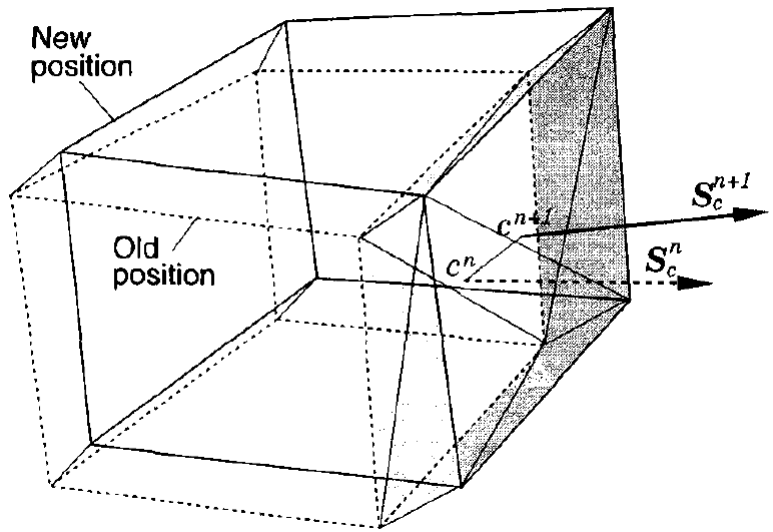
   may accumulate with time and cause serious problems. The error disappears if only one set of grid lines moves, or if the grid velocities are equal at opposite CV sides.

2. When the fluid and/or grid velocities are not constant, time discretization schemes can also produce artificial mass sources. Mass conservation can be obtained by enforcing the so-called **space (or grid) conservation law (SCL)**, which can be thought of as the continuity equation in the limit of zero fluid velocity:

$$\frac{\partial}{\partial t}\int_V dV - \int_S \boldsymbol{v}_b \cdot \boldsymbol{n} dS = 0$$

   This equation describes the conservation of space when the CV changes its shape and/or position with time.

Grid Conservation Law



Mass conservation can be obtained by enforcing the so-called **space (or grid) conservation law (SCL)**, which can be thought of as the continuity equation in the limit of zero fluid velocity:

$$\frac{\partial}{\partial t}\int_V dV - \int_S \boldsymbol{v}_b \cdot \boldsymbol{n} dS = 0$$

This equation describes the conservation of space when the CV changes its shape and/or position with time.

Grid Conservation Law (GCL)

$$\frac{\partial}{\partial t}\int_{\tilde{\Omega}(t)} \rho\phi \,\mathrm{d}V + \int_{\partial\tilde{\Omega}(t)} \rho\phi_f(\boldsymbol{U}-\boldsymbol{U}_b)\cdot\boldsymbol{n}\,\mathrm{d}S = 0$$

The discrete form of the GCL is obtained by exploiting the Reynolds' theorem to convert volume integrals into surface integrals:

$$\frac{\partial}{\partial t}\rho\,\phi V + \sum_f \rho_f\,\phi_f\left(\varphi_f - \varphi_{M,f}\right) - \sum_f \gamma_\phi \nabla\phi_f = s_\phi V$$

where:

- φ_f is the cell face flux $\varphi_f = [\![\boldsymbol{U}]\!]_f \cdot \boldsymbol{n}$
- S_f (with S_f the face area and $\boldsymbol{n}$ its normal unity vector)
- $\varphi_{M,f}$ is the corresponding fluxes due to point motion.

REMEMBER: with the co-located variable arrangement, all the primitive variables are assigned to the cell centroids, while face-centered variables are obtained by interpolation (operator $[\![\cdot]\!]_f$).

Grid Conservation Law: some notes

SPACE (OR GRID) CONSERVATION LAW (SCL):

$$\frac{\partial}{\partial t}\int_V dV - \int_S \boldsymbol{v}_b \cdot \boldsymbol{n} dS = 0$$

- Since solutions from previous time steps are not needed to compute surface and volume integrals, the grid can not only move but may also change its topology, i.e. both the number of CVs and their shape can change from one time step to another.

- By definition, it is impossible to pre-define mesh motion a-priori: boundary conditions are needed!
  - 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.

Dynamic Mesh Handling in OpenFOAM

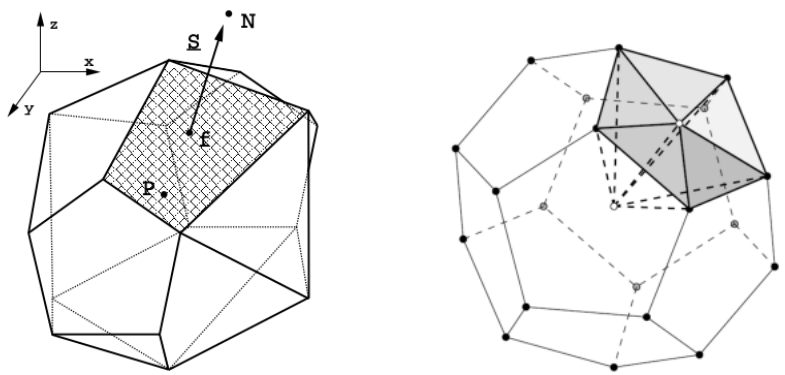
Polyhedral Mesh Support

Historically, CFD meshes use shape-based support: hexahedron, pyramid, prism, wedge, tetrahedron etc, defined in terms of vertices

... but OpenFOAM solvers is written using face addressing.

Main features:

- 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.



source: H. Jasak, PhD Thesis. Imperial College of London, UK.

Mesh Conversion and Manipulation

Basic Mesh Generation and Conversion

- `blockMesh` (Block-structured mesher with curved edges and flexible grading)
- `snappyHexMesh`: automatic split hex mesher. Refines and snaps to surface.

Mesh Converters ...

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

Mesh Manipulation Tools

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

... and Reverse Converters

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

Dynamic Meshes in OpenFOAM

Dynamic mesh handling is available and well established in OpenFOAM®:

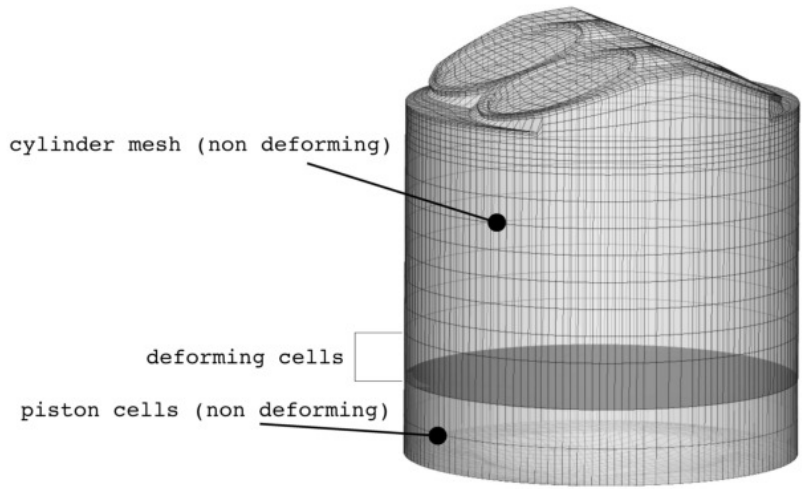
- deforming meshes;
- topological change support;
- complex dynamic mesh simulations;

and most recently:

- Native overset mesh
- Immersed Boundary Surface

layerAdditionRemoval

```
layerAdditionRemoval
{
    piston
    {
        layerFacesName pistonFaces;
        cellSetName   pistonCells;
        minLayerThickness 0.5e-3;
        maxLayerThickness 1.2e-3;
        fastThickness true;
    }
}
```



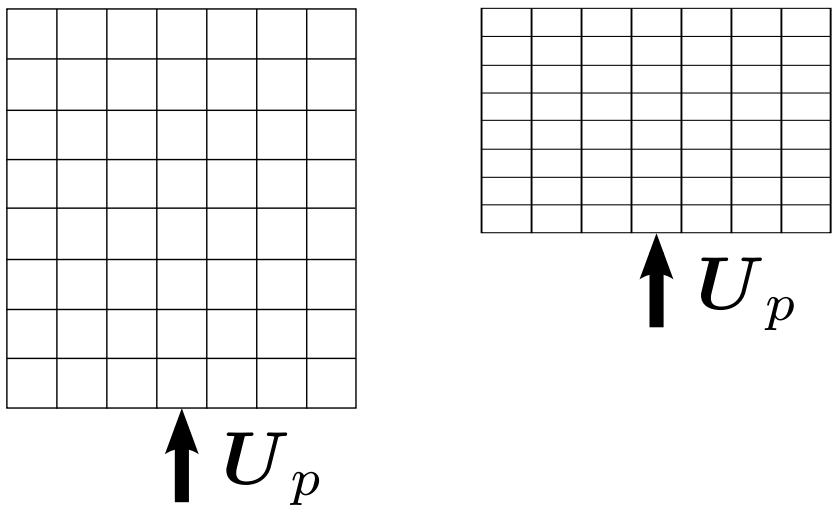
The `layerAdditionRemoval` mesh modifier:

- adds/removes dynamically one or more layers of cells as a consequence of a moving boundary (piston, valves)
- only one layer of cells undergoes actual deformation: **global mesh quality is preserved**
- Deforming layer does not have to lie on the moving boundary: constant aspect ratio of near-wall cells

Point stretching vs `layerAdditionRemoval`

During mesh motion will cell deformation/stretching:

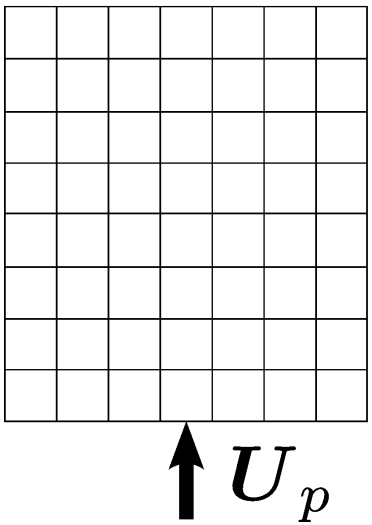
- variations in the mesh size Δx affect the discretization error and the numerical accuracy;
- if the cutoff length $\bar{\Delta} x$ is tied to Δx, mesh refinement will also induce a decrease in $\bar{\Delta}$ and make the subgrid model less influential on the results;
- how filtering behaves as cells increase in their mesh size is not trivial

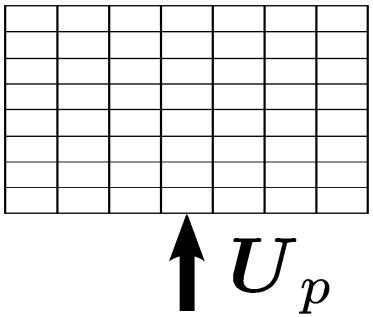


- A dynamic variation of the filter size requires to find an **error estimate** and a **bound** for the algorithm, in order to guarantee a sufficient resolution of the grid to resolve the main turbulent scales of the engine

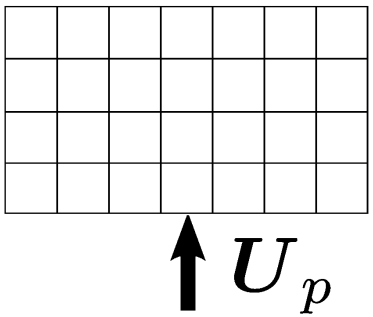
Point stretching vs `layerAdditionRemoval`

Mesh motion may change the filter size in **LES simulation**. Is there a way to keep the filter unchanged?





`pointStretching`

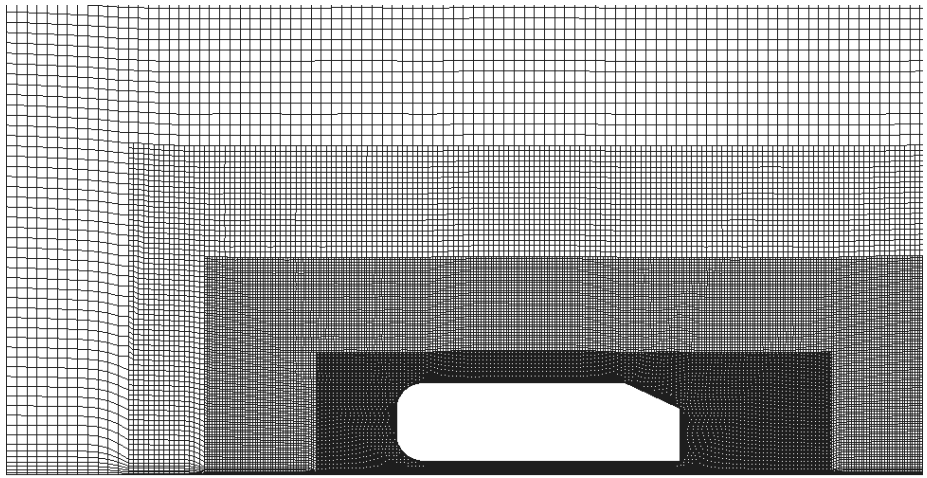


`layerAdditionRemoval`

FEATURES:

- variations in the mesh size Δx are limited and localized on a small number of cells: filter width almost constant despite the mesh is moving!
- overall number of cells varies during the engine cycle
- SGS model less influential on the results since it is not interacting with mesh motion

Non-conformal interfaces



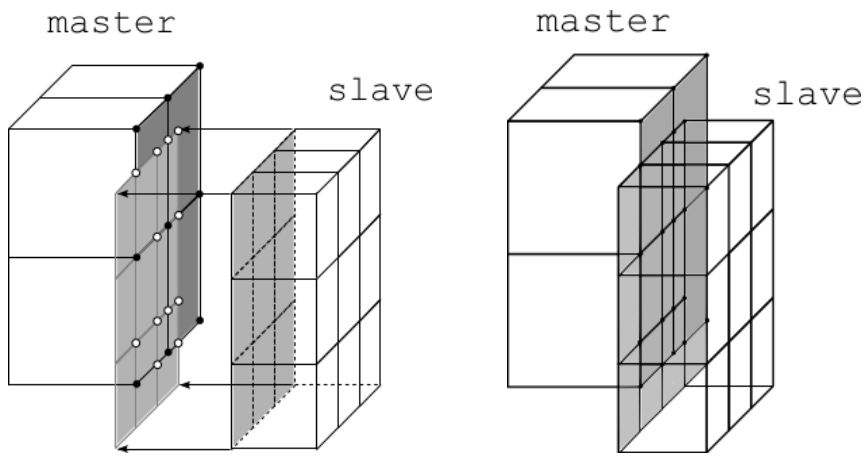
Sliding interface:

- can be used to create non-conformal interfaces between different mesh regions
- reduction of total cell number while mantaining fine mesh in critical regions (e.g. valve seat)
- hexahedral mapping of different regions can be decoupled
- cell quality preserved
- face fluxes are automatically handled

→ In OpenFOAM, `slidingInterface` is an example of DYNAMIC (moving) non conformal interface.

slidinginterface

`slidingInterface` allows for relative sliding of components.



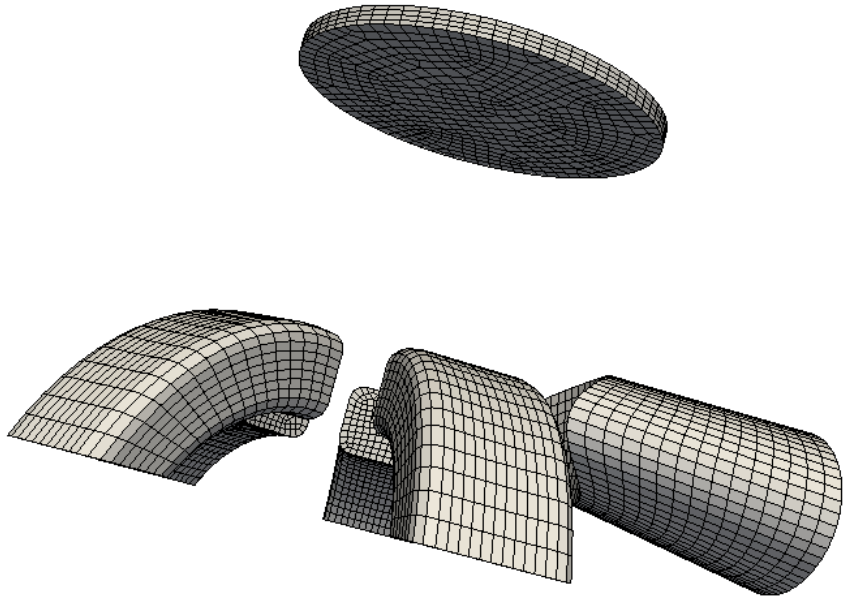
Source: SAE paper n. 2013-24-0027.

Definition:

- a master and slave patch, originally external to the mesh
- patches can be non-conformal interfaces, that may overlap either partially or fully
- points belonging to the “slave” side are projected onto the “master” patch
- allows uncovered master and slave faces to remain as boundaries
- the two different mesh regions are merged together

slidinginterface

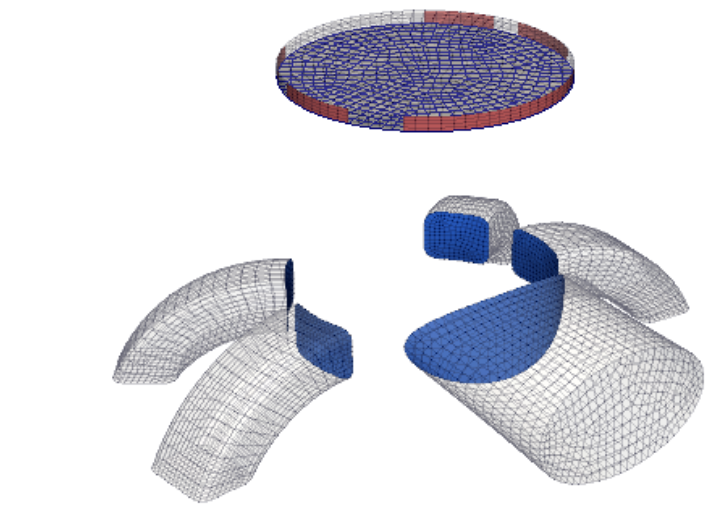
```
scavengingPorts
{
    interfaces
    XX
    (
      (cylPort01  inletPort01 )
      (cylPort02  inletPort02 )
      (cylPort03  inletPort03 )
      (cylPort04  inletPort04 )
      (cylPort05  inletPort05 )
      ...
      ...
      (cylPortXX  inletPortXX )
    )
}
```



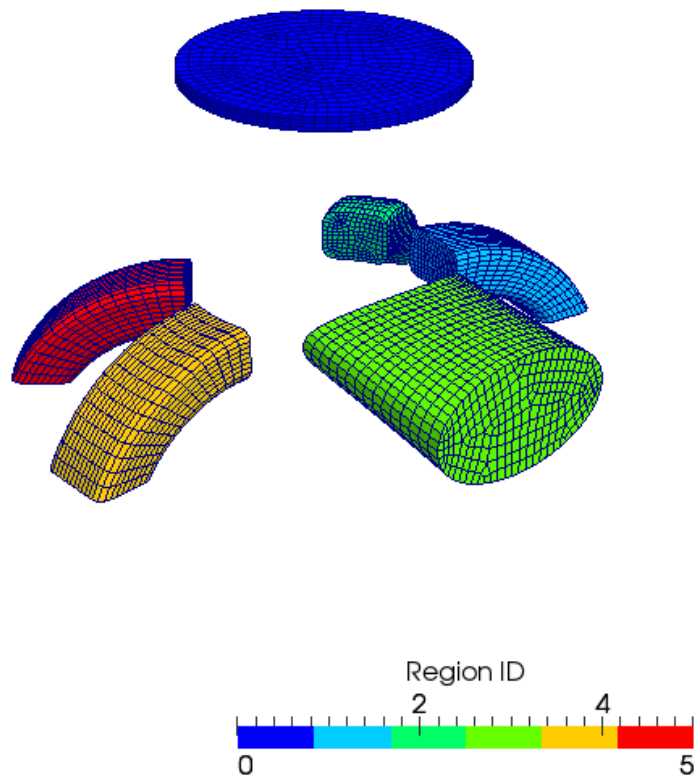
- To connect dynamically different mesh regions preserving the overall mesh quality and reducing the number of required meshes
- Definition consists in master and slave patches then the connection algorithms works automatically during mesh motion
- Polyhedral cells supported
- To be used in combination with dynamic mesh layering removed faces and points are stored in external objects

slidinginterface

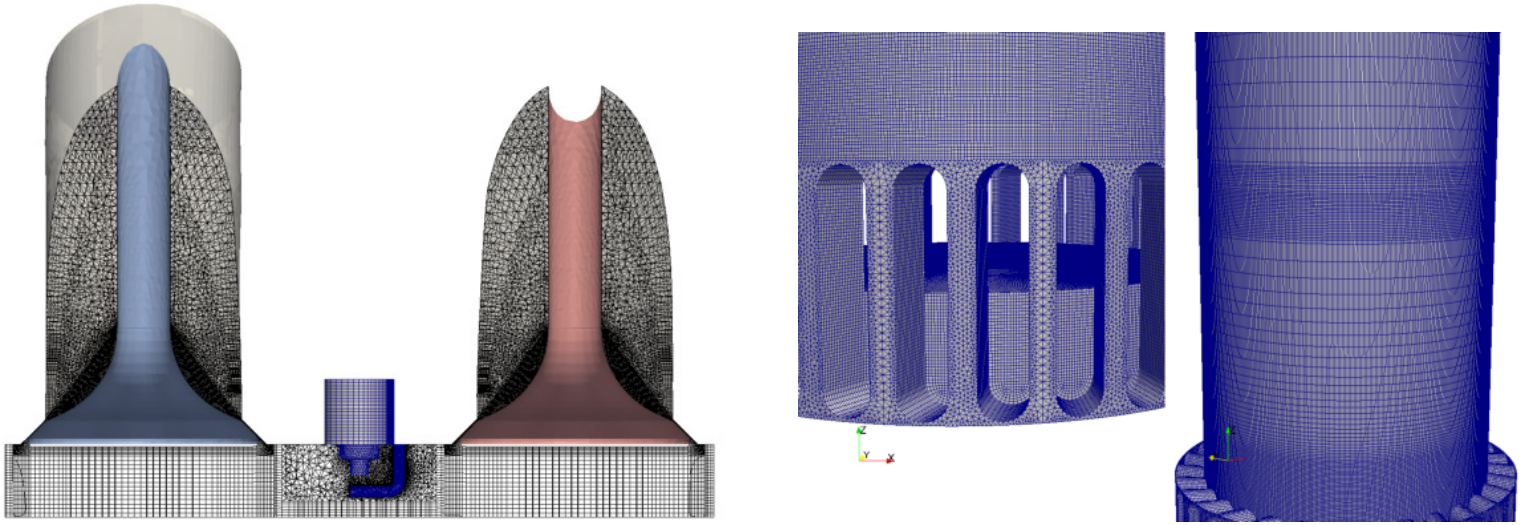
Interfaces definition



Fluid regions



Adaptive LES of Moving Geometries



WHAT IS NEEDED:

Automatic Mesh Motion with Parallel Topological Changes

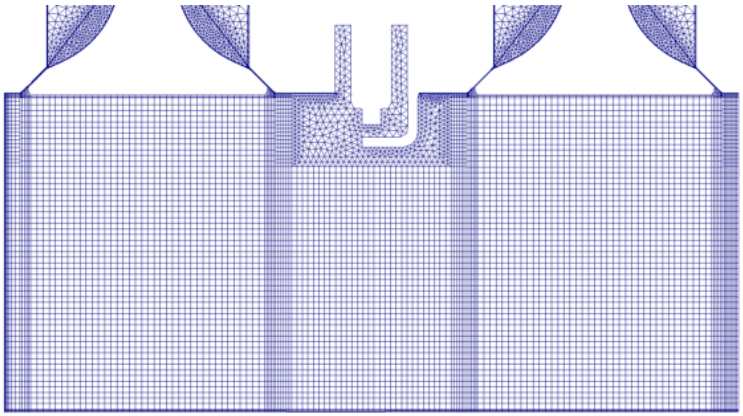
- `slidingInterface` to handle non-conformal interfaces
- `layerAdditionRemoval` of multiple layers of cells
- `attachDetach` to simulate valve opening/closure event
- AMI (Arbitrary Multigrid Interfaces)

Solver

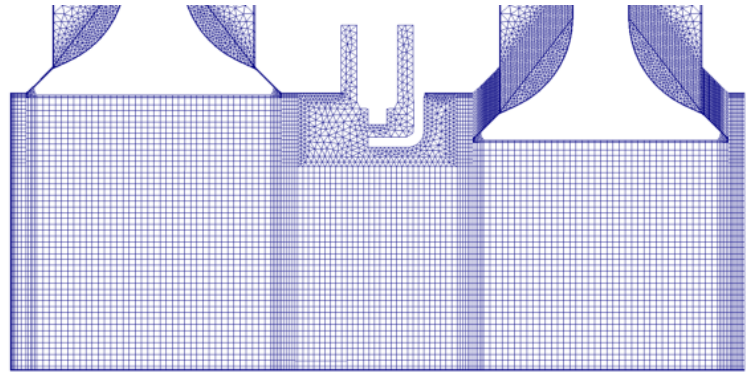
- `motionSolver` for topologically changing meshes
- a solver supporting topologically changing (moving) meshes

Examples of application

Large-bore Diesel Engines



Time: -66 CA-deg ATDCE

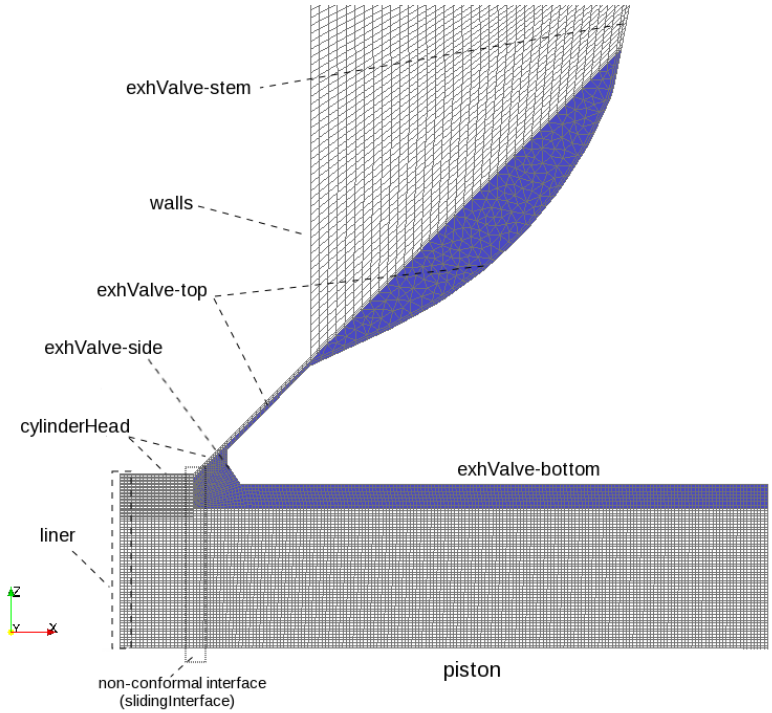


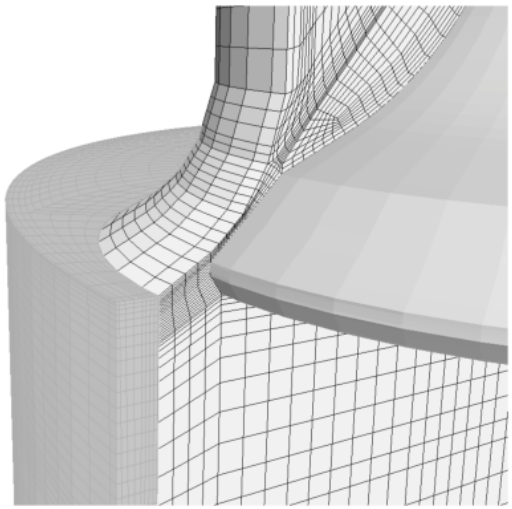
Time: 302 CA-deg ATDCE

Dynamic addition/removal of cell layers during:

- *piston motion*: cell layer over the piston head is removed/added during compression/expansion;
- *valve motion*: cells are added/removed both in the valve seat and on the valve bottom;
- grids at different time steps differs only for layers of hexahedral cells;
- initial *skewness* and *non-orthogonality* preserved during the whole engine cycle.

Non-conformal moving grids + topoChanges





- `cellSet` rigidly moving according to a prescribed motion law
- `layerAdditionRemoval` on valve top/bottom and on piston head
- `AMI/ACMI/slidingInterface`
- `attachDetach` of boundaries for valve opening/closure
- point deformation by a `fvMotionSolver`

Immersed Boundary Method

Immersed Boudary Method

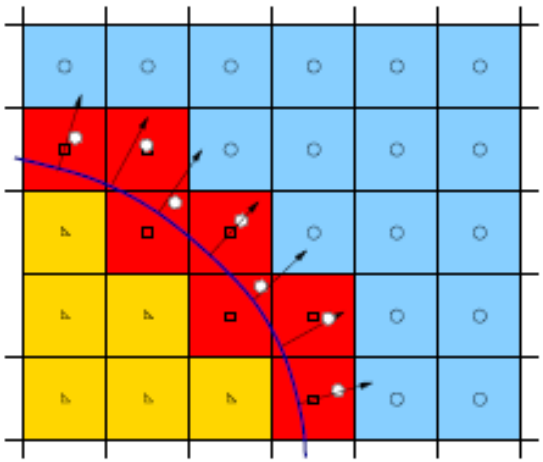
Immersed Boundary Surface

- 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!**

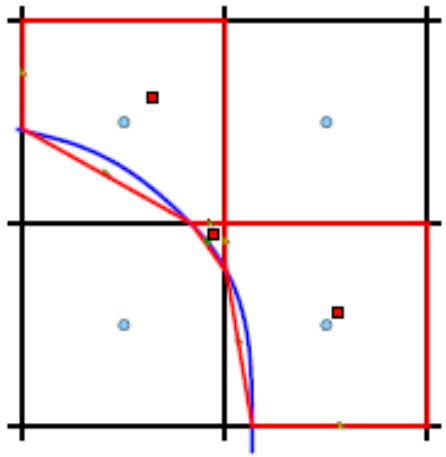
Advantages and Disadvantages

- 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 Boudary Method



Fluid cells: untouched
Solid cells: deactivated
IBS: intersected cells
Adjusted IBS centres



Background cell
Corrected face centre
Corrected cell centre
Immersed face centre

source: Prof. H. Jasak

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

Thank you for your attention!

contact: federico.piscaglia@polimi.it