"International Multidimensional Engine Modeling User's Group Meeting at the SAE Congress"
The Detroit Downtown Courtyard by Marriott Hotel, Detroit, MI (USA), April 20^{th}, 2015

A Compressible Dynamic Solver for the Simulation of Turbulent Flows in IC Engine Geometries

F. Piscaglia, A. Montorfano, A. Onorati
Dipartimento di Energia, Politecnico di Milano, via Lambruschini 4, 20156 Milano (Italy)
S. M. Aithal
Argonne National Laboratory, Lemont, IL 60439, United States

Dynamic mesh handling previously implemented in OpenFOAM® by the authors has been extended to combine the use of topological changes with the capability to handle arbitrary moving non-conformal mesh interfaces, where face fluxes are interpolated at run-time without recomputing the mesh at each time-step. The presented strategy has been used in combination with a PIMPLE-based compressible transient solver, where a novel formulation for the reconstruction of cell centered quantities from cell face fluxes is applied as the grid changes. The proposed enhancements enable the removal of several constraints that are typical in the dynamic decomposition of multi-block grids over multiple processors, thus increasing scalability and convergence. The proposed methodology has been used in the study of in-cylinder flows in IC engines.

INTRODUCTION

This paper discusses extensions and enhancement to the methodology discussed in [1, 2, 3], about time-resolved turbulence modeling of unsteady compressible flows in dynamic grids with topological changes in OpenFOAM®. In particular, most of the modeling and development authors' work was aimed to try to limit the stringent requirements on the computational resources (hardware and computational time) placed by LES simulation, where the mesh spacing is reduced to predict more and more of the turbulent scales. Traditional LES of turbulent unsteady flows of IC engines is not always feasible and will not be in the foreseeable future even on large computers unless CFD algorithm developments are made which allow significant timestep acceleration or time parallelization (spatial decomposition is the only option currently for CFD parallelization). The timestep limit is due to the fact that both the compressible transient dynamic solver and the time-resolved turbulence modeling are limited by Courant number; even if the transient solver is potentially able to work with higher Courant numbers, time resolution is crucial to maintain phase coherence for LES, so [3]:

$$Co = \frac{\Delta x}{|\vec{u}| \cdot \Delta t} < 5 \tag{1}$$

where $\vec{u}$ is the flow velocity. From Eq. (1), it is apparent how the timestep Δt is directly proportional to the mesh spacing Δx; reducing the grid resolution while maintaining the ability to resolve most of the influential turbulence dynamics would reduce the computational time for LES simulations. This can be accomplished by using a two-pronged approach:

- improvements in physical models for turbulence, which would provide an accurate estimate of the subgrid viscosity even with coarse grids;
- improvements in the efficiency of the numerical solver and dynamic mesh handling.

With this in mind, an application to IC engine geometries will be presented to further validate recent authors' development:

- the Dynamic Length Resolution Model (DLRM), a novel hybrid RANS/LES turbulence model for the simulation of in-cylinder turbulent flows with coarse grids, whose theory is described in [3];
- an improved, fast and reliable dynamic compressible solver to handle dynamic non-conformal interfaces; this includes a novel formulation for the correction and initialization of face fluxes through topological changes, to reduce the decoupling between primary variables and velocity flux in the pressure-velocity coupling algorithm and enhanced flux correction for critical (choked) flow conditions at the valve closure;
- an extension of the dynamic mesh handling procedure presented in [1, 2], to allow topological changes to work in combination with the so called AMI (Arbitrary Mesh Interpolation) technique, already available in the standard distribution of OpenFOAM®, released by the OpenFOAM® Foundation [4].

The developed code has been validated on two different IC engine geometries of increasing complexity, whose experimental data were available from the literature: a simple piston-cylinder assembly with a stationary open valve and harmonically moving flat piston [5, 6] and a laboratory single-cylinder Transparent Combustion Chamber (TCC) engine, with moving intake and exhaust valves [7]. Furthermore, scalability tests and performance analysis on the

code have been carried out at the Argonne National Lab to identify performance bottlenecks and impediments to good load balancing.

COMPRESSIBLE DYNAMIC SOLVER IMPLEMENTATION

A newly developed compressible dynamic solver used for the simulation is `topoEngineFoam`, which is an extension of the already existing transient solver for compressible flows on dynamic meshes, with some modifications to improve convergence with multiple attaching/detaching regions and choked flows. The fundamental equations governing compressible flow inside a moving domain [8] are written as:

$$\frac{\partial}{\partial t}\int_{V(t)} \rho \,\mathrm{d}V + \int_{S(t)} \rho\,(\boldsymbol{u}-\boldsymbol{u_b})\cdot\boldsymbol{n}\;\mathrm{d}S = 0 \tag{2}$$

$$\frac{\partial}{\partial t}\int_{V(t)} \rho\boldsymbol{u} \,\mathrm{d}V + \int_{S(t)} \rho\boldsymbol{u}\,(\boldsymbol{u}-\boldsymbol{u_b})\cdot\boldsymbol{n}\;\mathrm{d}S = \int_{V(t)} \boldsymbol{f}\,\mathrm{d}V \tag{3}$$

$$\frac{\partial}{\partial t}\int_{V(t)} \rho(h+K) \,\mathrm{d}V + \int_{S(t)} \rho(h+K)\,(\boldsymbol{u}-\boldsymbol{u_b})\cdot\boldsymbol{n}\;\mathrm{d}S - \int_{V(t)} \frac{\partial p}{\partial t} - \int_{S(t)} \alpha\nabla h\cdot\boldsymbol{n}\;\mathrm{d}S = \int_{V(t)} q\;\mathrm{d}V \tag{4}$$

where $\boldsymbol{u}$ and ρ are the fluid velocity and density, h is the sensible enthalpy, K is the kinetic energy and $\boldsymbol{u}_b$ is the velocity of the boundaries of the control volume. Despite the formulation with moving boundaries looks very similar to the formulation for a static domain, solution of Eq. (2), (3) and (4) requires particular care because of the term including the relative advection velocity $\boldsymbol{u}-\boldsymbol{u}_b$. In fact, advection velocities are substituted by cell face fluxes ϕ, when discretized in a FV framework; 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 [8], 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:

$$\Delta\dot{m} = \frac{\rho\Delta V}{\Delta t} \tag{5}$$

To avoid this spurious source term (5), one must guarantee that the *Space Conservation Law* (SCL) is fulfilled [9, 10]. SCL can be regarded as a continuity equation in case of a zero fluid velocity:

$$\frac{\mathrm{d}}{dt}\int_V \mathrm{d}V - \int_S \boldsymbol{u_b}\cdot\boldsymbol{n}\,\mathrm{d}S = 0 \tag{6}$$

Discretization of Eq. (6) 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:

$$\phi_M = (\boldsymbol{u}_b\cdot\boldsymbol{n})_f S_f = \dot{V}_f \tag{7}$$

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 is done according to the selected time 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:

$$\dot{V}_f = f(T - T^o) \tag{8}$$

Since a face in OpenFOAM® is stored as a list of point IDs, and not as a list of point coordinates, Eq. (8) 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.

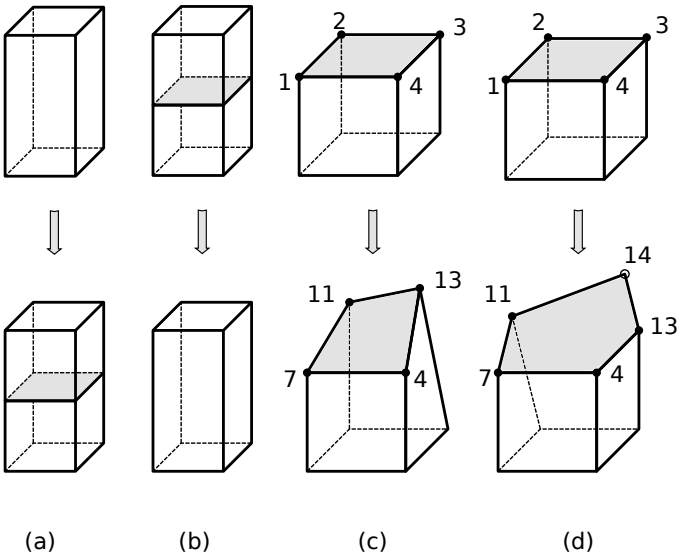


Figure 1: Different cases of topological changes. a) Face insertion; b) Face removal; c) Face does not topologically change, but its vertices get renumbered; d) Face is transformed and a new vertex is inserted.

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. 1-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. 1-b), its mesh flux does no longer exists. Continuity is thus enforced by solving a modified Poisson equation for pressure correction to apply to face fluxes, as it will be further discussed.

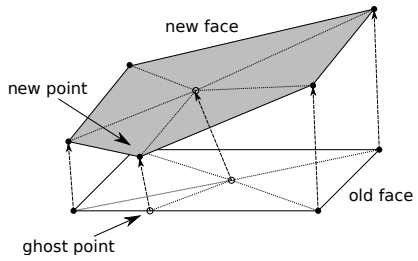


Figure 2: Handling of faces with inserted points. The new point is projected onto the counterpart edge originating a 'ghost' point, and the edge is split. Now both faces can be decomposed in the same number of triangles. In case the new face has less point than the old one, the ghost point is added on the new face instead.

If points are renumbered as a consequence of a topological change, but the owning faces do not show any substantial modification (Fig. 1-c), Eq. (8) 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. If a face maintains its definition (i.e. it still exists after the topological change), but points are added or removed (as in Fig. 1-d), it is necessary to ensure the new face to be 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 one (Fig. 2). '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 (2) and (3) on the updated mesh, old values of $\boldsymbol{u}$, p and ρ must still satisfy continuity when they are remapped onto the new grid [11, 12]: the old velocity field $\boldsymbol{u}_n^n = \boldsymbol{u}(\boldsymbol{x}^n, t^n)$ might not be compliant with the continuity equation (Eq. (2)), when it is re-sampled onto the new mesh. Therefore, a modified form of Poisson equation (Eq. (9)) need to be solved for a pressure corrector p_{corr}:

$$\nabla^2 p_{\text{corr}} + \frac{1}{\Delta t} \nabla \cdot \left[\rho\big(\boldsymbol{x}^{n+1}, t^n\big)\boldsymbol{u}\big(\boldsymbol{x}^{n+1}, t^n\big)\right] = 0 \qquad (9)$$

where $\boldsymbol{u}\big(\boldsymbol{x}^{n+1}, t^n\big)$ and $\rho\big(\boldsymbol{x}^{n+1}, t^n\big)$ denote the velocity and density fields computed at the previous timestep but remapped onto the new mesh. The differential equation (9) 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:

$$\begin{cases} \nabla^2 p_{\text{corr}} + \dfrac{1}{\Delta t} \nabla \cdot \left[\rho\big(\boldsymbol{x}^{n+1}, t^n\big)\boldsymbol{u}\big(\boldsymbol{x}^{n+1}, t^n\big)\right] = 0 \\ \dfrac{\partial p_{\text{corr}}}{\partial n} = 0 \qquad \text{on solid boundaries} \end{cases} \qquad (10)$$

During intake and exhaust strokes there is at least one open boundary, thus Eq. (10) 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. (10) 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:

$$\begin{cases} \nabla^2 p_{\text{corr}} + \dfrac{1}{\Delta t} \nabla \cdot \left[\rho\big(\boldsymbol{x}^{n+1}, t^n\big)\boldsymbol{u}\big(\boldsymbol{x}^{n+1}, t^n\big)\right] = 0 \\ \dfrac{\partial p_{\text{corr}}}{\partial n} = 0 \qquad \text{on solid walls} \\ p_{\text{corr}}(\boldsymbol{x}_0) = 0 \end{cases} \qquad (11)$$

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

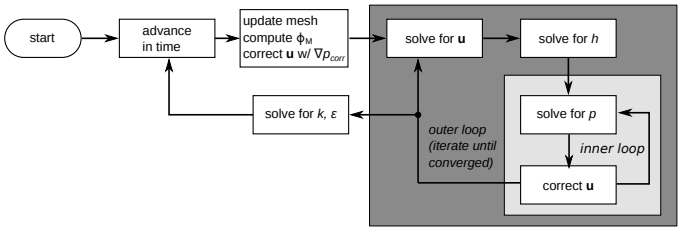
$$\boldsymbol{u}_{n+1}^n = \boldsymbol{u}_n^n - \sum \frac{1}{A_p} \nabla p_{\text{corr}} \qquad (12)$$

Eqs. (9) and (12) 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} [1]. 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.

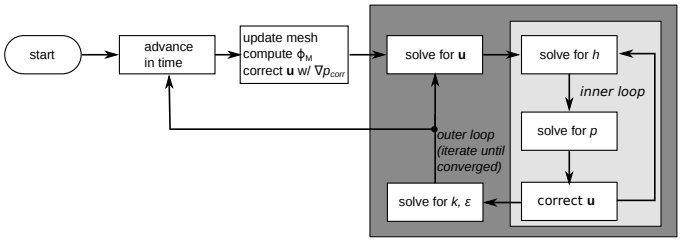
Numerical considerations for pressure-energy coupling

In OpenFOAM®, the base transient solver for compressible viscous flows is based on a merged PISO-SIMPLE algorithm (PIMPLE), which is represented in Fig. 3-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 (9) is solved and the iteration for the solution of the governing equations can start.

According to the original formulation of the PIMPLE algorithm (Fig. 3-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 ($CFL \leq 1$) [13]. 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 a topological change in the mesh occur, the energy equation is now solved together with mass conservation into the inner loop (Fig.



(a) Standard PIMPLE algorithm



(b) Modified PIMPLE algorithm

Figure 3: Original formulation (a) and the novel implementation (b) of the formulation of the PIMPLE algorithm. The compressible solver dynamically switches between the two formulations as topological changes in the mesh occur. Flux correction is calculated accordingly to the theory described in this paper.

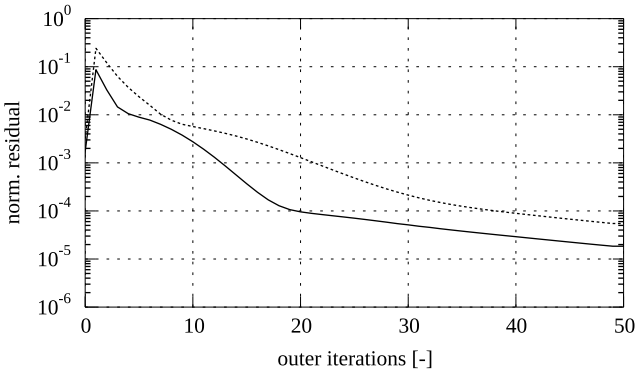


Figure 4: Numerical convergence of different implementations of the PIMPLE loop: comparison of pressure residuals between the old (- - -) and the new (—) algorithm versus the number of outer iterations within a single timestep.

3-b), to help global convergence (pressure and temperature are strongly linked in compressible flows); 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 the velocity field that might occur inside the outer loop, especially during the initial timesteps or during the attach/detach of multiple mesh regions (opening and closure of the valves, for instance). 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. 4 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 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. 4, 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.

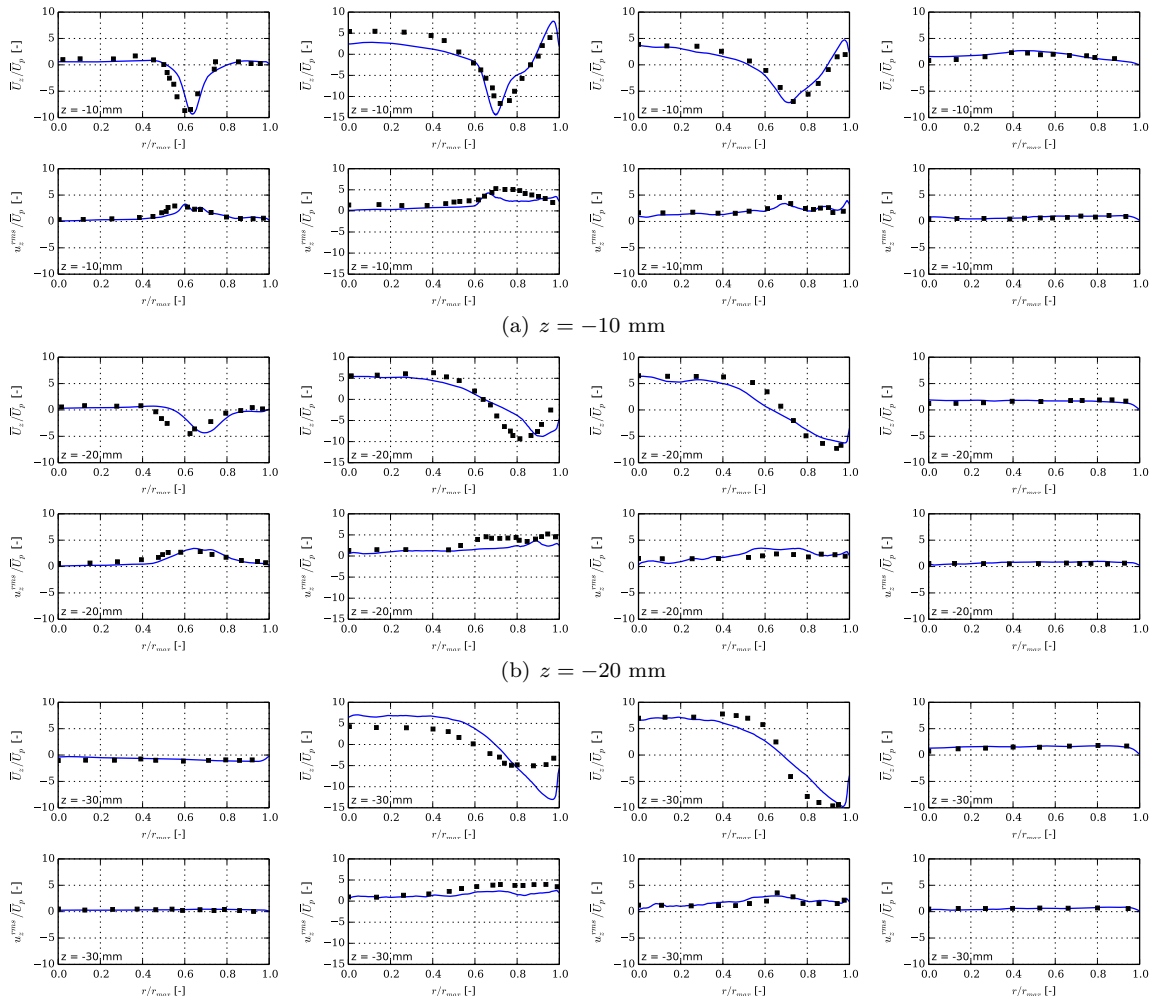
Solution procedure for the dynamic solver

The general solution procedure for the solver is as follows:

1. update timestep according to Courant number limit
2. calculate mesh motion
3. if topological changes (or remapping on different grids) occurs, construct equation for pressure correction and solve for preliminary values of the velocity fluxes;
4. calculate lagrangian transport of particles and update wall film calculation, if present;
5. solve pressure-velocity coupling according to Pressure Implicit with Splitting of Operators (PISO) algorithm:
   a) compute mass fluxes at cell faces;
   b) define and solve pressure equation (repeat multiple times for non-orthogonal mesh corrector steps);
   c) correct fluxes;
   d) correct velocities and apply BCs;
   e) repeat for number of PISO corrector steps;
6. compute turbulence and correct velocities;
7. repeat from 1 for next timestep.

RESULTS: EXAMPLE CASES

The following example cases demonstrate the capability of the simulation methods to capture the features of interest in the turbulence flow fields in IC engine simulations; it is also important to note that the implementation of the mesh motion is absolutely general and it can be applied to a wide variety of cases.
The first case studied is a piston-cylinder assembly with a stationary open valve [5], where a moving mesh strategy based on dynamic layer addition/removal is applied. In the other case (Transparent Combustion Chamber engine), both valves and piston are moving. In both cases, fluid dynamics has been solved by the novel compressible solver presented in the paper, while hybrid turbulence RANS/LES modeling has been applied. Visualizations have been done using ParaView version 4.1.0.



(a) $z = -10$ mm

(b) $z = -20$ mm

(c) $z = -30$ mm

Figure 5: Comparison between predictions (solid line, blue) and experiments (dark squares) for mean axial velocity and RMS fluctuations at selected axial locations at 36° (first column, from left), 90° (second column), 144° (third column), 270° (fourth column) ATDCE.

Piston-cylinder assembly with a stationary open valve

The 3-D geometry of the valve/piston assembly (Fig. 6), which was investigated experimentally by Morse et al. [5] using Laser Doppler Anemometry has a diameter of D =75 mm, a stroke of S =60 mm and the piston moves with a turning speed of n =200 rpm. A large plenum (not shown) upstream of the valve is employed to avoid specifying boundary conditions at the valve gap.

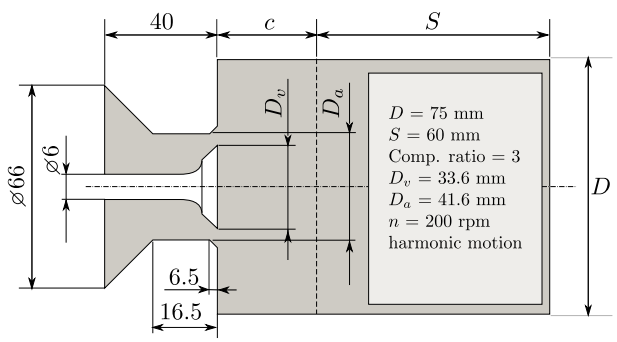


Figure 6: Geometry of the valve/piston assembly. The large upstream plenum connected to the intake section is not shown.

The valve is coaxial with respect to the cylinder and remains fixed throughout the whole engine cycle, that has a period of 360° Crank Angle (CA). The piston motion is purely sinusoidal. Due to the low piston velocity, the flow regime in the valve seat during the intake stroke is laminar. A circular jet is expected to form in the cylinder during this phase, together with primary and secondary vortex rings as a consequence of the interaction of the incoming flow with the fluid inside the cylinder. LES of the engine was carried out in [6, 14]: the WALE sgs model was used on a mesh changing dynamically by a motion strategy based on a point-stretching concept; results from LES were validated with simulations from a spectral element solver [15, 16, 17]. Thanks to the hybrid RANS/LES turbulence model applied, a very coarse mesh, whose resolution was ranging from 700K cells at the BDC to 150K cells at the TDC has been used for simulations (in [6, 14], the grid had about 4 million cells).

In Fig.5, mean and RMS fluctuations of the axial velocity are compared at selected axial distances z from the cylinder head. Similarly to what has been found in [14] and [6], even with an hybrid RANS/LES model jet penetration at 90° CA ATDCE looks quite difficult to capture. On the other hand, results at all angles are satisfying and

are extremely similar to the results obtained by LES, with a computational effort has been reduced of over one order of magnitude.

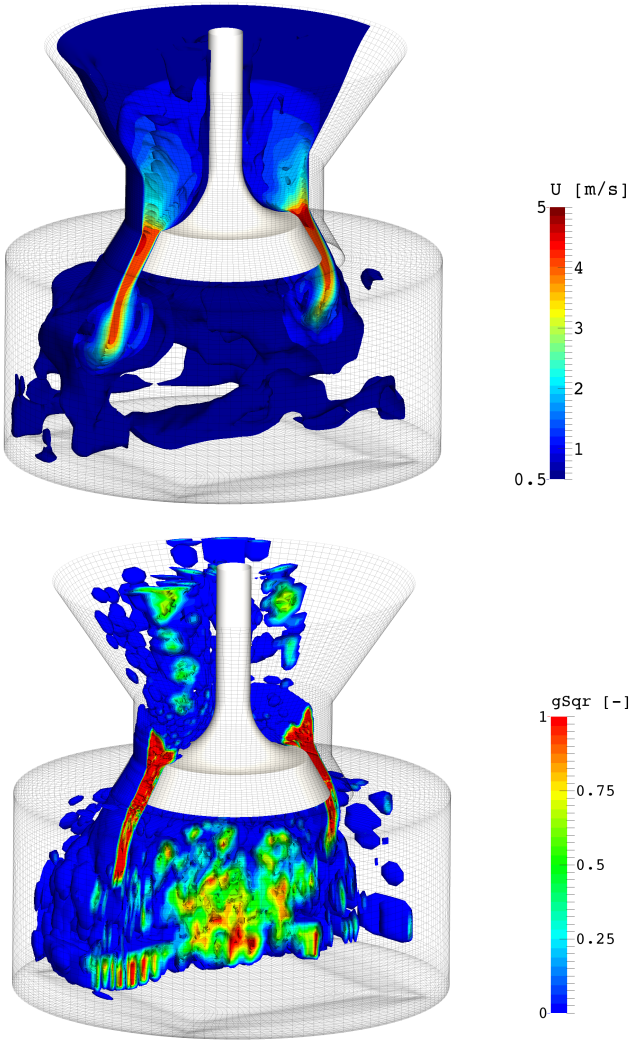


Figure 7: Valve/piston assembly, angle=30° CA deg ATDCE; a) contour plot of the flow velocity b) contour plot of the g^2 function (see [3] for more detail). Where $g^2 = 1$, DLRM applies pure RANS to solve turbulence. With $g^2 = 0$ implicit LES is used. The large upstream plenum connected to the intake section is not shown.

The Transparent Combustion Chamber Engine Case

The Transparent Combustion Chamber (TCC) engine is an optical engine that was set up at the University of Michigan [7] 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. 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 [18]. The TCC engine has been adopted as a test case in [1, 2, 19] to test authors' developments for dynamic mesh handling in OpenFOAM®: the `slidingInterface` algorithm has been used in [2] both to stitch the spark-plug and the in-cylinder region through cylindrical non-conformal interfaces and to dynamically employ valve motion through the in-cylinder region.

Bore	92 mm
Stroke	86 mm
Connecting rod length	234.95 mm
TDC clearance height	9.50 mm
Geometric compression ratio	10
Engine speed	1300 rpm

Table 1: Geometrical features of the TCC engine [7].

Extension of the dynamic mesh class

With respect to previous authors' work, a further development of the dynamic mesh handling has been adopted to simulate the TCC engine. As described in [1, 2], the `slidingInterface` coupling procedure generates a seamless joint between two non-conformal surfaces, by taking advantage of the capability of OpenFOAM® to handle polyhedral cells in the mesh: a significant advantage of `slidingInterface` is that no particular numerical technique is required to solve the equations across the interface; at the same time, as in any key-grid or target-mesh approach with local changes in topology, it requires a global transfer of solution variables prior to a solution restart and, most of all, mesh connectivity must be updated anytime the sliding interface operates, with a consequent lack of performance. The standard release of OpenFOAM® comes along with an interesting feature called AMI (Arbitrary Mesh Interface) interpolation: it consists of a calculation of a vertex-based solution using a bounded Galerkin projection approach [20] to interpolate fluid-dynamic quantities over a "supermesh", a virtual surface mesh made by triangles, whose vertices are the points coming from the intersection of the two sides of the non-conformal mesh interface. This method is slightly slower if compared to a target-mesh approach, requires a more complicated implementation procedure and a more expensive interpolation algorithm, so it does not represent the optimal solution to handle non-conformal interfaces in static mesh regions. On the other hand, the computational cost of the interpolation of the fluid-dynamic quantities over the supermesh is less expensive than updating the mesh topology anytime the `slidingInterface` operates (i.e. anytime the valve moves, for instance).

To summarize, AMI is very performing to handle dynamic full overlapping non-conformal interfaces (as it will be further shown for the valve motion of the TCC engine), since the computational overhead of topological changes of the sliding interface would be order of magnitude higher than AMI interpolation. On the other hand, `slidingInterface` looks convenient with:

- *static non-conformal interfaces:* the more complicated interpolation algorithm of AMI might represent an additional cost on the total simulation time, whereas the sliding interface has no overhead since it uses standard cell-to-cell interpolation. A validation work is underway to assess the numerical properties of both strategies (AMI vs. sliding interface) when applied to

static meshes.

- *dynamic interface with partial overlap*, when there is not full overlap between non-conformal interfaces (as in the simulation of two-stroke engines [21]). In this case, a sliding interface approach is preferred; at the time the paper is written, a stable numerical strategy to apply a fictitious wall boundary over "non-overlapping" faces and to block outgoing fluxes is not available; this aspect is particularly challenging by a numerical point of view since it might cause instabilities and it is the topic of the ongoing research.

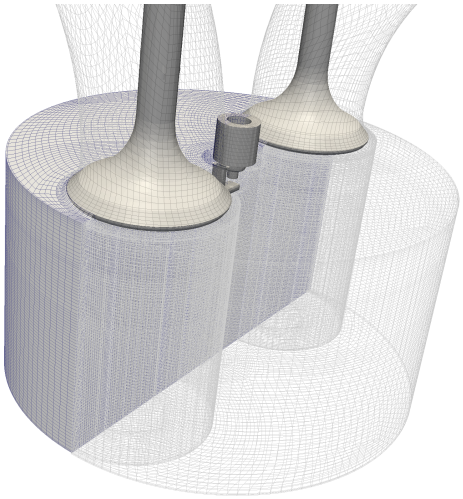


Figure 8: The 1M cell mesh used to simulate the TCC engine [2, 19], including the plenums connected to the intake and exhaust ducts.

In the TCC engine mesh presented here, AMI interpolation has been applied in place of `slidingInterface` to model the curtain area of the valves and the cylindrical surface between the cylinder and the volume below the valve (see Fig. 8), while `slidingInterface` has been used to couple the cylindrical surface between the spark plug and the in-cylinder region [2]. Authors' extensions to the dynamic mesh class allows for full flexibility in selecting the preferred strategy on a per-case basis, by simply modifying case dictionaries.

TCC Engine: validation

Code and model validation has been carried out on the basis of two main quantities:

- the instantaneous volume, to verify that dynamic addition and removal of cell layers was consistent;
- the instantaneous in-cylinder pressure trace, to verify the operation of the numerical solver and of the submodels used.

Experimental pressure and in-cylinder temperature have been set as initial conditions at the intake valve closure. In the simulation, `layerAdditionRemoval` has been applied to the third layer of cells above the piston; these cells were removed during compression when their thickness was lower than a threshold value defined by the user (0.5 mm in the example); conversely, single layers of cells were added

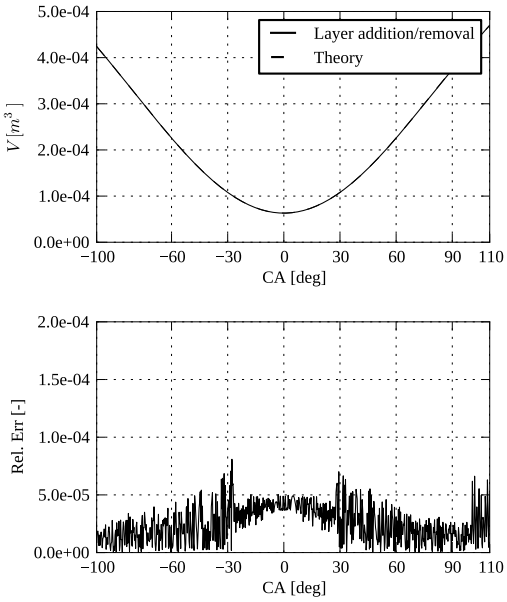


Figure 9: a) experimental in-cylinder volume during the compression and expansion phase with closed valves (calculated vs predicted); 2) instantaneous absolute error.

during expansion, as the cell thickness was higher than 1 mm.

In-cylinder volume of the simulated domain was compared with the theoretical value:

$$V = V_c + s_p(\theta) \cdot A_c \tag{13}$$

where V_c is the volume of the combustion chamber, A_c is the piston face area and s_p is the instantaneous piston displacement as a function of the crank angle θ. The cycle fraction simulated was between the IVC and EVO: the rationale for this choice was to verify possible errors on one single source only (addition and removal of cell layers on the piston surface).

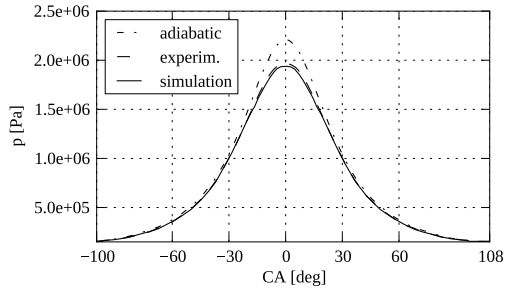


Figure 10: Average in-cylinder pressure of the motored engine from IVC (≈ 100° BTDC) to EVO (≈ 108° ATDC)

However, the generality of results on volume consistency is not lost: if no significant error on the volume computation is introduced by layer addition/removal on the piston during pure compression (IVC to EVO), it should happen the same with respect to layer addition/removal near the valves. As shown in Fig. 9, the volume history of the in-cylinder volume is perfectly replicated and the error in volume calculation is of the order of 0.01%. This small error is due to the round-off error in the calculation of the cell volume of hexahedral cells, rather than to the dynamic

addition/removal of cell layers. As expected, also the cylinder pressure trace (Fig. 10) has been correctly predicted, both during the compression and the expansion phase.

SOLVER PERFORMANCE

To be of industrial interest, a CFD solver must be scalable, in order to enable large-scale simulations in a relatively short time. Studies about the scalability of the standard solvers available in OpenFOAM® on traditional Linux clusters has been extensively proven for static grids [22]; in most of the studies, it has been shown that inter-processor communications represent a bottleneck for scalability only under a certain amount of cells per processor (CPP): for a standard PISO solver the current limit is about 10^4 CPP.

The performance of the dynamic mesh solver proposed in this paper has been analyzed by carrying out scalability tests on the TCC engine, that has been used as reference case. The computational mesh setup previously presented has been decomposed over different numbers of subdomains using the automatic graph partitioning library SCOTCH, constrained by all decomposition restrictions imposed by topological changes [2]. Two main tests have been carried out at Argonne National Lab:

- a scalability test for the mesh motion only. A complete engine cycle (720 CA degrees) has been simulated, starting from the TDC;
- a scalability of the complete solver (mesh motion + fluid dynamics) on 1 CA degree, starting from the BDC, when the total amount of cells is the highest.

Along with traditional scalability metrics, the overhead introduced by the topology solver has been evaluated, and cell quality indexes (skewness and non-orthogonality) have been monitored throughout the complete engine cycle. Test were operated on the Linux HPC cluster "Blues", consisting of 310 nodes; each node was featured by two Sandy Bridge 2.6 GHz Pentium Xeon processors (16 cores per node) and 64 GB of memory. Nodes are interconnected by Infiniband Qlogic QDR. Inter-processor communication was handled by openmpi-1.6.5 compiled with gnu gcc 4.7.2. Walltime, as well as load balancing and mesh quality, has been monitored for each test.

In Fig. 11 scalability of the solver `topoEngineFoam`, including the solution of the flow field, is reported. Findings are consistent with the general scalability properties of OpenFOAM®: the highest reduction in wall-time is found for 9000 CPP; after that, then the walltime increases because of increased communications among domains.

The same scalability test has been carried out for the mesh motion algorithm only and results are displayed in Fig. 12. The point motion solver exhibits a linear reduction of walltime with a monotone trend up to 2000 CPP, that is well beyond the scalability limit found when the fluid dynamic solution is calculated; the reason is that topological changes are local to a subdomain, so no information exchange is in theory needed between neighboring sub-domains. However, it can be observed that the slope of the curve in Fig. 12 is definitely lower than the theoretical slope required for a linear scaling. This can be explained by observing that some point synchronization is anyway

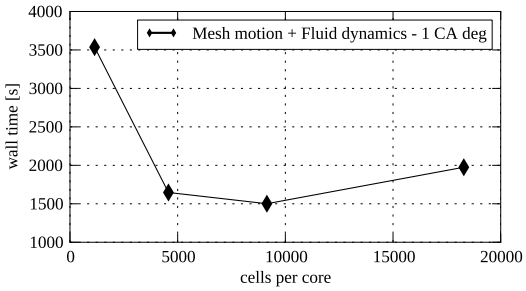


Figure 11: a) Scalability tests on the TCC engine geometry for the complete solver (mesh motion + fluid dynamics). The engine time simulated was 1 CA deg.

needed when mesh motion is operated in parallel, therefore some degree of inter-processor communication cannot be avoided. However, the amount of data exchanged between neighboring subdomains is not going to represent a bottleneck.

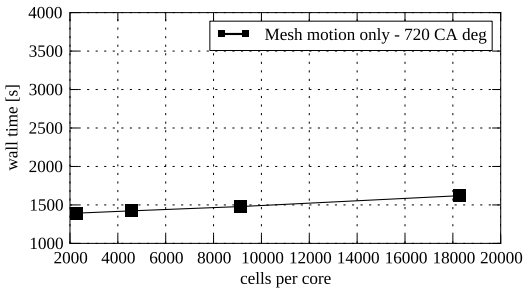


Figure 12: Scalability test for point motion solver only. Simulation time is 720 CA degrees.

Also, Fig. 12 would lead to the conclusion that the mesh motion algorithm is not scalable; on the other hand, a deeper look at the results evidences that the overall walltime for the mesh motion during an engine cycle (720 CA degrees) takes about the same time required by the fluid-dynamic solver to compute the fluid solution of 1 CA degree: the overhead introduced by the mesh motion on the simulation load is absolutely negligible if compared to the fluid mechanics solver, so its performance is almost the best that can be achieved; the limit to the overall duration of the simulation is therefore given by the constraints on the time-step.

Load balancing between processors can be estimated by considering the number of cells owned by each processor and how they vary as the mesh motion is calculated. Raw data representing the number of CPP versus crank angle is reported in Fig. 13. As it can be expected, the number of cells increases while the piston moves from the TDC towards the BDC, because of the dynamic addition of cell layers on the piston surface. Moving from the TDC to the BDC, the number of cells decreases again, because of the removal of cells on the piston (mainly) and on the valves. Processor load imbalance has been estimated by the ratio between the number of cells of the largest subdomain and the smallest one. This ratio is displayed in Fig. 13-b. It can be noted that the maximum imbalance decreases as the piston approaches BDC, as a consequence of the cell layer insertion into the cylinder region. A more balanced decomposition near the BDC, when the overall

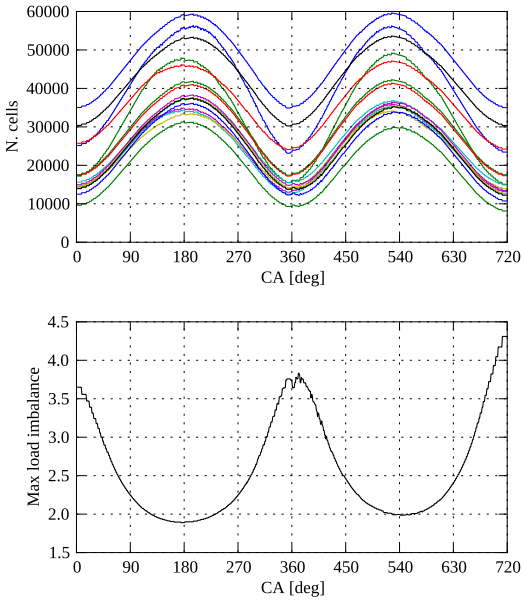


Figure 13: a)Number of cells per each subdomain as a function of crank angle; b) Maximum decomposition imbalance as a function of crank angle.

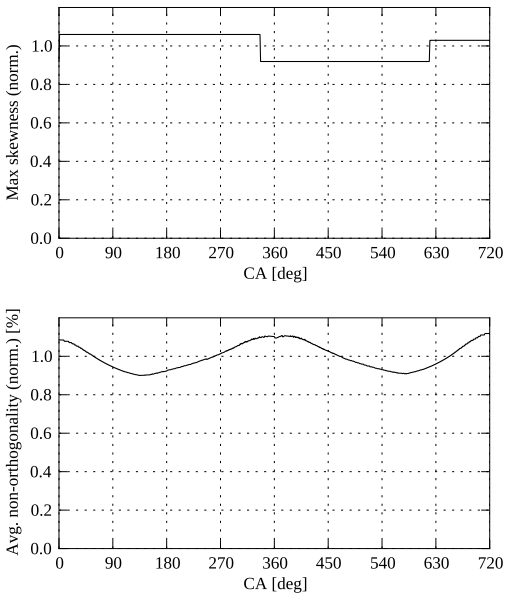


Figure 14: a) Maximum skewness vs crank angle; b) average non-orthogonality vs. crank angle. Values have been normalized by their mean value calculated over the thermodynamic cycle.

cell count is higher is a favorable circumstance because the computing effort is supposed to be highest. Finally, maximum skewness and average non-orthogonality have been reported in Fig. 14; all values have been normalized by their average value calculated over the thermodynamic cycle. As evidenced by the results, the mesh motion strategy adopted allows to maintain almost constant skewness and non-orthogonality at every crank angle position. Observed oscillations in the values (about 10%) are related to the deformation of the cell layers during the timesteps before the addition/removal of cell layers is applied.

CONCLUSIONS

The mesh motion technique described in [1, 2] based on moving non-conformal interfaces has been extended to allow for a combined use of topology modifiers (namely `slidingInterface`, `attach/detach` of boundaries and `layerAdditionRemoval`) and of the Arbitrary Mesh Interface algorithm, in order to improve the performance of the mesh motion solver. Together with an improved implementation of an unsteady solver for compressible flows and the use of a novel hybrid URANS/LES model [3], the developed framework seems able to capture the main features of turbulent in-cylinder flows and to be accurate to predict the fluid dynamic quantities. Scalability test carried out at the Argonne National Lab show a good performance of the code when used on HPC clusters on a wide number of cores. Two engine geometries have been used as an example of operation. 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 [2]: as a consequence, it will be possible to study the effectiveness of the turbulence model when applied to moving grids. There is a wide range of applications for this methodology: 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 rotating machines. The present implementation is completely based on the mesh definition of the OpenFOAM® versions released by the OpenFOAM® Foundation, where the information about the mesh topology is stored only for the current timestep of calculation and additional information about the topological changes need to be stored separately.

ACKNOWLEDGMENTS

Authors would like to thank:

- the LCRC (Laboratory Computing Resource Center), Argonne National Lab, for making available the computing resources through the HPC cluster Blues within the `PETSc-FOAM` project;

- Proff. V. Sick, D. Reuss, X. Yang and T. Kuo, who made available detailed data and geometry of the TCC engine; the TCC engine work has been funded by General Motors through the General Motors University of Michigan Automotive Cooperative Research Laboratory, Engine Systems Division.

.

REFERENCES

1. 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.

2. A. Montorfano, F. Piscaglia, and A. Onorati, "An Extension of the Dynamic Mesh Handling with Topological Changes for LES of ICE in OpenFOAM," *SAE Technical Paper 2015-01-0384*, 2015. http://dx.doi.org/10.4271/2015-01-0384.

3. F. Piscaglia, A. Montorfano, and A. Onorati, "A Scale Adaptive Filtering Technique for Turbulence Modeling of Unsteady Flows in IC Engines," *SAE Int. J. Engines, Paper n. 2015-01-0395*, 2015. http://dx.doi.org/10.4271/2015-01-0395.

4. OpenFOAM Foundation, *OpenFOAM User's Guide*, 2015.

5. A. P. Morse, J. H. Whitelaw, and M. Yanneskis, "Turbulent flow measurements by laser-doppler anemometry in motored piston-cylinder assemblies.," *Journal of Fluids Engineering*, vol. 101, pp. 208–216, 1979.

6. A. Montorfano, F. Piscaglia, M. Schmitt, C. E. Frouzakis, A. G. Tomboulides, K. Boulouchos, and A. Onorati, "Comparison of Direct and Large Eddy Simulation of the Turbulent Flow in a Valve/Piston Assembly," in *10th International ERCOFTAC Symposium on Engineering Turbulence Modelling and Measurements*, 2014.

7. Engine Combustion Network, "TCC-II CFD Input Dataset," 2013.

8. J. H. Ferziger and M. Perić, *Computational Methods for Fluid Dynamics*. Springer, 3rd edition ed., 2002.

9. 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.

10. 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.

11. H. Jasak, *Error analysis and estimation in the Finite Volume method with applications to fluid flows*. PhD thesis, Imperial College, University of London, 1996.

12. 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.

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

14. 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.

15. A. Montorfano, F. Piscaglia, M. Schmitt, Y. M. Wright, C. E. Frouzakis, A. G. Tomboulides, K. Boulouchos, and A. Onorati, "Comparison of direct and large eddy simulations of the turbulent flow in a valve/piston assembly," *Flow, Turbulence and Combustion*, vol. 95, no. 2, pp. 461–480, 2015. http://dx.doi.org/10.1007/s10494-015-9620-6.

16. M. Schmitt, C. Frouzakis, A. Tomboulides, Y. Wright, and K. Boulouchos, "Direct numerical simulation of multiple cycles in a valve/piston assembly," *Physics of Fluids*, vol. 26, no. 3, 2014.

17. P. F. Fischer, J. W. Lottes, and S. G. Kerkemeier, "nek5000 web page," 2008. http://nek5000.mcs.anl.gov.

18. 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.

19. F. Piscaglia, "Unsteady Engine Analysis with Moving Mesh in OpenFOAM." The Connector, Pointwise Inc., http://www.pointwise.com/theconnector/July-2014/Unsteady-Engine-Analysis.shtml, 2014.

20. 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.

21. 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.

22. M. Culpo, "Current bottlenecks in the scalability of OpenFOAM on massively parallel cluster." PRACE white paper.

23. 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.

24. 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.

25. 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.

26. 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.

27. 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.