The 8th European Congress on Computational Methods in Applied Sciences and Engineering
ECCOMAS Congress 2022
5–9 June 2022, Oslo, Norway

DEVELOPMENT OF A FRAMEWORK FOR INTERNAL COMBUSTION ENGINE SIMULATIONS IN OPENFOAM

CLEMENS GÖSSNITZER, STEFAN POSCH AND GERHARD PIRKER

Large Engines Competence Center GmbH
Inffeldgasse 19, 8010 Graz, Austria
e-mail: {clemens.goessnitzer,stefan.posch,gerhard.pirker}@lec.tugraz.at, www.lec.at

Key words: Internal Combustion Engines, OpenFOAM, Large Engines

Abstract. High-accuracy simulations of internal combustion engines (ICE) allow deep insight into the physical processes of the different phases of the engine cycle: gas exchange, mixture formation, compression, combustion and emission formation. The commercial solvers for ICE simulations provide a full package which covers these areas. However, the user of such software is unable to look into the source code, making it impossible to implement new models or investigate possible implementation errors in the code, and costs arise due to licensing requirements for commercial solvers.

Although the open source framework OpenFOAM already includes multiple classes and two solvers dedicated to internal combustion engine simulations, there is no way to move engine valves and piston simultaneously with its standard tools. Thus, this paper presents a new engine library for ICE simulations written for OpenFOAM.

The new framework is capable of simulating a complete fired engine cycle. The piston and the valves are moved simultaneously. To address large deformations in the mesh, a methodology to avoid insufficient mesh quality was developed. Ignition and combustion is modeled with standard tools from OpenFOAM. To validate the method, the simulation results for the averaged in-cylinder quantities pressure, temperature and mass are compared with experimental data.

1 INTRODUCTION

The simulation of internal combustion engines (ICE) is a challenging task for computational fluid dynamics (CFD) simulations since many different physical phenomena have to be considered: turbulence, trans-sonic flow conditions, wall heat transfer, species diffusion, ignition, turbulent combustion amongst others. The computational domain changes in time due to the movement of the piston and the valves, and the topology also does not remain constant since valves close and open, disconnecting parts of the domain.

Interest is growing in using OpenFOAM for engine simulations, and pioneering work has been done at Politecnico di Milano. The CFD group from this university developed a library LibICE for OpenFOAM, which implements automated meshing, boundary conditions for engine simulations, combustion and spray models [1]. More recently, researchers at TU Darmstadt have developed TFMotion, a similar library based upon the work of Politecnico di Milano [2]. However, none of these solutions for engine simulations in OpenFOAM are available to the public.

The original OpenFOAM framework already includes a class to handle the time transformation applied in engine CFD, and stub classes for piston and valves. However, in the standard package several things are missing or not provided for a successful full-cycle engine simulation:

- There is no way to simultaneously move the piston and the valves with the standard engine classes.
- It is not possible to move the valves with the valve class.
- Using a non-flat piston surface does not appear to be supported.
- Piston movement is assumed to be in the z direction only.
- Specification of multiple moving piston patches is not supported, and the names for the piston, liner and cylinder head patches are hard-coded.
- The (incomplete) implementation of valve movement is based on arbitrary mesh interfaces (AMI) which have conservation issues in the current implementation.

It was decided to address these issues and limitations in the current engine implementation in OpenFOAM. The work was split into two parts: (I) actual implementation of all the required features into the OpenFOAM source code, which is presented in the second chapter, and (II) automation of the pre- and post-processing steps, i.e. mesh generation, setup of initial conditions and boundary conditions, job submission and data collection, using a Python script, which is presented in the third chapter.

In the fourth chapter, the application of the developed methods to a large natural gas engine is presented. The framework is capable of performing full-cycle engine simulations.

2 IMPLEMENTATION INTO OPENFOAM

The implementation described below builds upon previously existing stub classes for the piston and the valves and the recently added `fvMeshMover` framework from the OpenFOAM Foundation version. It was decided to use an explicit mesh moving algorithm for the piston movement, i.e., the displacement of each cell inside the cylinder is determined by an analytical function. The standard velocity displacement solver from OpenFOAM was used to move the valves.

2.1 Piston movement

An explicit mesh movement was chosen instead of a diffusion-based Laplacian solver. This allows for higher deformations of the grid without any generation of cells with negative volume or high skew. Such an approach is also much faster than solving a partial differential equation for the mesh motion.

For the piston movement, a buffer region has to be specified where cells are uniformly deformed. In the ideal case, the buffer region is a perfect cylinder. All cells below the buffer region, i.e., the piston bowl, are uniformly translated. All cells above the buffer region do not move due to the piston movement—note that they can potentially move due to the movement

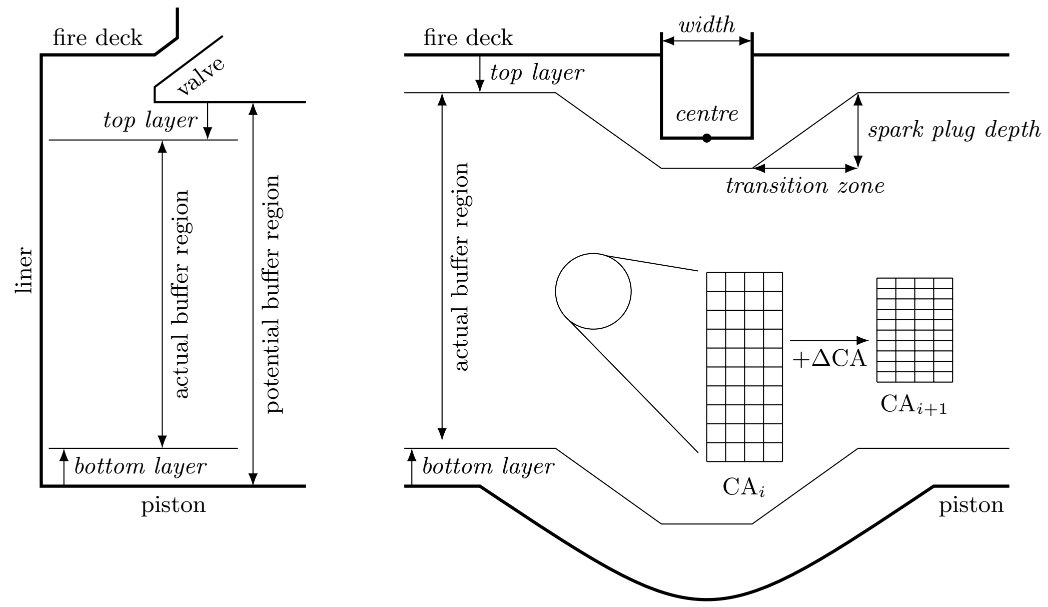

Figure 1: Illustration of the actual buffer region used in the piston movement strategy. The user must specify all parameters (*in italic*). Inside the buffer region, the mesh is explicitly scaled in the direction of the piston movement, as shown here from CA$_i$ to CA$_{i+1}$.

of the valves. Normally, the buffer region is the liner region excluding a user-specified distance from the fire deck and the piston. However, special care has to be taken to exclude the region of the valves and potentially a spark plug sleeve that extends into the piston bowl. Figure 1 gives an overview of all user-specified parameters for the piston movement. Though there may be multiple piston boundaries, only one boundary for the liner is allowed, and it has to be perfectly aligned with the direction of the piston movement.

2.2 Valve movement

The movement of the valve is implemented by solving a Laplacian equation for the cell motion:

$$\nabla \cdot (D\,\nabla m) = 0$$

where m is the cell motion and D is the matrix of diffusion coefficients. The boundary conditions are the velocity for the valve patches. Multiple options are available for D in OpenFOAM. The best results were achieved by setting D non-uniformly to the inverse volume of each cell. Note that the Laplacian-based motion solver is the main limitation concerning the quality of the deformed mesh. Furthermore, the valve lift varies from zero, i.e., the valve is closed and the corresponding port is disconnected from the cylinder, to the range of centimetres, which is impossible to capture with just one single mesh. Thus, the mesh must be updated regularly to ensure stable and accurate simulation. There may be multiple valve boundaries which move with the valve speed. Yet similarly to the liner, there may only be one valve stem boundary.

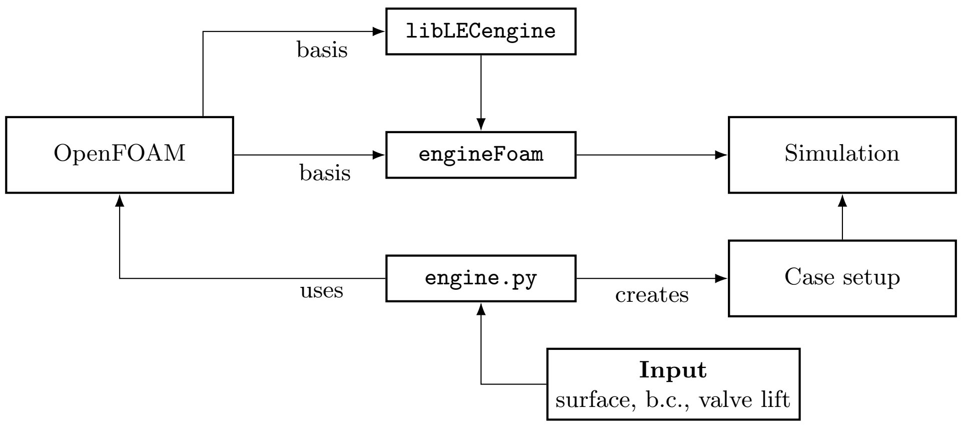

Figure 2: Illustration of the workflow automation.

2.3 Solver

The solver developed for engine simulations is based upon the existing `reactingFoam`. For engine CFD applications, it is necessary to record cylinder-averaged quantities of pressure and temperature and the total trapped mass in the cylinder to enable comparisons with measurement and 0D/1D simulations. Thus, the solver was extended to provide additional output for those quantities. For this purpose, the user must create three different cell sets that describe the cylinder, intake and exhaust regions.

3 WORKFLOW AUTOMATION

Since a full-cycle engine simulation requires dozens of individual meshes, an automated workflow to set up the simulations is required. This automated workflow is based on a Python implementation of the case setup, meshing and post-processing, cf. Figure 2. The input for the script consists of the surface of the engine geometry, where the piston is at top dead center and the valves are at minimum valve lift, inlet and outlet boundary conditions (time-resolved, cross-section averaged pressure and temperature at defined points in the intake and exhaust system, i.e., inlet and outlet of the CFD model), chosen turbulence model and valve lift curves. The output of the script are simulation setups (one per mesh), which are ready to be submitted to a high performance cluster (HPC) queueing system.

3.1 Meshing

Meshing was done with the OpenFOAM tool `snappyHexMesh`. After the surface is moved to the correct valve and piston positions, embeddings for the valve region are automatically applied depending on the valve lift and valve position. Once the meshing is done, the mesh is moved to its end position in a dry run to ensure that the mesh quality is sufficiently high during the actual simulation.

Generally, the duration of one mesh is limited mainly by the movement of the valves. The implementation of the piston movement is robust and requires much less mesh-to-mesh mapping than the movement of the valve. The most difficult is when the valves move ‘back’, i.e., towards the valve seat. Then the mesh quality deteriorates quickly and a new mesh is required. To overcome this difficulty, a special meshing strategy was implemented: If the valve moves towards the valve seat from CA_i to CA_{i+1}, the initial mesh is created at the end time, i.e. at CA_{i+1}, and subsequently moved 'backwards' to its position at the starting time of the simulation CA_i.

To find CA_{i+1}, starting from CA_i, i.e., to determine whether a new mesh is required, the relative and absolute change of valve lift and piston position are examined. The concrete values for the allowed changes are based on manual testing. In future versions of the meshing workflow, the mesh quality check should decide when a new mesh is necessary and reduce the manual tuning that is required with the current approach.

3.2 Case management

An independent simulation setup for each mesh is created by `engine.py`. Each of these simulations has to be submitted to the queuing system of an HPC cluster and run consecutively, since one simulation builds upon the preceding simulation, i.e., fields are mapped from one simulation to the next.

4 APPLICATION TO A LARGE NATURAL GAS ENGINE

The new implementation and workflow were tested on a large natural gas engine with a displaced volume of around 3 L. The engine is operated based on a lean-burn spark-ignited open-chamber natural gas combustion concept. Main purpose of this simulation was to test the implementation and automation.

Since the engine uses a central mixing device far upstream of the intake port, mixture formation was not modelled. The geometry was divided into three parts (intake, exhaust and cylinder) and four simulation stages (exhaust, valve overlap, intake and high pressure). While the two ports are deactivated according to the stage of the simulation, the cylinder region is always active. During the exhaust and intake stages, the intake port or exhaust port are deactivated, respectively. During valve overlap, all three regions are active. During high-pressure cycle, only the cylinder region is active.

4.1 Simulation settings

The dynamic k equation LES model from OpenFOAM applied for turbulence. The grid filter was set to the simple type. Bounded second-order schemes were used for discretisation of the gradients, divergence and Laplacian operator and the implicit Euler schemes for time discretisation. Time-varying total pressure and thermodynamic temperature were prescribed at the inlet and outlet and the mixture composition was prescribed according to the measurement data. The time step was constant piecewise to keep the Courant number below 5. The time step varied from 5×10^{-4} CA to 1×10^{-2} CA. Since the flow is critical after exhaust valve open, the transonic option was set in the `fvSolution` dictionary.

Pressure-velocity coupling was achieved using the PIMPLE algorithm (merged PISO-SIMPLE) with one outer corrector, two inner correctors and one non-orthogonal corrector. The momentum predictor was active and the consistent option for PIMPLE was set. The linear equation solvers had an absolute tolerance of 10^{-6} and a relative tolerance of 0.

Combustion was simulated using the finite rate chemistry model with no modeling of tur-

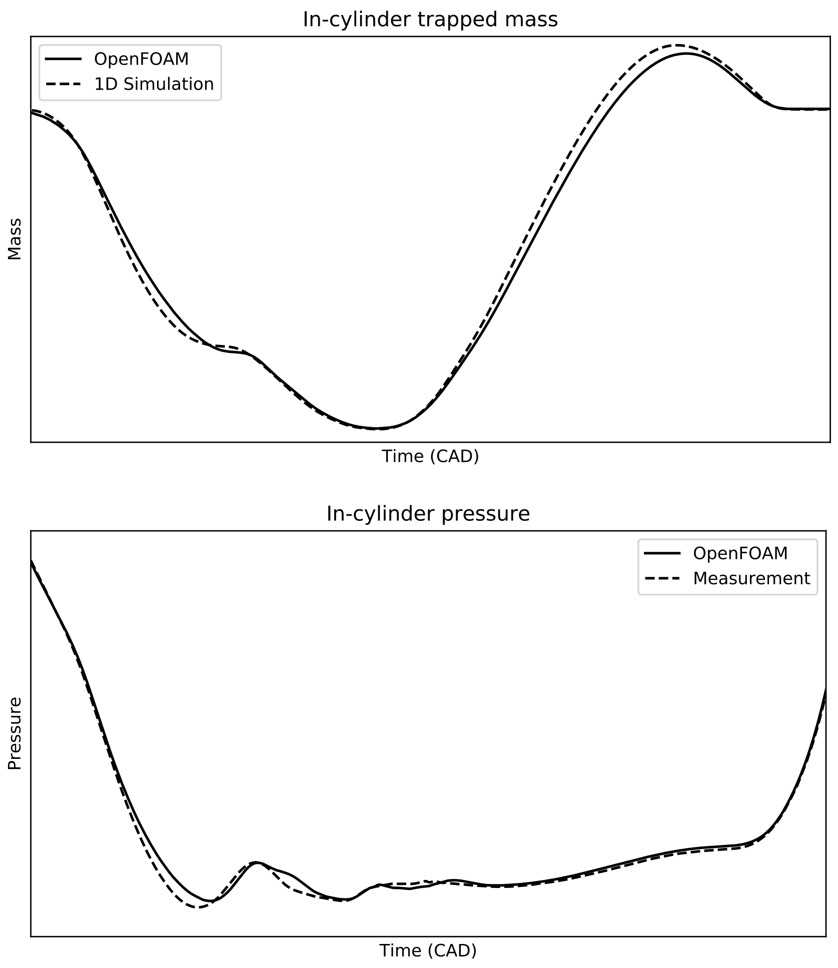

Figure 3: Results for the in-cylinder mass (top) and pressure (bottom) during the low-pressure cycle.

bulence chemistry interaction. Natural gas combustion was modeled with the GRI Mech 3.0 reaction mechanism. It consists of 53 species and more than 350 reactions [3].

4.2 Results

The results for the low-pressure cycle, i.e., the time between exhaust-valve open and intake valve closing (IVC), shows good agreement with measurement and 0D simulations, cf. Figure 3. The pressure characteristics are well captured by the 3D CFD simulations. Furthermore, the trapped mass after IVC agrees nicely with the 1D simulation, which is also key point to obtaining the correct amount of energy in the cylinder.

Figure 4 shows the in-cylinder pressure during combustion, i.e., the time after ignition. As expected, the combustion is too fast since the reaction mechanism is known to over predict the flame speed [4].

However, it has been shown that the developed framework—implementation and automation—

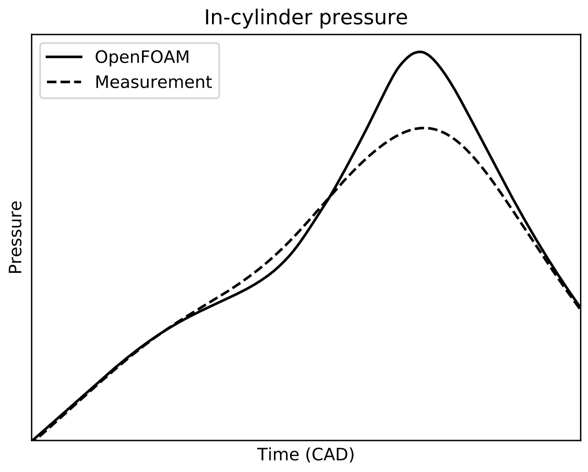

Figure 4: Results for the in-cylinder pressure during combustion.

is capable of running full-cycle engine simulations of a large, natural gas engine. Further improvements to the combustion part include switching to a better mechanism, or using a completely different combustion model.

5 CONCLUSIONS & OUTLOOK

This paper presented a new implementation of a full-cycle internal combustion engine simulation framework in OpenFOAM. It consists of a library for valve and piston movement, a CFD solver and an automation script which takes care of pre- and post-processing.

One of the main challenges in engine simulations is the generation and movement of the mesh. To avoid too frequent changes in mesh topology, a special meshing strategy was applied in the case of a valve moving towards its seat. Then meshing is done at the end of the corresponding time step, and the mesh is moved to the starting position at the starting time step.

In the future, more powerful meshing and mesh movement strategies will be required. Run time mesh-to-mesh mapping was recently implemented into the development branch of OpenFOAM. A new implementation of non-conformal interfaces called Non-Conformal Coupled (NCC) was introduced into OpenFOAM and promises to be a second-order accurate, conservative re-implementation of the AMI feature. It will improve mesh movement by adding NCC around the mesh and perform simple explicit stretching and shrinking of the cells. Since these features are very new, they do not appear in the workflow presented in this paper.

Another interesting mesh movement approach is layer addition/removal, which allows the piston to be moved with minimal mesh movement, i.e., only a single layer of cells. However, the current implementation in OpenFOAM has issues related to mass conservation, does not work well in parallel and is difficult to set up. In the future, better implementation of the layer addition/removal feature—if coupled with NCC—may make it possible to reduce the number of different meshes to just four: one for each of the four phases of the engine cycle.

Combustion modeling also requires more attention. To illustrate the capability of the framework presented here, combustion was modeled with finite rate chemistry, and no explicit turbulence-chemistry interaction was considered. With finite rate chemistry, the predictive ability of the combustion model highly depends on the chosen reaction mechanism. In the future, a simpler combustion model that considers turbulence-chemistry interaction will be used.

REFERENCES

[1] Lucchini, Tommaso, D'Errico, Gianluca, Ettorre, Daniele, Spagnoli, Emma and Ferrari, Giancarlo: Lib-ICE: A C++ object-oriented library for internal combustion engine simulations – spray and combustion modeling. *Talk at the 5th OpenFOAM Workshop*, Chalmers, Gothenburg, Sweden, 21-24th June 2010.

[2] Pati, Andrea, Paredi, Frederica Ferraro and Hasse, Christian, CFD Modelling of Fuel-Air mixture formation in a GDI engine using OpenFOAM. *Talk at the 8th OpenFOAM Conference*, online, 13-15th October 2020.

[3] Gregory P. Smith, David M. Golden, Michael Frenklach, Nigel W. Moriarty, Boris Eiteneer, Mikhail Goldenberg, C. Thomas Bowman, Ronald K. Hanson, Soonho Song, William C. Gardiner, Jr., Vitali V. Lissianski, and Zhiwei Qin, The GRI Mech 3.0 Reaction Mechanism, available at http://www.me.berkeley.edu/gri_mech/

[4] Yiqing Wang, Ashkan Movaghar, Ziyu Wang, Zefang Liu, Wenting Sun, Fokion N. Egolfopoulos and Zheng Chen, Laminar flame speeds of methane/air mixtures at engine conditions: Performance of different kinetic models and power-law correlations. *Combust. Flame* (2020) **218**:101–108.