A. Montorfano, F. Piscaglia, A. Onorati
An Extension of the Dynamic Mesh Handling with Topological Changes for Les of ICE in OpenFOAM
Paper presented at: SAE 2015 World Congress and Exhibition, Detroit, MI, USA, 21-23 April 2015, p. 1-15, 2015-01-0384
doi:10.4271/2015-01-0384

2015-01-0384

An extension of the dynamic mesh handling with topological changes for LES of ICE in OpenFOAM®

A. Montorfano, F. Piscaglia, A. Onorati
Dipartimento di Energia, Politecnico di Milano, Italy

Abstract

The paper focuses on the development of a mesh moving method based on non-conformal topologically changing grids applied to the simulation of IC engines, where the prescribed motion of piston and valves is accomplished by rigidly translating the sub-domain representing the moving component. With respect to authors previous work, a more robust and efficient algorithm to handle the connectivity of non-conformal interfaces and a mesh-motion solver supporting multiple layer addition/removal of cells, to decouple the time-step constraints of the mesh motion and of the fluid dynamics, has been implemented as a C++ library to extend the already existing classes for dynamic mesh handling of the finite-volume, open-source CFD code OpenFOAM®. Other new features include automatic decomposition of large multiple region domains to preserve processors load balance with topological changes for parallel computations and additional tools for automatic preprocessing and case setup. Finally, a transient solver for compressible viscous flows based on the transient SIMPLE algorithm has been implemented in order to enhance conservation of mass and energy for domains sliding over dynamically attached/detached boundaries. The advantages are significant: mesh changes in terms of topology and deformation are fully managed by the mesh motion solver without remeshing, with a consequent reduction of the overall simulation time. Most important, the method allows to preserve the quality of the mesh initially defined by the user (skewness, non-orthogonality and aspect ratio) during the whole engine cycle, favoring a faster convergence of the solver and a very accurate fluid-dynamic solution. Used in conjunction with LES turbulence modeling, the method allows to decouple mesh motion by LES filter operation, since the filter width is kept constant during the entire cycle. Validation tests have been performed on the full-cycle simulation of a Transparent Combustion Chamber (TCC) engine, whose experimental data are available through the Engine Combustion Network database (ECN). The implementation of the described methodology is absolutely general, it works on any number of processors and it can be applied to any application where moving parts and non-conformal interfaces are involved.

Introduction

The original aim of this work was to implement a parallel, fully automatic mesh motion strategy for Large Eddy Simulation (LES) of IC engines, where the LES filter operation could be fully independent by the mesh changes due to the point motion. The computational tool used is the open-source CFD software OpenFOAM® (in the version released by OpenCFD®), which has been extended by the authors with SGS models, boundary conditions, pre- and post-processing applications to perform LES of ICE [1, 2, 3, 4, 5, 6]. The implemented code was used to simulate the complex unsteady flow features in a engine-like geometry, consisting of a flat-top cylinder head with a fixed, axis-centered valve and a low-speed piston. A purely hexahedral mesh having about 4.6 million elements was used and mesh motion was based on a point-stretching concept [1]. The followed procedure proved to be accurate both in the prediction of average quantities and turbulence dynamics; on the other hand, when cell stretching (as well as remeshing) is used for mesh motion, cell volume and shape change. In particular, during cylinder compression:

- a reduction of the cell sizes Δx leads to a lower discretization error, yielding an improved numerical accuracy;

- if the cutoff length $\bar{\Delta}$ is tied to Δx, mesh refinement will also induce a decrease in $\bar{\Delta}$ and make the LES SGS model less influential on the results, since a larger part of the exact solution is directly captured.

As stated in [7], a dynamic reduction of the LES 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. Since the geometry of the engine is dynamically changing and the projection, the discretization and the modeling error are present at the same time, it is difficult to determine whether resolution of the turbulent scales improves with the reduction of the filter size during compression. At the same time, it is quite natural to think that the solution at a time step is correlated to the solution of the earlier time step; it is then

important that the mesh at the bottom dead center (BDC) before starting compression has a sufficient resolution to capture the main turbulent scales. This issue is even bigger during the expansion phase, as cells are stretched, since the grid resolution decreases unconditionally; to have sufficient resolution at the BDC, the resulting mesh at the top dead center (TDC) must be extremely fine, with strong constraints on the time step due to the CFL criterion. If remapping over multiple meshes between different time steps is adopted, the change of the LES filter size is even more difficult to handle and it is impossible to determine any criterion for error estimation, that will result uncontrolled. Hence, the choice of the pre-processing (mesh generation) and of the mesh motion strategy represents a crucial aspect for solver convergence and accuracy.

There is a wide range of examples about mesh handling of IC engine geometries [8, 9, 10, 11, 12, 13, 14] in the existing literature and also from the same authors' research group [15]. Most of these strategies use multiple grids, that are sometimes generated before the fluid dynamics is solved or automatically during the calculation. Despite algorithms for automatic mesh generation were significantly improved through the years and they represent reliable tools for massive CFD simulations, automatic generation of high quality grids suitable for LES still remains a difficult task to accomplish: any even small loss of quality in the grid is paid in LES in terms of an increased numerical viscosity, which affects the quality of the results and the speed of convergence of the solver. Also, mesh quality has direct influence on the stability of the solution, unless strongly limited numerical schemes are used [16, 17].
Among all the possible approaches, a strategy based on a wide use of so called *topology modifiers*, namely:

- `slidingInterface`, to connect dynamically different mesh regions through non conformal interfaces;
- `layerAdditionRemoval`, dynamic addition/removal of layers of hexahedral cells during piston and valve motion;
- `attachDetach` of boundaries, to model the valve opening and closure event;

has been chosen. During the simulation, a large part of mesh points remains fixed while boundaries (piston, valves) move; almost no cell deformation and remeshing are present, therefore mesh quality away from the boundaries is fully preserved and it can still be controlled near moving walls. The user is only asked to provide a single mesh with the piston at Top Dead Center (TDC), which can be generated both with commercial [**?**, 18] and by open-source grid generators (i.e. `snappyHexMesh`, [19]); an ad-hoc designed algorithm calculates the automatic mesh motion for the whole engine cycle, thus the case setup results simplified. The advantages of such a technique are multiple. First, this approach allows to preserve the initial mesh quality (skewness, non-orthogonality and aspect ratio) during the whole engine cycle (or, more in general, during all the simulation), since grids at different time steps differs only for layers of fully orthogonal hexahedral cells. As a consequence, in LES it is possible to keep the filter cell size unchanged during the whole engine cycle, favoring a very fast convergence of the solution over a multi-block grid. Finally, in terms of computational time, the technique is very efficient since the mesh changes are triggered only locally and the global morphology of the mesh is not recalculated during the simulation. Also, a novel formulation of a transient solver for compressible viscous flows based on a merged PISO-SIMPLE algorithm is presented, in order to efficiently enforce conservation of interface fluxes for domains sliding over dynamically attached/detached boundaries. The construction, activation and interaction of the mesh modifiers with the FV mesh is handled by the newly implemented `topoManager` class, whose structure makes the use of topology modifiers transparent to the developer, as new point motion strategies based on topological changes are written; in this way, the user can set up a large number of different dynamic mesh cases very quickly with little or no programming effort. The implementation presented in this work is completely based on the mesh definition of the OpenFOAM® version released by ESI-OpenCFD® [19], hence it is different from others available on other distributions [20]. At the time this paper is written, the features implemented and described in this work are missing in the official release of the code available for download on the website [19]. Thanks to the general and flexible implementation of the class, the presented methodology can be applied to the simulation of two- and four-stroke engines with moving valves, as well as to many simulation involving mesh motion with topological changes. The Transparent Combustion Chamber (TCC) engine, whose geometry and experimental data are publicly available through the Engine Combustion Network (ECN) database [21], represents a very good test case to compare different CFD approaches for mesh management, compressible solvers and turbulence modeling.

Dynamic mesh strategy

The dynamic mesh handling presented relies on a combination of prescribed motion laws to apply to boundaries combined with the use of topological changes. Fig. 1 shows in detail how the topology modifiers are used for valve motion and how cellSets/faceSet must be set. For each moving object of the geometry (either piston or valve), the user must define:

- a list of cells (`cellSet`) that will be rigidly moved along a given direction with a prescribed velocity
- one or multiple lists of faces (`faceSets`), where dynamic layer addition/removal will be applied. For any given faceSet, the algorithm runtime calculates the average cell height of the cell layer (hexahedral or prismatic), to check whether layer addition and removal must be triggered. Cells in a deleted layer are merged into neighboring cells, with physical variables in the resulting cell being the volume-average of deleted and neighboring cells; cells of added layers have a constant thickness that is defined by a dictionary;
- one or multiple lists of faces (at least one per valve bank) where attach/detach of the boundaries is applied.

In Fig. 1, the blue area represents the set of cells of a valve that are moving according to a prescribed motion law (lift profile); cells belonging to this cellset can be of any kind (hex, tet, prism, hybrid) and they are surrounded in the upper part by a layer of hexahedral cells. Additionally, `slidingInterface` topology modifier [6, 22] is applied on the sides of the cellset, because the static mesh of the engine head and the sliding valve are connected by non-conformal hybrid interfaces. A point motion strategy, based on addition/removal of cell layers, is applied both in the cylinder and in the valve region where uniform grid velocity of the moving boundaries is also set to the layers of cells nearby, that act as a rigid body. The remaining mesh points are fixed in space as the moving boundary is displaced: as a consequence, the cell layer located between the moving and the fixed points undergo a deformation (stretching or compression) that changes its thickness. As the height becomes too high or too low, the deforming layer is split into two parts or it is merged with the one right above. Threshold thickness values for addition/removal of layers are specified by the user.

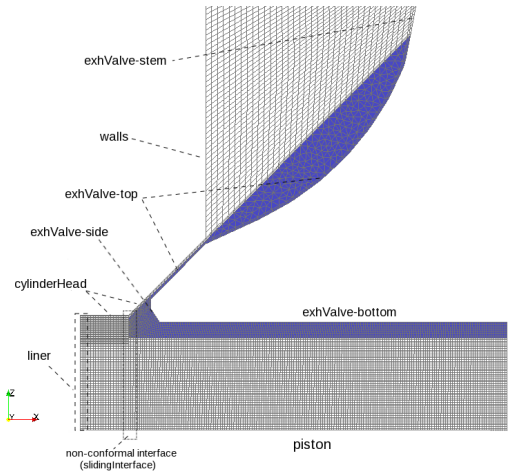


Figure 1: Schematic view of moving mesh strategy for an engine with vertical valves.

Layer addition/removal, a well-known strategy for mesh motion in IC engines [23], has been applied to piston and valves. The underlying assumption for the mesh motion technique presented in this work is that the cylinder (and valve) region must be composed by fully-structured hexahedra, or at least by extruded prisms; such a requirement is however easy to achieve, at least in proximity of the piston. On the other hand, there are no constraints related to the piston shape, for a suitable set of translating cells can be defined for any piston configuration (flat, convex, reentrant, ...). The algorithm distributed along with the base version of OpenFOAM® has been enhanced to make it more robust when inserted in complex meshes, and to improve its interaction with other topology modifiers. The insertion/deletion of layers on valve surfaces has to be limited to a restricted region around the valve axis, to reduce the number of cells affected by topological changes. Therefore, a discontinuity in mesh topology arises between the valve and the cylinder region, and it must be accounted for properly. This is done with the `slidingInterface` mesh modifier, that allows for a reversible coupling/decoupling of non-conformal mesh interfaces. Implementation of a robust and effective decoupling procedure for the `slidingInterface` mesh modifier required a thorough rewriting of the related classes in OpenFOAM® [19]. Finally, the overall topology of fluid volume changes as valves open/close, and a topological split has to be performed to account for this effect. Topological split is done by the attach-detach mesh modifier, that allows for creating a solid boundary between two fluid mesh regions. The point coordinates do not change during an attach/detach operation, so the mesh quality is preserved. Fig. 1 summarizes the type and number of topology modifiers for an engine with vertical valves. The increased complexity of the topological changes that have to be applied is balanced by the fact that the largest part of mesh points inside the cylinder do not move, as can be seen in Fig. 1.

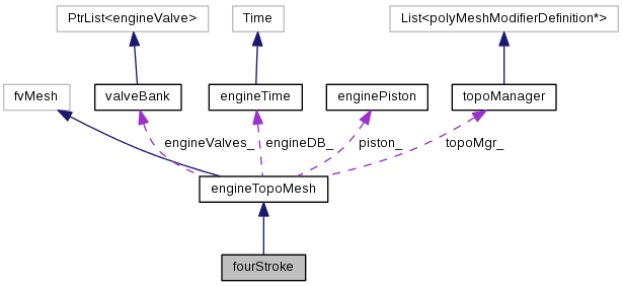


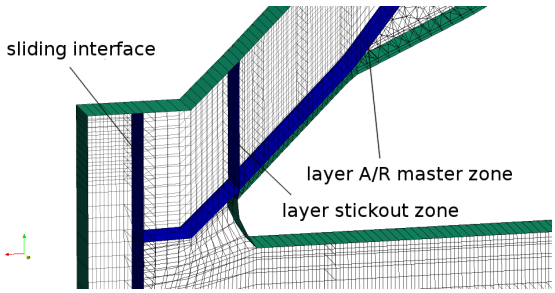
Figure 2: Collaboration diagram for the `engineTopoMesh` class.

Organization of the mesh class

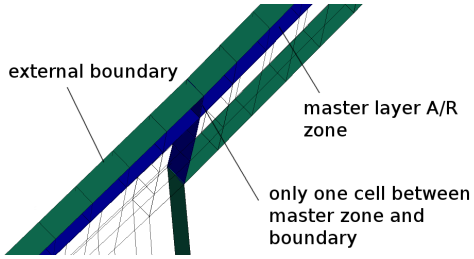
The required functionality for point motion and topological changes have been included in the extensions of the `dynamicMesh` class in the OpenFOAM® Technology and they are used by `engineTopoMesh`, a new class developed with expandability and modularity as primary requirements, that contains and handles all dynamic features required for the simulation of IC engines with topological changes. `engineTopoMesh` includes a set of subclasses, each one representing a different physical component (piston, valve, etc) as shown in Fig. 2. In turn, each component is supplied with all the information reqired by the mesh motion algorithm:

- *piston*: point motion is calculated according to crankshaft speed, connecting rod length and stroke. Layer addition/removal is performed on bowl surface;
- *valves*: point motion is calculated according to user-specified valve lift table. Layer addition/removal is performed on top and bottom surfaces. Sliding interfaces connect valve and cylinder regions. Attach-detach of boundaries simulates the valve opening and closing events, whenever the lift falls below a specified value.

All topology modifiers are referenced by a separate class



(a)



(b)

Figure 3: a) during layer addition/removal, newly inserted stick-out faces are dynamically included into the face zone, preserving mesh topology; b) layer removal is automatically deactivated when only one layer of cells exists between the master face zone and a mesh boundary.

`topoManager`, which handles their creation, activation and update. This allows a high flexibility of geometry definition, when different types of engines (e.g. two-stroke, canted valves, etc.) are to be simulated.

Layer addition/removal

Although the layer addition/removal feature has been included in OpenFOAM® since its earliest releases, some enhancements were required for its application to engine simulation. The main improvements regarded:

- enhancement of inter-processor communication for relevant data for dynamic mesh, that are exchanged between different processors in case of parallel computing. These quantities include mean layer thickness and addressing of the face sets used to add/remove layers;

- runtime update of face zones affected by layer A/R. If a face zone extends perpendicularly to the master layering face zone (even through it), newly inserted stick-out faces automatically are included into the face zone, thus preserving its topology (see Fig. 3-a);

- checking for boundary proximity: dynamic removal of cell layers is automatically deactivated, as only one layer of cells between the master face zone and a mesh boundary is left; in this way, deletion of boundary faces and subsequent topological inconsistency are prevented (see Fig. 3-b).

Sliding interface

A `slidingInterface` is a topology modifier that allows for the dynamic stitching and splitting of mesh regions with different mesh structures (see Fig. 4), so that a reversible, non-conformal interface can be created [24]. The sliding interface coupling procedure generates a seamless joint between the two involved surfaces, thus no particular numerical technique is required to solve the equations across the interface. In this aspect, `slidingInterface` differs from AMI (Arbitrary Mesh Interface technique [25, 26], already available in the standard distribution of the code), because in AMI fluid-dynamic coupling is achieved by interpolation of cell fluxes among two topologically separated mesh regions facing each other. The merging procedure of `slidingInterface` can be reversed without any loss of information to detach the interface, provided that all information about the coupling process has been appropriately stored. This part of the algorithm has been implemented by the authors and it has been already presented in previous works [6, 22, 27], according to the theory of Jasak [28].

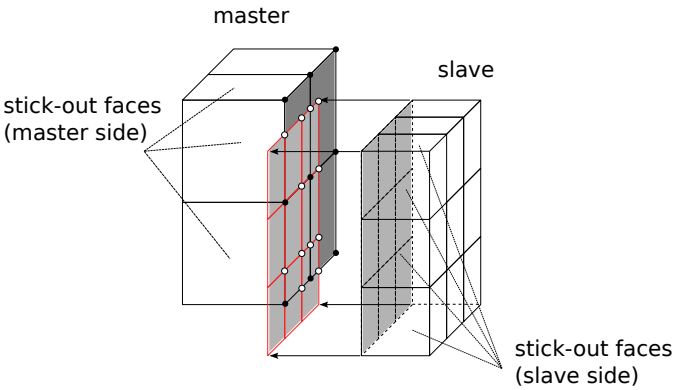


Figure 4: Point projection of slave patch (red) onto master. Solid dots are master points (which are retained), hollow circles are slave points added due to direct hits or intersections.

A further extension of the original theory [28] is proposed here, in order to allow `slidingInterface` to work with non-conformal interfaces presenting sharp corners (e.g. 90°) and to improve its robustness with complex geometries. In particular, the algorithm for the detection of the `stickout` faces over two arbitrary-shaped non conformal interfaces has been completely redesigned, to overcome some inherent shortcomings of the original implementation and to assign faces to the appropriate mesh region.
A stickout face shares one or more edges with the non-conformal interface; because of this, it is involved in the coupling procedure, as points are inserted or deleted on its edges (see Fig. 4). In the original formulation, detection of stickout faces is performed on a cell-face basis: starting from a face belonging to master (or slave) interface, the algorithm detects the owner cell and classifies the stick-out faces as those sharing an edge with that face. As a consequence, this procedure does have tight constraints about the shape of the interfaces to couple:

- the surface must be as smooth as possible (ideally speaking, planar);

- on both sides, all cells adjacent to the surface must be hexahedra or pyramids;

- an edge cannot belong to more than four faces.

In the original implementation, if the above requirements are not fulfilled (e.g. when there is an unstructured meshes on one side), the coupling algorithm might fail, since there can be stick-out faces other than those belonging to the master owner cell (see Fig. 5). To overcome this problem, a new procedure based on a point-face seeking has been developed for detection of stick-out faces.

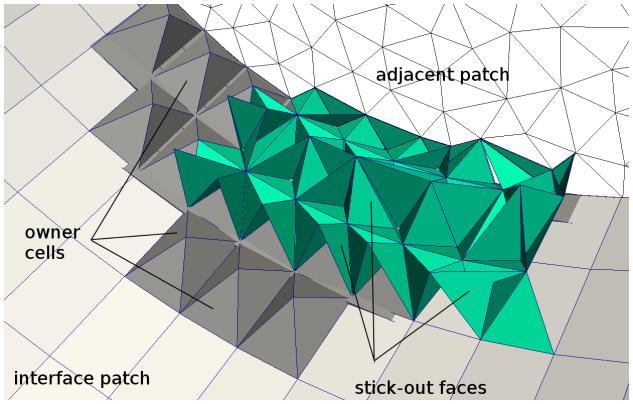


Figure 5: Stick-out faces of an unstructured mesh. They include also faces that does not belong to owner cells.

The algorithm consists of the following steps:

- a master (slave) face is selected;

- for all the points of the master (slave) face, all faces sharing a point with the selected master (or slave) face and are classified as stick-out faces.

The resulting novel algorithm for detection of stick-out faces now has only one tight requirement: no points can be shared by the master and the slave side. For this reason, a pre-processing application (`splitSharedPoints`) to split possibly shared points has been developed by the authors, to fix meshes generated by mesh generators (either commercial or open-source) that do no fulfill the above constraint. The new algorithm for detecting stick-out faces has proven to be indispensable when complex non-conformal interfaces need to be stitched.

An example of use of the algorithm is shown in Fig. 6: the cylinder and the spark plug region of the TCC engine are merged through non-conformal interfaces: in this way, a perfectly structured mesh (cylinder region) can be merged with a highly refined tetrahedral grid (spark plug), in order to have a very high quality mesh.

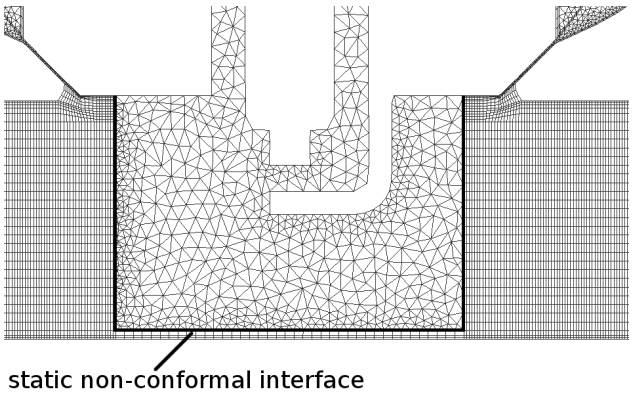


Figure 6: Example of a complex non-conformal interface achieved by a sliding interface.

Attach/detach of boundaries

Attach/detach mesh modifier is applied to simulate the valve closure event and it consists in a reversible interface between two conformal mesh regions. It is used to temporarily join or split different parts of the mesh starting from a prescribed and arbitrary set of internal faces (`detachFaces`), that will be used by the dynamic mesh solver to be transformed into boundary walls. In ICE simulation, the `attachDetach` C++ class separates the intake/exhaust ports from the combustion chamber at the valve closure (Fig. 7).

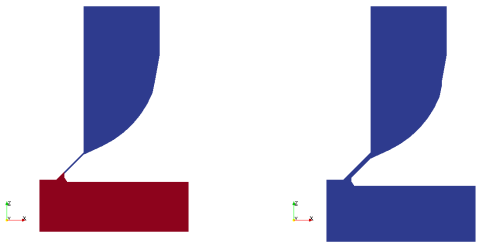


Figure 7: Attach/detach of boundaries: at valve closure, an internal faceSet is converted into two walls dividing the domain into different regions (a); boundaries are converted again into internal faces when the valve is open (b).

The original version of the class, as released in the standard release of OpenFOAM®, calculates the face matching between the two sides by implying that the point ordering is the same; since this seldom happens in complex meshes (like in IC engines), the original class has been extended to calculate the face matching on the basis of point projection. The followed strategy results to be very similar to the one employed for the sliding interface algorithm [6, 22, 27]. Despite the new version of algorithm results to be a slightly slower, it has proven to be more robust and it succeeded in all test cases performed so far.

Variable topology-driven time-stepping

Topological changes, being triggered in dependency of the current mesh status, pose some limits on the maximum time step size needed to guarantee topological consistency. Using too a large temporal integration step may cause the mesh handling algorithm to skip the point when a topological changes would be triggered, hence leading to a wrong mesh configuration. A typical example is when the piston upward displacement in a single time step is larger than the threshold value for dynamic layer removal. To avoid problematic situations and to ensure dynamic mesh consistency, adaptive topology-driven time stepping has been implemented. The expected displacements for all the moving components (piston, valves) $\Delta z'$ are computed *before* they are actually executed. As an example, for a piston moving by a velocity u_p during the compression phase:

$$\Delta z'_{\text{piston}} = u_p \cdot \Delta t \tag{1}$$

If the predicted displacement is larger than the average height of the cell layer to remove:

$$\Delta z'_{\text{piston}} \geq \Delta z^{\text{max}}_{\text{layer}} \tag{2}$$

then the time step is recalculated as:

$$\Delta t_{\text{lim}} = \Delta z^{\text{max}}_{\text{layer}} / u_p \tag{3}$$

In Eq. (2), $\Delta z^{\text{max}}_{\text{layer}}$ is the layer removal threshold and Δt_{lim} the maximum allowed time step. The multi-stepping on topology modifiers implies a proper handling of the contributions of the single step to the grid velocity vector on the conservation equations for scalar quantities, as it will be explained further.

Compressible solver for dynamic mesh

A newly developed compressible dynamic solver used for the simulation is `coldTopoEngineFoam`, which is an extension of the already existing transient solver for compressible flows on dynamic meshes, with some modifications to deal with multiple topological changes and improve convergency. The fundamental equations governing compressible flow inside a moving domain [29] are written as:

$$\frac{\partial}{\partial t}\int_{V(t)} \rho \mathrm{d}V + \int_{S(t)} \rho\left(\boldsymbol{u}-\boldsymbol{u_b}\right)\cdot\boldsymbol{n}\,\mathrm{d}S = 0 \tag{4}$$

$$\frac{\partial}{\partial t}\int_{V(t)} \rho\boldsymbol{u} \mathrm{d}V + \int_{S(t)} \rho\boldsymbol{u}\left(\boldsymbol{u}-\boldsymbol{u_b}\right)\cdot\boldsymbol{n}\,\mathrm{d}S = \int_{V(t)} \boldsymbol{f}\mathrm{d}V \tag{5}$$

$$\begin{aligned}\frac{\partial}{\partial t}\int_{V(t)} \rho(h+K) \mathrm{d}V + \int_{S(t)} \rho(h+K)\left(\boldsymbol{u}-\boldsymbol{u_b}\right)\cdot\boldsymbol{n}\,\mathrm{d}S \\ -\int_{V(t)}\frac{\partial p}{\partial t} - \int_{S(t)} \alpha\nabla h\cdot\boldsymbol{n}\,\mathrm{d}S = \int_{V(t)} q\,\mathrm{d}V\end{aligned} \tag{6}$$

where $\boldsymbol{u}$ and ρ are the fluid velocity and density, h is the sensible enthalpy, K is the kinetic energy and $\boldsymbol{u}_b$ are the velocities the control volume boundaries move with. Despite the formulation with moving boundaries looks very similar to the formulation with a non-moving domain, solution of Eq. (4), (5) and (6) requires particular care because of the term including the relative advection velocity $\boldsymbol{u}-\boldsymbol{u}_b$. In fact, when they are discretized in a FV framework, advection velocities are substituted by cell face fluxes ϕ; similarly, boundary velocities $\boldsymbol{u}_b$ are replaced by cell face fluxes originated by points motion, ϕ_M. Next paragraphs will focus on methods for calculating ϕ_M in case of motion with or without topological changes, and on a revised version of the solver algorithm for the computation of compressible flows with topological changes.

Enforcement of continuity without topological changes

As shown in [29], a cell mass source can appear in the mass conservation equation as cell faces move, even if mesh fluxes are inserted in the discretized equations:

$$\Delta\dot{m} = \frac{\rho\Delta V}{\Delta t} \tag{7}$$

To avoid this spurious source term (7), one must guarantee that the *Space Conservation Law* (SCL) is fulfilled [30, 31]. SCL can be regarded as a continuity equation in case of a zero fluid velocity:

$$\frac{\mathrm{d}}{dt}\int_V \mathrm{d}V - \int_S \boldsymbol{u_b}\cdot\boldsymbol{n}\mathrm{d}S = 0 \tag{8}$$

Discretization of Eq. (8) depends on the chosen temporal integration scheme and it allows for calculating the mesh motion flux (ϕ_M) on the basis of the swept volume $\dot{V}_b$; in the simplest case of Euler implicit integration, the mesh motion flux can be calculated as:

$$\phi_M = (\boldsymbol{u}_b\cdot\boldsymbol{n})_f S_f = \dot{V}_f \tag{9}$$

where $\dot{V}_f = \delta V/\Delta t$ is the volume swept by a cell face in a single time step. In case of a higher order scheme, a different discrete equation for ϕ_M must be used. In OpenFOAM®, the calculation of ϕ_M according to the time discretization scheme is done by the virtual function `fvc::meshPhi()` by the run-time selection of temporal discretization scheme. For a cell face with a generic shape, the swept volume is calculated as follows: first, the face is decomposed into several triangles, one for each edge, that have as common vertex the face centroid; then, the swept volume is calculated for each triangle, as the difference between its new point coordinates T and the old ones T^o:

$$\dot{V}_f = f(T - T^o) \tag{10}$$

Since a face is stored as a list of point IDs, and not as a list of point coordinates, Eq. (10) does hold as long as every point maintains its own ID during the mesh change (i.e., in the case of point motion without topological changes). When topological changes are triggered, points are renumbered and there is no correspondence between old and new point IDs, so the correlation between T and T^o is no longer valid. In this case, a different procedure has to be applied, as outlined in the following paragraph.

Enforcement of continuity with topological changes

Handling of mesh fluxes in case of topological changes is done in different ways, depending on whether a cell face is directly affected by a topology change (i.e. it is added or deleted), or it is modified by point addition/removal, or it simply changes its shape and not its definition. In the first case, if a face is added during a topology modification (Fig. 8-a), its mesh flux must be zero. This is easily ensured by explicitly setting the value of ϕ_M on newly created faces. On the other hand, if a face is removed (Fig. 8-b), its mesh flux does no longer exists. Continuity is thus enforced by solving a modified Poisson equation, as it will be explained later.

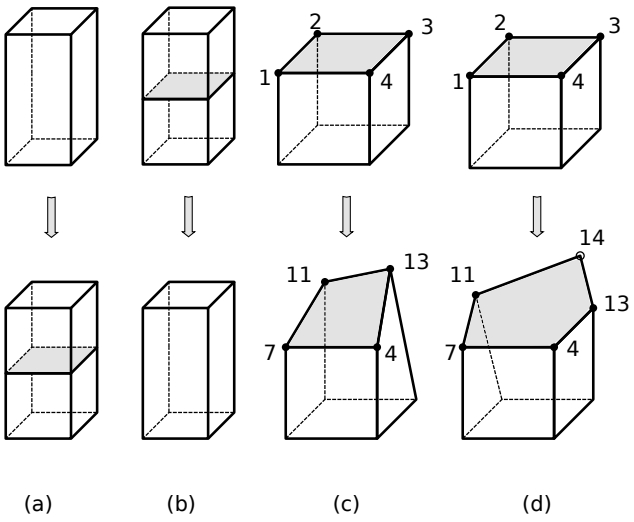


Figure 8: Different cases of topological changes. a) Face insertion; b) Face removal; c) Face does not topologically change, but its vertices get renumbered; d) Face is transformed and a new vertex is inserted.

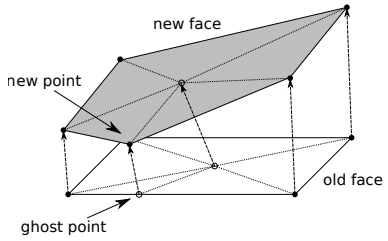


Figure 9: Handling of faces with inserted points. The new point is projected onto the counterpart edge originating a 'ghost' point, and the edge is split. Now both faces can be decomposed in the same number of triangles. In case the new face has less point than the old one, the ghost point is added on the new face instead.

If points are renumbered as a consequence of a topological change, but the faces they belong to do not undergo any substantial modification (Fig. 8-c), Eq. (10) can still be applied. However, face triangle decomposition T^o must be rewritten using the new point IDs, that are deduced using a point-to-point map generated during the topological change. Finally, if a face mantains its definition (i.e. it still exists after the topological change), but points are added or removed (as in Fig. 8-d), one must ensure that the new face is decomposed in the same number of triangles as the old one. This is achieved by adding vertexes on either the new or the old face, depending on whether the new face has less or more points than the original, as shown in Fig. 9. These 'ghost' points are inserted by splitting an existing edge, so that the global shape of the face remains unchanged. The coordinates of the ghost point is the result of a projection of the corresponding vertex on the old (or new) face.

Finally, before solving the governing equations (4) and (5) on the updated mesh, one must ensure that old values of $\boldsymbol{u}$, p and ρ still satisfy continuity when remapped onto the new grid [16, 32]. In fact, the old velocity field $\boldsymbol{u}_n^n = \boldsymbol{u}(\boldsymbol{x}^n, t^n)$ might not be compliant with the continuity equation (Eq. (4)) once resampled onto the new mesh. Therefore, a modified form of Poisson equation (Eq. (11)) is solved for a pressure corrector p_{corr}:

$$\nabla^2 p_{\text{corr}} + \frac{1}{\Delta t} \nabla \cdot \left[\rho(\boldsymbol{x}^{n+1}, t^n)\boldsymbol{u}(\boldsymbol{x}^{n+1}, t^n)\right] = 0 \qquad (11)$$

where $\boldsymbol{u}(\boldsymbol{x}^{n+1}, t^n)$ and $\rho(\boldsymbol{x}^{n+1}, t^n)$ denote the velocity and density fields computed at the previous timestep but remapped onto the new mesh. The differential equation (11) must be completed with appropriate boundary conditions. On solid walls they have to be of Neumann type ($\partial p_{\text{corr}}/\partial n = 0$), whereas on permeable walls a Dirichlet boundary condition is applied ($p_{\text{corr}} = 0$). The pressure correction problem assumes therefore the following form:

$$\begin{cases} \nabla^2 p_{\text{corr}} + \dfrac{1}{\Delta t} \nabla \cdot \left[\rho(\boldsymbol{x}^{n+1}, t^n)\boldsymbol{u}(\boldsymbol{x}^{n+1}, t^n)\right] = 0 \\ \dfrac{\partial p_{\text{corr}}}{\partial n} = 0 \qquad \text{on solid boundaries} \end{cases} \qquad (12)$$

During intake and exhaust strokes there is at least one open boundary, thus Eq. (12) usually poses no concerns upon the existence and uniqueness of its solution. On the other hand, a difficulty arises when both valves are closed: in this case, Eq. (12) is solved separately for each subdomain (cylinder, intake, exhaust). The cylinder region, however, is delimited exclusively by solid walls, thus no Dirichlet-type boundary conditions are applied and the elliptic problem has no unique solution. To overcome this intrinsic difficulty, a reference value of p_{corr} is imposed at an arbitrary location of the domain:

$$\begin{cases} \nabla^2 p_{\text{corr}} + \dfrac{1}{\Delta t} \nabla \cdot \left[\rho(\boldsymbol{x}^{n+1}, t^n)\boldsymbol{u}(\boldsymbol{x}^{n+1}, t^n)\right] = 0 \\ \dfrac{\partial p_{\text{corr}}}{\partial n} = 0 \qquad \text{on solid walls} \\ p_{\text{corr}} = 0 \end{cases} \qquad (13)$$

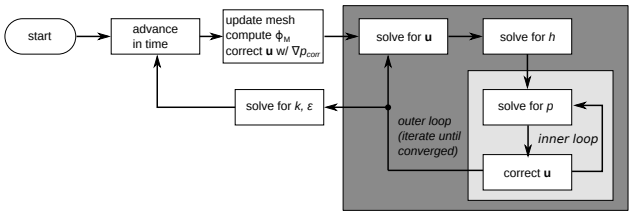
Once solved for p_{corr}, its gradient is then used to update the velocity:

$$\boldsymbol{u}_{n+1}^{n} = \boldsymbol{u}_{n}^{n} - \sum \frac{1}{A_p} \nabla p_{\text{corr}} \tag{14}$$

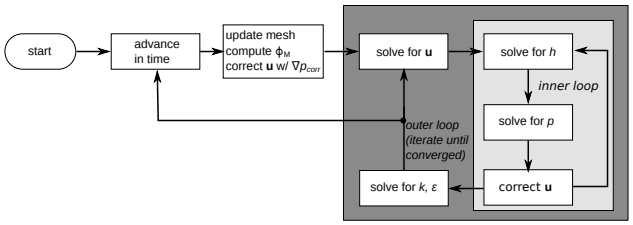
Eqs. (11) and (14) are solved iteratively any time the mesh changes, until convergence on pressure is reached. Tests made on a simplified geometry have shown that the (relative) continuity error can be kept below 10^{-8} [27]. Pressure correction applied after topological changes (either `layerAdditionRemoval`, or `slidingInterface` or `attachDetach`) leads to an improvement in the solver performance, as it is discussed in the following paragraph.

Enhanced pressure-energy coupling

In OpenFOAM®, the base transient solver for compressible viscous flows is based on a merged PISO-SIMPLE algorithm (PIMPLE), which is represented in Fig. 10-a. The outer loop is analogous to the pressure-correction algorithm of the steady SIMPLE solver, whereas the inner loop solves iteratively the equation of pressure. At the beginning of each timestep, the mesh is updated according to the piston and valve motion. As the mesh is updated, face fluxes are recalculated including the effect of the mesh motion, as described in the previous paragraph; finally, a remapping of the newly calculated quantities is performed, the velocity correction equation (11) is solved and the iteration for the solution of the governing equations can start.



(a) Old algorithm



(b) New algorithm

Figure 10: Original formulation (a) and the novel implementation (b) of the formulation of the PIMPLE algorithm. Also the equation for flux correction (performed after the update of the mesh) is different between the two versions.

According to the original formulation of the PIMPLE algorithm (Fig. 10-a), the inner loop is rarely executed more than once (in transient-SIMPLE mode), since the outer loop is deemed sufficient to achieve pressure-velocity coupling. The user can however choose to perform only one outer iteration: in this case two inner iterations are mandatory. This is the PISO algorithm, which is limited by the Courant-Friedrichs-Lewy criterion ($CFL \leq 1$) [33]. Under-relaxation must be applied on solved quantities to avoid numerical overshoots during the outer iteration. Values of relaxation factors range usually from 0.7 (for velocity) to 0.3 (for pressure). As shown in Fig. 10-b, the PIMPLE solver has been modified to achieve a stronger coupling between pressure and energy: energy equation is now solved together with mass conservation into the inner loop to help global convergence (pressure and temperature are strongly linked in compressible flows). For the same reason, several iterations of the inner loop are performed for each outer iteration. Moreover, the solution of turbulence-related quantities (k and ε, k and ω, etc.) is done every outer iteration, to account for strong changes in velocity field that might occur inside the outer loop, especially during the first timesteps or at the opening and closure of the valves. Despite solving the two additional equations of turbulence for each outer loop increases the computational effort of the single outer iteration, it favors for a faster convergence of the solution. In Fig. 11 a sample convergence history within a single timestep at 90° CA is reported for the three-dimensional simulation of the TCC engine. The graph shows the initial values of the normalized residual on pressure equation versus the number of outer iterations for a single timestep. The new algorithm reaches a specified tolerance (say 10^{-4}) in nearly half the number of outer iterations with respect to the previous scheme. As a consequence, the wallclock time for

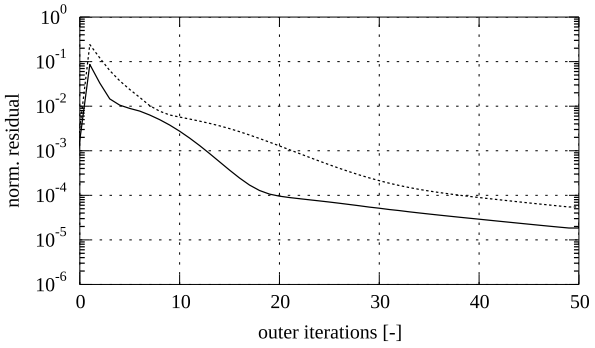


Figure 11: Numerical convergence of different implementations of the PIMPLE loop: comparison of pressure residuals between the old (- - -) and the new (—) algorithm versus the number of outer iterations within a single timestep.

the timestep results to be significantly lower. It is important to note that the walltime saving is not linear with the number of outer iterations: the outer iteration of the new solver is more expensive, since it includes at least two inner iterations. For the timestep studied in Fig. 11, the speedup is about 40%. Thanks to the stronger coupling between energy and pressure, high under-relaxation factors (up to 0.9) can be set, thus limiting the apparent overhead due to an increased number of inner iterations. Despite the modified solver is presented together with a specified mesh motion strategy, its formulation is fully general and compatible with any mesh motion strategy adopted.

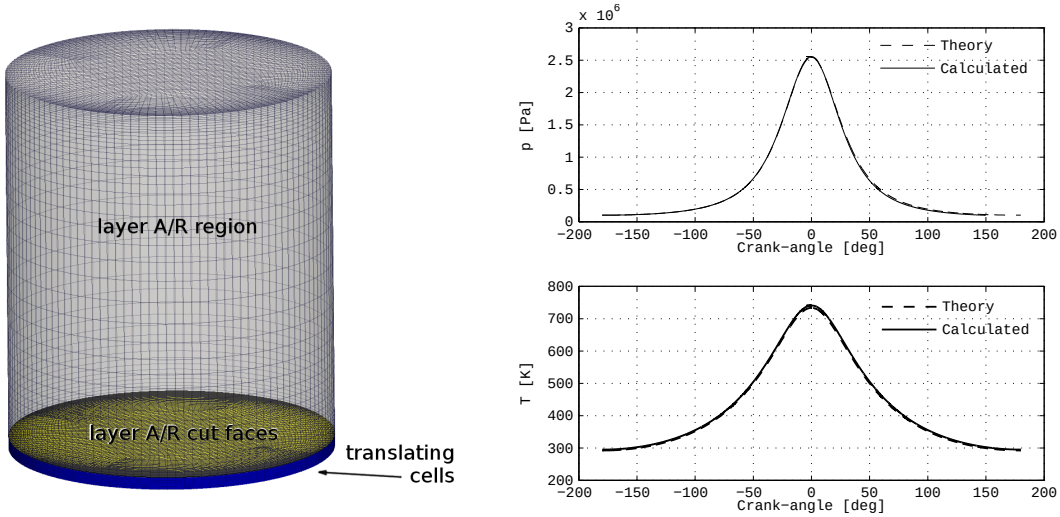


Figure 12: Left) Mesh setup for the basic test cases; six layers of cells attached to the bottom boundary move rigidly with it; Right) code validation: comparison between theoretical and simulated pressure and temperature during a single compression cycle. Top: pressure vs. crank-angle; bottom: absolute temperature vs. crank-angle.

Validation and testing

Conservation of physical properties: single cylinder test case

The basic validation of conservativeness of the new solver with respect to momentum, mass and energy has been done on a simple test-case represented by adiabatic compression within a closed volume with topological changes (dynamic mesh layering); the geometry is represented in Fig. 12. Cylinder diameter is 84 mm the height is 100 mm. The volume has been discretized by a fully structured hexahedral mesh with 225000 elements and the lower face has been imposed an harmonic motion with an amplitude of 90 mm and a frequency of 33.33 Hz; the geometric compression ratio is 10. Six layers of cells have been selected to move rigidly with the lower boundary, as shown in Fig. 12; `layerAdditionRemoval` is applied on the layer of cells just above, which is stretched during piston motion; these cells are removed during compression when their thickness is lower than a threshold value defined by the user (0.5 mm in the example); conversely, single layers of cells are added during expansion, as the cell thickness is higher of a certain threshold. Ambient temperature and pressure were set as initial conditions, since the flow field in the cylinder was initially at rest; walls were adiabatic. The simulation was carried out by using a standard $k-\epsilon$ RANS turbulence model, since the aim was to verify the solver performance (in terms of convergence and mass conservation across the cell region where addition or removal of cell layers occurred), rather than providing an accurate description of the flow field. Average values of pressure and temperature inside the cylinder were compared against the theoretical law of adiabatic compression for a perfect gas:

$$p\,V^k = \text{const} \tag{15}$$

where $k = c_p/c_v$ is the ratio of specific heats. Results are reported in Fig. 12: as it can be easily seen, there is almost no difference between the expected theoretical values and the ones obtained from the simulation; relative error is lower than 0.3% for pressure and lower than 0.5% for temperature. Total mass inside the cylinder is conserved with an error lower than 10^{-8}.

The TCC engine: case description, simulation setup and preliminary results

The Transparent Combustion Chamber optical engine configuration simulated in this section was set up at the University of Michigan [21] in order to gather a database of experimental data to be used to validate CFD models. The test configuration is characterized by a single-cylinder setup with a pancake-shaped head and two vertical valves, operated by a camshaft. Valve lift profiles are reported in Fig. 13. The engine is operating at motored conditions and intake and exhaust ducts are connected with plenums in order to damp pressure oscillations. All relevant engine data are reported in Tab. 1 [34].

Table 1: Geometrical features and valve timing for the TCC engine [21].

Bore	92 mm
Stroke	86 mm
Connecting rod length	234.95 mm
TDC clearance height	9.50 mm
Geometric compression ratio	10
Engine speed	800 rpm

The engine has optical access to the combustion chamber to allow for flow field measurements using non-intrusive techniques. In particular, Abraham *et al.* [34], used PIV to measure the instantaneous flow field inside the combustion chamber using a laser sheet to illuminate the seeded flow orthogonal to the X- axis and located on the cylinder mid section. In addition to detailed PIV, pressure transducers were installed at the valve ports to measure pressure pulses

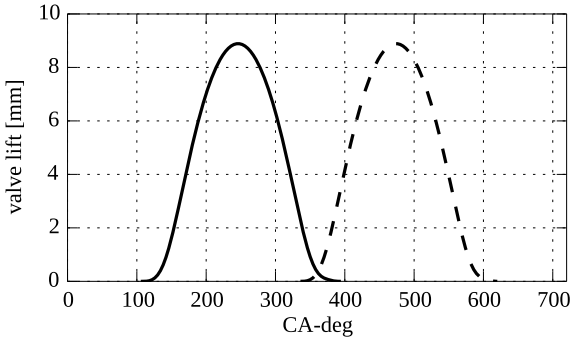


Figure 13: Valve lift diagram for the Transparent Combustion Chamber (TCC) engine. — exhaust valve; - - - intake valve.

due to the gas exchange process. The ambient conditions during the experimental tests were also monitored and their mean values are reported in Tab. 2 [35]. Flow field imaging from PIV was available inside the combustion chamber every 5° CA for 60 consecutive cycles. Datasets are divided in two groups: 'low' and 'high' resolution measurements. The former has a spatial resolution of 2.93 mm; the latter, near the spark plug, has a spatial resolution of 0.80 mm. Both series of samples were taken simultaneously.

Table 2: Relevant ambient conditions during experimental measurements (average values)

Cylinder surface temperature	44.9 °C
Air temperature at intake port	45.6 °C
Pressure at intake plenum inlet	95.0 kPa
Intake port pressure	94.6 kPa
Exhaust port pressure	101.5 kPa
Pressure at exhaust plenum outlet	101.4 kPa

To compare measured velocities with simulation results, experimental data sets have been ensemble-averaged by the authors to extract mean and RMS velocities. The point coordinates upon which PIV data are located have been used to construct a Delaunay triangulation; this allowed to generate a series of VTK files [36] that can be directly compared with the simulation results, as shown in Fig. 14 for CA = 55° ATDC. Fig. 14 shows also the approximate locations of PIV measurement windows for both 'high' and 'low' resolution data, that are plotted on the same image to have a more accurate snapshot of local velocity.

Automatic decomposition with topological changes and load balancing

In OpenFOAM® parallelism is implemented by the so-called domain decomposition technique: the whole mesh is divided into several sub-domains, each assigned to a separate process. Communication between sub-domains, that is carried out by specific boundary conditions based on the MPI protocol, ensures physical consistency disregarding the specific equations and models implemented in the solver. Insertion of topology modifiers in a decomposed

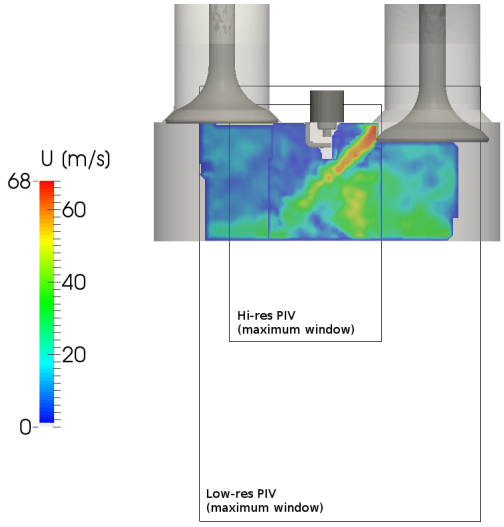


Figure 14: Location of the PIV measurement windows

mesh is straightforward, provided that sufficient care is taken about the decomposition strategy, to comply with some restrictions to make the topology modifiers work in parallel [6, 22]. Since topology modifiers change the addressing and definition of faces in the local mesh, topology must be synchronized over inter-processor boundaries. Alternatively, another way to proceed is to ensure during decomposition that pair of patches defining the topology modifiers are contained in one single sub-domain and that no inter-processor boundaries can lie on the interface itself. An extensive work has been done to allow automatic decomposition of the domains, including:

- extensions of the capabilities of the `meshTools` class, to automatically extract `cellSets` and `faceSets` from closed volumes and surfaces provided in the stereolithography (STL) format, in order to easily identify `cellSets` and `faceSets` needed by the topological modifiers in a complex mesh;
- the algorithm for automatic domain decomposition in the application `decomposePar` has been extended to handle multiple topology modifiers in the mesh, by considering the inter-dependencies between them. Since the implemented technique is based on cell addressing rather than on the mesh topology, the load balance between processors results to be improved.

Simulation strategy, mesh structure and preliminary results

The computational domain simulated includes the in-cylinder region, ports, runners, and plenums. The mesh topology is shown in Fig. 15 and 16. As described in the previous paragraphs, the dynamic mesh algorithm poses strong constraints on the mesh morphology, that has to be fully structured in the regions where layer addition/removal is applied, namely the in-cylinder region and the valve seats. Mesh topology of the inlet and outlet ducts is largely made of hexahedra, that represent a very good solution for

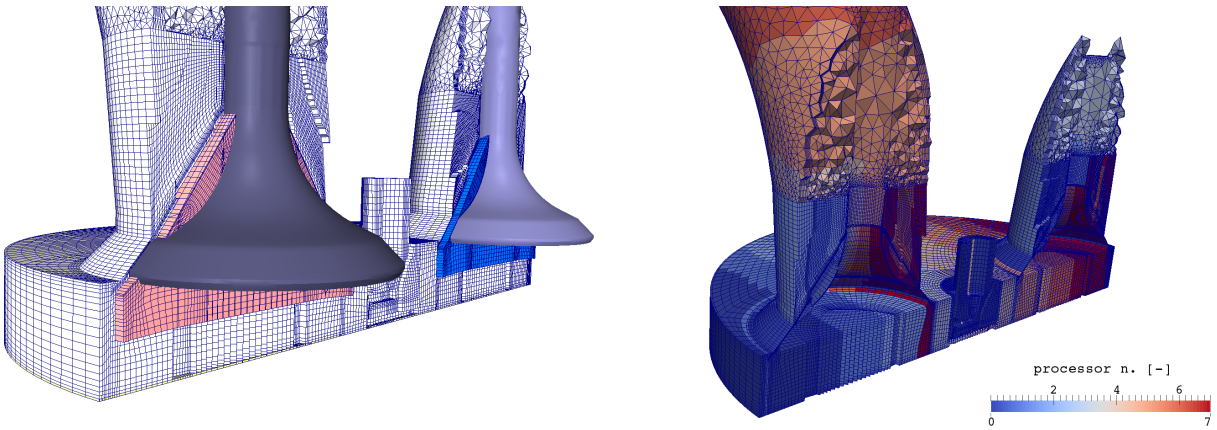


Figure 15: Left: mesh topology and definition of the sets rigidly moving during the simulation. Piston cells (yellow), exhaust (red) and intake (blue) valve cells are sliding through internal non-conformal mesh interfaces. Right: the implemented algorithm for domain decomposition is able to decompose the mesh on multiple processors preserving a good load balancing despite the constraints due to the presence of topological changes.

strongly oriented flows. Keeping a structured mesh near the valve stems is however quite complex, so hexahedra have been replaced by tetrahedra up to a few centimeters above the valve seat. In the unstructured tetrahedral region, near-wall mesh is composed by thin prisms generated by extrusion of the surface triangular mesh, to allow the use of wall functions for turbulent flow quantities. Both the valve seat and the cylinder regions are fully structured: details of the internal mesh is reported on Fig. 15. A non-conformal interface (`slidingInterface`) has been used to connect the spark plug region with the cylinder region. Significant efforts have been made to reduce the overall number of cells, without degrading the quality: a limited number of cells favors faster simulations, which is a very important aspect if several engine cycles must be simulated to gather statistical information on the resolved turbulence.

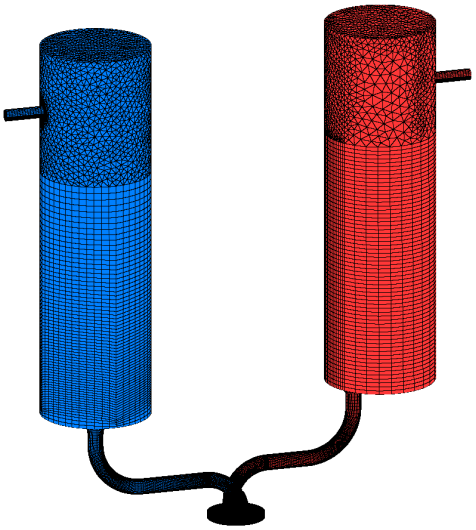
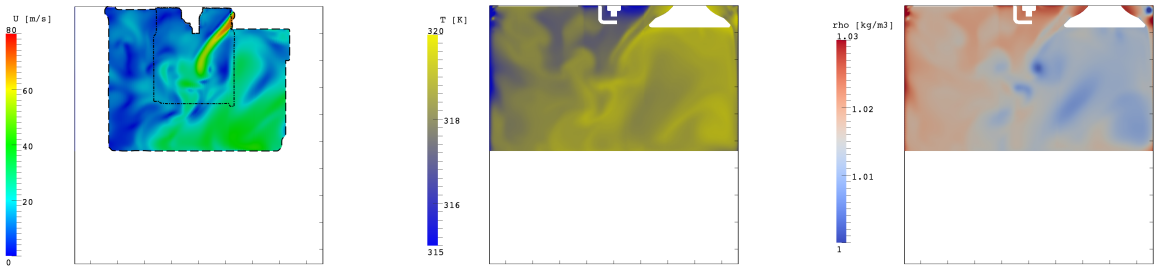


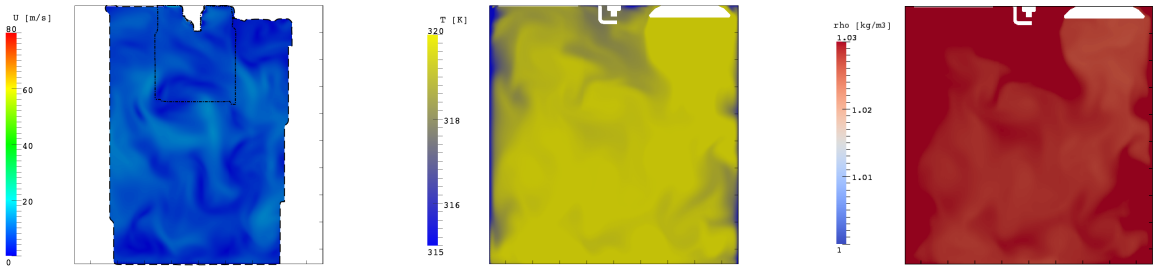
Figure 16: Computational domain for VLES of the TCC engine [21].

As shown in Fig. 15, the grid is oriented as the flow near the valve seats, in order to reduce the non-orthogonality error and to not limit the numerical schemes for the discretization of the laplacian operator [16, 17]. Unstructured tets have been used for regions near the upper part of the valve stem; plenum chambers have been mainly discretized with hexahedral cells, with the exception of the upper part, where another unstructured region has been used for simplicity. However, flow velocities are expected to be small near the plenum inlet/outlet and to have little influence on the cylinder flow. The resulting mesh is therefore hybrid. The overall number of cells is almost 1 million at Top Dead Center (TDC). The total number will increase while the piston will move towards the Bottom Dead Center (BDC) reaching a figure of about 1.8 million cells, since layers will be added on the top of the piston. The compressible solver and the mesh motion strategy just described have been used together with the Dynamic Length Resolution Model (DLRM) [**?**], a two-equation hybrid RANS/LES eddy viscosity model. As described in [**?**], the main idea behind DLRM is to retain the robust and accurate formulation of the RANS model in the near wall regions and in the free-stream region where the mesh resolution is not sufficiently high for the direct solution of the main turbulent scales and to resolve turbulent scales in the remaining cells. The solver `coldTopoEngineFoam` for time-resolved and pseudo-transient simulations of compressible turbulent flows with topological changes was used; the solver, developed by the authors, is based on the theory already described in the previous paragraphs. Second-order central differencing schemes in space for advection and diffusion were blended with linear-upwind schemes to stabilize solutions while maintaining second-order behavior [37]. The schemes applied result to be fully conservative and since the coefficients are always positive they are unconditionally bounded; also, they satisfy the transportiveness requirement for large values of local (cell) Peclet numbers. Time-marching was implicit, with the time derivative being discretized by a second-order backward approximation. The simulation time started from 450° ATDCE; the initial value of the cylinder pressure was set to the ambient value. At the time the paper is written, less than two full engine cycles of the TCC engine with DLRM turbulence model ran; for this reason, it is impossible to perform any statistical



(a) (b) (c)

Figure 17: Instantaneous flow fields calculated at the first cycle for CA = 450° ATDCE. Initial conditions in the simulation were set with flow at rest; (a) velocity, (b) temperature, (c) density. Dashed and dash-dot lines represent respectively the low and high-resolution PIV measurement windows [21].



(a) (b) (c)

Figure 18: Instantaneous flow fields calculated at the first cycle for CA = 540° ATDCE. Initial conditions in the simulation were set with flow at rest; (a) velocity, (b) temperature, (c) density. Dashed and dash-dot lines represent respectively the low and high-resolution PIV measurement windows [21].

analysis and comparison of the engine flow with the experimental data available. On the other hand, coherently to the original purpose of this paper, that was to present a novel mesh motion technique and a compressible solver, only some significant flow features will be highlighted, deferring the complete statistical analysis to a future work. In particular, some early results are reported to prove the consistency, the stability and the convergence of the solution and in order to prove that the proposed method for dynamic mesh handling ensure mass and energy conservation through the non-conformal moving interfaces and that it is robust and computationally efficient. In Fig. 17 and 18, the instantaneous velocity, temperature and density fields at 450° and 540° ATDCE are reported. As already mentioned, fields are not temporally averaged, since less than two cycles ran at the time this paper is written, and the number of samples would have been too small for any flow statistics to be meaningful. Hence, any comparison between time-resolved computed fields and time-averaged experimental fields has been omitted for it could have possibly led to misleading conclusions. At the same time, it can be noted that all fields are smooth across the non-conformal interfaces and no scattering is present: this indicates that continuity and energy are properly enforced across the topological changes.

Conclusions and future work

A mesh motion technique based on moving non-conformal interfaces has been presented, together with an improved implementation of an unsteady solver for compressible flows. Solver features include a different formulation of the equation for pressure correction and a stronger coupling between momentum and energy, that ensure an improved performance if compared to the traditional formulation. The presented solver structure is general for all the dynamic solvers for compressible reacting flows (independently by the mesh motion strategy adopted) and represents the base of the current and future applications developed by the authors. The standard implementation of `slidingInterface` has been deeply modified to allow for restore the original (split) mesh configuration after the initial coupling, without any loss of quality and it can be used to simulate partially overlapping interfaces with relative motion. The merging algorithm has been strengthened to correctly handle unstructured meshes, non-manifold patches and shared points between the sides of the non-conformal interface; thanks to these features, it is possible to stitch mesh regions in complex cases, such as unstructured grids, sharp corners, progressive refinement boxes. The `layerAdditionRemoval` class has been improved in its parallel operation, in order to deal with complex domain decompositions and avoid clashes with unstructured regions or mesh boundaries. An adaptive time-stepping procedure has been implemented in the `dynamicFvMeshClass` to ensure the mesh validity at any time and avoid geometrical inconsistencies (e.g. negative volume cells). The `attachDetach` class has been revised too, by implementing a geometric point matching algorithm that makes it more robust when included in complex meshes. Thanks to the completely redesigned class structure, the handling of mesh motion results significantly simplified: the construction, activation and interaction of the mesh modifiers with the FV mesh is handled by the new `topoManager` class; the structure of a `topoManager` object

is dynamic and makes the use of topology modifiers transparent to the developer, as new point motion strategies based on topological changes are written. The Transparent Combustion Engine (TCC) [21] has been used to show an example of operation of the mesh motion with topological changes and of the solver performance. When applied to LES of compressible flows with moving mesh, the method looks particularly interesting, since it allows SGS filter operation to be fully independent by the mesh changes: as a consequence, it will be possible to study the effectiveness of the turbulence model when applied to moving grids. Moreover, it is shown how the approach is able to preserve the initial mesh quality during the all engine cycle, since grids at different time steps differs only for layers of fully orthogonal hexahedral cells. There is a wide range of applications where this methodology can be applied: the simulation of two-stroke engines and four stroke engines with canted valves, as well for the simulation of injector nozzles, the non-uniform corrosion of solid propellants for aerospace applications and on rotating machines. The present implementation is completely based on the mesh definition of the OpenFOAM® version released by ESI-OpenCFD®.

Contact Information

Andrea Montorfano, PhD
Dipartimento di Energia, Politecnico di Milano
via Lambruschini 4a, 20156 Milano, ITALY
E-mail: andrea.montorfano@polimi.it

Federico Piscaglia, PhD
Dipartimento di Energia, Politecnico di Milano
via Lambruschini 4a, 20156 Milano, ITALY
E-mail: federico.piscaglia@polimi.it

Acknowledgments

All the simulations presented in this paper were running on Blues, high-performance computing cluster operated by the Laboratory Computing Resource Center (LCRC) at Argonne National Laboratory (ANL). Authors gratefully acknowledge ANL for the computing resources made available within the PETSc-Foam project.

References

[1] A. Montorfano, F. Piscaglia, and A. Onorati, "A LES Study on the Evolution of Turbulent Structures in Moving Engine Geometries by an Open-Source CFD Code," *SAE Technical Paper 2014-01-1147*, 2014.

[2] F. Piscaglia, A. Montorfano, and A. Onorati, "Development of a Non-Reflecting Boundary Condition for Multidimensional Nonlinear Duct Acoustic Computation," *Journal of Sound and Vibration*, vol. 332, no. 4, pp. 922–935, 2013, http://dx.doi.org/10.1016/j.jsv.2012.09.030.

[3] F. Piscaglia, A. Montorfano, and A. Onorati, "Improving the Simulation of the Acoustic Performance of Complex Silencers for ICE by a Multi-Dimensional Non-Linear Approach," *SAE Int. J. Engines*, vol. 2, no. 5, pp. 633–648, 2012.

[4] F. Piscaglia, A. Montorfano, A. Onorati, and F. Brusiani, "Boundary Conditions and SGS Models for LES of Wall-Bounded Separated Flows: An Application to Engine-Like Geometries," *Oil Gas Sci. Technol. - Rev. IFP Energies nouvelles*, vol. 69, no. 1, pp. 11–27, 2014, http://dx.doi.org/10.2516/ogst/2013143.

[5] F. Piscaglia, A. Montorfano, and A. Onorati, "Towards the LES Simulation of IC Engines with Parallel Topologically Changing Meshes," *SAE Int. J. Engines*, vol. 6, no. 2, pp. 926–940, 2013, http://dx.doi.org/10.4271/2013-01-1096.

[6] F. Piscaglia, A. Montorfano, and A. Onorati, "Development of Fully-Automatic Parallel Algorithms for Mesh Handling in the OpenFOAM-2.2.x Technology," *SAE Technical Paper 2013-24-0027*, 2013, http://dx.doi.org/10.4271/2013-24-0027.

[7] P. Sagaut, *Large-Eddy Simulation for Incompressible Flows: an Introduction*, ser. Scientific computation. Springer-Verlag, 2006.

[8] M. Brewer, L. Diachin, P. Knupp, T. Leurent, and D. Melander, "The mesquite mesh quality improvement toolkit," in *12th International Meshing Roundtable, Sandia National Laboratories report SAND 2003-3030P*, Sept. 2003.

[9] A. D. Gosman, "State of the art of multi-dimensional modeling of engine reacting flows," *Oil and Gas Science and Technology*, vol. 54, no. 2, 1999.

[10] N. Sinha, P. Cavallo, R. Lee, A. Hosangadi, D. C. Kenzakowski, S. Dash, H. Affes, and D. Chu, "Novel cfd techniques for in-cylinder flows on tetrahedral grids," *SAE Paper n. 980138*, 1998.

[11] P. Senecal, K. Richards, E. Pomraning, and T. e. a. Yang, "A new parallel cut-cell cartesian cfd code for rapid grid generation applied to in-cylinder diesel engine simulations," *SAE Technical Paper 2007-01-0159*, 2007.

[12] H. Si, J. Fuhrmann, and K. Gartner, "Boundary conforming delaunay mesh generation," *Comput. Math. Phys.*, vol. 50, pp. 38–53, 2010.

[13] K. Stapf, S. Menon, D. Schmidt, M. Rieb, and M. Sens, "Charge motion and mixture formation analysis of a disi engine based on an adaptive parallel mesh approach," *SAE Technical Paper 2014-01-1136*, 2014.

[14] S. Menon and D. P. Schmidt, "Conservative interpolation on unstructured polyhedral meshes: An extension of the supermesh approach to cell-centered finite-volume variables," *Computer Methods in Applied Mechanics and Engineering*, vol. 200, no. 4144, pp. 2797 – 2804, 2011. http://www.sciencedirect.com/science/article/pii/S0045782511001666

[15] T. Lucchini, M. Fiocco, R. Torelli, and G. D'Errico, "Automatic Mesh Generation for Full-Cycle CFD Modeling of IC Engines: Application to the TCC Test Case," *SAE Technical Paper 2014-01-1131*, 2014.

[16] H. Jasak, "Error analysis and estimation in the finite volume method with applications to fluid flows," Ph.D. dissertation, Imperial College, University of London, 1996.

[17] F. Juretić and A. D. Gosman, "Error analysis of the finite-volume method with respect to mesh type," *Numerical heat transfer, part B: fundamentals*, vol. 57, pp. 414–439, 2010.

[18] *ICEM CFD v15 User Manual*, ANSYS Inc.

[19] The OpenFOAM® Foundation. www.openfoam.com

[20] The Extend-Project, Community-driven Releases of OpenFOAM®. http://www.extend-project.de/

[21] Engine Combustion Network, "TCC-II CFD Input Dataset," 2013. http://www.sandia.gov/ecn/engines/engineFlows/TCCEngine.php

[22] F. Piscaglia, A. Montorfano, and A. Onorati, "Development of Fully-Automatic Parallel Algorithms for Mesh Handling in the OpenFOAM-2.2.x Technology," in *International Multidimensional Engine Modeling User's Group Meeting 2013, The Detroit Downtown Courtyard by Marriott Hotel, Detroit, MI (USA)*, April 14th 2013, https://imem.cray.com/2013/Meeting-2013/12-Piscaglia-Milano.pdf.

[23] A. Amsden, P. O'Rourke, and T. Butler, *KIVA-II: A Computer Program for Chemically Reactive Flows with Sprays*. LA 11560-MS, Los Alamos National Laboratory, 1989.

[24] E. L. Blades and D. L. Marcum, "A sliding interface method for unsteady unstructured flow simulations," *International Journal for Numerical Methods in Fluids*, vol. 53, no. 3, pp. 507–529, 2007. http://dx.doi.org/10.1002/fld.1296

[25] P. Farrell and J. Maddison, "Conservative interpolation between volume meshes by local galerkin projection," *Computer Methods in Applied Mechanics and Engineering*, vol. 200, no. 14, pp. 89 – 100, 2011. http://dx.doi.org/10.1016/j.cma.2010.07.015

[26] P. Farrell, M. Piggott, C. Pain, G. Gorman, and C. Wilson, "Conservative interpolation between unstructured meshes via supermesh construction," *Computer Methods in Applied Mechanics and Engineering*, vol. 198, no. 33-36, pp. 2632–2642, 2009.

[27] F. Piscaglia, A. Montorfano, and A. Onorati, "A Moving Mesh Strategy to Perform Adaptive Large Eddy Simulation of IC Engines in OpenFOAM," in *International Multidimensional Engine Modeling User's Group Meeting 2014, The Detroit Downtown Courtyard by Marriott Hotel, Detroit, MI (USA)*, April 7th 2014, https://imem.cray.com/2014/Meeting-2014/9-Piscaglia-Milano-IMEM2014.pdf.

[28] H. Jasak and Z. Tukovic, "Automatic Mesh Motion for the Unstructured Finite Volume Method," *Transactions of FAMENA*, vol. 30, no. 2, pp. 1–18, 2007.

[29] J. H. Ferziger and M. Perić, *Computational Methods for Fluid Dynamics*, 3rd ed. Springer, 2002.

[30] P. Thomas, "Geometric conservation law and its application to flow computations on moving grids." *AIAA journal*, vol. 17, no. 10, pp. 1030–1037, 1979, cited By (since 1996)511. http://www.scopus.com/inward/record.url?eid=2-s2.0-0018529022&partnerID=40&md5=e481b20ff53d71ba29cccb6ca5b09ba4

[31] H. Guillard and C. Farhat, "On the significance of the grometric conservation law for flow ccomputation on moving meshes," *Computer Methods in Applied Mechanics and Engineering*, vol. 190, pp. 1467–1482, 2000.

[32] M. Brenk, H. Bungartz, M. Mehl, I. Muntean, T. Neckel, and T. Weinzierl, "Numerical simulation of particle transport in a drift ratchet," *SIAM Journal on Scientific Computing*, vol. 30, no. 6, pp. 2777–2798, 2008. http://dx.doi.org/10.1137/070692212

[33] H. K. Versteeg and W. Malalasekera, *An Introduction to Computational Fluid Dynamics*. Prentice Hall College Div, 2nd edition, 2007.

[34] P. Abraham, D. Reuss, and V. Sick, "High-speed particle image velocimetry study of in-cylinder flows with improved dynamic range," *SAE technical Paper 2013-01-0542*, 2013. 10.4271/2013-01-0542

[35] P. Abraham, K. Liu, D. Haworth, D. Reuss, and V. Sick, "Evaluating large-eddy simulation (LES) and high-speed particle image velocimetry (PIV) with phase-invariant proper orthogonal decomposition (POD)," *Oil Gas Sci. Technol. Rev. IFP Energies nouvelles*, vol. 69, pp. 41–59, 2014. 10.2516/ogst/2013126

[36] VTK: the visualization toolkit. Kitware Inc. http://www.vtk.org/

[37] H. Weller, "Controlling the computational modes of the arbitrarily structured C grid," *Monthly Weather Review*, vol. 140, pp. 3220–3234, 2012. http://journals.ametsoc.org/doi/pdf/10.1175/MWR-D-11-00221.1